Treffen der ASIM Fachgruppe
Simulation Technischer Systeme

4. ASIM Workshop Wismar
Modellierung, Regelung und Simulation in Automotive und Prozessautomation

29.-30. Mai 2008

Wismar

Zusammenfassung der Vorträge

ARGESIM Report AR 33
ASIM-Mitteilungen AM 117

ARGESIM Report

ISBN print: 978-3901608-33-9, 2008
ISBN ebook: 978-3-901608-77-3, 2021; DOI: 10.11128/arep.33


asim


ARGESIM

ARGESIM Report AR 33

Christina Deatcu (Hrsg.)
Peter Dünow (Hrsg.)
Thorsten Pawletta (Hrsg.)
Sven Pawletta (Hrsg.)

ASIM – Mitteilungen AM 117

Treffen der ASIM Fachgruppe
Simulation technischer Systeme

4. ASIM Workshop Wismar
Modellierung, Regelung
und Simulation
in Automotive und Prozessautomation

29.-30. Mai 2008
Wismar

organisiert durch:


Hochschule Wismar
Universität Rostock
IAV GmbH Gifhorn
ASIM FG Simulation technischer Systeme

Zusammenfassung der Vorträge

ARGESIM - Verlag, Vienna
ISBN print: 978-3901608-33-9
ISBN ebook: 978-3-901608-77-3, 2021
DOI: 10.11128/arep.33

ARGESIM Reports

Published by **ARGESIM** and **ASIM**, Arbeitsgemeinschaft Simulation,
Fachausschuss 4.5 der GI

Series Editor:
Felix Breitenecker (ARGESIM / ASIM)
Div. Simulation, Vienna University of Technology
Wiedner Hauptstrasse 8 - 10, A - 1040 Vienna
Tel: +43-1-58801-11452, Fax: +43-1-58801-11499
Email: Felix.Breitenecker@tuwien.ac.at

ARGESIM
ASIM – Mitteilungen AM 117

Titel: Treffen der ASIM Fachgruppe
Simulation technischer Systeme

4. ASIM Workshop Wismar
Modellierung, Regelung und Simulation in Automotive und Prozessautomation

Herausgeber: Christina Deatcu
Peter Dünow
Thorsten Pawletta
Sven Pawletta

Email: christina.deatcu@hs-wismar.de

ISBN print: 978-3901608-33-9, 2008
ISBN ebook: 978-3-901608-77-3, 2021; DOI: 10.11128/arep.33

ARGE Simulation News (ARGESIM)
c/o F. Breitenecker, Div. Simulation, Vienna Univ. of Technology
Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria
Tel.: +43-1-58801-11452, Fax: +43-1-58801-42098
Email: info@argesim.org; WWW: http://www.argesim.org

Druck:
Hochschule Wismar

Vorwort

Der vorliegende Band enthält Zusammenfassungen der Beiträge des 4. ASIM-Workshops "Modellierung, Regelung und Simulation in Automotive und Prozessautomation" sowie des jährlichen Treffens der ASIM-Fachgruppe "Simulation technischer Systeme" am 29. und 30. Mai 2008 an der Hochschule Wismar.

Die beiden Veranstaltungen zeigen inhaltlich eine starke Verwandtschaft. Deshalb wurden die Beiträge nicht den einzelnen Veranstaltungen zugeordnet, sondern nach passenden Themengruppen zusammengestellt. Aus der Menge der eingereichten Beiträge wurden die Schwerpunkte "Sprachstandards", "Werkzeuge und Anwendungen", "Automotive", "Spezielle Anwendungen und thermische Systeme", "Modellbasierte Regelungen", "Medizintechnik", "Mechatronische Systeme", "Statistische Analyse" sowie "Modellbasierte Funktionsentwicklung" abgeleitet.

Die Herausgeber bedanken sich bei allen, die zur Vorbereitung und Durchführung des Workshops beigetragen haben, insbesondere bei den Herren W. Nietschke und M. Schultalbers von der IAV-GmbH für die materielle und organisatorische Förderung der Veranstaltung sowie bei Herrn Professor Breitenecker und Herrn Dr. Mammen für die Unterstützung von Seiten der ASIM.

Wismar, den 08.05.2008

P. Dünow, C. Deatcu, T. Pawletta, S. Pawletta

Inhalt

J. Kirscher, M. Lenz, D. Metzner, G. Pelz
Real-life parameter extraction for automotive electric drive applications….....……...1

E. Hessel
Die Modellbibliotheken des VDA in der Anwendung…….....3

F. Breitenecker, F. Judex, G. Zauner, N. Popper
DAEs, Modelica-Notation and Variable Structures in Simulators
–a Comparative Study.....……..7

M. Bockholt, J. Köhler, W. Tegethoff
Gradientenfreie dynamische Optimierung von Modelica-Modellen…….....27

F. Gottelt, J. Nocke, E. Hassel
Modellierung und Simulation eines konventionellen Steinkohleblocks mit Modelica.....35

O. Naujocks, R. Schütt
Funktionsblockentwicklung zur effizienten Codegenerierung
für Regelungen in Windenergieanlagen…….…….....43

E. Bröcker, S. Dominka, M. Merz
Systematische Qualitätssicherung bei verteilt erstellten Simulationsmodellen.....53

F. Judex, B. Hametner, F. Breitenecker, G. Zauner
E-Learning mit MATLAB in Mathematik Grundvorlesungen.....61

J. Schmidgal, T. Winsel, H. J. Theuerkauf
Iterative evidente Interpolation und ihre Anwendung zur Nachbildung von Abgasemissionen.....67

S. Dusemund, H. Schneider, K.-J. Langeheinecke, A. Horn
Simulation Abgasnachbehandlung: Ein modulares System zur Berechnung von Abgasanlagen.....77

F. Heßeler, R. Beck, D. Abel, J. Piewek, C. Nöthen, H.-G. Nitzke
Simulation eines Dieselluftpfades mit zwei Abgasrückführstrecken in Dymola.....87

R. Gehring, C. Wanke, H.-G. Herzog
Bewertung von Spannungsschwankungen im 12 V Kfz-Energiebordnetz mittels Simulation.....97

D. Jung, M. Speckert, K. Dressler
Modellierung und Simulation eines neuen Prüfstandkonzepts zur Achserprobung.....105

S. X. Ding, A. de Moll, S. Schneider, G. Nau, N. Weinhold, M. Schultalbers
Beobachtergestützte Implementierung Euler-diskretisierter nichtlinearer Systeme….....107

J. P. Blath
A Multivariable Speed Controller for a Hybrid Electric Vehicle.....……115

B. Alt, F. Svaricek
Experimentelle Identifikation und Sliding Mode Regelung einer elektronischen Drosselklappe.....127

G. Yang, T. Jeinsch, S.X. Ding, N. Weinhold, M. Schultalbers
Performanzindizes-basierte Selbsteinstellung der Reglerparameter im Motormanagementsystem.....137

C. Fritzsche, H.-P. Dünow, B. Lampe
Simulationsbasierte Entwicklung von Motorsteuerungsfunktionen
am Beispiel der Momentenregelung…….....145

C. Gehsat, T. Bertram, R. Trapp
Modellierung thermischer Systeme für die Hardware-in-the-Loop Simulation
am Beispiel eines Fahrzeugkabinenmodells.....155

T. Maschkio
Analyse von Wärmetransport- und Kondensationsprozessen in Kfz-Scheinwerfern unter Verwendung von CFD.....163

O. Zirn, R. Montavon
Gekoppelte Simulation von FE- und Mehrkörpermodellen für Werkzeugmaschinen.....165

I. Kirchner, E. Gerlach, S. Oberthür, K. Zimmermann
Erstellung eines MKS – Modells zur Untersuchung der Dynami einer Nanopositionier- und Messmaschine (NPMM).....171

G. Schneckenreither, F. Breitenecker, G. Zauner, A. Ponholzer
Mathematische Modelle zur Unterstützung der Prostatakrebsdiagnostik für Mediziner.....173

A. Brunberg, D. Abel, R. Autschbach
Ein objektorientiertes Modell des Herz-Kreislauf-Systems mit besonderer Betrachtung körpereigener Regelkreise.....181

O. Simanski, A. Schubert, Ch.N. Nguyen
Anwendung des Guyton-Modells bei der Regelung der tiefen Hypotension.....189

S. Heinke, M. Walter, S. Leonhardt, A. Brunberg
Objektorientierte Modellbildung des partiellen CO2-Gehalts bei beatmeten Patienten in Dymola.....191

B. Hametner, J.Kropf, X. Descovich, S. Wassertheurer, F. Breitenecker
Implementierung eines eindimensionalen Blutflussmodells mittels Finiter Elemente Methode.....193

A. Junghanns, J. Mauss, M. Tatar
Gesamtsystemanalyse komplexer mechatronischer Systeme.....201

Y. Chen, O. Lenord, D. Schramm
Modeling Discontinuous Elements in Mechatronic Systems by Using Regularized and Idealized Approaches.....207

P. Schneider, A. Stork, T. Bruder, E. Moeller
FunctionalDMU – ein Ansatz für die kooperative, virtuelle Analyse mechatronischer Produkte.....215

T. Schlage, J. Lunze
Ferndiagnose dynamischer technischer Systeme.....227

J. Bretschneider, A. Wilde
Entwurfsumgebung für hallbasierte 3D-Positionssensorsysteme…….....235

M. Sylvester
Statistische Analyse Verfahren für integrierte Analog und Mixed-Signal Schaltungen….....241

A. Burblies, N. Reichert, M. Busse
Einsatz von Leveled-Noise-Verfahren beim Computer Aided Robust Design.....243

D. Dammers, D. Schollän, L. M. Voßkämper
Yield prediction in an automotive MEMS application with help of statistical analysis packages SAE J2748.....251

J. Wegener, H. Sthamer
Automatischer Back-to-Back-Test von Modellen und Code.....257

J. Ladisch, W. Gottschalk, O. Magnor, M. Schultalbers
Modellierung des CAI-Brennverfahrens und die Anforderungen an die Motorsteuerung eines Versuchsfahrzeuges.....265

C. Joachim, H.-C. Reuss, J. Horwath
Simulation der Auswirkungen einer gesteuerten Motormomentenvorgabe auf das Schwingungsverhalten einer Sattelzugmaschine.....275

(*kursiv* gedruckte Vorträge sind ausschließlich mit Präsentationsfolien bzw. Abstract im Tagungsband enthalten)

Real-life parameter extraction for automotive electric drive applications

Jérôme Kirscher, Michael Lenz, Dieter Metzner, Georg Pelz,
Infineon Technologies AG
jerome.kirscher@infineon.com

Abstract

Apart from the creation of an electro-mechanical model's equation set, also the related parameters have to be identified. These parameters can be derived from measurements performed on the real device. Based on the example of a DC-Motor, modeled in VHDL-AMS, this paper will show how to extract these parameters.

1 Introduction

The ability of VHDL-AMS to describe heterogeneous systems allows its use for modeling elector-mechanical systems. The equations describing such systems can be found in textbooks. However some effort has to be put in determining the values of the physical parameters involved in these equations. Using a DC-Motor model as an example, a way to extract these parameters will be presented in this paper. The model will be validated by comparing the simulation results with the measurements.

2 DC-motor parametrization

The equations used to model the dependencies of the electro-mechanical quantities are given below.

$$U = K_T \cdot \varpi + R \cdot I + L \cdot d(I)/dt \quad (1)$$

$$T = -K_T \cdot I + D \cdot \varpi + J \cdot d(\varpi)/dt \quad (2)$$

U represents the voltage [V], I the current [A], ω the angular velocity [rad/s] and T the torque [N].

The strategy developed to extract the real-life parameters is presented in what follows.

The winding inductance (L) is derived from the current measured when the motor starts rotating (fig1), without any load. The slope when the current starts rising from 0A gives $d(I)/dt$. At this very point $\varpi = 0$ and $I = 0$. U is read out and L is calculated, using the eq. (1).

The moment of inertia (J) is derived from the same measurement. The angular acceleration $d(\varpi)/dt$ is approximated from the measurement right at the start of the motor, knowing that 4 current commutations correspond to one rotor rotation. The average current during this first rotation is read out. As there is no load, $T = 0$ and at the beginning, the viscous damping loss ($D \cdot \varpi$) is insignificant, so the eq. (2) becomes $J = K_T \cdot I/(d(\varpi)/dt)$. Once Kt is extracted, J is known.

The winding resistance and the torque coefficient extractions are shown in more detail in the section below.

Winding resistance (R) extraction:

The rotor is blocked, when the steady state is reached, the eq. (1) becomes:

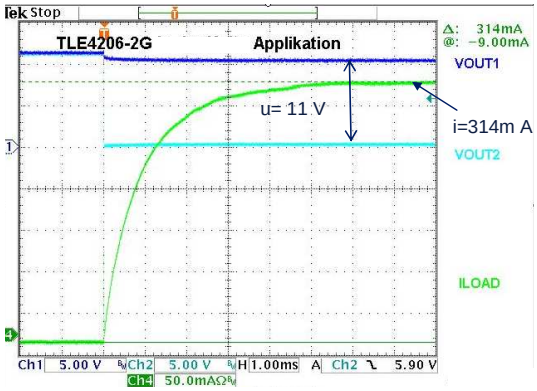
$$U = K_T \times 0.0 + R \times I + L \times 0.0$$

i.e.

$$R = U/I$$

So from the oscillogram below:

$$R = 35\Omega$$



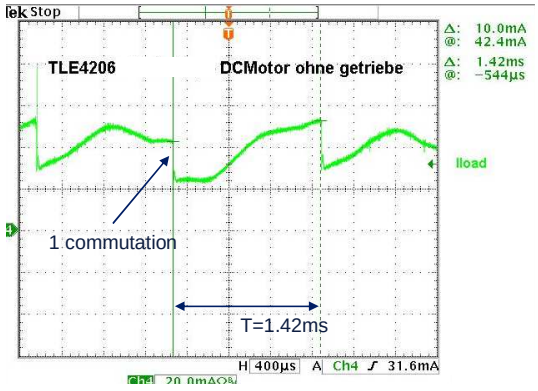
Torque coefficient (Kt) extraction:

The motor rotates, without load. One rotation induces 4 current commutations. The time between two commutations is measured and the angular velocity is derived:

$\varpi = 2\pi/(4 \times T)$, 1 rotation $\equiv 2\pi$ [rad]

From the eq. (1), Kt can be calculated:

$$K_T = 9.95\, mV/rads^{-1}$$



3 Simulation versus Measurement

The motor simulation and measurement results comparison shows very good correlation.

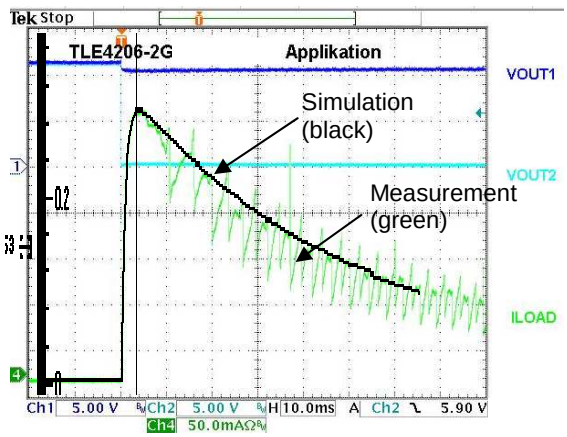


Fig 1: inrush current meas. & simulation

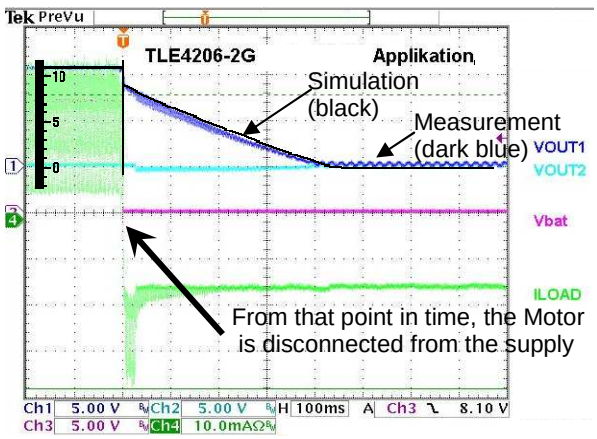


Fig 2: coasting phase meas. & simulation

Die Modellbibliotheken des VDA in der Anwendung

Ewald Hessel, Hella KGaA Hueck & Co.
Ewald.Hessel@hella.com

Zusammenfassung

Es werden zunächst die Modellbibliotheken vorgestellt, die im AK30 (VDA/FAT) in der neutralen Simulationssprache VHDL-AMS erstellt wurden. Diese sind nach Richtlinien des AK30 erstellt und für verschiedene Tools direkt verwendbar. Die praktische Anwendung der in den Bibliotheken enthaltenen Modelle wird an 2 Beispielen gezeigt. Im Ausblick werden die nächsten Aufgaben des AK30 beschrieben.

1 Modellbibliotheken des VDA

1.1 Aufgaben und Ziele des Ak30

Der AK30 ist ein Arbeitskreis der Forschungsgemeinschaft Automobiltechnik (FAT) im VDA. Wesentliche Ziele des Arbeitskreises sind die Förderung der Simulation gemischter Systeme und des Modellaustausches über die werkzeugneutrale Simulationssprache VHDL-AMS (IEEE 1076.1) [1].

1.2 Modellbibliotheken des AK30

Als Voraussetzung für diese Ziele hat der AK30 Richtlinien zur Modellbildung sowie einheitliche Strukturen für Modellbibliotheken festgelegt. Damit soll die leichte Kompilierung dieser Bibliotheken in unterschiedliche Werkzeuge ermöglicht werden. Alle Modelle in diesen Bibliotheken sind offen und frei anwendbar [2]. Das Modellverhalten beinhaltet jedoch lediglich die Grundfunktion und stellt kein schützenswertes Spezialwissens dar. Eine eigene Erweiterung der Verhaltensbeschreibung ist leicht möglich.
Folgende Bibliotheken sind bisher über das Internet verfügbar :

- SPICE2VHD : Umsetzung der Grundmodelle von SPICE 3F5 (Level1)
- SPICE2VHD_DEVICES : Parametrisierte Komponenten
- FUNDAMENTALS_VDA : Grundmodelle für verschiedene Domänen
- AUTOMOTIVE_VDA : Komponenten im Kraftfahrzeug
- MEGMA : zeitdiskrete Grundmodelle für Steuerung und Regelung
- MODELICA_TRANSLATIONAL : Umsetzung offener MODELICA-Modelle
- MODELICA_ROTATIONAL : Umsetzung offener MODELICA-Modelle

2 Modell der elektrischen Servolenkung (EPS)

2.1 Funktionsbeschreibung

Das EPS-System (Electronic Power Steering) stellt ein typisches mechatronisches System dar. In diesem Beispiel handelt es sich um ein Doppelspindelsystem. Über einen Drehmomentsensor wird das Drehmoment am Lenkrad gemessen und daraus ein Signal für die Ansteuerung eines Elektromotors abgeleitet, der die Lenkbewegung des Fahrers unterstützt [3].

2.2 Modellstruktur und Simulationsergebnisse

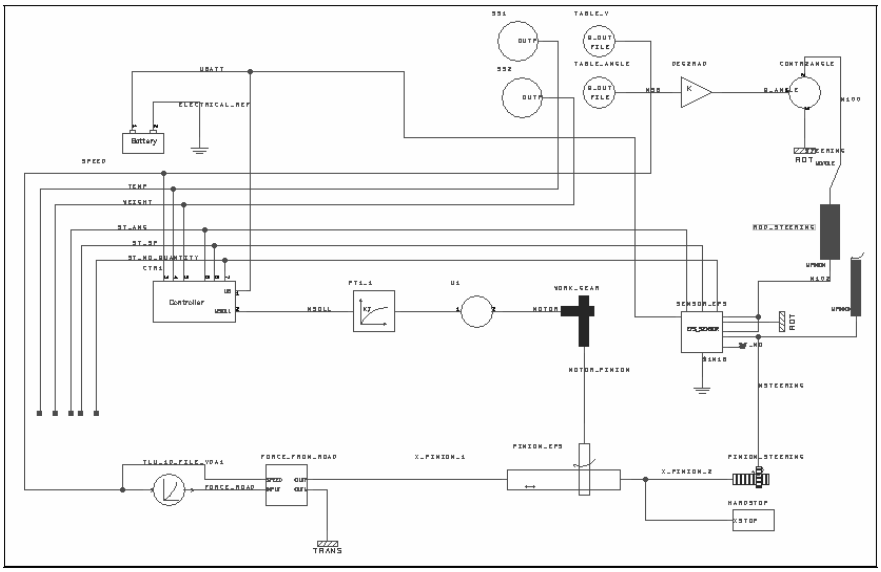


Abbildung 1: Schaltbild EPS (SystemVision/ Mentor Graphics)

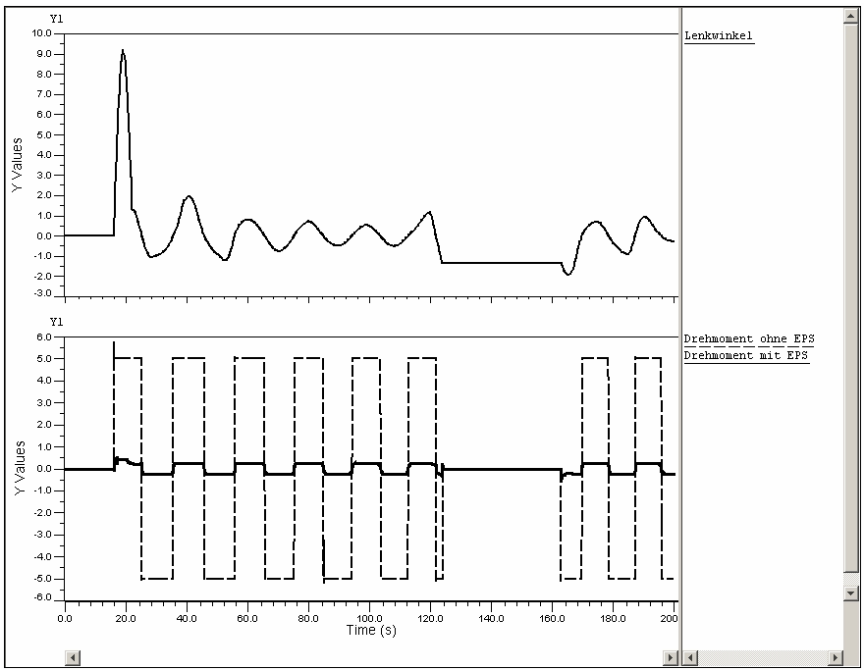


Abbildung 2: Drehmoment am Lenkrad mit und ohne EPS-Einfluss

3 Modell eines Leuchtweitenregulierers (LWR)

3.1 Funktionsbeschreibung

Beim Leuchtweitenregulierer (LWR) [3] wird über einen Schrittmotor und das Lampengestänge die Neigung des Scheinwerfers eingestellt. Im Steueralgorithmus können die Fahrzeugneigung, die Fahrgeschwindigkeit sowie Signale von Lichtschalter, Gaspedal und Bremspedal verwendet werden. Da eine Messung der Schweinwerferposition nicht erfolgt, wird die aktuelle Position von der Ansteuerung des Schrittmotors (Controller mit Steuerungsalgorithmus) errechnet und dient damit als neuer Ist-Wert für den Algorithmus.

3.2 Modellstruktur und Simulationsergebnisse

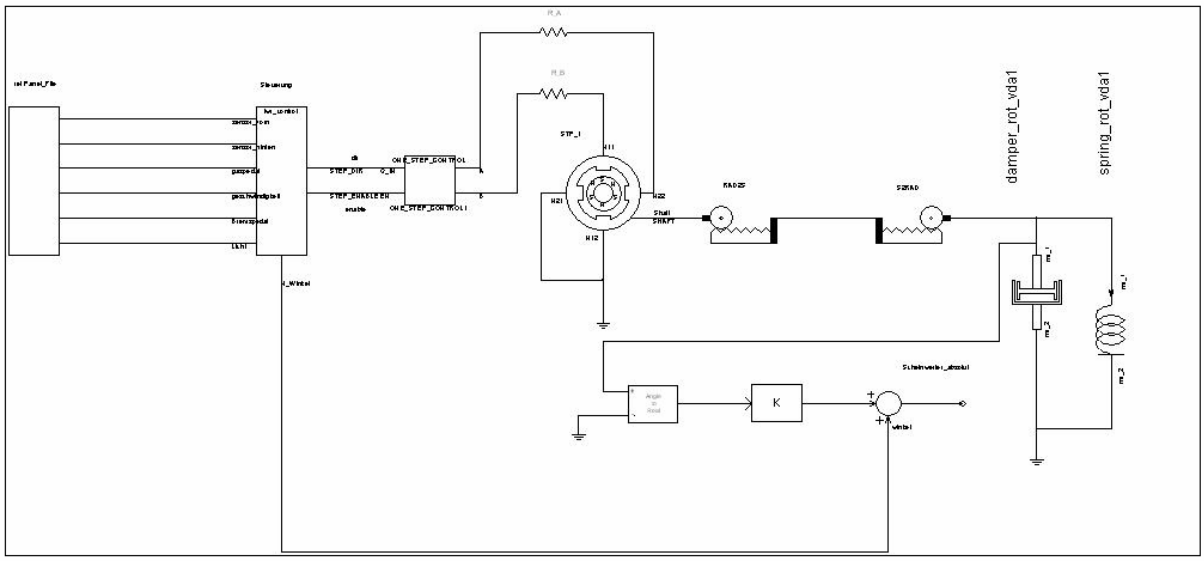


Abbildung 3: Schaltbild LWR (SaberHDL / Synopsys)

Im Schaltbild in Abbildung 3 ist vereinfacht die Struktur des LWR erkennbar. Die Daten des Fahrzyklus, die in der Praxis direkt im Fahrzeug gemessen werden, werden bei der Simulation aus einer Datei eingelesen.

Der Schrittmotor wird im Beispiel im Vollschrittverfahren betrieben. Der Algorithmus errechnet jeweils den nächsten notwendigen Schritt sowie die Pause bis zum Folgeschritt. Im nächsten Block ("One_Step_Control") werden die erforderlichen Spannungspulse für den nächsten Schritt erzeugt und an den Schrittmotor gegeben. Das Lampengestänge mit Scheinwerfer wird etwas vereinfacht durch 2 Wandler zwischen rotatorischer und translatorischer Bewegung sowie einen rotatorischen Dämpfer und eine Drehfeder nachgebildet.

Da in der Simulation die Scheinwerferposition immer nur relativ zum Fahrzeug errechnet wird, ist zur Darstellung der absoluten Position eine kleine Hilfsschaltung erforderlich, um die aktuelle Fahrzeugneigung zu berücksichtigen.

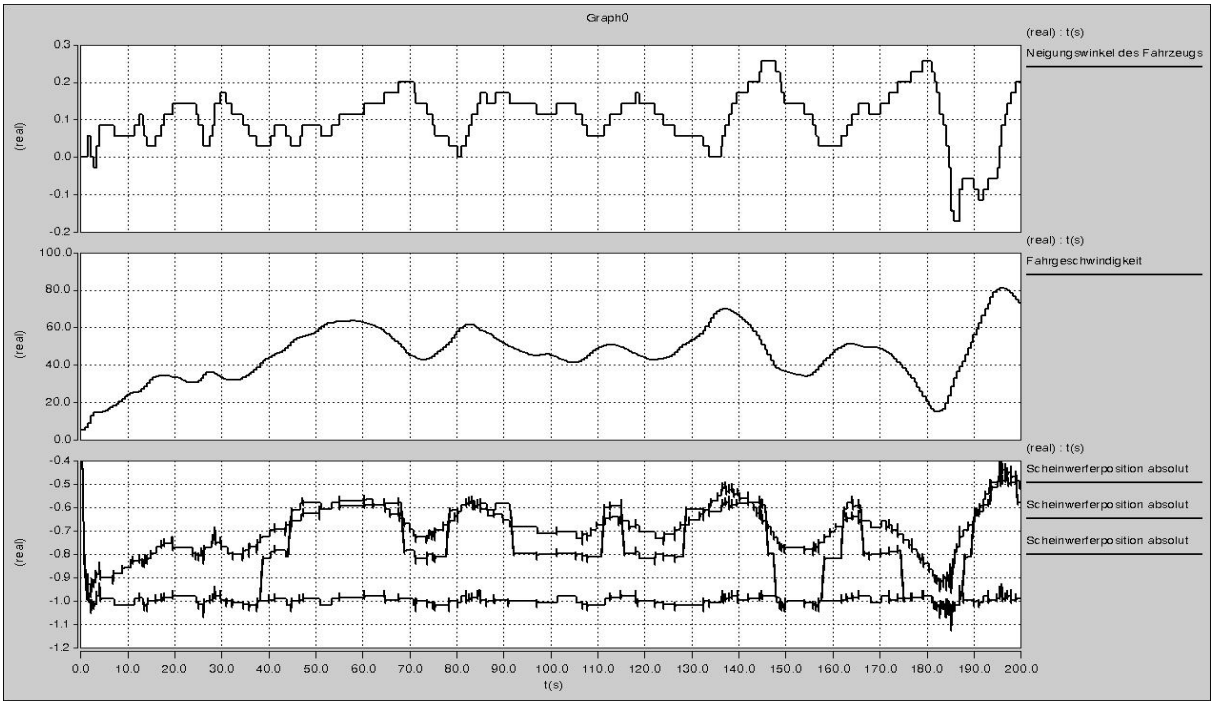


Abbildung 4: Simulationsergebnisse für verschiedene Regelungs-Algorithmen

Abbildung 4 zeigt die Simulationsergebnisse für 3 einfache Varianten des Steueralgorithmus. Neben der einfachen Festwertsteuerung ist eine Variante mit abgestufter sowie mit kontinuierlicher Berücksichtigung der Fahrgeschwindigkeit dargestellt. Die Simulation ermöglicht eine schnelle Analyse des Einflusses unterschiedlicher Steuerungsalgorithmen.

4 Ausblick

Als nächste Schritte sind im AK30 neben der weiteren Vervollständigung der vorhandenen Bibliotheken die Erstellung weiterer Bibliotheken geplant. So ist u.a. eine Bibliothek mit speziellen Modellen für die Komponenten im Hybridfahrzeug geplant, in der auch Probleme der elektromagnetischen Verträglichkeit (EMV) berücksichtigt werden sollen. Ein weiterer Schwerpunkt wird die Festlegung von einheitlichen Schnittstellen zu anderen Tools (z.B. Matlab/Simulink) bzw. Simulationssprachen (C++, SystemC, MODELICA) sein. Für den Modellaustausch ist eine Richtlinie in Vorbereitung, in der neben dem Ablauf auch ein Vorschlag für die Struktur der auszutauschenden Modelle enthalten sein wird.

Literatur

[1] Ashenden, P.; Peterson, G.D.; Teegarden, D. E.; *The System Designer's Guide to VHDL-AMS,* Morgan Kaufmann, 2002

[2] Schwarz, P.; Haase, J.: *Erstellung einer VHDL-AMS-Modellbibliothek für die Simulation von Kfz-Systemen***,** FAT-Schrift 207. 2007

[3] Hessel, E; Lang, Th.: *Systemsimulation aus einem Guss* , VDI-Berichte Nr. 2000.2007

DAEs, Modelica-Notation and Variable Structures in Simulators – a Comparative Study

Felix Breitenecker, Florian Judex, Institute for Analysis and Scientific Computing, Vienna University of Technology; Felix.Breitenecker@tuwien.ac.at
Günther Zauner, Nikolas Popper, ‚Die Drahtwarenhandlung' Simulation Services, Vienna; Guenther.Zauner@drahtwarenhandlung.at

Summary

Object-oriented approaches, DAE modelling, variable structure modelling, Modelica notation and other developments have extended the CSSL standard for simulation languages essentially. After a review of the extended CSSL structure, this contributions compares the availability of advanced and structural features in seventeen simulators - event description, event handling, DAE solving, physical modelling, index reduction, Modelica notation, state chart modelling, structural dynamic modelling, frequency analysis, and extended environment, complemented by examples.

1 CSSL – Standard and Extension

1.1 Implicit Models – Differential-Algebraic Equations

In classical CSSL Standard, for a long time the explicit state space description $\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t),\vec{u}(t),t,\vec{p}),\quad \vec{x}(t_0) = \vec{x}_0$ played the *dominant* role; additional constraints and implicit models had to be transformed 'manually'. From the 1990s on, the simulators started to take care on these very natural phenomena of implicit structures. Consequently, they started to deal with implicit state space descriptions and constraints, in general with so-called DAE models (differential algebraic equations) $F(\dot{\vec{y}}(t),\vec{y}(t),\vec{u}(t),t,\vec{p}) = \vec{0} \quad \vec{y}(t_0) = \vec{y}_0$, whereby the so-called extended state vector $\vec{y}(t)$ can be splitted into the differential state vectors $\vec{x}(t)$ and into the algebraic state vector $\vec{z}(t)$: $\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t),\vec{z}(t),\vec{u}(t),t,\vec{p}) = 0,\quad x(t_0) = \vec{x}_0,\quad g(\vec{x}(t),\vec{z}(t),\vec{u}(t),t,\vec{p}) = 0$

The above given DAEs can be solved by extended ODE solvers and by implicit DAE solvers. Three different approaches may be used:

i) *Nested Approach*, using classical ODE solver
   a. given x_n, solving first numerically $g(x_n,z_n) = 0 \Rightarrow z_n = z_n(x_n) = \hat{g}^{-1}(x_n)$, e. g. by modified Newton iteration, and
   b. applying ODE method, evolving $x_{n+1} = \Phi_E(x_n, z_n(x_n), t_n)$.

ii) *Simultaneous Approach*, using an implicit DAE solver;
   given x_n, solving $g(x_{n+1},z_{n+1}) = 0$ and $\Phi_I(x_{n+1},x_n,z_{n+1},t_{n+1}) = 0$ simultaneously.

iii) *Symbolic Approach*, determining in advance the explicit form solving $g(x,z) = 0 \Rightarrow z = z(x) = \hat{g}^{-1}(x)$ by symbolic computations e.g. within the model translator, and using classical ODE solvers.

The *Symbolic Approach* requires a symbolic inversion of the algebraic equations, which in many cases is not possible or not adequate; furthermore the model translator must not only sort equations, it must be able to perform symbolic manipulations on the equations. The Nested *Approach* – up to now most commonly used – requires a numerical inversion of the algebraic equations: each evaluation of the vector of derivatives (called by the ODE solver) has to start an iterative procedure to solve the algebraic equation. Here classical ODE solvers can be used. The *Simultaneous Approach* requires an implicit ODE solver – usually an implicit stiff equation solver. Although also working with iterations, these solvers show much more efficiency and provide more flexibility for modelling (DASSL, IDA-DASSL, and LSODE – solvers).

However, hidden is another problem: the 'DAE index' problem. Roughly speaking, a DAE model is of index *n*, if n differentiations of the DAE result in an ODE system (with an increased state space). The implicit ODE solvers for the *Simultaneous Approach* guarantee convergence only in case of DAE index $n = 1$. Models with higher DAE index must / should be transformed to models with DAE index $n = 1$. This transformation is based on symbolic differentiation and symbolic manipulation of the high index DAE system, and there is no unique solution to this index reduction. In object-oriented simulation systems, like in Dymola, physical a-causal modelling plays an important role, which results in DAEs with sometimes higher index. These systems put emphasis on index reduction (in the translator) to DAEs with index $n = 1$ in order to apply implicit ODE solvers (*Simultaneous Approach*

1.2 Discrete Elements in Continuous Simulation - Events

The CSSL standard also defines segments for discrete actions, first mainly used for modelling discrete control. So-called DISCRETE regions or sections manage the communication between discrete and continuous world and compute the discrete model parts. These *Time Events* cause the simulation engine to interrupt the ODE solver and handle the event. For generality, efficient implementations set up and handle event lists, representing the time instants of discrete actions and the calculations associated with the action, where in-between consecutive discrete actions the ODE solver is to be called.

Much more complicated, but defined in CSSL, are the so-called state events. Here, a discrete action takes place at a time *instant*, which is not known in advance, it is only known as a function of the states. As example, we consider the pendulum with constraints (*Constrained Pendulum*). If the pendulum is swinging, it may hit a pin positioned at angle φ_p with distance l_p from the point of suspension. In this case, the pendulum swings on with the position of the pin as the point of rotation. The shortened length is $l_s = l - l_p$. and the angular velocity $\dot{\varphi}$ is changed at position φ_p from $\dot{\varphi}$ to $\dot{\varphi} l / l_s$, etc. These discontinuous changes are state events, not known in advance. For such state events, the classical state space description is extended by the so-called state event function $h(x)$, the zero of which determines the event: $\vec{x}(t) = \vec{f}(\vec{x}(t), \vec{u}(t), \vec{p}, t), \quad h(\vec{x}(t), \vec{u}(t), \vec{p}, t) = 0$. In this notation, the model for *Constrained Pendulum* involves two different events: change of length parameter (SE-P), and change of state (SE-S):

$$\dot{\varphi}_1 = \varphi_2, \ \dot{\varphi}_2 = -\frac{g}{l}\sin\varphi_1 - \frac{d}{m}\varphi_2, \qquad h(\varphi_1, \varphi_2) = \varphi_1 - \varphi_p = 0$$

Generally, state events (SE) can be classified in four types:

Type 1 – parameters change discontinuously (**SE-P**),

Type 2 - inputs change discontinuously (**SE-I**),

Type 3 - states change discontinuously (**SE-S**), and

Type 4 - state vector dimension changes (**SE-D**), including change of model equations.

State events type 1 (SE-P) could also be formulated by means of IF-THEN-ELSE constructs and by switches in graphical model descriptions, without synchronisation with the ODE solver. The necessity of a state event formulation depends on the accuracy wanted. Big changes in parameters may cause problems for ODE solvers with stepsize control. State events of type 3 (SE-S) are essential state events. They must be located, transformed into a time event, and modelled in discrete model parts. State events of type 4 (SE-D) are also essential ones. In principle, they are associated with hybrid modelling: models following each other in consecutive order build up a sequence of dynamic processes.

The handling of a state event requires four steps:

i. Detection of the event, usually by checking the change of the sign of $h(x)$ within the solver step over $[t_i, t_{i+1}]$

ii. Localisation of the event by a proper algorithm determining the time t^* when the event occurs and performing the last solver step over $[t_i, t^*]$

iii. Service of the event: calculating / setting new parameters, inputs and states; switching to new equations

iv. Restart of the ODE solver at time t^* with solver step over $[\ t^* = t_{i+1}, t_{i+2}]$

State events are facing simulators with severe problems. Up to now, the simulation engine had to call independent algorithms, now a root finder for the state event function h needs results from the ODE solver, and the ODE solver calls the root finder by checking the sign of h. For finding the root of the state event function $h(x)$, either interpolative algorithms (MATLAB/Simulink) or iterative algorithms are used (ACSL, Dymola).

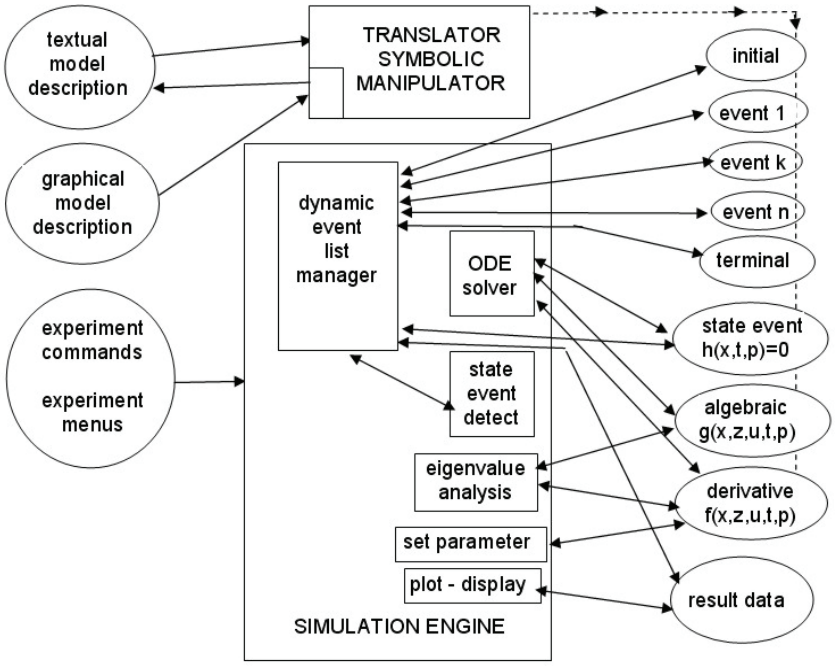


Figure 1: Extended Structure of a Simulation Systems due to Extensions with Discrete Elements and DAEs

In order to incorporate DAEs and discrete elements, the simulator's translator must now extract from the model description the dynamic differential equations (derivative), the dynamic algebraic equations (algebraic), and the events (event i) with static algebraic equations and event time; additionally, for state events the simulator's translator must extract from the model description additionally the state event functions (state event j) with the associated action.

In the simulator kernel, the static event management must be made dynamically: state events are dynamically handled and transformed to time events. In principle, the kernel of the simulation engine has become an event handler, managing a complex event list with feedbacks. In case of a structural change of the system equations (state event of type 4 – SE-D), simulators usually can manage only fixed structures of the state space. The technique used is to 'freeze' the states that are bound by conditions causing the event. In case of a complete change of equations, both systems are calculated together, freezing one according to the event. One way around is to make use of the experimental frame: the simulation engine only detects and localises the event, and updates the system until the event time. Then control is given back to the experimental frame. The state event is now serviced in the experimental frame, using features of the environment. Then a new simulation run is restarted (modelling of the structural changes in the experimental frame).

```
PROGRAM constrained pendulum
CONSTANT m = 1.02, g = 9.81, d =0.2
CONSTANT lf=1, lp=0.7
DERIVATIVE dynamics
  ddphi = -g*sin(phi)/l - d*dphi/m
  dphi  = integ ( ddphi, dphi0)
  phi   = integ ( dphi, phi0)
  SCHEDULE hit   .XN. (phi-phip)
  SCHEDULE leave .XP. (phi-phip)
END ! of dynamics
DISCRETE hit
  l = ls; dphi = dphi*lf/ls
END ! of hit
DISCRETE leave
  l = lf; dphi = dphi*ls/lf
END ! of leave
END ! of constrained pendulum
```

Figure 2: *Constrained Pendulum*: Continuous Model with State Events (ACSL)

The *Constrained Pendulum* example involves a state event of type 1 (SE-P) and type 3 (SE-S). A classical ACSL model description works with two discrete sections `hit` and `leave`, representing the two different modes, both called from the dynamic equations in the derivative section (Figure 2). Dymola defines events and their scheduling implicitly by WHEN – or IF - constructs in the dynamic model description, in case of the discussed example e.g. by

```
WHEN phi-phip=0 AND phi>phip
    THEN l = ls; dphi = dphi*lf/ls
```

In case of more complex event descriptions, the WHEN – or IF – clauses are put into an ALGORITHM section similar to ACSL's DISCRETE section. In graphical model descriptions, we again are faced with the problem that calculations at discrete time instants are difficult to formulate. For the detection of the event, SIMULINK provides the `HIT CROSSING` block (in new Simulink version implicitly defined). This block starts state event detection (interpolation method) depending on the input, the state event function, and outputs a trigger signal, which may call a triggered subsystem servicing the event.

1.3 Comparison of Extended Features of Simulators

Event description (ED), state event handling (SEH) and DAE support (DAE) with or without index reduction (IR) became desirable structural features of simulators, supported directly or indirectly. Table 3 compares the availability of these features in the MATLAB / Simulink System, in ACSL and in Dymola.

In Table 1, the availability of features is indicated by 'yes' and 'no'; a 'yes' in parenthesis '(yes)' means, that the feature is complex to use. MS - 'Model Sorting', is a standard feature of a simulator – but missing in MATLAB (in principle, MATLAB cannot be called a simulator).

	MS - Model Sorting	ED -Event Description	SHE - State Event Handling	DAE - DAE Solver	IR - Index Reduction
MATLAB	no	no	(yes)	(yes)	no
Simulink	yes	(yes)	(yes)	(yes)	no
MATLAB / Simulink	yes	yes	yes	(yes)	no
ACSL	yes	yes	yes	yes	no
Dymola	yes	yes	yes	yes	yes

Table 1: Comparison of Simulators' Extended Features (Event Handling and DAES

On the other hand, MATLAB's ODE solvers offer limited features for DAEs (systems with mass matrix) and an integration stop on event condition, so that SHE and DAE get a ('yes'). In Simulink, event descriptions are possible by means of triggered subsystems, so that ED gets a '(yes)' because of complexity. A combination of MATLAB and Simulink suggest putting the event description and handling at MATLAB level, so that ED and SHE get both a 'yes'. DAE solving is based on modified ODE solvers, using the nested approach (see before), so DE gets only a '(yes)' for all MATLAB/Simulink combinations. ACSL is a classical simulator with sophisticated state event handling, and since version 10 (2001) DAEs can be modelled directly by the residuum construct, and they are solved by the DASSL algorithm (a well-known direct DAE solver, based on the simultaneous approach), or by modified ODE solvers (nested approach) – so 'yes' for ED, SHE, and DAE. In case of DAE index $n = 1$, the DASSL algorithm guarantees convergence, in case of higher index integration may fail. ACSL does not perform index reduction (IR 'no'). Dymola is a modern simulator, implemented in C, based on physical modelling. Model description may be given by implicit laws; symbolic manipulations extract a proper ODE or DAE state space system, with index reduction for high index DAE systems.

2 From CSSL to Modelica and VHDL-AMS

In the 1990s, many attempts have been made to improve and to extend the CSSL structure, especially for the task of mathematical modelling. The basic problem was the state space description limiting modular and flexible modelling. Two developments helped to overcome this problem. On modelling level, the idea of physical modelling gave new input, and on implementation level, the object-oriented view helped to leave the constraints of input/output relations. In physical modelling, a typical procedure for modelling is to cut a system into subsystems and to account for the behaviour at the interfaces. Balances of mass, energy and momentum and material equations model each subsystem. The complete model is obtained by combining the descriptions of the subsystems and the interfaces. A model is considered as a constraint between system variables, which leads naturally to DAE descriptions. The approach is very convenient for building reusable model libraries.

In 1996, the situation was thus similar to the mid 1960s when CSSL was defined as a unification of the techniques and ideas of many different simulation programs. An international effort was initiated in September 1996 for bringing together expertise in object-oriented physical modelling (port based modelling) and defining a modern uniform modelling language – mainly driven by the developers of Dymola.

The new modelling language is called *Modelica*. Modelica is intended for modelling within many application domains such as electrical circuits, multibody systems, drive trains, hydraulics, thermodynamical systems, and chemical processes etc. It supports several modelling formalisms: ordinary differential equations, differential-algebraic equations, bond graphs, finite state automata, and Petri nets etc. Modelica is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users. Modelica is a not a simulator, Modelica is a modelling language, supporting and generating mathematical models in physical domains. The translator from Modelica into the target simulator must not only be able to sort equations, it must be able to process the implicit equations symbolically and to perform DAE index reduction (or a way around).

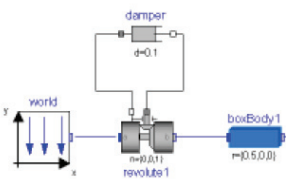
When the development of Modelica started, also a competitive development, the extension of VHDL towards VHDL-AMS was initiated. Both modelling languages aimed for general-purpose use, but VHDL-AMS mainly addresses circuit design, and Modelica covers the broader area of physical modelling; modelling constructs such as Petri nets and finite automata could broaden the application area, as soon as suitable simulators can read the model definitions. Modelica offers a textual and graphical modelling concept, where the connections of physical blocks are bidirectional physical couplings, and not directed flow.

Up to now – similar to VHDL-AMS – some simulation systems understand Modelica (2008; generic – new simulator with Modelica modelling, extension - Modelica modelling interface for existing simulator):

- Dymola from Dynasim (generic),
- MathModelica from MathCore Engineering (generic)
- SimulationX from ISI (generic/extension)
- Scilab/Scicos (extension)
- MapleSim (extension, announced)
- Open Modelica - since 2004 the University of Lyngby develops an provides an open Modelica simulation environment (generic),
- Mosilab - Fraunhofer Gesellschaft Dresden, Modelica simulator with dynamic structures

The *Constrained Pendulum* example can be formulated in Modelica textually as a physical law for angular acceleration. The event with parameter change is put into an `algorithm` section, defining and scheduling the parameter event (SE-P). Modelica allows combining textual and graphical modelling. So, the basic physical dynamics can be modelled graphically with joint and mass elements, and the event is described in an `algorithm` section, with variables interfacing to the predefined variables in the graphical model part (Figure 3).

```
equation /*pendulum*/
  v = length*der(phi);
  vdot = der(v);
  mass*vdot/length + mass*g*sin(phi)
  +damping*v = 0;
algorithm
 if (phi<=phipin) then length:=ls; end if;
 if (phi>phipin) then length:=ll;
  end if;
```




```
algorithm
if (revolute1.phi
     <= phipin then
    revolute1.length:=ls;
end if;
if (revolute1.phi
     < phipin then
    revolute1.length:=ll;
end if;
```

Figure 3: Textual Dymola Model (left) and Mixed Textual/Graphical Model (right)

3 Modelling with State Charts

In the end of the 1990s, computer science initiated a new development for modelling discontinuous changes. The *Unified Modelling Language* (UML) is one of the most important standards for specification and design of object oriented systems. This standard was tuned for real time applications in the form of a new proposal, *UML Real-Time* (UML-RT). By means of UML-RT, objects can hold the dynamic behaviour of an ODE.

In 1999, a simulation research group at the Technical University of St. Petersburg used this approach in combination with a hybrid state machine for the development of a hybrid simulator (*MVS*), from 2000 on *available* commercially as simulator *AnyLogic*. The modelling language of AnyLogic is an extension of UML-RT; the main building block is the *Active Object*. Active objects have internal structure and behaviour, and allow encapsulating of other objects to any desired depth. Relationships between active objects set up the hybrid model. Active objects interact with their surroundings solely through boundary objects: ports for discrete communication, and variables for continuous communication (Figure 6). The activities within an object are usually defined by state charts (extended state machine). While discrete model parts are described by means of state charts, events, timers and messages, the continuous model parts are described by means of ODEs and DAEs (CSSL-type notation) and with state charts within objects.
AnyLogic influenced further developments for hybrid and structural dynamic systems, and led to a discussion in the Modelica community with respect to a proper implementation of state charts in Modelica. The principle question is, whether state charts are to be seen as comfortable way to describe complex WHEN – and IF – constructs, being part of the model, or whether state charts control different models from a higher level. At present (2008) a free Modelica state chart library 'emulates' state charts by Boolean variables and IF – THEN – ELSE constructs.

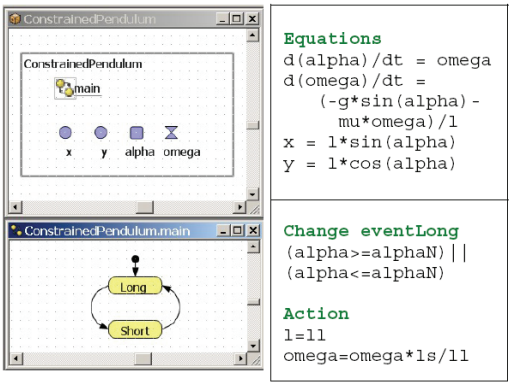


Figure 4: AnyLogic Model for *Constrained Pendulum*, Simple

An AnyLogic implementation for the *Constrained Pendulum* may follow the implementation for the bouncing ball (Figure 4). An primary active object (Constrained Pendulum)'holds' the equations for the pendulum, together with a state chart (main) switching between short and long pendulum. The state chart nodes are empty; the arcs define the events. Internally, AnyLogic restarts at each hit the same pendulum model (trivial hybrid decomposition).

4 Modelling with State Charts

In continuous and hybrid simulation, the explicit or implicit state space description is used as common denominator. This state space may be described textually, or by signal-oriented graphic blocks (e.g. SIMULINK), or by physically based block descriptions (Modelica, VHDL-AMS). In discrete simulation, we meet very different techniques for the model frame. Application-oriented flow diagrams, network diagrams, state diagrams, etc. allow describing complex behaviour of event-driven dynamics. Usually these descriptions are mapped to an

event-based description. On the other side, the simulator kernel is similar for discrete and continuous simulators. The model description is mapped to an event list with adequate update functions of the states within state update events. In discrete simulation, the states are usually the status variables of servers and queues in the model, and state update is simple increase or decrease by increments; complex logic conditions may accompany the scheduling of events. In continuous simulation the state space is based of various laws used in the application area, and usually defined by differential-algebraic equations. DAE solvers generate a grid for the approximation of the solutions. This grid drives an event list with state update events using complex formula depending on the chosen DAE solver and on the defined DAE. Additional time events and state events are inserted into the global event list.

Hybrid systems often come together with a change of the dimension of the state space, then called *structural-dynamic systems*. The dynamic change of the state space is caused by a state event of type SE-D. In contrary to state events SE-P and SE-S, states and derivatives may change continuously and differentiable in case of structural change. In principle, *structural-dynamic systems* can be seen from two extreme viewpoints. The one says, in a maximal state space, state events switch on and off algebraic conditions, which freeze certain states for certain periods. The other one says that a global discrete state space controls local models with fixed state spaces, whereby the local models may be also discrete or static.

These viewpoints derive two different approaches for structural dynamic systems modelling,

- *The maximal state space*, and the
- *hybrid decomposition*.

4.1 Maximal State Space for Structural-Dynamic Systems – Internal Events

Most implementations of physically based model descriptions support a big monolithic model description, derived from laws, ODEs, DAEs, state event functions and *internal events*. The state space is maximal and static, index reduction in combination with constraints keep a consistent state space. Dymola, OpenModelica, and VHDL-AMS follow this approach, which can be classified with respect to event implementation. The approach handles all events of any kind (SE-P, SE-S, and SE-D) within the ODE solver frame, also events which change the state space dimension (change of degree of freedoms) – consequently called *internal events*.

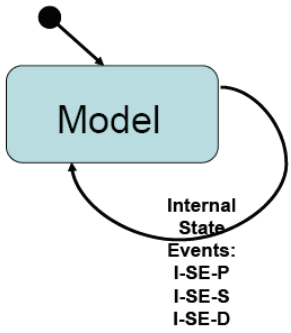


Figure 5: State Chart Control for *Internal Events* of one Model

Using the classical state chart notation, *internal state events* **I-SE** caused by the model schedule the model itself, with usually different re-initialisations (depending on the event type I-SE-P, I-SES, I-SE-D; Figure 5). VHDL-AMS and Dymola follow this approach, handling also DAE models with index higher than 1; discrete model parts are only supported at event level. ACSL and MATLAB / Simulink generate also a maximal state space.

4.2 Hybrid Decomposition for Structural-Dynamic Systems – External Events

The hybrid decomposition approach makes use of *external events* (E-SE), which control the sequence and the serial coupling of one model or of more models. A convenient tool for

switching between models is a state chart, driven by the *external events* – which itself are generated by the models. Following e.g. the UML-RT notation, control for continuous models and for discrete actions can by modelled by state charts. Figure 10 shows the hybrid coupling of two models, which may be extended to an arbitrary number of models, with possible events E-SE-P, E-SE-S, and ESE-D. As special case, this technique may be also used for serial conditional 'execution' of one model – Figure 6 (only for SE-P and SE-S).

This approach additionally allows not only dynamically changing state spaces, but also different model types, like ODEs, linear ODEs (to be analysed by linear theory), PDEs, etc. to be processed in serial or also in parallel, so that also co-simulation can be formulated based on external events. The approach allows handling all events also outside the ODE solver frame. After an event, a very new model can be started. This procedure may make sense especially in case of events of type SE-D and SE-S. As consequence, consecutive models of different state spaces may be used.

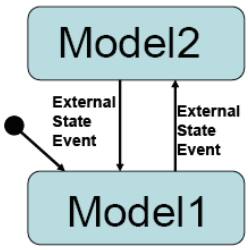


Figure 6: State Chart Control for *External Events* for two Models

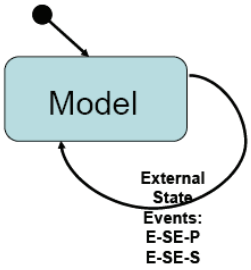


Figure 7: State Chart Control for *External Events* for one Model

Figure 7 shows a structure for a simulator supporting structural dynamic modelling and simulation. The figure summarises the outlined ideas by extending the CSSL structure by control model, external events and multiple models. The main extension is that the translator generates not only one DAE model; he generates several DAE models from the (sub)model descriptions, and external events from the connection model, controlling the model execution sequence in the highest level of the dynamic event list.

The approach allows handling all events also outside the ODE solver frame. After an event, a very new model can be started. This procedure may make sense especially in case of events of type SE-D and SE-S. As consequence, consecutive models of different state spaces may be used.

Figure 8 shows a structure for a simulator supporting structural dynamic modelling and simulation. The figure summarises the outlined ideas by extending the CSSL structure by control model, external events and multiple models. The main extension is that the translator generates not only one DAE model; he generates several DAE models from the (sub)model descriptions, and external events from the connection model, controlling the model execution sequence in the highest level of the dynamic event list. There, all (sub)models may be precompiled, or the new recent state space may be determined and translated to a DAE system in case of the external event (interpretative technique).

4.3 Mixed Approach with Internal and External Events

A simulator structure as proposed in Figure 8 is a very general one, because it allows as well external as ell as internal events, so that hybrid coupling of any kind with internal and external events is possible (Figure 9). Both approaches have advantages and disadvantages. The classical Dymola approach generates a fast simulation, because of the monolithic program.

However, the state space is static. Furthermore, Modelica centres on physical modelling. A hybrid approach handles separate model parts and must control the external events.

Consequently, two levels of programs have to be generated: dynamic models, and a control program – today's implementations are interpretative and not compiling, so that simulation times increase - but the overall state space is indeed dynamic.

A challenge for the future lies in the combination of both approaches.

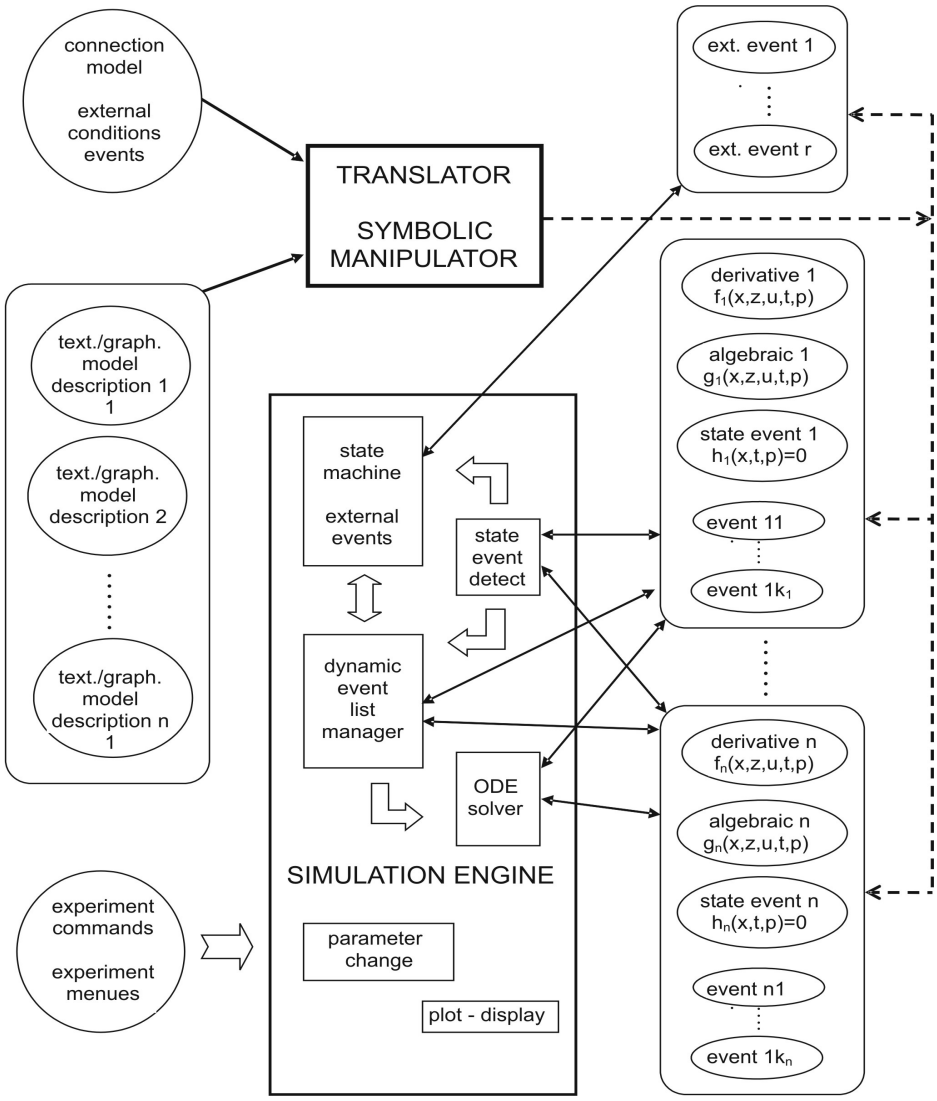


Figure 8: Structure for a Simulation System with *External State Events* E-SE and Classical Internal State Events I-SE for Controlling

The main ideas are:

- Moderate hybrid decomposition
- External and internal events
- Efficient implementation of models and control

For parameter state events (SE-P) an implementation with an internal event may be sufficient (I-SE-P), for an event of SE-S type implementation with an external event may be advantageous because of easier state re-initialisation (E-SE-S), and for a structural model change (SE-D) an implementation with an external event may be preferred (E-SE-D), because of much easier handling of the dynamic state change – and less necessity for index reduction. An efficient control of the models sequence can be made by state charts, but also by a well-defined definitions and distinction of IF - and WHEN - constructs, like discussed in extensions of Scilab/ Scicos for Modelica models.

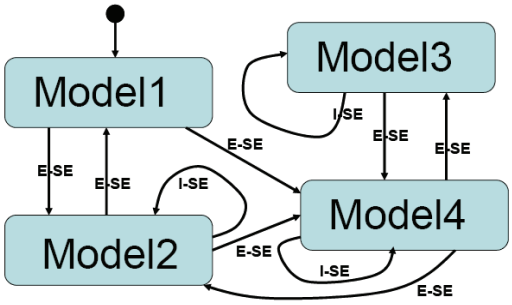


Figure9: State Chart Control for Different Models with *Internal* and *External Event*

5 Structural Features of Simulators

While the *extended features* discussed before address the CSSL-standard, *structural features* characterise features for physical modelling and for structural dynamic systems. This section investigates the availability or of structural features in some simulators, summarises in Table 2. Furthermore, is should be discussed, which software structure these simulators use (compared to Figure 1 and Figure 8). The extended features may be classified as follows:

- Support of a-causal physical modelling (sometimes called port-based modelling) at textual (PM-T) or graphical level (PM-G),
- Modelica standard (MOD) for a-causal physical modelling ,
- Decomposition of structural dynamic systems with dynamic features (SD) – features for external events, and
- Support of state chart modelling or of a similar construct, by means of textual (SC-T) or graphical (SC-G) constructs.

In principle, each combination of the above features is possible. By means of the maximal state space approach, each classic simulator can handle structural dynamic systems, but a-causal modelling may be supported or not, and state chart modelling may be available or not. Simulators with a-causal modelling may support hybrid decomposition or not, and state chart modelling may be available or not. Simulators with features for state chart modelling may support hybrid decomposition or not, and a-causal modelling may be offered or not. In general, interpreter-oriented simulators offer more structural flexibility, but modern software structures would allow also flexibility with precompiled models or with models compiled 'on the fly'.

In addition, of interest are also structural features as

- simulation-driven visualisation (visualisation objects defined with model objects; VIS),
- frequency domain analysis and linearization for steady state analysis (FA), and
- extended environment for complex experiments and data processing (ENV).

In the following sections, simulators and simulation systems are investigated in order to check the availability of these structural features. For some of the simulators, implementation templates with the *Constrained Pendulum* are discussed.

5.1 MATLAB / Simulink / Stateflow

The mainly interpretative systems MATLAB / Simulink offer different approaches. First, MATLAB itself allows any kind of static and dynamic decomposition (SD 'yes'), but MATLAB is not a simulator, because the model equations have to be provided in a sorted manner, to be called from an *ODE* solver (MS 'no'). Second, MATLAB allows hybrid decomposition at MATLAB level with Simulink models. There, from *MATLAB* different Simulink models are called conditionally, and in Simulink, a state event is determined by the hit-crossing block (terminating the simulation). For control, in MATLAB only IF – THEN constructs are available. Figure 10– MATLAB control model, and graphical Simulink model, show a hybrid decomposition of this type for the *Constrained Pendulum.*

MATLAB is a very *powerful* environment with various modules. Simulink is MATLAB's simulation module for block-oriented dynamic models (directed signal graphs), which can be combined with Stateflow, MATLAB's module for event-driven state changes described by state charts (SC-T and SC-G 'yes').

```
if ((phi_p-phi0)*phi_p<0 |
              (phi0==phi_p & phi_p*v>0))
   dphi0=v/ls;
   sim('pendulum_short',[t(length(t)),10]);
   v=dphi(length(dphi))*ls;
else
   dphi0=v/l;
   sim('pendulum_long',[t(length(t)),10]);
   v=dphi(length(dphi))*l;
end
```

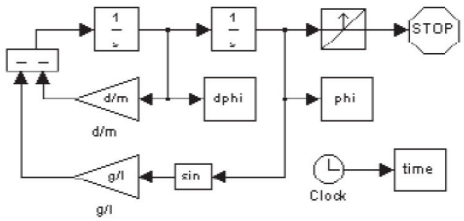


Figure 10: MATLAB Controlling a Simulink Model - *External Event* Approach

At Simulink level, Stateflow, Simulink's state chart modelling tool, may control different submodels. These submodels may be dynamic models based on ODEs (DAEs), or static models describing discrete actions (events). Consequently, Stateflow can be used for implementation of the *Constrained Pendulum*, where the state charts control length and change of velocities in case of hit by triggering the static changes. Alternatively, Stateflow could control two different submodels representing long and short pendulum enabled and disabled by the state chart control. Internally Simulink generates a state space with 'double' dimension, because Simulink can only work with a maximal state space and does not allow hybrid decomposition (SD 'no).

Neither MATLAB nor Simulink support a-causal modelling. New MATLAB modules for physical modelling (e.g. *Hydraulic Blockset*) are precompiled to a classical state space (PM-T and PM-G 'no'), and furthermore Modelica modelling is not supported (MOD 'no') – Mathworks developers are working hard on some kind of real physical modelling and on Modelica modelling. For DAEs, MATLAB and Simulink offer modified LSODE solvers (implicit solvers) for the nested DAE solving approach. In MATLAB any kind of simulation – driven visualisation can be programmed and used in MATLAB or Simulink or in both, but not based on the model definition blocks (VIS '(yes)'). From the beginning on, MATLAB and Simulink offered frequency analysis (FA 'yes'), and clearly, MATLAB is a very powerful environment for Simulink, Stateflow, for all other Toolboxes, and for MATLAB itself (ENV 'yes').

5.2 ACSL

ACSL – Advanced Continuous Simulation Language – has been developed since more than 25 years. ACSL was strongly influenced by the CSSL standard. ACSL' software structure is a direct mapping of the structure in Figure 2. Implementations of the *Constrained Pendulum* have been shown in the previous sections, as example for modelling due to CSSL standard. ACSL' development as simulator seems to have ended, as the new developers (Aegis Technologies) concentrate on application-oriented simulation solutions, with models are tailor-made for the specific application. Last extensions were a change to C as basic language (instead of FORTRAN), and DAE features using the nested approach with classical solvers, or direct implicit DAE solving with DASSL Code (DAE 'yes', IR 'no'). From the beginning on, steady state calculation, linearization and frequency analysis was a standard feature of ACSL's simulator kernel (FA 'yes').

Since 2000, the environment has been enriched by modules for modelling and environment modules. The first module was a graphical modeller. ACSL's graphic modeller seems to make use of physical modelling, but in behind classical state spaces as with Simulink's block-

sets for physical modelling are used – PM-T and PM-G 'no'). Furthermore, a simulation-driven visualisation system (third party) is offered (but hard to use) – VIS '(yes)'.

A very interesting module is an extended environment called ACSLMath. ACSLMath was intended to have same features as MATLAB; available is only a subset, but powerful enough for an extended environment (ENV 'yes'), which can be used for hybrid decomposition of a structural dynamic model in almost the same way than MATLAB does (SD 'yes'). Unfortunately the development of ACSLMath has been stopped. In general, there is no intention to make a-causal physical modelling available, also Modelica is not found in the developers' plans (PM-T, PM-G, and MOD 'no').

5.3 Dymola

Dymola has partly been discussed in a section before, together with an implementation for the *Constrained Pendulum* example (Dymola standard implementation, Figure 3). Dymola, introduced by F. E. Cellier as a-causal modelling language, and developed to a simulator by H. Elmquist, can be called the mother of Modelica.

Dymola is based on a-causal physical modelling and initiated Modelica; consequently, it fully supports Modelica these structural features (PM-T, PM-G, and MOD 'yes'). Together with the model objects, also graphical objects may be defined, so that simulation based pseudo-3D visualisation is available (VIS 'yes'). A key feature of Dymola is the very sophisticated index reduction by the modified Pantelides algorithm, so Dymola handles any DAE system, also with higher index, with bravura (DAE and IR 'yes'). For DAE solving, modified DASSL algorithms are used. In software structure, Dymola is similar to ACSL, using an extended CSSL structure as given in Figure 1 – with the modification that all discrete actions are put into one event module, where CASE - constructs distinguish between the different events (this structure is based on the first simulator engine Dymola used, the DS-Block System of DLR Oberpfaffenhofen).

Dymola comes with a graphical modelling and basic simulation environment, and provides a simple script language as extended environment; new releases offer also optimisation, as built-in function of the simulator. Furthermore, based on Modelica's matrix functions some task of an environment can be performed – so ENV ('yes') – available, but complex/uncomfortable. Dymola offers also a Modelica – compatible state chart library, which allows to model complex conditions (internally translated into IF – THEN – ELSE or WHEN constructs - SC-T and SC-G '(yes)'). Figure 11 shows an implementation of the *Constrained Pendulum* using this library.

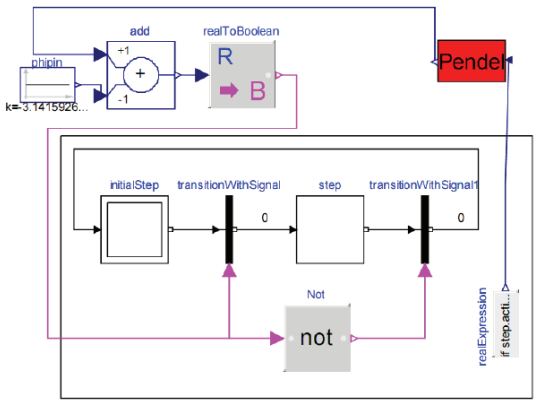


Figure11: Graphical Dymola Model for *Constrained Pendulum* with *Internal Events Managed* by Elements State Chart Library

Up to now (2008) the Modelica definition says nothing about structural dynamic systems, and Dymola builds up a maximal state space (Modelica notation is used in Dymola's state chart library) without dynamic structures.- SD 'no'.

Another interesting and remarkable development started in 2006 – Modelica's basic static calculation features become notable. These basic features include any kind of vector and matrix operations, and they can be extended by Modelica's generic extension mechanism. In principle, 'static' Modelica defines a MATLAB-like language. Simulators being capable of understanding Modelica, must consequently also support these static calculations (without any DAE around) – so that each Modelica simulator becomes a 'Mini-MATLAB'. In Dymola, such calculations may be performed in a textual Dymola consisting only of an algorithmic section, without any time advance from the simulator kernel.

5.4 MathModelica

MathModelica, developed by MathCoreAB, was the second simulation system, which understood Modelica modelling. MathModelica is an integrated interactive development, from modelling via simulation to analysis and code integration. As furthermore the MathModelica translator is very similar to Dymola's model translator, clearly all related features are available, including index reduction and use of implicit solvers like DASSL (all DAE, IR, PM-T, PM-G and MOD 'yes'). Model set up e. g. with MathModelica's Mechanics Package (Modelica modelling), looks almost exactly like the model in Dymola, SimulationX, etc.

MathModelica follows a software model different to CSSL standard. The user interface consists of a graphical model editor and notebooks. There, a *simulation center* controls and documents experiments in the time domain. Documentation, mathematical type setting, and symbolic formula manipulation are provided via Mathematica, as well as Mathematica acts as extended environment for MathModelica (ENV 'yes') – performing any kind of analysis and visualisation (FA and VIS 'yes'). By means of the Mathematica environment, also a hybrid decomposition of structural dynamic systems is possible, with the same technique like in MATLAB (SD – 'yes').

5.5 Mosilab

Since 2004, Fraunhofer Gesellschaft Dresden develops a generic simulator *Mosilab*, which also initiates an extension to Modelica: multiple models controlled by state automata, coupled in serial and in parallel. Furthermore, Mosilab puts emphasis on co-simulation and simulator coupling, whereby for interfacing the same constructs are used than for hybrid decomposition. Mosilab is a generic Modelica simulator, so all basic features are met (ED, SEH, DAE, PM-T, and PM-G 'yes', and MOD '(yes)' – because of subset implementation at present, 2008). For DAE solving, variants of IDA-DASSL solver are used.

Mosilab implements extended state chart modelling, which may be translated directly due to Modelica standard into equivalent IF – THEN constructs, or which can control different models and model executions (SC-T, SC-G, and SD 'yes'). At state chart level, state events of type SE-D control the switching between different models and service the events (E-SE-D). State events affecting a state variable (SE-S type) can be modelled at this external level (E-SE-S type), or also as classic internal event (I-SE-S). Mosilab translates each model separately, and generates a main simulation program out of state charts, controlling the call of the precompiled models and passing data between the models, so that the software model of Mosilab follows the structure in

Figure 8. The textual and graphical constructs for the state charts are modifications of state chart modelling in AnyLogic. Mosilab is in developing, so it supports only a subset of Modelica, and index reduction has not been implemented yet, so that MOD gets a '(yes)' in parenthesis, and IR gets a '(no)' – indicating that the feature is not available at present (2008), but is scheduled for the future. Index reduction at present not available in Mosilab, but planned (IR '(no)') - has become topic of discussion: case studies show, that hybrid decomposition of structural dynamic systems results mainly in DAE systems of index $n = 1$, so that index reduction may be bypassed (except models with contact problems).

Mosilab allows very different approaches for modelling and simulation tasks, to be discussed with the *Constrained Pendulum* example. Three different modelling approaches reflect the distinction between internal and external events as discussed before.

Mosilab Standard Modelica Model. In a standard Modelica approach, the *Constrained Pendulum* is defined in the MOSILAB equation layer as implicit law; the state event, which appears every time when the rope of the pendulum hits or 'leaves' the pin, is modelled in an `algorithm section` with if (or when) – conditions (Figure 12 7).

Mosilab I-SE-P Model with State Charts. MOSILAB's state chart approach models discrete elements by state charts, which may be used instead of IF - or WHEN - clauses, with much higher flexibility and readability in case of complex conditions. There, Boolean variables define the status of the system and are managed by the state chart. Figure 13 shows a Mosilab implementation of the *Constrained Pendulum*: the state charts initialise the system (`initial` state) and manage switching between long and short pendulum, by changing the length appropriately.

```
equation /*pendulum*/
 v = l1*der(phi); vdot = der(v);
 mass*vdot/l1 + mass*g*sin(phi)+damping*v = 0;
algorithm
 if (phi<=phipin) then length:=ls; end if;
 if (phi>phipin) then length:=l1; end if;
end
```

Figure 12: Mosilab Model for *Constrained Pendulum* – Standard Modelica Approach with *Internal Events* (I-SE-P)

```
event Boolean lengthen(start=false),
 shorten(start = false);
equation
lengthen=(phi>phipin); shorten=(phi<=phipin);
equation /*pendulum*/
 v = l1*der(phi); vdot = der(v);
 mass*vdot/l1 + mass*g*sin(phi)+damping*v= 0;
statechart
 state LengthSwitch extends State;
 State Short,Long,Initial(isInitial=true);
transition Initial -> Long end transition;
transition Long -> Short event shorten action
 length := ls;
end transition;
transition Short -> Long event lengthen action
 length := l1;
end transition; end LengthSwitch;
```

Figure 13: Mosilab Model for *Constrained Pendulum* – State Chart Model with *Internal Events* (I-SE-P)

Mosilab E-SE-P Model. Mosilab's state chart construct is not only a good alternative to IF - or WHEN - clauses within one model, it offers also the possibility to switch between structural different models. This very powerful feature allows any kind of hybrid composition of models with different state spaces and of different type (from ODEs to PDEs, etc.). Table 9 shows a Mosilab implementation of the *Constrained Pendulum* making use of two different pendulum models, controlled externally by a state chart. Clearly, in case of this simple model, different models would not be necessary.

Here, the system is decomposed into two different models, `Short` pendulum model, and `Long` pendulum model, controlled by a state chart. The model description (Table 9) defines now first the two pendulum models, and then the event as before. The state chart creates first instances of both pendulum models during the initial state (`new`). The transitions organise the switching between the pendulums (`remove`, `add`). The `connect` statements are used for mapping local to global state.

Mosilab offers also strong support for simulator coupling (e.g. MATLAB) and time-synchronised coupling of external programs. This feature may be used for any kind of visualisation not based the model definition (VIS '(yes)').

External events driven by external states charts open possibilities, which were not planned at begin of Mosilab development, but which became obvious during development. It turned out, that complex experiments can be defined and performed by means of external state charts - as well as a simple parameter loop, which makes use of the same model in each state change (change of parameter value). Furthermore, at level of the 'main' model, any kind of static calculations due to Modelica standard should be possible. There, Mosilab mixes model frame and experimental frame and sets up a common extended environment (ENV 'yes'), where also frequency analysis can be implemented (FA '(no)').

```
model Long
equation
 mass*vdot/l1 + mass*g*sin(phi)+damping*v = 0;
end Long;
model Short
equation
 mass*vdot/ls + mass*g*sin(phi)+damping*v = 0;
end Short;
event discrete Boolean lengthen(start=true),
 shorten(start = false);
equation
 lengthen =
 (phi>phipin);shorten=(phi<=phipin);
statechart
state ChangePendulum extends State;
 State Short,Long,startState(isInitial=true);
transition startState -> Long action
 L:=new Long(); K:=new Short(); add(L);
end transition;
transition Long->Short event shorten action
 disconnect ….; remove(L); add(K); connect …
end transition;
transition Short -> Long event lengthen
 action
 disconnect …;
 remove(K); add(L);
 connect ……
end transition; end ChangePendulum;
```

Figure 14: Mosilab Model for *Constrained Pendulum* – State Chart Switching between Different Pendulums Models by *External Events* (E-SE-P)

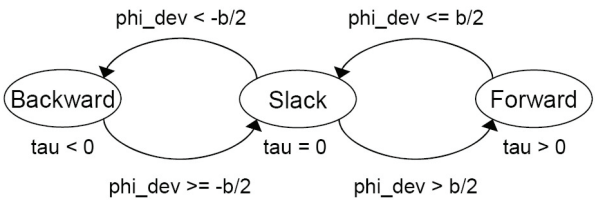
5.6 Open Modelica

The goal of the *Open* Modelica project is to create a complete Modelica modelling, compilation and simulation environment based on free software distributed in binary and source code form. The whole OpenModelica environment consists of open software: *OMC* – the Open Modelica Compiler translates Modelica models (with index reduction); *OMShell* as interactive session handler is a minimal experiment frame; Modelica models may be set up by a simple text editor or by a graphical model editor (here, for teaching purposes the model editor of MathModelica is allowed to be used!); the purpose of *OMNotebook* is to provide an advanced Modelica environment and teaching tool; the *DrModelica* notebook provides all the examples from P. Fritzson's book on Modelica; the other modules support environment interfacing and Open Modelica development.

Open Modelica is a generic Modelica simulator, so all basic features are met (ED, SEH, DAE, PM-T, PM-G, IR and MOD 'yes'; for DAE solving, variants of DASSL solver are used). P. Fritzson, the initiator of Open Modelica puts emphasis on discrete events and hybrid model-

ling, so documentation comes with clear advice for use of IF – and WHEN – clauses in Modelica, and with state chart modules in DrModelica – so SC-T gets 'yes'. Figure 15 shows the equivalence of a state chart and the correct definition as Modelica submodel. For graphical state chart modelling the experimental Modelica state chart library can be used – so SC-G '(yes)'.

The notebook features allow interfaces and extensions of any kind, e.g. for data visualisation and frequency analysis – FA and VIS '(yes)'; they allow also for controlled executive of different models, so that hybrid decomposition of structural dynamic systems is possible – SD '(yes)'.




```
partial model SimpleBacklash
Boolean backward, slack, forward;
parameter ……
equation
phi_dev = phi_rel - phi_rel0;
 backward = phi_rel < -b/2;
 forward = phi_rel > b/2;
 slack = not (backward or forward);
 tau = if forward then
             c*(phi_dev - b/2)
       else (if backward then
               c*(phi_dev + b/2)
             else 0);
end SimpleBacklash
```

Figure 15: OpenModelica State Chart Modelling

5.7 SimulationX

SimulationX is a new Modelica simulator developed by ITI simulation, Dresden. This almost generic Modelica simulator is based on ITI's simulation system ITI-SIM, where the generic IT-SIM modelling frame has been replaced by Modelica modelling. From the very beginning on, ITI-SIM concentrated on physical modelling, with a theoretical background from power graphs and bond graphs. Figure 16 shows graphical physical modelling in ITI-SIM – very similar to Modelica graphical modelling.

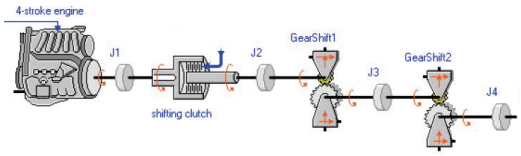


Figure 16: Physical Modelling in ITI-SIM / SimulationX

The simulation engine from ITI-SIM drives also SimulationX, using a sophisticated implicit integration scheme, with state event handling. Consequently, all features f related to physical modelling are available: (ED, SEH, DAE, PM-T, PM-G, and MOD 'yes'; index reduction is not really implemented – IR '(no)'. State chart constructs are not directly supported (SC-T 'no'), but due to Modelica compatibility the Modelica state chart library can be used (SC-G – '(yes)'. SimulationX (and ITI-SIM) put emphasis on physical application – oriented modelling and simulation, so frequency analysis is directly supported in the simulation environment (FA – 'yes'), which offers via additional modules (e.g. interfaces to multibody systems) connectivity to external systems (ENV – '(yes)'). The simulation engine drives also pseudo-3D visualisation (VIS – 'yes').

5.8 AnyLogic

AnyLogic – already discussed in a previous section) is based on hybrid automata (SC-T and SC-G - 'yes'). Consequently, hybrid decomposition and control by external events is possible (ED, SD 'yes'). AnyLogic can deal partly with implicit systems (only nested approach,

DAE ‘(yes)’), but does not support a-causal modelling (PM-T, PM-G - ‘no’) and does not support Modelica (MOD - ‘no’). Furthermore, new versions of AnyLogic concentrate more on discrete modelling and modelling with System Dynamics, whereby state event detection has been sorted out (SEH ‘(no)’. On the other hand, AnyLogic offers many other modelling paradigms, as System Dynamics, Agent-based Simulation, DEVS modelling and simulation. AnyLogic is Java-based and provides simulation-driven visualisation and animation of model objects (VIS ‘yes’) and can also generate Java web applets.

In AnyLogic, various implementations for the *Constrained Pendulum* are possible. A classical implementation is given in Figure 8, following classical textual ODE modelling, whereby instead of IF – THEN clauses a state chart is used for switching (I-SE-P, I-SE-S).

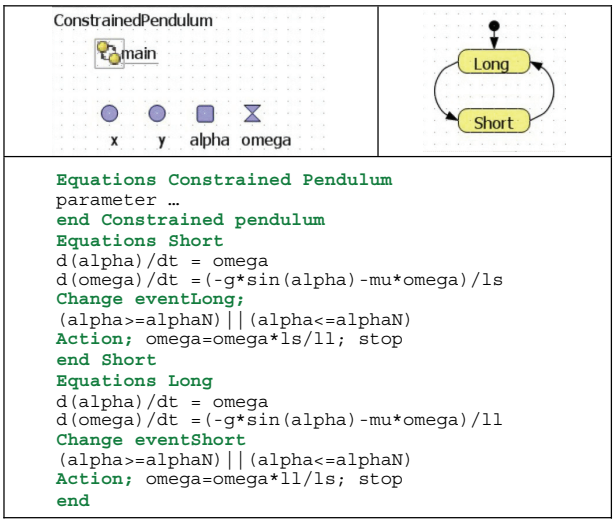


Figure 17: AnyLogic Model for *Constrained Pendulum*, Hybrid Model Decomposition with two Pendulum Models and *External Events*

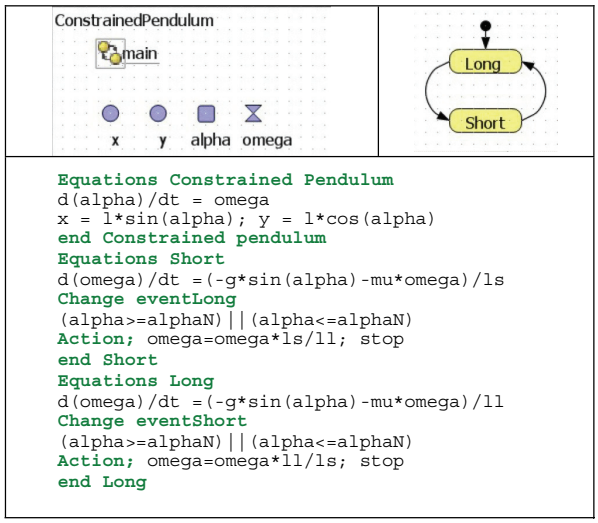


Figure 18 AnyLogic Model for *Constrained Pendulum*, Hybrid Model Decomposition with Two Models for Angular Velocity and Parallel Angle Model

AnyLogic E-SE-P Model with State Charts. A hybrid decomposed model may make use of two different models, each defined in substate / submodel `Short` and `Long`. – both part of a state chart switching between these submodels. The events defined at the arcs stop the actual model, set new initial conditions and start the alternative model (Figure 17).

AnyLogic E-SE-P Model with Parallel Models. AnyLogic works interpretatively, after each external event state equations are tracked and sorted anew for the new state space. This makes it possible, to decompose model not only in serial, but also in parallel. In *Constrained Pendulum* example, the ODE for the angle, which is not effected by the events, may be put in the main model, together with transformation to Cartesian coordinates (Figure 18), which seems to run in parallel with different velocity equations.

From software engineering view, AnyLogic is a programming environment for Java, with special features for ODE simulation. At each level Java code can be entered, and Java modules linked and called. The main module may be arbitrarily extended by Java code, stating not only the (predefined) simulation engine, but also frequency analysis packages, etc., with programming effort – so ENV ‘(yes)’

5.9 Model Vision Studium MVS

Model Vision Studium (MVS) – is an integrated graphical environment for modelling and simulation of complex dynamical systems. Development of MVS started in the 1990ies at Technical University of St. Petersburg. Basis of MVS are hybrid state charts (SC-T, SC-G - 'yes'), allowing any parallel, serial, and conditional combination of continuous models, described by DAEs, and controlled and interrupted by state events (ED, SHE - 'yes'). State models itself are objects to be instantiated in various kinds, so that structural dynamic systems can be modelled (SD - 'yes'). Textual physical and DAE modelling is supported by a mathematical formula editor (DAE and PM-T 'yes', PM-G no), but no Modelica compatibility (MOD – 'no'). For MVS, a subset of *UML Real Time* was chosen and extended to state chart activities (Java – based). Other modules (simulation kernel, environment) are linked modules (e.g. C-modules, Java-based simulation driven visualisation (VIS - 'yes'). In principle, MVS and AnyLogic have been developed in parallel. The continuous elements in AnyLogic were taken from MVS. State charts are similar to AnyLogic, with different implicit state space descriptions – and defining complex experiments (calling different models; ENV – 'yes), no frequency analysis (FA – 'no').

5.10 SCILAB / SCICOS

Scilab is a scientific software package for numerical computations with a powerful open computing environment for engineering and scientific applications. Scilab is open source software. *Scicos* is a graphical dynamical system modeller and simulator toolbox included in Scilab. Scilab / Scicos is an open source alternative to MATLAB / Simulink, developed in France. Consequently, Scilab as MATLAB – like tool has nearly the same features than MATLAB: no equation sorting– MS – 'no'!; DE, IR, PM-T, PM-G, MOD, SC-T, and SC-G – 'no'; SEH, DAE, and VIS – '(yes)', remarkably – SD, FA and ENV – 'yes'. Similarly, Scicos has extended features ED, SEH, and DAE – 'yes'.

The developers of Scicos started early with a kind of physical modelling – e. g. electrical modelling palette of Scicos (PM-T, PM-G – yes). They are working on extensions in two directions: extending the model description by full Modelica models (textually and graphically) – - so MOD and IR '(yes)' (subset), and refining the IF-THEN-ELSE – and WHEN – clause introducing different classes of associated events, resulting 'state chart clauses' - so SC-T – 'yes'

In Scicos, the Modelica state chart library allows graphical state chart modelling. Standalone Scicos has no features for frequency analysis, structural decomposition and extended environment (FA, SD, ENV – 'no'), but limited visualisation (VIS – '(yes)'); Scicos controlled by Scilab has all these features (VIS, FA, SD, ENV – 'yes').

5.11 Maple

Maple – developed by Maplesoft, Canada, is developing a toolbox *MapleSim*, which will understand Modelica models (PM-T, PM-G, and MOD – 'yes'). Maple acts as environment and provides sophisticated DAE solvers and index reduction (DAE, IR, ENV, VIS, FA – 'yes'). In development are constructs for events and event handling (ED – 'yes)', SEH – '(no)'); state chart modelling has not been discussed yet (SC-T – '(no)', SC-G – '(yes)').

5.12 Availability of Structural Features

Table 10 provides an availability comparison of the discussed features within the presented simulators. Clearly such comparison must be incomplete, and using simple 'yes' and 'no' might be too simple. Consequently, it should be a hint for further detailed feature comparison.

Literatur

[1] F. Breitenecker F., and I. Troch. 2004. 'Simulation Software – Development and Trends'. In *Modelling and Simulation of Dynamic Systems / Control Systems, Robotics, and Automation.* H. Unbehauen, I. Troch, and F. Breitenecker (Eds.). Encyclopedia of Life Support Systems (EOLSS), UNESCO, Eolss Publishers, Oxford ,UK, www.eolss.net.

[2] Cellier, F.E. (1991). *Continuous System Modeling*. Springer, New York.

[3] Cellier, F.E., and E. Kofman. 2006. *Continuous System Simulation*. Springer, New York.

[4] Fritzson, P. 2005. Principles of Object-Oriented Modeling and Simulation with Modelica. Wiley IEEE Press.

[5] Fritzson, P., F.E.Cellier, C. Nytsch-Geusen, D. Broman, and M. Cebulla, Eds. 2007. *EOOLT'2007 - Proc. 1st Intl. Workshop on Equation-based Object-oriented Languages and Tools*. TU Berlin Forschungsberichte, Vol. 2007-11.

[6] Nytsch-Geusen C., and P. Schwarz. 2005. 'MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics'. In *Proc. 4th Intern. Modelica Conference,* G. Schmitz (Ed.), Modelica Association - www.modelica.org, 527 – 535.

[7] Strauss J. C. 1967. 'The SCi continuous system simulation language (CSSL)'. Simulation 9, SCS Publ. 281-303.

	MS - Model Sorting	ED -Event Description	SEH -State Event Handling	DAE - DAE Solver	IR - Index Reduction	PM-T - Physical Modelling -Text	PM-G - Physical Modelling -Graphics	VIS – 'Onlie' - Visualisation	MOD – Modelica Modelling	SC-T – State Chart – Modelling - Text	SC-G – State Chart Modelling - Graphics	SD – Structural Dynamic Systems	FA – Frequency Analysis	ENV – Extended Environment
MATLAB	no	no	(yes)	(yes)	no	no	no	(yes)	no	no	no	yes	yes	yes
Simulink	yes	(yes)	(yes)	(yes)	no	no	(no)	(yes)	no	no	no	no	yes	(yes)
MATLAB / Simulink	yes	yes	yes	(yes)	no	no	(no)	(yes)	no	no	no	yes	yes	yes
Simulink / Stateflow	yes	yes	yes	(yes)	no	no	(no)	(yes)	no	(yes)	yes	no	yes	(yes)
ACSL	yes	yes	yes	yes	no	no	(no)	(yes)	no	no	no	no	yes	yes
Dymola	yes	yes	yes	yes	yes	yes	yes	yes	yes	(yes)	(yes)	no	(no)	(yes)
MathModelica	yes	yes	yes	yes	yes	yes	yes	(yes)	yes	(no)	(yes)	no	(no)	(no)
MathModelica / Mathematica	yes	yes	yes	yes	yes	yes	yes	yes	yes	(no)	(yes)	yes	yes	yes
Mosilab	yes	yes	yes	yes	(no)	yes	yes	(no)	(yes)	yes	yes	yes	no	(yes)
Open Modelica	yes	yes	yes	yes	yes	yes	(no)	(no)	yes	(no)	(yes)	no	no	no
SimulationX	yes	yes	yes	yes	yes	yes	yes	yes	yes	(no)	(yes)	no	yes	(yes)
AnyLogic	yes	yes	(yes)	(yes)	no	no	no	yes	no	yes	yes	yes	no	no
Model Vision	yes	yes	yes	yes	yes	yes	no	yes	no	yes	yes	yes	yes	no
Scilab	no	no	(yes)	(yes)	no	no	no	(yes)	no	no	no	yes	yes	yes
Scicos	yes	(yes)	yes	yes	(yes)	yes	yes	(yes)	(yes)	yes	(yes)	no	no	no
Scilab/ Scicos	yes	yes	yes	yes	(yes)	yes	yes	(yes)	(yes)	yes	(yes)	yes	yes	yes
(MapleSim)	yes	(yes)	(yes)	yes	yes	yes	yes	yes	yes	no	no	(yes)	(yes)	yes

Table 2: Availability of Structural Features in Simulators - DAEs, State Events, Modelica Notation, Structural Decomposition, and Related Features

Gradientenfreie dynamische Optimierung von Modelica-Modellen

Marcos Bockholt, Jürgen Köhler
Institut für Thermodynamik, TU-Braunschweig
m.bockholt, j.koehler@tu-bs.de
Wilhelm Tegethoff, TLK-Thermo-GmbH, Braunschweig
w.tegethoff@tlk-thermo.de

Zusammenfassung

Die Fortschritte in der Mikroprozessorenentwicklung und in der Simulationstechnik ermöglichen zunehmend die Online-Berechnung von Modellen der Regelstrecke in den Steuereinheiten selbst. Diese Modelle können beispielsweise in einer prädiktiven Regelungsstrategie (Model Predictive Control-MPC) eingesetzt werden, um optimierte Aktorensignale durch Optimierungsalgorithmen zu bestimmen. Bevor die entwickelte Strategie Online angewandt wird ist es üblich, diese Offline mit einem Streckenmodell zu testen (Model in the Loop-MIL). Ziel dieser Arbeit ist es, eine am Institut für Thermodynamik der TU-Braunschweig entwickelte externe Bibliothek (Dynamic Optimization Library-DOLI) für die Offline-Untersuchung von dynamischen Optimierungsstrategien mit Modelica-Modellen vorzustellen.

1 Einführung

1.1 Simulation thermischer Systeme mit Modelica und TIL

Die objektorientierte Modellierungssprache Modelica [9] wird für die Modellierung komplexer Systeme von der Industrie und den Forschungseinrichtungen zunehmend eingesetzt, vgl. [5] für einen Überblick. Die Entkopplung der Lösungsalgorithmen von der Modellerstellung vereinfacht die Austauschbarkeit von Teilmodellen und beschleunigt die Entwicklung von komponentenbasierten Simulationsbibliotheken. Die Modelica-Bibliothek TIL (TLK-Thermo-GmbH and Institut für Thermodynamik Library) wird von der TLK-Thermo-GmbH und dem Institut für Thermodynamik der TU-Braunschweig für die Modellierung thermodynamischer Systeme entwickelt [12] und stellt die Basis für die physikalische Beschreibung der zu optimierenden Systeme für die hier vorgestellte Optimierungsbibliothek (alias: DOLI) dar. Abbildung 1 zeigt beispielsweise einen mit TIL-Komponenten implementierten Luftkreislauf für den mobilen Einsatz.

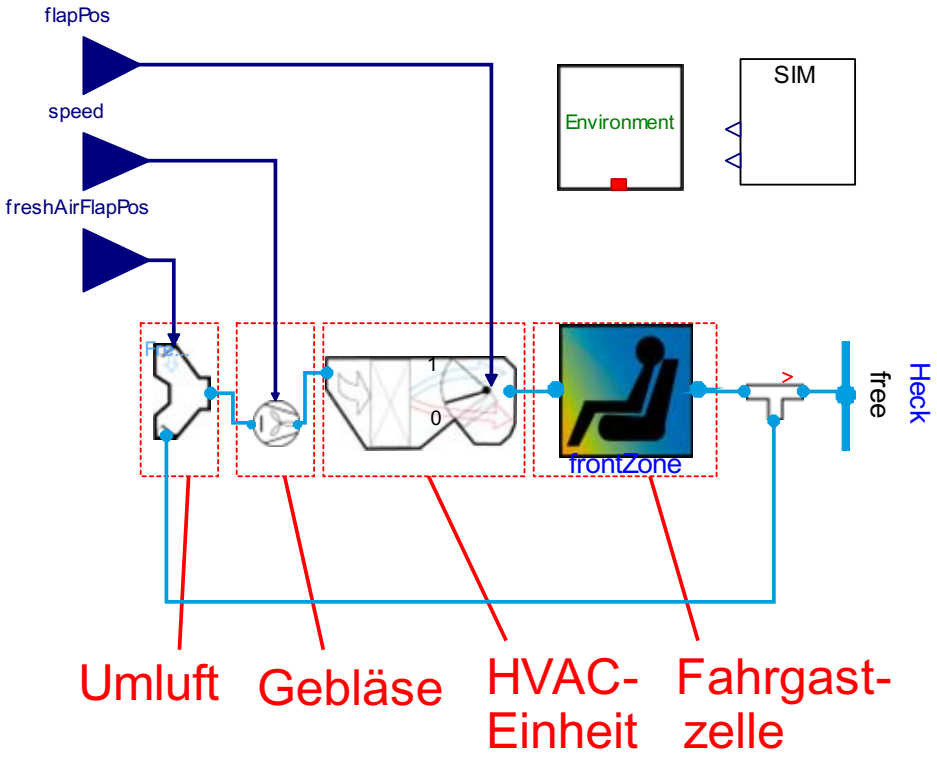


Abbildung 1: TIL-Modell des Gebläses, der HVAC-Einheit und der Fahrgastzelle. Die Steuergrößen Klappen-Einstellwinkel `flapPos` und Gebläsedrehzahl `speed` können von DOLI optimal berechnet werden.

1.2 Mathematische Formulierung der dynamischen Optimierung

Die klassische Formulierung des dynamischen Optimierungsproblems wird in Gleichung 1 vorgestellt [6]. In Gleichung 2 wird die Zustandsdifferentialgleichung des Streckenmodells beschrieben.

$$\begin{aligned}
\min_{\mathbf{u}(t)} J(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t) &= h\left(\mathbf{x}(t_e)\right) + \int_{t_0}^{t_e} L(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t) dt, \quad (1) \\
\textit{so dass} \quad \dot{\mathbf{x}}(t) &= \mathbf{f}\left(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t\right), \quad (2) \\
0 &= \mathbf{g}\left(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t\right), \\
\mathbf{u}_{min} \leq \mathbf{u}(t) &\leq \mathbf{u}_{max}, \\
\mathbf{x}_{min} \leq \mathbf{x}(t) &\leq \mathbf{x}_{max}.
\end{aligned}$$

Das Gütemaß J setzt sich aus dem Endzustandsgütemaß $h(\mathbf{x}(t_e))$ (Mayersches Gütemaß) und aus dem Übergangsverhaltensgütemaß $\int_{t_0}^{t_e} L(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t) \cdot dt$ (Lagrangesches Gütemaß) zusammen, wobei $\mathbf{x} \in \mathbb{R}^{n_x}$ und $\mathbf{y} \in \mathbb{R}^{n_y}$ die Zustandsvariablen bzw. die algebraischen Variablen sind. Die zeitinvarianten Systemparameter sind in $\mathbf{p} \in \mathbb{R}^{n_p}$ repräsentiert. Eine Steuerung $\mathbf{u}(t)$, die die Gleichung 1 erfüllt, wird als optimale Steuerung bezeichnet. Zusätzlich treten regelmäßig noch sogenannte Steuer- und Zustandsbeschränkungen ($\mathbf{u}_{min}$, $\mathbf{u}_{max}$, $\mathbf{x}_{min}$ und $\mathbf{x}_{max}$) auf, d.h. die Steuer- und Zustandsgrößen sind zusätzlich gewissen technischen Restriktionen unterworfen. Ziel der Bibliothek DOLI ist die Lösung von dynamischen Optimierungsproblemen, wie in Gleichung 1 aufge-

stellt, unter der Bedingung der Systemdifferentialgleichungen aus Gleichung 2, die aus der Modelica-Modell-Bibliothek TIL stammen. Eine wichtige Eigenschaft für die Effizienz bei der numerischen Lösung des Optimierungsproblems aus Gleichung 1, ist die Verwendung von Gradienteninformationen der Zielfunktional nach den Optimierungsvariablen, die sogenannten Sensitivitätsgleichungen. In Fällen, in denen kein Gradient der Zielfunktion vorhanden ist, oder die Zielfunktion nicht stetig differenzierbar ist, empfiehlt sich der Einsatz von gradientenfreien Methoden [8]. Dieses ist meistens der Fall bei der Optimierung mit gekoppelten Systemen (Co-Simulationen) oder mit komplexen Stoffdatenbibliotheken wie TILFluids, vgl. [12]. DOLI benötigt keine Sensitivitätsgleichungen wie im nächsten Abschnitt näher erläutert wird.

2 Kopplung zwischen dem Simulationsmodell und dem Optimierungstool

Zahlreiche Strategien für die Kopplung zwischen einem Modelica-Modell und einem Optimierungstool sind praktikabel. Franke [8] hat den Optimierungsalgorithmus HQP (Huge Quadratic Programming) mit dem Simulationsmodell eines Warmwasseraufbereitungssystems mit saisonaler Wärmespeicherung gekoppelt, um eine optimale Steuerungsstrategie zu erzielen. Für die Lösung mit Hilfe eines SQP (Sequential Quadratic Programming) Algorithmus benötigte er die Sensitivitätsgleichungen, d.h. Ableitungen der Zielfunktion zu den parametrisierten Steuerungssignalen ($d\mathbf{J}/d\mathbf{u}_p$). Åkesson führt in [1] den Optimica-Kompiler ein. Dieser ist im Wesentlichen ein Parser für die Übertragung von in Modelica definierten dynamischen Systemen in die Modellierungssprache für Optimierung AMPL (A Modelling Language for Mathematical Programming) [7]. Die Ansätze von Franke und Åkesson benötigen beide die Sensitivitätsgleichung aus dem Modelica-C-Code, der von der Simulationsumgebung abhängig ist und die Kopplung mit einem Standard-SQP-Algorithmus aufwendig macht.

Die verfolgte Strategie bei der Entwicklung von DOLI ist die Anwendung von Optimierungsalgorithmen, die keine Informationen über die Sensitivitätsgleichungen benötigen, die so genannten gradientenfreien Algorithmen (Derivative Free oder Direct Search Algorithms). Diese vereinfachen die Kopplung mit dem Simulationsmodell erheblich und sind unabhängig von der Simulationsumgebung, solange eine Schnittstelle zur Steuerung des Modell-Standalone vorhanden ist. Abbildung 2 zeigt die objektorientierte Struktur von DOLI mit dem bereits integrierten Nelder-Mead-Simplex-Verfahren (aus [10]) und dem Trust-Region-Algorithmus CONDOR [2]. Die Bibliothek kann mit einem Modelica-Streckenmodell in dem Modelica-Layer für die Offline-Untersuchung von Optimierungsstrategien gekoppelt werden. Der Informationsaustausch passiert in diesem Layer durch das Modelica-Standardpaket `Blocks.Interfaces`. Des Weiteren wird in dem C/C++-Layer das Streckenmodell als Stand-alone, wie beispielsweise `dymosim.exe` der Fa. Dynasim AB [4], für die Berechnung der Zielfunktional aus Gleichung 1 eingebettet und mit dem Optimierungsalgorithmus gekoppelt. In dem C/C++-Layer findet der Informa-

tionsaustausch mit dem Modelica-Standalone über Input- und Outputdateien statt. Im Gegensatz zu dem schon integrierten Optimierungstool von Dymola [4], bietet DOLI eine komfortable Schnittstelle für die Durchführung von dynamischer Offline-Optimierung, wie zum Beispiel die Parametrisierung der Steuersignale in stückweise konstante Segmente, linear-interpoliert oder als Lagrange-Polynome in dem Steuerintervall.

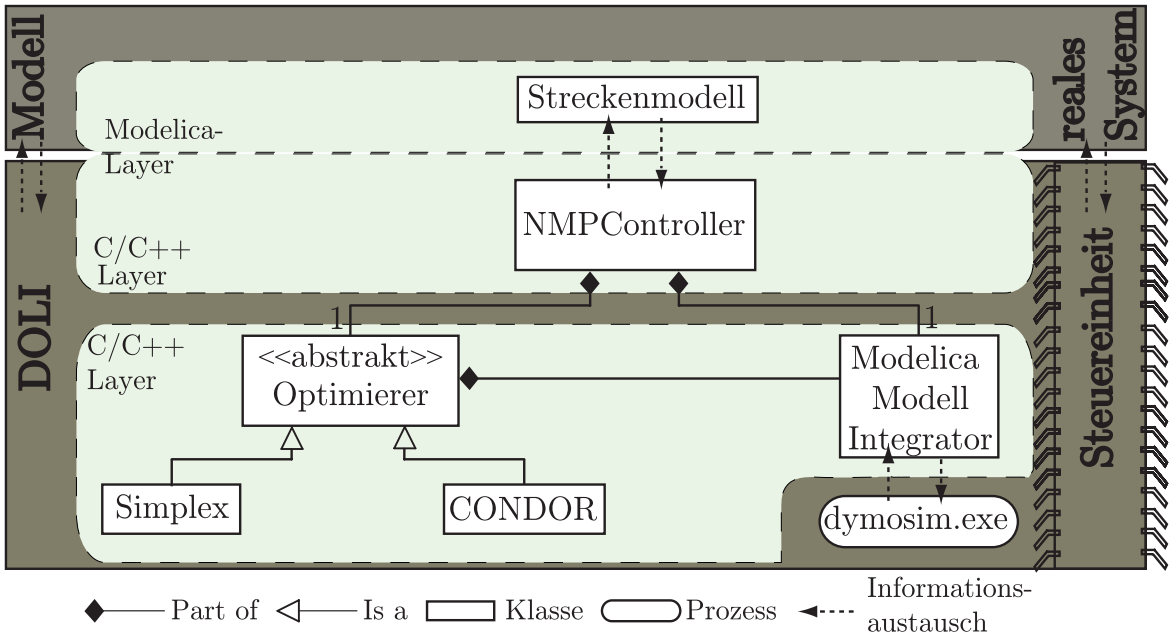


Abbildung 2: Exemplarische Darstellung einer MIL-Anwendung mit DOLI anhand eines UML-Klassendiagramms

3 Optimale Steuerung eines thermodynamischen Modells

Als Testproblem für die Bibliothek wird das Abkühlen einer adiabaten Box mit idealer Luftvermischung eingesetzt, siehe Abbildung 3. Die Differentialgleichung für das System

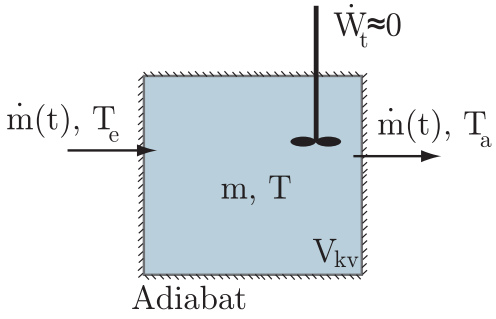


Abbildung 3: Thermodynamisches Testmodell

aus der Abbildung wird aus dem ersten Hauptsatz der Thermodynamik hergeleitet und ist in Gleichung 3 aufgestellt.

$$m \cdot c_v \cdot \frac{dT(t)}{dt} = \dot{m}(t) \cdot c_p \cdot (T_e - T_a(t)) \tag{3}$$

Die Lufteintrittstemperatur T_e ist konstant und gleich der Soll-Temperatur $T_e = T_{soll}$. Durch die ideale Vermischung gleicht sich die Luftaustrittstemperatur der Temperatur der Luft in der Box an und es gilt $T_a(t) = T(t)$. Die Dichte ρ und die spezifischen Wärmekapazitäten c_p und c_v der Luft in der Box sind näherungsweise konstant gehalten. Ein Vergleich mit Gleichung 2 liefert $\mathbf{x}^T = \{T\}$, $\mathbf{u}^T = \{\dot{m}\}$ und $\mathbf{p}^T = \{c_v, c_p, \rho, V_{kv}\}$.

3.1 Analytische Lösung

Das Gütemaß für das Testproblem kann als Kombination eines verlaufs- und eines verbrauchsoptimalen Kriteriums in Gleichung 4 aufgestellt werden. Die Gewichte w_1 und w_2 dienen der Normierung und Priorisierung der Kriterien. Ziel ist eine schnellere Abkühlung der Box mit einem minimalen Energieaufwand. Dieser entspricht in dem Testproblem aus Gleichung 3 dem Massenstrom eines Gebläses $\dot{m}(t)$.

$$\min_{\dot{m}(t)} J(T, \dot{m}(t), t) = \frac{1}{2} \cdot \int_{t_0}^{t_e} \left[w_1 \cdot (T_{soll} - T(t))^2 + w_2 \cdot (\dot{m}_{min} - \dot{m}(t))^2 \right] \cdot dt \quad (4)$$

Für die Lösung des optimalen Steuerungsproblems wird die Hamilton-Gleichung angewandt (siehe [6] für weitere Details):

$$\begin{aligned} H(T, \dot{m}(t), \lambda, t) = & \frac{1}{2} \cdot \left[w_1 \cdot (T_{soll} - T(t))^2 + w_2 \cdot (\dot{m}_{min} - \dot{m}(t))^2 \right] + \\ & + \lambda \left[\frac{\dot{m}(t)}{m} \cdot \frac{c_p}{c_v} \cdot (T_{soll} - T(t)) \right] \end{aligned} \quad (5)$$

Die adjungierte Variable λ, die dem Lagrange-Multiplikator des Extremalproblems in der Differentialrechnung entspricht, ist in Gleichung 6 definiert.

$$\begin{aligned} \frac{d\lambda}{dt} &= -\frac{\partial H}{\partial T} \\ \frac{d\lambda}{dt} &= w_1 \cdot (T_{soll} - T) + \lambda(t) \cdot \frac{\dot{m}(t)}{m} \cdot \frac{c_p}{c_v}. \end{aligned} \quad (6)$$

Mit Hilfe der notwendigen Optimalitätsbedingung:

$$\frac{\partial H}{\partial \dot{m}} = 0, \quad (7)$$

erhält man für das Testproblem den optimalen Gebläsemassenstrom:

$$\dot{m}^*(t) = \dot{m}_{min} - \frac{\lambda}{w_2 \cdot m} \cdot \frac{c_p}{c_v} \cdot (T_{soll} - T(t)). \quad (8)$$

Durch numerische Integration der Gleichungen 3, 6 und 8 kann das Problem gelöst werden.

3.2 Numerische Lösung

Für die numerische Lösung ist die Differentialgleichung des Testproblems, Gleichung 3, in Modelica implementiert, siehe Codeauflistung 1. Das Gütemaß ist in Zeile 16 aufgestellt

und dient als Zielfunktional für DOLI. Die Steuergröße ist der Massenstrom des Eingangskonnektors (`inPort.m_flow`) und wurde in 30 stückweise konstante Segmente innerhalb des Steuerintervals diskretisiert.

```
1  model Box
2   ... (Definition von Parameter, Konstanten und Startwerten)
3    SI.Temperature T(start=Tstart) "Zustandsvariable";
4     Real cost "Gütemaß";
5     Real w1,w2 "Gewichtung";
6    Connectors.Port inPort, outPort "Konnektoren";
7  equation
8  /********Massenbilanz***********/
9  inPort.m_flow+outPort.m_flow=0;
10 /********Energiebilanz**********/
11 rhoAir*V*cv*der(T)=inPort.m_flow*cp*(inPort.T-T);
12 outPort.T=T;
13 /********Gütemaß***************/
14 w1=1.5/(Tstart-Tsoll)^2;
15 w2=0.5/(mflowMin-mflowMax)^2;
16 der(cost)=0.5*(w1*(Tsoll-T)^2+w2*(mflowMin-inPort.m_flow)^2);
17 end Box;
```

Codeauflistung 1: Modelica-Code des thermodynamischen Problems

Die Ergebnisse der analytischen und der numerischen Lösung sind in Abbildung 4 dargestellt. Die adiabate Box mit einem internen Volumen von 1 m^3 wird von 40°C auf 20°C in 40 s optimal abgekühlt.

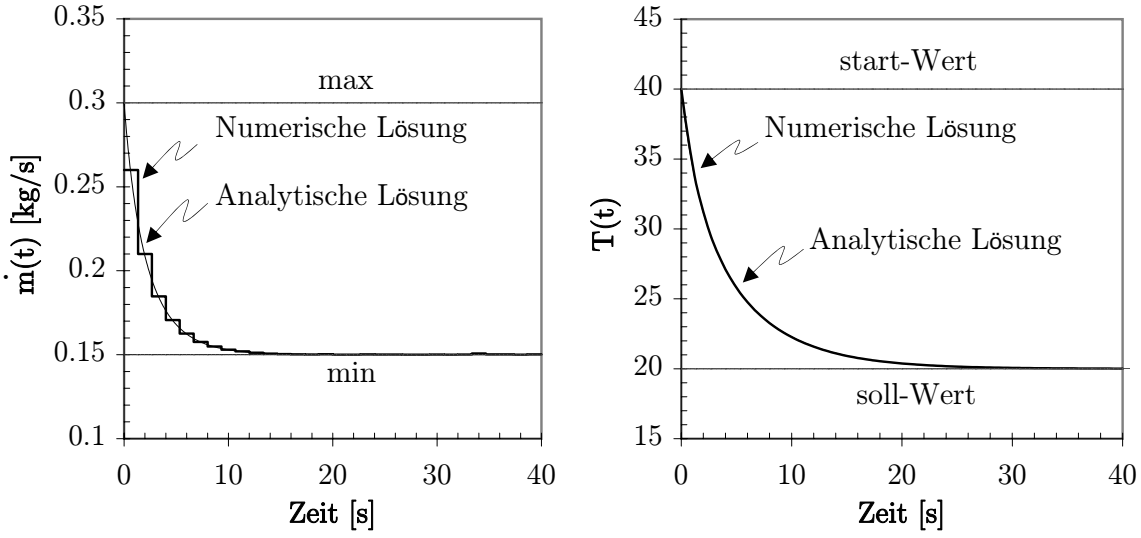


Abbildung 4: Vergleich zwischen der analytischen und der numerischen Lösung

Schließlich wird das Gewicht w_1 für das verlaufsoptimale Kriterium aus Gleichung 4 variiert, um den optimalen Massenstromverlauf zu untersuchen. In Abbildung 5 sind die Ergebnisse der Gewichtsvariation dargestellt. Zuerst wird der Wert $0.33w_1$, vgl. Zeile 14 aus der Codeauflistung 1, gesetzt, d.h. ein Optimierungsproblem mit einem dominanten

verbrauchsoptimalen Kriterium. In den folgenden Schritten werden Optimierungen mit den Werten w_1 und $1.33w_1$ durchgeführt. Wie erwartet, konvergiert bei $1.33w_1$ die Lösung zu dem maximal möglichen Massenstromverlauf, denn das verbrauchsoptimale Kriterium w_2 wird ausgeschaltet.

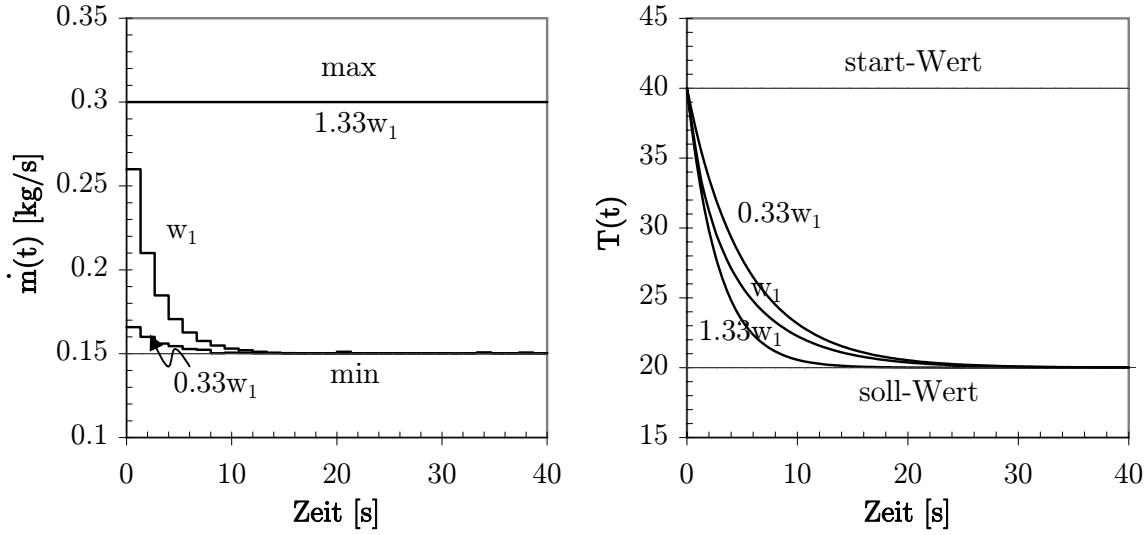


Abbildung 5: Gewichtsvariation des dynamischen Optimierungsproblems aus Gleichung 4

4 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Bibliothek für dynamische Optimierung DOLI (Dynamical Optimization Library) für die Offline-Optimierung von Modelica-Modellen vorgestellt. Die Besonderheit dieser Bibliothek ist die Anwendung von deterministischen gradientenfreien Optimierungsalgorithmen, welche sich für komplexe Optimierungsprobleme niedriger Dimension gut eignen. Der erste Schritt für die Offline-Untersuchung von dynamischen Optimierungstrategien mit komplexen innovativen Komponenten aus der Bibliothek für thermodynamische Systeme TIL ist erzielt. Es bleibt zu untersuchen, ob diese gradientenfreien Verfahren in einer Online-Anwendung, wie etwa in einem prädiktiven Regler (Nonlinear Model Predictive Control), umsetzbar sind.

Literatur

[1] Åkesson, J.: *Tools and languages for optimization of large-scale Systems*, PhD-thesis, Department of Automatic Control, Lund University, Sweden, 2007.

[2] Berghen, F V.: *CONDOR: a constrained, non-linear, derivative-free parallel optimizer for continuous, high computing load, noisy objective functions*, Dissertation, Univerité Libre de Bruxelles, Belgium, 2004.

[3] Bockholt, M.: *Optimization methods for time consuming computer simulations*, Master-thesis, Institut für Wissenschaftliches Rechnen, TU-Braunschweig, 2004.

[4] Dynamsim AB: *Dynamic Modeling Laboratory*, url: http://www.dynasim.se/.

[5] Elmqvist, H., Mattsson, S. E., Otter, M.: *Object-oriented and hybrid modeling in Modelica*, Proceedings of the $4^{th.}$ International Conference on Automation of Mixed Processes ADPM, Dortmund, 2000.

[6] Föllinger, O.: *Optimale Regelung und Steuerung*, R. Oldenburg Verlag, 3. Auflage, München, 1994.

[7] Fourer, R., Gay, D. M., Kernighan, B. W.: *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press Brooks, Cole Publishing Company, 2002.

[8] Franke, R.: *Integrierte dynamische Modellierung und Optimierung von Systemen mit saisonaler Wärmespeicherung*, Dissertation, TU-Ilmenau, Deutschland, 1998.

[9] Modelica Association: *Modeling of complex physical systems*, url: http://www.modelica.org/.

[10] Press, W., Teukolsky, S. ; Vetterling, W., Flannery, B.: *Numerical Recipes in C++ - The art of scientific computing*, Cambridge University Press, 2002.

[11] Qin, S. J., Badgwell,T. A..: *A survey of industrial model predictive control technology*, Control Engineering Practice 11, S. 733 - 764, 2003.

[12] Richter, C. C.: *Proposal of new object-oriented equation-based model libraries for thermodynamic systems*, Dissertation, TU-Braunschweig, Institut für Thermodynamik, Deutschland, 2008.

Modellierung und Simulation eines konventionellen Steinkohleblocks mit Modelica

F.Gottelt, J.Nocke, E.Hassel; Lehrstuhl für Technische Thermodynamik, Rostock

friedrich.gottelt@uni-rostock.de

Zusammenfassung

Eine stabile und qualitätsgerechte Energieversorgung der Bundesrepublik Deutschland ist Grundvoraussetzung für eine nachhaltige Entwicklung der Volkswirtschaft. Dabei spielt die CO_2-freie Energieerzeugung in Onshore- und Offshore-Windkraftanlagen eine immer größere Rolle. Das herausragende Problem beim Einsatz von Windenergieanlagen (WEA) sind aber deren witterungsbedingte, unterschiedlich große und schnelle Leistungsfluktuationen. Der geplante Ausbau der Windenergiekapazitäten wird für bestehende und zukünftige Kraftwerke erhebliche Mehrbelastungen zur Folge haben. Für die Kraftwerksbetreiber ist es daher wichtig, einzuschätzen, in wieweit bestehende Kraftwerke dem zu erwartenden Energiemarkt gerecht werden können und welchen Rahmenbedingungen zukünftige Kraftwerke genügen müssen. Im Rahmen des VGB-Forschungsprojekts 283 wird dazu ein detailliertes Modell des Steinkohleblocks Rostock erstellt, um exemplarisch die Auswirkungen von erhöhter Windenergieeinspeisung und dadurch bedingter stark dynamischer Fahrweise auf Nutzungsgrad und Belastungen kritischer Bauteile vorherzusagen. Vorgestellt werden der derzeitige Stand der Modellierung und aktuelle Simulationsergebnisse.

1 Einleitung

Die Energieversorgung durch Windkraft ist umweltschonend und unabhängig vom Import fossiler Energieträger wie Kohle oder Gas. Diesen Vorteilen steht die unsichere Versorgungslage gegenüber. Die wetterbedingten Schwankungen der Windenergieeinspeisung erfordern eine deutlich dynamischere Fahrweise konventioneller Kraftwerke. Das Verdrängen von aus Kohle produziertem Strom vom Markt durch Windkraft ist bereits heute zu beobachten. Der geplante Ausbau der installierten Windenergieanlagenleistung von ca. 20 GW auf ca. 48 GW bis 2020 lässt eine deutliche Verschärfung dieses Effekts erwarten [4]. Kraftwerke, die heute vorwiegend in Grund- und Mittellast bei nur wenigen An- und Abfahrvorgängen betrieben werden, müssen in Zukunft vermehrt in Teillast - bei reduziertem Wirkungsgrad - laufen oder vorübergehend stillstehen. Dabei sind große Lastwechsel und Anfahrvorgänge insbesondere kritische Vorgänge für ein Kraftwerk da die auftretenden Materialspannungen vor allem dickwandige Bauteile (z.B. Sammler, Turbinenwellen)

ermüden. Erhöhung der An- bzw. Abfahrhäufigkeiten führt zu kürzeren Wartungsintervallen und größeren Instandhaltungskosten.
Innerhalb des Projekts werden die zu erwartenden Auswirkungen des geplanten WEA-Kapazitätsausbaus auf die Kraftwerkseinsatzplanung und auf die Bauteilbelastungen einzelner Kraftwerke detailliert untersucht. Dazu wird ein auf thermodynamischen Ansätzen beruhendes instationäres Modell des Kraftwerks erstellt, das zur Simulation der Lastwechselvorgänge auch die Leittechnik des Kraftwerks abbildet.

2 Thermodynamische Modellierung von Dampf-Kreisprozessen mit Modelica

Zur Umsetzung der gestellten Aufgabe ist die Programmiersprache Modelica [3] gewählt worden. Als Entwicklungsumgebung wird Dymola® genutzt. Die Vorteile dieser Kombination liegen in den bereitstehenden leistungsfähigen numerischen Lösungsverfahren, der übersichtlichen Strukturierung von vorhandenen Standardbibliotheken und der Quelloffenheit der Bibliotheken. Außerdem zeichnet sich Modelica-Code durch die Objektorientierung der Programmiersprache durch eine hohe Wiederverwendbarkeit aus. Als Basis für die Modellierung der kraftwerkstechnischen Komponenten dient die Bibliothek `ThermoPower` [1],[2].

2.1 Gesamtkreislauf

Den aktuellen Stand der Modellierung des Gesamtkreislaufs stellt Abb. 1 dar. Zu sehen sind neben den Turbinengruppen, dem Dampferzeuger, dem Speisewasserbehälter sowie Kondensat- und Speisewasserpumpe auch die Hochdruck- und Niederdruck-Vorwärmstrecke, die jeweils durch mehrere Anzapfungen der Turbogruppe beheizt wird. Ziel dieser aufwendigen Vorwärmung ist eine Verbesserung des Wirkungsgrads. Soll von der implementierten Leittechnik auf das physikalisch basierte Modell der Verfahrenstechnik zugegriffen werden, so muss das thermodynamische Modell auch ein realitätsnahes Verhalten (also insbesondere gleiche Austritts-Dampfparameter, gleiche abgegebene Leistung bei gleichem Brennstoffeinsatz) aufweisen. Dies ist der Grund für den angestrebten hohen Detaillierungsgrad. Denn das Blockleitsystem vergleicht zur korrigierenden Regelung Messwerte (aus dem thermodynamischen Modell gewonnen) und vorausberechnete Sollwerte, die von einem leittechnikinternen Modell, das aus regelungstechnischen Übertragungsfunktionen besteht, berechnet werden [4]. Die Parametrierung des leittechnikinternen Modells erfolgt anhand der Daten des existierenden Steinkohleblocks Rostock.

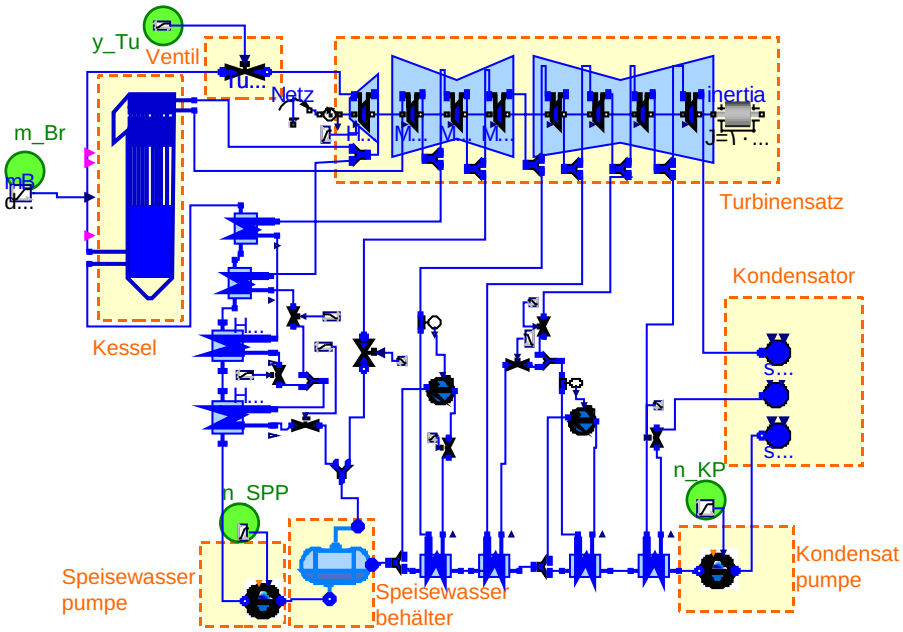


Abbildung 1: *Ansicht des Dampf-Kreislaufs in Dymola®*

Abbildung 2 zeigt exemplarisch die Auswirkungen eines Lastsprungs von 80% auf 100% Nennlast. Es sind dabei zwei Varianten der Vorwärmstrecke gegenübergestellt. Die Vorwärmer des ersten (blauen) Falls reagieren wesentlich schneller auf eine Beheizungsänderung als im zweiten (roten) Fall.

Dieses Ansprechverhalten lässt sich durch die Geometrie der Vorwärmer (energiespeichernde Massen, Strömungsführung) beeinflussen - ein großer, schwerer Wärmeübertrager speichert z.B. mehr Energie ein als ein leichter (dünnwandiger) Wärmeübertrager; die Energie des Anzapfdampfs wird nur zeit verzögert an das Speisewasser übertragen. Durch Nutzung von Anzapfklappen, die kurzzeitig erhöhte Anzapfmassenströme einstellen, kann ein schnelles Ansprechen der Vorwärmer gefördert werden.

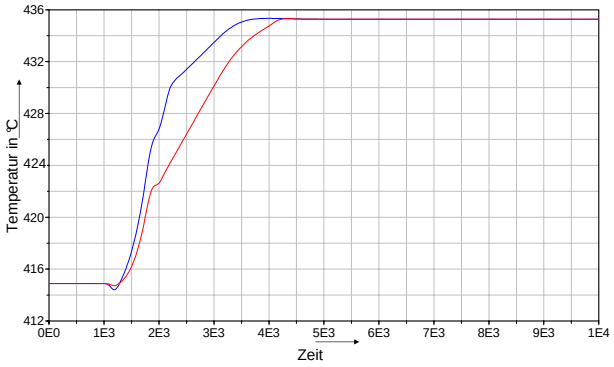


Abbildung 2: *Auswirkungen eines Lastsprungs auf die Verdampfer-Austritts-Temperaturen: bei schnell ansprechender und träge ansprechender Vorwärmstrecke*

Dabei wird durch schnelle Erhöhung des Beheizungsmassenstroms und damit der lokalen Geschwindigkeiten der Wärmeübergangskoeffizient (dampfseitig) erhöht. Dadurch steigt die Temperatur der entsprechenden Wärmeübertragerrohre rascher, es wir schneller Wärme an das Speisewasser abgegeben. Temporäre Temperaturspitzen des relativ kühlen Speisewassers werden zugunsten konstanter Dampfzustände des Frischdampfs in Kauf genommen. Insbesondere für dickwandige Bauteile, wie der Abscheideflasche, sind räumliche Temperaturgradienten zwischen Außen- und Innenfaser von Bedeutung und können die Laständerungs- und Anfahrgradienten begrenzen. Abbildung 3 zeigt die Gradienten in der Ab-

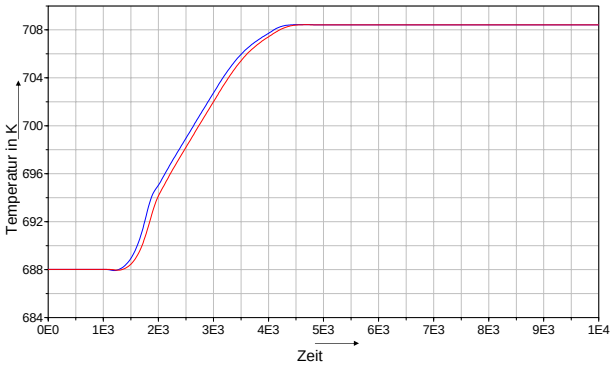


Abbildung 3: *Auswirkungen eines Lastsprungs auf die Innen- und Außen-Faser der Abscheideflasche*

scheideflasche für den Fall „träge Vorwärmstrecke“. Unter Annahme vernachlässigbarer Wärmeverluste an die Umgebung stellt sich im Stationärfall eine über die Rohrwand konstante Temperatur ein. Diese Vereinfachung führt auch im instationären Übergang zu relativ niedrigen Temperaturgradienten, weshalb Wärmeverluste in Zukunft detailliert mit betrachtet werden.
Da einige Modelle, die für die vollständige Abbildung des Rostocker Kraftwerks erforderlich sind, nicht oder nur stark vereinfacht in der `ThermoPower`-Bibliothek vorliegen, wurde diese Bibliothek erweitert. Zwei dieser Modelle sind im Folgenden exemplarisch vorgestellt.

2.2 Pumpe mit 3D-Kennfeld

Das bestehende Pumpen-Modell `Pump` der `ThermoPower`-Bibliothek extrapoliert von einer Kennlinie (Nenndrehzahl) unter Zuhilfenahme von Ähnlichkeitsgesetzen auf das gesamte Kennfeld. Da für das Zusammenspiel von Verfahrens- und Leittechnik die Kennfelder der Speisewasserpumpen und der Kondensatpumpe von hoher Genauigkeit erforderlich sind, wurde das bestehende Modell `Pump` erweitert. Dass reale Kennfelder von dieser idealisierten Vorstellungen deutlich abweichen können, zeigt Abbildung 4.

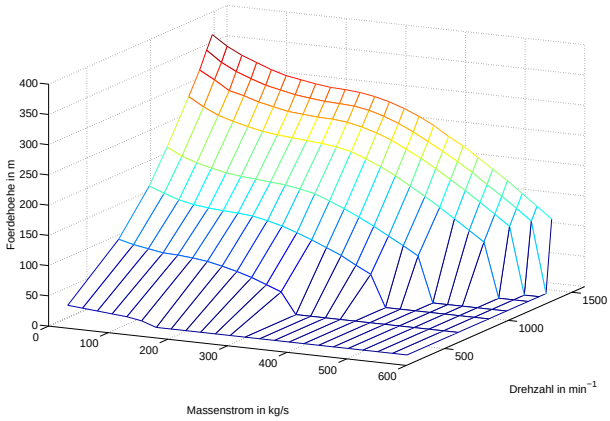


Abbildung 4: *3D-Kennfeld einer Kondensatpumpe*

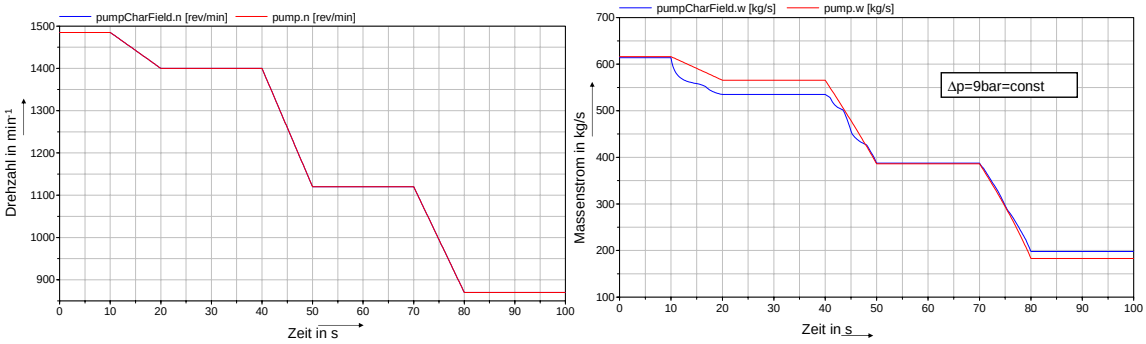


Abbildung 5: *Vergleich von* `Pump` *und* `PumpCharField`*: links - Verminderung der Pumpendrehzahl, rechts - geförderter Massenstrom bei* $\Delta p = const$

Dem neuen Modell `PumpCharField` kann das Pumpenkennfeld in Form einer Textdatei übergeben werden, dessen Syntax derjenigen für die Modelica-Standard-Klasse zum Einlesen von Daten aus Textdateinen (`Modelica.Blocks.Tables.CombiTable2D`) entspricht. In Abbildung 5 ist der Unterschied der beiden Modelle illustriert; beide Pumpenmodelle simulieren die Förderung von Wasser von einem niedrigen Druckniveau auf ein höheres Druckniveau. Dabei wird die Pumpendrehzahl schrittweise reduziert (links). Im rechten

Bild sind die Abweichungen der beiden Modelle anhand des Massenstroms zu erkennen: Es gibt offenbar Betriebspunkte, bei denen die extrapolierte Kennlinie und das reale Kennfeld gut übereinstimmen; weicht der Betriebspunkt aber von den gegebenen Stützstellen der Kennlinie ab, so werden erhebliche Unterschiede deutlich.

2.3 Speisewasserbehälter

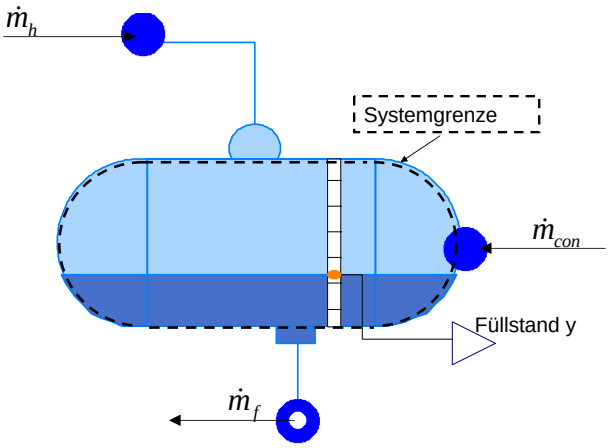


Abbildung 6: *Schematische Darstellung des Systems „Speisewasserbehälter“*

Der Speisewasserbehälter hat, neben der Entgasung des Kondensats, die Aufgabe der Mischvorwärmung und Speicherung des Speisewassers. Dabei entkoppelt der Behälter die Niederdruck-Vorwärmstrecke vom Hochdruckteil des Kreislaufs. So wird der Kessel während des Kondensatstaus aus der vorhandenen Vorlage des Speisewasserbehälters gespeist. Dabei werden die Anzapfklappen, die die ND-Vorwärmer speisen, geschlossen und der Kondensatstrom gedrosselt. Durch diese Maßnahme strömt kurzfristig mehr Dampf durch die Mitteldruck- und Niederdruckturbinen; die Leistung wird erhöht. Aber auch während langsamerer Lastwechsel wirkt der Speisewasserbehälter als Speicher, der so die Kondensatpumpe von der Speisewasserpumpe entkoppelt.
Das Modell des Speisewasserbehälters basiert auf der Massenbilanz und der Energiebilanz, die jeweils für das Gesamtsystem „Fluid ohne Wände“ (siehe Abb. 6) aufgestellt werden:

$$\frac{dM}{dt} = \dot{m}_{con} + \dot{m}_h - \dot{m}_f \tag{1}$$

$$\frac{dH}{dt} = \dot{m}_{con} h_{con} + \dot{m}_h h_h - \dot{m}_f h_f + V\frac{dp}{dt} \tag{2}$$

Hierin sind M, $\dot{m}$, h und p die Gesamtmasse im Behälter, Massenstrom, spezifische Enthalpie bzw. der Druck. H ist die Gesamtenthalpie des Systems, der Index h steht für „Beheizung (heating) durch Anzapfung“, *con* für „Kondensat (condensate)“ und f für „Speisewasser (feedwater)“. Dabei wird die Energiespeicherung in den Behälterwänden vernachlässigt. Als weitere Vereinfachung wird angenommen, dass die Flüssigkeit im Behälter vollständig aus siedender Flüssigkeit besteht und der Dampf gesättigt ist. Damit lassen sich die Randbedingungen definieren:

$$h_f = h'(p) \qquad p_h = p_s(T) \tag{3}$$

$$h_h = h''(p) \qquad p_{con} = p_s(T) \tag{4}$$

$$p_f = ps(T) + \rho' g y \tag{5}$$

Da die Speisewasserpumpe stets Wasser mit etwas höherem Druck als dem Sättigungs-

druck (vgl. Gl.5) ansaugt, ist sichergestellt, dass das Wasser leicht unterkühlt ist und an evtl. vorhandenen Einbauten (Drosseln) nicht verdampft. Speisewasser mit einem merklichen Dampfanteil würde zu deutlich reduziertem Massenstrom und ggf. einer Reduzierung des Wirkungsgrads führen - wie bei einer realen Pumpe im dampf- (oder luft-) ansaugenden Betrieb.
Zur Berechnung des Füllstands wird ein fiktiver Dampfanteil

$$x = M_v/(M_v + M_l) \tag{6}$$

im Behälter eingeführt, der von der Vorstellung ausgeht, dass flüssige und dampfförmige Phase im Behälter ideal gemischt sind. In dieser Vorstellung stellt der Quotient $h_{SWB} = H/M$ die mittlere Enthalpie im Behälter dar, mit:

$$x = \frac{h_{SWB} - h'}{h'' - h'} \tag{7}$$

Mit Gleichung 6 kann auf die flüssige Masse im Behälter und mit $V = M_l/\rho'$ auch auf das flüssige Volumen im Behälter geschlossen werden.
Das Volumen der Flüssigkeit V_l lässt sich bei bekanntem Füllstand y bei liegendem Behälter (Innenradius r_i, Länge L) berechnen nach:

$$V_l = Lr_i^2(acos(\frac{r_i - y}{r_i}) - (r_i - y)\sqrt{2r_iy - y^2}/r_i^2) \tag{8}$$

An dieser Stelle zeigt sich der Vorteil des gleichungsorientierten Ansatzes von Modelica: Es muss nicht zwangsläufig nach der unbekannten Größe aufgelöst werden wie bei einem Algorithmus. Das erhöht dabei auch die Wiederverwendbarkeit des Gleichungssystems. Sollte durch anderweitige Nutzung des Modells der Füllstand bekannt sein und das Volumen daraus berechnet werden, ist das Gleichungssystem Gl. 1 bis 8 gleichermaßen gültig.

In den Abbildungen 7 bis 9 ist zur Veranschaulichung der Charakteristik des Speisewasserbehälters die Simulation eines Kondensatstaus über fünf Minuten dargestellt. Dabei sinkt durch Verminderung der Kondensatpumpendrehzahl der Kondensatmassenstrom. Da der Druck im Speisewasserbehälter ansteigt, sinkt der an der Mitteldruckturbine abgezweigte Beheizungsmassenstrom ebenfalls und es strömt mehr Dampf durch Mitteldruck- und Niederdruck-Turbine. Die Speisewasserpumpe wird nicht zur Regelung des Speisewasserbehälterfüllstands genutzt, der aus dem Behälter abgeführte Massenstrom bleibt nahezu konstant.

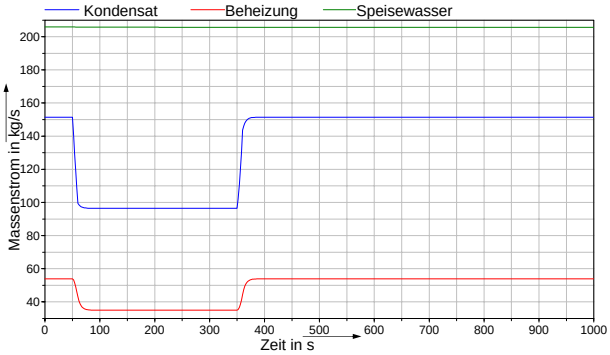


Abbildung 7: *Verlauf der in den Speisewasserbehälter ein- und ausgehenden Massenströme beim Kondensatstau*

Es wird dementsprechend während des Kondensatstaus mehr Wasser von den Speisewasserpumpen abgeführt als durch die Kondensatpumpe und die Anzapfungen der Turbine eingespeist wird. Der Füllstand im Behälter sinkt. Das Kondensat kann also nur eine begrenzte Zeit lang angestaut werden. Danach muss der Speisewasserbehälter wieder aufgefüllt und der Kondensatsammelbehälter (Hotwell) entleert werden.

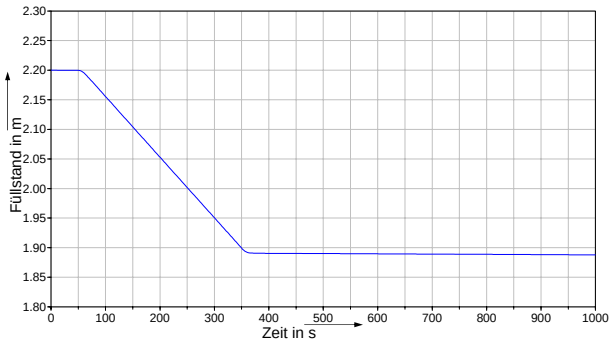


Abbildung 8: *Zeitlicher Verlauf des Speisewasserbehälter-Füllstands beim Kondensatstau*

Eine wichtige Funktion des Speisewasserbehälters ist, wie erwähnt, die Entkopplung von Hochdruck-Vorwärmer und Niederdruckvorwärmer. Während des Kondensatstaus steigt der Druck nur leicht an, so dass die Fördermenge der Speisewasserpumpe konstant bleibt und auch der energetische Zustand des Speisewassers nur geringen Schwankungen unterworfen wird.

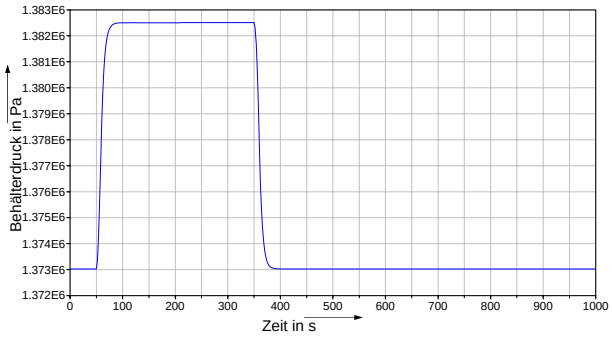


Abbildung 9: *Druckverlauf im Speisewasserbehälter beim Kondensatstau*

3 Zusammenfassung und Ausblick

Modelica ist zur Modellierung von Kreisprozessen aufgrund des verwendeten objektorientierten Ansatzes sehr gut geeignet. Die quelloffene Bibliothek `ThermoPower` stellt eine gute Ausgangsbasis für die Modellierung fehlender Bauteile dar. So ist das bestehende Modell `Pump` auf Parametrierung mit Hilfe eines dreidimensionalen Kennfeldes erweitert worden. Sollen wirkungsgradoptimale Betriebspunkte durch eine Funktionalität der Blockregelung angefahren werden, kann das Wirkungsgradkennfeld der Pumpe ebenfalls als Parametersatz hinterlegt werden. Der aktuelle hydraulische Wirkungsgrad kann dann als Optimierungskenngröße herangezogen werden.

Für die Zukunft sind u.a. folgende Modellverbesserungen vorgesehen: Einbindung von Wärmeverlusten an unbeheizten Bauteilen und das Erstellen neuer Modelle, die für die Modellierung von Anfahrprozessen (Zyklon-Abscheider, Anfahrflasche) von Bedeutung sind.

Literatur

[1] Casella, F. und A. Leva: *Modelica Open Library for Power Plant Simulation: Design and Experimental Validation.* In: *Proceedings of the 3rd International Modelica Conference, Linköping, Peter Fritzson (editor)*, 3 November 2003.

[2] Casella, F. und A. Leva: *Object-Oriented Modelling & Simulation of Power Plants with Modelica*. In: *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain*, 12 Dezember 2005.

[3] Fritzson, F.: *Prinziples of Object-Oriented Modeling and Simulation with Modelica*. Wiley Interscience, 2004.

[4] Weber, H., T. Haase, F. Gottelt, J. Nocke und E. Hassel: *Kraftwerksbetrieb bei Einspeisung von Windparks*. Konferenz Elektrotechnik, Leittechnik, Informationstechnik im Kraftwerk, Hamburg, 6 Mai 2008.

Funktionsblockentwicklung zur effizienten Codegenerierung für Regelungen in Windenergieanlagen

Dipl.-Ing. Olaf Naujocks, Fachhochschule Westküste
naujocks@fh-westkueste.de
Prof. Dr.-Ing. Reiner Schütt, Fachhochschule Westküste
schuett@fh-westkueste.de

Zusammenfassung

Die Entwicklung der Steuerung und Regelung dynamischer Systeme wird in zunehmendem Maße mit modellbasierten Werkzeugen durchgeführt. Insbesondere für die aufwändigen und komplexen Regelkreise in Windenergieanlagen ist es erforderlich, einen effizienten Code zu generieren.

Im Falle von digitalen Realisierungen kann durch die automatische Code-Generierung ein direkter Zusammenhang zwischen dem mathematischen Simulationsmodell und der Steuer- und Regelungssoftware des Prototyps hergestellt werden. Die Funktionsfähigkeit der entwickelten Lösung kann somit zu jeder Zeit am lauffähigen Objekt verifiziert werden.

Eines der am häufigsten verwendeten Entwicklungswerkzeuge ist Matlab/Simulink. Zusammen mit den Werkzeugkomponenten Real-Time Workshop und Real-Time Workshop Embedded Coder ist es in der Lage, automatische Code-Generierung durchzuführen. Die Eingabe des Modells erfolgt grafisch durch ein Blockschaltbild.

In diesem Beitrag wird gezeigt, wie Funktionsblöcke für Simulink entwickelt werden können. Dabei wird besonderes Augenmerk darauf gelegt, wie die Code-Generierung beeinflusst werden kann. Dadurch ist es möglich, Code erzeugen zu lassen, der eigenen Vorstellungen entspricht. Am Beispiel eines Bandpass Filters wird gezeigt, wie die Code-Generierung so beeinflusst werden kann, dass der erzeugte Code für die Ausführung auf einem Festkomma DSP (TMS320f2812) optimiert ist. Dieses Vorgehen wurde erfolgreich in einem Projekt angewendet, das zum Ziel hat, die Regelungen in Windenergieanlagen zu untersuchen und verbessern.

1 Einleitung

Die Fachhochschule Westküste (FHW) beteiligt sich an dem Kompetenzzentrum für Windenergie des Landes Schleswig Holstein (cewind). An der Fachhochschule Westküste wird ein Projekt in dem Bereich der Elektrotechnik/Informationstechnik bearbeitet. Ziel des

Projektes ist es, durch den Einsatz gehobener Regelungsverfahren das Betriebsverhalten von Windkraftanlagen zu verbessern. Ansätze hierfür finden sich in [1].

Für den Entwurf der Regler wird eine modellbasierte Entwicklungsumgebung eingesetzt. Sie ermöglicht es aus einem Simulationsmodell heraus, den Programmcode für eine digitale Steuergeräte-Implementierung automatisch zu generieren. Diese Lösung kann jederzeit an einem Prototypen getestet werden, so dass Probleme bei der Umsetzung (z.B. begrenzter Speicher oder zu langsame Programmausführung) frühzeitig erkannt werden.

In diesem Beitrag wird gezeigt, wie ein Funktionsblock für Simulink entwickelt wird, der einen Bandpass Filter realisiert. Dieses wird in dem durchgeführten Projekt verwendet, um das Netzspannungsmesssignal von hochfrequenten Störungen und Gleichanteil zu befreien. Abbildung 1 zeigt, wie das aufbereitete Signal zur Bestimmung der Phase der Netzspannung mittels PLL verwendet wird. Nur der Bandpass am Eingang und das Notch-Filter in der Regelschleife ermöglichen hier die dynamische, exakte Netzsynchronisation.

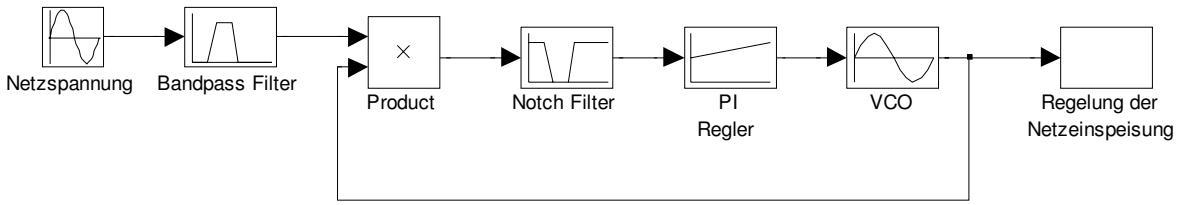


Abbildung 1: Blockschaltbild zur Bestimmung der Phase der Netzspannung

2 Zustandsraumdarstellung von Simulink Funktionsblöcken

Simulationsmodelle werden in Simulink durch Blockschaltbilder dargestellt. Die Funktionsblöcke sind untereinander durch Wirklinien verbunden. Rückwirkungen werden durch eine zurückgeführte Wirklinie dargestellt. Die Abbildung 2 zeigt ein Filter, das einen nicht vernachlässigbaren Einfluss auf die Signalquelle hat.

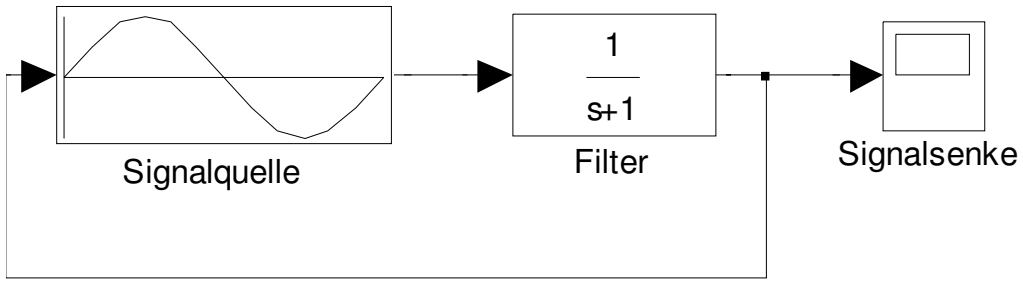


Abbildung 2 : Modell eines nicht rückwirkungsfreien Filters

Simulink verwendet für die mathematische Beschreibung des Verhaltens eines Funktionsblocks die Zustandsraumdarstellung. Dies gilt auch im Fall von Standard Funktionsblöcken wie Übertragungsfunktion oder Pol-Nullstellen Darstellung. Simulink rechnet diese Darstellungsformen intern in die Zustandsraumdarstellung um. Für lineare, zeitinvariante Systeme (LTI Systeme) hat die Zustandsraumdarstellung die Form :

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \quad (1)$$

$$y(t) = C \cdot x(t) + D \cdot u(t) \quad (2)$$

Das Verhalten des Systems wird in diesem Gleichungssystem durch die Matrizen A, B, C und D festgelegt. Die Eingänge u, die Zustände x, sowie deren Ableitungen $\dot{x}$ und die Ausgänge y sind jeweils Vektoren.

Um das zeitliche Verhalten des so spezifizierten Systems zu berechnen, muss die Differentialgleichung (1) gelöst werden. Anschließend kann nach (2) die Systemantwort auf den Eingangsvektor u berechnet werden. Für die Lösung der Differentialgleichung können in Simulink verschiedene Lösungsverfahren (Solver) gewählt werden.

Die Gleichungen (1) und (2) beschreiben das Verhalten eines zeitkontinuierlichen Systems. Daneben existieren zeitdiskrete Systeme, deren Verhalten über die Zeit nur für bestimmte Zeitpunkte (Abtastzeitpunkte) festgelegt ist. Für diese gilt die diskrete Variante der Zustandsraumdarstellung :

$$x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) \quad (3)$$

$$y(k) = C \cdot x(k) + D \cdot u(k) \quad (4)$$

Die diskrete Zustandsraumdarstellung nach (3) und (4) eignet sich für die Implementierung in Steuergeräten wie DSPs oder FPGAs.

3 Funktionsblockentwicklung mit S-Functions und TLC

Simulink bietet mit den so genannten S-Functions (Simulink System Functions) die Möglichkeit, das Verhalten eines Funktionsblocks in einer der Programmiersprachen c, Matlab, Ada oder Fortran zu beschreiben. Dazu muss eine Verhaltensbeschreibung in der Zustandsraumdarstellung vorliegen. Handelt es sich um ein zeitdiskretes System, kann zudem mit dem Target Language Compiler (TLC), der Teil der Matlab/Simulink Toolbox Real-Time Workshop (RTW) ist, der für den Funktionsblock zu erzeugende Code angegeben werden.

Die Entwicklung eigener Funktionsblöcke wird am Beispiel des Bandpassfilters aus Abbildung 1 veranschaulicht. Zunächst wird die Realisierung eines kontinuierlichen Tiefpasses für Simulationszwecke gezeigt. Anschließend wird eine zeitdiskrete Implementierung desselben Bandpasses gezeigt. Für die zeitdiskrete Implementierung wird Code für ein Steuergerät erzeugt. Schließlich wird die Code-Erzeugung so beeinflusst, dass der erzeugte Code hinsichtlich seiner Ausführungsgeschwindigkeit verbessert ist.

Es wird gezeigt, wie das Verhalten der Blöcke in den jeweiligen S-Function und TLC Quelldateien formuliert wird. Für die S-Functions wird dabei die Programmiersprache c verwendet. Eine vollständige Einführung in diese Thematik findet sich in [2].

3.1 Zeitkontinuierliches Simulationsmodell

Um eine Bandpass Charakteristik zu erhalten, werden ein Tiefpass und ein Hochpass hintereinandergeschaltet. Dadurch ergibt sich die Übertragungsfunktion

$$H_{BP}(s) = H_{TP}(s) \cdot H_{HP}(s) = \frac{s \cdot \frac{1}{\omega_{HP}}}{s^2 \cdot \frac{1}{\omega_{HP} \cdot \omega_{TP}} + s \cdot \frac{1}{\frac{1}{\omega_{HP}} + \frac{1}{\omega_{TP}}} + 1} \tag{5}$$

Ziel ist es, mit dem Bandpass Gleichanteil (Nullpunktfehler des Analog-Digital-Wandlers) und hochfrequente Störungen möglichst gut aus dem Messsignal zu entfernen. Das Netzspannungssignal soll nach der Filterung keine Phasendifferenz zum Originalsignal aufweisen. Der Bandpass nach (5) hat seine Mittenfrequenz bei

$$\omega_m = \sqrt{\omega_{HP} \cdot \omega_{TP}} \tag{6}$$

Bei dieser Frequenz hat der Phasengang für H_{BP} einen Nulldurchgang. Deswegen wird die Mittenfrequenz gewählt zu

$$\omega_m = 2 \cdot \pi \cdot 50 \ \frac{rad}{s} \tag{7}$$

Für die Wahl der Grenzfrequenzen muss berücksichtigt werden, dass das Messsignal nur dann ohne Phasendifferenz bleibt, wenn es genau 50 Hz hat. Tatsächlich kann die Netzspannung von diesem Wert abweichen. Je weiter entfernt die Grenzfrequenzen von der Mittenfrequenz liegen, desto flacher ist der Phasendurchgang. Andererseits bewirkt dies eine schlechte Ausfilterung der unerwünschten Signalanteile. Als Kompromiss werden die Werte aus (8) verwendet, welche über die Gleichung (6) miteinander zusammenhängen.

$$\omega_{HP} = 2 \cdot \pi \cdot 10 \ \frac{rad}{s} \ ; \ \omega_{TP} = 2 \cdot \pi \cdot 250 \ \frac{rad}{s} \tag{8}$$

Um von der Übertragungsfunktion (5) zu der Zustandsraumdarstellung zu gelangen, wird der Matlab Befehl tf2ss verwendet. Dieser liefert als Ergebnis die Matrizen

$$A = \begin{pmatrix} 1633.628 & 98696.04 \\ 1 & 0 \end{pmatrix}; B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; C = (1570.796, 0); D = 0 \tag{9}$$

Während der Simulation ruft Simulink verschiedene Funktionen der S-Function (Callback Functions) auf. Die Funktion *mdlOutputs* dient der Berechnung der Ausgangssignale nach (2). In der Funktion *mdlDerivatives* wird die Differentialgleichung (1) spezifiziert. Folgender Code zeigt die programmsprachliche Realisierung des Bandpasses-Verhaltens mit den Koeffizienten nach (9).

```
static void mdlDerivatives(SimStruct *S){
        const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
        real_T       *dx   = ssGetdX(S);
        real_T       *x    = ssGetContStates(S);
        dx[0]=A00*x[0]+A01*x[1]+B00*u[0];
        dx[1]=A10*x[0]+A11*x[1]+B10*u[0];
}

static void mdlOutputs(SimStruct *S, int_T tid){
        const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
```

```
        real_T          *y    = ssGetOutputPortRealSignal(S,0);
        real_T          *x    = ssGetContStates(S);
        y[0]=C00*x[0]+C01*x[1]+D00*u[0];
}
```

Der Zugriff auf Simulationsdaten wie die momentanen Werte von Ein-, Aus- und Zustandssignalen erfolgt durch die so genannten SimStruct[1] Functions. Diese beginnen immer mit den Buchstaben ss, wie im obigen Code die Funktion *ssGetInputPortSignal*. Einen Überblick über alle SimStruct Functions gibt [3].

3.2 Zeitdiskretes Simulationsmodell

Um den Bandpass nach (5) als zeitdiskretes System zu realisieren, wird die zeitkontinuierliche Zustandsraumdarstellung mit den Matrizen aus (9) diskretisiert. In [4] wird gezeigt, wie dies durchgeführt wird. Es ändern sich nur die Werte der Matrizen A und B. Sie erhalten deswegen den Index d. Berechnet werden die Werte für A_d und B_d nach (10) und (11) :

$$A_d = e^{A \cdot T} \tag{10}$$

$$B_d = A^{-1}\left(e^{(AT)} - I\right)B \tag{11}$$

Die Abtastperiode T wird entsprechend der Abtastperiode des Steuergerätes gewählt :

$$T = \frac{1}{5000\ \mathrm{Hz}} = 0.0002\ \mathrm{s} \tag{12}$$

In der Formel (11) ist I die Einheitsmatrix. Für den diskretisierten Bandpass ergibt sich damit für die Matrizen der Zustandsraumdarstellung

$$A_d = \begin{pmatrix} 0.71968 & -16.8277 \\ 1.705 \cdot 10^{-4} & 0.99822 \end{pmatrix};\ B_d = \begin{pmatrix} 1.705 \cdot 10^{-4} \\ 1.789 \cdot 10^{-8} \end{pmatrix};\ C = (1570.79, 0);\ D = 0 \tag{13}$$

Im Fall eines zeitdiskreten Funktionsblocks ruft Simulink anstelle der Funktion *mdlDerivatives* die Funktion *mdlUpdate* auf. Diese wird mit den veränderten Matrizen A_d und B_d folgendermaßen formuliert.

```
static void mdlUpdate(SimStruct *S,int_T tid){
        real_T* u = (real_T*) ssGetInputPortSignal(S,0);
        real_T          tempX[2] = {0, 0};
        real_T* x     = (real_T*)ssGetDiscStates(S);
        tempX[0]=Ad00*x[0]+Ad01*x[1]+Bd00*u[0];
        tempX[1]=Ad10*x[0]+Ad11*x[1]+Bd10*u[0];
        x[0]=tempX[0];
        x[1]=tempX[1];
}
```

Die Funktion *mdlOutputs* bleibt unverändert.

[1] Die Simulationsdaten werden in einer Struktur namens SimStruct gespeichert

3.3 Code-Generierung für den zeitdiskreten Bandpass

Damit für den zeitdiskreten Bandpass Code generiert werden kann, muss eine TLC Quelldatei für den Funktionsblock bereitgestellt werden. Darin haben die Funktionen Outputs und Update dieselben Funktionen wie *mdlOutputs* und *mdlUpdate* in der S-Function bzw. in der Simulation.

In diesem Beispiel wird Code für einen DSP des Typs TMS320f2812 von Texas Instruments (TI) erzeugt. Der Code wird in der Sprache c erzeugt. Die Erzeugung von Assembler Code wäre ebenfalls denkbar.

Für die Code-Erzeugung wird wie in der Simulation der Zugriff auf Daten benötigt, die sich erst zum Zeitpunkt der Programmübersetzung oder während der Laufzeit ergeben. Zu diesem Zweck bietet der TLC verschiedene Funktionen an. Diese Funktionen sind gekennzeichnet dadurch, dass sie mit dem Kürzel Lib beginnen. Einen Überblick über die Funktionen gibt [5]. Um den TLC anzuweisen, DSP-Code für den zeitdiskreten Bandpass aus Abschnitt 3.2 zu erzeugen, wird in einer Quelldatei folgendes eingegeben :

```
%function Update(block, system) Output
{
        real_T          tempX[2] = {0.0, 0.0};
        tempX[0]=Ad00*%<LibBlockDiscreteState("","",0)>
                +Ad01*%<LibBlockDiscreteState("","",1)>
                +Bd00*%<LibBlockInputSignal(0,"","",0)>;
        tempX[1]=Ad10*%<LibBlockDiscreteState("","",0)>
                +Ad11*%<LibBlockDiscreteState("","",1)>
                +Bd10*%<LibBlockInputSignal(0,"","",0)>;
        %<LibBlockDiscreteState("","",0)>=tempX[0];
        %<LibBlockDiscreteState("","",1)>=tempX[1];
}
%endfunction
%function Outputs(block, system) Output
        %<LibBlockOutputSignal(0,"","",0)>=C00*%<LibBlockDiscreteState("","",0)>
                +C01*%<LibBlockDiscreteState("","",1)>
                +D00*%<LibBlockInputSignal(0,"","",0)>;
%endfunction
```

Es ist eine starke Vermischung von dem eigentlichen DSP-Code und Anweisungen an den TLC zu erkennen.

3.4 Optimierung der Code-Generierung

Der in Abschnitt 3.3 erzeugte Code verwendet für die Darstellung der Werte Fließkomma Datentypen. Der verwendete DSP ist jedoch ein Festkomma Typ. Dies führt dazu, dass die gesamte Fließkommaarithmetik mit aufwendigen Funktionen nachgebildet wird. Mathematische Operationen und Funktionen wie Multiplikation oder Sinus benötigen deshalb ein Vielfaches an Zeit zur Ausführung wie eine entsprechende Festkomma Realisierung. Eine Umstellung des erzeugten Codes auf Festkommaarithmetik führt deshalb zu einer bedeutenden Verringerung der benötigten Prozessorzeit.

Festkomma Datentypen haben einen begrenzten Wertebereich und die relative Genauigkeit der Zahlendarstellung ist nicht wie beim Fließkomma Datentyp konstant. Dies kann dazu führen, dass der Funktionsblock ein Verhalten zeigt, das von dem gewünschten abweicht.

Es hat sich gezeigt, dass die Verwendung der Matrizenkoeffizienten aus (13) zu einer Verhaltensabweichung führt, die nicht akzeptabel ist. Es werden deshalb die Koeffizienten aus (14) verwendet, die sich aus einer empirischen Umskalierung ergeben. Ein systematisches Vorgehen für die Umskalierung der Matrizenkoeffizienten findet sich in [6].

$$A_d = \begin{pmatrix} 0.7197 & -0.016828 \\ 0.1705 & 0.99822 \end{pmatrix};\ B_d = \begin{pmatrix} 0.1705 \\ 0.01798 \end{pmatrix};C = (1.5708,0);D = 0 \qquad (14)$$

Für das Rechnen mit Festkommazahlen stellt TI die Bibliothek IQ-Math zur Verfügung. In dieser befinden sich Funktionen zur Berechnung mathematischer Operationen und Funktionen unter Verwendung einer Festkommaarithmetik. Folgender TLC-Code zeigt, wie DSP-Code erzeugt wird, der für die Berechnung der Multiplikation die Funktion IQ24mpy[2] aus der IQ-Math Bibliothek verwendet..

```
%function Update(block, system) Output
{
        int32 tempX[2] = {0.0, 0.0};
        int32* States;
        States=%<LibBlockPWork("","","",0)>;
        tempX[0]=_IQ24mpy(Ad00,States[0])
                +_IQ24mpy(Ad01,States[1])
                +_IQ24mpy(Bd00,%<LibBlockInputSignal(0,"","",0)>);
        tempX[1]=_IQ24mpy(Ad10,States[0])
                +_IQ24mpy(Ad11,States[1])
                +_IQ24mpy(Bd10,%<LibBlockInputSignal(0,"","",0)>);
        States[0]=tempX[0];
        States[1]=tempX[1];
}
%endfunction
```

In dem DSP Programm wird nun eine andere Darstellung- und Berechnungsweise als in der Simulation verwendet. Die TI Bibliothek "IQ Math" kann nicht für die S-Functions verwendet werden, da sie nicht ANSI c konform sind. Es kann jedoch eine Funktionsbibliothek verwendet werden, welche an der Fachhochschule Westküste im Rahmen des Projektes cewind entwickelt wurde. Diese Bibliothek ermöglicht die Simulation der Festkommaarithmetik der DSP Reihe TMS320f28x.

[2] Mit 24 wird die Anzahl der Nachkommastellen für die Festkommadarstellung angegeben

4 Ergebnisse

Um die Ergebnisse aus Abschnitt 3 zu bewerten, werden die Amplitudengänge aller Realisierungsarten in einem Diagramm dargestellt. Die Abbildung 3 zeigt die Ergebnisse für den Frequenzbereich von 0.1 Hz bis 2500 Hz (Abtastrate/2).

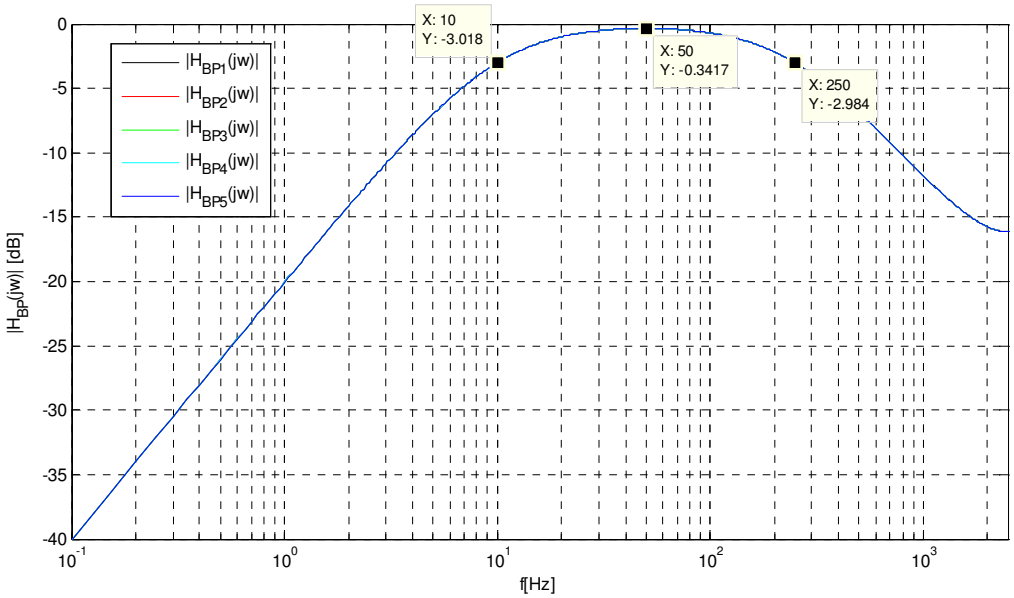


Abbildung 3 : Amplitudengänge der verschiedenen Realisierungsarten

Anhand der Legende ist zu erkennen, dass in dem Diagramm 5 Amplitudengänge dargestellt sind. Im einzelnen sind dies

- |H1(jw)| - zeitkontinuierliche Fließkomma Simulation
- |H2(jw)| - zeitdiskrete Fließkomma Simulation
- |H3(jw)| - zeitdiskrete Festkomma Simulation
- |H4(jw)| - zeitdiskretes Fließkomma DSP Programm
- |H5(jw)| - zeitdiskretes Festkomma DSP Programm

Es ist kaum ein Unterschied der Amplitudengänge erkennbar. Grundsätzlich liefern alle Realisierungsarten das erwartete Verhalten. Bei der Mittenfrequenz ist der Betrag der Übertragungsfunktion nicht 0 dB. Dies war auch nicht zu erwarten, da sowohl Hoch- als auch Tiefpass bei dieser Frequenz noch eine leichte Dämpfung verursachen. Durch eine Verstärkung hinter dem Bandpass wird der Betrag so angehoben, dass die Mittenfrequenz mit dem Betrag 0 dB übertragen wird.

Bei hohen Frequenzen zeigt sich, dass der Amplitudengang von dem eines idealen Bandpasses abweicht. Dies liegt daran, dass für dessen Bestimmung eine Impulsantwort aufgenommen wurde. Dabei werden Frequenzen erzeugt, die über der halben Abtastfrequenz liegen. Dies führt zum so genannten Alias-Effekt [7].

Literatur

[1] Schütt, Reiner : Gehobene Mehrgrößen-Regelungsverfahren in Windenergieanlagen – Schritte zur Integration, 9. Energietechnischen Forums der Fachhochschule Kiel,, 2005

[2] Naujocks, Olaf : Erstellung eigener Funktionsblöcke mit S-Functions und dem TLC, Fachhochschule Westküste, 2008

[3] The Mathworks, Inc.: SimStruct Functions - Alphabetical List, http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/sfg/bqa667w.html

[4] Ogata, Katsuhiko : Discrete-Time Control Systems, 2nd Edition, 5:312-321, 1995

[5] The Mathworks,Inc.: TLC Function Library Reference, http://www.mathworks.com/access/helpdesk/help/toolbox/rtw/tlc/f6010.html

[6] Steinbuch, Maarten; Schootstra, Gerrit; Goh, Hoon-Toh : Closed-Loop Scaling in Fixed-Point Digital Control, IEEE Transactions on Control Systems Technology Vol.2, 1994

[7] Götz, H. : Einführung in die digitale Signalverarbeitung, Teuber Studienskripten, 2.2:60-62, 1995

Systematische Qualitätssicherung bei verteilt erstellten Simulationsmodellen

Eduard Bröcker[1], Sven Dominka[2], Martin Merz[1]
[1]Technische Universität München, Lehrstuhl für Informationstechnik,
{broecker, merz}@itm.tum.de
[2]University of Melbourne, Australia, sdominka@unimelb.edu.au

Zusammenfassung

Bei der Entwicklung von großen Simulationsmodellen ist es auf Grund der Komplexität notwendig im Team zu arbeiten. Um eine hohe Güte der Simulationsmodelle zu erreichen, ergeben sich durch diese verteilte Entwicklung neue Herausforderungen hinsichtlich der Qualitätssicherung. Um den Aufwand für die Qualitätssicherung möglichst gering zu halten, müssen die eingesetzten Maßnahmen systematisch durchgeführt werden. Hierfür ist es sinnvoll, einzelne Schritte der Verifikation und Validierung zu automatisieren. Im vorliegenden Beitrag werden Ansätze zur systematischen Qualitätssicherung bei verteilt erstellten Simulationsmodellen vorgestellt sowie deren Umsetzung im Rahmen eines Simulationsprojekts beschrieben.

1 Motivation und Problemstellung

Durch die hohen Kundenanforderungen bezüglich steigender Produktfunktionalität steigt die Komplexität der Produkte gewaltig an. Gleichzeitig zwingt der Markt, Produkte schneller und kostengünstiger zu entwickeln [KOB98]. Diese Schere zwischen geforderter Produktkomplexität auf der einen Seite und Marktdruck auf der anderen Seite geht durch die Globalisierung der Märkte noch weiter auseinander. Die Internationale Konkurrenzsituation bestärkt den Innovations- und Kostendruck der Unternehmen.

Der Einsatz von Simulationen in der Produktentwicklung hilft, diese Herausforderungen zu bewältigen [BAL04]. Das Potential und der Nutzen von Simulationsstudien sind im modernen Engineering nahezu unumstritten um frühzeitig neuartige Produktkonzepte evaluieren zu können. Um jedoch den vielversprechenden Nutzen von Simulationen ausschöpfen zu können, müssen zunächst Simulationsmodelle des zu entwickelnden Produkts implementiert werden. Der Aufwand für die Erstellung von Simulationsmodellen darf nicht unterschätzt werden. Zwar gilt immer die Regel, dass die Detaillierung des Simulationsmodells immer an den Simulationszweck angepasst werden muss, handelt es sich jedoch bei einem Produkt um ein hochkomplexes, mechatronisches System, so ist im allgemeinen ein sehr umfangreiches

und ebenso komplexes Simulationsmodell unvermeidbar. Übersteigt die Größe und Komplexität eines Modells einen gewissen Umfang, ist es meist nicht mehr möglich dieses Modell alleine zu entwickeln. Oft müssen bei der Implementierung eines komplexen Modells Fachexperten aus unterschiedlichen Bereichen mit einbezogen werden. In diesen Fällen arbeiten im Allgemeinen mehrere Ingenieure zusammen an einem Modell, wobei jeder Ingenieur einen Teilbereich des Modells entwickelt, so dass die verschiedenen Kompetenzen sinnvoll eingesetzt werden können.

Sobald allerdings mehr als eine Person an einem Simulationsmodell arbeitet, kommt es zunächst fast unweigerlich zu zahlreichen Problemen: Meist arbeiten die einzelnen Ingenieure je an einer lokalen Kopie des Gesamtsimulationsmodells, was wiederum zu Redundanz-Problemen am Modell selber führen kann. Fehler, die im Simulationsmodell von Ingenieur A behoben sind, sind in der Kopie des Simulationsmodells bei Ingenieur B weiterhin vorhanden. Auch eine Versionsverwaltung alleine, wie man sie aus der Softwareentwicklung kennt, kann dieses Problem nur teilweise lösen, da nie garantiert werden kann, dass die von verschiedenen Ingenieuren erarbeiteten Modifikationen kompatibel zueinander sind.

Steigt die Komplexität eines Gesamtmodells so stark an, dass sie von einem einzelnen Ingenieur nicht mehr überblickt werden kann, hat dies zur Folge, dass nach jeder noch so kleinen Modelländerung, geprüft werden müsste, ob weiterhin alle (anderen) Funktionen des Gesamtsimulationsmodells lauffähig und korrekt sind [SAD05]. Gerade hierbei ist zu beachten, dass ein Gesamtsimulationsmodell nicht die Summe aller seiner Teilsimulationsmodelle ist. Es ist möglich dass das Gesamtsimulationsmodell nicht korrekt ist, obwohl alle Teilmodelle einzeln auf Korrektheit geprüft worden sind, da manche Fehler erst durch die Interaktion der Teilmodelle innerhalb des integrierten Gesamtsimulationsmodells auftreten.

Qualitätssichernde Tätigkeiten parallel zur Entwicklung des eigentlichen Modells sind unumgänglich, wenn eine korrekte und qualitativ hochwertige Simulation entstehen soll [BAL95]. Je größer das angestrebte Simulationsmodell ist und je mehr Entwickler bei der Erstellung beteiligt sind, desto höher wird der Aufwand für die Qualitätssicherung. Die ständige Kontrolle hinsichtlich eingehaltener Schnittstellenspezifikation und eines ablauffähigen Gesamtsimulationsmodells ist dabei eine zeitaufwändige und fehleranfällige Routineaufgabe.

Dieser Beitrag soll einen Einblick in die Vorgehensweise bei der Modellierung umfangreicher Systeme geben. Dabei handelt es sich vor allem um die systematische Qualitätssicherung bei verteilt erstellten Simulationsmodellen.

Im folgenden Abschnitt werden Vorgehensweisen und Methoden vorgestellt, die für die Entwicklung eines umfangreichen Simulationsmodells in einem multidisziplinären Team ausgewählt wurden. Im darauffolgenden Abschnitt wird die tatsächliche Anwendung einiger dieser Vorgehensweisen und Methoden, an Hand eines in Dymola entwickelten Simulationsmodells vorgestellt.

2 Methoden und Vorgehensweisen für die qualitätsorientierte Entwicklung von Simulationsmodellen

2.1 Empfohlene Vorgehensweise bei der Modellbildung und Simulation

Um bei einem komplexen Simulationsmodell eine systematische verteilte Erstellung zu ermöglichen, ist es sinnvoll sich vor Beginn der Modellierung Gedanken über das generelle Vorgehen zu machen. Abbildung 1 zeigt ein allgemeines Vorgehen für Modellbildung und Simulation (M&S).

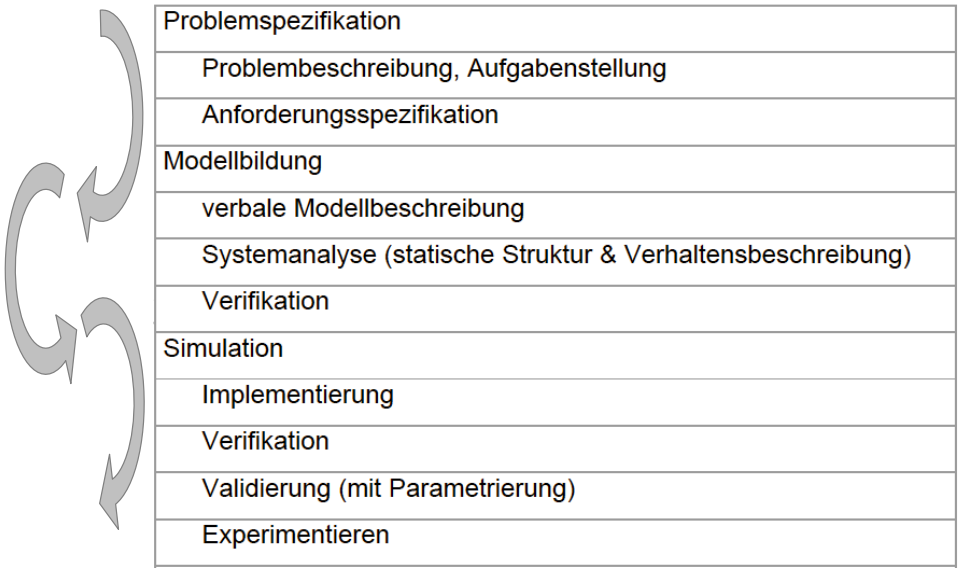


Abbildung 1: M&S Vorgehensweise

- Problemspezifikation: Die Problemspezifikation beinhaltet allgemein eine Beschreibung dessen was simuliert werden soll und wofür die Simulationsstudie durchgeführt werden soll.
  - Problembeschreibung: Hier sollte begründet werden, warum das Simulationsmodell notwendig ist und welche Fragestellungen durch die Simulation geklärt werden sollen.
  - Anforderungsspezifikation: An dieser Stelle sollen Anforderungen an das Simulationsmodell formal spezifiziert werden. Dabei wichtig sind die gewünschten bzw. geforderten Gültigkeitsbereiche des Simulationsmodells und die geforderte Genauigkeit der Simulation. Diese Anforderungen dienen in späteren Phasen zur Validierung des Simulationsmodells gegen das vorhandene oder gedachte System. Im Fall von neuen Lösungskonzepten können auch funktionale Anforderungen spezifiziert werden, wenn das Konzept neue Funktionen mitbringen soll.

- Modellbildung: Die Modellbildung dient dazu, das nachzubildende System zu analysieren, zu abstrahieren und für das Simulationsmodell ein Umsetzungskonzept zu entwerfen.
  - verbale Modellbeschreibung: Die Beschreibung in Prosatext dient zum allgemeinen Verständnis über den eigentlichen Aufbau und die Funktionsweise des nachzubildenden Systems.
  - Systemanalyse: In der Systemanalyse wird das System systematisch hinsichtlich aller Systemelemente und deren Wirkungen aufeinander untersucht. Die Komplexität des Systems wird dabei durch Zergliederung in einzelne Elemente aufgelöst und formal vollständig beschrieben.
  - Verifikation: Überprüfung des konzeptionellen Modells, also der verbalen Modellbeschreibung und dem Ergebnis der Systemanalyse auf Korrektheit, Verständlichkeit und Vollständigkeit.
- Simulation: Die Phase der Simulation besteht aus folgenden Teilphasen
  - Implementierung: Die Umsetzung des bei der Systemanalyse entwickelten konzeptionellen Modells in ein ablauffähiges Simulationsmodell. An dieser Stelle werden die statische Struktur und die formalen, mathematischen Verhaltensbeschreibungen meist mit Unterstützung eines Simulationstools implementiert.
  - Verifikation: Die Verifikation des Simulationsmodells überprüft die korrekte Implementierung des konzeptionellen Modells.
  - Validierung: Die Validierung überprüft für bereits existierende Systeme, ob sich das Simulationsmodell für die in der Anforderungsspezifikation festgelegten Randbedingungen genau so verhält wie das existierendes System. Für nichtexistierende Systeme umfasst die Validierung in der Regel eine Plausibilitätsüberprüfung.
  - Experimentieren: Nachdem überprüft wurde, dass das Simulationsmodell „richtig“ entwickelt wurde, und auch korrekte Ergebnisse liefert, kann an dieser Stelle mit den eigentlichen Simulationsuntersuchungen begonnen werden.

Methoden der Qualitätssicherung im Bereich der Modellbildung und Simulation lassen sich, ähnlich wie im benachbarten Bereich der Softwareentwicklung, in konstruktive und analytische Qualitätssicherungsmaßnahmen aufteilen. Während die konstruktiven Maßnahmen auf eine hohe Qualität der produzierenden, entwickelnden Tätigkeiten wie der eigentlichen Modellerstellung abzielen, haben analytische Maßnahmen das Ziel, die Qualität der Erzeugnisse, also der (Teil-)Ergebnisse auf ihre Qualität zu überprüfen. Die analytische Qualitätssi-

cherung im Bereich der Modellbildung und Simulation lässt sich prinzipiell in Verifikation und Validierung unterteilen.

2.2 Verifikation

Die Verifikation umfasst generell die Überprüfung auf 'Entwicklungsfehler'. Hierbei sind sämtliche Fehler enthalten, die von den Entwicklern in dem gesamten Entstehungsprozess des Simulationsmodells gemacht wurden.

Die erste Verifikation findet während der Anforderungsspezifikation statt, indem in einem „Schreibtischtest“ die Anforderungen geprüft werden. Bei einem Schreibtischtest wir ein (Zwischen-)Ergebnis einem Dritten zur Überprüfung hinsichtlich Korrektheit, Vollständigkeit und Plausibilität vorgelegt. Dabei kann es sich um Spezifikation und Dokumentationen, aber auch um Konzepte oder Programmcode handeln.

Die zweite Verifikationsphase findet während der Modellbildung statt. Hier sollten zunächst wieder Schreibtischtests zur Verifikation des konzeptionellen Modells durchgeführt werden. Nach diesem iterativen Prozess, der das Nachbessern des konzeptionellen Modells und das erneute Überprüfen beinhaltet, kann das Ergebnis schließlich einem Walkthrough unterzogen werden [SAR05] [SG04]. Dabei wird das Modell in einem Team von Simulations- und Qualitätssicherungsexperten Schritt für Schritt begutachtet und für die Simulationsphase freigegeben (bzw. abgelehnt).

Die dritte Verifikationsphase findet während der Simulationsphase statt. Hierbei kann die Verifikation wie folgt dreigeteilt werden:

- Verifikation des implementierten Simulationsmodells gegen das konzeptionelle Modell. Dies ist eine Überprüfung ob das implementierte Modell mit dem konzeptionellen Modell übereinstimmt. Dabei werden die Struktur, die Schnittstellen und das Verhalten verglichen. Programmfehler, wie syntaktische und semantische Fehler, können hierbei ebenfalls erkannt werden.

- Verifikation des implementierten Simulationsmodells gegen Syntax- und Styleguides: Bei der Code-Verifikation werden Codierrichtlinien, wie Namenskonventionen von Variablen und Parametern, die Verwendung bestimmter Anweisungen oder Vorgaben wie Orte für die Deklaration von Variablen und Konstanten überprüft [GOR07].

- Dokumentationsüberprüfung: Dabei wird zunächst überprüft, ob die Dokumentation zum Modell passt. Im nächsten Schritt sollte auch die Dokumentation gegen Richtlinien geprüft werden, unter anderem wird hier die Verständlichkeit und Lesbarkeit geprüft. Wenn die Richtlinien es verlangen, sollte für jedes Submodell eine Dokumentation erstellt werden. Dokumentationsrichtlinien können nicht nur ein spezielles Layout sondern auch einen bestimmten Inhalt erfordern, wie z.B. dass der Ersteller sowie die letzte Änderung und deren Zeitpunkt dokumentiert ist.

2.3 Validierung

Während die Verifizierung bewusst Fehler in der Entwicklung sucht, konzentriert sich die Validierung auf die Überprüfung des Ergebnisses, und hier speziell auf die Ergebnisse von durchgeführten Simulationsläufen. Es wird dabei geprüft, ob das Simulationsmodell, dem ursprünglich festgelegten Modellierungszweck entsprechend, korrekte Ergebnisse liefert. Hierzu wird das erstellte Simulationsmodell simuliert bzw. ausgeführt und dessen Verhalten und Resultate kontrolliert. Dies kann bei Simulation vorhandener Systeme durch Vergleich der simulierten Werte mit Messwerten oder bei neuen Systemen durch Plausibilitätschecks der simulierten Ergebnisse erreicht werden.

2.4 Automatisierung der Verifizierungs- und Validierungsprozesse

Um den Aufwand für die Verifizierung und Validierung bei der Modellbildung und Simulation zu reduzieren, können Teile dieser Prozesse automatisiert werden. Im Folgenden werden Möglichkeiten einer solchen Automatisierung beschrieben.

Automatisierung der Verifikation

Insbesondere die Verifikationen des implementieren Simulationsmodells weist ein Potential zur Automatisierung auf:

- Die Verifikation des implementierten Modells gegen das konzeptionelle Modell lässt sich nur mit hohem Aufwand automatisieren. In Fällen, in denen die Modellspezifikation in einer formalen Sprache in geeigneter, digitaler Form erfolgt ist, ist eine automatisierte Verifikation dennoch möglich. Normalerweise werden die konzeptionellen Modelle jedoch in gewöhnlichen Textverarbeitungsprogrammen formuliert, was eine automatisierte Verifikation für den praktischen Einsatz zur Zeit noch unmöglich macht.

- Die zweite Stufe der Verifikation, die Verifikation des implementierten Simulationsmodells gegen Syntax- und Styleguides, ist großteils automatisierbar. Viele Codierrichtlinien lassen sich durch einfache lexikalische Analysen des Quellcodes analysieren.

- Die Verifikation des Simulationsmodells gegen Syntax-Vorgaben der Modellierungssprache wird im Allgemeinen von dem entsprechenden Simulationssystem selbst durchgeführt, hier gilt es, das Ausführen des Simulationsmodells in der Simulationsumgebung automatisch durchzuführen und alle Warnungen und Fehlermeldungen automatisch abzufangen und aufzubereiten.

- Die Überprüfung der Dokumentation lässt sich teilweise automatisieren indem sie durch einfache lexikalische Analysen auf Basisinformationen hin untersucht wird. Unter Anderem ist es möglich die Dokumentation auf Einhalten von Formatvorlagen zu

überprüfen. Weiterhin kann durch einfache Stichwort-Suchen die Dokumentation daraufhin geprüft werden, ob alle Submodelle dokumentiert sind oder ob bestimmte verpflichtende Stichwörter in der Dokumentation vorkommen.

Automatisierung der Validierung

Eine teilweise Automatisierung der Validierung ist fast immer möglich. Wenn ein reales System simuliert wird und es entsprechende Messwerte gibt, kann die Validierung automatisch durchgeführt werden. Dabei können die Abweichungen vom simulierten Modell im Vergleich zu den gemessenen Werten automatisiert ausgewertet und aufbereitet werden. Bei der Validierung durch Plausibilitätstests kann analog verfahren werden.

3 Umsetzung der beschriebenen Maßnahmen in einem Simulationsprojekt

Die oben beschriebenen Qualitätssicherungsmaßnahmen wurden in einem Kooperationsprojekt der TU München zusammen mit der BMW Group umgesetzt. Das Projekt ist eines von vielen Projekten mit der BMW Group, die unter dem Dach CAR@TUM („Munich Centre of Automotive Research“) laufen. In diesem Projekt wurde unter anderem ein Simulationsmodell eines Fahrzeuges entwickelt, das alle relevanten Energieströme eines BMW 745i abbildet. Dieses Modell soll unter anderem zur Bewertung zukünftiger Alternativkonzepte dienen. Bei der Entwicklung des Simulationsmodells waren fünf Lehrstühle der TU München aus verschiedenen Fachrichtungen und die BMW Forschung und Technik GmbH beteiligt. Es waren über die Projektlaufzeit ca. 10 entwickelnde Ingenieure und zahlreiche Studenten an der Modellbildung beteiligt. Daher waren zeitgleich immer ca. 20 Personen mit der Modellierung von Teilkomponenten beschäftigt. Um Konsistenzprobleme zu vermeiden, wurden von Anfang an Richtlinien, ähnlich wie in Kapitel 2 angedeutet, ausgearbeitet und zur Verfügung gestellt. Zum gemeinsamen Arbeiten wurde ein zentrales Versionsverwaltungstool verwendet. Alle Änderungen am Modell wurden automatisch dokumentiert und sind auf einzelne Ent-

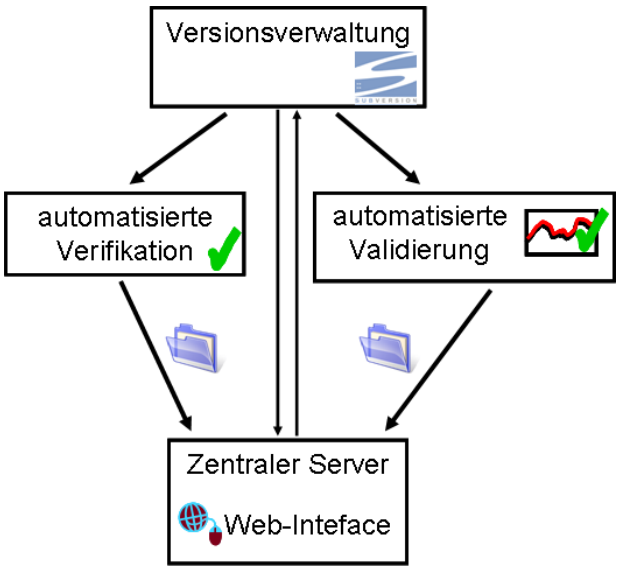


Abbildung 2: Automatisierte V&V

wickler zurückverfolgbar. Nach jedem „check-in“ in das Versionsverwaltungssystem wurde automatisch eine Verifikation durchgeführt (siehe Abb. 2). Dabei wurde jede Version automatisiert auf Einhaltung der Codierrichtlinien und auf den Stand der Dokumentation hin untersucht. Die Übersetzbarkeit wurde mit dem Simulationssystem Dymola automatisch überprüft. Hierfür wurde jede Revision automatisch „ausgecheckt“, übersetzt und ein kurzer Simulationslauf durchgeführt. Dabei wurden alle Warnungen und Fehlermeldungen dokumentiert und ausgewertet. Die Ergebnisse der Verifikation wurden grafisch in einer Baumstruktur, die an die Simulationsmodellstruktur angelehnt ist, aufbereitet.

Zusätzlich zur Verifikation wurde nach jeder Revision automatisch eine Validierung durchgeführt (siehe Abb. 2). Die Validierung des Fahrzeugmodells erfolgte durch Einspeisung von gemessenen Fahrzyklen und dem Vergleich der gemessenen Energieströme mit den simulierten. Die Ergebnisse der automatisierten Validierung wurden den Projektmitgliedern in Form einer management-tauglichen Pdf-Datei zur Verfügung gestellt.

Alle Ergebnisse der Verifikation und der Validierung wurden auf einem zentralen Server abgelegt und waren somit für alle Teammitglieder wie auch für Projektverantwortliche jederzeit einsehbar.

4 Fazit

In dem vorliegenden Beitrag wurden Möglichkeiten zur systematischen Qualitätssicherung bei verteilt erstellten Simulationsmodellen beschrieben. Hierbei wurde insbesondere auf die Automatisierung der Verifikation und Validierung im Rahmen der Modellbildung und Simulation eingegangen. Die vorgestellten Maßnahmen wurden im Rahmen eines Projekts eingesetzt, in dem in einem interdisziplinären Entwicklungsteam ein Simulationsmodell eines Fahrzeuges erstellt wurde. Das Ergebnis war nicht nur eine deutliche Einsparung des Aufwands für die Verifikation und Validierung des Modells, auch wurden alle zu Beginn des Projekts gestellten Anforderungen an das Simulationsmodell problemlos erreicht.

Literatur

[BAL04] Balci, O.: *Quality assessment, verification, and validation of modeling and simulation applications.* In: Simulation Conference. Proceedings of the 2004 Winter 1 (2004) S. 122 - 129

[BAL95] Balci, O.: *Principles and techniques of simulation validation, verification, and testing.* In: Simulation Conference Proceedings, 1995. Winter (1995) S. 147

[GOR07] Goran B.: *Überprüfung der korrekten Umsetzung von Modellierungsstandards.* In: TheMathWorksNews&Notes Juni (2007) S. 30-31

[KOB98] Kobriger, M.: *Neueinführung der Ablaufsimulation ins Unternehmen.* In: Ablaufsimulation –Anlagen effizient und sicher planen und betreiben. München: Utz, 1998.

[SAD05] Sadowski, D. A.: *Tips for Successful Practice of Simulation.* In: Winter Simulation Conference, 2005 Proceedings of the (2005) S. 56 - 61

[SAR05] Sargent, R. G.: *Verification and validation of simulation models.* In: Winter Simulation Conference, 2005 Proceedings of the (2005) S. 130 - 143

[SG04] Sadowski, D. A. ; Grabau, M. R.: *Tips for successful practice of simulation.* In: Simulation Conference, 2004. Proceedings of the 2004 Winter 1 (2004) S. 61

E-learning mit MATLAB in Mathematik Grundvorlesungen

Florian Judex, Bernhard Hametner, Felix Breitenecker
Institut für Analysis und Scientific Computing
Technische Universität Wien
efelo@osiris.tuwien.ac.at
Günther Zauner, „Die Drahtwarenhandlung" Simulation Services
Neustiftgasse 46-18, 1070 Wien, Österreich
guenther.zauner@drahtwarenhandlung.at

Zusammenfassung

Um dem Umstand, das Simulationsvorlesung im Zuge des Umstiegs auf das Bachelor/Master System in vielen Curricula vertreten sind, deren Teilnehmer keine Vorbildung in MATLAB/Simulink besitzen, wurden online Versionen der Lehrbeispiele der entsprechenden Vorlesungen den Studenten zugänglich gemacht. Dazu wurde der MATLAB Webserver um eine PHP basiertes Interface erweitert, dass es auch Nutzern ohne Kenntnisse dieser Skriptsprache erlaubt, ihre MATLAB Scripts in dieser Form zu veröffentlichen.

1 Einleitung

Im Zuge Bologna Prozesses - der Vereinheitlichung der Studien an europäischen Universitäten durch flächendeckende Einführung der Bachelor und Master Abschlüsse - wurde die Modellbildung und Simulation an der TU Wien in einige Ingenieursstudien integriert. Im Gegensatz zur den Studenten der Technischen Mathematik, die bisher die primäre Zielgruppe der allgemeinen Simulationsvorlesungen waren, haben diese Hörer keine Vorkenntnisse in den einschlägigen Softwarepaketen, insbesondere in MATLAB.

Um es Studenten zu ermöglichen, an Hand der grundlegenden Lehrbeispiele wie z.B. dem Übertragungsglied erster Ordnung ein Verständnis für dynamisches Systemverhalten zu entwickeln, reicht allerdings die Frontalvorlesung nicht aus. Die Studenten müssen die Gelegenheit haben, mit diesen Grundelementen der Simulation zu experimentieren. Studenten ohne Kenntnisse der verwendeten Sprachen können dies allerdings erst nach Erlernen der Selben. Da jedoch eine profunde Ausbildung in Simulationssprachen nicht Ziel der Grundlehrveranstaltungen in Modellbildung und Simulation ist, kann nur eine

sehr schnelle und oberflächliche Einführung in diese in die Präsenzlehre integriert werden.

So steht den Studenten ihre Unkenntnis der Sprache beim Versuch mit den Beispielen zu experimentieren im Weg, was auch den Interessierteren unter ihnen meist die Motivation nimmt, sich mit dem Stoff der Lehrveranstaltungen mehr als unbedingt notwendig zu beschäftigen. Ziel der Autoren war es daher, den Studenten dies Beispiele in eine Art und Weise zugänglich zu machen, die sowohl für Anfängern als fortgeschrittenen Anwender attraktiv ist.

2 E-learning an der TU Wien

Wie auf den meisten Universitäten waren die Anfänge des E-Learning an der TU Wien auch Insellösungen. Besonders an den Instituten für Informatik gab es in Folge der großen Studentenzahlen und dem großen administrativen Aufwand durch Programmierübungen eine Anzahl an rudimentären Systemen, die Dokumente für Hörer einzelner Vorlesungen online zugänglich machten und die Verwaltung der von Studenten abzugebenden Programme ermöglichte. Auch existierten kleine online Dienste zur Prüfungsverwaltung.

Alle diese frühen Formen des e-learning waren allerdings auf einzelne Lehrende, Abteilungen oder Institute beschränkt, und genau auf die Bedürfnisse einiger weniger handelnder Personen - und selten auf die Bedürfnisse der Studenten - abgestimmt.

2.1 TUWIS++

Ab dem Wintersemester 2003 wurde an der TU Wien schließlich die Funktionalität des sogenannten „Lehrzielkatalog"(ein rudimentäres elektronisches Vorlesungsverzeichnis) und der Plattform „Sides-4mi" (füür den Kontakt zwischen Lehrenden einer LVA und den Besuchenden Studenten bzw. die LVA Evaluation), die beide von einer Fremdfirma konzipiert und betreut wurden, in das TUWIS (TU Wien InformationsSystem) integriert, das zuvor primär der Internen Kommunikation bzw. der Verwaltung diente.

Da dieses TUWIS++ genannte System eine Eigenentwicklung der TU Wien (auf Basis der ZOPE[1] Umgebung) war, wurden so auch die bis dahin vorhandenen Sicherheitsbedenken im Bezug auf die Prüfungsverwaltung (vor allem die elektronische Benotung) ausgeräumt. Diese deutliche Erleichterung der Verwaltung führte zu einer sehr schnellen Akzeptanz des TUWIS++ durch die Lehrenden. Es wurde daher auch sehr schnell um Diskussionsforen und Dateiverwaltungen für die einzelnen Lehrveranstaltungen erweitert.

[1] http://www.zope.org

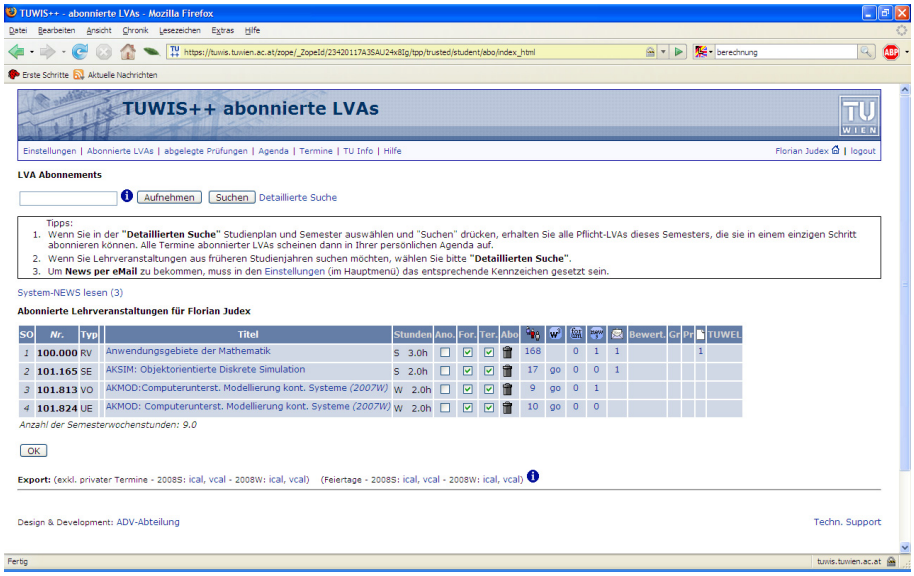


Abbildung 1: TUWIS++ aus der Sicht der Studenten

2.2 TUWEL

2006 nahm schließlich die TU Wien E-learning Plattform - eine Version der Open-Source Plattform Moodle[2] ihren Betrieb auf. Sie verwendet die gleiche Authentifizierung wie das TUWIS++ System, und Verfügt über Schnittstellen für den Import von LVA Teilnehmern und den Export von Noten, was Zweigleisigkeiten verhindert.

TUWEL als designierte E-Learning Plattform hatten allerdings von Anfang das Problem, dass während die mit E-Learning noch nicht vertrauten User der Meinung waren, dass das System erst mit Hilfe der erfahrenen Anwender von den vorhandenen Kinderkrankheiten befreit werden sollte, von deren Seite kaum Bereitschaft kam, ihre Nischenlösungen zu Gunsten von TUWEL aufzugeben.

Ein Aufruf des damaligen Vizerektors für Lehre, dem TUWEL System eine Chance zu geben, fiel mit der Eingangs erwähnten Umstellung der Curricula zusammen, so dass die E-Learning Unterstützung der Simulations LVAs beschlossen wurde.

3 Das Interface

In der zu diesem Zeitpunkt aktuellen MATLAB Release 2006a war noch der MATLAB Webserver enthalten, der es ermöglicht, über einem passenden Webserver wie z.B. Apache MATLAB Scripts mit vom Webserver übernommenen Eingabeparametern berechnen zu lassen und Ergebnisse über den Webserver auszugeben. Um ein Script für den Webserver zu adaptieren, müssen nur die in einer einzigen Variable vom Typ „struct" übergebenen Eingangsparameter in die benötigten Typen umgewandelt werden.

[2] http://www.moodle.org

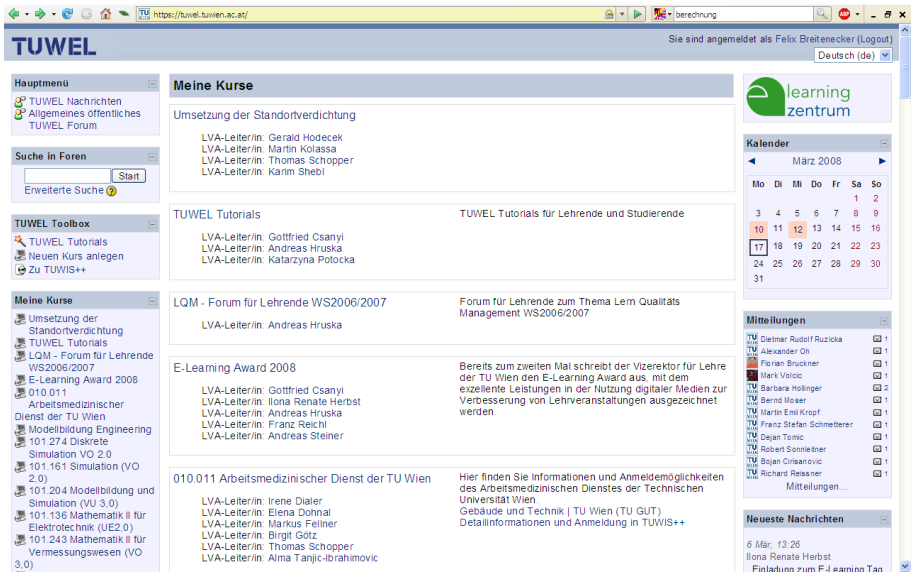


Abbildung 2: TUWEL

Dazu wurde von der Firma „Die Drahtwarenhandlung"[3] ein PHP Interface entwickelt, dass es auch kaum mit PHP bzw. HTML vertrauten Usern erlaubt, ihre MATLAB Inhalte auf diese Weise über das Netz zu veröffentlichen. Dazu müssen nur auf ein Verzeichnis am Server eine Handvoll Dateien abgelegt werden, aus denen dann eine wie in Abbildung 4 ersichtliche Web-Page generiert wird. Diese Web page ist in mehrere Abschnitte unter-

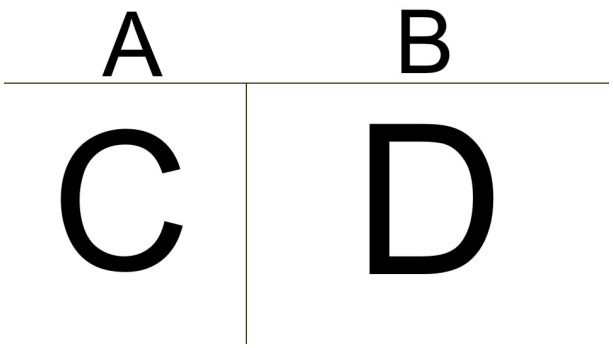


Abbildung 3: Die Aufteilung des Interface

teilt, die jeweils einzelnen Dateien entsprechen. Abbildung 3 zeigt diese Struktur, wobei die einzelnen Teile wie folgt sind

A Eine Plain-Text Datei mit einer kurzen Beschreibung des entsprechenden Themenblocks, die genauso wie die

B Namen der einzelnen Beispiele immer angezeigt werden. Jedes Beispiel hat sein eigenes Verzeichnis, in dem sein Name ebenfalls als Textdatei abgelegt ist.

C Enthält die Detailinformation zum jeweiligen Beispiel. Um z.B. Illustrationen zu ermöglichen, kann in dieser Datei beliebiger HTML Code stehen. Die Eingabefelder werden über kleine PHP Elemente generiert, die der Ersteller des Beispiels allerdings

[3] http://simulation.drahtwarenhandlung.at

nur aus einer Vorlage kopieren muss, ohne sich mit der Funktionalität zu beschäftigen. Es stehen textuelle Eingabefelder sowie Auswahlmenüs zur Verfügung.

D Der Ausgabe der Berechnungen. Es ist möglich Text (inklusive Fehlermeldungen) oder Graphik auszugeben. Ist beides zugleich erwünscht, muss der Text im Skript in die Graphik eingefügt werden. Für die Ausgabe ist jeweils nur eine weitere Zeile MATLAB Code nötig, die der Anwender aus einer Vorlage übernehmen kann.

Wenn es vom Benutzer gewünscht wird, kann in C zusätzlich ein Link auf das MATLAB Script dargestellt werden, der das Ansehen bzw. Herunterladen des Skripts ermöglicht. Das erlaubt Benutzern mit größerer MATLAB Erfahrung, neben dem Beispiel auch seine Implementierung zu sehen, und auf dem eigenen Rechner weiter zu verwenden.

Da das Webinterface nicht direkt ins TUWEL eingebunden werden konnte, sondern von dort nur verlinkt ist, bestand die Firma Mathworks auf einen Passwortschutz, um das System vor dem Zugriff von Benutzern, die nicht Angehörige der TU Wien sind, zu schützen. Dieser wurde auch über das PHP Interface implementiert - der Benutzer muss nur eine Textdatei mit Benutzern und Passwörtern im passenden Verzeichnis ablegen. Die Studenten beziehen die Passwörter über das TUWEL System.

3.1 Beispiel

Eines der typischen Beispiele für die Anwendung des Interface ist das Pendel mit beweglicher Aufhängung. Es besitzt mehr Freiheitsgrade, und keine hinreichend einfache analytische Lösung.

Wie man in Abbildung 4 sieht, hat hier der Student die Möglichkeit, einen Teil der Parameter und Anfangswerte selber zu wählen, und bekommt Position und Geschwindigkeit von Pendel und Aufhängung graphisch dargestellt. Im Beispieltext wird er darauf hingewiesen, dass das Verhältnis der Massen zueinander das Systemverhalten zentral beeinflusst, und kann an dem Modell verschiedenste Kombinationen nach Gutdünken ausprobieren.

4 Erfahrungen

Die Qualität der von den Studenten im Rahmen der zur LVA gehörenden Übung erstellten Modelle und deren Implementierung ist durch den Einsatz von MATLAB Code am Webserver deutlich erhöht worden, besonders auch dadurch, das MATLAB für Studenten der TU Wien zu eine annehmbaren Preis für die Nutzung daheim zur Verfügung steht.

Der Einsatz des Webinterfaces machte den Vortragenden deutliche flexibler, da zusammen mit im PDF-Format am Server gespeicherten Folien beinahe jeder auf der Universität vorhandene Rechner verwendete werden konnte. Außerdem war die Stabilität des Webservers mehr als zufriedenstellend - Einzelplatz Installationen von MATLAB vor allem unter

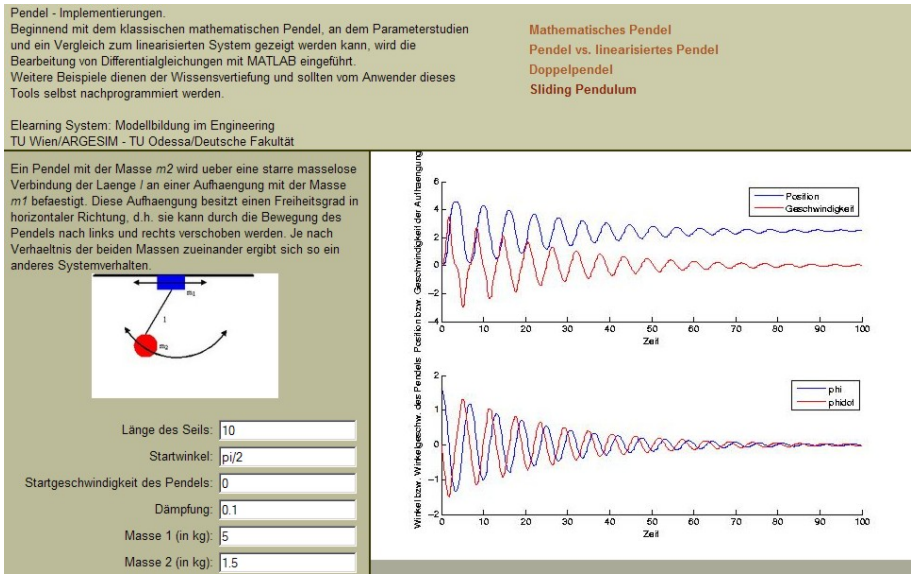


Abbildung 4: Beispiel-Pendel mit Beweglicher Aufhängung

Windows Betriebssystemen sind auch bei gut getesteten Programmen leider manchmal instabil.

Da sich MATLAB nicht nur für Modellbildung und Simulation eignet, ist der Server auch in der Grundlehre im Einsatz. Mittlerweile stehe für die Studenten der Geodäsie und Geoinfomatik im Rahmen der Vorlesungen Mathematik I und Mathematik II über 90 Beispiele aus verschiedensten Gebieten, von der Linearen Algebra bis zur Interpolation, zur Verfügung.

Das Projekt des E-Learning mit MATLAB wurde außerdem im Rahmen des TU Wien E-Learning Awards mit dem zweiten Platz und damit einem Geldpreis bedacht.

Iterative evidente Interpolation und ihre Anwendung zur Nachbildung von Abgasemissionen

Dipl.-Ing. Jurij Schmidgal, IEE-AT, Uni-Kassel
sch@at.e-technik.uni-kassel.de
Dr.-Ing Thomas Winsel, IEE-AT, Uni-Kassel
thw@at.e-technik.uni-kassel.de
Prof.-Dr.-Ing. Heinz J. Theuerkauf, IEE-AT, Uni-Kassel
the@at.e-technik.uni-kassel.de

Zusammenfassung

Die steigende Anzahl frei variierender Einflussgrößen bei der Regelung aktueller Verbrennungsmotoren sowie die stets knapp bemessene Anzahl an Versuchen zur Gewinnung von Messdaten beeinflusst die Einsatzmöglichkeit datengetriebener Modelle in zunehmendem Maße. Zur obligatorischen Validierung der Modellfunktionen dringend benötigte Messdaten stehen oftmals nicht in ausreichender Menge zur Verfügung.

Dem hier vorgestellten Verfahren genügt bereits eine geringe Anzahl von Messdaten, aus denen in einem iterativen Prozess zusätzliche Interpolationspunkte ermittelt werden, die sich evident in ihre unmittelbare Umgebung einfügen. Dadurch entsteht eine ausreichende Anzahl an Zwischenpunkten, um beispielsweise das Training eines neuronalen Netzes erfolgreich durchführen zu können. Hierbei wird eine hohe Approximationsgüte an den Messstellen sichergestellt und in den Zwischenbereichen ein plausibler Verlauf angestrebt.

Als Anwendung für dieses Verfahren wird der stark nichtlineare Verlauf der Stickoxidemission eines großvolumigen Ottomotors mit Direkteinspritzung echtzeitfähig nachgebildet.

Die Leistungsfähigkeit des vorgestellten Ansatzes zur NO_x-Simulation wird anhand von Messfahrten an einem hochdynamischen Verbrennungsmotorenprüfstand validiert. Die hierfür eingesetzte Abgasanalysenanlage gestattet eine Messung von Effekten, die innerhalb einzelner Arbeitszyklen stattfinden.

1 Einleitung

Zur Unterstützung bei der Kalibrierung von Steuergeräten ist heutzutage eine HiL-gestützte Simulation nahezu unerlässlich, zumal hierdurch Kosten und Umfang erheblich minimiert werden können. Eingesetzt werden hierfür häufig datengetriebene Modellansätze, die das Prozessverhalten des Verbrennungsmotors und seiner Umgebung in Echtzeit simulieren. Kern eines solchen Modells bildet meistens eine Funktion, die anhand der tatsächlich durchgeführten Messung parametriert wird. Es erweist sich oftmals als schwierig - besonders bei einem stark nichtlinearen Prozess - die geeignete Basisfunktion zu finden, mit der das Prozessverhalten samt aller Effekte und Wechselwirkungen im ausreichenden Maße wiedergegeben werden kann. In der Praxis werden häufig Polynomfunktionen niedrigen Grades eingesetzt. Mit steigendem Grad solcher Polynomfunktionen lassen sich die Funktionswerte an den gemessenen Stützstellen zwar besser approximieren, dabei entstehen aber unerwünschte Oszillationen in den Zwischenbereichen. Gutmütiger verhalten sich häufig Funktionsapproximatoren auf Basis neuronaler Netze. Auch bei stark nichtlinearen Prozessen weisen sie eine hohe Approximationsfähigkeit auf. Mit steigender Anzahl der Neuronen in der verdeckten Schicht treten zwar ebenfalls Unregelmäßigkeiten bei der Interpolation auf, was aber mit Hilfe von zusätzlichen dort platzierten Punkten verhindert werden kann, die schließlich zum Training des Netzes mitverwendet werden.

Das im Kapitel 2 vorgestellte Verfahren einer diskreten evidenten Interpolation erzeugt in einem iterativen Prozess eine beliebige Anzahl von Stützstellen, die ein möglichst plausibles Verhalten des zugrunde liegenden Prozesses wiedergeben. Die Leistungsfähigkeit des Verfahrens ist insbesondere bei der Verwendung in einem mehrdimensionalen Eingangsraum und beliebig verteilten Messdaten gegeben.

Eine Anwendung des Verfahrens ist im Kapitel 3 beschrieben. Dort wird ein Modellansatz zur Nachbildung der NO_x-Konzentration vorgestellt.

Die Ergebnisse im Kapitel 4 bestätigen die Leistungsfähigkeit des vorgestellten Verfahrens und zeigen die Simulationsergebnisse des NO_x-Modells.

2 Verfahren einer diskreten, evidenten Interpolation (DEI)

Aufgrund der geringen Anzahl an Messpunkten werden vom vorgestellten Verfahren in einem iterativen Prozess zusätzliche Interpolationspunkte erzeugt, die sich evident in ihre unmittelbare Umgebung einfügen. Dadurch entsteht eine ausreichende Anzahl an Zwischenpunkten, um anschließend das Training eines neuronalen Netzes erfolgreich durchführen zu können. Hierbei wird eine hohe Approximationsgüte an den Messstellen sichergestellt und in den Zwischenbereichen ein plausibler Verlauf angestrebt.

Gegeben sei ein Eingangsraum $U = \left\{ u_i \in R^{Nu} : i = 1, \cdots, M \right\}$, mit N_u Dimension und M Anzahl der Messpunkten. Der zugehörige Prozessausgang sei

$$y_i := f_p(u_i) \in R \tag{1}$$

wobei f_p unbekannt ist.

Durch eine einmalige Durchführung des nachfolgend beschriebenen Verfahrens wird ein zusätzlicher Punkt u_{M+1} erzeugt und der dazugehörige Funktionswert

$$y_{M+1} = f_{DEI}(u_{M+1}) \tag{2}$$

wird nach dem Kriterium einer evidenten Interpolation gebildet (vg. Definition 1) und befindet sich innerhalb einer konvexen Hülle, die die Menge aller Messpunkte u_i umschließt.

Definition 1: Eine Interpolationsfunktion $f_{\text{int}} : u \in R^{N_u} \rightarrow y \in R$ ist evident, wenn der Funktionsverlauf im Interpolationsbereich ohne unerwartete Oszillationen erfolgt.

Das Verfahren unterteilt sich in drei Schritte:

2.1 Bestimmung von Tangentialebenen an den Stützstellen

Im ersten Schritt werden zu jedem Punkt u_i eine Tangentialebene

$$v_i(u) = y_i + \frac{\partial f_p(u)}{\partial u} \cdot (u - u_i) \tag{3}$$

bestimmt. Die partiellen Ableitungen $\frac{\partial f_p(u)}{\partial u}$ werden mit Hilfe der natürlichen Nachbarn (vgl. Definition 2) $u_i^j \in U_i^j \subseteq U$ und den dazugehörigen Funktionswerten $y_i^j = f_p(u_i^j)$ mit $j = 1,\ldots,N_n^i$, wobei N_n^i Anzahl der natürlichen Nachbarn des Punktes u_i ist, abgeschätzt zu $\frac{\partial f_{DEI}(u)}{\partial u} \approx \frac{\partial f_p(u)}{\partial u}$.

Definition 2: Zwei Punkte sind natürliche Nachbarn, wenn sie einen gemeinsamen Thiessen-Scheitelpunkt besitzen, wie in [4] beschrieben.

Die Abschätzung erfolgt mit Hilfe des Least-Squares-Verfahrens, indem man ein lineares Gleichungssystem der Form:

$$(y_i^j - y_i) = \frac{\partial f_{DEI}(u)}{\partial u} \cdot (u_i^j - u_i) \tag{4}$$

aufstellt und unter Minimierung der Fehlerquadrate die partielle Ableitungen $\frac{\partial f_{DEI}(u)}{\partial u}$ ermittelt. Eine Tangentialebene kann nur eindeutig beim Vorhandensein mindestens N_u Nachbarn ($j \geq N_u$)

bestimmt werden. Im Grenzfall $j = N_u$ dürfen keine zwei oder mehrere Nachbarn in einen Punkt zusammenfallen. Für den Fall, dass eine Anhäufung der Nachbarpunkte in einem Gebiet vorkommt, wird der Einfluss der einzelnen Nachbarn zusätzlich mit einem Gewichtsfaktor versehen. Die Ermittlung der Gewichtungsfaktoren erfolgt ähnlich wie auch bei der Methode Voronoi-Interpolation, vgl. [4].

An dieser Stelle wird die Methode zur Ermittlung der Gewichte anhand eines Beispiels für $N_u = 2$ beschrieben:

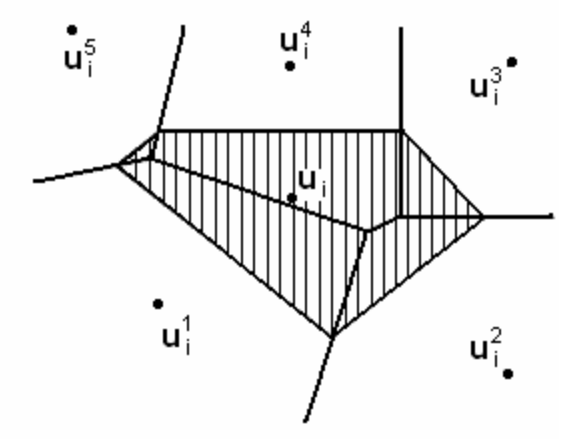


Abb. 1: Voronoi-Interpolation

Man bildet zuerst die Voronoi-Zellen mit Bezug auf die natürlichen Nachbarn u_i^j des Punktes u_i, ohne Berücksichtigung derselben. Die Flächen der einzelnen Zellen werden mit F_i^j bezeichnet. Anschließend bildet man die Voronoi-Zellen angewandt auf die Elemente $\{ u_i^j, u_i \}$. Die in Abb. 1 schraffierte Fläche der Voronoi-Zelle um den Punkt u_i entspricht dabei dem Wert F_i. Die Gewichte der Punkte u_i^j, die die Neigung der Tangentialebene beeinflussen, errechnen sich dann mit

$$g_i^j = \frac{F_i^j \cap F_i}{F_i}. \tag{5}$$

2.2 Bildung optimaler Punkt-zu-Punkt-Verbindungen

Im zweiten Schritt werden nach dem im [5] beschriebenen Verfahren *Delaunay-Triangulation* für mehrdimensionalen Raum Simplexe $\Delta_S \subseteq U$ mit $s = 1,\ldots,N_S$ erzeugt, wobei N_S die Anzahl der Simplexe ist. Mit den Eigenschaften:

- Punkte eines Simplex bilden eine konvexe Hülle, die eine leere Menge umschließt,
- Der Durchschnitt zwei verschiedener Simplexe ist entweder leer, ein gemeinsamer Eckpunkt, eine gemeinsame Kante, eine gemeinsame Dreiecksfläche oder im u-Dimensionalen ein gemeinsames $N_u - 1$ Polytop.
- Die Verbindungslinien zwischen den Punkten eines Simplex weisen einen maximalen Innenwinkel auf.

Die Bildung einer Delaunay-Triangulation ist eigentlich nur ein Zwischenschritt zur Bildung optimaler Punkt zu Punkt Verbindungen $D_S^k \subseteq \Delta_S$ mit $k = 1,\ldots,\sum_{i=1}^{N_u} i$, die sich aus den Kanten der Simplexe ableiten lassen. Die doppelten Verbindungen, die aus zwei benachbarten Simplexen stammen, werden aussortiert. Nach dem Aussortieren bleiben die Einfachverbindungen $D_l \subseteq D$ mit $l = 1,\ldots,N_D$, wobei N_D die Anzahl aller Einfachverbindungen ist. Die Elemente einer Verbindung sind $\{ u_l^1, u_l^2 \} \in D_l$ und werden zukünftig als Endpunkte einer Verbindung bezeichnet.

Jeder Endpunkt besitzt eine Tangentialebene $v_l^1(u)$ und $v_l^2(u)$, die aus der Menge der Tangentialebenen v_i (vgl. Kap. 2.1) sind.

Im Weiteren wird für jede Verbindung mit den dazu gehörigen Tangentialebenen an jeweils beiden Endpunkten eine Richtungsableitung zu gegenüber liegendem Endpunkt bestimmt

$$a_{l,1\to 2} = grad(v_l^1(u)) \cdot \frac{u_l^1 - u_l^2}{\left|u_l^1 - u_l^2\right|}, \quad bzw. \quad a_{l,2\to 1} = grad(v_l^2(u)) \cdot \frac{u_l^2 - u_l^1}{\left|u_l^2 - u_l^1\right|}. \tag{6}$$

So besitzt jeder der Verbindungen $D_{k,l}$ vier charakteristische Merkmale:

- Funktionswerte an den Endpunkten $y_l^1 = f(u_l^1)$, $y_l^2 = f(u_l^2)$ und
- Richtungsableitungen in Richtung des gegenüber liegenden Punktes $a_{l,1\to 2}, a_{l,2\to 1}$.

Jetzt wird eine parametrierte Kurve gesucht, die außer der Erfüllung der vier Randbedingungen ohne unerwünschte Oszillationen im Zwischenbereich verläuft:

$$f: \ \left[u_l^1, u_l^2\right] \to R^{Nu}, \quad t \mapsto f(t). \tag{7}$$

Für die gestellte Aufgabe eignet sich sehr gut eine Bezier-Kurve dritten Grades

$$f(t) = f_B(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} p_i, \quad t \in [0,1] \tag{8}$$

wobei

$$p_0 = y_l^1, \quad p_3 = y_l^2, \quad p_1 = \frac{\left\|a_{l,1\to 2}\right\|}{3} + p_0 \quad und \quad p_2 = \frac{\left\|a_{l,2\to 1}\right\|}{3} + p_3.$$

2.3 Erzeugung eines zusätzlichen Punktes

Im dritten und letzten Schritt des Verfahrens wird ein neuer Punkt u_{M+1} erzeugt, indem die längste Verbindung $D_L = \max\left\|D_l\right\|$ in der Mitte geteilt wird, wobei

$$u_{M+1} = \frac{u_L^1 + u_L^2}{2}. \tag{9}$$

Der Funktionswert der Bezier-Kurve $f_B(u_{M+1})$ an der Stelle u_{M+1} bildet somit einen neuen zusätzlichen Punkt, der zu der Menge der Punkte u_i zugefügt wird. Als nächstes wird zu diesem Punkt eine neue Tangentialebene

$$v_{M+1}(u) = f_B(u_{M+1}) + \frac{\partial f_{DEI}(u)}{\partial u_k} \cdot (u - u_{M+1}) \tag{10}$$

bestimmt.

Um eine Tangentialebene an der Stelle u_{M+1} eindeutig bestimmen zu können, ist ein lineares Gleichungssystem mit $N_u + 1$ Gleichungen notwendig. Die ersten beiden Gleichungen sind durch den Funktionswert $f_B(u_{M+1})$

$$v_{M+1}(u_{M+1}) = f_B(u_{M+1}) \tag{11}$$

sowie die Richtungsableitung in Richtung $\frac{u_L^2 - u_{M+1}}{\left|u_L^2 - u_{M+1}\right|}$ gegeben. Die Richtungsableitung entspricht dabei dem Wert der ersten Ableitung der Bezier-Kurve $f_B(u_{M+1})$ an der Stelle u_{M+1}

$$grad(v_{M+1}(u)) \cdot \frac{u_L^2 - u_{M+1}}{\left|u_L^2 - u_{M+1}\right|} = f_B^{'}(u_{M+1}) . \tag{12}$$

Im Fall $N_u > 1$ sind weitere $Nu - 1$ Gleichungen notwendig und es werden dazu $Nu - 1$ normierte Richtungsvektoren an der Stelle u_{M+1} mit den Eigenschaften:

$$a_1(u) \perp a_2(u) \perp, \ldots, \perp a_{N_u - 1}(u) \perp \frac{u_L^2 - u_{M+1}}{\left|u_L^2 - u_{M+1}\right|} \tag{13}$$

gebildet. Die dazugehörigen Richtungsableitungen A_k mit $k = 1, \ldots, N_u - 1$ werden aus den Richtungsableitungen an den Stellen u_L^1 und u_L^2 mit den Tangentialebenen $v_L^1(u)$ und $v_L^2(u)$ in die Richtung der Richtungsvektoren a_k gemittelt

$$A_k = \frac{grad(v_L^1(u)) \cdot a_k(u) + grad(v_L^2(u)) \cdot a_k(u)}{2} \tag{14}$$

so dass die dazugehörigen Gleichungen lauten

$$grad(v_{M+1}(u)) \cdot a_k(u) = A_k . \tag{15}$$

Durch das Lösen des Gleichungssystems aus den Gleichungen (11), (12) und (15) werden die unbekannten Koeffizienten der neuen Tangentialebene $v_{M+1}(u)$ bestimmt.

Durch die Bildung eines neuen Punktes u_{M+1} entstehen gleichfalls neue Verbindungen. Zunächst entstehen aus der Verbindung D_L zwei neue Verbindungen D_L^1 und D_L^2 mit den Elementen $\left\{u_L^1, u_{M+1}\right\} \in D_L^1$ und $\left\{u_{M+1}, u_L^2\right\} \in D_L^2$. Anschließend werden weitere Verbindungen wie folgt gebildet: $D_L \subseteq \Delta_m$ mit $m = 1, \ldots, N_m$, N_m ist die Anzahl der Simplexe, die eine gemeinsame Kante D_L enthalten. Die neuen Verbindungen D_p mit $p = 1, \ldots, (N_u - 1)N_m$ entstehen aus den Elementen $\left\{ u_{M+1}, u_p \right\} \in D_p$, mit $u_p \in \left\{\left\{ \Delta_1 \cup, \ldots, \cup \Delta_m \right\} \sim D_L \right\}$.

Durch Widerholen des dritten Schrittes können weitere zusätzliche Punkte iterativ gebildet werden.

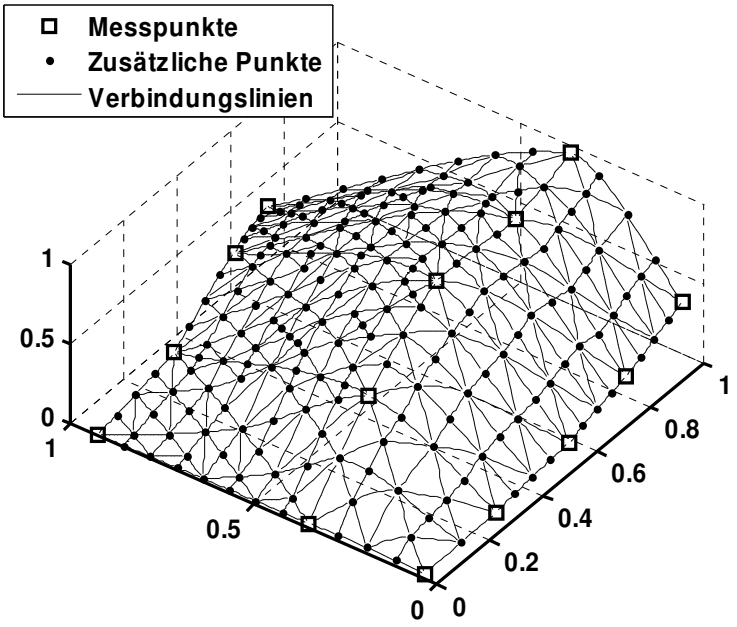


Abb. 2: Ergebnis des Verfahrens einer *diskreten, evidenten Interpolation* (DEI)

3 NO_x-Modell

Bei den Stickoxiden unterscheidet man zwischen den Stickstoffmonoxiden NO und den Stickstoffdioxiden NO_2, wobei während einer ottomotorischen Verbrennung hauptsächlich die Stickstoffmonoxide entstehen. Aus diesem Grund wird der Anteil der Stickstoffdioxide NO_2 bei der nachfolgend beschriebenen Simulation nicht berücksichtigt.
Laut der Veröffentlichungen [1] und [2] sind die lokalen Faktoren

- Konzentrationen der Reaktionsgase O_2 und N_2 sowie
- Verbrennungstemperatur

primäre Einflussgrößen bei der Bildung von Stickstoffmonoxid NO.

Die genannten Reaktionsverhältnisse sind in der Praxis am Motorenprüfstand nur mit unverhältnismäßig hohem Aufwand zu ermitteln. Mit Hilfe umfassender und sehr rechenzeitintensiver 3D-Simulation gelingt es zwar, die Prozesse der Verbrennung auf rein physikalisch-chemischer Basis mit lokaler Auflösung zu bestimmen. Man beschränkt sich aber auch dort auf die Haupteffekte und trifft starke Vereinfachungen der chemischen Reaktionsgleichungen.

Das Ziel dieser Arbeit ist die Ableitung von so genannten *sekundären* Einflüssen, die sich auf die oben erwähnten primären Einflüsse direkt auswirken und somit indirekt zur Bildung des Stickstoffmonoxides NO beitragen. Diese sind

- eingeschlossene Kraftstoffmenge pro Arbeitsspiel $m_{K,AS}$,
- Dauer des Arbeitsspiels t_{AS},
- Zeitpunkt der Zündung a_{ZW} und
- stöchiometrisches Luft-Kraftstoff-Verhältnis λ.

Somit ergibt sich die funktionale Abhängigkeit des Prozesses der NO_x-Konzentration zu

$$NO_x := f_{NO_x}(m_{K,AS}, t_{AS}, a_{ZW}, \lambda). \tag{16}$$

Der hier vorgestellte Modellansatz beschränkt sich auf die Betriebspunkte des Motors bei einem Luft-Kraftstoff-Verhältnis $\lambda \approx 1$. Somit lautet die Abgleichvorschrift

$$NO_{x,m} := f_{NO_x,m}(m_{K,AS}, t_{AS}, a_{ZW}) \approx NO_x \quad \forall \quad \lambda \approx 1. \tag{17}$$

Die für den Abgleich der Modellfunktion $f_{NO_x,m}$ benötigten Messdaten NO_x werden am Prüfstand an stationären Betriebspunkten aufgenommen. Die konvexe Hülle, welche die Messdaten NO_x umschließt, definiert den Gültigkeitsbereich des Modells. Mit Hilfe des im Kapitel 2 vorgestellten DIE-Verfahrens werden zusätzlichen Sollwerte $NO_{x,z}$ generiert, mit deren Hilfe dann zusammen mit den Sollwerten NO_x die Modellfunktion $f_{NO_x,m}$ abgeglichen werden kann. Als Modellfunktion wird ein dreischichtiges Multi-Layer-Perceptron (MLP) der Form

$$f_{NO_x,m} := f_{NO_x,MLP} = B_2 + W_2 \tanh(B_1 + W_1 u) \tag{18}$$

verwendet, mit $u := \left[m_{K,AS}, t_{AS}, a_{ZW}\right]$ und einem geeignet dimensionierten Wichtungssatz $\{B_2, W_2, B_1, W_1\} \in W$, wie in [3] detailliert beschrieben. Um eine bessere Approximation an den tatsächlich gemessenen Stützstellen NO_x zu erreichen, werden diese beim Abgleich höher gewichtet. Der Abgleichalgorithmus arbeitet nach dem Prinzip einer nichtlinearen Optimierung und wird auch als Lernverfahren oder Training bezeichnet. Durch eine mehrfache zufällige Initialisierung des Wichtungssates W wird nach einem lokalen Minimum gesucht, das dem unbekannten globalen Minimum möglichst „nahe“ kommt.

4 Ergebnisse

Um das Verfahren der diskreten evidenten Interpolation (DEI) zu testen, wurden am Prüfstand zusätzliche Validierungsdaten $NO_{x,Val}$ aufgenommen, die im Verfahren bei der Erzeugung der zusätzlichen Punkte nicht verwendet wurden. Das DEI Verfahren wurde in diesem Beitrag auf einen Modellansatz mit einem dreidimensionalen Eingangsraum angewendet. Um das Ergebnis des Verfahrens anschaulich darzustellen, wurden aus der Menge der Messpunkte NO_x, zusätzlicher Punkte $NO_{x,z}$ sowie Validierungspunkte $NO_{x,Val}$, Punkte mit Arbeitspieldauer $t_{AS} \approx 120\, m\text{sec}$ ausselektiert. D.h. es wurde ein „Schnitt“ durch die Menge aller Daten gemacht. Jetzt kann die NO_x-Konzentration in Abhängigkeit der eingeschlossenen Kraftstoffmenge pro Arbeitsspiel im Zylinder $m_{K,AS}$ sowie dem Zeitpunkt der Zündung a_{ZW} in einer 3-D Darstellung, hier beispielhaft an einem Betriebspunkt bei $t_{AS} \approx 120\, m\text{sec}$, dargestellt werden (vgl. Abb. 3). Die Validierungspunkte bestätigen Eignung des Verfahrens.

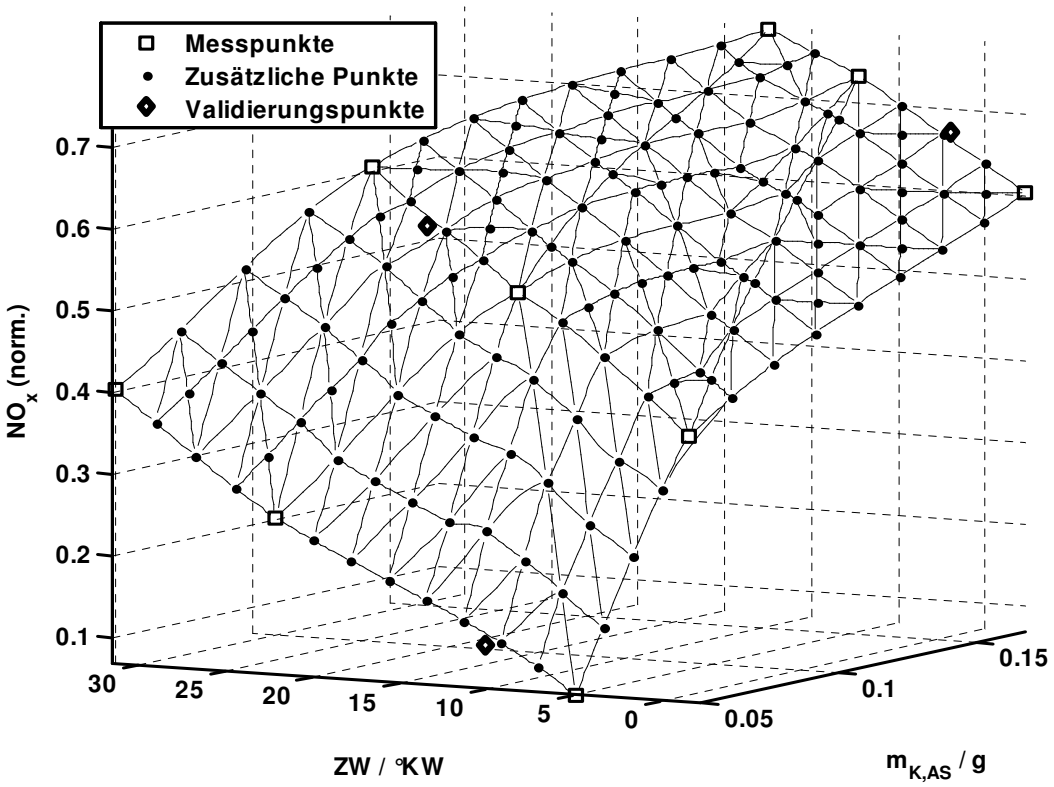


Abb. 3: Ergebnis des DEI Verfahrens bei $t_{AS} = 120\,m\sec$

In Abb. 4 werden an unterschiedlichen Betriebspunkten, definiert durch die Einflussgrößen $t_{AS}, m_{K,AS}, a_{ZW}$, Messung und Simulationsergebnisse des NOx-Modells abgebildet. Im linken Graf sind die tatsächlich gemessenen Punkte und im rechten eine zufällige Auswahl der zusätzlich erzeugten Punkten mit dem Ergebnis der Simulation dargestellt.

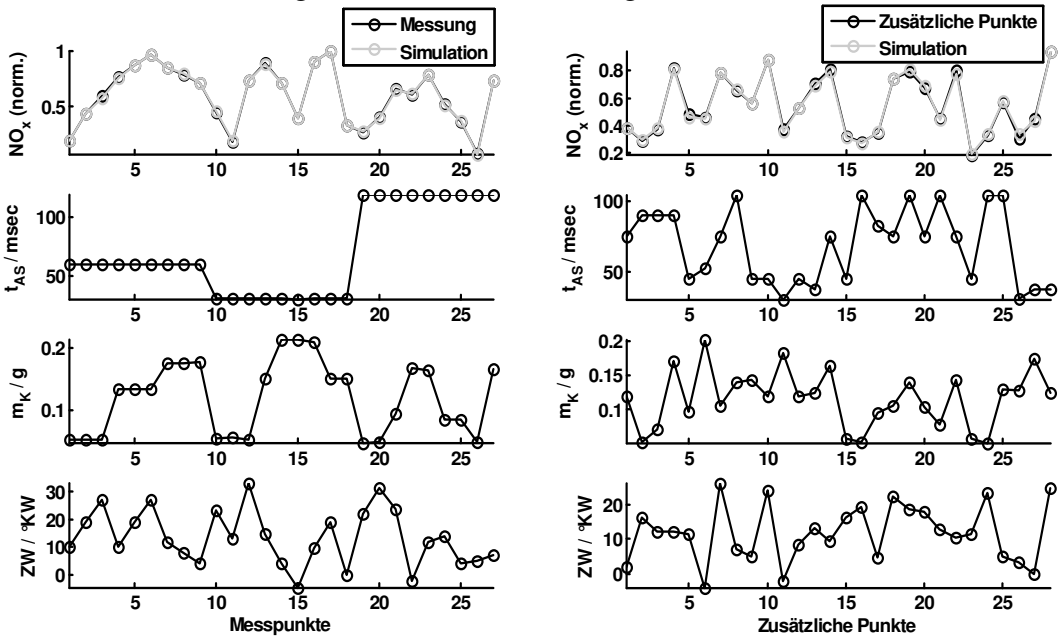


Abb. 4: Messung und Simulationsergebnisse an statischen Stützstellen

Das Ergebnis bestätigt eine gute Approximation des neuronalen Netzes an den tatsächlich gemessenen Stützstellen sowie einen plausiblen Verlauf im Interpolationsbereich.

5 Fazit, Ausblick

Mit dem vorgestellten Verfahren erfolgt im Prinzip ein gesteuertes Anlernen eines neuronalen Netzes mit Hilfe zusätzlich erzeugter Punkte im Interpolationsbereich. So lässt sich das Prozessverhalten der Stickoxidbildung in stationären Betriebspunkten auch bei minimalem Messaufwand gut nachbilden. Neuronale Netze weisen sich durch ihre Anpassungsfähigkeit erneut als geeignete Funktionsapproximatoren aus, mit denen bei stark nichtlinearen Prozessen eine hohe Approximationsgüte erreicht werden kann. Unplausibles Verhalten im Interpolationsbereich, besonders bei hoher Anzahl an Neuronen in der verdeckten Schicht, wird hierdurch zuverlässig verhindert.

Im weiteren Vorhaben wird das Verfahren der diskreten evidenten Interpolation (DEI) um die Erzeugung zusätzlicher Punkte auch im unmittelbaren Extrapolationsbereich ergänzt, die dann zum Training der neuronaler Netze verwendet werden können. Dadurch soll das Prozessmodell auch in engen Grenzen außerhalb des Gültigkeitsbereichs ein plausibles und evidentes Verhalten aufweisen.

Literatur

[1] Merker, G. P.; Stiesch, G.: Technische *Verbrennung Motorische Verbrennung* Teubner Verlag Stuttgart Leipzig, 1999

[2] Warnatz, J.; Maas, U.; Dibble, R. W.; *Verbrennung Physikalisch-Chemische Grundlagen*, (3. Auflage) Springer Verlag, 2001

[3] Winsel, T.: *Stabile neuronale Prozessmodelle,* Dissertation, VDI-Fortschritt-Berichte, Reihe 12, VDI-Verlag Düsseldorf, 2002

[4] http://de.wikipedia.org/wiki/Voronoi-Diagramm

[5] http://de.wikipedia.org/wiki/Delaunay-Triangulation

Simulation Abgasnachbehandlung: Ein modulares System zur Berechnung von Abgasanlagen

S. Dusemund, IAV GmbH
sandra.dusemund@iav.de
Dr. H. Schneider, IAV GmbH
hellfried.schneider@iav.de
Dr. K.-J. Langeheinecke, IAV GmbH
kay-jochen.langeheinecke@iav.de
A. Horn, IAV GmbH
anderas.horn@iav.de

Zusammenfassung

Komponenten zur Abgasnachbehandlung wie Dieselpartikelfilter und Katalysatoren sind aufgrund der immer strengeren Abgasgesetzgebung nicht mehr aus der Automobilindustrie wegzudenken. Es besteht dringender denn je die Notwendigkeit, diese Komponenten zu optimieren und effizienter zu gestalten.
Neben der verfahrenstechnischen Auslegung wird die Unterstützung durch numerische Simulation bei der Entwicklung von Steuergerätefunktionen und deren Bedatung sowie der Entwicklung neuer Diagnosestrategien immer bedeutender.
Im Folgenden wird ein modulares System zur Simulation monolithischer Abgaskatalysatoren (IAV KATsim) vorgestellt. Am Beispiel des SCR-Katalysators wird demonstriert, wie das eindimensionale Modell in verschiedene Simulationsumgebungen eingebunden und somit ein breites Feld an Anwendungsmöglichkeiten geschaffen werden kann.

1 Einleitung und Grundlagen

Neben innermotorischen Maßnahmen, die die Entstehung von Stickoxiden und Partikeln minimieren, sind Katalysatoren und Dieselpartikelfilter wesentliche Bestandteile von Abgasnachbehandlungskonzepten zur Einhaltung aktueller und zukünftiger Abgasnormen. Zum Einsatz und Betrieb der Abgasnachbehandlungskomponenten im Fahrzeug müssen umfangreiche Funktionen im Motorsteuergerät bereitgestellt werden. Bei der Entwicklung neuer Steuergerätefunktionen sowie deren effizienter Bedatung gewinnen numerische Simulationsmethoden immer mehr an Bedeutung.

Im Verlauf der folgenden Darstellungen wird ein modulares System zur Simulation monolithischer Katalysatoren (IAV KATsim) vorgestellt. Dieses System basiert auf einem eindimensionalen, instationären Modell (im Folgenden als Kernmodell bezeichnet) zur Berechnung der thermodynamischen und chemischen Vorgänge im Katalysator und kann, je nach Anwendungsgebiet, in verschiedene Simulationsumgebungen eingebunden werden (siehe Abbildung 1).

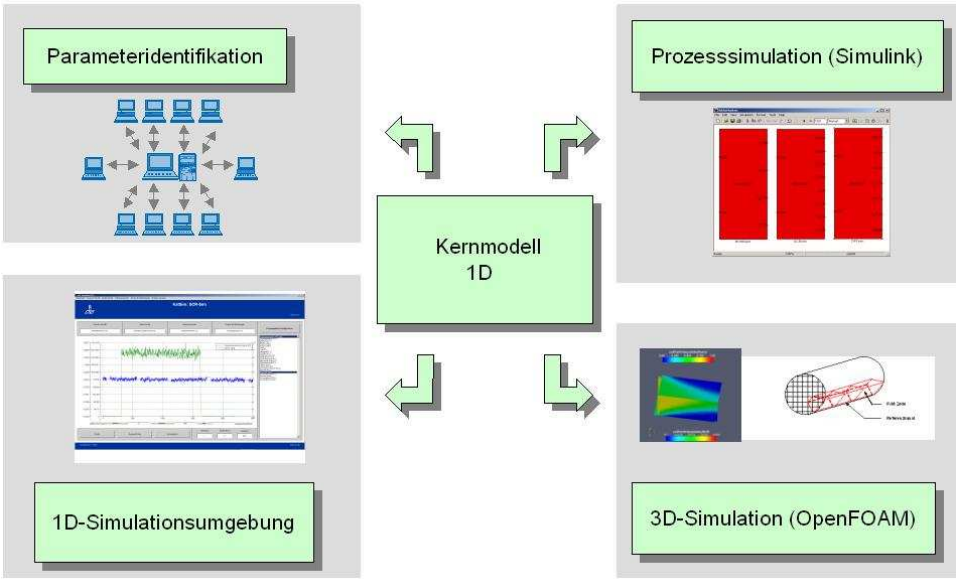


Abbildung 1: Einbindungsmöglichkeiten des Kernmodells in verschiedene Simulationsumgebungen

Die Anbindung an die Parameteridentifikation erlaubt die Ermittlung der Modellparameter auf Basis experimenteller Daten. Desweiteren steht eine graphische Benutzeroberfläche zur Verfügung, mit der eindimensionale, instationäre Berechnungen durchgeführt werden können. Durch die Einbindung des Kernmodells in MATLAB-Simulink können Prozesssimulationen durchgeführt werden. Insbesondere bietet sich die Möglichkeit, verschiedene Abgasnachbehandlungskomponenten zu kombinieren und Regelungsstrukturen aufzubauen. Zusätzlich steht eine dreidimensionale Simulationsumgebung zur Verfügung, mit der inhomogene Verteilungen untersucht werden können. Durch diese Flexibilität kann die numerische Simulation in Bereichen wie der Entwicklung von Steuergerätefunktionen und deren Bedatung als auch der Entwicklung von Diagnosestrategien sowie der verfahrenstechnischen Auslegung eingesetzt werden.

Die beschriebenen Möglichkeiten zur Nutzung des modularen Simulationssystems werden im folgenden am Beispiel des SCR-Katalysators („Selective Catalytic Reduction“) dargestellt.

Beim SCR-Verfahren wird eine Harnstoff-Wasser-Lösung (HWL) als Reduktionsmittel mittels einer Dosierpumpe vor dem Katalysator in die Abgasanlage eingebracht. Durch Thermolyse und Hydrolyse entsteht aus der HWL Ammoniak. Ammoniak adsorbiert auf der katalytischen Oberfläche des SCR-Katalysators und reagiert dort mit den im Abgas vorhandenen Stickoxiden. Dies geschieht maßgeblich durch die „Standard-SCR-Reaktion“

$$4NH_3 + 4NO + O_2 \rightarrow 4N_2 + 6H_2O \quad (1)$$

und die „schnelle SCR-Reaktion“

$$4NH_3 + 2NO + 2NO_2 \rightarrow 4N_2 + 6H_2O, \quad (2)$$

bei denen Stickstoff und Wasser als Produkte entstehen[1].
Das physikalische Kernmodell betrachtet stellvertretend für den gesamten Katalysator einen Referenzkanal (siehe Abbildung 2). Für den Festkörper (Solidphase) und die Gasphase werden die Energieerhaltungsgleichungen und die Stofferhaltungsgleichungen für die aktiven Abgaskomponenten unter Berücksichtigung der chemischen Umsetzungen numerisch gelöst. Gradienten in radiale Richtung werden dabei vernachlässigt[2].

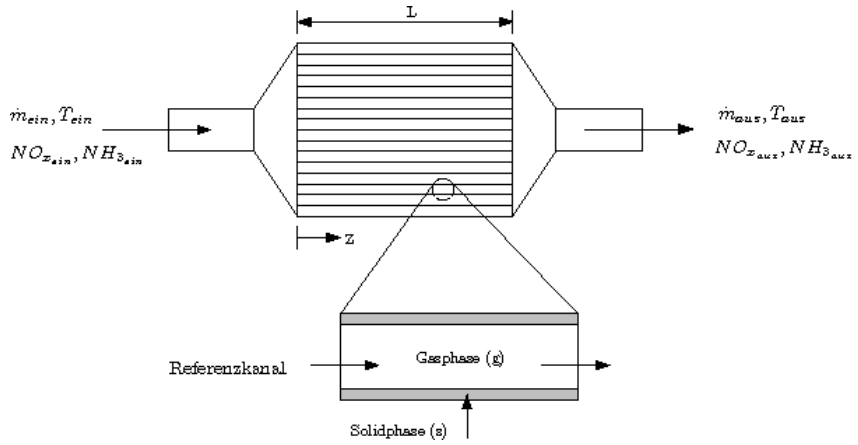


Abbildung 2: Darstellung des Referenzkanals für das physikalische Modell

2 Das modulare Simulationssystem

2.1 Die Parameteridentifikation

Die Bestimmung reaktionskinetischer Parameter ist hinsichtlich der Simulationsgüte besonders wichtig. Die Reaktionsraten r werden durch einen Arrheniusansatz beschrieben

$$r_j = Ae^{-\frac{E}{RT}} \prod_i x_i. \quad (3)$$

Dabei geben die reaktionskinetischen Parameter E und A an, ab welchem Energieniveau und mit welcher Geschwindigkeit die Reaktionen ablaufen.
Zur Ermittlung der Systemparameter werden verschiedene Optimierungsverfahren (z.B. genetische Algorithmen und konjugierte Gradientenverfahren) eingesetzt, die wahlweise

[1] Neben den beschriebenen Reaktionen finden weitere statt wie z.B. Oxidationsreaktionen von NH_3. Der in IAV KATsim implementierte Reaktionsmechanismus ist in Anhang A zu finden.
[2] Die Erhaltungsgleichungen sind in Anhang B angegeben.

miteinander kombinierbar sind.
Grundlage für die Parameteridentifikation sind experimentelle Daten aus Prüfstandsuntersuchungen unter definierten Randbedingungen. Die Bestimmung der reaktionskinetischen Parameter für die Adsorption und Desorption von Ammoniak (NH_3) im SCR-Katalysator erfolgt anhand isothermer Versuche am Synthesegasprüfstand. Hierbei wird der vorkonditionierte Katalysator zunächst definiert mit Ammoniak unter Abwesenheit von Stickoxiden (NO_x) be- und entladen.
Auf Basis der experimentellen Daten werden dann eine Anzahl n zufälliger Parametersätze mit den zu identifizierenden Größen (hier: E_{ads}, E_{des}, A_{ads} und A_{des}) in einem definierten Wertebereich erstellt. Zudem wird eine Zielfunktion, z.B. NH_3 am Katalysatoraustritt, bestimmt, nach der identifiziert werden soll. Es können auch mehrere Zielfunktionen definiert werden.
Für jeden Parametersatz wird die Simulation gestartet und der mittlere quadratische Fehler zur Zielfunktion, im Folgenden als „Fitness“ bezeichnet, berechnet. In den sogenannten „Turnieren“ treten immer zwei Parametersätze gegeneinander an, wobei der mit der besseren Fitness (der mit dem kleinsten Fehler) einer neuen Menge an Parametersätzen („Elterngeneration“) beigefügt wird. Nach dem Verfahren der simulierten binären Kreuzung (siehe [3] und [4]) werden danach aus jeweils zwei Elternparametersätzen zwei Kinderparametersätze erzeugt und der beschriebene Algorithmus wird erneut durchlaufen. Ändert sich die Fitness innerhalb einer bestimmten Anzahl von Identifikationsdurchläufen um nicht mehr als einen vorgegebenen Prozentsatz, terminiert der genetische Algorithmus.
Die Identifikationsroutinen sind auf einem Rechencluster umgesetzt. Hierbei übernimmt ein Rechner des Clusters die Funktion des „Masters“ und koordiniert die Vergabe einzelner Parametersätze an die Client-Rechner.

2.2 Eindimensionale Simulationsumgebung

Mit den ermittelten Parametern kann das Kernmodell bedatet werden.
Die eindimensionale Simulationsumgebung integriert das Kernmodell und stellt darüber hinaus eine graphische Benutzeroberfläche zur optimierten Bedienbarkeit zur Verfügung. Geometrische Daten (z.B. Abmessungen des Katalysators), Stoffwerte, Reaktionsparameter sowie Anfangs- und Randbedingungen können menügesteuert angegeben werden. Darüber hinaus stehen Schnittstellen bereit, um INCA[3]-Messungen und verschiedene andere Datenformate einzulesen.
Zusätzlich bietet die Oberfläche die Möglichkeit, die Ergebnisse der Simulation graphisch darzustellen sowie Messdaten und verschiedene Simulationen direkt miteinander zu vergleichen.
In Abbildung 3 sind Berechnungsergebnisse zur Adsorption und Desorption von NH_3 experimentellen Daten gegenüber gestellt. Gasförmiges NH_3 wird dabei in den konstanten Abgasmassenstrom vor einem unbeladenen SCR-Katalysator bei konstanter Temperatur eingebracht. Das Ammoniak diffundiert in den Washcoat und adsorbiert an freien Ober-

[3]INCA ist ein ETAS-Produkt zur Applikation von Steuergerätefunktionen.

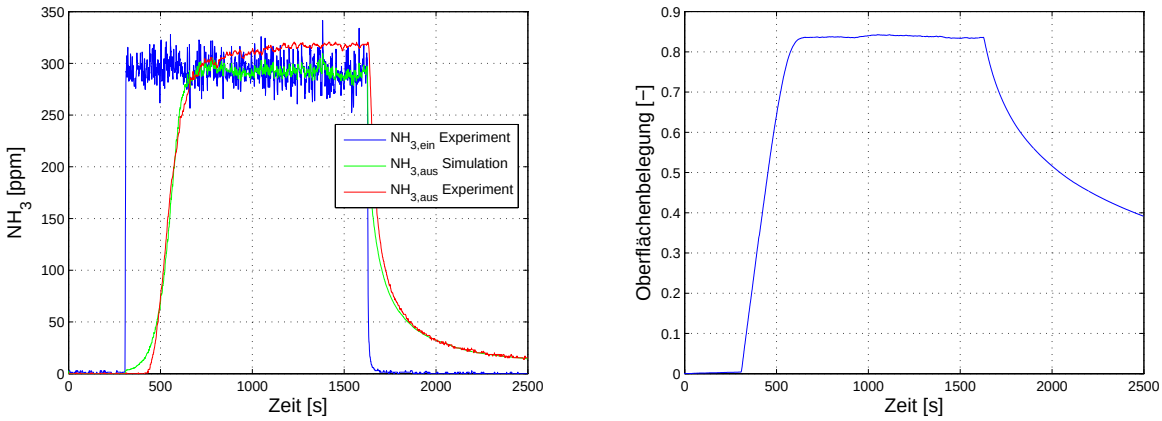


Abbildung 3: Simulation eines Adsorptions- und Desorptionsvorgangs unter Berücksichtigung der identifizierten reaktionskinetischen Daten

flächenplätzen. Die Oberflächenbeladung nimmt dabei zu, bis die Gleichgewichtsbeladung erreicht ist und damit Ammoniak am Katalysatorausgang entsprechend der Eintrittskonzentration „durchbricht“. Mit den identifizierten Reaktionsparametern kann der Verlauf der NH_3-Konzentration am Austritt sehr gut dargestellt werden.

2.3 Kopplung mit Simulink

Zur Entwicklung von Motorsteuergerätefunktionen und deren Bedatung kann das eindimensionale Kernmodell als S-Funktion in Simulink-Bibliotheken eingebunden werden. Neben Katalysatormodellen stehen thermische Modelle für Rohrleitungen und Modelle zur Beschreibung von Partikelfiltern zur Verfügung. Durch Kombination dieser Simulinkmodelle lassen sich verschiedene Konfigurationen von Abgasanlagen sowie die gegenseitige Beeinflussung der Komponenten untersuchen.

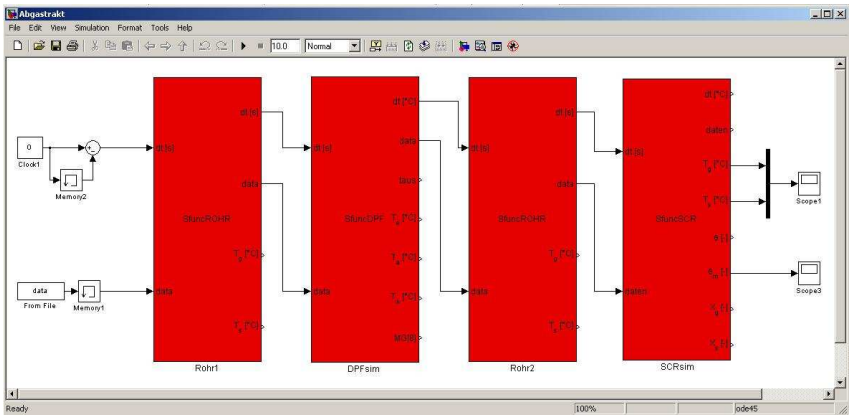


Abbildung 4: Nachbildung eines Abgastrakts bestehend aus Dieselpartikelfilter, Rohrstücken und SCR-Katalysator

Abbildung 4 zeigt eine Konfiguration bestehend aus Partikelfilter, SCR-Katalysator und zwei Rohrstücken. Die Auswirkungen einer Partikelfilterregeneration auf das Adsorptions- und Desorptionsverhalten des SCR-Katalysators werden in Abbildung 5 dargestellt.
Der Partikelfilter wurde mit 15g vorbeladen und eine Partikelfilterregeneration bei ent-

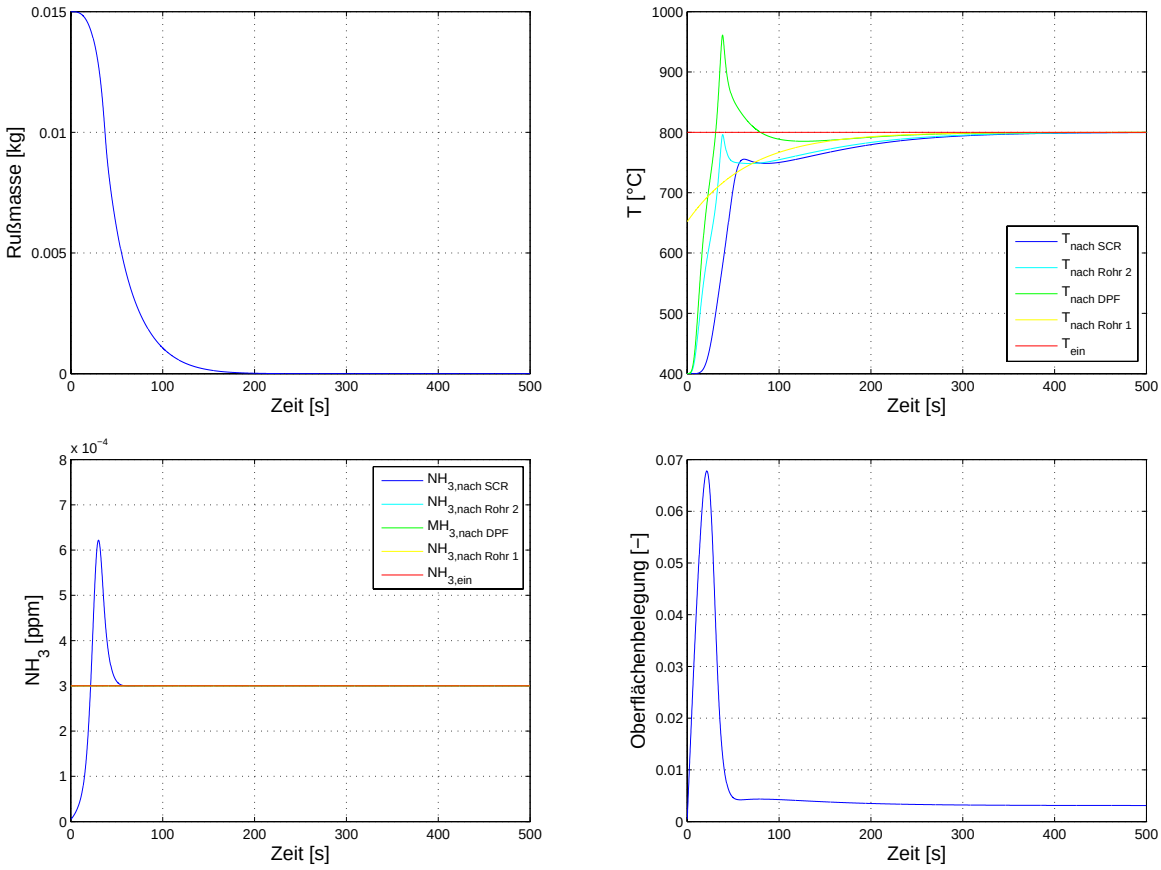


Abbildung 5: Auswirkungen einer Partikelfilterregeneration auf die NH_3-Beladung im SCR-Katalysator

sprechend hoher, konstanter Temperatur durchgeführt. Hinter dem Partikelfilter kann ein Temperaturanstieg bedingt durch den Rußabbrand beobachtet werden. Zu Beginn belädt sich die Oberfläche des SCR-Katalysators mit NH_3. In Folge der hohen Temperaturen desorbiert das Ammoniak nahezu vollständig und der Gleichgewichtszustand der Oberflächenbelegung stellt sich auf einem niedrigen Niveau ein.

2.4 Kopplung mit dreidimensionaler Simulationsumgebung

Eine eindimensionale Simulation des SCR-Katalysators ist für Fragestellungen wie der Wirkungsweise bei inhomogener Anströmung oder inhomogener Ammoniakverteilung vor Katalysator nicht ausreichend.

Durch Kopplung des Kernmodells mit der Simulationsumgebung OpenFOAM („Open Field Operation and Manipulation“) können räumliche Verteilungen von Temperaturen, Konzentrationen und Reaktionsraten berechnet werden. Dafür müssen Anströmung, Konzentrationen und Gastemperaturen als zeit- und ortsabhängiges Feld am Katalysatoreintritt vorgegeben werden. OpenFOAM übernimmt dabei die Berechnung der Wärmeleitungsgleichung im Monolithen und stellt die Kopplung zwischen den einzelnen Kanälen dar. Die Referenzkanalstruktur wird insofern beibehalten, als dass jeweils ein Referenzkanal für ein Gebiet im Katalysator steht. Je feiner das dreidimensionale Gitter erstellt wird, desto höher wird die Auflösung. Im Grenzfall wird für jeden echten Kanal im Katalysator

ein Referenzkanal berechnet.
Abbildung 6 zeigt eine inhomogene NH_3-Verteilung sowie unterschiedliche Temperaturen am Katalysatoreintritt. Während drei Quadranten des SCR-Katalysators gleichmäßig mit 500 ppm angeströmt werden, nimmt die NH_3-Konzentration linear bis auf 0 ppm im letzten Quadranten ab. Ebenso verhalten sich die Temperatur, die sich von 200°C um 20°C auf 180°C verringert, und die Geschwindigkeit, die sich auf $0\frac{m}{s}$ verringert.

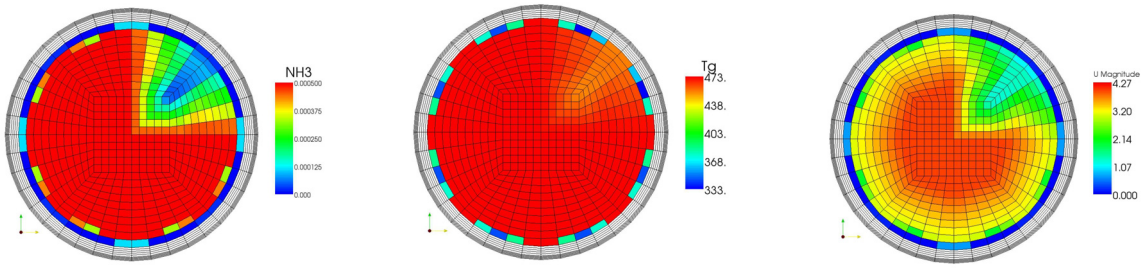


Abbildung 6: Inhomogene NH_3-, Temperatur- und Geschwindigkeitsverteilung vor dem SCR-Katalysator

Um das zeitliche und örtliche Verhalten zu untersuchen, können durch den dreidimensionalen Katalysator beliebige Schnittebenen gelegt werden. Abbildung 7 zeigt das dreidimensionale Gitter und drei gewählte Schnittebenen. Dargestellt ist die Oberflächenbelegung des SCR-Katalysators mit NH_3 (θ) nach 225 s. Die unterschiedliche Beladung bedingt durch die NH_3-Konzentrations- und Geschwindigkeitsgradienten am Eintritt kann so untersucht werden.

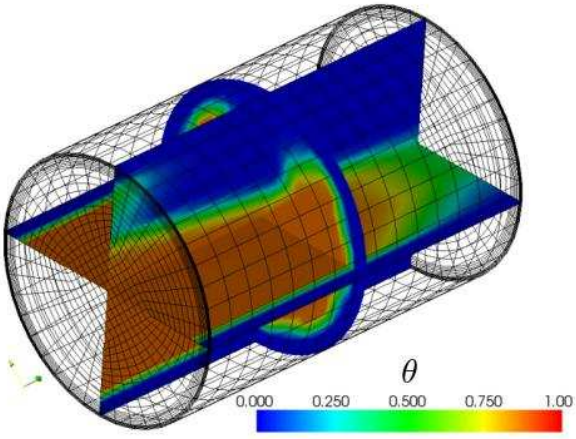


Abbildung 7: Oberflächenbelegung mit Gitter und Schnittebenen durch den Katalysator

Danksagung

Wir bedanken uns bei der Universität Rostock, insbesondere bei Herrn Dipl.-Ing. R. Bank und Herrn Dipl.-Ing. H. Kröger, für die Unterstützung bei der Entwicklung des Simulationssystems IAV KATsim.

A Reaktionsmechanismus SCR-Katalysator

Hydrolyse	$HNCO + H_2O$	$\rightarrow$	$NH_3 + CO_2$	(4)
Adsorption	NH_3	$\rightarrow$	$NH_3(s)$	(5)
Desorption	$NH_3(s)$	$\rightarrow$	NH_3	(6)
Standard SCR	$4NH_3(s) + 4NO + O_2$	$\rightarrow$	$4N_2 + 6H_2O$	(7)
Schnelle SCR	$4NH_3(s) + 2NO + 2NO_2$	$\rightarrow$	$4N_2 + 6H_2O$	(8)
Nebenreaktion	$8NH_3(s) + 6NO_2$	$\rightarrow$	$7N_2 + 12H_2O$	(9)
Oxidationsreaktion 1	$4NH_3(s) + 3O_2$	$\rightarrow$	$2N_2 + 6H_2O$	(10)
Oxidationsreaktion 2	$2NH_3(s) + 2O_2$	$\rightarrow$	$N_2O + 3H_2O$	(11)
Oxidationsreaktion 3	$4NH_3(s) + 5O_2$	$\rightarrow$	$4NO + 6H_2O$	(12)
Langsame NO-Reaktion	$4NH_3(s) + 6NO$	$\rightarrow$	$5N_2 + 6H_2O$	(13)
N_2O-Reduktion	$2NH_3(s) + 3N_2O$	$\rightarrow$	$4N_2 + 3H_2O$	(14)

B Erhaltungsgleichungen

Energieerhaltungsgleichungen:

$$\text{Gasphase} \quad \rho_g c_{p,g} \frac{\partial T_g}{\partial t} + \frac{\partial}{\partial z}\left(\rho_g u_g c_{p,g} T_g\right) = \frac{4}{d_k}\alpha\left(T_s - T_g\right) \tag{15}$$

$$\text{Solidphase} \quad \rho_s c_{p,s} \frac{\partial T_s}{\partial t} = \frac{\partial}{\partial z}\left(\lambda_s \frac{\partial T_s}{\partial z}\right) + \frac{2}{w}\alpha_m\left(T_g - T_s\right) + \dot{q}_{reak} \tag{16}$$

Stofferhaltungsgleichungen:

$$\text{Gasphase} \quad \frac{\partial c_i}{\partial t} + \frac{\partial}{\partial z}\left(\tilde{u} c_i\right) = \frac{4}{d_k}\beta\left(c_{i,s} - c_{i,g}\right) \tag{17}$$

$$\text{Solidphase} \quad \frac{\partial c_i}{\partial t} = \frac{\beta}{d_{wc}}\left(c_{i,g} - c_{i,s}\right) + \sum_j \nu_{ij} r_j \tag{18}$$

$$\text{Oberflächenbelegung} \quad \Phi \frac{\partial \theta}{\partial t} = \sum_j \nu_{ij} r_j \tag{19}$$

Literatur

[1] Hayes, R.E., Kolaczkowski, S.T.: *Introduction to Catalytic Combustion*, Overseas Publishers Association, 1997.

[2] Käfer, Sebastian: *Trockenharnstoff-SCR-System und Betriebsstrategie für Fahrzeuge mit Dieselmotor*, Dissertation, Kaiserslautern 2004.

[3] Deb, K., Agrawal, R.B.: *Simulated binary crossover for continous search space*, Complex Systems, 1995.

[4] Deb, K., Beyer, H.: *Self-adaptive genetic algorithms with simulated binary crossover*, Technical report No. CI-61/99, University Dortmund, 1999.

Nomenklatur

Chemische Abkürzungen

CO_2	Kohlendioxid
H_2O	Wasser
HNCO	Isocyansäure
N_2	Stickstoff
N_2O	Distickstoffmonoxid (Lachgas)
NH_3	Ammoniak
NO	Stickstoffmonoxid
NO_2	Stickstoffdioxid
NO_x	Stickoxide
O_2	Sauerstoff
ppm	parts per million

Lateinische Buchstaben

A	Frequenzfaktor [mol/m^{3}s]
E	Aktivierungsenergie [J/mol]
c	Stoffkonzentration [-]
c_p	spez. Wärmekapazität [J/kg K]
d	Durchmesser [m]
$\dot{m}$	Massenstrom [kg/s]
$\dot{q}$	Quellterm [J/m^{3}s]
r	Reaktionsrate [mol/m^{3}s]
$\bar{R}$	allgemeine Gaskonstante [J/mol K]
T	Temperatur [K]
t	Zeit [s]
u	Geschwindigkeit [m/s]
w	Wandstärke [m]
x	Stoffmengenanteil [-]
z	Axialkoordinate [m]

Indizes

ein	Eintritt
aus	Austritt
g	Gasphase, Abgas
s	Solidphase, Abgas
i	Index über alle chem. Komponenten
j	Index über alle Reaktionen
ads	adsorbiert
des	desorbiert
$reak$	Reaktion
k	Kanal

Griechische Buchstaben

α	Wärmeübergangszahl [W/m^{2}K]
λ	Wärmeleitzahl [W/m K]
ν	stöchiometrischer Koeffizient [-]
ρ	Dichte [kg/m^3]
θ	Oberflächenbelegung [-]
Φ	max. Oberflächenbelegung [mol/m^3]

Simulation eines Dieselluftpfades mit zwei Abgasrückführstrecken in Dymola

Frank Heßeler, Ralf Beck, D. Abel
Institut für Regelungstechnik RWTH Aachen
f.hesseler@irt.rwth-aachen.de
Jan Piewek, Christian Nöthen, Hans-Georg Nitzke
Volkswagen AG
jan.piewek@volkswagen.de

Zusammenfassung

In diesem Beitrag wird ein in der Modellierungssprache Modelica entwickeltes Simulationsmodell für den Luftpfad eines Dieselmotors mit Hoch- und Niederdruck-Abgasrückführung beschrieben, welches in Zusammenarbeit mit der Konzernforschung der Volkswagen AG entstanden ist. Das Modell dient als Basis für eine spätere Reglerentwicklung und wird zum Teil als Mittelwertmodell sowie mit Hilfe der Füll- und Entleermethode mit dem Simulationstool Dymola entwickelt. Es wird beispielhaft auf die Modellierung einzelner Komponenten wie Motorblock und Turbolader eingegangen und deren Integration zum Gesamtmodell dargestellt. Als Abschluss werden Ergebnisse des Modells für einen dynamischen Testzyklus anhand wichtiger Ausgangsgrößen diskutiert.

1 Einleitung

Die stetig sinkenden Grenzwerte für Emissionen von Verbrennungsmotoren erfordern in zunehmenden Maße die Verbesserung des Verbrennungsprozesses sowie der damit verbundenen Regelungen. Für den Dieselmotor stehen insbesondere die Stickoxid- (NO_x) und Rußemissionen im Vordergrund. Zur Reduzierung der NO_x-Emissionen hat sich die Hochdruck-Abgasrückführung (HD-AGR) bewährt, bei der verbranntes Abgas vor der Turbine entnommen wird und zum Frischluftmassenstrom nach dem Kompressor zurückgeführt wird.

Um die Abgasrückführraten weiter erhöhen zu können und um eine bessere Homogenisierung des Abgas-Luftgemisches zu erreichen, wird zusätzlich eine Niederdruck-AGR-Strecke (ND-AGR) eingeführt. Bei diesem Verfahren wird das gereinigte Abgas im Niederdruckteil des Abgasstrangs entnommen und vor dem Kompressor der Frischluft zugeführt. Neben konstruktiv bedingten Problemen insbesondere am Kompressor ist auch die Simulation und Regelung eines solchen Systems aufwendiger.

Ziel dieser Arbeit ist es, ein Simulationsmodell des oben beschriebenen Dieselmotors mit dem Simulationstool Dymola zu erstellen. Dieses Modell dient als Grundlage für eine später stattfindende Reglerentwicklung und soll daher die für eine Regelung relevanten Dynamiken des Luftpfades in einem weiten Betriebsbereich abbilden.

2 Modelica / Dymola

Modelica ist eine offene, objektorientierte Hochsprache zur Beschreibung komplexer physikalischer Systeme. Modelica ist eine noch recht junge Sprache, die erste Version 1.0 wurde im September 1997 veröffentlicht. Seitdem wird der Sprachstandard kontinuierlich innerhalb der Modelica Association weiterentwickelt. Erst kürzlich wurde Modelica in der Version 3.0 veröffentlicht. Für weiterführende Informationen auch bezüglich der Syntax sei an dieser Stelle auf [1] und [2] verwiesen.

Wesentliches Merkmal von Modelica ist der objektorientierte Ansatz, durch den es möglich wird, sehr komplexe Systeme auf einem hohen Abstraktionsgrad zu beschreiben. Ziel dieser Abstraktion ist es, das zu beschreibende System auf Basis seiner physikalischen Struktur in eine mathematische Beschreibungsform zu überführen. In Modelica ist es möglich, die das System beschreibenden Differentialgleichungen nicht fortlaufend aufzustellen, wie es in einer Programmiersprache wie z.B. C++ üblich ist, sondern deklarativ. Dies bedeutet, dass die Reihenfolge und sogar die Form der Gleichungen frei wählbar sind. Das so entstehende Gleichungssystem wird im allgemeinen Fall ein DAE-System sein. Alle Simulationsumgebungen bringen daher effiziente Methoden mit, um diese Systeme manipulieren und effizient simulieren zu können.

Aufgrund der Offenheit des Sprachstandards Modelica und dem enthaltenen objektorientierten Ansatz sind viele frei verfügbare Bibliotheken in zahlreichen Anwendungsfeldern entstanden. In dieser Arbeit wurde die *ThermoPower-Bibliothek* von Francesco Casella [3] als Basis verwendet, die bereits die Grundelemente wie Volumen, Drosseln oder Quellen und Senken beinhaltet. Diese Grundelemente können für die Eigenentwicklung neuer Komponenten verwendet werden. Die einzelnen Modell-Komponenten werden über den in der ThermoPower-Bibliothek definierten *Connector* miteinander verbunden. Dieser besteht aus den Elementen Massenstrom w, Druck p, spezifische Enthalpie h und der Stoffzusammensetzung X. Für die beiden letzteren Größen werden ein eintretender und ein austretender Anteil übergeben, um Rückströmen berücksichtigen zu können. Mit Hilfe dieses Connectors und über die in der Modelica-Bibliothek enthaltende Media-Bibliothek können alle übrigen thermodynamischen Zustandsgrößen wie z.B. Temperatur, Wärmekapazitäten oder spezifische Entropie automatisch in Abhängigkeit eines Stoffmodells auch für Mehrstoffkomponenten berechnet werden.

Für die Modellierung des Systems, die Manipulation der Gleichungen und für die Simulation des Systems haben sich mehrere Simulationsumgebungen etabliert. In dieser Arbeit wurde das Programm *Dymola* der Firma *Dynasim* [4] verwendet.

3 Dymola-Modell

3.1 Struktur des Motors

In Abbildung 1 ist der schematische Aufbau des zu modellierenden Dieselmotors dargestellt. Es handelt sich dabei um einen Vierzylinder-Dieselmotor mit Common-Rail-Direkteinspritzung und einer Luftpfadkonfiguration auf neustem technischem Stand. Dazu gehören ein Turbolader mit variabler Turbinengeometrie (VTG) für die Ladedruckerzeugung und zwei Abgasrückführstrecken zur Stickoxidminimierung. Die Niederdruck-AGR-Strecke besitzt einen AGR-Kühler, wohingegen die Hochdruck-AGR-Strecke ohne Kühler betrieben wird. Das Einlasssystem des Motors besitzt eine Drallklappe zur Beeinflussung der Ladungsbewegung im Zylinder. Zusätzlich existieren noch eine Drosselklappe und eine Abgasklappe, um auch bei niedrigen Lastzuständen ein ausreichendes Druckgefälle über die AGR-Strecken zu gewährleisten. Insgesamt besitzt das Luftsystem fünf Stellglieder und damit fünf Eingangsgrößen für die Simulation. Im Folgenden werden die Motorblockmodellierung und die Turboladermodellierung näher beschrieben.

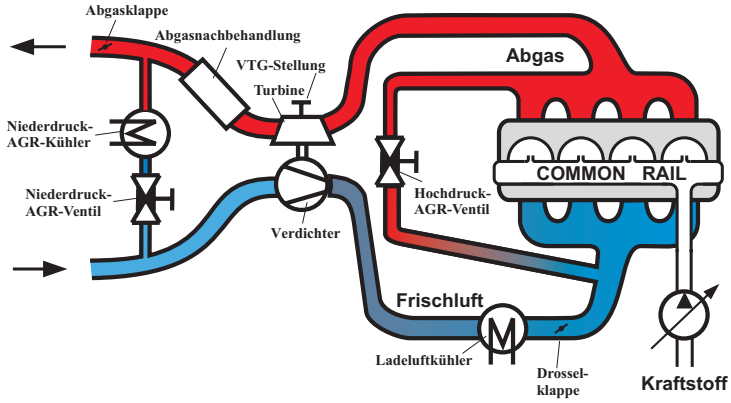


Abbildung 1: Schematischer Aufbau des Dieselmotors

3.2 Komponenten

Motorblock

Für die Bestimmung des Motoreinflusses auf den Luftpfad bzw. dessen Regelung reicht es aus, den winkelsynchron betrachteten Verbrennungsprozess und das Schluckverhalten durch Mittelwertmodelle abzubilden. Dies bedeutet, dass die berechneten Werte über einen kompletten Zyklus mit 720° Kurbelwinkel pro Zylinder gemittelt ausgegeben werden.

Die Mittelwertmodelle haben für das Gesamtmodell drei Aufgaben zu erfüllen: die Bestimmung eines in den Motor eintretenden Massenstroms ($\dot{m}_{ein}$), die Bestimmung eines austretenden Massenstrom ($\dot{m}_{aus}$) und zu guter Letzt die Berechnung der Temperaturerhöhung aufgrund der Verbrennung des eingespritzten Kraftstoffes. Der Abgasgegendruck wird nicht durch das Mittelwertmodell bestimmt, sondern durch die Charakteristik des Abgasstrangs selbst und des aufgeprägten Abgasmassenstroms. Weitere Motorgrößen wie

z.B. das Drehmoment werden für die Modellierung des Luftpfades nicht benötigt und daher auch nicht berechnet. Darüber hinaus besitzt das Mittelwertmodell des Motors keine Massenträgheit, sodass der Motor keine dynamischen Eigenschaften besitzt.

Die Berechnung des vom Motor angesaugten Massenstroms $\dot{m}_{ein}$ sowie der Temperaturerhöhung ΔT erfolgt über Polynome, deren Parameter mit Hilfe eines Optimierungsverfahrens an statische Messdaten vom Motorenprüfstand angepasst werden. Für die Bestimmung der drei Größen werden folgende Eingangsgrößen verwendet:

$$\dot{m}_{ein} = f(T_{Ein}, p_{Ein}, \rho_{Ein}, T_{Öl}, Drallklappe, Drehzahl) \quad (1)$$

$$\dot{m}_{aus} = \dot{m}_{ein} + \dot{m}_{Kraftstoff} \quad (2)$$

$$\Delta T = g(\dot{m}_{Kraftstoff}, \rho_{ein}, \lambda, X_{AGR}, Drehzahl, Einspritzwinkel) \quad (3)$$

Turbolader

Der Turbolader dient im Wesentlichen der Leistungssteigerung eines Verbrennungsmotors. In Verbindung mit der Simulation eines Dieselluftpfades hat der Turbolader zusammen mit den Volumina großen Einfluss auf die Dynamik des Gesamtsystems. Aus diesem Grund sollte der Turboladermodellierung große Aufmerksamkeit gewidmet werden.

Neben physikalisch basierten Ansätzen für die Modellierung von Turboladern haben sich kennfeldbasierte Ansätze für die effiziente Simulation durchgesetzt. Dabei wird in Kennfeldern die Charakteristik der Turbine bzw. des Kompressors in Form von normierten Massenströmen und Wirkungsgraden über dem Druckverhältnis und einer normierten Turboladerdrehzahl hinterlegt. Ein Beispiel wird in Abbildung 2 für den aktuell verwendeten Turbolader gezeigt. Um die Ergebnisse der Simulation zu verbessern, werden diese Kennfelder mit Hilfe von mathematischen Funktionen beschrieben, damit auch zwischen den vermessenen Punkten Informationen über das Verhalten des Turboladers zur Verfügung stehen.

Kompressor

Die in der gängigen Literatur zu findenden Ansätze für die Beschreibung des Kennfeldes für den Kompressormassenstrom gehen davon aus, dass die Verläufe für konstante Drehzahlen im Bereich der Pumpgrenze ihren maximalen Wert besitzen. Wie in Abbildung 2 unten links zu sehen, ist diese Annahme nur für niedrige Drehzahlen (untere Kurven) erfüllt. Ein Künstlich Neuronales Netz (KNN) ist im Gegensatz zu den klassischen Ansätzen in der Lage, die gegebene Charakteristik des Kennfelds gut zu beschreiben.

Bei der Beschreibung des Kompressorkennfeldes kann prinzipiell zwischen zwei Möglichkeiten gewählt werden. Die Erste beschreibt den *Massenstrom* durch den Kompressor als Funktion des Druckverhältnisses und der Turboladerdrehzahl. Die zweite Möglichkeit beschreibt das *Druckverhältnis* in Abhängigkeit des Massenstroms und der Turboladerdrehzahl. Beim Training des KNNs zeigt sich, dass die letztere Möglichkeit vorteilhaft für die Ergebnissqualität ist, so dass diese Modellierungsform gewählt wurde. Dies hat jedoch

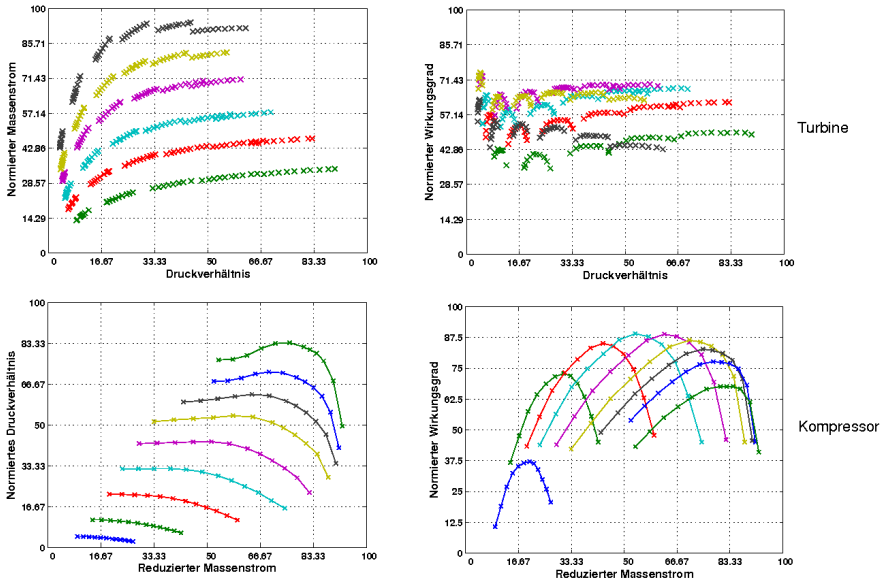


Abbildung 2: Normierte Messdaten des Turboladers

zur Folge, dass im Gesamtmodell direkt nach dem Kompressor eine zusätzliche Drossel vorgesehen werden muss, um über das berechnete Druckverhältnis den Massenstrom durch den Kompressor zu berechnen.

Als KNN wurde ein *Feedforward-Back-Propagation-Netz* mit einem versteckten Layer, zehn Neuronen und einer *TANSIG*-Aktivierungsfunktion sowie einem linearen Ausgangslayer gewählt. Für das Training wurde die *Neural Network* Toolbox von Matlab verwendet [5]. Eine höhere Anzahl von Layern bzw. Neuronen erhöhen natürlich die Fähigkeit des Netzes, die Messdaten abzubilden, es leidet jedoch die Interpolationsfähigkeit. Generell muss zwischen einer ausreichenden Güte der Optimierung und einer guten Inter- bzw. Extrapolationsfähigkeit des Netzes abgewägt werden, wobei die Extrapolationsfähigkeiten eines KNNs kritisch geprüft werden müssen.

Für den Wirkungsgrad wurde ein quadratischer Ansatz gewählt, der in [6] beschrieben wird.

$$\eta_c = \eta_{cMax} - \chi^T Q \chi \tag{4}$$

mit

$$\pi_C = 1 + \sqrt{\mathrm{Pr} - 1}, \quad Q = \begin{bmatrix} a_1 & a_3 \\ a_3 & a_2 \end{bmatrix}, \quad \chi = \begin{bmatrix} \dot{m}_c - \dot{m}_{Copt} \\ \pi_C - \pi_{Copt} \end{bmatrix}$$

Für diesen Ansatz werden die sechs Parameter (η_{cMax}, $\dot{m}_{Copt}$, π_{Copt}, a_1, a_2 und a_3) mit Hilfe der Funktion *fminsearch* von Matlab an Messdaten optimiert. Die Optimierungsergebnisse für das Druckverhältnis und den Wirkungsgrad des Kompressors sind in Abbildung 3 auf der linken Seite dargestellt. Bis auf wenige Ausreißer werden die Kennfelddaten durch die beschriebenen Modelle mit einer Genauigkeit von ±5% berechnet.

Turbine

Bei der verwendeten Turbine handelt es sich um eine Turbine mit variabler Turbinengeometrie (VTG). Diese Art der Turbine ist sehr gut für den Einsatz in einem Fahrzeugantrieb geeignet, da sie aufgrund der veränderlichen Charakteristik über einen weiten Betriebsbereich eingesetzt werden kann. Diese Eigenschaft macht jedoch die Simulation im Vergleich zum Kompressor aufwendiger, da prinzipiell für jede VTG-Stellung ein eigenes Kennfeld für die Charakteristik berechnet werden muss. Ähnlich wie beim Kompressor müssen für jede diskrete VTG-Stellung zwei Kennfelder für den normierten Massenstrom und den Wirkungsgrad in Abhängigkeit des Druckverhältnisses und einer normierten Turboladerdrehzahl bestimmt werden.

Für diese Arbeit wurden Ansätze aus [7] als Basis verwendet und für die Verwendung einer VTG-Turbine erweitert. Für den normierten Massenstrom ϕ wurde folgender Ansatz einer modifizierten adiabaten Drosselgleichung mit variablem Querschnitt gewählt:

$$\phi = A_t \cdot \sqrt{\frac{2\gamma}{\gamma - 1}\left(\left(\frac{p_{aus}}{p_{ein}} - g\right)^{\frac{2}{\gamma}} - \left(\frac{p_{aus}}{p_{ein}} - g\right)^{\frac{\gamma+1}{\gamma}}\right)} \tag{5}$$

mit

$$\begin{aligned} A_t &= k_{t1} \cdot \sqrt{PR} + k_{t2} \cdot VTG + k_{t3} \cdot VTG^2 + k_{t4} \\ k_{ti} &= x_1 \cdot N_{Tred} + x_2 \\ g &= x_9 + x_{10} \cdot VTG + x_{11} \cdot VTG^2 \end{aligned}$$

Für den Wirkungsgrad wurde ein Ansatz gewählt, in dem die Messdaten über dem Blade-Speed-Ratio $\frac{U}{C}$ aufgetragen werden.

$$\begin{aligned} \eta_T = {} & x_1 + x_2 \cdot N_{Tred} + (x_3 + x_4 \cdot N_{Tred}) \cdot \frac{U}{C} + \\ & (x_5 + x_6 \cdot N_{Tred}) \cdot \left(\frac{U}{C}\right)^2 + x_7 \cdot VTG + x_8 \cdot VTG^2 \end{aligned} \tag{6}$$

mit

$$\frac{U}{C} = \frac{\pi \cdot D \cdot N_T}{60 \cdot \sqrt{2 \cdot c_p \cdot T_{ein} \cdot \left(1 - \left(\frac{p_{aus}}{p_{ein}}\right)^{\frac{\gamma-1}{\gamma}}\right)}} \tag{7}$$

In den Formeln bezeichnet PR das Druckverhältnis, N_{Tred} die reduzierte Turboladerdrehzahl, $\frac{U}{C}$ das Blade-Speed-Ratio, γ den Isentropenexponenten, VTG die VTG-Position und x_i die zu bestimmenden Parameter. Wie beim Kompressor wurden die Parameter mit Hilfe der Matlab-Funktion *fminsearch* ermittelt. Die Ergebnisse sind in Abbildung 3 auf der rechten Seite zu sehen. Für einige Punkte konnte die Grenze von ±5% nicht eingehalten werden. Hier können weitere Arbeiten an den mathematischen Ansätzen Verbesserungen bieten.

Problematisch für die Kompressor- wie auch für die Turboladermodellierung ist jedoch

die Extrapolation der Kennfelddaten hin zu sehr geringen Massenströmen und Druckverhältnissen, wie sie sich in niedriger Teillast beim Motorbetrieb ergeben. Für diesen Bereich stehen leider keine Messdaten zur Verfügung und die Güte der Modelle ist in diesen Bereichen schlecht vorhersagbar.

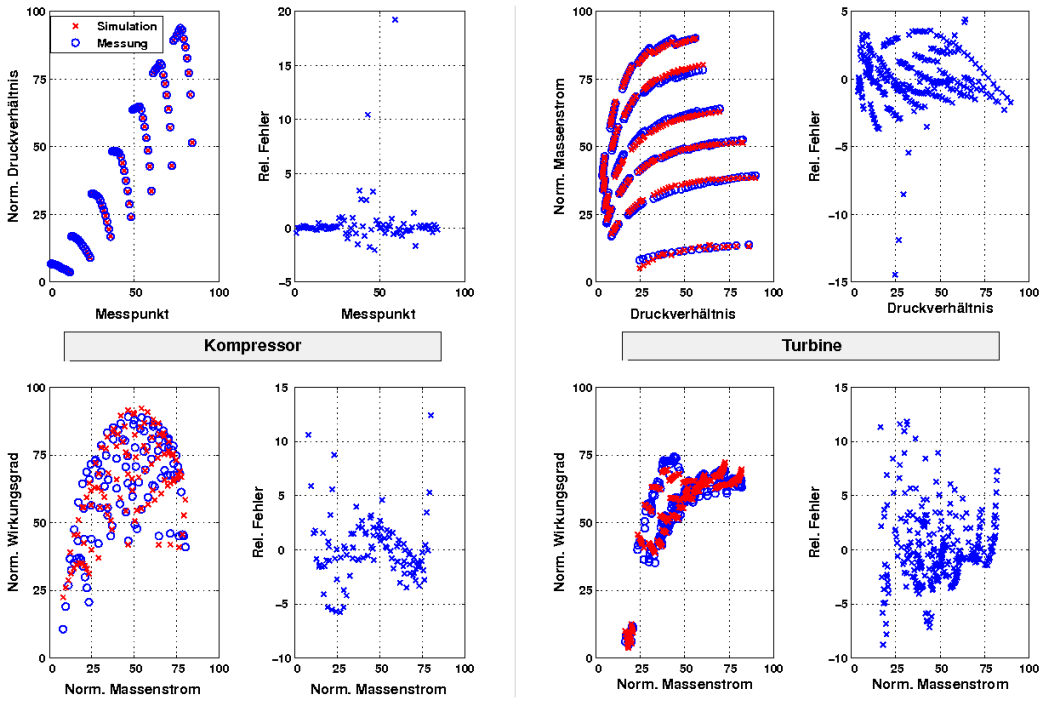


Abbildung 3: Ergebnisse der Turboladermodellierung

3.3 Gesamtmodell

Nachdem im vorangegangenen Kapitel auf ausgewählte Einzelkomponenten des Modells näher eingegangen wurde, ist in Abbildung 4 das Gesamtmodell des Motors in Dymola zu erkennen. Es besteht aus sechs Volumina mit den jeweiligen Zustandsgrößen Druck und Temperatur, dem Turbolader mit den Zustandsgrößen Wellenwinkel ϕ und Winkelgeschwindigkeit ω, den AGR-Ventilen für ND- und HD-AGR, den beiden Drosselklappen, dem Motorblock und einer Druckquelle bzw. Drucksenke für die Beschreibung der Umgebungsbedingungen für Druck und Temperatur.

In Abbildung 4 ist ein partielles Modell dargestellt, in dem noch nicht alle Eingangsgrößen definiert sind. Dieses partielle Modell kann vererbt werden und mit verschiedenen Eingangsgrößen z.B. Konstantwerten oder Sprungfunktionen simuliert werden. Dadurch ist sichergestellt, dass Änderungen am partiellen Modell auch in den vererbten Modellen zur Simulation enthalten sind.

Die Parametrierung der Modelle erfolgt über jede Einzelkomponente, da die Konfiguration des Modells flexibel gestaltet werden sollte. Ist die Konfiguration des Gesamtmodells und der Einzelmodelle abgeschlossen, wird die Parametrierung des Modells über eine Rekord-Struktur erfolgen, wie dies bereits bei der Parametrierung des Betriebspunktes

geschieht. Hierzu wird der Rekord *recWPData* verwendet, der aus einem MAT-File die benötigten Start- und Eingangswerte für das Modell entsprechend einer Prüfstandsmessdatei ausliest. Um den Betriebspunkt des Modells zu wechseln, genügt es, die Betriebspunktnummer im Rekord zu ändern. Auf diese Weise können automatisierte Tests des Modells leicht realisiert werden.

Für das Gesamtsystem entsteht ein DAE-Gleichungssytem mit 807 Gleichungen. Daraus werden durch Umformungen 295 zeitlich veränderliche Variablen, drei numerische Jacobi-Matrizen und 14 Zustandsgrößen für die Simulation generiert. Das Initialisierungsproblem, welches zu Beginn der Simulation gelöst werden muss, umfasst 35 Variablen und eine numerische Jacobi-Matrix. Für die Zustandsgrößen wird eine Steady-State-Initialisierung gefordert und für die Berücksichtigung des Betriebspunktes werden 39 Startwerte für Drücke, Temperaturen und Massenströme vorgegeben.

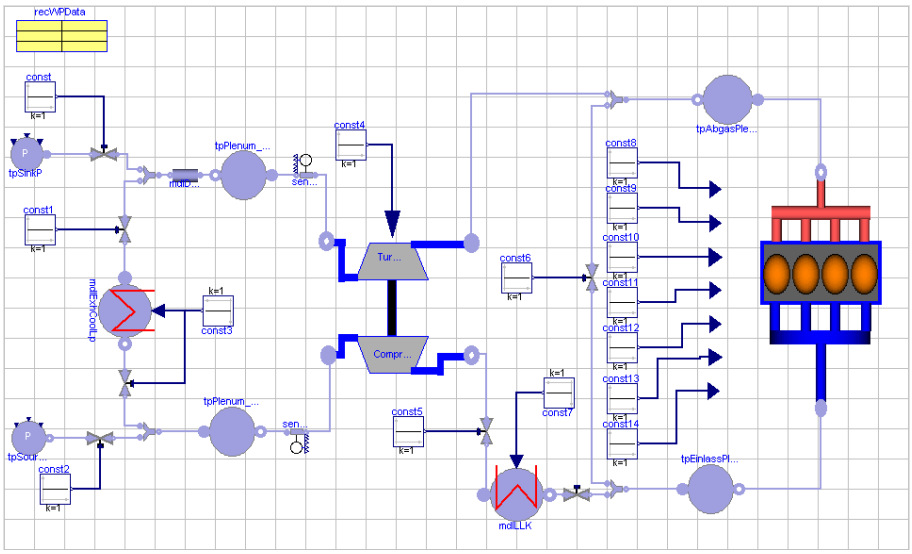


Abbildung 4: Gesamtmodell des Motors in Dymola

4 Ergebnisse

Für die Verifikation des dynamischen Verhaltens des Dymola-Modells können verschiedene Möglichkeiten gewählt werden. Zum einen kann das Modell anhand von Sprungversuchen der einzelnen Stellglieder für verschiedene Betriebspunkte mit gemessenen Daten verglichen werden. Auf diese Weise kann für jede Stellgröße einzeln die Gültigkeit des Modells überprüft werden. Eine weitere Möglichkeit besteht in der Nutzung eines dynamischen Testzykluses für den Vergleich zwischen Modell und Messung. Für diese Arbeit wurde der *FTP-Testzyklus* (Federal Test Procedure) der *US Environmental Protection Agency [8]* als ein gängiger dynamischer Standardtest für Verbrennungsmotoren verwendet.

Als beispielhaftes Ergebnis des Dymola-Modells ist in Abbildung 5 ein Ausschnitt aus dem FTP-Testzyklus für verschiedene Ausgangsgrößen dargestellt. Im oberen Graphen ist der normierte Ladedruck des Motors über einer normierten Zeitachse aufgetragen. Die Simulation folgt gut dem dynamischen Verlauf der Messung. Für die Absolutwerte der

Druckspitzen liegt das Simulationsmodell stets unterhalb der Messwerte, der angestrebte Genauigkeitsbereich von ±10% wird aber nicht verlassen.

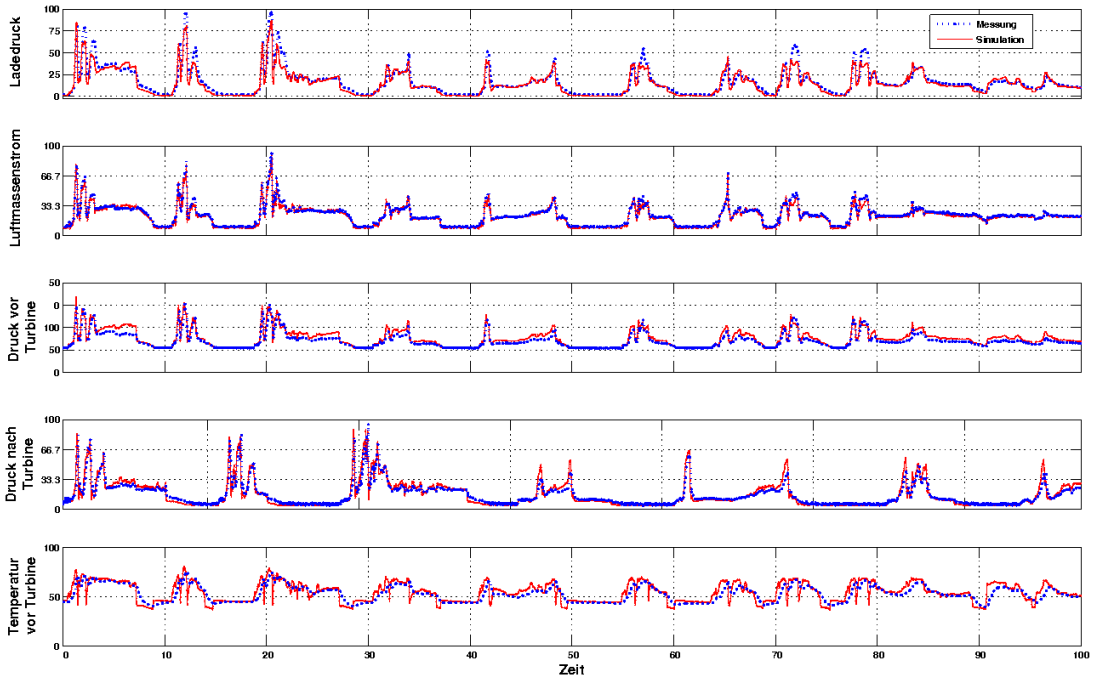


Abbildung 5: Ergebnisse für einen Ausschnitt des FTP-Zykluses

Eine zweite wichtige Größe ist der Frischluftmassenstrom des Motors. Der Vergleich der normierten dynamischen Verläufe von Messung und Simulation ist im zweiten Graphen dargestellt. Es zeigt sich eine gute Übereinstimmung zwischen Messung und Simulation. Wie beim Ladedruck, weicht der Frischluftmassenstrom auch bei den hochdynamischen Anteilen etwas stärker ab, als bei den etwas weniger dynamischen Anteilen.

Die nächsten beiden Graphen zeigen zwei weitere, für die Abgasrückführung wichtige Drücke: Druck vor Turbine (HD-AGR) und Druck nach Turbine (ND-AGR). Das dynamische Verhalten wird gut simuliert, wobei die Absolutwerte leichte Abweichungen zeigen. Diese sind auf die Problematik mit den Kompressor- und Turbinen-Kennfeldern zurückzuführen, wie sie in Kapitel 3.2 beschrieben wurden. Der letzte Graph zeigt beispielhaft den Temperaturverlauf vor Turbine. Obwohl diese Temperatur nur mit Hilfe eines statischen Polynoms berechnet wird, folgt die Simulation gut der dynamischen Messung. Die Spitzen im gezeigten Verlauf rühren von der Vernachlässigung der Dynamik des Temperatursensors und der Vernachlässigung von thermischen Effekten wie z.B. Wärmeleitung her. Das Ergebnis zeigt auch, dass die Vernachlässigung der Dynamik des Verbrennungsmotors für die Simulation des Luftpfades zulässig ist.

5 Fazit

Zusammenfassend kann gesagt werden, dass das Dymola-Modell für die gezeigten Betriebsbereiche das Verhalten des Motors gut simuliert. Für den gesamten Betriebsbereich, insbesondere in den Betriebspunkten, in denen keine Messdaten für den Kompressor und Turbine vorliegen, liegen die Abweichungen teilweise außerhalb des angestrebten Genauigkeitsbereichs. Weitere Arbeiten an den verwendeten Ansätzen, insbesondere für die Wirkungsgrade für Turbine und Kompressor, werden die Ergebnisse auch für diese Bereiche verbessern können. Das Modell bietet bereits in dem jetzigen Zustand eine gute Ausgangsbasis für die Entwicklung einer Regelstrategie für den Luftpfad.

Die Verwendung von Modelica als Sprachstandard für die Beschreibung dynamischer Systeme zeigt viele Vorteile gegenüber der blockorientierten Modellierung bei der Modellerstellung. Demgegenüber stehen jedoch Schwierigkeiten bei der Initialisierung des Simulationsmodells, die mit Hilfe von gezielt gewählten Startwerten gelöst werden können.

Literatur

[1] Homepage der Modelica Association. http://www.modelica.org.

[2] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley, IEEE Press, 2004.

[3] Homepage der ThermoPower-Bibliothek. http://home.dei.polimi.it/casella/thermopower/.

[4] Homepage der Firma Dynasim. http://www.dynasim.com.

[5] Howard Demuth, Mark Beale, and Martin Hagan. User guide: Neural network toolbox version 5. *The Mathworks, Matlab R2006a*, 2006.

[6] L. Guzzella and Christopher H. Onder. *Introduction to modeling and control of internal combustion engine systems.* Springer, 2004.

[7] Paul Moraal and Ilya Kolmanovsky. Turbocharger modeling for automotive control applications. *SAE International*, 1999.

[8] Homepage der US Environment Protection Agency. http://www.epa.gov.

Bewertung von Spannungsschwankungen im 12 V Kfz-Energiebordnetz mittels Simulation

Dipl.-Ing. Rainer Gehring (Rainer.Gehring@bmw.de),
BMW Group, Abteilung Energiebordnetz und Hybrid
Dipl.-Ing. (FH) Christoph Wanke (Christoph.Wanke@bmw.de),
BMW Group, Abteilung Energiebordnetz und Hybrid
Prof. Dr.-Ing. Hans-Georg Herzog (hg.herzog@tum.de),
Technische Universität München, Fachgebiet Energiewandlungstechnik

29./30. Mai 2008

Zusammenfassung

Die zunehmende Anzahl elektrischer Verbraucher stellt eine immer größere Herausforderung für die elektrische Energieversorgung im Kraftfahrzeug dar. Es muss eine stabile Versorgung aller Verbraucher mit einer geforderten Mindestspannung gewährleistet werden. Dabei stellt die Simulation des Energiebordnetzes ein wichtiges Hilfsmittel dar, um Aussagen zur Spannungsstabilität bereits früh im Entwicklungsprozess beantworten zu können. Am Beispiel des Fahrwerksregelsystem **D**ynamic **S**tability **C**ontrol (DSC) wird gezeigt, wie mittels Simulation die Spannungsstabilität im Energiebordnetz untersucht werden kann.

1 Einleitung

In den letzten Jahren ist die Anzahl elektrischer Komponenten im Fahrzeug stark angestiegen. Diese Entwicklung ist auf den zunehmenden Einsatz zahlreicher Innovationen aus den Bereichen Fahrwerksregel-, Fahrerassistenz-, Informations- und Entertainmentsysteme im Fahrzeug zurückzuführen. Ein weiterer Grund ist die Elektrifizierung von Nebenaggregaten. Hierunter ist zu verstehen, dass bisher mechanisch starr an den Verbrennungsmotor gekoppelte Aggregate, wie z.B. die Wasserpumpe, nun elektrisch betrieben werden. Dies wirkt sich zum einen durch den hohen Wirkungsgrad elektrischer Aktoren und zum anderen durch ihre Regelbarkeit positiv auf den Kraftstoffverbrauch aus, da diese Aggregate nun je nach Bedarf betrieben und unter Umständen auch zeitweise vollständig abgeschaltet werden können.
Mit der zunehmenden Elektrifizierung gehen allerdings auch immer größere Herausforderungen an die Energieversorgung im Kraftfahrzeug einher.

1.1 Motivation

Das elektrische Energiebordnetz muss, trotz zunehmender Komplexität, so dimensioniert werden, dass eine stabile Versorgung aller Verbraucher garantiert wird. Stabilität bezieht sich hierbei vor allem auf die Versorgung aller Komponenten mit einer gewissen Mindestspannung.
Die Spannungsstabilität mittels Messungen am Fahrzeug zu bestätigen ist allerdings nur in sehr begrenztem Maße möglich. Einerseits sind in der frühen Entwicklungsphase eines Fahrzeugs, wenn noch keine Prototypen vorhanden sind, verschiedene Konzepte für die Architektur des Energiebordnetzes zu bewerten; andererseits ist aufwandsbedingt, selbst wenn bereits Prototypen vorhanden sind, keine vollständige Messung aller Varianten des Energiebordnetzes, die sich durch die Kombination von Sonderausstattungen und Ländervarianten ergeben, möglich. Aus diesem Grund ist es erforderlich, Spannungsschwankungen im Energiebordnetz mittels Simulation abzubilden und somit die Spannungsstabilität zu bewerten. Auf diese Weise können sowohl frühzeitig Aussagen zur Bordnetzstabilität getroffen werden als auch verschiedene Varianten virtuell abgesichert werden.

1.2 Grundlegendes

Im Folgenden wird der im Rahmen des Beitrages grundlegende Begriff der Bordnetzstabilität näher erläutert und die Funktion des im Simulationsbeispiel betrachteten Fahrwerksregelsystems DSC erklärt.

- Unter Bordnetzstabilität ist die stabile Spannungsversorgung in Hinsicht auf Funktion und Komfort zu verstehen. Es muss also die Versorgung aller elektrischen Komponenten im Fahrzeug mit einer geforderten Spannung gewährleistet werden; sowohl das Unterschreiten einer Mindestspannung als auch das Überschreiten einer Höchstspannung muss vermieden werden. Darüber hinaus ist das Auftreten von für den Fahrer wahrnehmbaren Einschränkungen bezüglich Komfort, wie zum Beispiel Lichtflackern oder kurzzeitige Unterbrechungen von Informations- oder Entertainmentfunktionen (z.B. Navigationssystem oder Radio), zu vermeiden.

- Ein Beispiel für einen besonders dynamischen Verbraucher im Energiebordnetz ist die so genannte DSC-Pumpe. Das Fahrwerksregelsystem DSC greift zur Stabilisierung des Fahrzeugs bei hoch dynamischen Fahrmanövern ins Motor- und Bremsmanagement des Fahrzeugs ein. Es werden radindividuelle Bremseingriffe vorgenommen, wobei der dazu nötige Bremsdruck von einer elektrisch betriebenen Hydraulikpumpe, der DSC-Pumpe, erzeugt wird. Da äußerst kurze Reaktionszeiten für das System gefordert sind, muss die DSC-Pumpe möglichst schnell den geforderten Bremsdruck erzeugen. Die dabei fließenden Ströme von bis zu 100 A führen zu Spannungsschwankungen im Energiebordnetz. Die im Rahmen der Bordnetzstabilität geforderte Mindestspannung darf hierbei allerdings nicht unterschritten werden.

2 Systembeschreibung

Das Energiebordnetz setzt sich aus verschiedenen Komponenten zusammen. Diese können in unterschiedliche Kategorien (Energiewandler, Energiespeicher, Verbraucher und Komponenten zur Energieverteilung) unterteilt werden. Anschließend werden die verschiedenen Komponenten näher beschrieben und es wird auf das Zusammenspiel von Energiewandlern, Energiespeichern und Verbrauchern im Energiebordnetz eingegangen.

2.1 Komponenten des Energiebordnetzes

Als Energiewandler im Kraftfahrzeug werden Generatoren oder DC/DC-Wandler verwendet. Typischer Weise kommt im konventionellen Kraftfahrzeug ein Klauenpolgenerator als Energiewandler zum Einsatz. Der Generator ist dabei über einen Riemen mit der Kurbelwelle des Verbrennungsmotors verbunden. Es wird mechanische Energie in elektrische Energie umgewandelt. Die Leistung, die der Generator abgeben kann, ist hierbei vor allem von der Drehzahl und von der Temperatur abhängig.
Im Hybridfahrzeug findet anstelle des Generators ein DC/DC-Wandler Anwendung. Er stellt das Bindeglied zwischen einer höheren und der 12 V Spannungsebene dar. Es wird elektrische Energie einer Spannungslage in elektrische Energie einer anderen Spannungslage gewandelt. Zu verschiedenen Arten von DC/DC-Wandlern und deren Funktion wird auf [1] verwiesen.
Als Energiespeicher im Kfz wird in der Regel eine Blei-Säure-Batterie verwendet. Hierbei kann zwischen so genannten AGM- (**A**bsorbent **G**lass **M**at), Blei-Gel- und Nass-Batterien unterschieden werden. Bei Ersteren ist der Elektrolyt entweder in einem Vlies aus Glasfaser oder in einem Gel festgelegt. Bei Nass-Batterien liegt der Elektrolyt in flüssiger Form vor. Als batterieergänzende Speicher sind zudem Doppelschichtkondensatoren (Supercaps) denkbar. Bei diesen liegt die spezifische Energiedichte zwar nur im Bereich von 10%, die spezifische Leistungsdichte ist allerdings circa 15 mal größer als die von Blei-Säure-Batterien. Zur Deckung von kurzzeitigen Leistungsspitzen bietet sich also die Verwendung von Supercaps an. Betrachtungen zum Einsatz von Doppelschichtkondensatoren im Kfz sind z.B. in [2] zu finden.
Eine Vielzahl von verschiedensten elektrischen Verbrauchern kommt heutzutage im Fahrzeug zum Einsatz. So werden in einem modernen Oberklassefahrzeug etwa 70 Steuergeräte und weit über 100 Elektromotoren verbaut. Die stabile Spannungsversorgung ist hierbei grundlegende Voraussetzung für die Funktion der elektrischen Verbraucher.
Energiewandler, Energiespeicher und Verbraucher sind durch elektrische Leitungen, die in Form eines Kabelbaums im Kfz verlegt werden, miteinander verbunden. Es werden sowohl Kupfer- als auch Aluminiumleitungen verwendet. Hinzu kommt, dass die Stromkreise jeweils durch Sicherungen gegen zu hohe Ströme abgesichert sind und über die Karosse, die wiederum mit dem Minuspol der Batterie verbunden ist, geschlossen werden. Letzteres wird als Masserückleitung bezeichnet.

2.2 Zusammenspiel der Komponenten des Energiebordnetzes

Aufgabe des Generators ist es, die zum Betrieb der elektrischen Verbraucher benötigte Energie zur Verfügung zu stellen und die Batterie zu laden. Dies ist links in Abbildung 1 dargestellt. In Situationen, in denen der Generator keine Leistung liefern kann, z.B. im Ruhezustand des Fahrzeugs oder beim Motorstart, wird die Versorgung mit elektrischer Energie allein von der Batterie übernommen. Aber auch in Situationen, in denen der Generator elektrische Leistung liefert, kann es zu einer Entladung der Batterie kommen. Dies ist der Fall, wenn der Leistungsbedarf der Verbraucher die maximal verfügbare Leistung des Generators übersteigt (Abbildung 1 rechts). Es wird Strom aus der Batterie entnommen, wodurch die Spannung an den Klemmen der Batterie sinkt. Besonders letz-

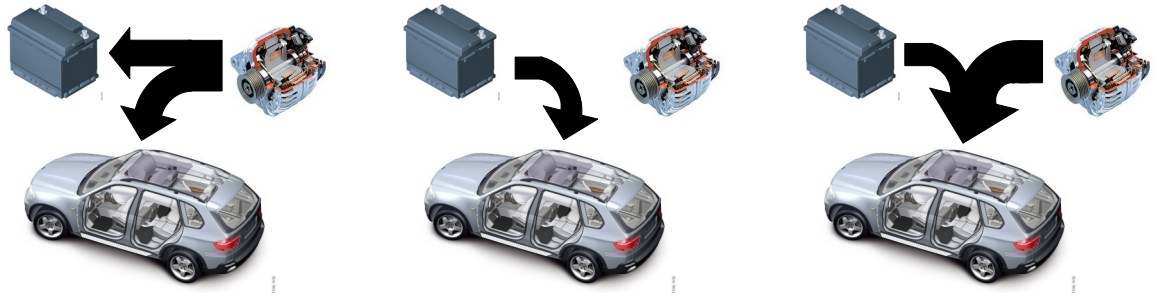


Abbildung 1: Energieflüsse

tere beiden Fälle sind für Betrachtungen zur Bordnetzstabilität interessant, da abhängig vom Strom, der aus der Batterie entnommen wird, die Spannung der Batterie so weit einbrechen kann, dass die geforderte Mindestspannung im Bordnetz unterschritten wird.

3 Modellierung

Ziel der hier betrachteten Simulation ist es Spannungsschwankungen im Energiebordnetz abzubilden. Es ist also das elektrische Verhalten der oben beschriebenen Komponenten von Interesse. Im Folgenden werden die einzelnen Modelle näher beschrieben und es wird erläutert, warum eben diese Modelle gewählt wurden.

3.1 Batterie

Modelle für Batterien können in elektrochemische, mathematische und elektrische Modelle unterteilt werden [3]. Elektrochemische und mathematische Modelle sind in der Regel sehr komplex und beanspruchen lange Rechenzeiten. Um Aussagen zum elektrischen Verhalten an den Klemmen der Batterie zu erhalten und um Spannungsschwankungen im Energiebordnetz zu untersuchen, bietet sich die Verwendung eines elektrischen Modells an. Es wurde ein elektrisches Klemmenspannungsmodell gewählt, wie es in Abbildung 2 dargestellt ist. Die Parameter wurden hierbei aus Messungen an realen Batterien generiert. Die Induktivität L_i, der Innenwiderstand R_i und die Ruhespannung U_0 werden

als konstant angesetzt. Die Parameter der RC-Glieder sind strom-, ladezustands- und temperaturabhängig und im Simulationsmodell jeweils als Kennfelder hinterlegt.

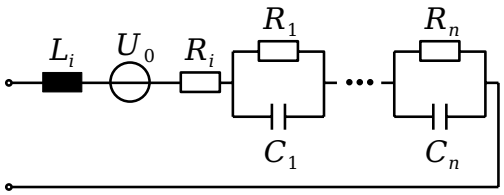


Abbildung 2: Batteriemodell

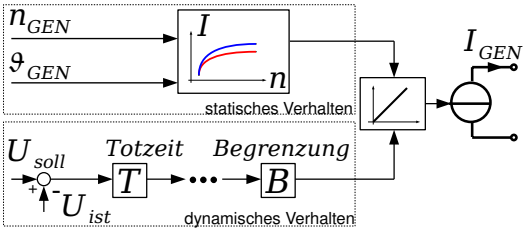


Abbildung 3: Generatormodell

3.2 Generator

Übliche Verfahren, um elektrische Maschinen abzubilden, sind Finite Elemente Simulationen, Reluktanznetzwerke oder elektrische Ersatzschaltbilder. Da das Verhalten des Generators im Systemverbund des Energiebordnetzes allerdings vorwiegend durch den Generatorregler bestimmt wird, wurde das in Abbildung 3 dargestellte Modell verwendet. Es handelt sich um eine geregelte Stromquelle. Das statische Verhalten des Generators wird dabei über ein Kennfeld, in dem abgegebener Strom über Drehzahl und Temperatur hinterlegt sind, abgebildet. In das dynamische Verhalten gehen Totzeiten und Begrenzungen mit ein. Die Totzeiten kommen durch die Signalverarbeitung innerhalb des Systems zustande. Begrenzungen ergeben sich einerseits durch die physikalischen Eigenschaften des Generators, zum anderen werden bestimmte Begrenzungen bewusst gefordert, um eine zu hohe Drehmomentaufnahme des Generators vom Verbrennungsmotor zu verhindern, da dies unter Umständen zu einem Abwürgen des Verbrennungsmotors führen kann.

3.3 Verbraucher

Für die Simulation im Rahmen dieses Beitrages wurde eine möglichst einfache Modellierung der Verbraucher gewählt. Die Modelle basieren hierbei auf Stromprofilen aus Messungen an dem Fahrzeug, an dem auch die Referenzmessung für die Simulation durchgeführt wurde.

3.4 Kabelbaum

Im Modell für den Kabelbaum sind die ohmschen Widerstände der Leitungen, die Kontakt- und Übergangswiderstände sowie die ohmschen Widerstände der verbauten Sicherungen abgebildet. Der ohmsche Spannungsabfall durch die Karosse wird in der Modellierung auch berücksichtigt. Hierzu wurde auf Grundlage von Messungen ein Berechnungsmodell erstellt, mit dem über die geometrische Entfernung zweier Massepunkte auf den ohmschen Widerstand zwischen den zwei Massepunkten geschlossen werden kann.

4 Anwendungsbeispiel DSC

4.1 Simuliertes Bordnetz

Eine schematische Darstellung des simulierten Energiebordnetzes zeigt Abbildung 4. Der Generator ist über die Ladeleitung, die als Widerstand R_1 abgebildet ist, mit dem Pluspol der Batterie verbunden. Der Batterieminuspol ist über eine Leitung an der Karosse angeschlossen. Von der Batterie aus werden die Stromverteiler versorgt. In den Stromver-

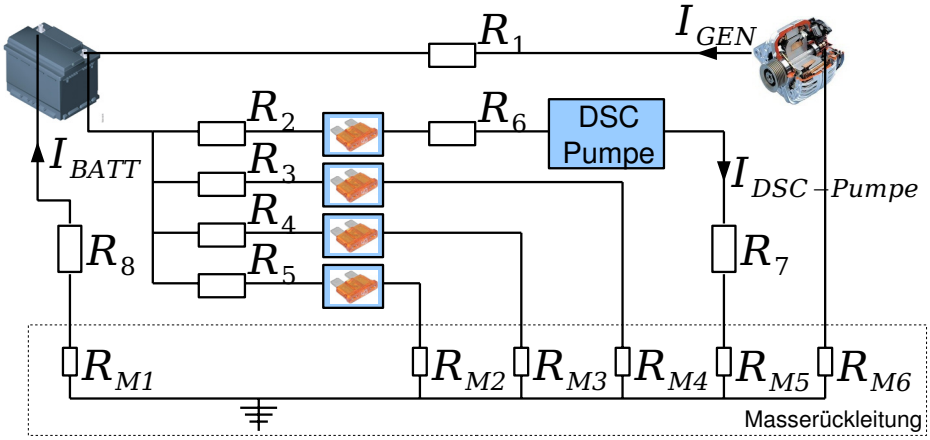


Abbildung 4: Vereinfachte Topologie des simulierten Energiebordnetzes

teilern verzweigen sich die einzelnen Stromkreise und es befinden sich dort die jeweiligen Sicherungen der einzelnen Stromkreise. Exemplarisch ist der Stromkreis für die Versorgung der DSC-Pumpe dargestellt. Die Verbraucher sind jeweils über Leitungen mit der Karosse verbunden. Die Widerstände der Masserückleitung sind ebenfalls dargestellt.
Als Simulationsplattform wurde Dymola/Modelica verwendet. Die Ergebnisse der Simulation werden im nächsten Abschnitt diskutiert.

4.2 Simulationsergebnisse

Von besonderem Interesse ist der durch die kurzzeitigen Stromspitzen verursachte Spannungseinbruch im Energiebordnetz. In Abbildung 5 ist die Spannung an der DSC-Pumpe über der Zeit dargestellt. Es sind sowohl der simulierte als auch der gemessene Verlauf der Spannung eingezeichnet.
Beim simulierten Fahrtszenario handelt es sich um eine Slalomfahrt. Hierbei kommt es zu mehreren DSC-Eingriffen. Diese werden in Abbildung 5 durch die kurzzeitigen Spannungseinbrüche ersichtlich. Ein Ausschnitt mit höherer zeitlicher Auflösung ist rechts dargestellt. Hier sind die Spannungseinbrüche deutlich zu erkennen. Ausgehend von einer Spannung von 11,5 V bricht die Spannung auf etwas unter 9 V ein.
Der Verlauf der simulierten Spannung folgt mit geringen Abweichungen dem Verlauf der gemessenen Spannung. Die Spannungseinbrüche werden mit Abweichungen von weniger als 300 mV abgebildet. Beim Spannungsverlauf bei etwa 11,5 V ist beim gemessenen Verlauf eine stärkere Dämpfung als beim simulierten Verlauf zu beobachten. Ein naheliegender

gender Grund hierfür ist, dass bei der Modellierung des Kabelbaums nur ohmsche Anteile berücksichtigt und Einflüsse durch Induktivitäten und Kapazitäten nicht modelliert wurden. Somit kann diese Simulation die in der Realität auftretende Dämpfung nicht abbilden.

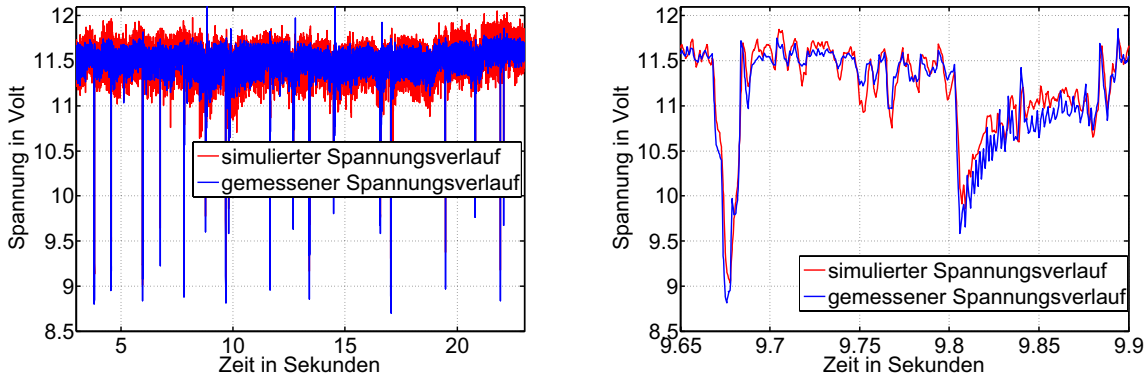


Abbildung 5: Vergleich Simulation und Messung

5 Zusammenfassung und Ausblick

Die große Anzahl elektrischer Verbraucher und der sehr hohe und dynamische Leistungsbedarf einiger dieser Verbraucher stellt eine Herausfoderung an die Energieversorgung im Fahrzeug dar. Um die Funktion der elektrischen Verbraucher zu garantieren, muss eine stabile Spannungsversorgung gewährleistet werden. Es wurde gezeigt, wie mittels Simulation Spannungsschwankungen im Energiebordnetz abgebildet werden können. Die Simulationsergebnisse bilden, abgesehen von der Dämpfung aufgrund von Induktivitäten und Kapazitäten, die realen Verhältnisse gut nach. Es ist also noch eine Erweiterung des Kabelbaummodells um induktive und kapazitive Anteile nötig, um deren Einflüsse in der Simulation abzubilden. Darüber hinaus ist angedacht, anhand von Konstruktionsdaten des Kabelbaums eine automatisierte Erstellung von Kabelbaummodellen zu realisieren.

Literatur

[1] Emandi, A.: *Handbook of automotive power electronics and motor drives*, CRC Press, Taylor & Francis Group, 2005.

[2] Schupbach, R.M.; Balda, J.C.; Zolot, M.; Kramer B.: *Design methodology of a combined battery-ultracapacitor energy storage unit for vehicle power management*, IEEE Power Electronics Specialist Conference (PESC), 2003.

[3] Chen, M.; Rincón-Mora, G.A.: *Accurate electrical battery model capable of predicting runtime and I-V performance*, IEEE Transactions on Energy Conversion, Vol 21, No. 2, 2006.

Modellierung und Simulation eines neuen Prüfstandkonzepts zur Achserprobung

D. Jung, M. Speckert, K. Dressler, Fraunhofer ITWM Kaiserslautern
dominik.jung@itwm.fraunhofer.de

Zusammenfassung

Um den gestiegenen Anforderungen an Leistung einerseits und Kosteneffizienz andererseits im Bereich der Erprobung von Fahrzeugachsen zu genügen, wurde in Zusammenarbeit mit der Firma Moog-FCS B.V. ein neuartiges Hexapoden-Achsprüfstandkonzept bis zur Einsatzreife entwickelt und für die Volkswagen AG realisiert. Aufgabe des ITWM war es, die Entwicklung und den Einsatz der neuen Konstruktion durch die Modellierung des Gesamtsystems aus Prüfstand und Prüfling simulationstechnisch zu begleiten. Damit konnten nicht nur optimierende Abschätzungen zu komplexen Konstruktionsfragestellungen frühzeitig ermittelt werden. Auch im späteren realen Testbetrieb ist der Einsatz des Simulationsmodells etwa zur Anpassung der Controllereinstellungen oder zur virtuellen Drive-File-Iteration von hohem Wert.

Die Erprobung neuer Fahrzeugachsen oder Achsvarianten auf Basis von Lastdaten aus dem Fahrbetrieb erfolgt meist mit Hilfe komplexer mehrkanaliger Prüfstände. Bei solchen Erprobungen sollen im Allgemeinen die im Fahrbetrieb gemessenen Radnabenkräfte und Momente vom Prüfstand reproduziert werden. Aufgrund der komplexen Wechselwirkungen zwischen Prüfling und Prüfmaschine stellt sich bei jedem neuen Konzept die nichttriviale Frage, ob die statische und dynamische Auslegung des Prüfstands den späteren Anforderungen im Testbetrieb genügen kann.

Ein vollständiges Computermodell von Prüfstand und Prüfling einschließlich Hydraulik und deren Ansteuerung kann hierbei nicht nur im Vorfeld wertvolle Informationen zur zielführenden Auslegung von Geometrie, Aktuatorik und Regelung liefern, sondern auch im späteren Betrieb die Vorbereitung von Messläufen unterstützen. Am Gesamtmodell können alle beim realen Prüfsystem auftretenden Arbeitsschritte wie Controllereinstellung oder Drive-File-Iteration virtuell durchgerechnet werden. Geometrische oder hydraulische Parameter sind in Hardware schonender Weise testbar, um eine optimale Anpassung des Prüfsystems an den Prüfling und die vorgegebenen Lastdaten zu ermöglichen.

Abbildung 1 zeigt die neu entwickelte Hexapod-Plattform, mit der im Betrieb beidseitig Prüfkräfte und Momente auf die Achsnaben übertragen werden. Seine Geometrie wurde von den ersten Entwürfen bis zur realisierten Form während der Entwicklungsphase anhand der Simulationsergebnisse auf Kraftdynamik und Robustheit optimiert. Dazu mussten auch virtuelle Prüflinge (Achsmodelle) entworfen und verifiziert werden. Die weitergehende Modellierung umfasste neben der Prüfstandgeometrie auch die Hydraulik sowie den internen

Controller (Abbildung 2). Das Prüfsystemmodell wurde als so genanntes Template innerhalb des Fahrzeugsimulationsprogramms ADAMS/Car entwickelt und kann mit verschiedensten Achsmodellen unkompliziert zu einem Gesamtsystem gekoppelt werden.

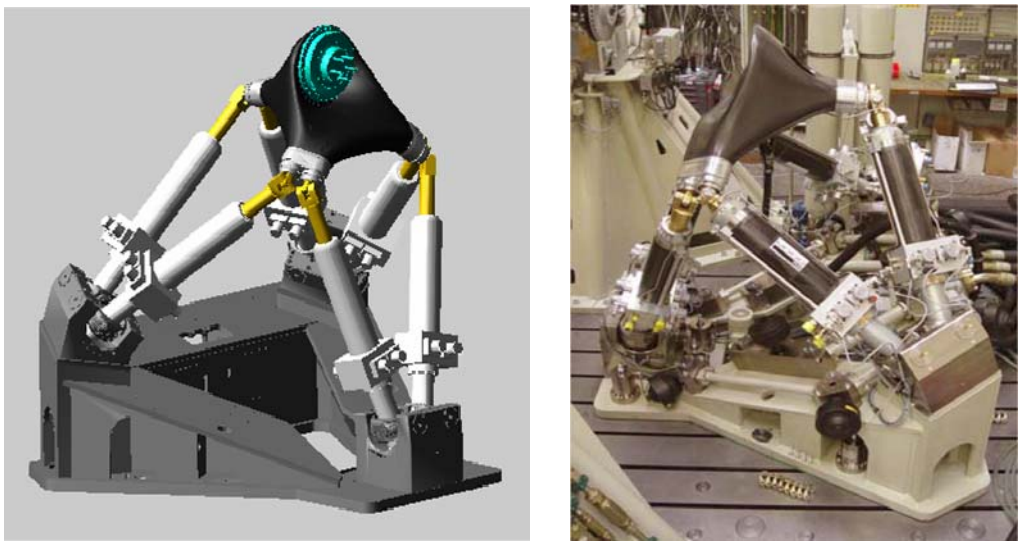


Abbildung 1: Simulationsmodell und Foto eines der beiden Achsprüfstand-Hexapoden.

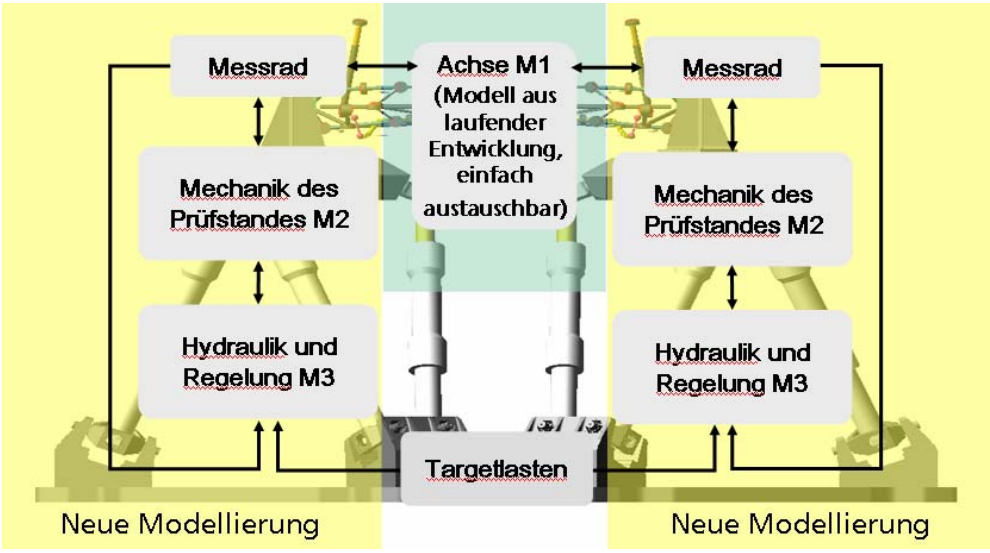


Abbildung 2: Gesamtkonzept des Prüfstands mit den Komponenten Achse, Hexapodenmechanik sowie Hydraulik und Regelung.

Literatur

[1] Bitsch G., Dreßler K., Speckert M.: *Virtual Test Rigs*, MULTIBODY DYNAMICS 2007, ECCOMAS Thematic Conference, C.L. Bottasso, P. Masarati, L. Trainelli (eds.) Mailand, Italien, 25.–28. Juni 2007

[2] Speckert, M., Dreßler, K., Mauch, H.: *MBS Simulation of a hexapod based suspension test rig*, NAFEMS Seminar, Wiesbaden, Germany, Mai 2006

Beobachtergestützte Implementierung Euler-diskretisierter nichtlinearer Systeme

S. X. Ding, A. de Moll, S. Schneider und G. Nau
Fachgebiet Automatisierungstechnik und komplexe Systeme (AKS)
Universität Duisburg-Essen
Bismarckstrasse 81 BB, 47057 Duisburg
N. Weinhold und M. Schultalbers
Ingenieurgesellschaft Auto und Verkehr GmbH (IAV)
Rockwellstraße 16, 38518 Gifhorn

Zusammenfassung

In diesem Beitrag werden zwei Ansätze vorgestellt, in denen verfügbare Messdaten in die Implementierung des Euler-Verfahrens zur Lösung nichtlinearer Differentialgleichungen integriert werden, um die numerische Stabilität zu verbessern. Diese Ansätze ermöglichen eine zuverlässige Implementierung der Euler-diskretisierten nichtlinearen Modelle in Motorsteuergeräten.

1 Einführung und Problemformulierung

Numerische Lösung des Anfangswertproblems der nichtlinearen Differentialgleichung (DGL)

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

mit dem Zustandsvektor $x(t) \in \mathcal{R}^n$, Eingangsvektor $u \in \mathcal{R}^q$ und der stetigen nichtlinearen Funktion f wird häufig in die Motorsteuerung eingebettet, um z.B. moderne Steuerungs- und Regelungskonzepte zu realisieren. Die dort eingesetzten Verfahren beschränken sich meistens auf solche numerischen Lösungen, deren Implementierung vergleichsweise wenigere Rechen- und Speicherkapazität beansprucht, denn die Steuergeräte der heutigen Generation verfügen über nur begrenzte Rechen- und Speicherkapazität.

In diesem Beitrag betrachten wir das explizierte Euler-Verfahren zur Lösung der DGL (1), das aufgrund seiner Einfachheit häufig in der Motorsteuerung angewendet wird. Das explizierte Euler-Verfahren beschreibt eine rekursive Berechnung der Form

$$x_{i+1} = x_i + hf(x_i, u_i), i = 0, 1, \cdots, \tag{2}$$

für eine numerische Lösung der DGL (1) [3], wobei h die (konstante) Schrittweite, x_i den berechneten Wert des Zustandsvektors x bei der i. rekursiven Berechnung und $u_i = u(ih)$ bezeichnen.

Dem Vorteil des minimalen Rechenaufwands des Euler-Verfahrens stehen zwei Nachteile gegenüber

- kleiner Bereich absoluter Stabilität und
- niedrige Konsistenzordnung [2, 3].

In der Motorsteuerung wird die Lösung der DGL (1) meistens in Verbindung mit Regelung in das Motorsteuergerät integriert, wo auch Sensoren eingebettet und somit Messdaten verfügbar sind. Im Allgemeinen lassen sich die Sensoren durch

$$y(t) = c(x(t)) \in \mathcal{R}^m$$

modellieren, wobei c einen Vektor mit stetigen nichtlinearen Funktionen bezeichnet. Somit stehen Messdaten

$$y(t_k), t_k = kT_a, k = 0, 1, \cdots, \tag{3}$$

zur Verfügung, wobei T_a die Abtastzeit kennzeichnet.

Die Grundidee des vorliegenden Beitrags besteht darin, mit Hilfe der verfügbaren Messdaten

- die numerische Stabilität der Rekursion (2) zu verbessern und
- die Robustheit gegenüber der Veränderung der Schrittweite zu erhöhen.

Es werden dabei zwei Ansätze entwickelt, die auf der Beobachtertheorie basieren.

2 Lösungsansätze

In diesem Abschnitt werden zwei Lösungsansätze vorgestellt. Es werden dabei folgende Annahmen getroffen:

- A1: $T_a = lh, l$ Integer
- A2: Die Änderung des Zustandsvektors beschränkt sich auf einen Bereich um den stationären Arbeitspunkt (x_o, u_o)

2.1 Lösungsansatz I

Die Euler-Rekursion (2) wird zum

Algorithmus

$$x_{i+1} = x_i + hf(x_i, u_i), \qquad i/l \neq \text{integer} \tag{4}$$

$$x_{i+1} = x_i + hf(x_i, u_i) + L(x_o, u_o)\left(y(t_i) - c(x_i)\right), i/l = \text{integer} \tag{5}$$

erweitert, wobei $L(x_o, u_o) \in \mathcal{R}^{n\times m}$, auch Beobachtermatrix genannt, die Entwurfsparametermatrix ist, deren Wahl die numerische Stabilität des Algorithmus entscheidend beeinflußt und die Kernaufgabe des Lösungsansatzes bildet.

In der Numerik verwendet man den Begriff des **Globalen Diskretisierungsfehlers**

$$e_i = x(t_i) - x_i, t_i = ih, i = 0, 1, \cdots ,$$

für die Performanzbewertung eines numerischen Verfahrens. Es gilt für den Algorithmus (4)-(5)

$$e_{i+1} = e_i + h\left(f(x(t_i), u_i) - f(x_i, u_i)\right) + r\left(x(t_{i+1}), x(t_i), u_i\right), \ i/l \neq \text{integer} \tag{6}$$

$$\begin{aligned} e_{i+1} = e_i + h\left(f(x(t_i), u_i) - f(x_i, u_i)\right) - L(x_o, u_o)\left(y(t_i) - c(x_i)\right) \\ + r\left(x(t_{i+1}), x(t_i), u_i\right), \ i/l = \text{integer}, \end{aligned} \tag{7}$$

wobei

$$r\left(x(t_{i+1}), x(t_i), u_i\right) = x(t_{i+1}) - x(t_i) - hf(x(t_i), u_i) \tag{8}$$

lokales Residuum genannt wird, das ausschließlich von der Systemdynamik und der Schrittweite abhängig ist. Der Gleichung (7) ist zu entnehmen, dass bei einer geeigneten Wahl von L die Diskretisierungsfehler e_{i+1} und ferner $e_{i+j}, j = 1, \cdots , l$, minimiert werden könnten. Zu diesem Zweck wird die folgende **Entwurfsmethode** vorgeschlagen.

Lassen sich die Linearisierungen von

$$e_{i+1} = e_i + h\left(f(x(t_i), u_i) - f(x_i, u_i)\right), y(t_i) - c(x_i)$$

um den Arbeitspunkt (x_o, u_o) durch

$$\bar{e}_{i+1} = A_o\bar{e}_i, C_o\bar{e}_i \tag{9}$$

kennzeichnen mit

$$A_o = I + h\frac{\partial f(x,u)}{\partial x}\,|_{x=x_o,u=u_o}, C_o = \frac{\partial c(x)}{\partial x}\,|_{x=x_o},$$

erhält man

$$\bar{e}_{i+l} = \left(A_o^l - L(x_o, u_o)C_o\right)\bar{e}_i, i/l = \text{integer}. \tag{10}$$

Ausgehend von (10) kann man dann die Beobachtermatrix $L(x_o, u_o)$ so wählen, dass die Eigenwerte der Matrix $A_o^l - L(x_o, u_o)C_o$, $\lambda_i\left(A_o^l - LC_o\right), i = 1, \cdots , n$, im gewünschten Bereich innerhalb des Einheitskreises liegen. Zu diesem Zweck existiert eine Vielzahl von Verfahren, welche zum Beobachterentwurf entwickelt wurden [4].

2.2 Lösungsansatz II

Im oben beschriebenen Ansatz wird die numerische Berechnung von x_i periodisch und ummittelbar nach dem Erhalten einer Messung korrigiert. Alternativ kann man bei jeder rekursiven Berechnung eine Korrektur einführen, wie der nachstehende Algorithmus beschreibt:

Algorithmus

$$\text{wenn } \frac{i}{l} = \text{integer} \Leftrightarrow j = 0 : \tag{11}$$

$$x_{i+1} = x_i + hf(x_i, u_i) + L^0_{i,1}\left(y(t_i) - c(x_i)\right), \tag{12}$$

$$x_i = x_i + L^0_{i,2}\left(y(t_i) - h(x_i)\right); \tag{13}$$

$$\text{wenn } \frac{i-1}{l} = \text{integer} \Leftrightarrow j = 1 : \tag{14}$$

$$x_{i+1} = x_i + hf(x_i, u_i) + L^1_{i,1}\left(y(t_{i-1}) - c(x_{i-1})\right), \tag{15}$$

$$x_{i-1} = x_{i-1} + L^1_{i,3}\left(y(t_{i-1}) - c(x_{i-1})\right), \cdots, \tag{16}$$

$$\text{wenn } \frac{i-l+2}{l} = \text{integer} \Leftrightarrow j = l-2 \tag{17}$$

$$x_{i+1} = x_i + hf(x_i, u_i) + L^{l-2}_{i,1}\left(y(t_{i-l+2}) - c(x_{i-l+2})\right), \tag{18}$$

$$x_{i-l+2} = x_{i-l+2} + L^{l-2}_{i,l}\left(y(t_{i-l+2}) - c(x_{i-l+2})\right); \tag{19}$$

$$\text{wenn } \frac{i-l+1}{l} = \text{integer} \Leftrightarrow j = l-1 \tag{20}$$

$$x_{i+1} = x_i + hf(x_i, u_i) + L^{l-1}_{i,1}\left(y(t_{i-l+1}) - c(x_{i-l+1})\right), \tag{21}$$

wobei $L^j_{i,1} \in \mathcal{R}^{n\times m}, j = 0, \cdots, l-1, L^j_{i,j+2} \in \mathcal{R}^{n\times m}, j = 0, \cdots, l-2$ eine periodische Beobachtermatrix bilden.

Es gilt dann für den **Globalen Diskretisierungsfehler**

$$e_{i+1} = e_i + h\left(f(x(t_i), u_i) - f(x_i, u_i)\right) - L_i(j)\left(y(t_{i-j}) - c(x_{i-j})\right) + r\left(x(t_{i+1}), x(t_i), u_i\right). \tag{22}$$

Zum **Entwurf der periodischen Beobachtermatrix** wird zunächst unter der Verwendung der Linearisierungen in (9) ein neuer Zustandvektor definiert

$$E_{i,l} = \begin{bmatrix} \bar{e}_i \\ \vdots \\ \bar{e}_{i-l+1} \end{bmatrix}. \tag{23}$$

Man erhält die Zustandsgleichung

$$E_{i+1,l} = \left(\begin{bmatrix} A_o & 0 & 0 & \cdots & 0 \\ I & 0 & & \cdots & \\ 0 & I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I & 0 \end{bmatrix} - L_i(j) C_{i,j} \right) E_{i,l} \text{ mit} \tag{24}$$

$$L_i(j) = \begin{bmatrix} L_{i,1}^j \\ \vdots \\ L_{i,l}^j \end{bmatrix}, C_{i,j} E_{i,l} = C_o \bar{e}_{i-j}, \frac{i-j}{l} = \text{integer}, j = 0, \cdots, l-1 \tag{25}$$

$$C_{i,0} = \begin{bmatrix} C_o & 0 & \cdots & 0 \end{bmatrix}, \cdots, C_{i,l-1} = \begin{bmatrix} 0 & \cdots & 0 & C_o \end{bmatrix},$$

welche ein periodisches System darstellt. Der Entwurf der (periodischen) Beobachtermatrix $L_i(j)$ besteht in der Wahl der Eigenwerte der Transitionsmatrix des Systems (24), so dass die gewünschte Konvergenz erreicht wird [1].

Die in (13)-(21) gegebenen Einstellregel für $L_i(j)$ ergeben sich aus der folgenden Betrachtung:

- Wenn $i = k, k/l = \text{integer}$

$$\begin{aligned} x_{k+1} &= x_k + hf(x_k, u_k) + L_{k,1}^0 \left(y(t_k) - c(x_k) \right) && (26) \\ x_k &= x_k + L_{k,2}^0 \left(y(t_k) - c(x_k) \right); && (27) \end{aligned}$$

- für $i = k + 1$

$$\begin{aligned} x_{k+2} &= x_{k+1} + hf(x_{k+1}, u_{k+1}) + L_{k,1}^1 \left(y(t_k) - c(x_k) \right) && (28) \\ x_k &= x_k + L_{k,3}^1 \left(y(t_k) - c(x_k) \right), \cdots, && (29) \end{aligned}$$

- für $i = k + l - 2$

$$\begin{aligned} x_{k+l-1} &= x_{k+l-2} + hf(x_{k+l-2}, u_{k+l-2}) + L_{k,1}^{l-2} \left(y(t_k) - c(x_k) \right) && (30) \\ x_k &= x_k + L_{k,l}^{l-2} \left(y(t_k) - c(x_k) \right); && (31) \end{aligned}$$

- für $i = k + l - 1$

$$x_{k+l} = x_{k+l-1} + hf(x_{k+l-1}, u_{k+l-1}) + L_{k,1}^{l-1} \left(y(t_k) - c(x_k) \right).$$

2.3 Kurze Diskussion

Es ist ersichtlich, dass die beiden Ansätze für $l = 1$ identisch sind.

Durch geeignete Wahl der Beobachtermatrix und die Nutzung der verfügbaren Messdaten können beide Verfahren den Bereich der absoluten Stabilität signifikant vergrößern.

Ferner wird dieser Bereich wesentlich robuster gegenüber der Veränderung von h.

Im Vergleich zum ersten Lösungsansatz ist der zweite Ansatz rechenintensiver. Die zusätzliche Berechnung, die (27), (29) bzw. (31) darstellen, läßt sich als eine iterative Lösung der nichtlinearen (algebraischen) Gleichung

$$y(t_k) = c(x(t_k))$$

für $x(t_k)$ interpretieren. Diese durch die Iteration verbesserte Schätzung von $x(t_k)$ könnte die numerische Zuverlässigkeit und Stabilität des Verfahrens verbessern.

3 Anwendungsbeispiel

Als Anwendungsbeispiel dient das isotherme Modell des Luftpfads eines direkteinspritzenden Ottomotors, welches im Rahmen der Füllungsregelung Verwendung findet. Die Luftfüllung des Motors wird maßgeblich über Drosselklappe und Motordrehzahl beeinflusst. Der über die Drosselklappe fließende Massenstrom wird über einen Sensor erfasst und dient als Messung, die mit der Abtastzeit T_a zur Verfügung steht. In der Realisierung der Euler Diskretisierung wird diese Messung zur Verbesserung der Konvergenz verwendet. Die Zustandsgrößen des Modells sind Druck hinter dem Luftfilter und Saugrohrdruck, welcher zudem füllungsrelevant ist und zum Zweck der Regelung genau geschätzt werden soll.

Die im Modell betrachteten Massenströme berechnen sich durch

$$\dot{m}_{LF} = K_0 \cdot (p_U - p_{hLF}) \tag{32}$$

$$\dot{m}_{DK} = K_1 \cdot \Psi\left(\frac{p_S}{p_{hLF}}\right) \cdot p_{hLF} \cdot A(\alpha) \tag{33}$$

$$\dot{m}_{Zyl} = K_2 \cdot p_S \cdot n \cdot \eta(n). \tag{34}$$

Die Drücke berechnen sich aus der Integration der zu- und abfließenden Massenströme durch

$$\dot{p}_{hFL} = K_3 \cdot (\dot{m}_{LF} - \dot{m}_{DK}) \tag{35}$$

$$\dot{p}_S = K_4 \cdot (\dot{m}_{DK} - \dot{m}_{Zyl}). \tag{36}$$

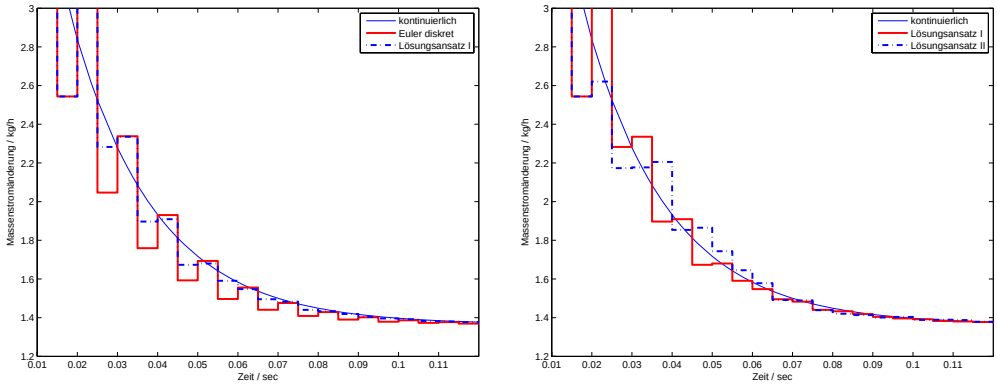
Die verwendeten Formelzeichen haben dabei folgende Bedeutung:

Variable	Einheit	Bedeutung
$\dot{m}_{LF}$	$[kg/h]$	Luftmassenstrom über Luftfilter
$\dot{m}_{DK}$	$[kg/h]$	Luftmassenstrom über Drosselklappe
$\dot{m}_{Zyl}$	$[kg/h]$	Luftmassenstrom in Zylinder
p_{hLF}	$[hPa]$	Druck hinter Luftfilter
p_S	$[hPa]$	Druck im Saugrohr
n	$[1/min]$	Motordrehzahl
α_{DK}	$[\%]$	Drosselklappenwinkel

Die Konstanten $K0 \ldots K4$ beinhalten Parameter wie etwa Temperaturen, Volumina, Gaskonstanten etc. Die Funktionen $\Psi(\frac{p_S}{p_{hLF}})$, $\eta(n)$ und $A(\alpha)$ sind motorspezifische, nichtlineare Funktionen für deren Herleitung hier auf [5] verwiesen wird.

Zur Validierung der numerischen Lösungsansätze I und II wurde das Modell exemplarisch in einem Arbeitspunkt (bei $n = 2000$ und $\alpha_{DK} = 10\%$) linearisiert und die nachfolgenden Betrachtungen auf dieses lineare Teilmodell angewendet.

In Abbildung 1(a) ist der Verlauf der Änderung des Drosselklappenmassenstroms, ausgehend von einer kleinen, sprungförmigen Änderung von n und α_{DK} dargestellt. Die durchgezogene Linie stellt den kontinuierlichen Verlauf dar, die beiden treppenförmigen Verläufe zeigen die Berechnung über den Euler-Algorithmus, einmal mit und einmal ohne periodisch korrigierende Messung. Es ist klar zu erkennen, dass diese Messung das Konvergenzverhalten des Algorithmus stark verbessert. In Abbildung 1(b) wird das Konvergenzverhalten beider Lösungsansätze miteinander verglichen, wobei sich das Verfahren II durch noch besseres Konvergenzverhalten, bedingt durch Korrektur bei jedem Schritt, äußert, wie auch in Abbildung 2 hinsichtlich des quadratischen Fehlermaßes ersichtlich ist.



(a) Vergleich Euler und Lösungsansatz I (b) Vergleich Lösungsansatz I und II

Abbildung 1: Vergleich der Lösungsansätze

4 Zusammenfassung

In diesem Beitrag wurden zwei Ansätze vorgestellt, in denen verfügbare Messdaten in die Implementierung des Euler-Verfahrens zur Lösung nichtlinearer Differentialgleichungen integriert werden, um die numerische Stabilität zu verbessern. Diese Ansätze ermöglichen eine zuverlässige Implementierung der Euler-diskretisierten nichtlinearen Modelle in Motorsteuergeräten.

Die vorgestellten Ansätze können auch zum Entwurf und zur Implementierung von Beobachtern für nichtlineare dynamische Systeme angewendet werden.

Literatur

[1] Bittanti, S. und Colaneri, P.: An LMI characterization of the class of stabilizing controllers for periodic descrete-time systems, *Proc. of the 14th IFAC world congress, 1999.*

[2] Eich-Soellner, E. und Führer, C.: *Numerical Methods in Multibody Dynamics*, B. G. Teubner Stuttgart, 1998.

[3] Hoffmann, A., Marx, B. und Vogt, W.: *Mathematik für Ingenieure 2*, Pearson Studium, 2006.

[4] O'Reilly, J.: *Observers for Linear Systems*, Academic press, 1983.

[5] Weinhold, N.: *Einbettung modellgestützter Fehlerdiagnose in Regelungssysteme und deren Anwendung für die On-Board Diagnose in Fahrzeugen*, Dissertation, Universität Duisburg-Essen, 2007.

Anhang: Abbildungen

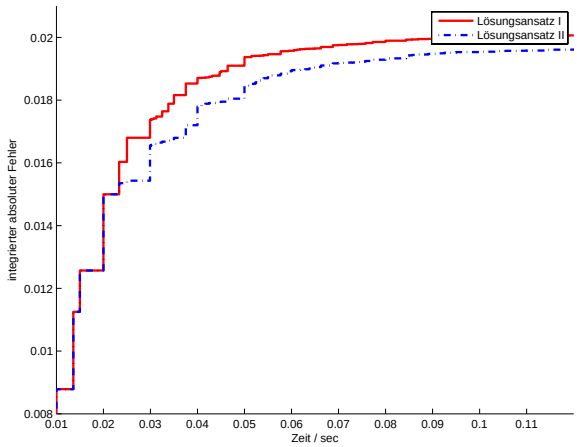


Abbildung 2: integrierter absoluter Fehler

A Multivariable Speed Controller for a Hybrid Electric Vehicle

Jan P. Blath, Ingenieurgesellschaft Auto und Verkehr GmbH
jan.peter.blath@iav.de

Abstract

Subject of this contribution is the decoupling control of a parallel hybrid vehicle drivetrain. The control problem under consideration consists of synchronizing the speeds of two propulsion aggregates via a controllable clutch without disturbing the longitudinal vehicle motion. For the purpose of controller design and validation a detailed nonlinear model of the drivetrain is derived. The multivariable controller consists of two PID speed controllers which interact via a dynamic decoupling transfer element.

1 Introduction

Drivetrains of modern vehicles become more and more complex due to the presence of a growing number of actuators which, of course, is accompanied by an increasing number of degrees of freedom. This is especially true for hybrid electric vehicles which call for an adequate coordination of propulsion aggregates, clutches and transmissions leading to a number of control problems which are multivariable by nature.

The drivetrain considered in this contribution belongs to the class of so-called *parallel* hybrid drivetrains, i.e. the internal combustion engine and the electrical machine both work on the same shaft, see Figure 1. Thus, the resulting overall propulsion torque consists of the sum of the respective torque contributions from both aggregates. The internal combustion engine can be connected and disconnected from the rest of the drivetrain by a separation clutch. During electrical drive, i.e. when the electrical machine propels the vehicle and the combustion engine is at rest, the separation clutch is open thus relieving the electrical machine from friction, throttling and compression losses caused by the unfired combustion engine. If the combustion engine has to be started in this driving situation, e.g. due to a low state of charge of the battery system, the separation clutch has to be closed in such a way that deterioration of driving comfort is avoided. Closing of the clutch is followed by the start of injection and ignition.

The resulting control problem, which will be treated in the succeeding sections, is to derive a controller which on the one hand synchronizes the combustion engine and the electrical machine and on the other hand conserves driving comfort. Naturally, this can

be formulated in a multivariable framework since the basic control problem is obviously to decouple the motions of combustion engine and electrical machine thus preventing the clutch from disturbing the longitudinal vehicle motion.

A control problem which is to a certain degree similar to that considered in this contribution is treated in [1]. The drivetrain considered therein differs from that regarded in this contribution mainly by a second electrical machine which actuates on the crankshaft. This machine is used to pull the combustion engine speed up to an appropriate starting level. The synchronization process is then carried out by the superimposition of the torque contributions from the electrical machine and the combustion engine. The separation clutch is open during this process. The other electrical machine serves for vehicle propulsion. Since the separation clutch is not used for synchronization the motions of the vehicle and of the internal combustion engine are already decoupled. Due to this strategy there is no need for a *decoupling* controller even though the control problem is of course multivariable, too. It is solved by the authors making use of a predictive control algorithm.

This paper is structured as follows. In section 2, a detailed model of the parallel hybrid drivetrain is derived. The controller design is performed in section 3. Simulation results are presented in section 4 and conclusions are drawn in section 5.

2 Process Model

2.1 Rotational Dynamics

Figure 1 depicts the parallel hybrid electric drivetrain under consideration. The combustion engine with its moment of inertia J_1 can be separated from the drivetrain by the separation clutch (sc). The electrical machine actuates on the same shaft as the combustion engine. The inertias of the elctrical machine and of the impeller side of the torque converter (tc) are lumped into the inertia J_2. The turbine side of the torque converter and the transmission input shaft are represented by the inertia J_3. The conversion ratio of the automatic transmission is denoted by i_t. The output shafts of the transmission and the input shafts of the final drive are lumped into the inertia J_4. The conversion ratio of the final drive is i_d. The inertias of the drive shafts are denoted by J_5 and the wheel and vehicle inertias are lumped into J_6. The torsional flexibility of the drive shafts is modelled by the stiffness k and the damping d.

The lock-up clutch of the hydrodynamic converter can either be closed or slipping. The case of an open clutch can be handled mathematically in the same way as a slipping clutch. If the lock-up clutch is closed, one degree of freedom of the rotational motion is lost since J_2 and J_3 are forced to rotate with the same angular velocity, i.e. $\dot{\varphi}_2 = \dot{\varphi}_3$. In this contribution only the case of an open lock-up clutch will be considered.

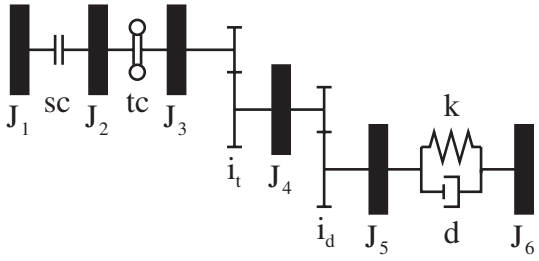


Figure 1: Parallel hybrid drivetrain

The rotational dynamics are described by:

$$
\begin{aligned}
J_1\ddot{\varphi}_1 &= T_{\mathrm{sc}} - T_{\mathrm{ice}}(p_{\mathrm{m}}, \dot{\varphi}_1) && (1)\\
J_2\ddot{\varphi}_2 &= T_{\mathrm{em}} - T_{\mathrm{sc}} - T_{\mathrm{imp}}(\dot{\varphi}_2, \dot{\varphi}_3) && (2)\\
J_3'\ddot{\varphi}_3 &= T_{\mathrm{turb}}(\dot{\varphi}_2, \dot{\varphi}_3) - \frac{1}{i}\left(k(\varphi_3/i - \varphi_6) + d(\dot{\varphi}_3/i - \dot{\varphi}_6/i)\right) && (3)\\
J_6\ddot{\varphi}_6 &= k(\varphi_3/i - \varphi_6) + d(\dot{\varphi}_3/i - \dot{\varphi}_6/i) - T_{\mathrm{res}} && (4)
\end{aligned}
$$

Here $\dot{\varphi}_1$, $\dot{\varphi}_2$, $\dot{\varphi}_3$ and $\dot{\varphi}_6$ denote the angular velocities of the combustion engine, the electrical machine, the torque converter's turbine and the wheels, respectively. The overall conversion ratio is given by $i = i_{\mathrm{t}} i_{\mathrm{d}}$. The inertias J_3 to J_5 are lumped into

$$J_3' := J_3 + J_4/i_t^2 + J_5/i^2. \tag{5}$$

The drag toque of the combustion engine is denoted by T_{ice}. It depends on manifold pressure p_{m} and engine angular velocity $\dot{\varphi}_1$, see Figure 2. The drag torque map reflects the behaviour of a warm engine. The torque transmitted by the separation clutch is T_{sc}. The electrical machine delivers the torque T_{em}. The resistance torque T_{res} is proportional to the sum of air-drag, rolling resistance and gravitational force:

$$T_{\mathrm{res}} = r_{\mathrm{dyn}}\left(F_{\mathrm{ad}} + F_{\mathrm{rr}} + F_{\mathrm{g}}\right). \tag{6}$$

The three resistance forces are given by

$$
\begin{aligned}
F_{\mathrm{ad}} &= \frac{1}{2} c_{\mathrm{w}} A_{\mathrm{veh}} \rho_{\mathrm{air}} r_{\mathrm{dyn}}^2 \dot{\varphi}_6^2, && (7)\\
F_{\mathrm{rr}} &= f_{\mathrm{rr}}(\dot{\varphi}_6) m_{\mathrm{veh}} g \cos\chi, && (8)\\
F_{\mathrm{g}} &= m_{\mathrm{veh}} g \sin\chi. && (9)
\end{aligned}
$$

Here r_{dyn} denotes the dynamic tyre radius, c_{w} represents the air-drag coefficient, A_{veh} and m_{veh} denote the vehicle's cross-sectional area and its mass, respectively, f_{rr} represents the speed-dependent rolling resistance coefficient, g is the gravitational constant and χ denotes the road slope.

In general, the transfer characteristics of hydrodynamic torque converters are deter-

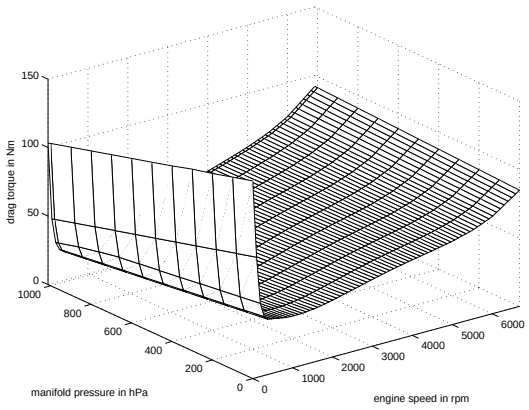


Figure 2: Drag torque of the internal combustion engine

mined experimentally, yielding the impeller torque T^*_{imp} and the torque ratio

$$\mu = \frac{T_{\text{turb}}}{T^*_{\text{imp}}} \tag{10}$$

for a fixed impeller speed n^*_{imp} as a function of the speed ratio

$$\nu = \frac{n_{\text{turb}}}{n_{\text{imp}}}, \tag{11}$$

see Figure 3. Here n_{turb} denotes turbine speed. The impeller torque for variable impeller speeds is then given by

$$T_{\text{imp}} = T^*_{\text{imp}}(\nu) \cdot \left(\frac{n_{\text{imp}}}{n^*_{\text{imp}}}\right)^2, \tag{12}$$

see e.g. [6]. Hence, the corresponding turbine torque can be computed from the impeller torque via (10):

$$\begin{aligned} T_{\text{turb}} &= \mu(\nu) T_{\text{imp}} \\ &= \mu(\nu) T^*_{\text{imp}}(\nu) \left(\frac{n_{\text{imp}}}{n^*_{\text{imp}}}\right)^2. \end{aligned} \tag{13}$$

Both expressions (12) and (13) only depend on impeller and turbine speed and thus on the angular velocities $\dot{\varphi}_2$ and $\dot{\varphi}_3$.

2.2 Intake Manifold Pressure Dynamics

The mass balance of air within the intake manifold is

$$\frac{\mathrm{d}m_{\text{air}}}{\mathrm{d}t} = \dot{m}_{\text{th}}(p_{\text{m}}, \alpha_{\text{th}}) - \dot{m}_{\text{cc}}(p_{\text{m}}, \dot{\varphi}_1). \tag{14}$$

Introducing the ideal gas law

$$pV = mR\vartheta \tag{15}$$

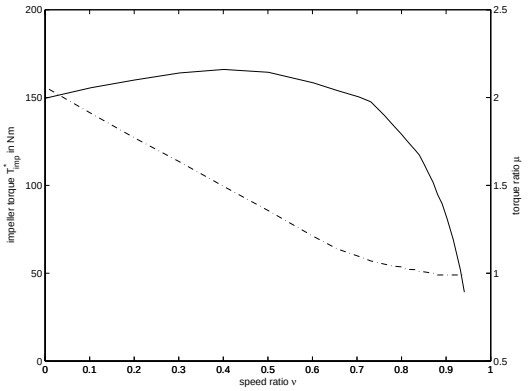


Figure 3: Impeller torque (solid) and torque ratio (dash-dotted) for fixed impeller speed (example)

into (14) yields

$$\dot{p}_{\mathrm{m}} = \frac{R_{\mathrm{air}}\vartheta_{\mathrm{a}}}{V_{\mathrm{m}}}\left(\dot{m}_{\mathrm{th}}(p_{\mathrm{m}},\alpha_{\mathrm{th}}) - \dot{m}_{\mathrm{cc}}(p_{\mathrm{m}},\dot{\varphi}_1)\right), \tag{16}$$

Here R_{air} denotes the gas constant of air, ϑ_{a} represents ambient temperature, which is assumed to correspond with the temperature of the air flowing past the throttle plate and V_{m} denotes the intake manifold volume. The relationship (16) is well known in the context of mean-value engine modeling, see e.g. [2, 3, 4].

The throttle mass flow rate is described by the orifice equation which can be written as

$$\dot{m}_{\mathrm{th}} = \dot{m}^*_{\mathrm{th}}(\alpha_{\mathrm{th}}) \cdot \frac{p_{\mathrm{a}}}{p^*_{\mathrm{a}}} \cdot \sqrt{\frac{\vartheta^*_{\mathrm{a}}}{\vartheta_{\mathrm{a}}}} \cdot \tilde{\Psi}\left(\frac{p_{\mathrm{m}}}{p_{\mathrm{a}}}\right), \tag{17}$$

see [7]. The pressure drop along the air-filter is neglected. Ambient pressure p_{a} and temperature ϑ_{a} are assumed to be constant. The normalized mass flow rate $\dot{m}^*_{\mathrm{th}}$ is determined experimentically under reference conditions defined by p^*_{a} and ϑ^*_{a}. The symbol α_{th} denotes throttle angle and $\tilde{\Psi}$ represents the normalized flow function. Obviously, the underlying assumptions lead to a mathematical description of the throttle mass flow rate which depends only on intake manifold pressure and throttle angle.

The mass flow rate into the combustion chamber is given by

$$\dot{m}_{\mathrm{cc}} = c_1\left(p_{\mathrm{m}} - c_2\right), \tag{18}$$

see [7]. Here c_1 and c_2 are functions which depend mainly on engine speed and camshaft timing. The function c_2 additionally depends on intake manifold pressure. The camshaft timing is assumed to be constant.

2.3 Actuator Dynamics

The torque modulation of the separation clutch and the torque generation dynamics of the electrical machine are approximately described by first order lags with delay:

$$\dot{T}_{\mathrm{sc}}(t) = \frac{1}{\tau_{\mathrm{sc}}} \left(u_{\mathrm{sc}}(t - \delta_{\mathrm{sc}}) - T_{\mathrm{sc}}(t) \right), \tag{19}$$

$$\dot{T}_{\mathrm{em}}(t) = \frac{1}{\tau_{\mathrm{em}}} \left(u_{\mathrm{em}}(t - \delta_{\mathrm{em}}) - T_{\mathrm{em}}(t) \right). \tag{20}$$

Here u_{sc} and u_{em} denote the torque reference values of separation clutch and electrical machine, respectively. The delays δ_{sc} and δ_{em} are caused mainly by the data transfer via the CAN bus and by the fact that the descriptions above approximate of course some higher order dynamics.

2.4 Model Summary

Equations (1) - (4), (16), (19) and (20) describe the overall plant. This system description can be visualized by the block diagram shown in Figure 4. Obviously, the overall plant consists of two subsystems. The upper subsystem represents the torque generation dynamics of the electrical machine and the rotational dynamics downstream of the separation clutch. The lower subsystem contains the dynamics of the separaton clutch, the rotational dynamics of the combustion engine and the intake manifold filling dynamics. Both subsystems interact only via the torque transmitted by the separation clutch.

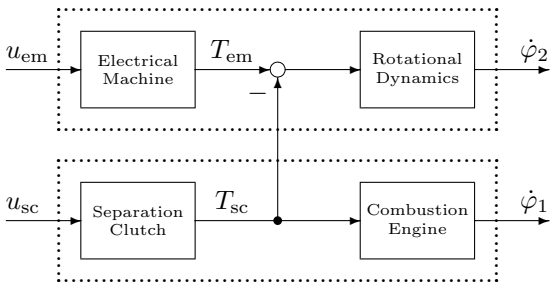


Figure 4: Model structure

3 Controller Design

For the forthcoming treatment of the control problem, the following definitions are made: $u_1 := u_{\mathrm{sc}}$, $u_2 := u_{\mathrm{em}}$, $y_1 := \dot{\varphi}_1$ and $y_2 := \dot{\varphi}_2$.

Figure 5 depicts the structure of the overall control system. It consists of two speed controllers, a block for clutch load compensation and a reference model which reflects the rigid drivetrain motion which transforms the torque demand generated by the driver into a corresponding speed reference value. For frequencies below the drivetrain resonance, the speed control loop of the electrical machine can be interpreted as a model following

approach with the reference dynamics being equal to the original plant dynamics thus yielding a direct feedthrough from the torque demand to the plant input. The underlying idea assumes the drivetrain elasticity as some kind of model uncertainty, against which the feedback controller is intended to ensure robustness. The engine speed control loop is structured rather simple without a feedforward path. The speed margin is introduced in order to prevent the engine speed from crossing the speed of the electrical machine which would result in a sign change of clutch torque.

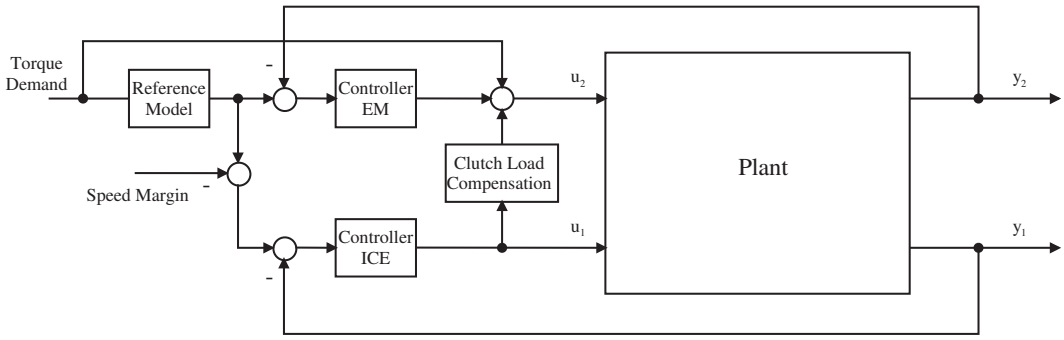


Figure 5: Overview of the speed control system

3.1 Compensation of Clutch Load

Linearization of the plant model for a fixed operating point yields the transfer matrix description

$$\Delta Y(s) = G(s) \cdot \Delta U(s), \tag{21}$$

where

$$\Delta U(s) := \left[\begin{array}{cc} \Delta U_1(s) & \Delta U_2(s) \end{array} \right]^{\mathrm{T}}, \tag{22}$$

$$\Delta Y(s) := \left[\begin{array}{cc} \Delta Y_1(s) & \Delta Y_2(s) \end{array} \right]^{\mathrm{T}}. \tag{23}$$

Here $\Delta U_1(s)$, $\Delta U_2(s)$, $\Delta Y_1(s)$ and $\Delta Y_2(s)$ denote the Laplace transforms of the deviations $\Delta u_1 := u_1 - \bar{u}_1$, $\Delta u_2 := u_2 - \bar{u}_2$, $\Delta y_1 := y_1 - \bar{y}_1$ and $\Delta y_2 := y_2 - \bar{y}_2$. The constant values belonging to the operating point are marked with bars. The structure of the transfer matrix is independent of the status of the lock-up clutch. It has the following triangular shape:

$$G(s) = \left[\begin{array}{cc} G_1 G_{\mathrm{sc}} & 0 \\ -G_2 G_{\mathrm{sc}} & G_2 G_{\mathrm{em}} \end{array} \right]. \tag{24}$$

Here G_{sc} represents the transfer function of the separation clutch corresponding to (19), G_{em} denotes the transfer function of the electrical machine corresponding to (20), G_1 and G_2 denote the transfer paths from torque changes to changes of angular velocity.

The choice

$$\Delta U_2(s) = \Delta \tilde{U}_2(s) + \frac{G_{\mathrm{sc}}(s)}{G_{\mathrm{em}}(s)} \Delta U_1(s), \tag{25}$$

with $\Delta\tilde{U}_2(s)$ denoting the Laplace transform of changes of the new system input $\tilde{u}_2$ yields

$$\Delta Y(s) = \begin{bmatrix} G_1 G_{\mathrm{sc}} & 0 \\ 0 & G_2 G_{\mathrm{em}} \end{bmatrix} \begin{bmatrix} \Delta U_1(s) \\ \Delta\tilde{U}_2(s) \end{bmatrix}. \tag{26}$$

The resulting transfer matrix is diagonal and the system is thus decoupled. Since the actuator dynamics are assumed to be linear and the interaction structure of the plant is not changed by linearization, the decoupling also holds for the nonlinear system. Hence,

$$U_2(s) = \tilde{U}_2(s) + \frac{G_{\mathrm{sc}}(s)}{G_{\mathrm{em}}(s)} U_1(s) \tag{27}$$

decouples the nonlinear system.

Since

$$\frac{G_{\mathrm{sc}}(s)}{G_{\mathrm{em}}(s)} = \frac{\tau_{\mathrm{em}} s + 1}{\tau_{\mathrm{sc}} s + 1} \cdot e^{-(\delta_{\mathrm{sc}} - \delta_{\mathrm{em}})s}, \tag{28}$$

the decoupling control law (27) is realizable only if

$$\delta_{\mathrm{sc}} \geq \delta_{\mathrm{em}}. \tag{29}$$

The decoupled system with the inputs u_1 and $\tilde{u}_2$ can be controlled by two speed controllers which can be designed independently of each other.

3.2 Speed Control Loops

The PID speed controller for the electrical machine is designed using a vector performance index and applying the vector performance index optimization strategy described in [5]. The objective function consists of criteria measuring stability, integrated squared tracking error, control effort, steady-state exactness and damping. The controller design is performed for the linearized plant $G_{\mathrm{em}} G_2$. In order to avoid windup effects, an anti-windup gain is provided. The derivative term is approximated by a first order highpass. The controller parameters are scheduled via gear and lock-up clutch status.

The PID speed controller of the combustion engine is optimized using the same strategy as for the speed controller of the electrical machine. The performance criteria measure stability, overshoot, control effort, rise time, steady-state exactness and positiveness of the controller output. The last criterion is necessary, since the sign of the transmissible clutch torque is determined by the sign of the clutch slip. Hence, for pulling the combustion engine speed up to the speed of the electrical machine without overshoot, only non-negative clutch torque is physically realizable. The engine speed controller is designed for the linearized plant $G_{\mathrm{sc}} G_1$. It also makes use of an anti-windup gain. The derivative term is applied only to the plant output and it is approximated by a first order highpass. The controller parameters are chosen to be fixed, but it is expected, that at least a temperature dependency will have to be taken into account in a practical implementation.

The engine speed controller output has to be limited, otherwise the electrical machine

might fail to compensate the load caused by the separation clutch which would result in deterioration of driving comfort. A reasonable choice of upper limit is the difference between the maximum torque of the electrical machine and the current driver demand. Of course, the lower limit is zero.

4 Simulation Results

The simulations are performed using the nonlinear model. The combustion engine was assumed to be warmed up.

Figure 6 shows a speed synchronization during constant driving in the second gear with open lock-up clutch. The engine start request may be caused e.g. by a low state-of-charge of the traction battery. The maximum torque of the electrical machine is chosen as 250 Nm. The clutch load can be fully compensated by the electrical machine. Hence, the control variables are completely decoupled and thus show no interaction.

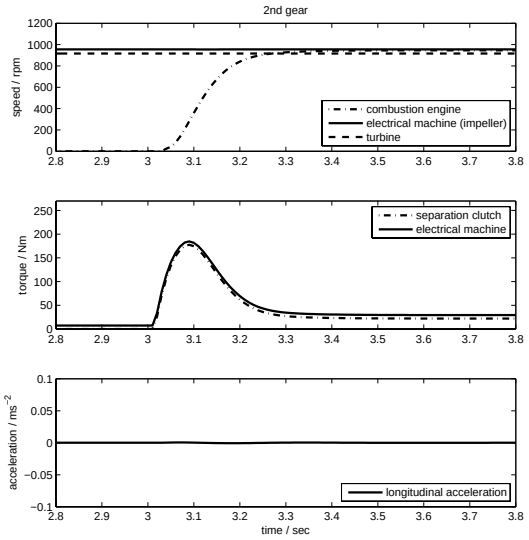


Figure 6: Speed synchronization, lock-up clutch open, constant driving

Perfect decoupling is possible, if the transfer characteristics of the electrical machine and of the separation clutch are exactly known. In this case, the decoupling property of the proposed control strategy is neither affected by variations of drivetrain parameters like e.g. vehicle mass or torque converter characteristics nor by variations of the combustion engine system. Unfortunately, there are some sources of uncertainty, especially considering the communication delays of the CAN bus and the clutch dynamics, resulting in deteriorated decoupling control performance. Figure 7 shows the synchronization process where the delay δ_{sc} has been increased by 50 percent.

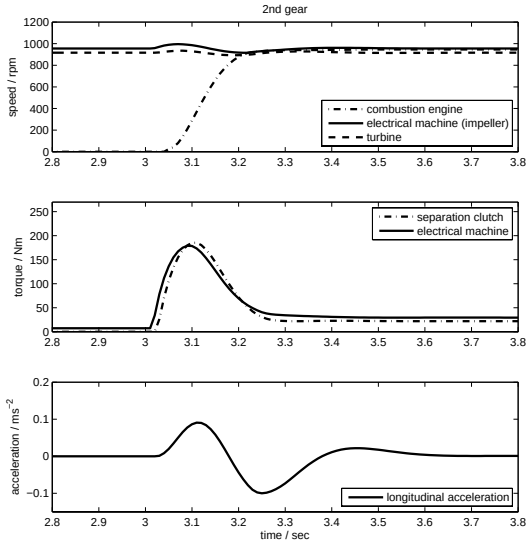


Figure 7: Speed synchronization, lock-up clutch open, constant driving, increased communication delay of separation clutch

5 Conclusions

A drivetrain model of a parallel hybrid vehicle and a multivariable controller for the problem of speed synchronization were proposed. The controller is capable of decoupling the rotational dynamics of the combustion engine and the rest of the drivetrain and it is simple enough to be calibrated manually. The results show, that the decoupling performance depends on the precision of the actuator models. If the dynamics of the separation clutch and of the electrical machine are exactly known, perfect decoupling, which is not influenced by parameter variations of the combustion engine and drivetrain, is achievable. The model following approach chosen for speed control of the electrical machine ensures, that no loss of driving comfort occurs due to the necessary change from open loop torque based operation mode of the propulsion management system to the closed loop speed based operation mode during synchronization.

Further work on the control strategy should focus on the enhancement of robustness considering model uncertainties of the torque actuators, especially the clutch. This can be done by extension of the vector performance index chosen for controller design. For example, worst-case scenarios could be taken into account. Furthermore, the nonlinear model could be used for optimization. Using feedback controllers other than PID might increase the decoupling performance. On the other hand controllers of greater complexity are in most cases no longer tunable by hand. This can be a problem for practical implementation, since the real plant will certainly show effects, which are not captured by the model.

References

[1] R. Beck, S. Saenger, F. Richert, A. Bollig, K. Neiss, T. Scholt, K.-E. Noreikat, D. Abel: *Model Predictive Control of a Parallel Hybrid Vehicle Drivetrain*, Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference, Sevilla, Spain, 2005.

[2] L. Guzzella and C.H. Onder: *Introduction to Modeling and Control of Internal Combustion Engine Systems*, Springer-Verlag Berlin Heidelberg, 2004

[3] J. B. Heywood: *Internal Combustion Engine Fundamentals*, McGraw-Hill, 1988

[4] U. Kiencke and L. Nielsen: *Automotive Control Systems*, Springer-Verlag Berlin Heidelberg, 2000

[5] G. Kreisselmeier and R. Steinhauser: *Systematische Auslegung von Reglern durch Optimierung eines vektoriellen Guetekriteriums*, Regelungstechnik, vol. 27, pp. 76-79, 1979

[6] G. Lechner and H. Naunheimer: *Automotive Transmissions*, Springer-Verlag Berlin Heidelberg New York, 1999

[7] H.-M. Streib and E. Wild: *Modellbasierte Füllungserfassung und -steuerung zur Verbindung von hoher dynamischer Aktualität und bester stationärer Genauigkeit*, Proceedings of the Symposium Steuerungssysteme für den Antriebsstrang von Kraftfahrzeugen, Berlin, 1997

Experimentelle Identifikation und Sliding Mode Regelung einer elektronischen Drosselklappe

Benedikt Alt und Ferdinand Svaricek
Institut für Systemdynamik und Flugmechanik
Universität der Bundeswehr München
benedikt.alt@unibw.de

Zusammenfassung

In diesem Beitrag wird ein modellbasiertes Reglerentwurfsverfahren vorgestellt, das in Zukunft zur robusten Lageregelung des Öffnungswinkels einer elektronischen Drosselklappe eingesetzt werden kann. Das Entwurfsverfahren basiert auf einem nichtlinearen Zustandsraummodell der Regelstrecke. Für die Modellbildung müssen wichtige Kenngrößen und Parameter experimentell identifiziert werden. Das vorgestellte Regelgesetz aus dem Bereich der Sliding Mode Regelung ist in der Lage, selbst bei Modellunsicherheiten und externen Störungen eine präzise Lageregelung des Öffnungswinkels der elektronischen Drosselklappe zu ermöglichen.

1 Einleitung

Die elektronische Drosselklappe stellt einen wichtigen Bestandteil des Motormanagements eines modernen Ottomotors dar, da das abgegebene Drehmoment sowohl bei Motoren mit Saugrohreinspritzung als auch mit Benzindirekteinspritzung von der zugeführten Luftmasse abhängt ([1]). Um das maximal mögliche Antriebsmoment des Motors zu beeinflussen, muß die Luftfüllung gedrosselt werden. Hierzu wird eine elektronische Drosselklappe eingesetzt, die den Öffnungsquerschnitt des Ansaugrohrs reduzieren kann. Der Öffnungsquerschnitt ergibt sich aus dem Öffnungswinkel $\theta(t)$ der Drosselklappe, wobei der zugehörige Sollwert $\theta_{soll}(t)$ im Motormanagementsystem berechnet wird. Bei der Sollwertgebung werden dabei sowohl die Momentenanforderung des Fahrerwunsches als auch die Anforderungen weiterer Verbraucher oder bestimmter Funktionen, wie z.B. der Leerlaufregelung berücksichtigt. Zwei berührungslose Sensoren liefern eine Rückmeldung über den aktuellen Drosselklappenwinkel $\theta(t)$. Aus dem Sollwert und der aktuellen Winkelposition kann dann eine geeignete Eingangsspannung $U_a(t)$ berechnet werden, so dass im gesamten Arbeitsbereich eine präzise Lageregelung des Öffnungswinkels ermöglicht wird. In dem vorliegenden Beitrag soll der bestehende Serienregler durch einen robusten und einfach zu implementierenden Regler ersetzt werden. Hierzu soll eine modellbasierte Entwurfsmethode ([2] - [4]) aus dem Bereich der Sliding Mode Regelung ([5] - [6]) eingesetzt werden, mit der vor allem

die nichtlinearen Reibungseffekte zuverlässig kompensiert werden können ([7] - [10]). In Abschnitt 2 wird der Aufbau und die Modellbildung für die elektronische Drosselklappe beschrieben. Danach wird das modellbasierte Regelgesetz zur robusten Lageregelung des Öffnungswinkels der elektronischen Drosselklappe hergeleitet. In Abschnitt 3 werden erste Simulationsergebnisse gezeigt. Abschnitt 4 enthält schließlich eine Zusammenfassung sowie einen Ausblick auf laufende sowie weiterführende Untersuchungen.

2 Modellbildung für die elektronische Drosselklappe

In diesem Abschnitt wird das nichtlineare Modell für die elektronische Drosselklappe beschrieben. In der Abbildung 1 ist der prinzipielle Aufbau der vorliegenden elektronischen Drosselklappe dargestellt. Die elektronische Drosselklappe besteht aus einem Metallgehäuse mit einer kreisrunden Öffnung, die durch eine Ventilklappe verschlossen werden kann. Diese Klappe ist auf einer Welle drehbar gelagert und kann über ein Getriebe mit einem Gleichstrommotor geöffnet oder geschlossen werden.

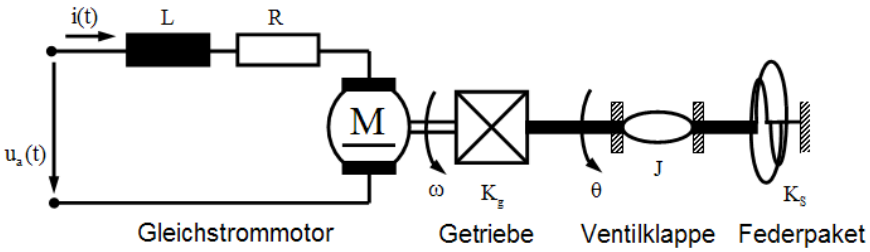


Abbildung 1: Aufbau der elektronischen Drosselklappe

Zudem sorgen zwei gegensinnig arbeitende Rückstellfedern dafür, dass die Drosselklappe bei einem Ausfall des elektrischen Bordnetzes geschlossen wird (Abbildung 2). Allerdings soll die Drosselklappe im unbestromten Zustand dennoch leicht geöffnet sein, damit der Motor in solchen Fällen nicht plötzlich abgestellt wird. Die Position des zugehörigen Öffnungswinkels wird auch als Notluftposition bezeichnet und beträgt ca. $\theta_{NLP} = 6.6\ ^\circ$.

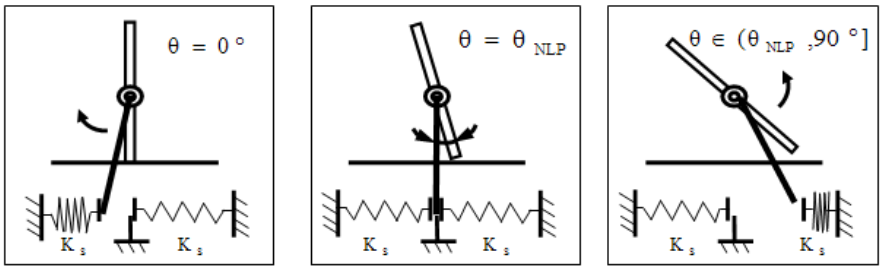


Abbildung 2: Federpaket der elektronischen Drosselklappe

Zur Modellbildung wird für den elektrischen Antrieb sowie für die rotierende mechanische Komponente der Drosselklappe jeweils ein eigenes mathematisches Modell hergeleitet. Die Beschreibung der Rotationsbewegung der mechanischen Komponenten der Drosselklappe erfolgt mit dem Drallsatz um die Welle der elektronischen Drosselklappe.

Dabei wirkt dem Moment aus dem elektrischen Antrieb $M_{el}(t)$ jeweils ein rückstellendes Moment durch das Federpaket $M_{Feder}(t)$ sowie durch Reibungseffekte $M_{Reibung}(t)$ entgegen:

$$J_{ges}\ddot{\theta}(t) = M_{el}(t) - M_{Feder}(t) - M_{Reibung}(t) \tag{1}$$

In Gleichung (1) bezeichnet J_{ges} das gesamte Massenträgheitsmoment aller rotierenden mechanischen Komponenten. Für das Moment $M_{el}(t)$ aus dem elektrischen Antrieb wird angenommen, dass ein linearer Zusammenhang zu dem Ankerstrom $i(t)$ existiert. Der zugehörige Proportionalitätsfaktor heißt Motorkonstante und wird mit K_t bezeichnet. Die Notluftposition führt dazu, dass das Rückstellmoment durch die Feder $M_{Feder}(t)$ sowie durch die Reibung $M_{Reibung}(t)$ genau an dieser Stelle das Vorzeichen umkehrt und sowohl im Winkel $\theta(t)$ als auch in der Winkelgeschwindigkeit $\dot{\theta}(t)$ jeweils eine Unstetigkeit auftritt. Für Winkel unterhalb sowie oberhalb der Notluftposition wird ein linearer Zusammenhang zwischen dem Rückstellmoment für die Reibung $M_{Reibung}(t)$ und der Winkelgeschwindigkeit $\omega(t)$ angenommen. Ebenso soll dort auch ein linearer Zusammenhang zwischen dem Rückstellmoment der Feder $M_{Feder}(t)$ und dem Öffnungswinkel $\theta(t)$ angemommen werden. Streng genommen unterscheiden sich auch die Rückstellmomente durch die Feder unterhalb und oberhalb der Notluftposition in ihren Federkonstanten. Dieser Effekt wird aber für den Reglerentwurf in diesem Ansatz nicht weiter berücksichtigt. Für die Rückstellmomente durch die Feder sowie durch die Reibung gilt somit:

$$M_{Feder} = -K_S \operatorname{sgn}(\theta - \theta_{NLP}) - K_L(\theta - \theta_{NLP}) \tag{2}$$

$$\text{und } M_{Reibung} = -M_S \operatorname{sgn}(\omega) - M_C\omega \ , \tag{3}$$

wobei $\omega = K_g\dot{\theta}$ ist. Für den Gleichstrommotor im elektrischen Antrieb verwendet man einen linearen Modellansatz. Somit kann unter Berücksichtigung von Abbildung 2 folgende Differentialgleichung für den Ankerstrom $i(t)$ im Gleichstrommotor aufgestellt werden:

$$L \cdot \frac{d}{dt}i(t) = -Ri(t) + u_a(t) - K_m\omega(t) \ . \tag{4}$$

Dabei bezeichnet der Modellparameter L die Induktivität und R den Widerstand des Gleichstrommotors. Im stationären Betrieb des Gleichstrommotors kann die Motorkonstante K_m zudem mit der Motorkonstanten K_t gleichgesetzt werden. Die Gleichungen (1) - (4) können somit als nichtlineares Zustandsraummodell formuliert werden

$$\dot{\theta} = \frac{1}{K_g}\omega \tag{5}$$

$$\dot{\omega} = -\frac{K_L}{K_gJ}(\theta - \theta_{NLP}) - \frac{M_C}{K_gJ}\omega + \frac{K_t}{J}i - \frac{K_S}{K_gJ}\operatorname{sgn}(\theta - \theta_{NLP}) - \frac{M_S}{K_gJ}\operatorname{sgn}(\omega) \tag{6}$$

$$\frac{d}{dt}i = -\frac{K_t}{L}\omega - \frac{1}{\tau}i + \frac{1}{L}u_a \ , \tag{7}$$

wobei $\tau = \frac{L}{R}$ ist. Im Folgenden sollen die Modellparameter aus dem nichtlinearen Zustandsraummodell der Gleichungen (5) - (7) experimentell identifiziert werden. Zunächst sollen die Modellparameter des elektrischen Antriebs (Widerstand R, Induktivität L und Motorkonstante K_m) bestimmt werden. Hierzu wird der Gleichstrommotor aus der Drosselklappe ausgebaut. Um den Widerstand R des Gleichstrommotors zu bestimmen, wird der Motor langsam gedreht und alle 30 Grad eine Widerstandsmessung durchgeführt.

Aus den gemessenen Werten ergibt sich der Mittelwert schließlich zu $R = 2.83\ \Omega$. Die Abbildung 2 zeigt die entsprechenden experimentellen Ergebnisse.

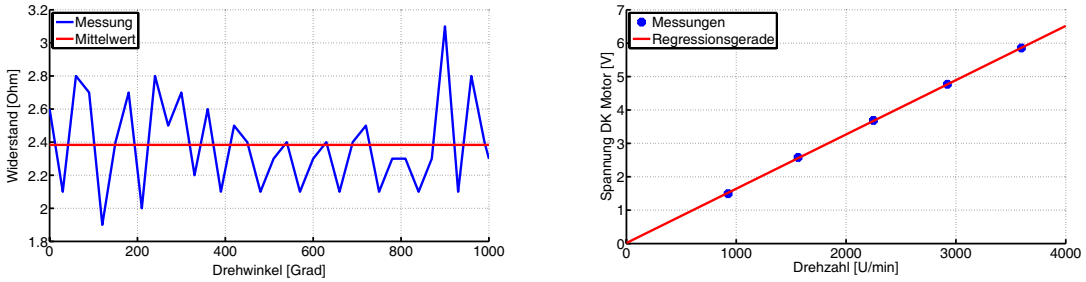


Abbildung 3: Messungen zur Identifikation des Widerstands des Gleichstrommotors (links), Messungen zur Identifikation der Motorkonstante des Gleichstrommotors (rechts)

Um die Motorkonstante $K_m = K_t$ zu identifizieren, wird der Gleichstrommotor der elektronischen Drosselklappe mit einem zweiten Motor über eine Kupplung verbunden. Somit kann der Gleichstrommotor der elektronischen Drosselklappe extern angetrieben werden. Gleichzeitig wird an den Klemmen des Motors der elektronischen Drosselklappe die sich einstellende Spannung U_{mot} gemessen. Die gemeinsame Drehzahl N beider Motoren wird über einen Tachogenerator bestimmt, der auf der Rückseite des antreibenden Motors angebracht ist. In Abbildung 2 sind die stationären Werte der Spannung U_{mot} für verschiedene Drehzahlen über den stationären Werten der Drehzahl N aufgetragen. Die Motorkonstante $K_m = K_t$ ergibt sich schließlich aus der Steigung der Regressionsgeraden und sie beträgt $K_m = 0.015$ Vmin/U.

Zur Identifikation der Induktivität L wird der Gleichstrommotor wieder in die elektronische Drosselklappe eingebaut. Um die Induktivität experimentell zu bestimmen, werden ähnliche Laborversuche durchgeführt wie in [4]. Dabei wird die Induktivität jeweils bei verschiedenen Öffnungswinkeln θ und verschiedenen Eingangsspannungen zwischen $U_a = 5$ V und $U_a = 15$ V identifiziert. In der Abbildung 4 sind die Werte der Induktivität für verschiedene Eingangsspannungen U_a und für verschiedene Öffnungswinkel θ aufgetragen. Aus allen gemessenen Werten erhält man schließlich als Mittelwert für die Induktivität einen Wert von $L = 7.29$ mH. Zur Identifikation der übrigen unbekannten Modellparameter werden ebenfalls ähnliche Laborversuche durchgeführt wie in [4]. Aus Platzgründen kann darauf aber an dieser Stelle nicht genauer eingegangen werden.

Im Abschnitt 3 werden die Modellparameter für den Reglerentwurf zu verallgemeinerten Parametern zusammengefasst. Danach wird auf Grundlage des nichtlinearen Zustandsraummodells der Gleichungen (5) - (7) ein Regelgesetz zur robusten Lageregelung des Öffnungswinkels der elektronischen Drosselklappe hergeleitet.

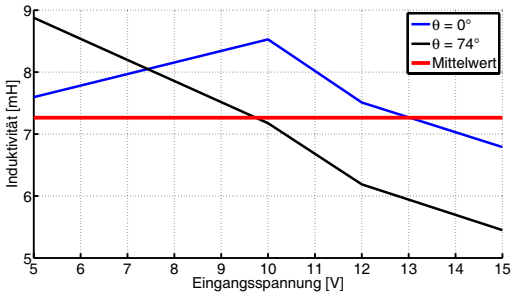


Abbildung 4: Messungen zur Identifikation der Induktivität des Gleichstrommotors

3 Robuste Lageregelung des Öffnungswinkels der elektronischen Drosselklappe

In diesem Abschnitt soll ein Regelgesetz zur robusten Lageregelung des Öffnungswinkels hergeleitet werden. Hierzu soll ein Entwurfsverfahren aus dem Bereich der Sliding Mode Regelung eingesetzt werden ([10]). Die Sliding Mode Regelung gehört zu den strukturvariablen Reglerentwurfsverfahren und ist insbesondere für ihre Robustheit gegenüber Modellunsicherheiten und externen Störungen bekannt.

$a_{12} = \frac{1}{K_g}$	$a_{21} = \frac{K_L}{K_g J}$	$a_{22} = \frac{M_C}{K_g J}$
$a_{23} = \frac{K_t}{J}$	$a_{32} = \frac{K_t}{L}$	$a_{33} = \frac{R}{L}$
$\kappa = \frac{K_S}{K_g J}$	$\mu = \frac{M_S}{K_g J}$	$b = \frac{1}{L}$

Tabelle 1: Parameter des Modells der Drosselklappe

Im ersten Schritt werden die Modellparameter aus dem nichtlinearen Zustandsraummodell der Gleichungen (5) - (7) gemäß Tabelle 1 zusammengefasst. Somit erhält man für das nichtlineare Zustandsraummodell der elektronischen Drosselklappe die folgende Form:

$$\dot{\theta} = a_{12}\omega \tag{8}$$

$$\dot{\omega} = -a_{21}(\theta - \theta_{NLP}) - a_{22}\omega + a_{23}i - \kappa \text{sgn}(\theta - \theta_{NLP}) - \mu \text{sgn}(\omega) \tag{9}$$

$$\frac{d}{dt}i = -a_{32}\omega - a_{33}i + bu \ . \tag{10}$$

Im nächsten Schritt definiert man für das System aus den Gleichungen (8) - (10) zwei neue Zustandsgrößen mit $x_1 = \theta$ und $x_2 = \dot{x}_1 = a_{12}\omega$. Weiterhin wird berücksichtigt, dass die Dynamik des elektrischen Antriebs wesentlich schneller ist als die Dynamik der rotierenden mechanischen Komponenten und somit die Annahme $L \frac{d}{dt}i(t) << iR$ gerechtfertigt ist. Daraus ergibt sich das reduzierte nichtlineare Zustandsraummodell

$$\dot{x}_1 = x_2 \tag{11}$$

$$\dot{x}_2 = a_{12}\left[-a_{21}(x_1 - x_{1,NLP}) - \frac{a_{22}}{a_{12}}x_2 + a_{23}\tilde{u} - \kappa \text{sgn}(x_1 - x_{1,NLP}) - \mu \text{sgn}\left(\frac{x_2}{a_{12}}\right)\right], \tag{12}$$

wobei $\tilde{u} = \frac{1}{b}u$ ist. Im Folgenden soll auf Grundlage des reduzierten Zustandsraummodells aus den Gleichungen (11) und (12) ein Regelgesetz hergeleitet werden, so dass der Öff-

nungswinkel $\theta(t)$ seinem Sollwert $\theta_{soll}(t)$ nachgeführt wird. Hierzu wird im Folgenden ein Regelgesetz der Form

$$\tilde{u}(t) = \tilde{u}_1(t) + \tilde{u}_2(t) \tag{13}$$

verwendet. Zunächst soll der Regelanteil $\tilde{u}_1(t)$ hergeleitet werden, der dafür sorgt, dass die Nichtlinearitäten der Gleichung (12) im Fall einer konstanten Winkelgeschwindigkeit $\omega(t)$ kompensiert werden. Hierzu wird die Schaltfläche

$$\sigma_1 = x_2 - x_{2,soll} = 0 \tag{14}$$

definiert. Um den Regelanteil $\tilde{u}_1(t)$ zu berechnen, wird die zeitliche Ableitung der Schaltfunktion σ_1 gebildet und gleich null gesetzt:

$$\dot{\sigma}_1 = \dot{x}_2 - \dot{x}_{2,soll} = 0 \tag{15}$$

Aus der Voraussetzung einer konstanten Winkelgeschwindigkeit folgt die Bedingung $x_{2,soll} = 0$ und ebenso $\dot{x}_{2,soll} = 0$. Damit erhält man schließlich

$$\tilde{u}_1 = \frac{1}{a_{12}a_{23}} \left[a_{12}a_{21}(x_1 - x_{1,NLP}) + a_{22}x_2 + a_{12}\kappa \mathrm{sgn}(x_1 - x_{1,NLP}) + a_{12}\mu \mathrm{sgn}\left(\frac{x_2}{a_{12}}\right) \right] . \tag{16}$$

Wie oben bereits beschrieben können die Nichtlinearitäten aus der Gleichung (12) nur bei einer konstanten Winkelgeschwindigkeit $\omega(t)$ sowie unter Nominalbedingungen kompensiert werden. Wenn diese Voraussetzungen nicht eingehalten werden, treten Modellunsicherheiten und Auswirkungen auf die Regelgüte auf. Daher soll im zweiten Entwurfsschritt der Regelanteil $\tilde{u}_2(t)$ hergeleitet werden, der dafür sorgt, dass der Öffnungswinkel $\theta(t)$ auch bei Modellunsicherheiten und externen Störungen seinem Sollwert $\theta_{soll}(t)$ nachgeführt wird. Hierzu werden alle möglichen Unsicherheiten und Störungen im System in der Funktion $\Phi(x_1, x_2, t)$ zusammengefasst. Zudem wird angenommen, dass diese Funktion mit $\Phi(x_1, x_2, t) \leq \Phi_{Max}$ eine obere Schranke besitzt. Die Modellunsicherheiten und Störungen lassen sich nun im reduzierten Zustandsraummodell berücksichtigen, indem das Regelgesetz aus der Gleichung (13) in das Modell aus den Gleichungen (11) und (12) eingesetzt wird und zusammen mit den Modellungenauigkeiten der Funktion $\Phi(x_1, x_2, t)$ betrachtet wird:

$$\dot{x}_1 = x_2 \tag{17}$$

$$\dot{x}_2 = \Phi(x_1, x_2, t) + \tilde{u}_2 \tag{18}$$

Im nächsten Schritt wird die Schaltfläche

$$\sigma_2 = (x_1 - x_{1,soll}) + \lambda_2 x_2 = 0 \tag{19}$$

mit $\lambda_2 > 0$ definiert. Um den Regelanteil $\tilde{u}_2(t)$ zu berechnen, wird die zeitliche Ableitung der Schaltfunktion σ_2 gebildet und eine Dynamik für σ_2 durch

$$\dot{\sigma}_2 = (\dot{x}_1 - \dot{x}_{1,soll}) + \lambda_2 \dot{x}_2 = -K_{L,2}\sigma_2 - K_{S,2}\mathrm{sgn}(\sigma_2) \tag{20}$$

vorgegeben. Löst man die Gleichung (20) nach $\dot{x}_2$ auf, so erhält man unter der Annahme von Gleichung (11) und $\dot{x}_{1,soll} = 0$

$$\dot{x}_2 = -\frac{1}{\lambda_2}\left[x_2 - K_{L,2}\sigma_2 - K_{S,2}\mathrm{sgn}(\sigma_2)\right] \ . \tag{21}$$

Danach können die Gleichungen (13) und (16) in die Gleichung (12) eingesetzt werden. Zusammen mit den Gleichungen (18) und (21) kann dann der Regelanteil $\tilde{u}_2(t)$ berechnet werden:

$$\tilde{u}_2 = -\frac{1}{\lambda_2 a_{12} a_{23}}\left[x_2 + K_{L,2}\sigma_2 + K_{S,2}\mathrm{sgn}(\sigma_2)\right]$$

Unter der Annahme, dass die Bedingung $\frac{K_{L,2}}{\lambda_2}\left|\sigma_2\right| + \frac{K_{S,2}}{\lambda_2} \geq \Phi_{Max}$ erfüllt ist, wird sichergestellt, dass $\sigma_2\dot{\sigma}_2 < 0$ ist und somit die Schaltfläche $\sigma_2 = 0$ in endlicher Zeit erreicht wird. Der zugehörige Stabilitätsnachweis basiert auf einem Lyapunovansatz und ist in [6] enthalten. Im letzten Entwurfsschritt wird ein Regelgesetz berechnet, das sicherstellt, dass der Ankerstrom $i(t)$ seinem Sollwert $i_{soll}(t) = \tilde{u}(t)$ aus der Gleichung (13) nachgeführt wird. Hierzu wird die Schaltfläche

$$\sigma_3 = i - i_{soll} = 0 \tag{22}$$

definiert. Um die Eingangsspannung $u(t) = U_a(t)$ des elektrischen Antriebs zu berechnen, wird die zeitliche Ableitung der Schaltfunktion σ_3 gebildet und eine Dynamik für σ_3 durch

$$\dot{\sigma}_3 = \dot{i} - \dot{i}_{soll} = -K_{S,3}\mathrm{sgn}(\sigma_3) \tag{23}$$

vorgegeben. Zusammen mit der Gleichung (10) und der Annahme $\frac{d}{dt}i_{soll} = 0$ könnte nun ein Regelgesetz für die Eingangsspannung $U_a(t)$ des elektrischen Antriebs berechnet werden. Setzt man allerdings wie oben erwähnt voraus, dass die Dynamik des elektrischen Antriebs wesentlich schneller ist als die Dynamik der rotierenden mechanischen Komponenten und die Annahme $\frac{d}{dt}i = 0$ gilt, so vereinfacht sich das Regelgesetz für die Eingangsspannung des elektrischen Antriebs zu

$$u(t) = -\frac{K_{S,3}}{b}\mathrm{sgn}(\sigma_3(t)) \ . \tag{24}$$

Mit dem Regelgesetz aus Gleichung (24) und der Bedingung $\frac{K_{S,3}}{b} \leq U_{a,max}$ kann die Stellgrößenbeschränkung des elektrischen Antriebs direkt im Reglerentwurf mitberücksichtigt werden. Somit ist der Reglerentwurf abgeschlossen. Bei der vorliegenden elektronischen Drosselklappe liegt lediglich der Öffnungswinkel $\theta(t)$ als Messgröße vor. Zur Realisierung des oben beschriebenen Regelgesetzes werden allerdings zusätzlich auch die Winkelgeschwindigkeit $\omega(t)$ und der Ankerstrom $i(t)$ benötigt. Somit muss ein Zustandsbeobachter entworfen werden, der aus dem gemessenen Öffnungswinkel $\theta(t)$ und der berechneten Stellgröße $u(t)$ alle benötigten Zustandsgrößen rekonstruiert. Um eine robuste Schätzung $\hat{\theta}(t)$, $\hat{\omega}(t)$ und $\hat{i}(t)$ der Zustandsgrößen zu ermöglichen, soll ein Sliding Mode Beobachter der Form

$$\dot{\hat{\theta}}_1 = a_{12}\hat{\omega} + l_1\mathrm{sgn}(\theta - \hat{\theta}) \tag{25}$$

$$\dot{\hat{\omega}} = -a_{21}(\hat{\theta} - \theta_{NLP}) - a_{22}\hat{\omega} + a_{23}\hat{i} - \tilde{\kappa}\mathrm{sgn}(\hat{\theta} - \theta_{NLP}) - \tilde{\mu}\mathrm{sgn}(\hat{\omega}) + l_2\mathrm{sgn}(\theta - \hat{\theta}) \tag{26}$$

$$\frac{d}{dt}\hat{i} = -a_{32}\hat{\omega} - a_{33}\hat{i} + b_3 u + l_3(\theta - \hat{\theta}) \ . \tag{27}$$

eingesetzt werden. Dabei stehen $\tilde{\kappa}$ und $\tilde{\mu}$ für die Werte der beiden Modellparameter κ und μ unter Nominalbedingungen. Definiert man den Fehler zwischen den gemessenen und den geschätzten Zustandsgrößen der elektronischen Drosselklappe als $e_\theta = \theta - \hat{\theta}$, $e_\omega = \omega - \hat{\omega}$ sowie als $e_i = i - \hat{i}$, so erhält man für die Dynamik des Beobachtungsfehlers

$$\dot{e}_\theta = a_{12}e_\omega - l_1\text{sgn}(e_\theta) \tag{28}$$

$$\dot{e}_\omega = a_{21}e_\theta + a_{21}\theta_{NLP} + a_{22}e_\omega + a_{23}e_i + \tag{29}$$

$$- \left[\kappa\text{sgn}(\theta - \theta_{NLP}) - \tilde{\kappa}\text{sgn}(\hat{\theta} - \theta_{NLP})\right] - [\mu\text{sgn}(\omega) - \tilde{\mu}\text{sgn}(\hat{\omega})] - l_2\text{sgn}(e_\theta) \tag{30}$$

$$\dot{e}_i = a_{32}e_\omega + a_{33}e_i - l_3e_\theta \ . \tag{31}$$

Wenn die Rückführverstärkung l_1 die Bedingung $l_1 > |a_{12}e_\theta|$ erfüllt, wird sichergestellt, dass $e_\theta\dot{e}_\theta < 0$ ist und die Schaltfläche $e_\theta = 0$ in endlicher Zeit erreicht wird. Sofern die Bedingung $l_2 > a_{22}\,|\theta_{NLP}| + 3\,|\tilde{\kappa}| + 3\,|\tilde{\mu}| + a_{23}e_i$ erfüllt ist, kann ebenso sichergestellt werden, dass die Schaltfläche $e_\omega = 0$ in endlicher Zeit erreicht wird. Der zugehörige Stabilitätsnachweis ist wiederum in [6] enthalten. Wie oben bereits beschrieben, ist die Dynamik des elektrischen Antriebs wesentlich schneller als die Dynamik der Rotationsbewegung der elektronischen Drosselklappe. Um sicherzustellen, dass der Fehler e_i zwischen dem gemessenen und dem geschätzten Ankerstrom gegen null konvergiert, reicht es somit bereits aus, wenn der Modellparameter l_3 die Bedingung $l_3 > \frac{a_{32}}{a_{12}}l_1$ erfüllt. Genau dann ist ein asymptotisches Konvergenzverhalten gegen die Schaltfläche $e_i = 0$ sichergestellt.

4 Simulationsergebnisse

In diesem Abschnitt soll mit repräsentativen Simulationsergebnissen gezeigt werden, dass das Regelgesetz aus dem Abschnitt 3 in der Lage ist, den Öffnungswinkel $\theta(t)$ seinem Sollwert $\theta_{soll}(t)$ nachzuführen.

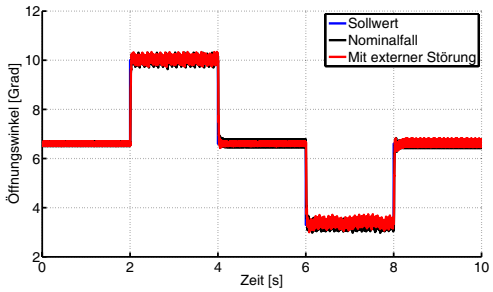


Abbildung 5: Kleinsignalverhalten: Sprungantworten im Öffnungswinkel der elektronischen Drosselklappe

Hierzu soll sowohl das Kleinsignal- als auch das Großsignalverhalten der elektronischen Drosselklappe betrachtet werden. Dabei werden die Reglerparameter in der nichtlinearen Simulation wie folgt festgelegt: $\lambda_2 = 1200$, $K_{L,2} = 100$, $K_{S,2} = 100$ und $K_{S,3} = 13.8$. In den Abbildungen 5 und 6 sind der Sollwert des Öffnungswinkels $\theta_{soll}(t)$ und der Öffnungswinkel $\theta(t)$ über der Zeit dargestellt. Dabei ist deutlich zu sehen, dass das Regelgesetz aus Abschnitt 3 selbst bei Modellunsicherheiten und externen Störungen in der Lage ist, sowohl

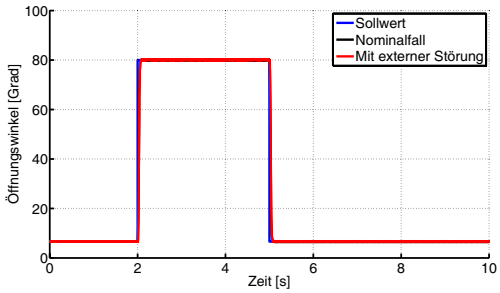


Abbildung 6: Großsignalverhalten: Sprungantworten im Öffnungswinkel der elektronischen Drosselklappe

ein ausreichend schnelles Einschwingverhalten als auch eine hohe stationäre Genauigkeit sicherzustellen. Um eine externe Störung mit zu berücksichtigen, wurde in der Simulation zusätzlich ein Störsignal der Form $z(t) = \sin(2\pi\frac{t}{T}) + 10$ mit $T = \frac{\pi}{5}$ auf die Winkelbeschleunigung aufgeschaltet. In den aufgeführten Simulationen wird die Eingangsspannung $U_a(t)$ mit einer fest vorgegebenen Frequenz von $f = 1$ kHz zwischen ihren Maximalwerten umgeschaltet. Zudem ist durch das Regelgesetz aus Gleichung (24) gewährleistet, daß die Stellgrößenbeschränkung von $U_{a,max} = 13.8$ V selbst im Großsignalverhalten nicht überschritten wird.

5 Zusammenfassung und Ausblick

In diesem Beitrag wird ein neues Konzept zur robusten Lageregelung der elektronischen Drosselklappe vorgestellt. Zunächst wurde ein nichtlineares Zustandsraummodell der elektronischen Drosselklappe hergeleitet, das sowohl die Rotationsbewegung als auch die Dynamik des Ankerstroms beschreibt. Zudem wurden die Auswirkungen der Notluftposition auf den Öffnungswinkel und die Winkelgeschwindigkeit der elektronischen Drosselklappe berücksichtigt. Im nächsten Schritt wurden die benötigten physikalischen Modellparameter experimentell identifiziert. Danach wurde ein Regelgesetz berechnet, das eine robuste Lageregelung des Öffnungswinkels der elektronischen Drosselklappe ermöglicht. Hierzu wurde eine modellbasierte Entwurfsmethode aus dem Bereich der Sliding Mode Regelung eingesetzt. Dabei wurde gezeigt, dass das Regelgesetz in der Lage ist, die Auswirkungen von Modellunsicherheiten und externen Störungen zu kompensieren. Um das Regelgesetz an der realen Strecke implementieren zu können, wird ein Zustandsbeobachter benötigt, der aus dem gemessenen Öffnungswinkel und der Stellgröße die Winkelgeschwindigkeit und den Ankerstrom rekonstruieren kann. Hierzu wurde ein Sliding Mode Beobachter entworfen, der eine robuste Schätzung der Zustandsgrößen ermöglicht. Anschließend wurde in einer Simulationsstudie gezeigt, dass das vorgestellte Regelgesetz sowohl im Klein- als auch im Großsignalverhalten in der Lage ist, den Öffnungswinkel seinem Sollwert nachzuführen. Die Stellgrößenbeschränkungen der realen Strecke wurden in dieser Simulationsstudie dabei ebenfalls bereits mitberücksichtigt. Die aktuellen Arbeiten haben das Ziel, das vorgestellte Regelgesetz zu diskretisieren und danach am institutseigenen La-

borprüfstand zu implementieren. Dabei soll darauf geachtet werden, dass eine spätere Verwendung des Reglers im Serienfahrzeug soweit wie möglich vorbereitet wird. Somit muß z.B. die berechnete Stellgröße in ein PWM-Signal umgerechnet werden. Das Ziel der zukünftigen Untersuchungen besteht darin, die Auswirkungen der Modellungenauigkeiten durch Temperatureinflüsse oder Alterungserscheinungen zuverlässig zu kompensieren. Hierzu könnten die Modellparameter der Drosselklappe z.B. in regelmäßigen Abständen online neu identifiziert werden.

Literatur

[1] Bosch, R., GmbH: *Ottomotor-Management, Systeme und Komponenten, 3. Auflage*, Vieweg und Sohn Verlag, Wiesbaden 2005.

[2] Teuscher, E., Alt, B., Svaricek, F., Blath, J.P. und Schultalbers, M.: *Linear and Nonlinear Modeling for an Electronic Throttle Body*, Proceedings of the Eurosim 2007, Ljubiljana, Slovenia, 2007.

[3] Winkelhake, J.: *Entwicklung eines Regelkonzepts für die Lageregelung einer Drosselklappe*, Diplomarbeit, Institut für Elektrische Informationstechnik, Universität Clausthal, Germany, 2000.

[4] Pavkovic, D., Deur, J., Jansz, M., und Peric, N.: *Experimental Identification of an Electronic Throttle Body*, Proceedings of the 10th European Conference of Power Electronics and Applications (EPE 2003), Toulouse, France, 2003.

[5] Edwards, C. und Spurgeon, S.K.: *Sliding Mode Control, Theory and Applications*, Taylor & Francis LTD, London, UK, 1999.

[6] Utkin, V., Guldner, J. und Shi, J.: *Sliding Mode Control in Electromechanical Systems*, Taylor & Francis LTD, London, UK, 1999.

[7] Deur, J., Pavkovic, D., Peric, N., Jansz, M. und Hrovat, D.: *An Electronic Throttle Control Strategy Including Compensation of Friction and Limp-Home Effects*, IEEE Transactions on Industry Applications, 2000, Vol. 40, Seiten 821-834.

[8] Pan, Y., Dagci, O. und Özgüner, Ü.: *Variable Structure Control of Electronic Throttle Valve*, Proceedings of the IEEE International Vehicle Electronics Conference 2001, Seiten 103-108, 2001.

[9] Dagci, O.H., Pan, Y. und Özgüner, Ü.: *Sliding Mode Control of Electronic Throttle Valve*, Proceedings of the American Control Conference, Anchorage, AK, USA, 2002.

[10] Hatipoglu, C., und Özgüner, Ü.: *Handling Stiction with Variable Structure Control*, Variable Structure Systems, Sliding Mode and Nonlinear Control, Springer-Verlag, London, Berlin, Heidelberg, 1999.

Performanzindizes-basierte Selbsteinstellung der Reglerparameter im Motormanagementsystem

G. Yang[*], T. Jeinsch[+], S.X. Ding[*], N. Weinhold[+], M. Schultalbers[+]

*Universität Duisburg-Essen, Institut für Automatisierungstechnik und komplexe Systeme, 47057 Duisburg, Germany (steven.ding@uni-due.de)

[+]Ingenieurgesellschaft Auto und Verkehr GmbH, Powertrain Mechatronik Ottomotoren Systeme, 38518 Gifhorn, Germany (torsten.jeinsch@iav.de)

Zusammenfassung

Das moderne Motormanagement von Verbrennungsmotoren fußt auf der Anwendung komplexer Steuerungs- und Regelungskonzepte, um den wachsenden Anforderungen in Bezug auf die Verringerung des Kraftstoffverbrauchs und der Schadstoffemission, sowie zur Verbesserung der Leistungsfähigkeit gerecht zu werden. In heutigen Steuergeräten finden eine Vielzahl von Regelungs- und Diagnosefunktionen mit zahlreichen Konstanten, Kennlinien und Kennfeldern Anwendung. Die Aufgabe der Applikation ist die Bestimmung der optimalen Werte für diese Kennlinien und Kennfelder, um gesetzliche Vorgaben und die Erwartungen des Kunden zu erfüllen. Aufgrund des oftmals nichtlinearen Verhaltens und der großen Anzahl von Einflussgrößen der einzelnen Regelstrecken, ist hierfür eine große Anzahl von Messungen am Motorenprüfstand erforderlich. Die zunehmende Komplexität der Steuergeräte führt somit zu steigenden Applikationskosten. Gleichzeitig sinken jedoch die zur Verfügung stehenden Entwicklungszeiten- und -budgets. Es müssen somit neue Wege gefunden werden, die Applikationskosten und -zeiten zu senken. Neben der Automatisierung der Prüfstände und der Anwendung von modellgestützten Methoden zur Applikation stellen Performanzindizes-basierte Verfahren für die Selbsteinstellung von Reglerparametern eine neue Herausforderung dar.
In diesem Beitrag wird ein Performanzindizes-basiertes Verfahren für die Selbsteinstellung von Reglerparameter vorgestellt. Die angewendeten Performanzindizes, sowohl klassische als auch Indizes aus den CPM (Control Loop Performance Monitoring) Methoden, sind robust gegenüber Rauschen und lassen sich ohne wesentliche Systemvorkenntnisse direkt aus ungefilterten Rohdaten errechnen. Zur Validierung wurde das Verfahren in ein Steuergerät, zur Einstellung der Reglerparameter im Raildruckregelkreis, implementiert und an einem Motorenprüfstand der IAV GmbH getestet. Die Ergebnisse der Untersuchung werden am Ende des Beitrages dargestellt.

1. Einleitung

Das moderne Motormanagement von Verbrennungsmotoren fußt auf der Anwendung komplexer Steuerungs- und Regelungskonzepte, um den wachsenden Anforderungen in Bezug auf die Verringerung des Kraftstoffverbrauchs und der Schadstoffemission sowie zur Verbesserung der Leistungsfähigkeit gerecht zu werden. In heutigen Steuergeräten finden eine Vielzahl von Regelungs- und Diagnosefunktionen mit zahlreichen Konstanten, Kennlinien und Kennfeldern Anwendung. Diese erlauben es, den Verbrennungsmotor trotz seines stark nichtlinearen Verhaltens noch effizienter zu beherrschen. Dies hat allerdings zur Folge, dass die Anzahl der zu applizierenden Regler- und Vorsteuerungsparameter schnell anwächst und der Applikationsprozess immer komplexer und zeitintensiver wird. Gleichzeitig werden die verfügbaren Entwicklungszeiten ständig verkürzt. Aus diesem Grund wurden in den

vergangenen Jahren Methoden untersucht, die bei Gewährleistung der Applikationsqualität den Messaufwand vermindern sollen. In den letzten Jahren wurde in (Millich et al. 2006) und (Tomforde et al. 2007) Verfahren vorgeschlagen, die durch den Einsatz von physikalischen Simulationsmodellen den Messaufwand deutlich reduzieren und die Effizienz des Applikationsprozesses steigern. Allerdings werden die Applikationsergebnisse wesentlich durch die Qualität des Simulationsmodells bestimmt. Aufgrund der Komplexität und der starken Nichtlinearität des Motors ist ein hoher Engineeringaufwand notwendig, um ein entsprechendes physikalisches Modell für die Vielzahl der Arbeitspunkte zu finden.

Eine Idee zur Reduzierung dieses bestehenden Nachteils ist die Kombination der simulationsgestützten Applikation mit Verfahren zur Selbsteinstellung. Das Verfahren besteht aus zwei Schritten: Erstens wird eine simulationsgestützte Vorapplikation ausgeführt, um eine Erstbedatung mit dem physikalischen Simulationsmodell zu erhalten. Es ist zu beachten, dass die Erstapplikation keine optimalen Parameter liefern muss. Vielmehr liefert sie als Voreinstellung die Anfangswerte für die Selbsteinstellung. Aus diesem Grund kann der Engineeringaufwand für die Modellierung deutlich reduziert werden. Zweitens wird ein online Verfahren zur Selbsteinstellung eingesetzt, um die Parameterisierung aus der Vorapplikation rekursiv im Sinne des gewählten Gütekriteriums zu optimieren. Durch die Kombination der simulationsgestützten Vorapplikation und der online Selbsteinstellung lassen sich, ohne Qualitätsverlust der Applikation, der Aufwand bei der Modellbildung und die Applikateursarbeiten am Prüfstand durch das automatisierte Verfahren zur Selbsteinstellung deutlich reduzieren.

Diese Idee wird im Folgenden am Beispiel der Parametrisierung des Reglers eines mengengesteuerten Hochdruck-Kraftstoffsystems für Ottomotoren mit Benzin-Direkteinspritzung erläutert. Im zweiten Teil des Beitrags werden die Grundlagen des Hochdruck-Kraftstoffsystems und seine Regelungsstruktur erläutert, im dritten Teil wird ein Selbsteinstellungsverfahren auf Basis von Performanzindizes vorgestellt und im vierten Teil werden die Messergebnisse am Motorprüfstand für die Validierung der vorgestellten Idee gezeigt.

2. Überblick über das Hochdruck-Kraftstoffsystem

Das betrachtete Hochdruck-Kraftstoffsystem für Ottomotoren mit Benzin-Direkteinspritzung besteht aus einer Kraftstoffhochdruckpumpe, dem Kraftstoffverteiler, einem Drucksensor, einem Druckbegrenzungsventil und den Einspritzventilen. In der Abbildung 1 ist das Hochdruck-Kraftstoffsystem schematisch dargestellt.

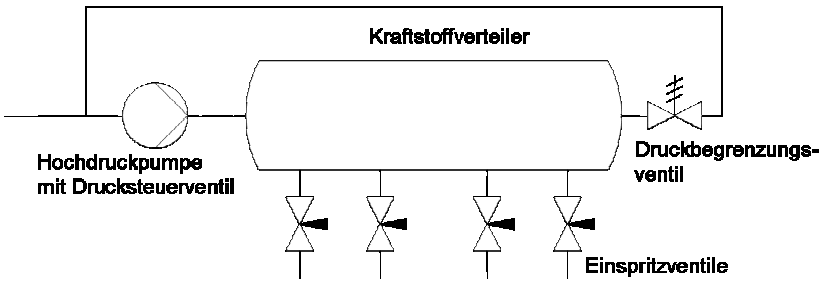


Abbildung 1. Bedarfsgeregelter Hochdruckkreis eines Ottomotors (Blath 2006)

Die durch die Nockenwelle angetriebene Hochdruckpumpe erzeugt den Hochdruck im Kraftstoffsystem und fördert den Kraftstoff aus dem Niederdruckbereich in den Kraftstoffverteiler. Der Kraftstoff wird im Kraftstoffverteiler gespeichert und gelangt durch

die angeschlossenen Einspritzventile in die Brennräume. Bei zu hohem Kraftstoffdruck öffnet das Druckbegrenzungsventil zum Schutz der Bauteile.

Abhängig vom Arbeitspunkt des Motors wird der Druck im Verteiler, wie in der Abbildung 2 gezeigt, mit einem PI-Regler und einer Vorsteuerung eingestellt. Der PI-Regler hat die Aufgabe, auftretende Störungen auszuregeln und den Regelkreis zu stabilisieren, während die Vorsteuerung dafür sorgt, die Regelgüte im Führungsverhalten zu verbessern. Die Parameter des Reglers und der Vorsteuerung werden aufgrund des stark nichtlinearen Verhaltens der Regelstrecke in arbeitspunktabhängigen Kennlinien bzw. Kennfeldern gespeichert und werden im Applikationsprozess eingestellt, um die Regelgüte in allen Arbeitspunkten zu gewährleisten.

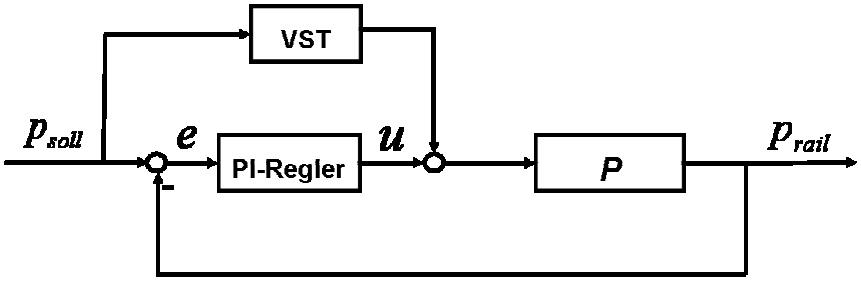


Abbildung 2. Regelungsstruktur des Hochdrucks

3. Applikation für Reglerparameter

3.1. Simulationsgestützte Vorapplikation

Die Grundlagen für eine Modellierung des Hochdruck-Kraftstoffsystems wurden in (Blath 2006) vorgestellt. Dieser Ansatz wird in (Tomforde et al. 2007) unter Berücksichtigung der Temperaturabhängigkeit der Kraftstoffdichte und der Kompression des Kraftstoffes in der Hochdruck-Kraftstoffpumpe erweitert.
Auf Basis der physikalischen Simulationsmodelle erfolgt die Grundbedatung des PI-Reglers in Abhängigkeit von den Arbeitspunkten. Die gewonnenen Parameter aus der Vorapplikation bilden die Startwerte für ein anschließendes Selbsteinstellungsverfahren mit bestimmtem Optimierungskriterium.

3.2. Selbsteinstellungsverfahren auf Basis von Performanzindizes

Das vorgeschlagene Selbsteinstellungsverfahren basiert sowohl auf Performanzindizes der klassischen Regelungstechnik als auch auf Indizes aus CPM-Methoden (Qin 1998, Jelali 2006), die normalerweise zur Performanzüberwachung in der Prozessindustrie eingesetzt werden. Die Vorteile dieses Verfahrens sind, dass keine umfangreichen Vorkenntnisse über die Regelstrecke oder deren Parameter benötigt werden und die Anforderungen an Rechenleistung und Speicherplatzbedarf für die Implementierung im Steuergerät sehr gering sind. In (Visioli 2005) wurde eine Methode mit zwei CPM-Indizes, Flächenindex (Visioli 2005) und Trägheitsindex (Hägglund 1999), zur Selbsteinstellung des PI-Reglers entwickelt. Aufgrund von Rauschen sowie Schwingungen im Kraftstoffsystem des Motors, deren Frequenzen abhängig von der Drehzahl sind, kann der Trägheitsindex hier leider nicht eingesetzt werden, da er sensitiv gegenüber Schwingung und Rauschen ist und trotz einiger Maßnahmen nicht richtig errechnet werden kann, wie in (Yang et al. 2006) gezeigt wurde.

Der PI-Regler in der Abbildung 2 kann mit

$$u(s) = K_p\left(1 + {}^{1}\!/_{T_i s}\right) \cdot e(s)$$

beschrieben werden, wobei *Kp* und *Ti* abhängig vom Arbeitspunkt sind. Die Selbsteinstellung der Reglerparameter wird deswegen separat in allen vordefinierten Arbeitspunkten durchgeführt. Unter der Berücksichtigung der Ungenauigkeit im Simulationsmodell werden die Startwerte von *Kp* und *Ti* sehr konservativ für den „worst case" vorbedatet, d.h. kleines Kp und großes Ti. Da der PI-Regler hauptsächlich für die Regelgüte im Störverhalten und die Stabilisierung der Regelstrecke zuständig ist, wird als Ziel der Selbsteinstellung die Optimierung des Störverhaltens festgelegt. In diesem Beitrag wird speziell die Minimierung der Ausregelzeit bei einer sprungförmigen Störung als Optimierungsziel definiert.

Um gleichzeitig die Robustheit des PI-Reglers nach der Selbsteinstellung zu gewährleisten, muss der eingestellte Regler die folgenden Güteanforderungen erfüllen:

i) keine große Überschwingung in der Stellgröße und

ii) keine Oszillation im Regelkreis wegen des zu aggressiv eingestellten Reglers.

Der Grad der Überschwingung und Oszillation kann durch die relative Überschwingweite und den Flächenindex quantifiziert werden und lässt sich je nach Anwendung festsetzen.

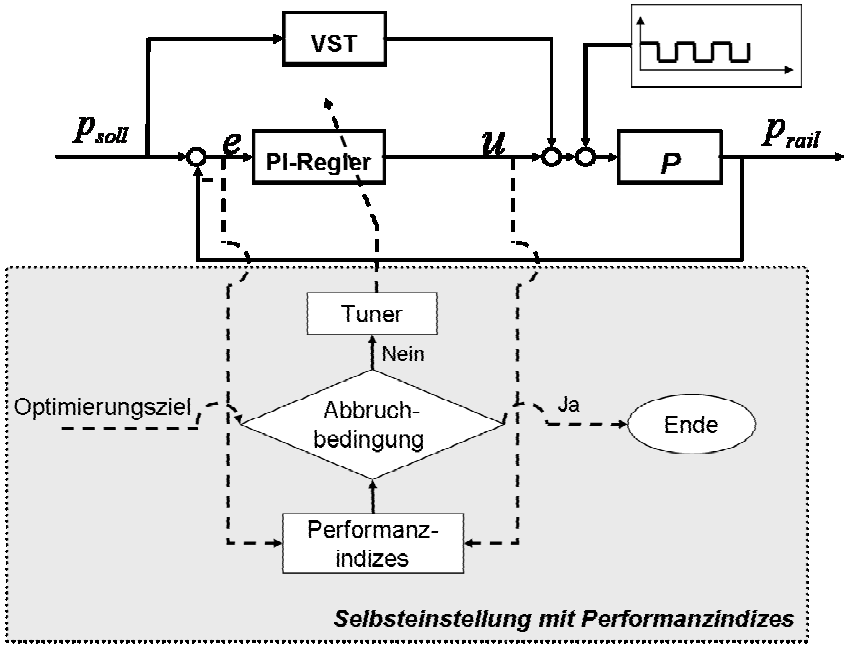


Abbildung 3. Flussdiagramm der Selbsteinstellungsprozedur

Das Flussdiagramm der Selbsteinstellungsprozedur wird in der Abbildung 3 dargestellt. Der Hochdruckregelkreis wird durch eine Reihe von sprungförmigen Störungen angeregt. Nach jeder Störung werden die Performanzindizes, der Flächenindex, die relative Überschwingweite und die Ausregelzeit aus den zeitlichen Signalverläufen der Regeldifferenz ***e*** und des Reglerausgangs ***u*** errechnet und bewertet.

Im Unterschied zum klassischen Reglerentwurf braucht die vorgeschlagene Selbsteinstellungsprozedur keine Identifikation der Regelstreckenparameter, d.h. die errechneten Performanzindizes entscheiden nicht direkt, wie groß *Kp* und *Ti* sein sollen, sondern geben durch den Gütevergleich an, ob die aktuellen *Kp* und *Ti* zu groß oder zu klein eingestellt wurden. Nach dem Update von *Kp* und *Ti* wird die Abbruchbedingung, die im Zusammenhang mit dem Optimierungsziel steht, auf Basis der Performanzindizes geprüft. Wenn die Abbruchbedingung erfüllt wird endet die Selbsteinstellungsprozedur.

Die Selbsteinstellungsprozedur kann in drei Stufen eingeteilt werden:

- In der ersten Stufe wird *Kp* stetig vergrößert, bis die vordefinierten Güteanforderungen (keine Oszillation und kein großes Überschwingen) nicht mehr erfüllt werden.
- In der zweiten Stufe versucht man die Ausregelzeit durch Verkleinerung von *Ti* zu optimieren. Die dadurch verursachte geringe Vergrößerung der Überschwingung lässt sich anschließend durch Reduzierung von *Kp* wieder kompensieren. Nach jeder Einstellung werden ggf. die aktuellen Werte von *Kp* und *Ti* gespeichert.
  Wenn *Ti* zu sehr reduziert wurde und einen Bereich erreicht, in dem die Überschwingung oder Oszillation nicht mehr durch *Kp* kompensiert werden kann ist die Abbruchbedingung erfüllt. Es ist zu beachten, dass eine mögliche Oszillation hier zur Erhöhung der Ausregelzeit führen kann.
- In der dritten Stufe werden die vorher gespeicherten Werte von *Kp* und *Ti*, die der minimalen Ausregelzeit entsprechen, als Endwerte der Selbsteinstellung gespeichert und die Selbsteinstellungsprozedur endet.

Es ist zu beachten, dass die im Verfahren verwendeten Performanzindizes unabhängig von den Amplituden der sprungförmigen Störungen sind. Man kann innerhalb einer Selbsteinstellungsprozedur die Störungen mit unterschiedlichen Amplituden als Anregungen verwenden, wenn keine starke Nichtlinearität um den aktuellen Arbeitspunkt vorhanden ist.

4. Ergebnisse am Prüfstand

Das vorgeschlagene Selbsteinstellungsverfahren wurde in einem Steuergerät implementiert und an einem Prüfstand der IAV GmbH für die Applikation des Kraftstoff-Hochdruck-Reglers eingesetzt. Im Verfahren wird der stationäre bzw. dynamische Zustand des Hochdruckregelkreises bei jeder Anregung online automatisch detektiert. Die entsprechenden Performanzindizes lassen sich, ohne zusätzliche Filterung, aus den rohen Messdaten im dynamischen Zustand berechnen. Die Selbsteinstellung läuft automatisiert und nur mit Kontrolleingriffen des Applikateurs ab. Die Reglerparameter wurden in mehreren Arbeitspunkten mit unterschiedlicher Drehzahl und Last appliziert und die Ergebnisse werden im Folgenden dargestellt.

Das Einstellungsergebnis im Arbeitspunkt von 4000 [1/min] Drehzahl und 20% Last wird in der Abbildung 4 gezeigt. Die Selbsteinstellungsprozedur beginnt bei etwa *30s*. *Kp* erreicht seinen maximalen Wert bei *110s* und wird dann reduziert, damit die Güteanforderung an Überschwingung und Oszillation stets erfüllt wird. Bei etwa *660s* wurde die optimale Ausregelzeit gefunden und eine weitere Reduzierung von *Ti* führt zu keiner weiteren Optimierung der Ausregelzeit. Die Abbruchbedingung wird dann bei etwa *700s* aktiviert und die Prozedur geht zu Ende. *Kp* und *Ti* werden auf die zuvor gespeicherten optimierten Werten von *Kp=0.39* und *Ti=0.13* eingestellt.

In der Abbildung 4 wurden drei Paare von *Kp* und *Ti* bei der Selbsteinstellung markiert. Die Antworten auf eine sprungförmige Störung mit diesen drei Paaren von *Kp* und *Ti* werden in der Abbildung 5 und 6 gezeigt. Daraus ist leicht zu erkennen, dass die durchgezogenen Linien, mit dem eingestellten Wertepaar Nummer 3, die beste Regelgüte in Bezug auf das Optimierungsziel zeigen.

Es ist zu beachten, dass der Zeitaufwand für die Selbsteinstellung der Reglerparameter stark abhängig von den Startwerten von *Kp* und *Ti* ist. Aufgrund der Modellungenauigkeit und aus Sicherheitsgründen wurden hier konservative Startwerte von *Kp* und *Ti* in der

Selbsteinstellung verwendet, wie in der Abbildung 4 und 7 gezeigt. Die Zeit für die Selbsteinstellung kann deutlich reduziert werden, wenn günstigere Startwerte durch die Verbesserung der Vorapplikation geliefert werden. Die Ergebnisse der Selbsteinstellung haben auch gezeigt, dass man, auf Kosten eines erhöhten Zeitaufwands, trotz schlechter Vorapplikation stets die optimalen *Kp* und *Ti* im Sinne des gewählten Gütekriteriums finden kann.

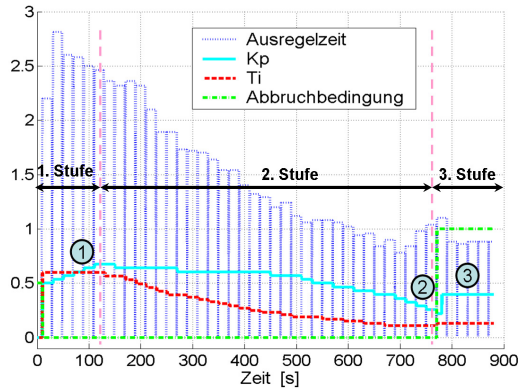


Abbildung 4. Messergebnisse

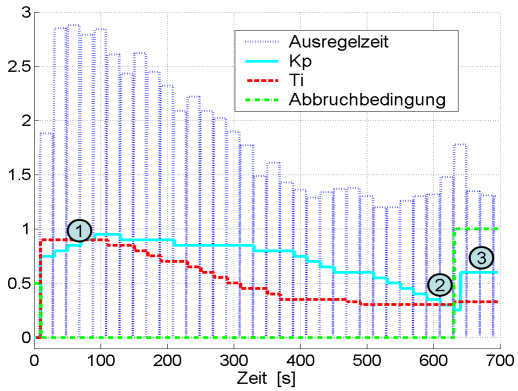


Abbildung 7. Messergebnisse

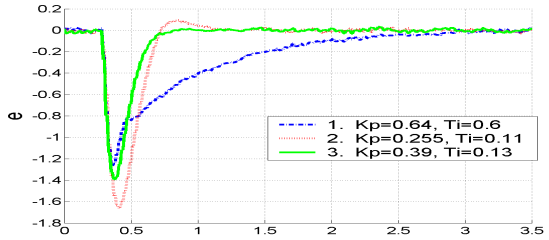


Abbildung 5. Regeldifferenz

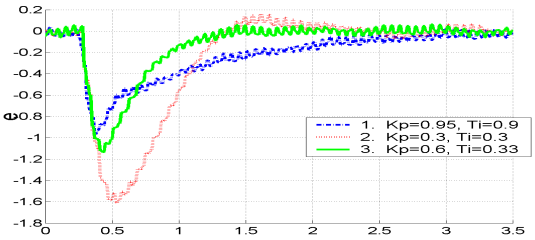


Abbildung 8. Regeldifferenz

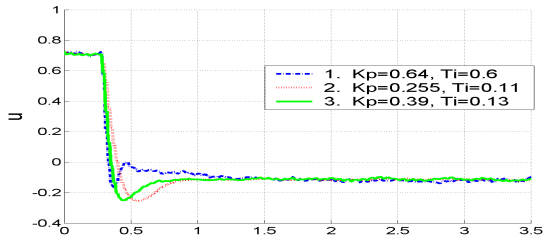


Abbildung 6. Reglerausgang

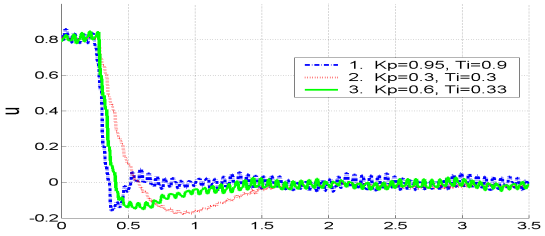


Abbildung 9. Reglerausgang

In der Abbildung 7 werden die gleichen Einstellungen im Arbeitspunkt von 2000 [1/min] Drehzahl und 40% Last dargestellt. Der Gütevergleich des eingestellten *Kp* von *0.6* und *Ti* von *0.33* mit zwei anderen Paaren von *Kp* und *Ti* wird in der Abbildung 8 und 9 gezeigt. Wenn der IAE (Integrated Absolute Error)-Index als Kriterium für die Bewertung der Selbsteinstellungsergebnisse genommen wird, ist die Regeldifferenz des eingestellten Reglers (durchgezogene Linie in Abbildung 8) mit *IAE = 0.7* deutlich kleiner als die der beiden anderen mit *IAE = 0.96* (strichpunktierte Linie) bzw. *IAE = 1.18* (gepunktete Linie).

Wie im Abschnitt 3 erwähnt, wird hier die Minimierung der Ausregelzeit im Störverhalten bei der Selbsteinstellung gegenüber anderen Kriterien bevorzugt. Mit den eingestellten Reglerparametern erhält man die optimierte Ausregelzeit, allerdings auf Kosten von maximaler Regelabweichung, wie in der Abbildung 5 und 8 gezeigt. Durch die entsprechende Anpassung der Abbruchbedingung bzw. die Einführung zusätzlicher Performanzindizes ist es möglich, das vorgestellte Selbsteinstellungsverfahren ohne große Änderung für andere Optimierungsziele, wie zum Beispiel optimaler IAE mit begrenzter maximaler

Regelabweichung, einzusetzen. Es ist zu beachten, dass man bei der Selbsteinstellung der Reglerparameter im Regelkreis ohne Vorsteuerung die Regelgüte sowohl im Führungsverhalten als auch im Störverhalten berücksichtigen muss.

5. Zusammenfassung und Ausblick

In diesem Beitrag wurde ein online performanzindizes-basiertes Selbsteinstellungsverfahren, mit einer simulationsgestützten Vorapplikation, zur Steigerung der Applikationseffizienz und Erlangung reproduzierbarer Applikationsergebnisse vorgestellt. Das vorgestellte Verfahren wurde in einem Steuergerät implementiert und am Beispiel der Applikation der Parameter eines Kraftstoffhochdruckreglers in mehreren Arbeitspunkten am Motorprüfstand validiert. Dieses Verfahren ist ohne wesentliche Veränderungen für die Implementierung auf Regelkreise mit ähnlicher Struktur geeignet.

Wenn die benötigten Regelkreisanregungen im Fahrzeugbetrieb vorhanden sind, ist eine online Selbsteinstellung der Reglerparameter, zum Beispiel für die Adaption der Bauteilalterung, mit dem Ziel eine vordefinierte Regelgüte zu gewährleisten möglich.

Literatur

[1] Blath, J.P. (2006): *Modelling of fuel pressure dynamics.* 3. ASIM Workshop – Modelling, control und simulation in automotive and process automation, Wismar, pp. 67-75.

[2] Hägglund, T. (1999): *Automatic detection of sluggish control loop,* Cont. Eng. Prac. 7, pp. 1505–1511.

[3] Jelali, M. (2006): *Regelkreisberwachung in der Metallindustrie-Teil 1: Klassifikation und Beschreibung der Methoden*, Automatisierungstechnik Vol. 54, No. 1, pp. 36-46

[4] Millich, E., W. Gottschalk, M. Köller, H. Braun and M. Schultalbers (2006): *Calibration of torque structure and charge control for SI engines supported by physical simulation model,* SAE 2006-01-0854.

[5] Qin, S.J. (1998): *Control performance monitoring - a review and assessment, Computers and Chemical Engineering,* vol. 23, pp. 173-186

[6] Tomforde, M., H.P. Dünow, J.P. Blath and T. Jeinsch (2007): *Modellierung eines mengengesteuerten Hochdruck-Kraftstoffsystems für Ottomotoren mit Benzin-Direkteinspritzung,* VDI-Berichte Nr. 1971, pp. 803-810.

[7] Visioli, A. (2005): *Assessment of tuning of PI controller for self-regulating processes,* Proc. IFAC World Congress, 2005, Prague.

[8] Yang, G., N. Weinhold, S.X. Ding, T. Jeinsch und M. Schultalbers (2006): *Application of Control Loop Performance Monitoring Methods to Engine Control Systems,* 3. ASIM Workshop – Modelling, control und simulation in automotive and process automation, Wismar, pp. 29-36.

Simulationsbasierte Entwicklung von Motorsteuerungsfunktionen am Beispiel der Momentenregelung

C. Fritzsche, H.-P. Dünow, Hochschule Wismar - University of Technology, Business and Design
c.fritzsche@stud.hs-wismar.de, p.duenow@et.hs-wismar.de
B. Lampe, Universität Rostock - Fachbereich Elektrotechnik und Informatik
bernhard.lampe@uni-rostock.de

Zusammenfassung

Neben der Realisierung des Antriebsmomentes müssen bei der Steuerung eines Verbrennungsmotors verschiedene andere Regelungsziele, wie die Einhaltung von Abgasgrenzwerten oder die Verbrauchsminimierung, berücksichtigt werden. Um die steigende Anforderungen zu gewährleisten, werden bei der Entwicklung von neuen Steuerungsfunktionen simulationsbasierte Untersuchungen durchgeführt. Diese ermöglichen eine Analyse und eine Abschätzung der Güte der entwickelten Verfahren.

In diesem Beitrag werden ein neuer Steuerungsansatz zur Momentenregelung vorgestellt und zwei Varianten der Umsetzung einer modellprädiktiven Regelung untersucht. Auf der Basis eines Verbrennungsmotormodells wird zwischen einer Realisierung des Momentenreglers mit und ohne Berücksichtigung von Beschränkungen unterschieden.

1 Einleitung

Das Antriebsmoment ist eine der wichtigsten Größen bei der Regelung des Verbrennungsmotors. Durch die große Anzahl an Einflussgrößen, ausgeprägten Nichtlinearitäten oder die Verkopplung von Teilprozessen ist es zunehmend schwieriger, die Anforderung an die Regelgüte mit klassischen Steuerungsstrukturen, wie z.B. Kennfeldreglern, zu erfüllen. In diesem Beitrag wird ein Steuerungsansatz beschrieben, bei dem statt der realen Stellgrößen normierte Ersatzstellgrößen für die Momentenregelung verwendet werden. Diese Ersatzstellgrößen werden in Form von Sollwerten in unterlagerten Regelkreisen realisiert. Ein wesentlicher Vorteil dieses Ansatzes besteht darin, dass die unterlagerten Regelkreise

ausreichend genau durch lineare Modelle beschrieben und für den Steuerungsentwurf genutzt werden können. Dabei müssen allerdings variable Stellbegrenzungen berücksichtigt werden.

An einem komplexen nichtlinearen Verbrennungsmotormodell mit Antriebsstrang wird ein modellprädiktiver Ansatz zur Regelung des Antriebsmomentes untersucht. Um eine Aussage zu treffen, in wie weit sich die Berücksichtigung von Stellgrenzen auf die Regelgüte auswirkt, wird dabei die Regelung zwischen einer Realisierung mit Berücksichtigung und ohne Berücksichtigung von Begrenzungen unterschieden.

In Abschnitt 2 wird der Prozess und die das Antriebsmoment beeinflussenden Größen beschrieben. Daraus wird in Kapitel 3 ein Mehrgrößenmodell als Basis für eine überlagerte Steuerung abgeleitet und der modellprädiktive Ansatz zur Momentenregelung vorgestellt. In Abschnitt 4 wird die Regelung mit und ohne Berücksichtigung von Beschränkungen untersucht.

2 Prozessverhalten

Das von einem Verbrennungsmotor maximal realisierbare Moment hängt maßgeblich von der während der Verbrennung im Zylinder befindlichen Frischluftmenge ab. Klassischerweise wird die Menge der in den Zylinder gesaugten Frischluft durch eine Drosselklappe gesteuert. Bei aufgeladenen Motoren wird der Druck vor der Drosselklappe durch einen Kompressor oder einen Abgasturbolader erhöht, so dass in gleicher Zeit mehr Frischluft in den Zylinder gelangen kann. Durch Einspritzen einer bestimmten Kraftstoffmenge wird eine gewünschte Gemischzusammensetzung eingestellt, die als Lambda bezeichnet wird. Diese beschreibt das Verhältnis von Frischluft- zu Kraftstoffmasse. Über diese Größe kann das Moment im begrenzten Maße beeinflusst werden. Das maximale Moment ergibt sich für Ottomotoren bei einem Lambda von etwa 0.9. Bei Motoren mit Direkteinspritzung ändert sich das Moment unmittelbar mit der Änderung der Kraftstoffmenge, wodurch eine schnelle Momentenerhöhung erreicht werden kann. Entsprechend lässt sich innerhalb eines einzigen Arbeitstaktes eine schnelle Reduzierung des Momentes durch Verringerung der Einspritzmenge realisieren.

Durch variable Steuerung der Ein- und Auslassventile kann erreicht werden, dass neben der Frischluft auch Abgas in den Zylinder gelangt. Eine Anreicherung des Gemisches mit Abgas wirkt sich vorteilhaft auf die Verbrennung und die Schadstoffemissionen aus. Die durch die Ventilüberschneidung gesteuerte Änderung der Abgasmenge im Zylinder führt zu einer Abweichung der Frischluftmasse vom Sollwert. Erlaubt der Stellmechanismus für die Überschneidung der Ein- und Auslassventile eine schnelle Änderung der Abgasmenge, lassen sich über diesen Weg entsprechend schnelle Momenteneingriffe realisieren, da sich der Frischluftanteil im Zylinder unmittelbar mit der Verstellung ändert.

Für den Wirkungsgrad entscheidend ist der Brennbeginn, der beim Ottomotor hauptsächlich durch den Zündwinkel bestimmt wird. Der Zündwinkel, der zum maximalen Moment führt, wird als optimaler Zündwinkel bezeichnet. Dieser ist abhängig von der aktuel-

len Drehzahl und der im Zylinder vorhandene Gemischmenge. Eine Verzögerung des Zündzeitpunktes gegenüber dem Optimalwert führt zu einer Verringerung des Motormomentes und damit zu Wirkungsgradeinbußen. Der Zusammenhang zwischen Zündzeitpunkt und Motormoment ist ähnlich wie der Einfluss von Lambda stark nichtlinear.

Neben der Frischluftmenge, der Abgasmenge, dem Lambda und dem Zündzeitpunkt gibt es weitere Stelleinrichtungen mit denen das Verhalten des Verbrennungsmotors beeinflusst werden kann. Zum Beispiel lässt sich die Gemischqualität durch verschiedene konstruktive Parameter, dem Einspritzzeitpunkt, dem Raildruck oder der Ladungsbewegungsklappe variieren. Die Gemischqualität beeinflusst in erster Linie den Wirkungsgrad des Motors und wird normalerweise nicht für die Steuerung des Motormomentes genutzt. In Abbildung 1 sind die Wirkpfade bei der Energieumsetzung und damit bei der Momentenerzeugung zusammengefasst.

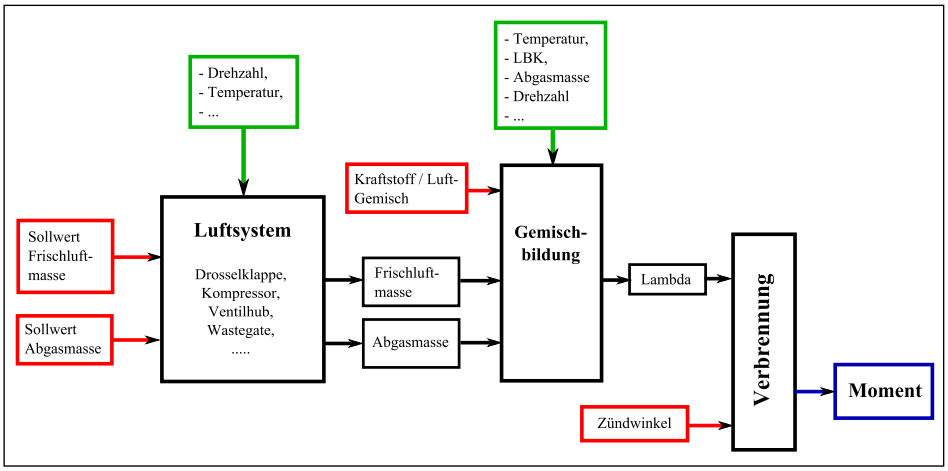


Abbildung 1: Wirkpfade bei der Energieumsetzung

3 Regelung des Motormoments

Da für das Motormoment hauptsächlich die Frischluftmenge im Zylinder, nicht aber der Weg der Realisierung dieser Menge entscheidend ist, ist es sinnvoll, für die Momentenregelung nicht die Steuergrößen für die Frischluftmenge direkt zu verwenden, sondern statt dessen den Sollwert für eine unterlagerte Füllungsregelung als Stellgröße zu nutzen. In der unterlagerten Regelung bzw. Steuerung können einerseits Nichtlinearitäten kompensiert werden, anderseits lassen sich Last- bzw. Betriebspunkt bedingte Einschränkungen des Stellbereiches der direkten Stellgröße für die Füllung einfacher berücksichtigen.

Aus Abb. 1 und den bisherigen Ausführungen lassen sich vier wesentliche Stellgrößen ableiten, die zur Steuerung des Motormomentes vorzugsweise verwendet werden können. Das sind die Frischluftmasse (einstellbar über Drosselklappe, Ventilhub, Ventilüberschneidung, Ladedruck, ...), die Abgasmasse (einstellbar über die Ventilüberschneidung), die Kraftstoffmenge (steuerbar über die Einspritzventile) und der Zündzeitpunkt.

Die Abgasmasse, die Kraftstoffmenge und der Zündzeitpunkt sind durch variable Begrenzungen in ihrem Stellbereich eingeschränkt. Für die Momentenregelung ist es sinnvoll, den Gesamtprozess in mehrere gesteuerte bzw. geregelte Teilsysteme zu zerlegen.

Die unterlagerten Regelkreise lassen sich direkt aus der in Abb. 1 dargestellten Struktur ableiten. Das System mit der langsamsten Dynamik ist die Regelung der Frischluftmasse. Aufgrund der starken Verkopplung zwischen Frischluftmasse und Abgasmasse ist es hier sinnvoll diese beiden Regelgrößen in einem System zu steuern. Diese unterlagerte Steuerung muss eine Reihe verschiedener Einflüsse und Nichlinearitäten berücksichtigen bzw. kompensieren und soll vorrangig dafür sorgen, dass den Sollwerten der Frischluftmasse und der Abgasmasse gefolgt wird und diese möglichst schnell und ohne Überschwingen eingestellt werden. Das zweite unterlagerte System stellt den Kraftstoffpfad dar. Dieses stellt das vorgegeben Kraftstoff-Luft-Gemisch unter anderem über die Einspritzung ein und kann bei Direkteinspritzung eine Führungsänderung innerhalb eines Arbeitstaktes realisieren. Ein letzter unterlagerter Regelkreis wird durch das Zündsystem realisiert.

Da die Abhängigkeit des Moments von den genannten Größen dennoch nichtlinear wäre, was die Auslegung der überlagerten Steuerung entsprechend schwierig gestalten würde, werden diese Sollwerte in daraus resultierende Ersatzgrößen umgewandelt. Diese Größen werden so gewählt, dass zwischen der Änderung einer Ersatzgröße und der resultierenden Änderung des Motormomentes ein linearer Zusammenhang entsteht. Damit ergibt sich die in Abb. 2 dargestellte Struktur. Der von der Momentenregelung zu steuernde Prozess setzt sich aus den unterlagerten Regelkreisen und Steuerungen für die Frischluftmasse, der Abgasmasse, dem Gemischverhältnis und dem Zündwinkel sowie aus einer vorgelagerten Funktion zur Linearisierung und Transformation der Ersatzstellgrößen in Sollwerte für die physikalischen Größen zusammen. Die von der Momentenregelung zu

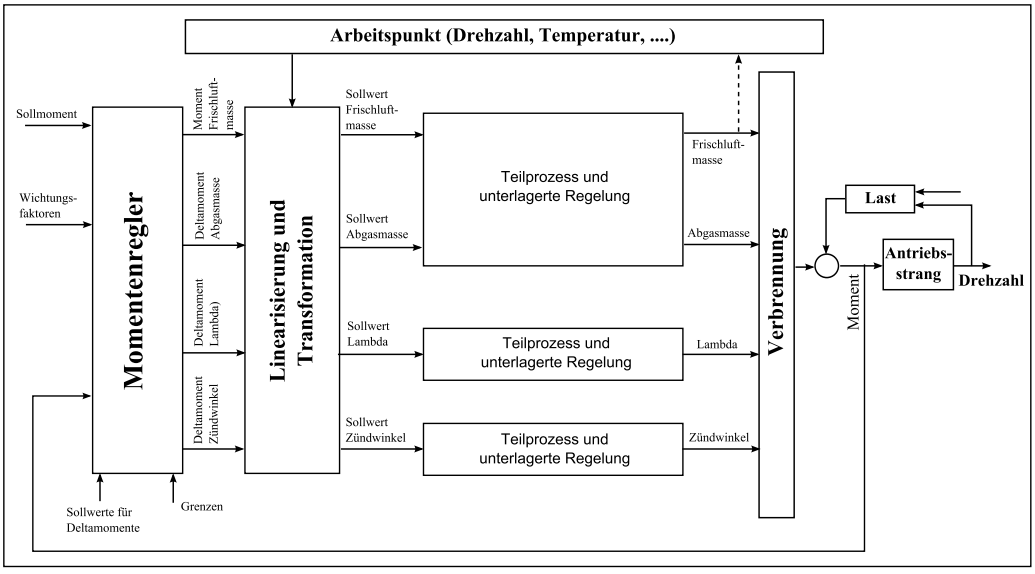


Abbildung 2: Struktur zur Momentenregelung

realisierenden Sollmomentenverläufe werden aus dem Fahrerwunsch oder externen Anforderungen, wie z.B. einem automatischen Schaltgetriebe, generiert. Die Regelung muss in der Lage sein variablen Anforderungen hinsichtlich der Regelungsziele zu genügen. Für die Umsetzung stehen nicht immer alle der oben beschriebenen Stellgrößen zur Verfügung. Die Stellgrößen sind häufig in ihrem Stellbereich stark eingeschränkt oder sollen überhaupt nicht von einem vorgegebenen Sollwert abweichen. Ein Regelungskonzept mit dem nöti-

gen Potential ist die Klasse der Modellprädiktiven Regelung. Bei diesem Verfahren werden sowohl Stellbegrenzungen als auch prädizierbare Soll- und Störgrößenverläufe berücksichtigt.

Der Grundgedanke besteht darin, ein Optimierungsproblem in jedem Abtastschritt zu lösen. Ein geeignetes Gütefunktional ist in Gleichung 1 dargestellt. Die Optimierung wird über einen Prädiktionshorizont durchgeführt, welche es ermöglicht das optimale Eingangssignal bei gleichzeitiger Berücksichtigung von Beschränkungen auszugeben.

$$\begin{aligned} J(k) &= \sum_{i=0}^{H_p} (\hat{y}(k+i|k) - r(k+i))^T Q_i (\hat{y}(k+i|k) - r(k+i)) \\ &+ \sum_{i=0}^{H_u-1} \Delta\hat{u}^T(k+i|k) R_i \Delta\hat{u}(k+i|k). \end{aligned} \tag{1}$$

In Gleichung 2.1 ist $\hat{y}$ die prädizierte Regelgröße und w die Führungsgröße. Die Bestrafungen werden durch Q für die Regeldifferenz und durch R für die Stellgrößenänderungen dargestellt. Hp und Hu sind der Prädiktions- und der Stellhorizont. Für die Implementierung der modellprädiktiven Regelung werden lineare Zustandsraummodelle der Form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2}$$

verwendet. Für detailliertere Beschreibungen der modellprädiktiven Regelungsmethode sei auf Grimble [4] oder Maciejowski [7] verwiesen. Ein Verfahren zur Echtzeitanwendung der modellprädiktiven Regelung an einem Verbrennungsmotor wurde in [1, 5] vorgestellt. Werden die Stellgrößen nur in wenigen Arbeitspunkten durch den Regler an die Begrenzungen geführt, so ist eine Umsetzung der modellprädiktiven Regelung ohne Berücksichtigung der Stellgrenzen denkbar. In diesem Fall vereinfacht sich die Implementierung stark.

Der optimale Verlauf der Stellgrößenänderung $\Delta u(k)$ lässt sich aus der in Gleichung 1 gegebenen Kostenfunktion unter Einbeziehung der Modelle, des Stellhorizontes und des Prädiktionshorizontes ermitteln [1, 2, 7]. Die Lösung für den Fall, dass keine Stellbegrenzungen wirksam sind, lässt sich wie folgt zusammenfassen:

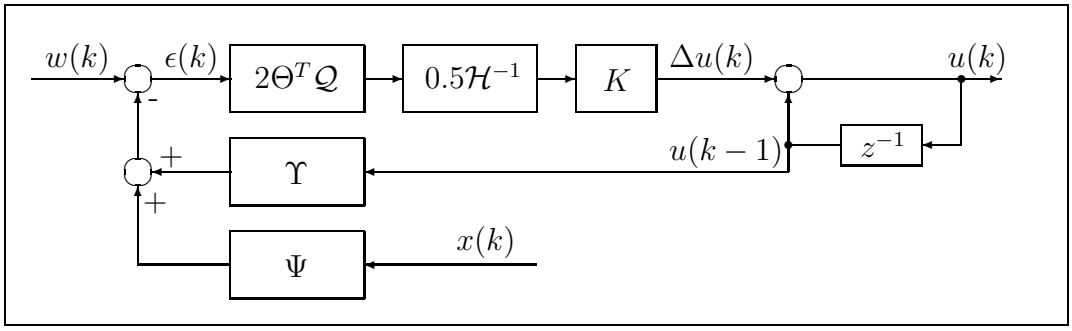


Abbildung 3: Modellprädiktive Regelung ohne Berücksichtung der Grenzen

Der Block K koppelt die aktuellen Stellgrößen aus. Weiterhin sind die Stellgröße $u(k)$,

der Verlauf der Führungsgröße $w(k)$, die Systemzustände $x(k)$, $\mathcal{H} = \Theta^T \mathcal{Q} \Theta + \mathcal{R}$,

$$\Psi = \begin{bmatrix} CA \\ \vdots \\ CA^{H_u} \\ CA^{H_u+1} \\ \vdots \\ CA^{H_p} \end{bmatrix}, \Upsilon = \begin{bmatrix} CB \\ \vdots \\ C\sum_{i=0}^{H_u-1} A^i B \\ C\sum_{i=0}^{H_u} A^i B \\ \vdots \\ C\sum_{i=0}^{H_p-1} A^i B \end{bmatrix} und\ \Theta = \begin{bmatrix} CB & ... & 0 \\ C(AB+B) & ... & 0 \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_u-1} A^i B & ... & CB \\ C\sum_{i=0}^{H_u} A^i B & ... & C(AB+B) \\ \vdots & \vdots & \vdots \\ C\sum_{i=0}^{H_p-1} A^i B & ... & C\sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix}. \quad (3)$$

4 Simulationsbasierte Untersuchungen

Die in Abschnitt 3 entwickelten Verfahren zur Regelung eines Verbrennungsmotors werden im folgenden Abschnitt simulativ untersucht. Dazu wird der modellprädiktive Regler zur Regelung des Antriebsmomentes an einem komplexen Verbrennungsmotormodell eingesetzt, wobei zwischen einer Realisierung mit Berücksichtigung und einer Realisierung ohne Berücksichtigung der Begrenzungen unterschieden wird. Das Verbrennungsmotormodell inklusive Antriebsstrang besteht aus einer Kombination von physikalischen Modellen und Steuergerätefunktionen und ist vereinfacht in Abb. 4 dargestellt.

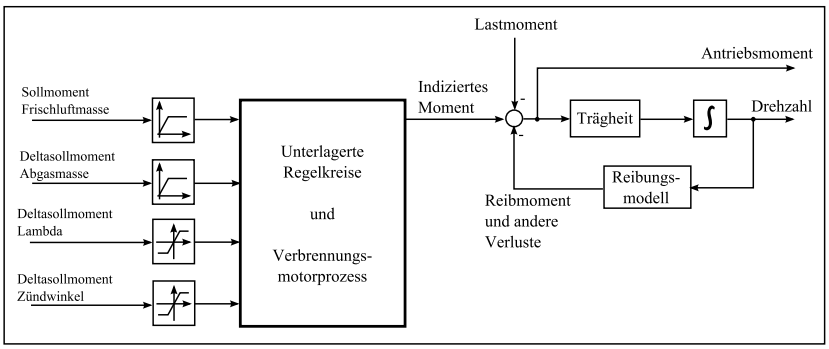


Abbildung 4: Simulationsmodell (vereinfacht)

Für die Simulation wurde ein Prädiktionshorizont Hp von 40 und ein Stellhorizont Hu von 5 Abtastschritten gewählt, wobei die Abtastzeit 10 Millisekunden beträgt. Die für den modellprädiktiven Regler notwendigen Prozessmodelle wurden durch Identifikation mit Hilfe von Eingangssprüngen erstellt.

Wie oben erwähnt ist bei einer Realisierung eines modellprädiktiven Reglers mit Berücksichtigung der Begrenzungen eine Optimierung in jedem Abtastschritt notwendig. Die Umsetzung eines Ansatzes zur Implementierung eines solchen Reglers mit Berücksichtigung der Grenzen auf einer Echtzeitplattform wurde in [1, 5, 6] vorgestellt und soll daher hier nicht näher betrachtet werden. Die Implementierung eines modellprädiktiven Reglers ohne Berücksichtigung sei jedoch in Abb. 5 dargestellt. Der Regler besteht lediglich aus einem Zustandsraummodell (blau), drei Verstärkungen (grün) und einer Begrenzung der Stellgrößen (gelb). Die Verstärkungen zur Realisierung des modellprädiktiven Reglers

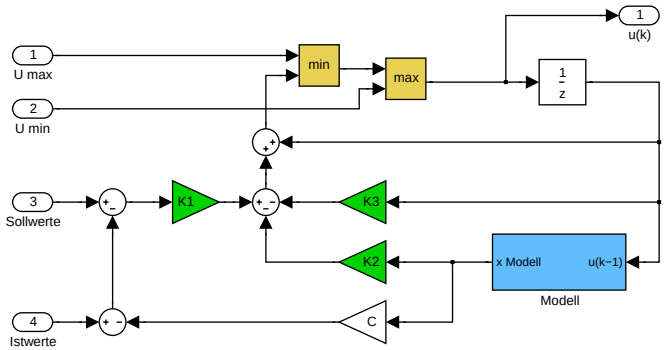


Abbildung 5: Umsetzung MPC ohne Berücksichtigung der Stellgrenzen

setzen sich wie folgt zusammen:

$$K_1 = 0.5 \cdot \mathcal{H}^{-1} \cdot 2 \cdot \Theta^T \cdot \mathcal{Q} \tag{4}$$

$$K_2 = 0.5 \cdot \mathcal{H}^{-1} \cdot 2 \cdot \Theta^T \cdot \mathcal{Q} \cdot \Psi \tag{5}$$

$$K_3 = 0.5 \cdot \mathcal{H}^{-1} \cdot 2 \cdot \Theta^T \cdot \mathcal{Q} \cdot \Upsilon \tag{6}$$

In Abb. 6 ist das Ergebnis des modellprädiktiven Reglers ohne Berücksichtigung der Stellgrenzen dargestellt. Darin sind in der oberen Graphik das Antriebsmoment und in den unteren drei Graphiken die Stellgrößen des Reglers abgebildet. Die Abgasrückführrate wurde hier zur Übersichtlichkeit als Stellgröße vernachlässigt. Wie in Abschnitt 3 dargelegt, sind aus Sicht des MP-Reglers die unteren beiden Stellgrößen (Deltamoment Zündwinkel und Deltamoment Lambda) gleichzeitig Regelgrößen. Die oberen Grenzen sind jeweils in grün, die unteren Grenzen in schwarz gekennzeichnet.

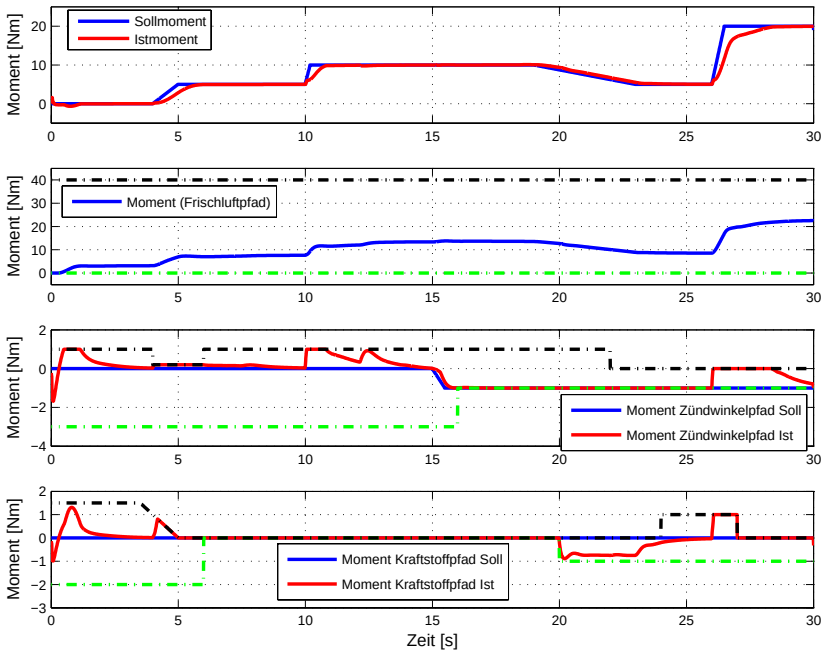


Abbildung 6: Ergebnis der Regelung ohne Berücksichtigung der Stellgrenzen

Die Deltamomente über den Zündwinkel- und Kraftstoffpfad werden nur in dynamischen Bereichen zur Regelung des Antriebsmomentes genutzt. In stationären Bereichen

werden diese Stellgrößen auf den vorgegebenen Sollwert zurückgeführt. Diese Stellgrößen werden lediglich hart abgeschnitten und befinden sich in dynamischen Bereichen häufig in den Begrenzungen. Im ersten Abschnitt, bis etwa 5 Sekunden, konnte der Regler sowohl den Zündwinkelpfad als auch den Kraftstoffpfad in einem gewissen Schlauch nutzen. Ab etwa 6 Sekunden wird der Schlauch sowohl für den Zündwinkelpfad als auch für den Kraftstoffpfad variiert. Dem Regler stehen so unterschiedliche Freiheitsgrade in den Stellgrößen zur Verfügung. Teilweise werden die Grenzen zu Null geführt, so dass die Stellgröße in diesem Bereich auf einem festen Wert gesetzt wird und somit nicht für die Regelung zur Verfügung steht.

Betrachtet man in Abb. 7 das Ergebnis der Regelung mit Berücksichtigung der Stellgrenzen wird deutlich sichtbar, dass bei gleichen Wichtungsverhältnissen die Stellgrößen hier wesentlich weniger in ihren Begrenzungen hängen. Die Stellgrößen werden jedoch wesentlich dynamischer zur Regelung des Antriebsmomentes genutzt. Die Regelgüte ist sowohl bei der Momentenregelung als auch bei der Einhaltung der Sollwerte für die Deltamomentenpfade (Zündwinkelpfad, Kraftstoffpfad) besser als bei einer Realisierung ohne Berücksichtigung der Begrenzungen. Der aufsummierte absolute Fehler für den betrachteten Beispielverlauf ist zum Vergleich in Tabelle 1 zusammengefasst.

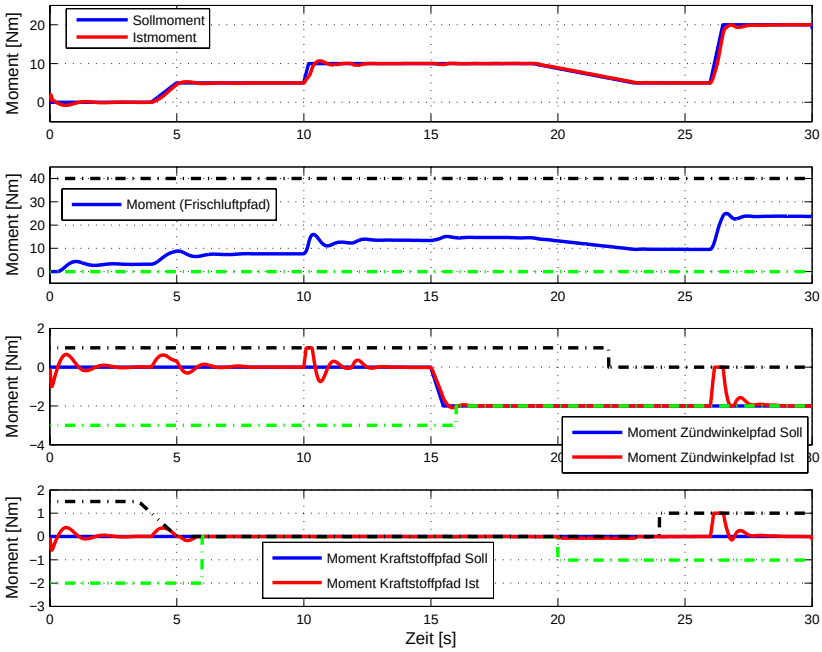


Abbildung 7: Ergebnis der Regelung mit Berücksichtigung der Stellgrenzen

Sowohl durch den Regler mit als auch ohne Berücksichtigung der Begrenzungen werden momentenneutrale Eingriffe realisierbar. Das Deltamoment über den Zündwinkelpfad wird bei etwa 15 Sekunden auf einen neuen Sollwert geführt, ohne eine Änderung im Istmoment zu bewirken.

Eine Berücksichtigung von zukünftigen Sollgrößenänderungen oder Störungen ist sowohl durch den Regler mit Kenntnis der Einschränkungen [2, 3] als auch ohne Kenntnis der Begrenzungen möglich. In Abb. 8 ist das Ergebnis der Regelung ohne Berücksichtigung von Stellgrenzen dargestellt, jedoch sind die Führungsgrößen- und Laständerungen

Regelgröße	Regelung ohne Kenntnis der Stellgrenzen	Regelung mit Kenntnis der Stellgrenzen
Antriebsmoment	13.7 Nm	6.5 Nm
Deltamoment Zündwinkel	8.5 Nm	4.5 Nm
Deltamoment Kraftstoff	5.3 Nm	3.0 Nm

Tabelle 1: Aufsummierter absoluter Fehler

vorzeitig bekannt. Noch bevor der Führungssprung bei etwa 10 Sekunden auftritt, wird durch den Regler automatisch eine Reserve über den Zündwinkel aufgebaut, indem das Moment über den Frischluftpfad erhöht und über den Zündwinkelpfad verringert wird. Zum Zeitpunkt des Führungssprung wird die realisierte Reserve genutzt. Auch ein prädizierter Lastsprung (bei 7 Sekunden) wird durch den automatischen Aufbau einer Reserve schnell ausgeregelt. Eine negative Führungsänderung bei etwa 15 Sekunden ist dem MPC vorzeitig bekannt gemacht worden. Hier wird nun das Moment über die Frischluftmasse vorzeitig verringert und das Deltamoment über den Zündwinkel erhöht. Da dieses Deltamoment jedoch begrenzt ist (wird nicht berücksichtigt), eilt die Regelgröße der Führungsgröße voraus und es kommt in diesem Bereich zu einer schlechten Regelgüte. Dieser Fehler ist bei einer Realisierung mit Berücksichtigung der Grenzen nicht zu beobachten.

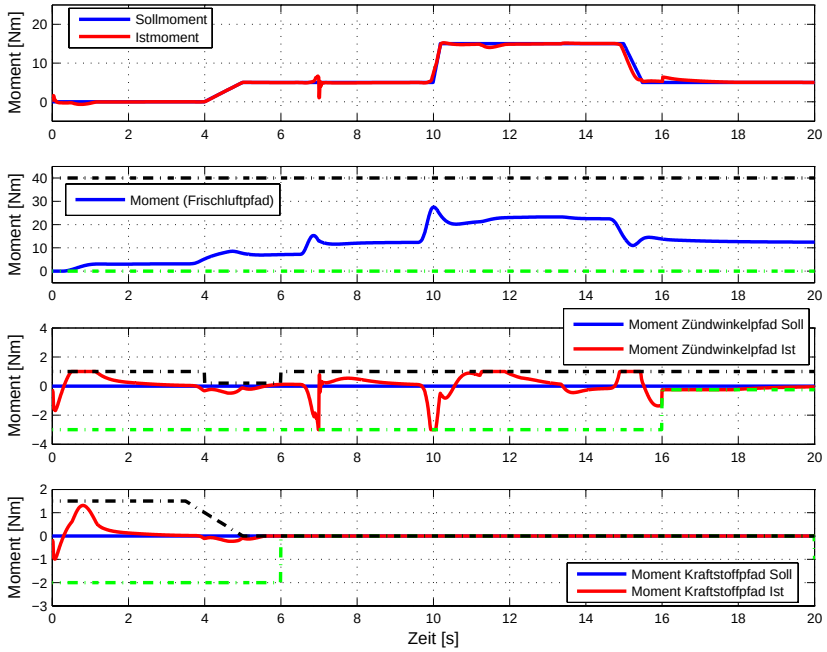


Abbildung 8: Regelung ohne Stellgrenzenberücksichtigung und mit prädizierten Verläufen

Aufgrund der erzielten Ergebnisse kann geschlussfolgert werden, dass bei der Realisierung eines Momentenreglers die modellprädiktive Regelung mit Berücksichtigung der Stellgrenzen besser geeignet ist, als eine Implementierung ohne Berücksichtigung der Beschränkungen. Der Vorteil ist vor allem dann groß, wenn die Stellgrößen häufig in den Begrenzungen hängen. Da jedoch die Implementierung eines solchen Reglers mit Stellgrenzenberücksichtigung wesentlich aufwändiger und die Berechnung der Stellgrößen zeitaufwändiger ist, kann für erste Untersuchungen auch ein Regler ohne Stellgrenzenberücksichtigung genutzt werden. Je kleiner der Stellhorizont Hu gewählt wird, desto größer

wird in beiden Varianten der Regelfehler. Die Regelgüte nimmt jedoch bei Nichtberücksichtigung der Stellgrenzen wesentliche schneller ab als bei einer Berücksichtigung der Begrenzungen im Regelungsalgorithmus.

5 Zusammenfassung und Ausblick

Im Beitrag wurde gezeigt, wie der Entwicklungsprozess von Motorsteuerungsfunktionen anhand von komplexen Modellen und simulationsbasierten Untersuchungen vereinfacht werden kann. Es wurde eine neuartige Regelungsstruktur für die Steuerung des Motormoments vorgestellt, die darauf basiert, dass das Moment nicht direkt durch die verfügbaren Stellgrößen, sondern durch mehrere Stufen in einer Kaskade eingestellt wird und verfügbare Zwischengrößen in unterlagerten Regelungen und Steuerungen verwendet werden.

Für den Regler wurde ein modellprädiktiver Ansatz gewählt, wobei zwischen einer Realisierung mit und ohne Berücksichtigung von Beschränkungen unterschieden wurde. Eine Implementierung mit Berücksichtigung von Stellgrenzen ist sehr aufwendig und ist während der Ausführung sehr rechenintensiv. Sofern die Stellgrößen häufig in die Grenzbereiche gelangen, ist diese Realisierung jedoch zu bevorzugen, da im Gegensatz zu einer Realisierung ohne Berücksichtigung der Grenzen eine größere Regelgüte erreicht wird. Die Annahme, dass bei geringem Stellhorizont die Berücksichtigung der Stellgrenzen im Regelungsalgorithmus weniger ins Gewicht fällt, konnte nicht bestätigt werden. Die Regelgüte nimmt bei der Variante ohne Berücksichtigung der Stellgrenzen mit fallenden Stellhorizont schneller ab als bei einer Berücksichtigung der Begrenzungen.

Literatur

[1] Dünow, H.P., Lekhadia, K.N., Köller, M., Jeinsch, T.: *Model based predictive control of spark ignition engine process*, Proceedings of MMAR, 2005.

[2] Fritzsche, C., Dünow, H.P.: *Advanced Torque Control*, Automation and Robotics, I-Tech Education and Publishing, Vienna, Austria, 2008.

[3] Fritzsche, C., Dünow, H.P., Lampe, B., Schultablers, M.: *Torque Coordination of Spark Ignition Engines based on Predictive Control*, Proceedings of MMAR, 2007.

[4] Grimble, Michael J.: *Industrial Control Systems Design*, John Willey & Sons, LTD Chichester, New York, Weinheim, Brisbane, Singapore, Toronto, 2001.

[5] Lekhadia, K.: *Development of qp-algorithms for realtime prdictive control systems*, Master Thesis, Fachhochschule Darmstadt, 2004.

[6] Lekhadia, K.: *Implementation of MPC Algorithm in TC1796*, Forschungsbericht - Hochschule Wismar, 2006.

[7] Maciejowski, J.M.: *Predictive Control with constraints*, Prentice Hall, 2002.

Modellierung thermischer Systeme für die Hardware-in-the-Loop Simulation am Beispiel eines Fahrzeugkabinenmodells

Dipl.-Ing. Carsten Gehsat, TU Dortmund
carsten.gehsat@tu-dortmund.de
Prof. Dr.-Ing. Prof. h.c. Torsten Bertram, TU Dortmund
torsten.bertram@tu-dortmund.de
Dr.-Ing. Ralph Trapp, Behr-Hella Thermocontrol GmbH
ralph.trapp@bhtc.com

Zusammenfassung

In den letzten Jahren hat sich die standardmäßige Ausstattung von Kraftfahrzeugen mit Klimaanlagen immer mehr durchgesetzt. Die Entwicklung von Steuergeräten für die Fahrzeugklimatisierung erfolgt daher in immer kürzeren Entwicklungszeiten. Bereits Fahrzeuge der Mittelklasse verfügen über komplexe Klimasysteme, die geeignete Entwicklungsumgebungen benötigen, um einen schnellen sowie kostengünstigen Systementwurf zu realisieren. Für die Entwicklung, Verifikation und Validierung wird heute eine Entwicklungsumgebung auf Basis der Hardware-in-the-Loop Simulation (HiL Simulation) eingesetzt. Dieses Vorgehen setzt eine vollständige mathematische Beschreibung des gesamten Klimasystems als Simulationsmodell voraus. Anhand eines Fahrzeugkabinenmodells, das die mittlere Kabinentemperatur unter Berücksichtigung thermischer Einflüsse berechnet, wird die Entwicklung von thermischen Simulationsmodellen für die HiL Simulation dargestellt.

1 Einleitung

Die Fahrzeugklimaanlage hat sich in den letzten Jahren zu einem komplexen Klimasystem mit vielen Sensoren, Aktoren und Regelkreisen weiterentwickelt. Das Klimasteuergerät verarbeitet sämtliche Stell-, Regel- und Störgrößen und ist sowohl für die Informationsverarbeitung als auch die Informationsgenerierung zuständig. Bei der Regelung werden verschiedene Einflussgrößen berücksichtigt, die mithilfe von Sensoren erfasst werden. Zu den wichtigsten Größen zählen die Außen- und Innentemperatur, die Außen-

und Innenluftfeuchtigkeit, die Fahrgeschwindigkeit, die Sonnenintensität und der Sonneneinstrahlwinkel. Zudem ermöglichen moderne zweizonige Klimaanlagen eine getrennte Einstellung der Temperatur- und Luftverteilung für den Fahrer und Beifahrer. Durch geeignete Steuerungs- und Regelungsstrategien versucht das Klimasteuergerät eine optimale Temperatur- und Luftverteilung einzustellen. Die komplexen Klimasysteme lassen sich nur bedingt mithilfe von realen Komponenten oder im Prototypenfahrzeug testen. Spezielle Testbedingungen können schwer oder nur mit erhöhtem Aufwand realisiert werden. Zudem beginnt die Entwicklung der Algorithmen für die Innenraumklimatisierung und der Klimasteuergeräte meist im frühen Produktentstehungsprozess des gesamten Fahrzeugs. Reale Komponenten oder sogar Prototypen sind in dieser führen Entwicklungsphase meist nicht verfügbar. Eine Alternative zu den bisherigen Entwicklungswerkzeugen bietet die HiL Simulation, mit der eine fahrzeugunabhängige Entwicklungs- und Prüfumgebung geschaffen worden ist [1][2].

2 HiL Simulationsumgebung

2.1 HiL Simulator

Für eine HiL Simulation wird ein modular aufgebauter HiL-Simulator eingesetzt. Auf diesem wird die Simulation eines klimatechnischen Fahrzeugmodells, das das gesamte Klimasystem des realen Fahrzeugs als Modell abbildet, in Echtzeit ausgeführt. Über eine universelle Schnittstelle (Nullkraftstecker) auf der Rückseite des HiL-Simulators kann das zu prüfende Steuergerät über einen Kabelbaum angeschlossen und dann wie im realen Fahrzeug bedient werden, Abbildung 1. Durch diese universelle Schnittstelle ist der Simulator unabhängig vom angeschlossenen Steuergerätetyp verwendbar. Die Bedienung des Simulators erfolgt daher über einen Host-Rechner. An diesem lassen sich alle Systemgrößen mithilfe einer grafischen Oberfläche darstellen, wodurch sich vielfältige Möglichkeiten zum Test des Steuergerätes ergeben. Beispielsweise können Fehlerfälle systematisch in die Umgebung eingebracht und das Verhalten des Steuergerätes geprüft werden.

Abbildung 1: HiL Simulator mit angeschlossenem Steuergerät

2.2 Klimatechnisches Fahrzeugmodell

Das klimatechnische Fahrzeugmodell bildet alle Komponenten des realen Fahrzeugs als echtzeitfähiges mathematisches Modell ab. Darunter befinden sich Modelle für die Aktoren der Luftverteilungklappen, sämtliche Sensoren sowie die Buskommunikation zwischen dem Klimasteuergerät und dem Fahrzeug. Für den modellbasierten Entwicklungsprozess für Klimasteuergeräte wird als Entwicklungswerkzeug für die Modelle die Software „MATLAB/Simulink“ eingesetzt. Eine besondere Herausforderung stellt die Modellierung der thermodynamischen Wärmeübergänge und strömungstechnischen Prozesse dar. Dies betrifft die Modelle für den Kältekreis, den Heizkörper, den Luftkanal, das Mischmodul sowie das Kabinenmodell. Um die Echtzeitfähigkeit der Modelle zu garantieren, werden speziell adaptierte Berechnungsmodelle gewählt. Abbildung 2 zeigt eine schematische Darstellung der einzelnen Teilmodelle innerhalb des virtuellen klimatechnischen Fahrzeugmodells. Heizkörper, Kältekreis, Luftkanalmodell, Mischmodul, Klappenmotoren, Gebläse und Sensoren bilden zusammen das HVAC-Modell (Heating, Ventilating and Air Conditioning) [2]. Die Modelle sind alle parametrisiert und ermöglichen eine flexible und zeiteffiziente Modellbildung von neuen Komponenten sowie deren Anpassung und modulorientierte Zusammenstellung zu Fahrzeugmodellen für die Klimatisierung.

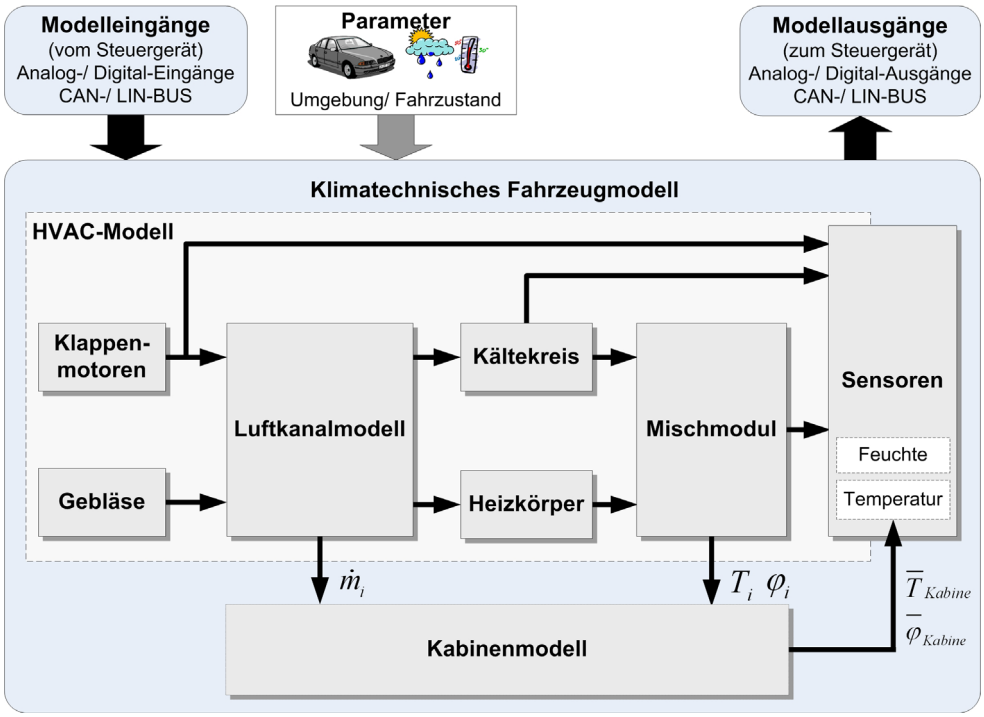


Abbildung 2: Klimatechnisches Fahrzeugmodell

3 Kabinenmodell

Das thermodynamische Fahrzeugkabinenmodell erweitert das klimatechnische Fahrzeugmodell und ermöglicht eine vollständige Simulation des gesamten realen Klimasystems [3].

Ziel des Kabinenmodells ist die Berechnung der mittleren Temperatur und mittleren Luftfeuchte innerhalb der Fahrzeugkabine unter Berücksichtigung der wichtigsten Temperatur- und Feuchteeinflüsse. Dazu zählen die Außentemperatur, die Sonneneinstrahlung, die Fahrgeschwindigkeit, der Temperatur- und der Feuchteeintrag der Fahrzeuginsassen sowie der Energieeintrag durch die Klimaanlage. Für die vollständige HiL Simulation wurden bisher die Werte für die Kabinentemperatur und die Kabinenfeuchte manuell vorgegeben und blieben während der Simulation der Klimaanalage daher konstant. Zukünftig berechnet das Kabinenmodell diese Werte und stellt sie dem Temperatur- und Feuchtesensormodell im klimatechnischen Fahrzeugmodell zur Verfügung. An das Kabinenmodell werden in Bezug auf die Modellierung besondere Anforderungen gestellt. Der Einsatzzweck fordert eine echtzeitfähige Berechnung sowie eine einfache Parametrisierung des Modells. Begründet ist letzteres in einer einfachen und schnellen Anpassung des Modells an neue Fahrzeugkabinen. Die Entwicklung des Kabinenmodells beschränkt sich zunächst auf eine einzonige Modellierung. Das bedeutet, dass ein gemittelter Wert über den Fuß- und Kopfraum für die Lufttemperatur und die Luftfeuchtigkeit in der Fahrzeugkabine berechnet wird. In der realen Fahrzeugkabine ist allerdings eine gewollte Temperaturschichtung vorzufinden. Da die berechnete Kabinentemperatur im klimatechnischen Fahrzeugmodell dem Temperatursensormodell als Eingangsgröße dienen soll und der reale Temperatursensor im Fahrzeug diese Schichtung auch nicht berücksichtigen kann, ist die Betrachtung einer mittleren Temperatur zulässig. Die Fahrzeugkabine wird für eine physikalische Mo-

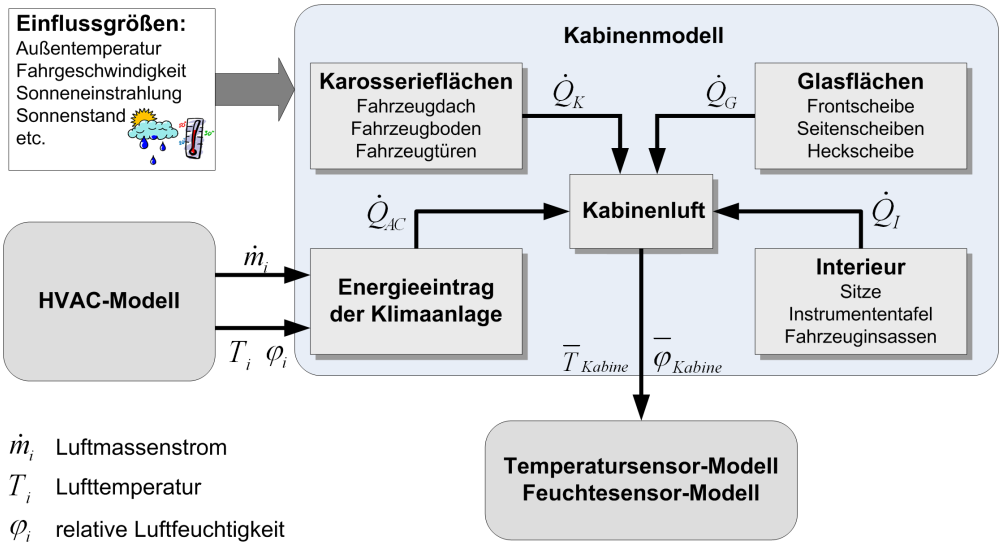


Abbildung 3: Struktur des Kabinenmodells

dellierung in einzelne Elemente unterteilt. Unterschieden wird dabei in Karosserie- und Glasflächen sowie den Einbauten in der Kabine. Zu den modellierten Einbauten zählen beispielsweise die Instrumententafel und die Sitze. Die physikalische und diskrete Modellierung erlaubt eine Parametrisierung der einzelnen Teilmodelle und ermöglicht zugleich eine einfache Anpassung. Die Beschreibung der Wärmeübertragung an den einzelnen Elementen erfolgt unter Berücksichtigung der Wärmeübertragungsmechanismen wie Konvek-

tion, Wärmeleitung und Strahlung. Die Abbildung 3 zeigt den vereinfachten strukturellen Aufbau eines Fahrzeugkabinenmodells. Im Folgenden sollen einige Modellierungsmöglichkeiten verschiedener Kabinenelemente näher vorgestellt werden. Die über die Karosserie und Klimaanlage zu- und abgeführten Wärmeströme erwärmen oder kühlen die Luft in der Fahrzeugkabine. Die Temperaturänderung in der Kabine kann mithilfe der allgemeinen Energiebilanz, die sich nach Gleichung (1) ergibt, berechnet werden.

$$\frac{\mathrm{d}h_{1+X}}{\mathrm{d}t} m_{L,Kabine} + \frac{\mathrm{d}m_{L,Kabine}}{\mathrm{d}t} h_{1+X} = \Sigma\dot{Q} \tag{1}$$

Dabei ist $m_{L,Kabine}$ die Masse und h_{1+X} die spezifische Enthalpie der Kabinenluft. Die Summe $\Sigma\dot{Q}$ der vorzeichenbehafteten Wärmeströme gibt dabei die zu- und abgeführten Wärmeströme an. Im Beispiel nach Abbildung 3 ergibt sich somit für $\Sigma\dot{Q}$ die Gleichung (2) mit den Wärmeströmen $\dot{Q}_K$, $\dot{Q}_G$ und $\dot{Q}_I$ die gemäß den jeweiligen Berechnungsmodellen bestimmt werden.

$$\Sigma\dot{Q} = \dot{Q}_{AC} + \dot{Q}_K + \dot{Q}_G + \dot{Q}_I \tag{2}$$

Der durch die Klimaanlage zugeführte Wärmestrom $\dot{Q}_{AC}$ ergibt sich aus dem Luftmassenstrom $\dot{M}_{zu}$, der Wasserbeladung X und der spezifischen Enthalpie h_{1+X} der zugeführten und der bereits in der Kabine vorhandenen Luft, Gleichung (3).

$$\dot{Q}_{AC} = \frac{\mathrm{d}h_{1+X,zu}\dot{M}_{zu}}{1+X_{zu}} - \frac{\mathrm{d}h_{1+X,Kabine}\dot{M}_{zu}}{1+X_{Kabine}} \tag{3}$$

Aufwendig gestaltet sich die Modellierung der Glasflächen. Neben den konvektiven und wärmeleitenden Eigenschaften ist eine Berücksichtigung der Sonneneinstrahlung auf diese Flächen notwendig. Dabei fließen das verwendete Glasmaterial, die Geometrie der Glasflächen, die Sonnenintensität sowie der Winkel der auftreffenden Sonneneinstrahlung mit in die Berechnung ein. Der auftreffende Sonnenwinkel lässt sich aus dem Azimut- und Elevationswinkel der Sonne und dem Neigungswinkel sowie die Orientierung der Glasfläche im Raum berechnen. Mithilfe des berechneten Winkels lassen sich die reflektierenden, absorbierenden und transmittierenden Anteile der auftreffenden Sonneneinstrahlung und somit der Energieeintrag der Sonne in die Fahrzeugkabine berechnen. Für eine mehrschichtige Glasfläche nach Abbildung 4 können die Bilanzen nach Gleichung (4) bis Gleichung (6) aufgestellt werden.

$$\dot{m}_1 c_{p1} \frac{\mathrm{d}T_1}{\mathrm{d}t} = \dot{Q}_{konv,a} - \dot{Q}_{lw_{ab,a}} + \dot{Q}_{lw_{zu,a}} + \dot{Q}_{Sonne} + \frac{3A\lambda}{d}(T_2 - T_1) \tag{4}$$

$$\dot{m}_2 c_{p2} \frac{\mathrm{d}T_2}{\mathrm{d}t} = \frac{3A\lambda}{d}(T_1 - 2T_2 + T_3) \tag{5}$$

$$\dot{m}_3 c_{p3} \frac{\mathrm{d}T_3}{\mathrm{d}t} = \dot{Q}_{lw_{zu,i}} - \dot{Q}_{lw_{ab,i}} - \dot{Q}_{konv,i} + \frac{3A\lambda}{d}(T_2 - T_3) \tag{6}$$

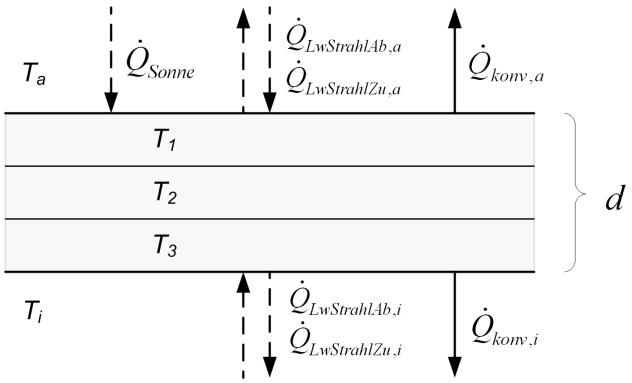


Abbildung 4: Wärmeübertragung an einer mehrschichtigen Glasfläche

Für die nach außen weisende Schicht berechnet sich die mittlere Temperatur nach der Bilanzgleichung (4). Berücksichtigt wird hier die Wärmeübertragung durch langwelligen Strahlungsaustausch mit der Umgebung, Konvektion mit der Außenluft sowie die Wärmeleitung zwischen der äußeren und mittleren Schicht. Der Energieeintrag durch die auftreffende Sonnenstrahlung wird durch den Wärmestrom $\dot{Q}_{Sonne}$ berücksichtigt. Die Bilanzgleichung (5) der mittleren Schicht berücksichtigt die Wärmeleitung zwischen den einzelnen Schichten mit der Wärmeleitfähigkeit λ und der Querschnittsfläche A. Bis auf den zugeführten Energieeintrag der Sonne ergibt sich anlog zur äußeren Schicht die Bilanzgleichung für die nach innen weisende Schicht nach Gleichung (6).

4 Verifikation des Kabinenmodells

Für die Verifikation des Kabinenmodells ist ein Versuchfahrzeug mit Messtechnik ausgestattet worden. Dabei ist Sensorik zur Temperatur- und Luftfeuchtigkeitsmessung speziell im Kopf- und Fußraum der Fahrzeugkabine verbaut worden. Die weiteren Eingangsgrößen, die das Kabinenmodell benötigt, werden entweder über die serienmäßig oder die zusätzlich im Fahrzeug verbaute Sensorik erfasst. Die meisten Größen werden auf dem CAN-Bus des Fahrzeugs bereitgestellt und können aufgezeichnet werden. Zu diesen Größen zählen beispielsweise die Außentemperatur, die Fahrgeschwindigkeit sowie die Temperatur und der Luftmassenstrom der von der Klimaanlage in die Kabine eintretenden Luft. Zur Verifikation sind Messfahrten unter verschiedenen Temperaturbedingungen durchgeführt worden. Dabei werden die Temperaturen im Kopf- und Fußraum sowie die durch die Sensorik erfassten Umgebungszustände wie beispielsweise Außentemperatur, Fahrgeschwindigkeit und Sonnenintensität aufgezeichnet. Mit diesen Daten wird eine Simulation des Kabinenmodells durchgeführt und die gemessene Temperatur und Luftfeuchtigkeit mit den simulierten Werten verglichen. Dazu wird aus den Messwerten der Sensoren aus Kopf- und Fußraum das arithmetische Mittel gebildet. Abbildung 5 zeigt die Langzeitmessung einer Autobahnfahrt bei wechselnden Umgebungsbedingungen.

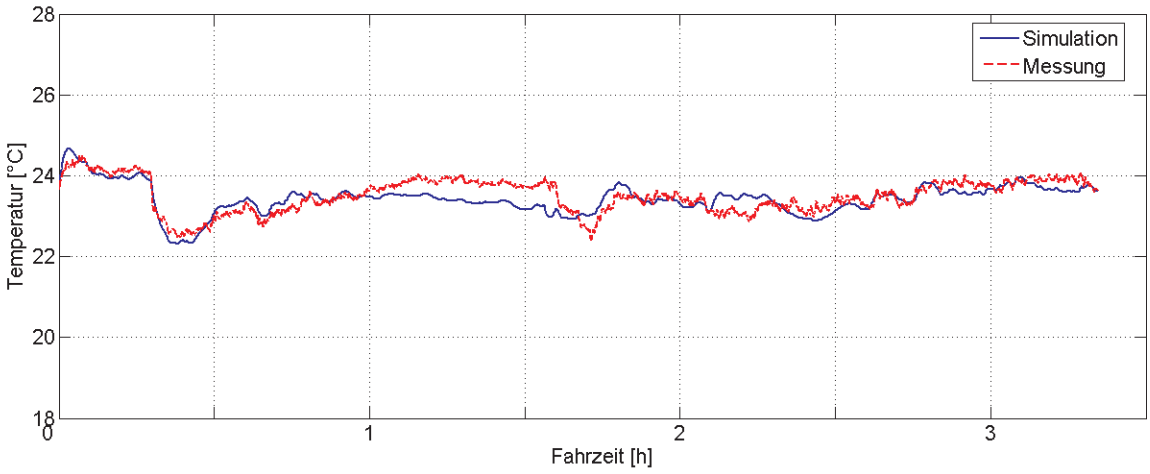


Abbildung 5: Simulation und Messung der mittleren Kabinentemperatur

5 Zusammenfassung und Ausblick

Für die HiL Simulation ist ein physikalisches Kabinenmodell entwickelt worden, dass die mittlere Kabinentemperatur und Luftfeuchte innerhalb einer Fahrzeugkabine berechnet. Die Anforderungen an das Kabinenmodell hinsichtlich der Echtzeitfähigkeit und einer einfachen und parametrisierbaren Struktur konnten erfüllt werden. Weiterhin ermöglicht das Kabinenmodell, integriert in das klimatechnische Fahrzeugmodell, eine vollständige Simulation des gesamten Klimasystems. Zukünftig ist eine Erweiterung des Kabinenmodells angedacht, welches neben einer mittleren Temperatur für die Kabine jeweils eine Temperatur für den Kopf- und Fußraum angeben kann. Dann wären auch einfache Aussagen über die Temperaturverteilung innerhalb der Fahrzeugkabine möglich.

Literatur

[1] Michalek, D.; Bertram, T.; Hiller, M.; Trapp, R.: *Steuergeräte-HIL-Simulation in der Kfz-Klimatisierung als flexibles Entwicklungswerkzeug*, 11. Internationaler Kongress - Berechnung und Simulation im Fahrzeugbau, VDI-Berichte Nr. 1701, VDI Verlag GmbH, Düsseldorf 2002.

[2] Michalek, D.; Bertram, T.; Trapp, R.: *Hardware-in-the-Loop-Simulation für verteilte mechtronische Systeme am Beispiel der Kfz-Klimatisierung*, ASIM 2003, Magdeburg, September 2003.

[3] Gehsat, C.; Bertram, T.; Trapp, R.: *Kabinenmodell für die Hardware-in-the-Loop-Simulation zur Berechnung der Kabinetemperatur im Kraftfahrzeug*, VDI/VDE Mechatronik 2007 - Innovative Produktentwicklung, VDI-Berichte Nr. 1971, VDI Verlag GmbH, Düsseldorf 2007.

Analyse von Wärmetransport- und Kondensationsprozessen in Kfz-Scheinwerfern unter Verwendung von CFD

Dr.-Ing. Thorsten Maschkio, Hella KGaA Hueck & Co.
thorsten.maschkio@hella.com

Zusammenfassung

Der Einsatz von Simulationsmethoden ermöglicht eine erhebliche Verkürzung von Entwicklungszeiten, indem umfassende Systemanalysen bereits in sehr frühen Entwicklungsphasen möglich werden. Im Bereich der automobilen Beleuchtungstechnik finden u.a. CFD-Simulationstools (*Computational Fluid Dynamics*) Verwendung, um die thermische Belastung eingesetzter Materialien sowie eine mögliche, unerwünschte Kondensatbildung in von außen sichtbaren Bereichen zu untersuchen.

1 CFD-Simulation in der Entwicklung von Kfz-Scheinwerfern

1.1 CFD-Simulationsmethode

Als *CFD-Simulation* (auch: *Numerische Strömungsmechanik*) wird die numerische Lösung der (vollständigen) *Navier-Stokes-Gleichungen* bezeichnet, die aus den Erhaltungsgleichungen für Masse, Impuls und Energie gebildet werden. Sie beschreiben u.a. Wärme- und Stofftransportvorgänge unter Beteiligung von Fluiden. Es gibt mittlerweile eine Vielzahl kommerzieller CFD-Simulationstools, die über grafische Benutzerschnittstellen eine komfortable Beschreibung des zu analysierenden Systems, die Lösung des (automatisch generierten und diskretisierten) Gleichungssystems sowie die ansprechende Visualisierung der Ergebnisse ermöglichen.

1.2 Analyse der Wärmetransportvorgänge

Kfz-Scheinwerfer bestehen mittlerweile fast ausschließlich aus Kunststoffen, die während des Scheinwerferbetriebs hohen thermischen Belastungen ausgesetzt sein können. Für die Auswahl eines Kunststoffs sind folgende Kriterien zu berücksichtigen:

- Der Kunststoff muss eine ausreichende Wärmebeständigkeit aufweisen, damit es im Scheinwerfer-Betrieb nicht zu thermischen Schädigungen und/oder Funktionsstörungen kommt.

- Eine Überdimensionierung des Kunststoffs hinsichtlich seiner Wärmebeständigkeit ist zu vermeiden, um eine kosteneffiziente Produktion des Scheinwerfers gewährleisten zu können.

Durch den Einsatz eines CFD-Simulationstools können die drei Wärmetransportmechanismen *Wärmeleitung*, *Wärmestrahlung* und *Konvektion* berücksichtigt werden, wodurch die Vorhersage ermöglicht wird, welche maximalen Bauteiltemperaturen unter definierten Bedingungen zu erwarten sind [1]. Dies wiederum gestattet die Auswahl eines geeigneten Kunststoffs sowie die Planung und Konstruktion der für die Verarbeitung der Kunststoffe erforderlichen Spritzguss-Werkzeuge noch bevor erste Prototypen der Geräte verfügbar sind.

1.3 Analyse der Kondensationsprozesse

Kfz-Scheinwerfer stellen offene Systeme dar und ermöglichen daher den Austausch von Feuchtigkeit (Wasserdampf) mit der Umgebung. Kondensiert ein Teil dieser Feuchtigkeit innerhalb des Scheinwerfers, so wird dies als *Betauung* bezeichnet. Betauung ist insbesondere in von außen sichtbaren Bereichen des Scheinwerfers unerwünscht. Daher ist nach aufgetretener Betauung eine möglichst schnelle *Enttauung* (Verdampfung) erwünscht.
Die Bestimmung derjenigen Bereiche im Scheinwerfer, die unter gegebenen Bedingungen zu Betauung neigen, ist mit Hilfe der CFD-Simulation möglich [2]. Die Abbildung 1 zeigt links diejenigen Bereiche, in denen sich während eines Laborversuchs auf der Lichtscheiben-innenseite Kondensat gebildet hat. Zum Vergleich sind rechts die entsprechenden Ergebnisse einer CFD-Simulation dargestellt.

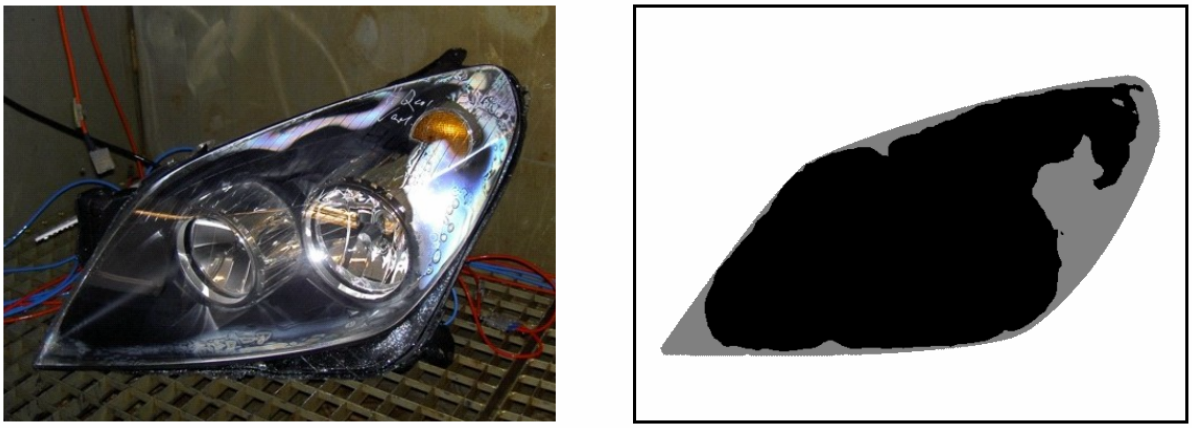


Abbildung 1: Betauungskritische Bereiche im Scheinwerfer

Literatur

[1] Nolte, Sascha: *Eine Methode zur Simulation der Temperatur- und Strömungsverteilung in lichttechnischen Geräten.* Dissertation, Paderborn 2005.

[2] Maschkio, Thorsten: *CFD-Simulation der Be- und Enttauungsprozesse in Kfz-Scheinwerfern.* Dissertation, Cuvillier Verlag Göttingen, Paderborn 2006.

Gekoppelte Simulation von FE- und Mehrkörpermodellen für Werkzeugmaschinen

Oliver Zirn, AG Mechatronik, FH Giessen, D-35390 Gießen,
oliver.zirn@ei.fh-giessen.de
Raphael Montavon, Laboratoire de Conception et Simulation, HE-ARC,
CH-2610 St. Imier, Raphael.Montavon@he-arc.ch

Zusammenfassung

Während FE-Modelle zur Steifigkeitsdimensionierung von Strukturbauteilen an Werkzeugmaschinen heute zum Stand der Technik gehören, ist die dynamische FE-Simulation von Achs- oder Bahnbewegungen nur sehr eingeschränkt möglich. Mit Hilfe von MATLAB/Simulink können ganze Produktionsmaschinenmodelle aus FE-Teilmodellen und Mehrkörpermodellen effektiv nachgebildet und dynamisch simuliert werden. Diese Kombination bietet vor allem für mittelständische Unternehmen mit naturgemäß begrenzten eigenen numerischen Analysemöglichkeiten eine gute Voraussage von Maschineneigenschaften bei vergleichsweise kleinem Aufwand. Der Beitrag zeigt die praxisnahe Vorgehensweise sowie die erreichbare Modellgenauigkeit am Beispiel einer ausgeführten Werkzeugmaschine auf.

1 Einleitung

Die Voraussage der Regelgüte von Servoachsen und der dynamischen Bahngenauigkeit ganzer Werkzeugmaschinen stellen eine Schlüsselkompetenz für die moderne Produktionsmaschinenentwicklung dar. Hierzu bedarf es ausreichend genauer und gleichzeitig schnell implementierbarer Modelle sowie effizienter Simulationswerkzeuge.
Die Modellbildung unter besonderer Berücksichtigung elastischer Struktur und flexibler Übertragungsglieder kann in vier Komplexitätsstufen unterteilt werden. Auf der einfachsten Stufe werden Servoachsen als die zentralen Subsysteme von Werkzeugmaschinen modelliert. Die Wirkung der dominierenden Elastizitäten auf die Regelgüte des Servoantriebs kann anhand eines vereinfachten allgemeinen Systems 4. Ordnung umfassend beschrieben werden [1]. Die Kopplungen zwischen kinematisch verketteten Achsen wird mit den Beziehungen aus der Roboterdynamik auf einem zweiten Komplexitätsniveau beschrieben [2]. Mehrkörper- oder MB-Modelle (engl. Multi-Body) und Finite-Elemente- (FE-) Modelle ermöglichen eine genauere Nachbildung der Manipulatorstruktur für einen vorgegebenen Arbeitspunkt (drittes Komplexitätsniveau). Die Kombination von Roboterdynamik und Starrkörpermodellen ergibt flexible Mehrkörpersysteme mit so genannten „Superelementen“ als höchstem Komplexitätsniveau.

Abbildung 1 zeigt das FE-Modell sowie das MB-Modell des in Kapitel 4 beispielhaft diskutierten Werkzeugmaschinenmanipulators. Das FE-Modell ist zwar sehr detailliert aber würde für die dynamische Simulation [3] – so überhaupt als Sonderfunktion im CAE-Werkzeug vorhanden – enorme Rechenleistung benötigen.

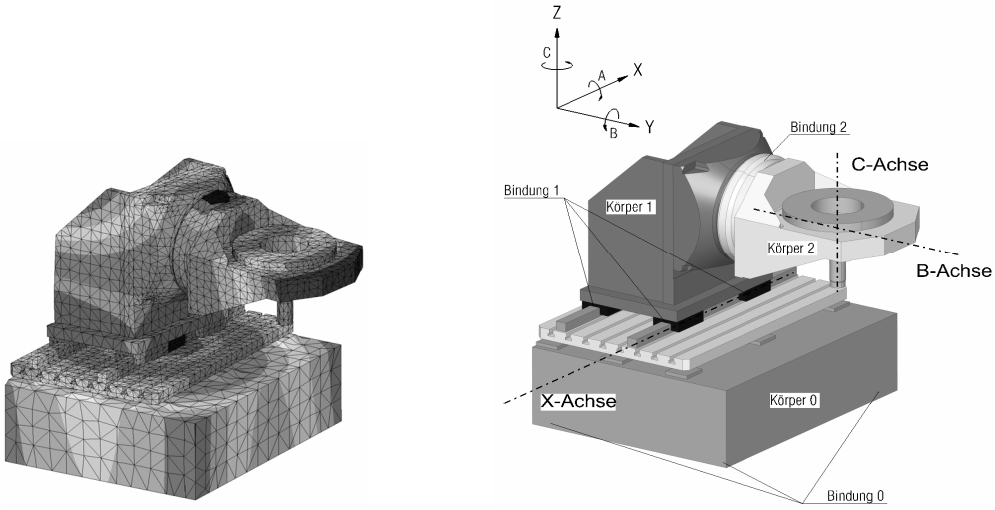


Abbildung 1: FE- und MB-Modell eines Werkzeugmaschinenmanipulators

Realistische Berechnungen für vergleichsweise einfache Achsbewegungen werden auch mittelfristig viele Stunden dauern. Das MK-Modell hingegen kann direkt als Zustandsraumdarstellung in Simulink-Modelle integriert werden und erlaubt die Simulation von Bahnbewegungen nahezu in Echtzeit [4]. Für Werkzeugmaschinen, bei denen meist die Gelenke (Führungen, Lagerungen) für die Struktureigenschwingungen verantwortlich sind, können so recht genaue Simulationsergebnisse erreicht werden.
Oft werden bei mittelständischen Werkzeugmaschinenherstellern nur Teile der Maschinenstruktur mit FE-Werkzeugen analysiert. Zudem verfügen insbesondere Komponentenhersteller nur über FE-Teilmodelle.
Für die praktische Modellbildung ist es daher wünschenswert, Resultate aus der FE-Analyse und der experimentellen Modalanalyse in MATLAB/Simulink-Modelle zu importieren und nicht FE-modellierte Maschinenteile als MK-Teilmodelle nachzubilden.

2 Export des dynamischen FE-Modells nach MATLAB

Die Bewegungsgleichungen in physikalischen Koordinaten

$$\boldsymbol{M} \cdot \ddot{\boldsymbol{x}} + \boldsymbol{K} \cdot \boldsymbol{x} = \boldsymbol{F} \tag{1}$$

mit $\boldsymbol{M}$ - Massenmatrix
$\boldsymbol{K}$ - Steifigkeitsmatrix
$\boldsymbol{F}$ - Vektor der äusseren Kräfte

können mit Hilfe der Eigenvektoren und Eigenwerte für die dominierenden Eigenschwingungsformen und relevanten Knoten (z.B. Kraftangriffspunkte, Meßsystempositionen, Tool Center Point – TCP) in die modalen Koordinaten transformiert werden [5]:

$$\underbrace{\boldsymbol{Z_n}^T \cdot \boldsymbol{M} \cdot \boldsymbol{Z_n}}_{\boldsymbol{M_p}} \cdot \underbrace{\boldsymbol{Z_n}^{-1} \cdot \ddot{\boldsymbol{x}}}_{\ddot{\boldsymbol{x}}_p} + \underbrace{\boldsymbol{Z_n}^T \cdot \boldsymbol{K} \cdot \boldsymbol{Z_n}}_{\boldsymbol{K_p}} \cdot \underbrace{\boldsymbol{Z_n}^{-1} \cdot \boldsymbol{x}}_{\boldsymbol{x_p}} = \underbrace{\boldsymbol{Z_n}^T \cdot \boldsymbol{F}}_{\boldsymbol{F_p}} \tag{2}$$

mit $\boldsymbol{x_p}$ - modale Koordinaten
$\boldsymbol{M_p}$ - modale Massenmatrix (Einheitsmatrix $\boldsymbol{E}$)
$\boldsymbol{K_p}$ - modale Steifigkeitsmatrix (Diagonalmatrix mit den Quadraten der Eigenw.)
$\boldsymbol{F_p}$ - Stellgrößenvektor in modalen Koordinaten
$\boldsymbol{Z_n}$ - modale Matrix der Eigenvektoren für die dominierenden Modes und relevanten DOF (massennormiert)

Die massennormierten Eigenvektoren und die Eigenwerte sind als Ergebnisse der FE-Modalanalyse mit vergleichsweise kleinem Aufwand in MATLAB exportierbar. Die modale Zustandsraumdarstellung ergibt sich – inklusive proportionaler Dämpfung - somit zu:

$$\begin{pmatrix} \dot{\boldsymbol{x}}_p \\ \ddot{\boldsymbol{x}}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{K}_p & -2 \cdot D \cdot \sqrt{\boldsymbol{K}_p} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_p \\ \dot{\boldsymbol{x}}_p \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_n \end{pmatrix} \cdot \boldsymbol{F}$$
$$\boldsymbol{y} = \boldsymbol{C} \cdot \begin{pmatrix} \boldsymbol{Z}_n & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Z}_n \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_p \\ \dot{\boldsymbol{x}}_p \end{pmatrix} \tag{3}$$

mit $\boldsymbol{C}$ - Ausgangsmatrix (Meßsystem- und TCP-Positionen)

Diese Zustandsraumdarstellung gemäß Gleichung (3) kann leicht als Struktur-Teilmodell in Simulink implementiert werden, wie Abbildung 5 b beispielhaft zeigt.

3 Hybride Modelle

Oft existieren FE-Modelle nur für einige kritische Komponenten einer Produktionsmaschine. Diese FE-Teilmodelle können in ein Mehrkörpermodell integriert werden, wie dies Abbildung 2 beispielhaft für den in Kapitel 4 beispielhaft diskutierten Werkzeugmaschinen-manipulator zeigt:

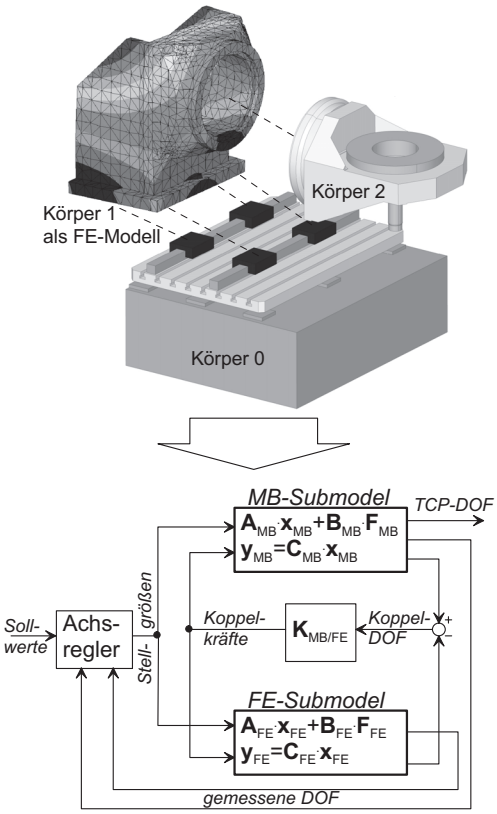


Abbildung 2 : Kombination von FE- und MK-Teilmodellen

Diese hybriden Modelle sind heute jedoch noch vergleichsweise aufwändig zu implementieren. Dabei hat sich die in Abbildung 2 dargestellte Zustandsdarstellung für die Teilmodelle als hilfreich erwiesen. In der Koppelmatrix $\boldsymbol{K_{MB/FE}}$ sind die Steifigkeiten an den Schnittstellen zwischen den Teilmodellen zusammengefasst. Diese Schnittstellen liegen geeigneterweise an Führungen, Lagern oder anderen in ihren Eigenschaften klar definierbaren Bauteilen. Die mit diesen hybriden Modellen erreichbaren Simulationsgenauigkeiten liegen bei geeigneter Wahl der FE-Teilmodelle nahe an den mit FE-Komplettmodellen erreichbaren Genauigkeiten.

4 Modellbildungsbeispiel

Die mittels MB- und FE-Modellierung erreichbaren Simulationsgenauigkeiten werden am der in Abbildung 3 dargestellten Werkzeugmaschine verdeutlicht. Diese Anordnung ist typisch für den werkstückseitigen Manipulator an Fräsmaschinen. Durch den exzentrischen Schwerpunkt der B-Achse kommt es bei Beschleunigungsvorgängen zu koppelkraftbedingten Störungen zwischen B- und X-Achse. Die dabei auftretenden Abweichungen werden an den Achsmeßsystemen sowie direkt im TCP erfasst.

a) b)

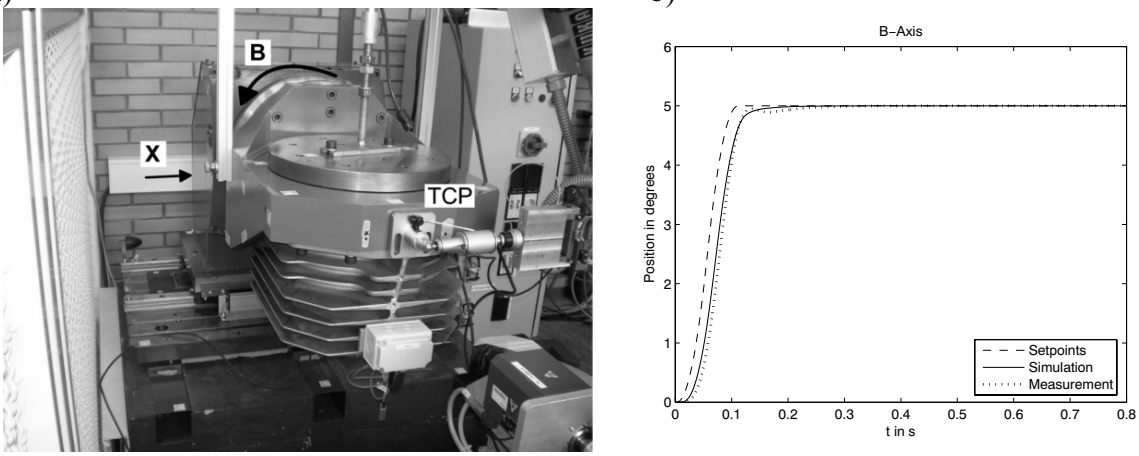


Abbildung 3 Direktangetriebener Manipulator (a) mit 3 DOF (Rückle GmbH / FH Giessen) und Positionierbewegung der B-Achse (b)

Aufgrund des kompakten Aufbaus des Manipulators liegt zunächst eine Nachbildung mit drei räumlichen Starrkörpern gemäß Abbildung 1 nahe. Das dazugehörige Simulink-Modell zeigt Abbildung 5 b. Damit können alle Eigenschwingungsformen (engl. Modes), die für die Achsen wirksam werden, gut nachgebildet werden. Dies wird durch die Antriebsfrequenzgänge (siehe Abbildung 6) verdeutlicht. Ebenso wird der Verlauf der Achspositionen (am Beispiel eines Positioniervorgangs der B-Achse in Abbildung 3 b) sehr zuverlässig simuliert. Abbildung 7 verdeutlicht, dass die koppelkraftbedingten Störungsamplituden mit und ohne Koppelkraftkompensation (CTC – engl. Computed Torque Control) zuverlässig vorausgesagt werden.
Betrachtet man hingegen die gemessenen und simulierten Verläufe der TCP-Position, so fällt auf, dass sowohl die Störungsamplitude, als auch das schlecht gedämpfte Nachschwingen nur unzureichend vom Modell wiedergegeben werden.

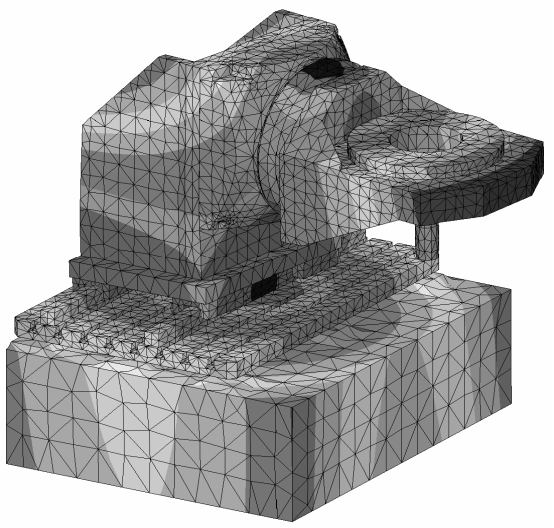


Abbildung 4 : Animierte Darstellung der durch die Strukturelastizität des Turmes der X-Achse bedingten Eigenschwingungsform

Offenbar ist eine wesentliche Strukturelastizität - die elastische Struktur des Turms der X-Achse - nicht durch das MB-Modell erfasst, wie Abbildung 4 verdeutlicht.
Ersetzt man das MB-Modell durch das in Abbildung 5 dargestellte FE-Modell, so wird die Simulationsgenauigkeit wesentlich gesteigert, wie die Achsposition (Abbildung 8 a) und vor allem die TCP-Position in Abbildung 8 b verdeutlicht.

a) b)

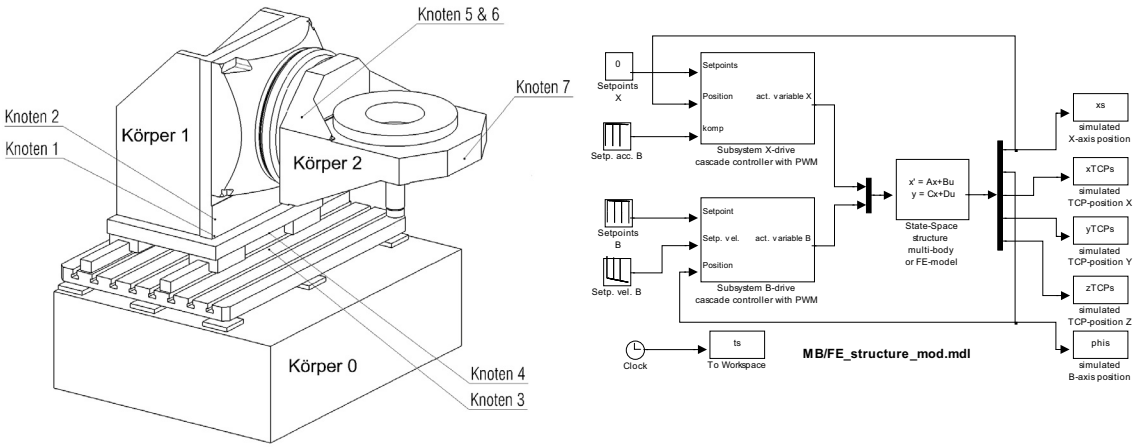


Abbildung 5 : FE-Modell (a) des Manipulators mit den für das MATLAB-Strukturmodell (b) relevanten Knoten

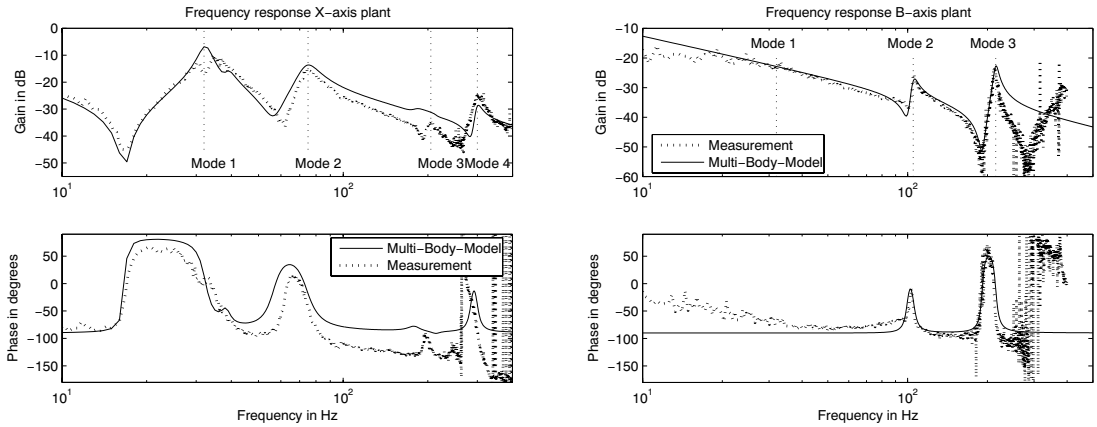


Abbildung 6 : Antriebsfrequenzgänge der X- und B-Achse

a) b)

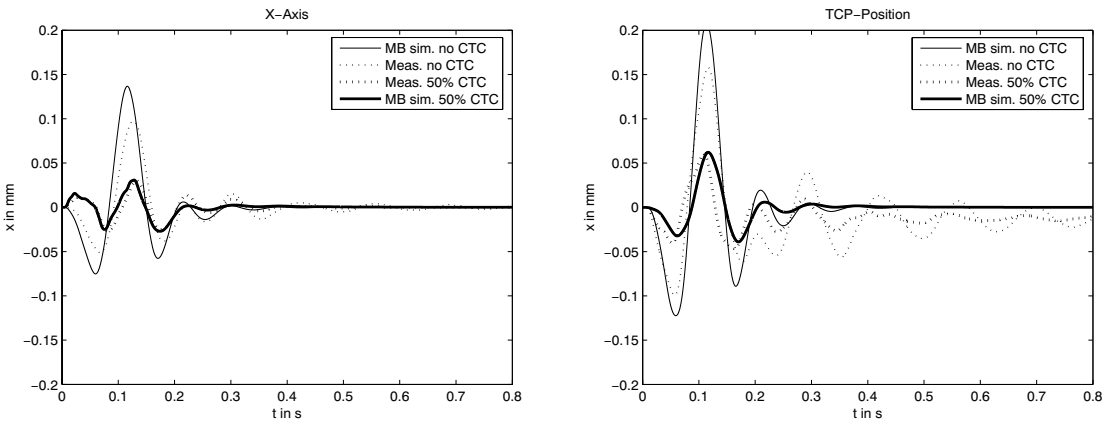


Abbildung 7 : Störungen X-Achse (a) und im TCP (b) mit den Auswirkungen der Koppelkraftkompensation (MK-Modell)

a) b)

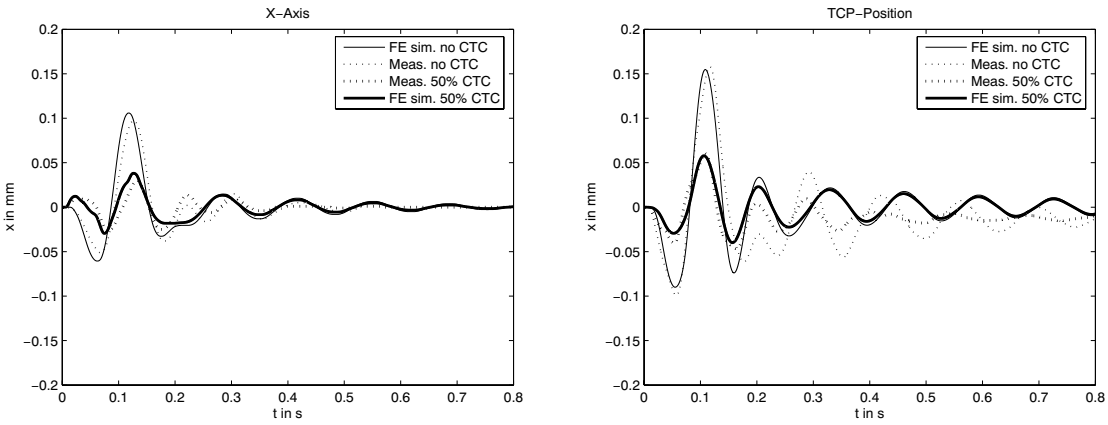


Abbildung 8 : Störungen X-Achse (a) und im TCP (b) mit den Auswirkungen der Koppelkraftkompensation (FE-Modell)

5 Zusammenfassung

Dieser Beitrag zeigt, wie Ergebnisse aus der FE-Analyse in die dynamische Modellierung von Werkzeugmaschinen einfließen können. Die damit gegenüber den recht effizient implementierbaren MB-Modellen erzielbare Verbesserung der Simulationsgenauigkeit kann signifikant sein. Jedoch steigt der Implementierungsaufwand – vor allem bei hybriden Modellen – erheblich.

Danksagung

Die dargestellten Ergebnisse wurden im Rahmen des angewandten Forschungsprojektes AiF-FH^3 FKZ 1733 X04 erarbeitet.

Literatur

[1] Zirn O, Weikert S (2005) *Modellbildung und Simulation hochdynamischer Fertigungssysteme.* Springer-Verlag, Heidelberg, 2005.

[2] Tsai L (1999) *Robot Analysis - The Mechanics of Serial and Parallel Manipulators*. Wiley & Sons.

[3] Kehl G (2007) *Untersuchungen zur gekoppelten Simulation von Strukturdynamik und Regelungstechnik.* CADFEM GmbH Seminar on ANSYS, 25.4.2007, www.cadfem.de.

[4] Lorenzer Th, Weikert S, Wegener K (2007) *Decision-making aid for the design of reconfigurable machine tools.* International Conference on Changeable Agile Reconfigurable and Virtual Production (CARV). Toronto, Ontario, Canada, July 23-24 2007.

[5] Hatch M R (2000) *Vibration simulation using MATLAB and ANSYS*. CRC PressLLC, Boca Raton FL.

Erstellung eines MKS – Modells zur Untersuchung der Dynamik einer Nanopositionier- und Messmaschine (NPMM)

Isabel Kirchner, Erik Gerlach, Siegfried Oberthür, Klaus Zimmermann
TU Ilmenau, Fakultät für Maschinenbau, Fachgebiet Technische Mechanik
Isabel.Kirchner@tu-ilmenau.de

Zusammenfassung

Im Rahmen des Sonderforschungsbereiches 622 werden an der TU Ilmenau die Grundlagen zur Entwicklung von Nanopositionier- und Nanomessmaschinen (NPM-Maschinen) erarbeitet. Im Rahmen eines Virtuellen Prototyping, für welches die Modellgenerierung eine wesentliche Basis darstellt, sollen die dynamischen Eigenschaften des Systems in frühen Entwicklungsphasen bestimmt werden. Die Analyse der Dynamik erfolgt durch die Simulation eines Mehrkörpermodells. Durch Integration von Antrieben und Regelungen in das Modell wird ein mechatronisches System computergestützt analysiert.

1 Einleitung

Die computergestützte Simulation des dynamischen Systemverhaltens ist im klassischen Maschinenbau, der Fahrzeugtechnik und der Robotik Stand der Technik. Im Rahmen des Sonderforschungsbereiches 622 „Nanopositionier- und Nanomessmaschinen“ werden Modelle für NPM-Maschinen entwickelt. Diese Maschinen sind technologische Ausrüstungen, welche die Positionierung, Messung, Antastung, Modifizierung und Manipulation von dreidimensionalen Objekten mit Nanometerpräzision ermöglichen. Für die effiziente Weiterentwicklung von NPM-Maschinen und Untersuchungen hinsichtlich ihres dynamischen Verhaltens erfolgt eine computergestützte Analyse.

2 Modellgenerierung für die NPM-Maschine

Es sind Modelle zu erstellen, die Aussagen über die Eigenschaften der realen Bewegungssysteme ermöglichen, wodurch eine Einschätzung der erreichbaren Positioniergenauigkeit für alle drei Raumachsen erreicht wird.

Ausgehend von einem Konstruktionsentwurf oder einem realen System wird ein Mehrkörpersystem(MKS)-Modell erstellt. Im ersten Modellierungsschritt besteht dieses aus starren Körpern mit physikalischen und/oder geometrischen Kopplungen. Die für die Koppelstellen erforderlichen Parameter werden theoretisch oder experimentell ermittelt und

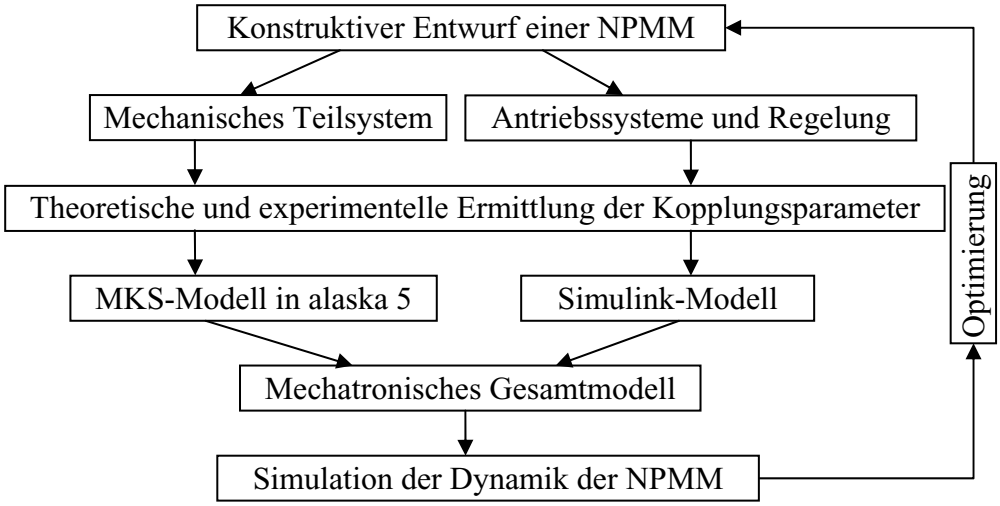


Abbildung 1: Methodik der Simulation

als Funktionen hinterlegt. Später erfolgt für ausgewählte Körper der Übergang vom starren zum elastischen System. Diese hybriden MKS-Modelle als Grundlage für die modellgestützte Untersuchung der Dynamik werden zu mechatronischen Gesamtsystemen erweitert. Dazu werden mittels MATLAB/Simulink antriebstechnische und regelungstechnische Komponenten in das mit alaska 5 (Institut für Mechatronik e.V. Chemnitz) erstellte mechanische System integriert. Abbildung 1 zeigt die methodische Vorgehensweise.

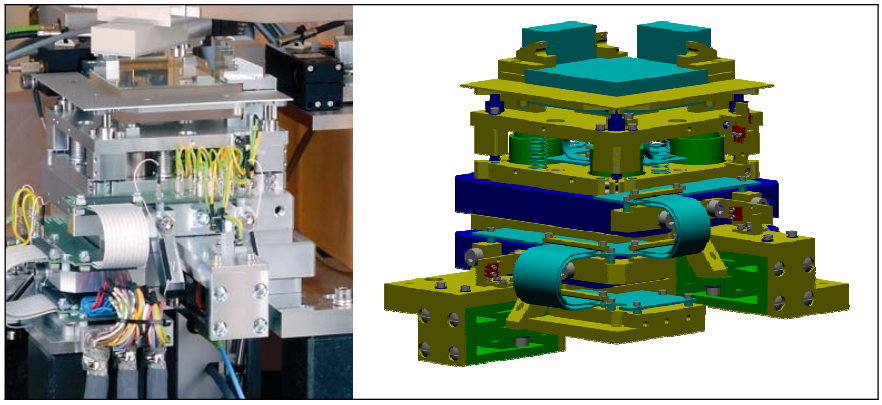


Abbildung 2: Bewegungssysteme der NPM-Maschine (l.) und Modell in alaska 5 (r.)

Der Umfang der erstellten Modelle hängt dabei von den jeweiligen Fragestellungen ab. So können die Module zur Realisierung der drei translatorischen Bewegungen unabhängig voneinander oder, wie in Abbildung 2, gekoppelt betrachtet werden. In weiteren Modellen werden zukünftig auch die Aufstellungsbedingungen in die Betrachtungen mit einbezogen.

Literatur

[1] Gerlach, E., Zimmermann, K., Tröbs, A.: Simulationsgestützter Entwurf einer hochpräzisen Positioniereinrichtung für Nanopositionier- und Nanomessmaschinen, *Mechatronik 2005*, VDI-Bericht 1892, Wiesloch, 2005

[2] Gerlach, E., Zimmermann, K., Oberthür, S., Tröbs, A.: Simulating the dynamics of Nanopositioning and Nanomeasuring Machines using Methods of Multi-body System Dynamics, Proceedings of the 6th EUROSIM Congress, Ljubljana, Slovenia, 2007

Mathematische Modelle zur Unterstützung der Prostatakrebsdiagnostik für Mediziner

Günter Schneckenreither, Felix Breitenecker
Institut für Analysis und Scientific Computing, TU Wien
{gschneck,fbreiten}@osiris.tuwien.ac.at
Günther Zauner
"Die Drahtwarenhandlung" – Simulation Services, Wien
Anton Ponholzer
Urologische Abteilung, Donauspital – SMZ-Ost, Wien

Zusammenfassung

Die Messung von Krebs-Indikatoren im Blut und die Gewebeentnahme aus der Prostata stellen zwei zentrale Aspekte der Prostatakrebsdiagnostik dar. Die mathematischen Modelle in diesem Beitrag sollen die Interpretation von Untersuchungsergebnissen aus diesen Bereichen erleichtern, ein besseres Verständnis für die angewendeten Methoden liefern und helfen, neue Diagnosestrategien zu entwickeln.

1 Einleitung

Die erfolgreiche Behandlung von Prostatakrebs ist oft nur in frühen symptomlosen Krankheitsstadien möglich. Regelmäßige Vorsorgeuntersuchungen erhöhen jedoch die Wahrscheinlichkeit der Diagnosestellung. Bei regelmäßigen Blutabnahmen wird die Konzentration des Prostata-spezifischen Antigens (PSA) gemessen. Dieser PSA-Wert dient als Indikator für Prostatakrebs; steigt jedoch auch ohne Prostatakarzinom linear mit dem Alter des Patienten [6]. Aufgrund unerwartet hoher Messdaten kann sich der Arzt zu einer Gewebeentnahme zwecks genauerer Untersuchung entscheiden (Abschnitt 2).

Je nach Alter des Patienten wird in Abhängigkeit der Größe zwischen klinisch signifikanten und nicht signifikanten Tumoren unterschieden [2, 8]. Ziel der Gewebeentnahme ist es, mit einer bestimmten Wahrscheinlichkeit das Vorhandensein von Tumoren einer gewissen Signifikanz ausschließen zu können. Die Entnahme von Gewebe aus der Vorsteherdrüse erfolgt ultraschallgestützt mittels Stanznadeln (Biopsie) und ist für den Patienten in der Regel schmerzhaft. In der medizinischen Literatur werden verschiedenste Stichmodelle untersucht [1, 3, 4], die die optimale Positionierung der Stiche behandeln. Dabei wird versucht mit einer möglichst geringen Anzahl an Stichen eine möglichst verlässliche

Aussage treffen zu können. Anhand geometrischer Modelle [7, 8] oder computergestützter Simulation [1, 2, 4, 5] wird die Effizienz solcher Stichmodelle analysiert (Abschnitt 3).

2 PSA Messung

Das Prostata-spezifische Antigen ist ein körpereigenes Enzym und ist unter anderem in Abhängigkeit vom Alter des männlichen Patienten in bestimmten typischen Mengen im Blut vorhanden. Im Fall eines Karzinoms ist diese Konzentration untypisch höher und es kann mit einem – wie in der medizinischen Literatur [8] übereinstimmend angenommen – exponentiellen Anstieg gerechnet werden. Dieses Verhalten dient als Basis für die Verwendung des PSA-Wertes als Indikator für Prostatakrebs.

Stehen dem behandelnden Arzt mehrere datierte Messwerte zur Verfügung, kann er aus der zeitlichen Entwicklung auf das Vorhandensein von Krebs schließen. Im Allgemeinen kann aber erst ab einer höheren Anzahl von Messungen eine zuverlässige Prognose abgegeben werden. Liegt der momentane PSA-Wert unterhalb einer kritischen Schwelle, wird die erwartete Verdoppelungszeit als Grundlage für die Festlegung des nächsten Untersuchungstermins verwendet.

Ein mathematisches Modell, das den Arzt in seine Entscheidungsfindung unterstützen soll, wurde von unserer Arbeitsgruppe im Rahmen eines E-Learning Projektes und als Web-Diagnose Applikation (Abbildung 1) realisiert [6]. Dem Anwender werden eine lineare $f_{\text{lin}}(t)$ und gleichzeitig eine exponentielle $f_{\text{exp}}(t)$ Interpolation der Messwerte geliefert. Zusätzlich werden die Verdoppelungszeit oder auch Schwankungsbreiten basierend auf den typischen Streuungen von PSA-Messwerten angezeigt. Ein wesentliches Ziel dabei ist, den Anwender auf die Auswirkung von unvermeidbaren Messfehlern aufmerksam zu machen und ein Bewusstsein für die richtige Anwendung von mathematischen Modellen zu schaffen.

Der zugrundeliegende Interpolationsalgorithmus verwendet die Methode der kleinsten Fehlerquadrate und liefert Funktionen mit zwei bzw. drei Parametern.

$$f_{\text{lin}}(t) = k \cdot t + d \qquad \text{bzw.} \qquad f_{\text{exp}}(t) = a + b \cdot e^{c \cdot t} \tag{1}$$

3 Biopsie

Um allgemeine Aussagen über Stichmodelle treffen zu können, müssen diese an verschiedensten Szenarien getestet werden. Vor allem die Größe (Signifikanz) der Krebsgeschwüre, deren Verteilung innerhalb der Prostata und das Volumen der Prostata spielen dabei eine Rolle – auch im Kontext des Patientenalters. Solche Daten werden zu einem gewissen Teil bei Biopsien gewonnen. Vor allem aber Computer-Tomographien (CT) und histologische Schnittanalysen oder sogar Biopsien an Präparaten von radikalen

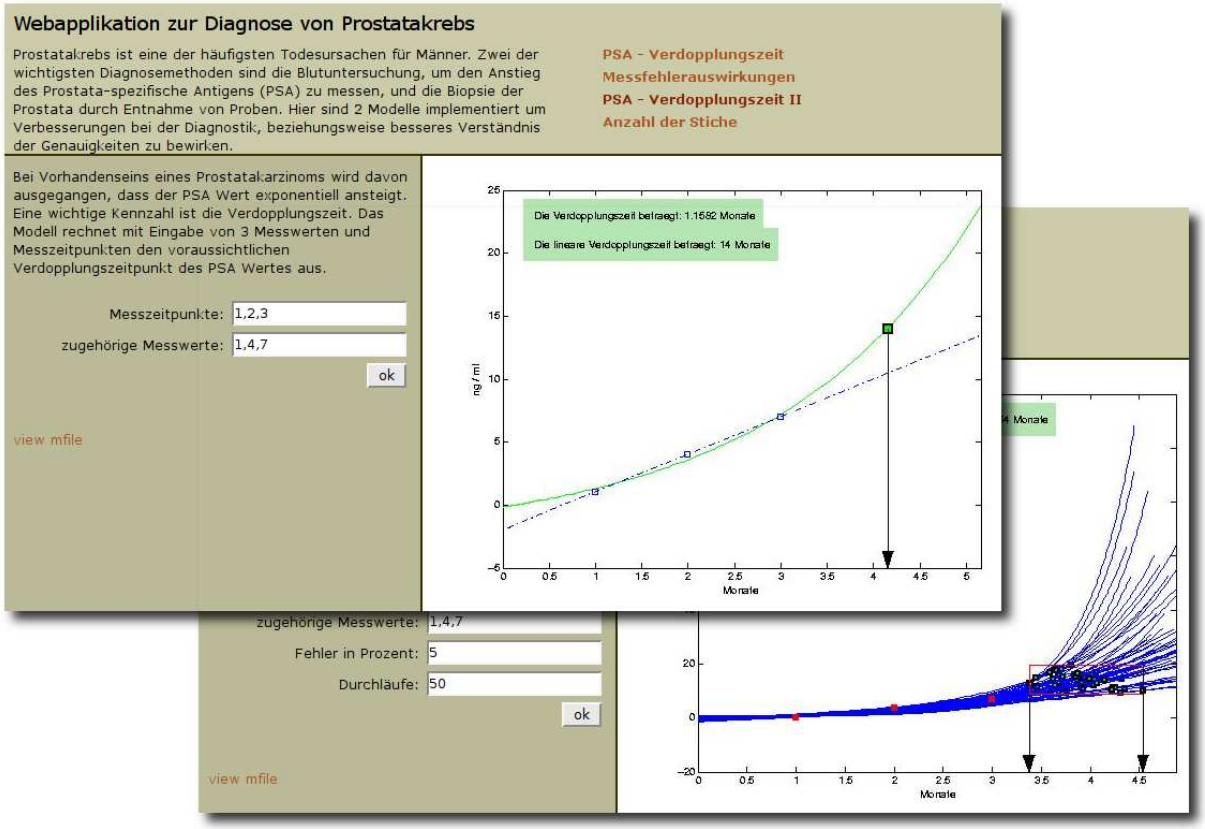


Abbildung 1: Web-Oberfläche für die Berechnung der zeitlichen Entwicklung des PSA-Wertes in Abhängigkeit der eingegebenen Messdaten und der zugehörigen Zeitpunkte. Die Berechnungen erfolgen durch ein MATLAB Programm auf einem Server, der via PHP die Ergebnisse in Form einer Grafik an den Anwender zurücksendet. Die Verdoppelungszeit wird durch einen Pfeil in der Grafik angezeigt. Auch die Schwankungsbreite, die bei gestörten Messwerten entsteht, kann ausgegeben werden.

Prostatektomien (entnommenen Drüsen) liefern Aufschluss über die Verteilung und Größe von Krebsgeschwüren [1, 2, 3, 4, 5].

Rein statistische Analysen von Stichmodellen beinhalten Ergebnisse von Untersuchungen mit der jeweiligen Stichmethode, den weiteren Krankheitsverlauf des Patienten und allgemein bekannten Daten bezüglich der Größe und Häufigkeit von Tumoren. Computergestützte Vergleichsstudien von Stichmodellen können zum Beispiel auch das Auftreten von Tumoren spezifischer Signifikanz in den unterschiedlichen Zonen der Prostata (vgl. Abbildung 2) untersuchen [1, 4, 5].

3.1 Verhältnismodell

Die grundlegende Idee eines ersten Ansatzes von Stricker et al. in [7] ist die Betrachtung des Verhältnisses von Tumorvolumen V_T zu gesamtem Volumen der Prostata V_G (gland volume). Durch dieses Verhältnis $p = \frac{V_T}{V_G}$ lässt sich die Wahrscheinlichkeit auf einen Tumor zu treffen beschreiben. Weiterführend wird die Wahrscheinlichkeit bei k Proben auf einen Tumor zu treffen mit $1 - (1 - p)^k$ berechnet [8].

In [8] erweitern bzw. verbessern Vashi et al. diesen Ansatz, indem sie das Verhältnis von untersuchtem Prostatavolumen V_C (core volume) zu gesamtem Prostatavolumen V_G

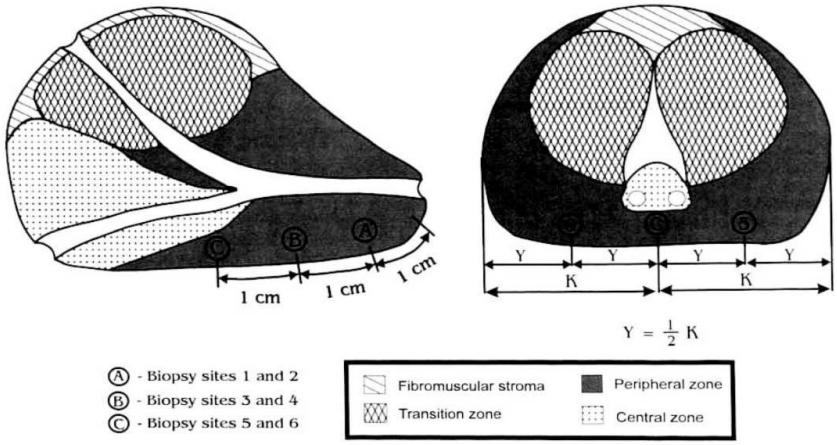


Abbildung 2: [2] Aufbau der Prostata.

betrachten. Dabei wird angenommen, dass sich der kugelförmige Tumor mit bestimmtem Radius r an einer gleichförmig zufälligen Stelle ganz innerhalb der Prostata befindet. Liegt der Mittelpunkt des Tumors um weniger als r Längeneinheiten von dem gedachten Nadelstich mit Eintrittstiefe l entfernt, wird er von der Nadel getroffen. Somit ist der untersuchte Raum gegeben durch alle Punkte, die einen Abstand von weniger als r von der Nadel haben, was einem Zylinder mit aufgesetzter Halbkugel entspricht (Abbildung 3).

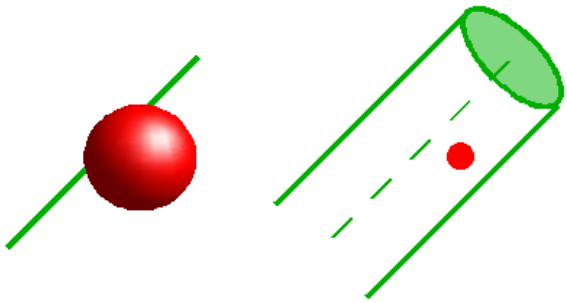


Abbildung 3: Nadelstich, der einen kugelförmigen Tumor mit Radius r trifft. Eine dazu äquivalente Situation ist, wenn sich der Mittelpunkt des Tumors innerhalb des Zylinders mit Radius r um die Nadel befindet.

Will man diesen Ansatz auf mehr als einen Stich erweitern, müssen eventuelle Überlappungen mehrerer Zylinder und in Folge dessen die Lage der Stiche innerhalb der Prostata berücksichtigt werden. Die Prostata kann vereinfachend als Ellipsoid mit den Längen a, b und c der Halbachsen angenähert werden. In der folgenden Diskussion befindet sich der Koordinatenursprung im Mittelpunkt des Prostata-Ellipsoids. Die Einstichebene wird entsprechend der Dimensionierung des Ellipsoids parallel zur xy-Ebene durch den Punkt $(0, 0, -c)$ angenommen (vgl. Abbildung 5).

Für ein mathematisches Modell mit variierender Stichzahl und beliebig definierbaren Stichwinkeln ist die analytische Berechnung des gesamten untersuchten Volumens kaum bewältigbar. Eine Diskretisierung des dreidimensionalen Raums in Zellen erlaubt die Approximation dieses Volumens mit beliebiger Genauigkeit. Das Verhältnis von unter-

suchtem zu gesamtem Volumen lässt sich durch das Verhältnis der Anzahl der untersuchten Zellen N_C zur Anzahl der Zellen, die in der Prostata liegen, N_G annähern.

$$p = \frac{V_C}{V_G} \approx \frac{N_C}{N_G} \tag{2}$$

Da die Zylinder i.A. schief zum Koordinatensystem liegen, werden vor allem wegen der Implementierung zwei Drehungen $D_{i,1}$ und $D_{i,2}$ und eine Translation T_i verwendet, um einen senkrechten Zylinder S_0 (der Länge l) mit Eintrittspunkt $(0, 0, -c)$ in die Gewünschte Position zu bringen. Diese linearen Abbildungen sind für jeden Stich bzw. Zylinder S_i $(i = 1, \ldots, k)$ verschieden. Damit kann für jede Zelle des diskretisierten Raumes entschieden werden, ob sie innerhalb eines Zylinders liegt und somit als untersucht gilt bzw. ob die Zelle überhaupt Teil des Prostata-Volumens ist (Abbildung 4).

Zur Verdeutlichung der Funktionsweise auf Programmebene, wird eine Zelle mit den Raumkoordinaten (x, y, z) betrachtet.

$$\text{Zelle der Prostata (G)} \quad \Leftrightarrow \quad \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \leq 1 \tag{3}$$

$$\text{untersuchte Zelle (C)} \quad \Leftrightarrow \quad \begin{cases} \exists i \in \{1, \ldots, k\} \text{ sodass} \\ (x, y, z) \in S_i = T_i \circ D_{i,2} \circ D_{i,1}(S_0) \end{cases} \tag{4}$$

$$\Leftrightarrow \quad \begin{cases} \exists i \in \{1, \ldots, k\} \text{ sodass} \\ D_{i,1}^{-1} \circ D_{i,2}^{-1} \circ T_i^{-1}(x, y, z) \in S_0 \end{cases} \tag{5}$$

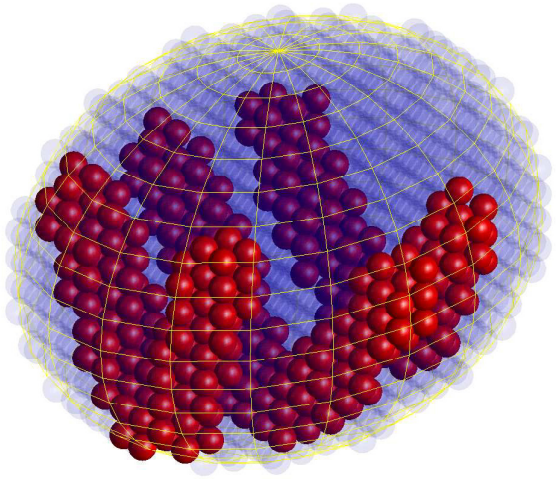


Abbildung 4: Diese mit MATLAB erstellte Abbildung zeigt die untersuchten Zellen (rote Kugeln) und die Prostata-Zellen (transparente blaue Kugeln) eines sechs-Stich Modells auf einem Gitter mit einer Auflösung von $20 \times 20 \times 20$ Zellen.

3.2 Stochastisches Modell

Der stochastische Ansatz bedient sich der Monte Carlo Methode um die "Trefferrate" eines Stichmodells zu ermitteln. Die Stichpositionen werden vor der Simulation festgelegt

und eventuell mit Störungen behaftet. Aufgrund einer bekannten Verteilung der Größen und Positionen der Krebsgeschwüre, werden zufällige Szenarien erzeugt, mit denen die Trefferhäufigkeit ermittelt werden kann.

Wurden in einem konkreten Simulationsdurchlauf die Größe (Radius r) und Position $\vec{x}_T = (x_T, y_T, z_T)$ des Tumors festgelegt, entscheiden die Schnittpunkte der Nadel mit der Tumor-Kugel ob ein Treffer vorliegt. Die Nadelstiche mit Länge l und bestimmter Positionierung (Eintrittsstelle $\vec{x}_E$, Richtungsvektor $\vec{x}_R$) sind gegeben durch

$$\vec{x} = \vec{x}_E + t \cdot \vec{x}_R, \qquad t \in [0, l] \tag{6}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_E \\ y_E \\ z_E \end{pmatrix} + t \cdot \begin{pmatrix} x_R \\ y_R \\ z_R \end{pmatrix}, \qquad t \in [0, l]. \tag{7}$$

Einsetzten in die Gleichung der Tumoroberfläche ergibt

$$(x_E + tx_R - x_T)^2 + (y_E + ty_R - y_T)^2 + (z_E + tz_R - z_T)^2 = r^2. \tag{8}$$

Wenn die Lösung t dieser Gleichung reell ist und im Intervall $[0, l]$ liegt, wurde der Tumor getroffen. In einer Implementierung können diese Werte nach Umformen von (8) als Nullstellen eines Polynoms vom Grad zwei berechnet werden.

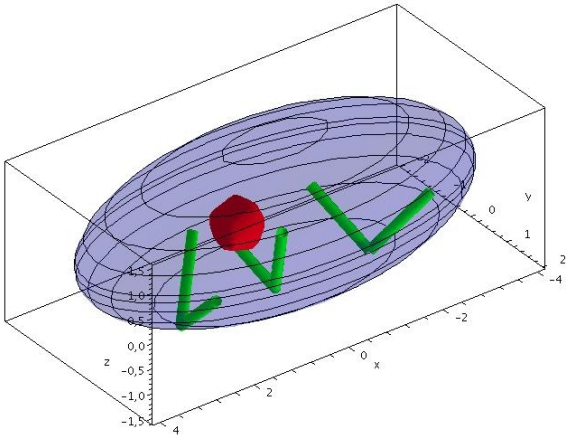


Abbildung 5: Diese mit MAPLE erzeugte Abbildung zeigt eine Stichsituation mit sechs Stichen (grüne Zylinder) und einem zufällig gewählten Tumor (rote Kugel).

4 Diskussion

Der Vorteil des stochastischen Biopsie-Modellansatzes im Gegensatz zu ersterem ist, dass nicht nur eine gleichmäßige Verteilung eines einzelnen Tumors mit bestimmter Größe innerhalb der Prostata untersucht werden kann. Es ist wie schon erwähnt möglich, bestimmte Verteilungen von mehreren Geschwüren mit unterschiedlichen Größen zu verwenden. Erweiterungen dieses Ansatzes, wie in [5], verwenden auch realistische Formen der Prostata, die mit Hilfe von CT anhand von Präparaten digital erstellt wurden. Weiters wird auch die Menge an Krebsgewebe in den Stichproben analysiert, wodurch Aussagen über die Signifikanz von Geschwüren und deren Häufigkeit verbessert werden können [1].

Wird im stochastischen Ansatz eine gleichmäßige Verteilung eines Tumors mit bestimmter Größe verwendet, stimmen die Resultate mit denen des Verhältnismodells gut überein.

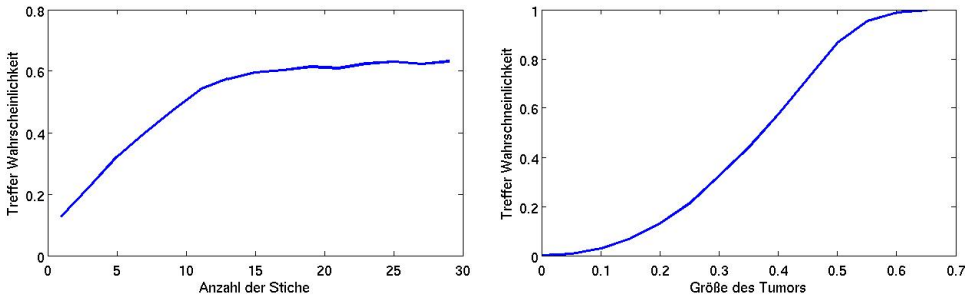


Abbildung 6: Die linke Abbildung zeigt die Trefferwahrscheinlichkeit für einen Tumor mit Radius 0.3 über die Anzahl der Stiche. Das Ergebnis wurde über 20 000 Simulationsdurchläufe gemittelt. Der rechte Plot zeigt die Trefferwahrscheinlichkeit über die Größe des Tumors. Hier wurde das Ergebnis über 3 000 Durchläufe gemittelt. Die Stiche wurden mit einer Nadel der Länge 2 simuliert und liegen in der yz-Ebene mit Winkel $\frac{\pi}{8}$ zur z-Achse. Es wurden jeweils zwei Stiche mit der selben Eintrittsstelle simuliert. Die Eintrittsstellen sind gleichmäßig über die Länge der Prostata ($a = 2, b = 1.5, c = 1$) verteilt (vgl. Abbildung 5).

Zwei Resultate der Computersimulation werden in Abbildung 6 gezeigt. Das deutliche Abflachen der Trefferrate bei Erhöhung der Stichanzahl erklärt sich dadurch, dass gewisse Bereiche der Prostata schwer oder bedingt durch die Länge der Nadel überhaupt nicht zu erreichen sind. Weiters wird deutlich, dass die Trefferrate stark von der Größe des Tumors abhängt und kleine Karzinome praktisch nicht aufzufinden sind.

Wie in Abschnitt 2 gilt auch hier, dass mathematische Modelle bzw. die Interaktion mit diesen Modellen ein besseres Verständnis für die Materie liefern können. Auch in der geometrisch stark vereinfachten Situation, werden “reale” Gegebenheiten wiedergespiegelt, die für die Effizienz der Untersuchung maßgeblich sind. Als konkretes Beispiel sei angeführt, dass (unter den Voraussetzungen der linken Grafik in Abbildung 6) bei der Erhöhung der Stichzahl von 15 auf 20 keine Steigerung der Trefferrate erwartet werden kann. Eine größere Anzahl von Stichen ist zudem schmerzhafter für den Patienten und bereitet ein höheres gesundheitliches Risiko.

Hingegen bei der Computersimulation mit stochastischen Modellen können Schwankungen der Ergebnisse oft nur durch eine sehr hohe Anzahl an Durchläufen herausgefiltert werden. Bei der Simulation zu der linken Grafik in Abbildung 6 waren zum Beispiel 20 000 Durchläufe notwendig, um eine relativ glatte Kurve zu erhalten.

Die derzeit noch gängige “Sextantenbiopsie” mit sechs Stichen wird bei allen Modelluntersuchungen und Computersimulationen als überholt betrachtet. Mit den Testumgebungen werden verschiedenste Stichmodelle getestet und entwickelt.

Literatur

[1] M.E. Chen, P. Troncoso, K. Tang, R.J. Babaian, D. Johnston: *Comparison of prostate biopsy schemes by computer simulation*, Urology, Elsevier Science, 1999.

[2] E.D. Crawford, D. Hirano, P.N. Werahera, M.S. Lucia, E.P. DeAntoni, F. Daneshgari, P.N. Brawn, V.O. Speights, J.S. Stewart, G.J. Miller: *Computer modeling of prostate biopsy: Tumor size and location – not clinical significance – determine cancer detection*, The Journal of Urology, Vol. 159, 1260-1264, American Urological Association, 1998.

[3] K.G. Fink, G. Hutarew, B. Esterbauer, A. Jungwirth, O. Dietze, N.T. Schmeller, *Effizienz der Rebiopsie der Prostata: Untersuchung von Transitionalzonen- und lateralen Biopsien*, Journal für Urologie und Urogynäkologie, 10(4)/2003, Krause & Pachernegg, 2003.

[4] M. Noguchi, D. Deguchi, J. Toriwaki, K. Mori, Y. Mekada, K. Matsuoka: *Evaluation of a prostate biopsy strategy for cancer detection using a computer simulation system with virtual needle biopsy for three-dimensional prostate models*, International Journal of Urology, Vol. 13-10, 1296-1303, Blackwell Science, 2006.

[5] M.B. Opell, J. Zeng, J.J. Bauer, R.R. Connelly, W. Zhang, I.A. Sesterhenn, S.K. Mun, J.W. Moul, J.H. Lynch: *Investigating the distribution of prostate cancer using three-dimensional computer simulation*, Prostate Cancer and Prostatic Diseases, 5/2002, 204-208, Nature Publishing Group, 2002.

[6] N. Popper, G. Zauner, V. Bodmann: *Time behaviour of prostata tumor markers: From modelling to a web tool for physicians*, Proceedings of the 6th EUROSIM Congress on Modelling ans Simulation, Vol. 2, ARGESIM, 2007.

[7] H.J. Stricker, L.J. Ruddock, J. Wan, W.D. Beleville: *Detection of non-palpable cancer: A mathematical and laboratory model*, British Journal of Urology, Vol. 71-43, Blackwell Science, 1993.

[8] A.R. Vashi, K.J. Wojno, B. Gillespie, J.E. Oesterling: *A Model for the number of cores per prostate biopsy based on patient age and prostate gland volume*, The Journal of Urology, Vol. 159, 920-924, American Urological Association, 1998.

Ein objektorientiertes Modell des Herz-Kreislauf-Systems mit besonderer Betrachtung körpereigener Regelkreise

A. Brunberg, D. Abel
Institut für Regelungstechnik, RWTH Aachen
R. Autschbach
Klinik für Thorax-, Herz- und Gefäßchirurgie, UK Aachen
A.Brunberg@irt.rwth-aachen.de

Zusammenfassung

Im Beitrag wird ein Simulationsmodell des menschlichen Herz-Kreislauf-Systems in der Form einer objektorientierten „Organbibliothek“ vorgestellt. Basierend auf einem strömungsmechanischen Modell des Herzens und wichtiger Gefäßsegmente werden verschiedene Mechanismen zur kurzfristigen Stabilisierung des arteriellen Blutdrucks modelliert, unter anderem der arterielle Barorezeptorreflex.

Ein solches Modell kann z. B. zur Analyse körpereigener Regelungen oder zum durchgängigen Entwurf von technischen Unterstützungssystemen (z. B. Kunstherzen) verwendet werden.

1 Einleitung

Die (ungestörte) Funktion des Herz-Kreislauf-Systems ist für das Überleben bzw. die Lebensqualität des Menschen von elementarer Bedeutung. Ist z. B. bei einer fortgeschrittenen Herzinsuffizienz eine medikamentöse Behandlung nicht mehr ausreichend und steht kein geeignetes Spenderorgan für eine Herztransplantation zur Verfügung, so muss die Kreislauffunktion durch den Einsatz eines technischen Unterstützungssystems, z. B. eines Kunstherzens (TAH) oder Ventricular Assist Devices (VAD), aufrechterhalten werden. Um mit einem solchen System auch bei wechselnden körperlichen Belastungen eine ausreichende Perfusion zu erreichen, muss die Pumpleistung an externe Parameter, z. B. Treppensteigen, hohe Umgebungstemperatur etc., angepasst werden. Eine solche Regelung erfordert jedoch eine genaue Kenntnis des zu regelnden Prozesses. Dabei ist es nicht ausreichend, lediglich die einzelnen, über Blutgefäße miteinander verbundenen Organe zu betrachten, zusätzlich müssen die Wechselwirkungen zwischen den Organen und körpereigenen Regelkreisen berücksichtigt werden.

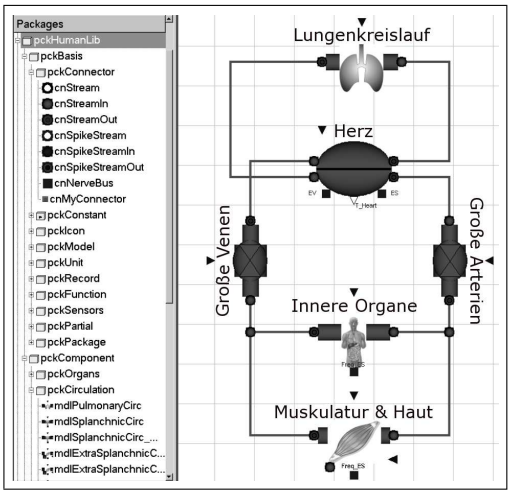


Abbildung 1: Übersicht des Kreislaufmodells in Dymola mit einem Ausschnitt aus der HumanLib.

Um aus diesem komplexen System einen einzelnen, gestörten Regelkreis herauszunehmen und durch ein technisches System zu ersetzen, bedarf es somit zunächst einer geeigneten mathematischen Beschreibung des Gesamtsystems, die in einer Simulationsumgebung implementiert werden kann. Ein solches (Simulations-)Modell ermöglicht

- ein besseres Verständnis der Interaktionen physiologischer Regelungen
- den Entwurf einer Regelung eines technischen Unterstützungssystems
- den durchgängigen Entwicklungsprozess des Unterstützungsgeräts, z. B. durch Hardware-in-the-Loop-Simulationen.

In diesem Beitrag wird ein objektorientiertes Modell in Form einer Bibliothek in Modelica/Dymola vorgestellt, das das Herz-Kreislauf-System des Menschen abbildet. Dabei wird insbesondere auf die Modellierung physiologischer Regelkreise und ihrer Interaktionen Wert gelegt. Anhand einiger beispielhafter Simulationen werden erste Ergebnisse präsentiert.

2 Modellbildung

2.1 Stand der Forschung

In der Literatur sind verschiedenste Modelle des Kreislaufsystems beschrieben, die das gesamte Spektrum von einer Abbildung auf Zellebene bis hin zur Untersuchung langfristiger Phänomene umfassen. Für die betrachtete Problemstellung ist jedoch eine Modellierung der wesentlichen Organfunktionen und -interaktionen in einem Zeitraum von Sekunden bis hin zu einigen Stunden zunächst ausreichend.

In diesem zeitlichen Bereich werden in der Literatur verschiedene strömungsmechanische Modelle, die insbesondere die Ausbreitung von Druck- und Flusspulswellen in den Gefäßen sehr gut abbilden, beschrieben [1, 2]. Diese Modelle umfassen jedoch in den meisten Fällen keine Abbildung körpereigener Regelkreise, oder lediglich ein Modell kurzfristiger Phänomene.

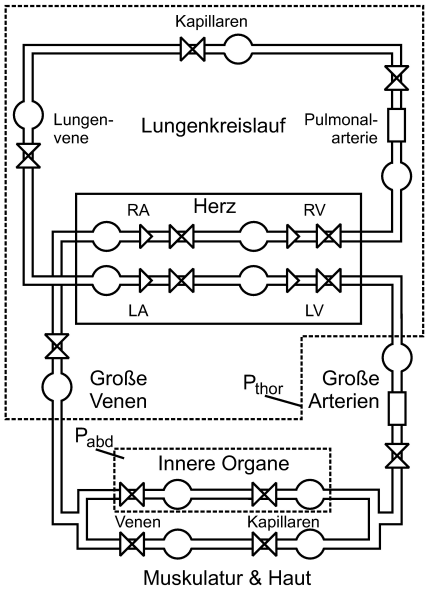


Abbildung 2: Hydraulisches Ersatzschaltbild des Kreislaufsystems: Verschaltung von hydraulischen Widerständen, Ventilen sowie Compliance- und Trägheits-Elementen.
P_{thor}: intrathorakaler Druck
P_{abd}: intraabdominaler Druck

Die sicherlich bekannteste und umfassendste Modellierung physiologischer Regelungen ist im „großen Kreislaufmodell“ von *Guyton* beschrieben [3]. Dieses Modell war zwar schon in seiner ursprünglichsten Form dazu in der Lage, die klinischen Verläufe bei bestimmten Krankheitsbildern über mehrere Wochen vorherzusagen, aber es ist auf mittel- bis langfristige Vorgänge fokussiert und nutzt ein Mittelwertmodell anstelle einer pulsatilen Modellierung des Pumpvorgangs des Herzens.

Um eine Kombination der Vorteile der beiden Modellansätze zu erhalten, können z. B. die beiden Modelle miteinander gekoppelt werden [4].

2.2 Anforderungen an eine Modellierung

Im Hinblick auf den in Kap. 1 beschriebenen Verwendungszweck des Modells bzw. der verwendeten Simulationsumgebung ergeben sich die folgenden Anforderungen an das zu erstellende Modell:

- Flexibilität, d. h. einfache und schnelle Umsetzung verschiedener Modellierungsziele
- Erweiterbarkeit
- Interdisziplinär verständliche Darstellung des Modells
- Möglichkeit, Hardware-in-the-Loop-Simulationen durchzuführen
- Lösen eines steifen Differentialgleichungssystems

Insbesondere die Forderung nach Flexibilität – z. B.im Hinblick auf simulierte Ausgangsgrößen – wird jedoch von den in Kap. 2.1 beschriebenen Modellansätzen nur schlecht umgesetzt, da sie signalorientiert programmiert sind. Dies bedeutet, dass zwischen zwei Komponenten über gerichtete Schnittstellen dimensionslose, zeitabhängige Signale übermittelt werden. Dadurch wird eine Wirkrichtung im gesamten Modell festgelegt und die verwendeten Gleichungen müssen explizit nach einer Ausgangsgröße umgeformt werden. Das bedeutet, dass zum Zeitpunkt der Modellerstellung festgelegt werden muss, welche

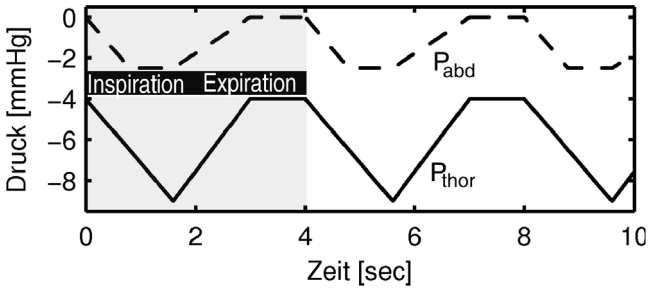


Abbildung 3: Zeitliche Verläufe des intrathorakalen (P_{thor}) und intraabdominalen (P_{abd}) Druckes unterteilt in Ein- und Ausatmungsphase bei einer Atemfrequenz von 15 Zügen/Minute (der grau hinterlegte Bereich entspricht einem Atemzug).

Größen simuliert und ausgegeben werden. Weiterhin entstehen bei direkten (d. h. unverzögerten) Rückwirkungen einer Größe auf die sie verursachende Größe algebraische Schleifen, die i. A. bei der Simulation numerische Probleme hervorrufen.

Im Gegensatz dazu können die genannten Anforderungen an das Modell mit einem Ansatz als objektorientiert programmierte Komponenten- bzw. „Organbibliothek“ besser erfüllt werden. Dieser Modellansatz hat sich bei anderen technischen Fragestellungen bereits bewährt [5].

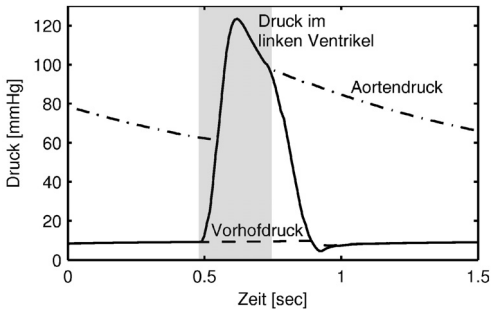
2.3 Objektorientierte „Organbibliothek“

In der Modellierungs- und Simulationsumgebung Modelica/Dymola wird eine offen erweiterbare Bibliothek, bestehend zunächst aus einfachen Modellen für Herz und Gefäßsegmente, aufgebaut. Ein besonderer Fokus wird dabei auf die körpereigenen Regelkreise gesetzt, so dass als weitere Komponenten z. B. afferente und efferente Nervenpfade sowie Teile des Zentralen Nervensystems in die Bibliothek integriert werden. Aus diesen Modulen lässt sich schnell und flexibel ein Simulationsmodell zusammenstellen. Gleichzeitig ist es möglich, einzelne Teile vor ihrer Integration ins Gesamtmodell separat zu testen. Ausgehend von diesen Grundkomponenten ist es leicht möglich, die Bibliothek um weitere Organe (z. B. Lunge, Niere) und/oder Regulationsmechanismen (z. B. durch hormonelle Veränderungen, Einflüsse der Blutgase etc.) zu ergänzen. Die Basis für die „Organbibliothek“ bilden häufig verwendete Grundelemente, deren abstrahierte Eigenschaften in Basisklassen definiert sind und an andere Komponenten vererbt werden.

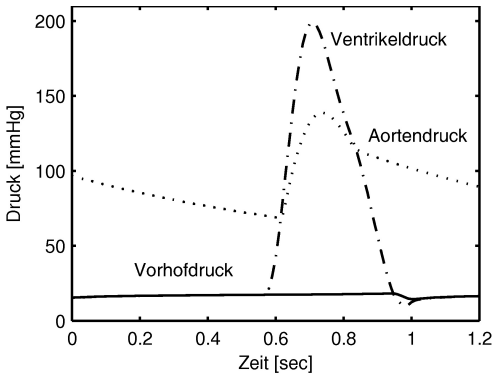
Neben den physiologischen Komponenten werden in die in Abb. 1 dargestellte Bibliothek „HumanLib“ Modelle beispielhafter pathologischer Veränderungen sowie Modelle technischer Unterstützungssysteme integriert. Dabei sollen verschiedene Modelle eines bestimmten Organs bzw. Organsystems, die unterschiedlich detailliert aufgebaut sind, in der Bibliothek zusammengefasst werden. So ist es dem Nutzer möglich, seinen Modellierungszielen entsprechend Komponenten des gewünschten Komplexitätsgrades miteinander zu kombinieren [6].

2.4 Hydraulisches Modell des Gefäßsystems

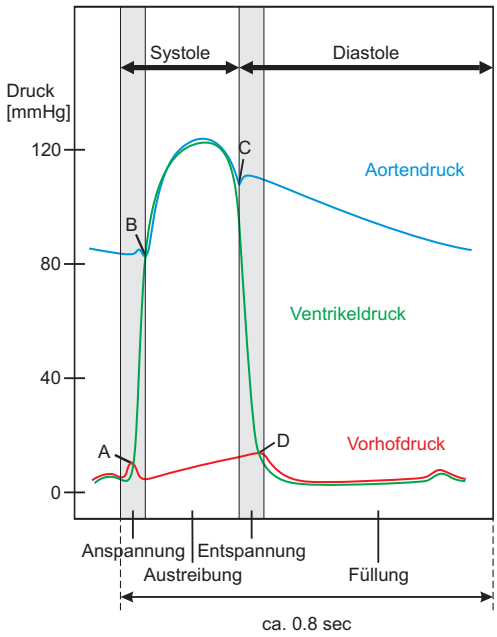
Zunächst wird ein fluiddynamisches Modell des Blutkreislaufs aufgebaut (s. Abb. 2). Dabei wird basierend auf [7] das Gefäßsystem in 5 Segmente (Aorta und große Arterien, innere Organe, Muskulatur und Haut, große Venen, Lungenkreislauf) unterteilt, die aus



(a) Simulation der zeitlichen Verläufe des Druckes im linken Atrium und Ventrikel und in der Aorta während eines Herzschlags. Die Dauer der Systole ist grau hinterlegt.



(b) Herzzyklus bei Aortenklappenstenose (durch Erhöhung des hydraulischen Widerstandes).



(c) Herzzyklus
A: Schluss der Mitralklappe
B: Öffnung der Aortenklappe
C: Schluss der Aortenklappe (Inzisur sichtbar)
D: Öffnung der Mitralklappe

Abbildung 4: Simulationen des Drucks im linken Herzen.

konzentrierten Elementen aufgebaut sind. Die Dynamik der Gefäßsegmente wird durch resistive und kapazitive Terme (*Hydraulischer Widerstand* und *Compliance*) beschrieben. In den großen Gefäßen in Herznähe (Aorta bzw. Pulmonalarterie) wird zusätzlich die *Trägheit* der in jedem Herzschlag zu beschleunigenden Blutmasse berücksichtigt. In den weiter vom Herzen entfernten Gefäßen ist dieser Einfluss vernachlässigbar.

Im kapazitiven Term werden die dehnungsabhängigen Änderungen von Gefäßdurchmesser und -länge in Abhängigkeit zum transmuralen Druck (P_{tm}) berechnet. P_{tm} ist die Differenz zwischen dem Druck im Gefäß und in der Umgebung. Der Umgebungsdruck ist entsprechend Abb. 2 P_{thor}, P_{abd} oder 0. Der Druck in Thorax bzw. Abdomen wird durch die Atemmechanik beeinflusst (s. Abb. 3), in den anderen Gefäßsegmenten ist eine Vernachlässigung des extravaskulären Druckes zulässig, da davon ausgegangen wird, dass sich der Mensch in Ruhe befindet. Andernfalls müsste der durch Muskelkontraktion verursachte Gewebedruck berücksichtigt werden.

Das Modell des Blutkreislaufs wird mit einem pulsatilen Herzmodell gekoppelt, in dem die Kontraktilität durch eine periodische Änderung des Verhältnisses von Druck zu Volumen im Ventrikel (*Elastanz*) hervorgerufen wird [2, 8, 9, 10, 11]. Der Einfluß der Vorhofsystole wird somit vernachlässigt. Die Herzklappen werden durch ideale Rückschlagventile abgebildet.

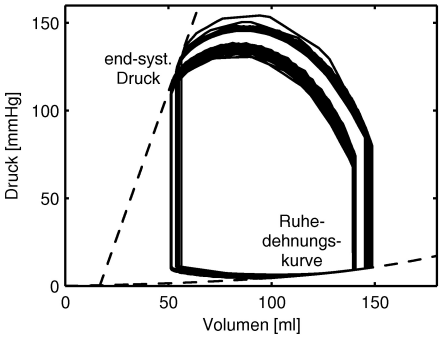


Abbildung 5: Druck-Volumen-Diagramm des linken Ventrikels bei Druckbelastung: Nach kurzer Übergangsphase annähernd gleiches Schlagvolumen.
Gestrichelt: End-syst. Druck und Ruhedehnungskurve.

3 Simulation

3.1 Druck- und Volumenverläufe im Herzen

Um das Modell des Herzens im Zusammenspiel mit der Kreislaufmodellierung zu testen, wurden zunächst die zeitlichen Verläufe der Drücke im linken Herzen während eines Herzschlags simuliert (Abb. 4a). Es zeigt sich im Vergleich mit den physiologischen Daten (Abb. 4c), dass die relevanten physiologischen Effekte gut abgebildet werden – nur die (nicht modellierte) Vorhofsystole und die normalerweise beim Schluss der Aortenklappe sichtbare Inzisur sind nicht sichtbar.

Abb. 4b zeigt eine Simulation der selben Größen für eine Verengung der Aortenklappe, die durch Erhöhung des Durchflußwiderstandes modelliert wurde. Es sind ein deutlich erhöhter Ventrikeldruck und ein großer Druckgradient über die Aortenklappe zu erkennen.

In Abb. 5 ist ein Druck-Volumen-Diagramm des linken Ventrikels dargestellt. Hier wird die Anpassung des Schlagvolumens an eine akute Druckbelastung (Nachlast, Erhöhung des Druckes in der Aorta) mittels des Frank-Starling-Mechanismus gezeigt. Im Arbeitsdiagramm wird die Kurve auf der Ruhedehnungskurve nach rechts verschoben, gleichzeitig steigt das end-systolische Volumen an, so dass nach kurzer Übergangszeit ein nahezu konstanten Volumen gefördert wird.

3.2 Blutdruckstabilisierung durch den Barorezeptorreflex

Kurzfristige, d. h. im Sekundenbereich stattfindende Änderungen des arteriellen Blutdrucks werden hauptsächlich durch den Barorezeptorreflex reguliert. Solche Störungen können z. B. durch Blutverlust oder durch Blutverschiebungen beim schnellen Aufstehen aus dem Liegen hervorgerufen werden. Im Hochdrucksystem wird die Druckänderung hauptsächlich über Dehnungssensoren, die im Aortenbogen und im Carotissinus liegen, festgestellt. Über den arteriellen Barorezeptorreflex wird Einfluss genommen auf Herzfrequenz und -kontraktionskraft, den peripheren Gefäßwiderstand sowie das ungedehnte Volumen der peripheren venösen Gefäße [2]. Der Regelkreis ist in Abb. 6 schematisch dargestellt.

Weitere Rezeptoren befinden sich im Niederdrucksystem in den großen Venen und den Vorhöfen. Im Modell wird als Messort die Lungenvene verwendet. Über den Cardiopulmonalen Barorezeptorreflex erfolgt eine Wirkung auf Herzfrequenz und -kontraktionskraft

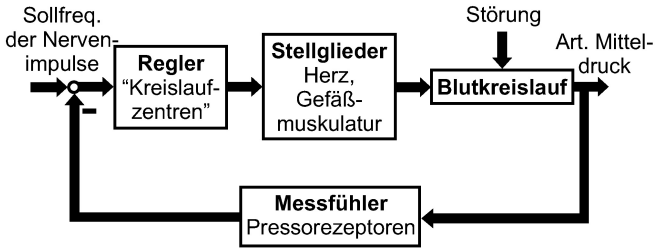


Abbildung 6: Schematische Darstellung des Barorezeptorreflexes als Regelkreis. Als Störungen können z. B. Blutverlust oder Lagewechsel wirken.

und auf den peripheren Gefäßwiderstand [7].

Als Funktionstest dieser Regulationsmechanismen wurden die Auswirkungen eines Verlusts von Blutvolumen simuliert (Abb. 7). Es ist zu erkennen, dass als Reaktion auf den Blutverlust der arterielle Druck zunächst sinkt. Durch den Barorezeptorreflex wird daraufhin die Herzleistung über die Frequenz und die Schlagkraft erhöht und gleichzeitig der Gefäßtonus in den peripheren Gefäßen vergrößert. Dadurch kann der Blutdruck stabilisiert werden.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein neuer Ansatz zur Modellierung des menschlichen Herz-Kreislauf-Systems und der damit verbundenen physiologischen Regelungen als objektorientierte Modellbibliothek vorgestellt. Aus den Bibliothekskomponenten kann einfach und komfortabel ein Modell zusammengestellt werden, mit dem z. B. die Reaktion des Körpers auf Blutverlust simuliert werden kann. Es zeigt sich, dass schon mit einer relativ einfachen Modellierung die relevanten physiologischen Effekte gut abgebildet werden.

Ein solches Modell kann nicht nur zur Analyse physiologischer Regelkreise oder zum Entwurf von Unterstützungsgeräten eingesetzt werden, sondern auch ein Einsatz in der medizinischen Ausbildung ist denkbar, denn die an den physiologischen Strukturen orientierte Modellierung und Darstellung ist für Menschen ohne ausgeprägt technische Sichtweise besser verständlich als z. B. ein Wirkungsplan.

Literatur

[1] Avolio, A. P.: *Multi-branched model of the human arterial system*, Med. & Biol. Eng. & Comput. 18 (1980) Nr. 6, S. 709-718.

[2] Ursino, M.: *Interaction between carotid baroregulation and the pulsating heart: a mathematical model*, Am. J. Physiol. 275 (1998) Nr. 5, S. H1733-H1747.

[3] Guyton, A. C., Coleman, T. G., Granger, H. J.: *Circulation: overall regulation*, Ann. Rev. Physiol. (1972) Vol. 34, S. 13-44.

[4] Werner, J., Böhringer, D., Hexamer, M.: *Simulation and prediction of cardiotherapeutical phenomena from a pulsatile model coupled to the Guyton circulatory model*, IEEE Trans. Biomed. Eng. 49 (2002) Nr. 5, S. 430-439.

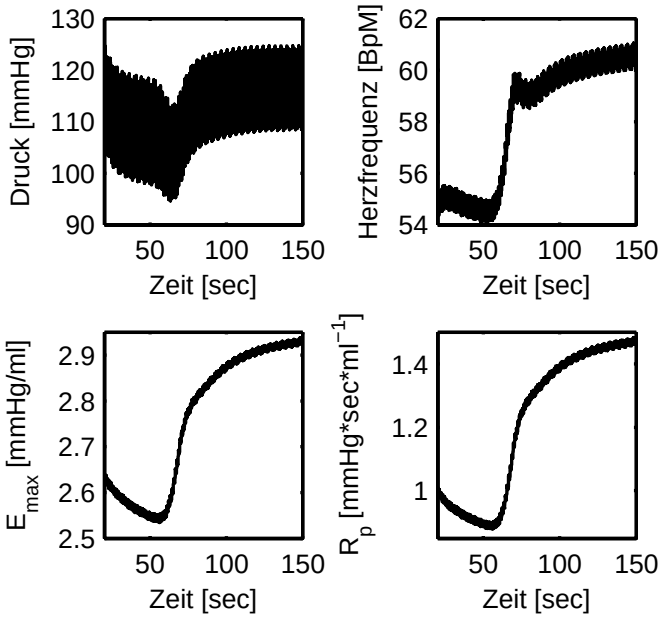


Abbildung 7: Darstellung der zeitlichen Verläufe von Blutdruck in den großen Arterien, Herzfrequenz, Ventrikelelastanz zum Zeitpunkt maximaler Kontraktion und dem peripherem Gefäßwiderstand in Folge eines Verlustes von Blutvolumen (10 % des Gesamtvolumens von 5300 ml) zum Zeitpunkt $t = 50$ sec bei Verwendung eines Modells mit arteriellen und cardiopulmonalen Barorezeptoren.

[5] Nötges, T., Hölemann, S., Bayer Botero, N., Abel, D.: *Objektorientierte Modellierung, Simulation und Regelung dynamischer Systeme am Beispiel eines Oxyfuel-Kraftwerksprozesses*, at-Automatisierungstechnik 55 (2007) Nr. 5, S. 236-243.

[6] Brunberg, A., Stützle, T., Abel, D., Autschbach, R.: *Objektorientierte Modellierung des Herz-Kreislaufsystems*, Automatisierungstechnische Verfahren für die Medizin, Automed 2007, VDI Berichte Reihe 17 Nr. 267, Düsseldorf, S. 55-56.

[7] Magosso, E., Biavati, V., Ursino, M.: *Role of the Baroreflex in Cardiovascular Instability: A Modeling Study.*, Cardiovascular Engineering 1 (2001) Nr. 2, S. 101-115.

[8] Suga, H., Sagawa, K.: *Instantaneous pressure-volume relationships and their ratio in the excised, supported canine left ventricle*, Circ. Res. 35 (1974) Nr. 1, S. 117-126.

[9] Gaasch, W. H., Cole, J. S., Quinones, M. A., Alexander, J. K.: *Dynamic determinants of letf ventricular diastolic pressure-volume relations in man*, Circulation 51 (1975) Nr. 2, S. 317-323.

[10] Piene, H.: *Impedance matching between ventricle and load*, Ann. Biomed. Eng. 12 (1984) Nr. 2, S. 191-207.

[11] Hunter, W. C., Janicki, J. S., Weber, K. T., Noordergraaf, A.: *Systolic mechanical properties of the left ventricle. Effects of volume and contractile state.*, Circ. Res. 52 (1983) Nr. 3, S. 319-327.

Anwendung des Guyton-Modells bei der Regelung der tiefen Hypotension

O. Simanski, A. Schubert, Institut für Automatisierungstechnik,
Universität Rostock, Olaf.Simanski@uni-rostock.de
Ch. N. Nguyen, Cantho University Vietnam

Zusammenfassung

Das Guyton-Modell beschreibt die komplexe, nichtlineare, körpereigene Autoregulation des arteriellen Blutdrucks. Darauf basierend wurde eine verbesserte Modellstruktur entwickelt, mit deren Hilfe die Parameter zur Regelung der tiefen Hypotension mit Sodium-Nitroprussid online eingestellt werden können oder das Patientenverhalten während der Anästhesie untersucht werden kann.

1 Einleitung

Von Hypotension wird gesprochen, wenn der arterielle Blutdruck unterhalb des normotensiven Bereiches auf Werte zwischen 90-40 mmHg abgesenkt werden soll. Diese Therapie wird z.B. intraoperativ zur Verringerung der Blutungen im Operationsgebiet oder auf der Intensivstation zur Unterstützung der Wundheilung besonders nach Herzoperationen angewendet. Die Blutdruckabsenkung erfolgt medikamentös mit Sodium-Nitroprussid (SNP). Für den Kreislauf kann der Wunsch auf Blutdruckabsenkung als Störung interpretiert werden und eine Reihe von Autoregulationsmechanismen werden je nach Dauer und Amplitude der Absenkung aktiviert, um zum individuellen Normaldruck zurückzukehren. Das komplexe Kreislauf-Modell von Guyton et al. [1] beschreibt diese Blutdruckregulation. Zur Entwicklung einer Blutdruckregelung für die tiefe Hypotension wurde das Guyton-Modell unter MATLAB/ Simulink für die Kreislaufreaktion bei der Applikation von SNP modifiziert.

2 Modifiziertes Guyton-Modell

Um die Wirkung von SNP auf den Kreislauf zu modellieren, wurden zunächst im Tierexperiment am Hausschwein die Änderungen des invasiv gemessenen mittleren arteriellen Blutdrucks (ΔMAP) bei einer konstanten Rate von SNP aufgezeichnet. Abbildung 1 zeigt das Ergebnis dieser Untersuchungen.

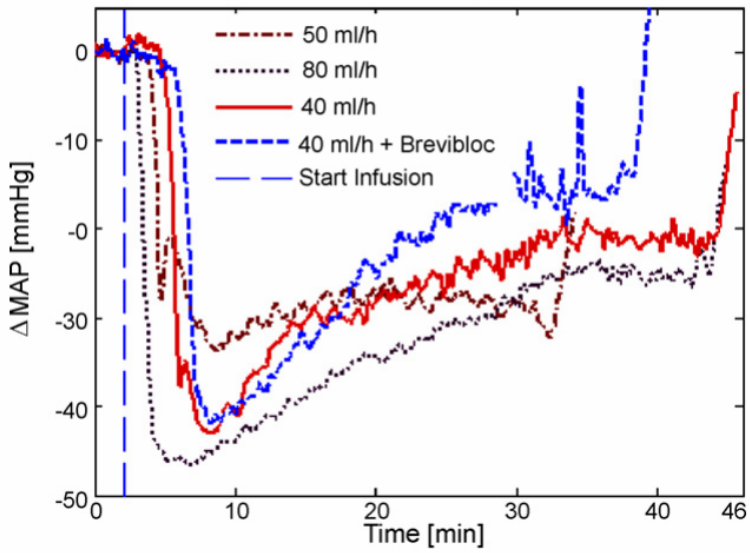


Abbildung 1: ΔMAP als Reaktion auf unterschiedliche Infusionsraten Sodium-Nitroprussid (SNP)

Die Auswirkungen der Autoregulation auf den Verlauf der Blutdruckänderung ist in Abbildung 1 zur erkennen. Trotz gleichbleibender Infusion, schafft es die Autoregulation den Blutdruck wieder anzuheben. Die Arbeit beschreibt die Modellierung des Effektes von Sodium-Nitroprussid auf die Zirkulation. Das bekannte Guyton-Modell wurde um entsprechende Module erweitert, mit dem Ziel, dass so entstandene Simulationsmodell für den Reglerentwurf oder die online- Parameterbeobachtung einzusetzen. Die Ergebnisse der Modellerweiterung und der Reglerentwicklung wurden erfolgreich in [2] und [3] publiziert.

Literatur

[1] Guyton, A.C., Coleman, T.G. , Granger, H.J.: Circulation: overall regulation, Annu. Rev. Physiol. 4 (1972) 13–46.

[2] Nguyen, Ch.N., Simanski,O., Kähler, R., Schubert, A., Janda, M., Bajorat, Hofmockel, R., Lampe, B.: Regelung des mittleren arteriellen Blutdrucks im Rahmen einer kontrollierten Hypotension, at-Automatisierungstechnik, 53 (2005) 12, 573-580.

[3] Nguyen, Ch.N., Simanski, O., Kähler, R., Schubert, A., Janda, M., Bajorat, , Lampe, B.: The benefits of using Guyton's model in an hypotensive control system, Computer Methosd and Programs in Biomedicine, 89 (2008) 2, 153-161.

Objektorientierte Modellbildung des partiellen CO_2- Gehalts bei beatmeten Patienten in Dymola

Stefanie Heinke, Marian Walter, Steffen Leonhardt
Lehrstuhl für Medizinische Informationstechnik, RWTH Aachen
heinke@hia.rwth-aachen.de, leonhardt@hia.rwth-aachen.de,
walter@hia.rwth-aachen.de
Anja Brunberg,
Institut für Regelungstechnik, RWTH Aachen
A.Brunberg@irt.rwth-aachen.de

Zusammenfassung

Vorgestellt wird ein Modell, welches den CO_2- Gehalt und den CO_2- Transport im Körper beatmeter Patienten nachbildet. Umgesetzt ist das Modell in der objektorientierten Simulationssprache Dymola. Modellbasis sind das arterielle und das venöse Kompartiment sowie der Gewebe-Flüssigkeitsspeicher. Die produzierte CO_2- Menge der Körperzellen wird dem Körper über das Gewebe zugeführt, die Elimination von CO_2 erfolgt über die Atmung. Verschiedene Annahmen, wie eine konstante CO_2- Produktion der Körperzellen, werden getroffen. Im Rahmen von Tierversuchen wurden invasiv die Blutpartialdrücke venös und arteriell gemessen. Nichtinvasiv erfolgte die Bestimmung der CO_2- Konzentration im Atemsignal. Die Messergebnisse sollen der Verifikation des Modells dienen.

Im Kreislauf des menschlichen Körpers wird mit dem Blut der Transport von Nahrungsstoffen, Elektrolyten, O_2, CO_2, etc., realisiert. Im großen (systemischen) Kreislauf erfolgt in den Organen die Metabolisierung von O_2 in CO_2. Der Gasaustausch d.h. die Abgabe von CO_2, die Aufnahme von O_2 und die Anreicherung des Blutes mit O_2 vollzieht sich über die Lunge. Die Atmung ist damit eine wichtige Komponente bei der Regulation des Säure-Basen-Haushaltes. Zielgrößen der natürlichen Atemregulation sind der Sauerstoff- und der Kohlendioxidpartialdruck im arteriellen Blut. [1] Gleiches gilt bei beatmeten Patienten, allerdings müssen hier die Sauerstoff- und Kohlendioxidpartialdrücke kontinuierlich mit einem invasiven Katheter gemessen werden. Dies birgt verschiedene Nachteile wie Belastung für den Patienten, Kosten usw. Hier wurde in Dymola ein Modell umgesetzt, bei dem basierend auf der Messung der CO_2- Konzentration im exspiratorischen Atemfluss eine Berechnung der Sauerstoff- und Kohlendioxidpartialdrücke erfolgt. Bei Bestimmung

[1] Der ph-Wert des arteriellen Blutes beträgt im Normalfall 7,63-7,44 und spiegelt die H+ Ionenkonzentration der Körperflüssigkeiten wider.

der CO_2- Verteilung im menschlichen Körper, ist es möglich den CO_2- Partialdruck zu berechnen. Annahmen für das Modell sind eine konstante Körpertemperatur und eine konstante Produktion von CO_2 im Gewebe. Für die Blutmengen (Gesamtvolumen sowie arterielles und venöses Volumen) und für die Durchblutungsgeschwindigkeit werden verschiedene Approximationen eingeführt.
Das Modell besteht aus einem arteriellen und venösen Kompartiment und dem Gewebe-Flüssigkeitsspeicher. Der Gewebe-Flüssigkeitsspeicher unterscheidet sich im Wesentlichen durch den konstanten Zufluss aufgrund der CO_2- Produktion des Gewebes. Die Gleichungen für das arterielle, das venöse Kompartiment und das Gewebe weisen die gleiche Form auf und unterscheiden sich nur durch die Werte der einzelnen Parameter. Daher konnten die drei Kompartimente in Dymola mit einem „model compartment“ umgesetzt werden.

$$mCO_2(t) = \int [\dot{m}_{Zufl}(t - T_{Zufl}) - \dot{m}_{Abfl}(t)]dt + mCO_{2,0} \tag{1}$$

$$cCO_2(t) = mCO_2(t)/V_{Blut} \tag{2}$$

$$\dot{m}_{Abfl}(t) = cCO_2(t) \cdot \dot{V}_{Blut} \tag{3}$$

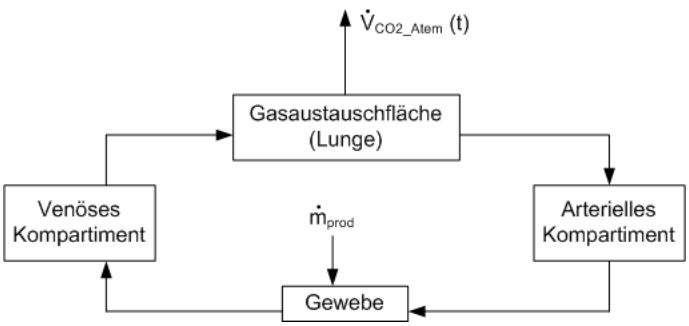


Abbildung 1: Modell CO2- Transport

Gleichung 1 gibt die Massenbilanz an, mit Gleichung 2 erfolgt die Berechnung der CO_2-Konzentration in Abhängigkeit von der Blutmenge des jeweiligen Kompartimentes und mit der dritten Gleichung kann der Abfluss an CO_2 (mmol/min) aus einem Kompartiment bestimmt werden. Mit Hilfe der Konzentration von CO_2 können über die linearisierte Dissoziationskurve die Partialdrücke errechnet werden.
Das Modell ist in Abbildung 1 schematisch dargestellt.

Literatur

[1] Larsen,R., Ziegenfuß, T.: *Beatmung*, 3., vollst. überarbeitete und erweiterte Auflage, Springer Verlag, 2004.

[2] Silbernagl, S., Despopoulos, S.: *Taschenatlas Physiologie*, 7., vollst. überarbeitete und erweiterte Auflage, Thieme Verlag, 2006.

[3] Fritzson, P.: *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, Wiley-IEEE Press, 2004.

Implementierung eines eindimensionalen Blutflussmodells mittels Finiter Elemente Methode

Bernhard Hametner, Institut für Analysis und Scientific Computing, TU Wien
bhametner@osiris.tuwien.ac.at
Johannes Kropf, Biomedical Engineering, Austrian Research Centers
johannes.kropf@arcsmed.at
Xenia Descovich, Institut für Analysis und Scientific Computing, TU Wien
xdescovich@osiris.tuwien.ac.at
Siegfried Wassertheurer, Biomedical Engineering, Austrian Research Centers
siegfried.wassertheuer@arcsmed.at
Felix Breitenecker, Institut für Analysis und Scientific Computing, TU Wien
felix.breitenecker@tuwien.ac.at

Zusammenfassung

Aufgrund der weiten Verbreitung von Herzkreislauferkrankungen ist die Simulation des Herzkreislaufsystems von großem Interesse. Diese Arbeit beschäftigt sich mit der Herleitung und Simulation eines Blutflussmodells für Arterien. Ausgehend von den Navier-Stokes-Gleichungen wird ein eindimensionales partielles Differentialgleichungssystem mit zwei Zustandsgrößen hergeleitet. Zur numerischen Lösung wird die Finite Elemente Methode, genauer ein Taylor-Galerkin Verfahren zweiter Ordnung verwendet.
Die Lösung des Systems erfolgt mit Hilfe von Matlab. Um die Implementierung zu verifizieren, werden Experimente mit verschiedenen Testeingängen und unterschiedlichen Arterienstücken durchgeführt.
Mit dem Modell kann der Blutfluss in den großen Arterien beschrieben werden, und wird in Folge in ein dynamisches, geregeltes und identifizierbares Modell des menschlichen Herzkreislaufsystems integriert. Damit lassen sich die Einflüsse physiologischer Veränderungen des Gefäßsystems auf das globale Verhalten des Blutkreislaufs simulieren und untersuchen.

1 Einleitung

In vielen hochentwickelten Ländern stellen Erkrankungen des Herzkreislaufsystems die häufigste Todesursache dar. Daher ist die Forschungsarbeit auf diesem Gebiet von großer Bedeutung. Die Modellbildung und Simulation des Herzkreislaufsystems nimmt dabei eine wichtige Rolle ein. Mit Hilfe von Simulationen können die Auswirkungen von Operationen bereits vor dem eigentlichen Eingriff untersucht werden. Eine Simulation bietet weiters die Möglichkeit, Auswirkungen von Medikamenten zu untersuchen, indirekte Messverfahren zu entwickeln oder Messdaten virtuell für weitere Berechnungen zu generieren.

Die Modellbildung und Simulation des menschlichen Herzkreislaufsystems befindet sich in einem Spannungsfeld. Einerseits existieren relativ leicht handhabbare Modelle des ganzen Blutkreislaufs, die allerdings nicht alle in der Wirklichkeit auftretenden Phänomene abbilden. Andererseits gibt es sehr komplexe Modelle, die jedoch nur einen Teilbereich des Kreislaufsystems betrachten.

Diese Arbeit beschäftigt sich mit dem Blutfluss in den großen Arterien in einer Raumdimension. Das hier entwickelte Modell soll als Teil eines geregelten Modells für den gesamten Herzkreislauf verwendet werden.

In den folgenden Abschnitten wird zuerst ein Überblick über die Modellherleitung geboten. Danach wird darauf eingegangen, wie die Modellgleichungen diskretisiert werden, um eine numerische Lösung mittels Finiter Elemente Methode berechnen zu können. Abschließend werden die Ergebnisse einiger Experimente präsentiert.

2 Modellherleitung

Das Modell geht von den inkompressiblen Navier-Stokes-Gleichungen aus:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} + \frac{1}{\rho} \nabla P - \operatorname{div}\left[\nu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)\right] = 0 \tag{1}$$

$$\operatorname{div} \boldsymbol{u} = 0 \tag{2}$$

Als zugrunde liegender Definitionsbereich dient ein zylindrischer Bereich Ω_t, dessen Rand sich auf Grund der durch den Fluss verursachten Wandbewegung über die Zeit t verändert. (x, y, z) bezeichnet das kartesische Koordinatensystem. Der Vektor $\boldsymbol{u}$ stellt die Flussgeschwindigkeit dar, somit beschreiben seine drei Komponenten die Flussgeschwindigkeit in x-, y- und z-Richtung. Weiters ist P der Druck, ν die kinematische Viskosität und ρ die Blutdichte.

Zur Vereinfachung werden folgende Modellannahmen gemacht:

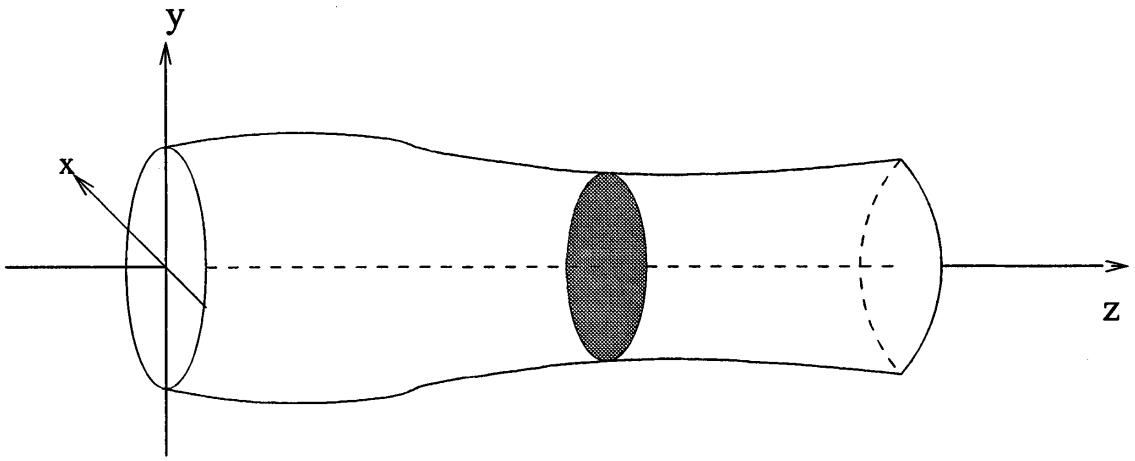


Abbildung 1: Darstellung eines Arterienstücks als zylindrisches Rohr

Axiale Symmetrie: Alle auftretenden Größen sind unabhängig vom Winkel θ. Das bedeutet u.a., dass jeder axiale Querschnitt $z = \text{const}$ über den ganzen Zeitverlauf kreisförmig bleibt. Der Radius R des Rohrs ist somit eine Funktion des Ortes z und der Zeit t.

Radiale Verschiebung: Die Arterienwand verschiebt sich nur in radialer Richtung. Sei R_0 ein Referenzradius, dann kann die Auslenkung der Gefäßwand durch $\eta = R - R_0$ beschrieben werden.

Konstanter Druck: Der Druck ist auf jedem Querschnitt konstant, d.h. er ist nur von z und t abhängig.

Keine Volumenkräfte: Volumenkräfte wie z.B. die Gravitation werden vernachlässigt.

Dominanz der axialen Geschwindigkeit: Geschwindigkeitskomponenten orthogonal zur z-Achse können vernachlässigt werden im Vergleich zu der Geschwindigkeitskomponente entlang der Hauptausbreitungsrichtung (z-Achse). Bezeichnet man diese mit u_z, so ergibt sich für die Geschwindigkeit:

$$u_z(t, r, z) = \bar{u}(t, z)\, s\left(\frac{r}{R(t, z)}\right) \tag{3}$$

Dabei ist $\bar{u}$ die mittlere Geschwindigkeit auf jedem axialen Querschnitt und die Funktion $s : \mathbb{R} \to \mathbb{R}$ beschreibt ein Geschwindigkeitsprofil.

Um nun von einem dreidimensionalen System (bezüglich der Ortskoordinaten) zu einem eindimensionalen System zu kommen, wird über eine allgemeine Querschnittsfläche $S(z, t)$ integriert.

Das daraus resultierende System besteht aus zwei Gleichungen, beinhaltet aber drei Zustandsgrößen, nämlich die Querschnittsfläche A, den mittleren Volumenfluss Q und den mittleren Druck P. Um das System lösen zu können, braucht man daher eine dritte Zustandsgleichung. Diese erhält man aus einem mechanischen Modell für die Verschiebung der Gefäßwand (siehe [3]). Der Zusammenhang zwischen Druck und Fläche wird mit folgender algebraischen Gleichung beschrieben, die in der Literatur häufig zu finden ist (siehe z.B. [1]):

$$P - P_{ext} = \beta \frac{\sqrt{A} - \sqrt{A_0}}{A_0} \tag{4}$$

Dabei ist β ein Parameter, in dem der Elastizitätsmodul E und die Arterienwandstärke h_0 linear eingehen. A_0 ist die dem Referenzradius R_0 entsprechende Querschnittsfläche.

Aufgrund der Annahmen kann das Modell auf eine Dimension reduziert werden und hat folgende Form:

$$\boldsymbol{U} = \begin{pmatrix} A \\ Q \end{pmatrix} \tag{5}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial z}(\boldsymbol{U}) = \boldsymbol{B}(\boldsymbol{U}), \quad z \in (0, L),\ t > 0 \tag{6}$$

$$\boldsymbol{F}(\boldsymbol{U}) = \begin{pmatrix} Q \\ \alpha \frac{Q^2}{A} + \frac{\beta}{3\rho A_0} A^{\frac{3}{2}} \end{pmatrix} \tag{7}$$

$$\boldsymbol{B}(\boldsymbol{U}) = \begin{pmatrix} 0 \\ -K_R \frac{Q}{A} - \frac{A}{A_0 \rho}\left(\frac{2}{3}A^{\frac{1}{2}} - A_0^{\frac{1}{2}}\right)\frac{\partial \beta}{\partial z} + \frac{\beta}{\rho}\frac{A}{A_0^2}\left(\frac{2}{3}A^{\frac{1}{2}} - \frac{1}{2}A_0^{\frac{1}{2}}\right)\frac{\partial A_0}{\partial z} \end{pmatrix} \tag{8}$$

K_R ist dabei ein Widerstandsparameter und α bezeichnet den Coriolis Koeffizienten.

3 Diskretisierung

Im letzten Abschnitt wurde ein eindimensionales Blutflussmodell hergeleitet. Um diese Gleichungen zu lösen, muss ein numerisches Verfahren angewandt werden. Es stehen verschiedene Methoden zur Verfügung, so werden für Problemstellungen dieser Art oft Finite Volumen Verfahren benutzt. Hier wird eine Finite Elemente Methode, genauer ein Taylor-Galerkin Verfahren zweiter Ordnung herangezogen, siehe auch [1].

Die Zeitachse wird in Zeitschritte der Länge Δt diskretisiert. Um den Wert von $\boldsymbol{U}$ zu

einem nächsten Zeitpunkt t^{n+1} zu erhalten, verwendet man eine Taylorentwicklung zweiter Ordnung.

Der Ort wird diskretisiert, indem man das Intervall $[0, L]$ in Elemente $[z_j, z_{j+1}]$ unterteilt. Die Länge eines Elements ist somit $h_j = z_{j+1} - z_j$.
Der Raum der stückweise linearen Funktionen soll mit V_h bezeichnet werden. Da eine Diskretisierung für beide Zustandsgrößen A und Q durchgeführt werden muss, werden folgende Räume eingeführt:

$$\begin{aligned} \boldsymbol{V}_h =& V_h^2 \\ \boldsymbol{V}_h^0 =& \{\boldsymbol{v}_h \in \boldsymbol{V}_h |\ \boldsymbol{v}_h = \boldsymbol{0} \text{ für } z = 0,\ z = L\} \end{aligned}$$

Somit kann das Problem für die approximierte Lösung formuliert werden. Gesucht ist also ein $\boldsymbol{U}_h^{n+1} \in \boldsymbol{V}_h$, das für alle $\varphi_h \in \boldsymbol{V}_h^0$ die folgenden Gleichungen sowie die zugehörigen Randbedingungen erfüllt.

$$\begin{aligned} \left(\boldsymbol{U}_h^{n+1}, \varphi_h\right) = (\boldsymbol{U}_h^n, \varphi_h) + \Delta t \left(\boldsymbol{F}_{LW}(\boldsymbol{U}_h^n), \frac{d\varphi_h}{dz}\right) - \frac{\Delta t^2}{2} \left(\boldsymbol{B}_{\boldsymbol{U}}(\boldsymbol{U}_h^n) \frac{\partial \boldsymbol{F}(\boldsymbol{U}_h^n)}{\partial z}, \varphi_h\right) + \\ + \frac{\Delta t^2}{2} \left(\boldsymbol{H}(\boldsymbol{U}_h^n) \frac{\partial \boldsymbol{F}(\boldsymbol{U}_h^n)}{\partial z}, \frac{d\varphi_h}{dz}\right) + \Delta t \left(\boldsymbol{B}_{LW}(\boldsymbol{U}_h^n), \varphi_h\right) \end{aligned} \tag{9}$$

4 Implementierung und Simulationsergebnisse

Die numerische Lösung wird mit Hilfe von Matalb durchgeführt. Dabei folgt die Implementierung der Finiten Elemente Methode den Ausführungen in [4].

In den bisherigen Modellbetrachtungen im vorigen Abschnitt wurden die Randwerte vernachlässigt. Um mit Hilfe der Finiten Elemente Methode zu einer Lösung zu gelangen, ist die Vorgabe von Randwerten bei $z = 0$ und $z = L$ aber unumgänglich. Außerdem kann erst mit geeigneten Randbedingungen das betrachtete Arterienstück in den Blutkreislauf des Körpers eingebettet werden.

Da die beiden Zustandsgrößen des Blutflussmodells die Querschnittsfläche A und der mittlere Volumenfluss Q sind, müssen für diese beide Größen bei jedem Zeitschritt geeignete Werte bei $z = 0$ und $z = L$ vorgegeben werden. Allerdings hat man für die beiden Zustandsgrößen oft keine passenden Messwerte oder andere Daten, um die Randwerte an den benötigten Punkten vorgeben zu können. Dagegen ist der entsprechende Druck eine häufiger bekannte Systemgröße. Daher verwendet man die charakteristischen Variablen des Systems, um die erforderlichen Randwerte berechnen zu können.

In den folgenden Experimenten wird als proximaler Eingang jeweils eine Druckfunktion vorgegeben. Dabei wird immer eine halbe Sinusschwingung verwendet. Das betrachtete

Arterienstück hat eine Länge von 10 cm. Bei der Diskretisierung wird eine konstante Elementgröße von $h = 0.1$ cm verwendet. Der Zeitschritt Δt beträgt $1 \cdot 10^{-4}$ s.

Für das erste Experiment wird eine Druckkurve als Eingangsfunktion verwendet, deren Impulslänge 0.165 s beträgt. Das Ergebnis ist in Abbildung 2 ersichtlich, wo der Druckverlauf bei $z = 0$ cm, $z = 5$ cm und $z = 10$ cm dargestellt ist. Es ist zu erkennen, dass der Eingangsdruck nahezu unverändert durch das Arterienstück weitergeleitet wird. Dies entspricht auch den Erwartungen, weil am distalen Ende kein peripherer Druck angenommen wird. Auf diese Weise wird ein unendlich langes Rohr simuliert.

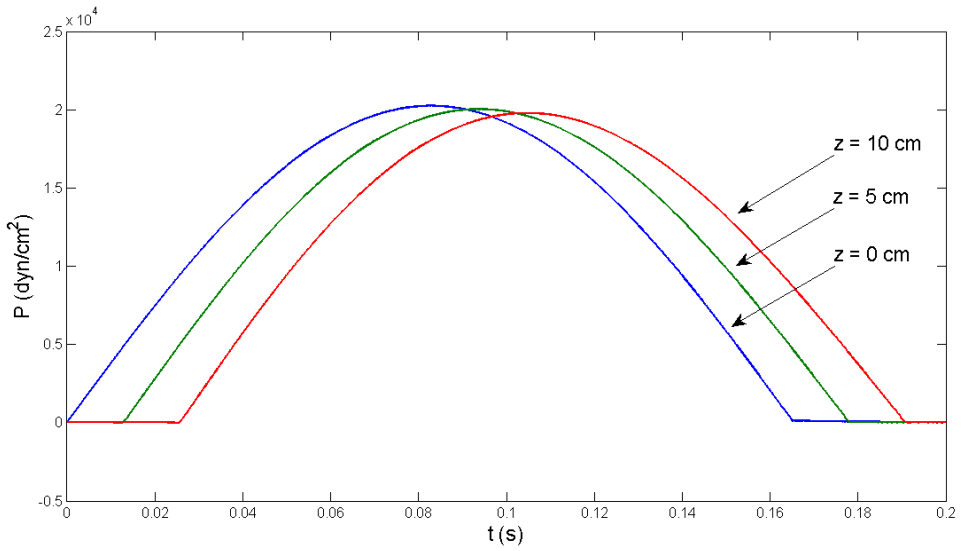


Abbildung 2: Druckverlauf an drei Punkten der Arterie

In einem zweiten Experiment wird das Verhalten bei einer Bifurkation untersucht. Dazu werden drei Arterienstücke mit einer Länge von je 10 cm bei einer Verzweigung vereinigt. Die proximale Arterie hat dabei einen Referenzradius R_0 von $0,5$ cm, die beiden weiteren Gefäße besitzen einen Anfangsradius von $0,4$ cm bzw. $0,3$ cm.

Der hier verwendete Eingangsimpuls hat eine Länge von $0,1$ s. Betrachtet man wieder den Druckverlauf in der ersten Arterie, so ist eine Druckreflexion an der Verzweigung zu erkennen. Bei $z = 0$ cm verlängert sich der Druckimpuls durch die rücklaufende Druckwelle. Bei $z = 10$ cm fällt die Reflexion noch zur Gänze in die hinlaufende Druckwelle, als Auswirkung ist ein erhöhter Druck beobachtbar, in der Mitte der Arterie ist eine Kombination dieser beiden Effekte zu sehen (siehe Abbildung 3). Abbildung 4 zeigt den Volumenfluss in der Mitte der drei Arterien ($z = 5$ cm) über die Zeit. Man erkennt, dass sich Q bei der Verzweigung in die beiden weitergehenden Gefäße aufteilt, dabei fließt mehr Blut durch die größere der zwei Arterien. In der Flusskurve der proximalen Arterie ist ein geringer negativer Fluss sichtbar. Dies ist der Rückfluss, der durch die Reflexion an der Bifurkation entsteht. Vergleicht man die Ergebnisse dieser Experimente mit den in [1] und [2] veröffentlichten Resultaten, so kann eine gute Übereinstimmung festgestellt werden.

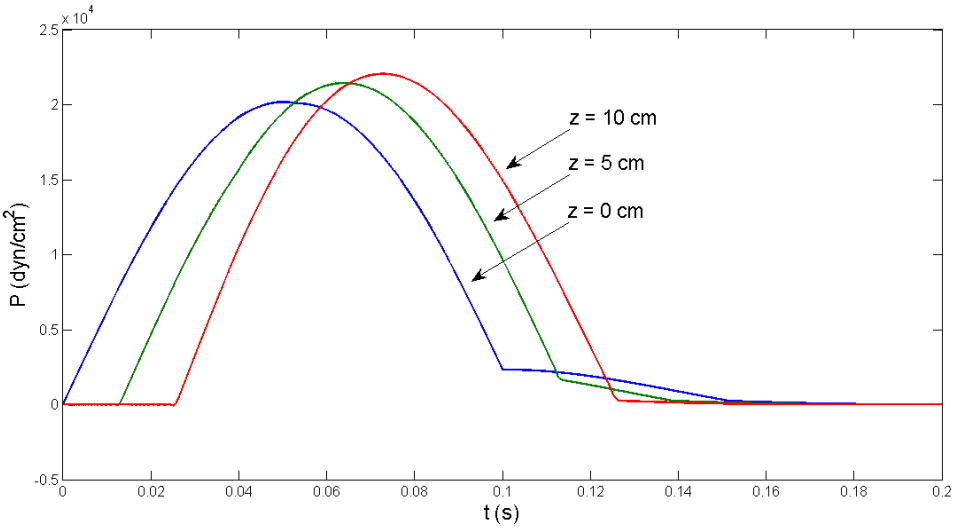


Abbildung 3: Druckverlauf an drei Punkten der ersten Arterie

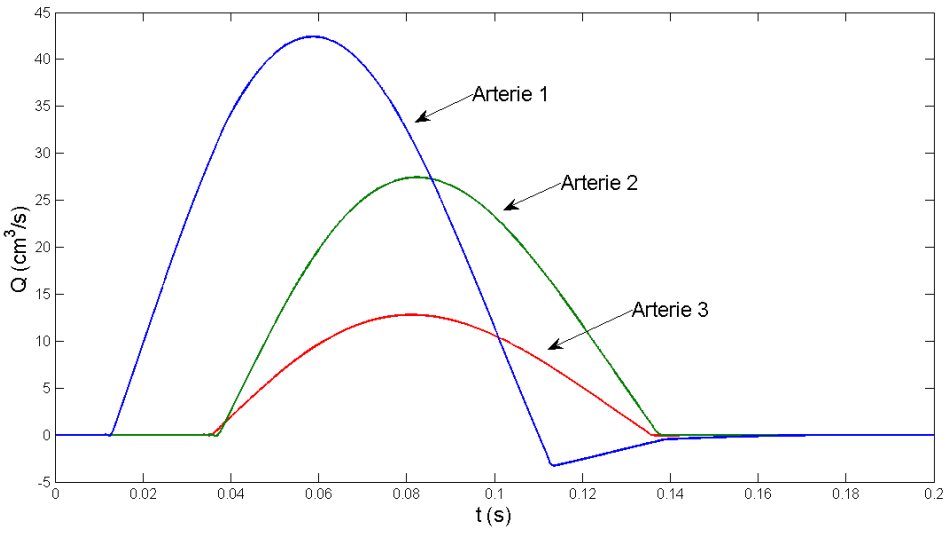


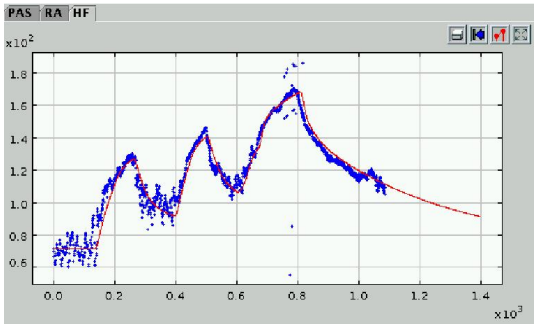
Abbildung 4: Volumenfluss an den drei Arterienstückmittelpunkten

5 Ausblick

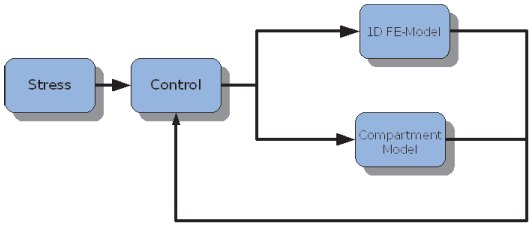
Mit dem oben beschriebenen Modell kann nun der Fluss und der Druck in einem Teil (z.B. Arm, Thorax) oder im gesamten Netzwerk der großen Arterien simuliert werden. Das Gefäßsystem wird dazu näherungsweise mit Stücken von geraden Rohrsegmenten modelliert.

In weiterer Folge soll das Netzwerkmodell mit einem Modell für die Regelung des Herzkreislaufsystems gekoppelt werden (siehe [5]). Damit können dann Einflüsse wie zum Beispiel körperliche Belastung und deren Einfluss auf die Druckverhältnisse in einzelnen Arterienabschnitten untersucht werden. Ein Simulationsschema ist in Abb. 5b dargestellt. Eine Schwierigkeit stellt dabei die Multiskalarität dar. Das geregelte Modell liefert nur

einen über eine Herzschlagperiode gemittelten Wert, wobei für das Netzwerkmodell ein transienter Verlauf der Eingangskurve (z.B. Fluss in der Aorta, siehe Abb. 5a) notwendig ist (siehe [6]).



(a) Vergleich von Simulation und Messwerten



(b) Schema einer Herzkreislaufsimulation

Abbildung 5: Schema einer Herzkreislaufsimulation

Literatur

[1] Formaggia, Luca, Lamponi, Daniele und Quarteroni, Alfio: *One-dimensional models for blood flow in arteries*, Journal of Engineering Mathematics, 47 (3-4), S. 251-276, 2003.

[2] Formaggia, Luca, Nobile, Fabio und Quarteroni, Alfio: *A one dimensional model for blood flow: Application to vascular prosthesis*, Mathematical Modeling and Numerical Simulation in Continuum Mechanics, 19, S. 137-153, 2001.

[3] Quarteroni, Alfio und Formaggia, Luca: *Mathematical modelling and numerical simulation of the cardiovascular system*, Handbook of Numerical Analysis, Special Volume: Computational Models for the Human Body, S. 3-127, Elsevier B. V., 2004.

[4] Carstensen, Carsten, Funken, Stefan und Alberty, Jochen: *Remarks around 50 lines of Matlab: Short finite element implementation*, Numerical Algorithms, 20 (2-3), S. 117-137, 1999.

[5] Kropf, Johannes: *Dynamische Herzkreislaufsimulation unter Berücksichtigung verteilter Parameter*, Diplomarbeit, Technische Universiät Wien, 2003.

[6] Kropf, Johannes: *Multiscale Blood Flow Modelling and Simulation*, Dissertation, Technische Universiät Wien, 2007.

Gesamtsystemanalyse komplexer mechatronischer Systeme

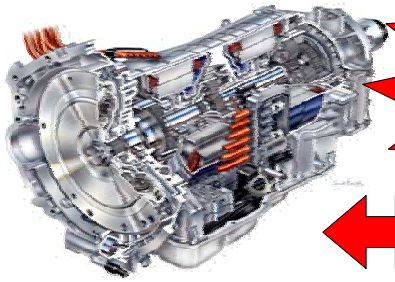
Andreas Junghanns, Jakob Mauss, Mugur Tatar

QTronic GmbH, Berlin

Asim-Workshop, Wismar, 29./30. Mai 2008

Mechatronische Systeme: Herausforderung

Multiple Disziplinen
- Software
- Mechanik
- Hydraulik
- Elektrik
- Elektronik
- ...

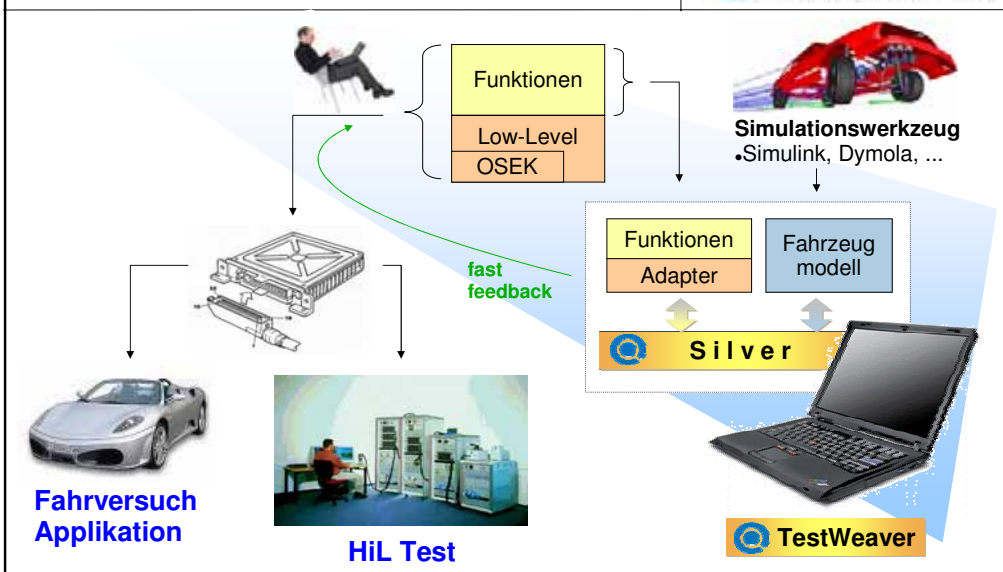


Kombinatorische Interaktion: Einzelgewerketest nicht ausreichend

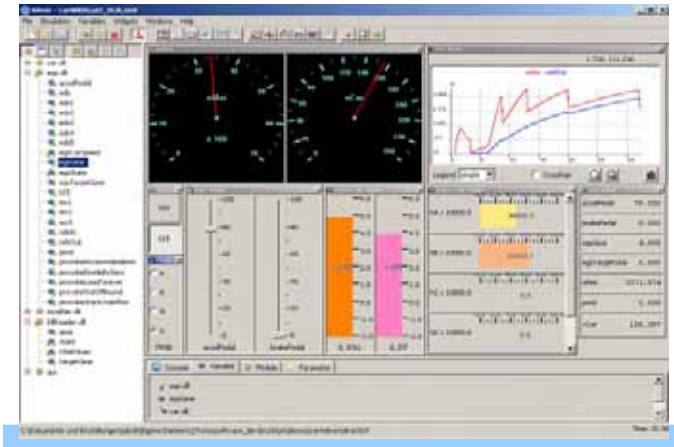
Umgebung
- Wetter
- Straße
- Fahrer
- HW-Fehler
- Toleranzen
- Alterung
- Interaktion mit anderen Systemen
- ...

Fehler passieren...
Wichtig ist nur, sie rechtzeitig zu finden

Funktionsentwicklung mit HiL und SiL



Silver



Silver
- Umgebung zum Ausführen kompilierter Module
- Moduleimport z.B. aus
  - Matlab/Simulink
  - Dymola
  - SimulationX
  - Visual C/C++
- Debug Unterstützung
- Signalfluss Visualisierung
- konfigurierbares GUI mit vielen Instrumenten
- professioneller Support durch QTronic

silver variablen

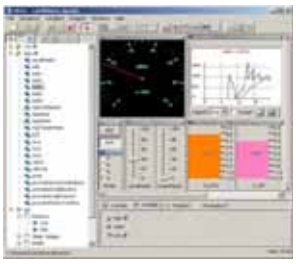
dllReader | dllWriter | a2lReader | simulink | dymola | MDFReader

Warum Co-Simulation?

QTronic SIMULATION FOR ENGINEERING

Nutzen für die Funktionsentwicklung

- komfortable Integration von Funktion und Strecke auf dem Laptop des Funktionsentwicklers
- mächtige Test- und Debuggingmöglichkeiten
- Performancegewinn (Laufzeit), weil die Module in kompilierter Form ausgeführt werden (spürbar z.B. bei grossen Simulink Modellen!)
- kurze Entwicklungszyklen
- kostengünstige Automatisierbarkeit, z.B. für Test



und ...

- **Reduzierte Abhängigkeit** von Toolherstellern bzgl. SiL/MiL: Es ist egal, wie die Module erzeugt werden (Simulink, Ascet, Dymola ...). Die Modul-erzeugenden Tools sind austauschbar, ohne dass andere Prozesse (Test, Applikation, ...) betroffen sind.
- **Schutz von IP**: Vorteile bei Zusammenarbeit mit OEMs und anderen Zulieferern: Module werden binär ausgetauscht.

Vergleich: Test mit HiL und SiL

QTronic SIMULATION FOR ENGINEERING

HiL

\+
- Test von Low-Level Funktionen: CAN, Signalverarbeitung, Laufzeiten
- Test der SG Hardware

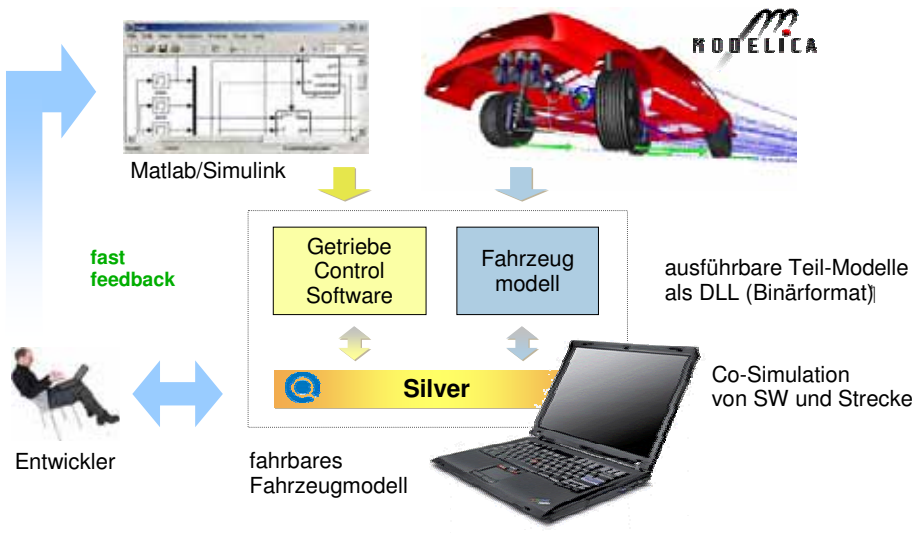
\-
- HiL ist teure und begrenzte Resource:
  - Wartezeiten beim Test
  - kein schnelles Feedback
- Streckenmodelle müssen echtzeifähig sein. Deshalb
  - hoher Aufwand für Modell Tuning oder
  - Modellvereinfachung: fehlende Effekte

SiL und MiL

\+
- keine Echzeitanforderung. Deshalb:
  - Modellierung billiger
  - Modelle dürfen schneller sein (z.B. 10x für FAS)
  - Modelle dürfen detailliert sein Anwendung
    - * Fehlersimulation
    - * Applikation
    - * Test von Funktionsqualität, Alterung, Verbrauch
- preiswert und breit verfügbar: Agilität, schnelles Feedback an Funktionsentwickler

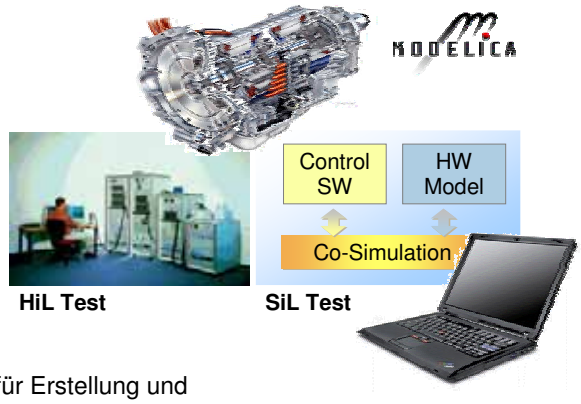
\-
- keine Nachbildung von Low-Level Funktionen. Geht nur im HiL oder Fahrversuch.

Demo: Funktionsentwicklung mit Silver



Test Automatisierung mit Testskripten

```
Test #31
start_car();
shift_lever = D;
while(gear<2)
 accelPedal = 20;
if(time>2)
 error("no shift")
...
```



Probleme

- Hohe Arbeitskosten für Erstellung und Pflege der Skripte
- Testabdeckung

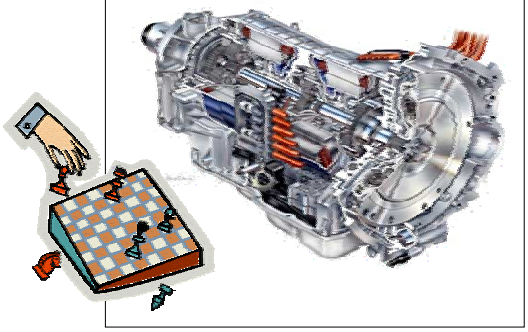
TestWeaver

QTronic SIMULATION FOR ENGINEERING

Idee
Automatisches Erzeugen
10 000er verschiedener Tests

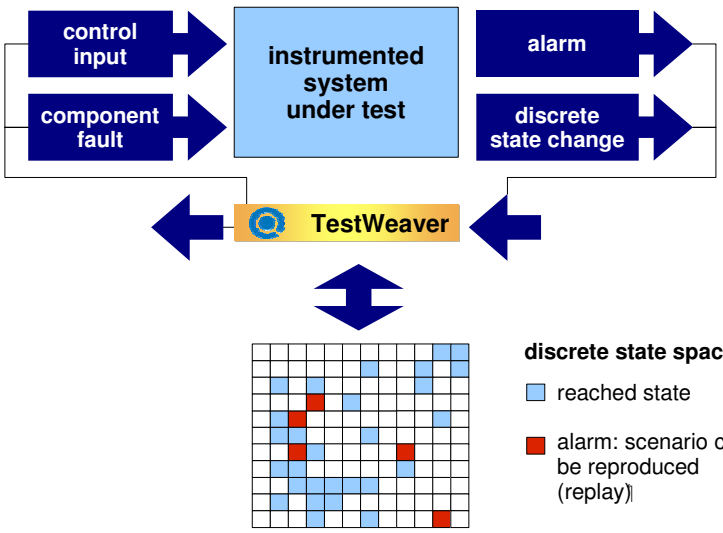
Nutzen
- hohe Abdeckung
- bei niedrigen Arbeitskosten
  kein Schreiben und Pflegen
  von Test-Skripten

Test = Spiel
gegen das (simulierte) System



Test-Fall Generator

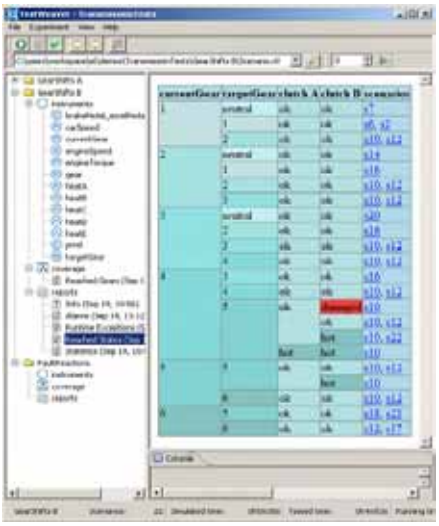
QTronic SIMULATION FOR ENGINEERING



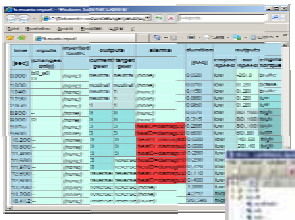
TestWeaver - Experimentauswertung

QTronic SIMULATION FOR ENGINEERING

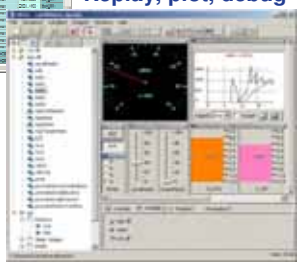
Overview report for all scenarios



Detailed reports for individual scenarios

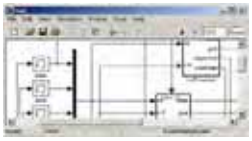


Replay, plot, debug



Demo: automatisierter Test

QTronic SIMULATION FOR ENGINEERING



Matlab/Simulink

Modell

Getriebe Control Software

Fahrzeug modell

ausführbare Teil-Modelle als DLL (Binärformat)

Silver

Co-Simulation von SW und HW

TestWeaver

Generieren, Ausführen und Bewerten von Tests

Modeling Discontinuous Elements in Mechatronic Systems by Using Regularized and Idealized Approaches

Yongbo Chen, Bosch Rexroth AG
yongbo.chen@boschrexroth.de
Dr.-Ing. Oliver Lenord, Bosch Rexroth AG
oliver.lenord@boschrexroth.de
Prof. Dr.-Ing. Dieter Schramm, Institut Mechatronik, Uni. Duisburg-Essen
dieter.schramm@uni-due.de

Abstract

D&C Engine is a software package to model and simulate mechatronic systems. It is based on the well-known object-oriented modeling method. In context of engineering applications, many mechatronic systems contain discontinuities, which result from ideal switches, ideal diodes, dry friction and impacts. To model such discontinuities, classically a regularized modeling approach is used. In this paper we present an idealized approach, as a useful alternative to the classical one. Along with an event-handling algorithm, these approaches were implemented in D&C Engine. By means of simple examples, the performance of these two approaches is subsequently compared. Thereby it is shown that the idealized approach may be advantageous for highly stiff systems.

1 Introduction

Modeling and Simulation are becoming more and more important in many industrial branches due to their potential for reducing costs and providing a better understanding of the inherent effects governing the behaviour of the systems. The modern technical systems are usually composed of components from different physical domains. Examples are mechatronic systems in automotive, aerospace and machine tool applications. Using *general* tools to deal with multi-domain models results usually in a huge amount of efforts, which are caused by manual rewriting of all the equations into an explicit form. The *domain-specific* tools, such as circuit simulator or multi-body simulation program, are not able to handle the models of other domains in an efficient way.

The object-oriented modeling paradigm for complex physical systems, introduced by Hilding Elmqvist in the late 70s [3], provides a platform to simulate systems with compo-

nents from diverse domains. *D&C Engine*, abbreviation for *Drive and Control Engine*, is a software package to model and simulate mechatronic systems. It is developed by the department of advanced engineering of the Bosch Rexroth AG, and aimed to support developments and applications in engineering of its own. It follows from the methodology of the object-oriented modeling for complex physical systems and is implemented with the programming language *C++*.

1.1 Object-oriented Modeling of Complex Physical Systems in *D&C Engine*

According to the Kirchhoff's Current Law, the currents, flowing towards and away from a point, must add up to zero, while the potentials of all branches that are connected at this point must be equal. Variables of type 'current' are called *through* variable f; Variables of type 'potential' are called *across* variable e. These two type variables (called *effort* and *flow* in bond graphs) exist also in other physical domains.

The physical components, such as a resistor, valve or mass, are described as *transmission* or *state* elements (objects) [6]. A *transmission* element has interfaces which consist of *across* and *through* variables for the respective domain. *Through* variables f_i are summarized in the *state* elements, so called *nodes*. According to physical principles (e.g. Kirchhoff's Current Law in electrics, or Newton's Law in mechanics), the summation of the *through* variables at each *node* must be equal to zero. In most cases there is an *across*-causes-*through* causality in *transmission* elements, e.g. electric resistors, hydraulic resistances, pipes and springs-damper-elements. Exceptions are ideal gears and ideal electrical transformers. However, their transfer behavior can be regularized by use of constraint equations. By using *transmission* elements, the Modeler is able to encapsulate his knowledge related to a particular component in a compact fashion in one place with well-defined interfaces towards the outside.

A *node* can be connected to arbitrary interfaces. Besides the capability for interconnection of *transmission* elements, or rather the models, the local power balance across the *node* should be guaranteed. It is ensured by the following measurements: The *node* offers a unique *across* variable e, which can be referenced by all of the connected interfaces. The *through* variable f_i of the each connected interface is summarized together in the node, and then formulated as right side of the differential equations or residuum. Equations given by the dynamics of *nodes* and transfer behavior of *transmission* elements, result in a set of differential-algebraic equations (DAE) of the form

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{w}, \mathbf{u}, t) = 0 \tag{1}$$

where $\mathbf{x}$ is a set of state variables, $\mathbf{w}$ is a set of algebraic variables, $\mathbf{u}$ is a set of inputs, and t is the time. Unlike, software based on Modelica, in which the equations are sorted symbolically and reduced eventually to a set of ordinary differential equations (ODE), the resulting DAEs in *D&C Engine* are solved directly by using a numerical DAE solver.

2 Dealing with Discontinuities in Modeling and Simulation

If a physical component is modeled detailed enough, there are usually no discontinuities in the system [8]. When neglecting some"fast" dynamics, in order to reduce the simulation time and identification effort, discontinuities appear in a physical model. In engineering applications, most models of dynamic systems contain some sort of discontinuities due to dry friction, impacts, electrical switches and diodes. Integration across a discontinuity results in a significant reduction of the integration step size and incorrect results. To avoid these difficulties, *event-driven* integration is applied to handle these discontinuities during simulation. In the context of modeling we present the classical regularized approach, a kind of "mild" detailed modeling of discontinuous components, and another approach to model idealized discontinuous components.

2.1 *Event-driven* Integration in *D&C Engine*

One way to avoid the problem of discontinuity was first introduced by F. E. Cellier [2]. The idea is to lock the system equations for each subinterval. The transition points from one interval to another are described as events. Event conditions are specified in the form of a set of root functions. An event occurs if a value of the root functions crosses through zero. Its implementation in *D&C Engine* consists of four steps in the followings:

- Detect change in sign of the root function and stop integration
- Locate root
- Handle event
- Restart Integration

2.2 Regularized Approach for Modeling of Discontinuous Elements

A typical example of discontinuous elements is a diode shown in figure 1, where i is the current through the diode and u is the voltage drop between the pins of the diode. It is straightforward to model the real diode characteristics as in the left part of figure 1. Here the current is a function of voltage drop, therefore it fulfils the *across*-causes-*through* causality for *transmission* elements. On the other hand it is much more difficult to model the ideal diode as in the right part of figure 1, because the current i at $u = 0$ is not a function of u any more, i.e., a mathematical description of $i = f(u)$ is not possible. In this case the current i depends upon the variables outside the model, which are not accessible to modeler who encodes in the diode object. It is often sufficient to use regularized diode characteristics shown in middle part of the figure 1, which is modeled by small and large resistances, R_{on} and R_{off}, for different operation modes. Depending

on parameterization of resistances and system configurations, it can sometimes lead to stiff differential equations. Therefore it is suggested that systems with regularized models of discontinuous elements should be solved by an implicit integration method.

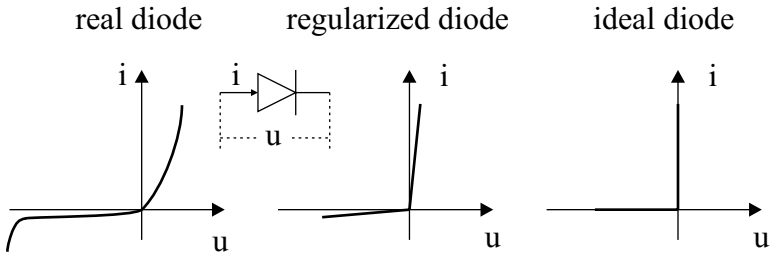


Figure 1: The diode characteristics

If voltage drop u across diode becomes positive, it is switched on. It remains in this mode, until the current through it becomes negative. The equation for the regularized diode is

$$i = \begin{cases} \dfrac{u}{R_{on}} & \text{if } u > 0 \\ \dfrac{u}{R_{off}} & \text{if } u < 0 \end{cases} \tag{2}$$

A regularized model for friction is shown in [1]. A very small, but finite, region ($|v| < v_s$) in the zero vicinity is introduced to eliminate the discontinuity. The friction force F in this region is assumed to be linearly proportional to the velocity, and is always opposed to the velocity. The equations are

$$F = \begin{cases} -\text{sign}(v)\ (F_C + (F_S - F_C)\ e^{(-k|v|)}) - d\ v & \text{if } |v| > v_s \\ -\dfrac{v}{v_S}\ ((F_C + (F_S - F_C)\ e^{(-k|v_s|)}) + d\ v_s) & \text{if } |v| \leq v_s \end{cases} \tag{3}$$

Where F_C is dry (Coulomb) friction, F_S is stiction, k is transition approximation coefficient and d is viscous friction coefficient.

2.3 Modeling of Idealized Discontinuous Elements by Using Constraint Equations

Ideal discontinuous elements have been handled by (a) using variable structure equations which are controlled by finite automata to describe the switching behaviour [7], or by (b) using a complementarity formulation [9], or by (c) using parameterized curve descriptions [8]. (a) has the disadvantage that the algorithms with better convergence properties for the determination of a consistent switching structure cannot be used and a global iteration over all model equations is involved. (b) is difficult to apply for the object-oriented modeling because of difficulties of formulating the equations for linear complementarity problem. (c) needs the external support of symbolic operation to solve algebraic loops.

Modeling of idealized discontinuous elements by using constraint equations was first developed in mechanics for friction and impact, see e.g. [9]. The circuit diagram in figure

2 is taken as an example to illustrate this method for electrical circuits with switching elements. We introduce a variable λ, which stands for the current through the diode.

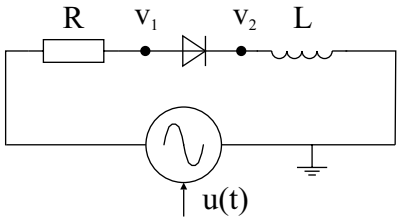


Figure 2: A electrical circuit

The constraint equation is then $v_1 - v_2 = 0$ if it is *on*, and $\lambda = 0$ if the diode is *off*. The generated equations are

$$\begin{aligned} \frac{u(t) - v_1}{R} - \lambda &= 0 && (4) \\ \textbf{if on } v_1 - v_2 \textbf{ else if off } \lambda &= 0 && (5) \\ \lambda - i_L &= 0 && (6) \\ \dot{i}_L &= \frac{v_2}{L} && (7) \end{aligned}$$

The equation system, regardless of diode's *on* or *off*, has always four equations for four unknowns, v_1, v_2, λ and i_L. The change of the constraint equation for the diode is formulated as the change of the residuum for the constraint equation.

Modeling of friction using constraint equations will also be demonstrated here with an example shown in the left part of figure 4, which is used to explain the stick-slip effect in [10]. A box m, is put on a moving conveyor with the velocity v_0, connected to a spring with stiffnes c, which is fixed on the ground. Like for the diode, an additional variable λ is introduced into the friction component. The generated equations for the model result in:

$$\begin{aligned} \dot{x} &= v && (8) \\ \ddot{x} &= \frac{1}{m}(-c\ s - \lambda) && (9) \\ 0 &= \textbf{if Sticking} \quad v - v_0 \\ &\quad \textbf{else if Forward} \quad \lambda + (F_C + (F_S - F_C)\ e^{(-k\lvert v\rvert)}) + d\ v \\ &\quad \textbf{else} \quad \lambda - (F_C + (F_S - F_C)\ e^{(-k\lvert v\rvert)}) - d\ v && (10) \end{aligned}$$

The residuum for the constraint equation changes when mode transition occurs, which is controlled by a state transition diagram shown in figure 3. This approach can also be applied to model rigid impacts.

2.4 Simulation Experiments

Two system models are simulated with both, the regularized approach and the approach for modeling ideal discontinuous elements by using constraint equations. The first system

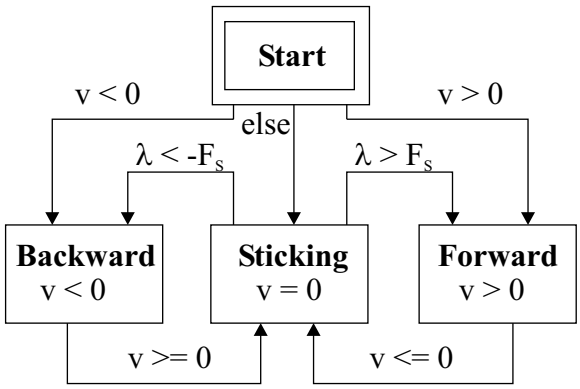


Figure 3: State transition diagram for ideal friction characteristics

model is shown in left part of the figure 4. The parameters are: $m = 1kg$, $c = 80N/m$, $F_S = 10N$, $F_C = 7N$, $v_0 = 0.1m/s$, $k = 10s/m$ and $d = 0sN/m$. The second model is a buck converter, which was used to study the chaos behaviour in [4]. The simulation is performed with the following parameters: $L = 20mH$, $C = 47\mu F$, $R = 22\Omega$, $a = 8.4$, $V_{ref} = 11.3V$, $V_l = 3.8$, $V_u = 8.2V$ and $T = 400us$. The numerical solver is RADAU5 [5].

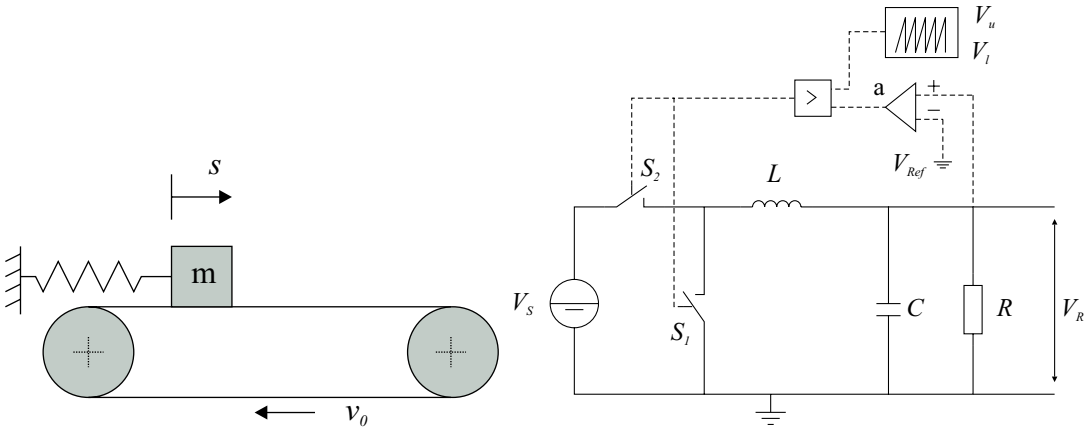


Figure 4: Models of a stick-slip box and a buck converter

The settings of the solver for the regularized model and the ideal model are the same. The error tolerance is 10^{-8}, the maximal integration step size is one-hundredth of the simulation time, the minimal integration step size is ten times of machine accuracy. The simulation time for the first model is 5 seconds and 0.2 second for the second one. The upper left part in figure 5 shows the stick-slip effect for the box on conveyor. The absolute deviation in position for the regularized and ideal model is shown in the lower left figure. The upper right figure shows the chaos behavior of the output voltage V_R in 0.002 second. The lower right figure shows the absolute deviation of V_R between the two approaches. The computational effort regarding time is listed in table 1.

In case of these experiments, the deviations of results are caused by different characteristics of discontinuous elements (regularized and idealized). This tiny difference can be neglected in consideration of dynamics of the whole system. With these two examples the regularized models have slight advantage of time performance compared to the ideal models. There are two reasons generally: The first is that the implicit integration method

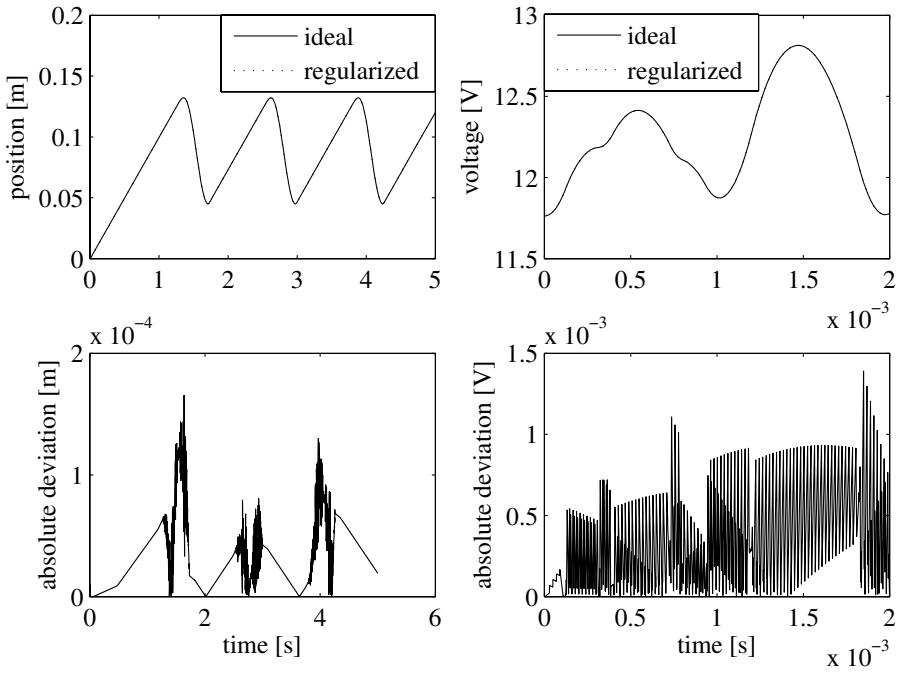


Figure 5: Results of the stick-slip box and the buck converter

is applied to simulate the regularized models, which can handle the stiff differential equations efficiently. The second reason is that the ideal models contain algebraic constraint equations, which need to be solved by Newton iteration once per integration step. This additional computation, of course, partly contributes to the drawback concerning time performance. But the ideal models are advantageous on parameter identification, and are not sensitive to the parameterization, e.g. the resistance in diode, which usually causes a stiff gradient for the right side of differential equations. Another situation for the models of ideal discontinuous elements should be considered, i.e. the equation system could be unsolvable due to "bad" combination of operation modes. For example in case of the buck converter it occurs if both of the switches are *on*. A suitable algorithm should be designed to avoid this situation.

	Regularized model	**Ideal model**
Stick-slip box	0.125 s	0.186 s
Buck converter	0.797 s	1.0 s

Table 1: Time performance of regularized and ideal models

2.5 Conclusions

The software package, *D&C Engine*, was designed to model and simulate mechatronic systems in industrial applications. It is based on the object-oriented modeling approach for complex physical systems. Discontinuities during simulation are treated by an algorithm for *event-driven* integration. In the context of modeling, two approaches were presented. The classical approach for regularizing discontinuous elements is in general

sufficient in reference to performance of computational time and accuracy, as long as the implicit integration method is applied. The approach for modeling the ideal discontinuous elements by using the constraint equation may be useful, if the parameters for regularization are difficult to identify or the systems becomes highly stiff due to regularization. Avoiding the situation of a "bad" combination of operation modes should be considered in the future research.

References

[1] Armstrong, B., Canudas de Wit, C.: *Friction Modeling and Compensation, The Control Handbook*, CRC Press, 1995.

[2] Cellier, F.E.: *Combined Continuous/Discrete System Simulation by Use of Digital Computer: Techniques and Tools*, PhD Thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, 1979.

[3] Elmqvist, H.: *A Structured Model Language for Large Continuous Systems*, PhD dissertation, Depart. of Automatic Control, Lund Institute of Technology, Lund, Schweden, 1978.

[4] Fossas, E., Olivar, G.: *Study of Chaos in the Buck Converter*, IEEE Transactions on Circuits Systems I, Fundamental Theory and Applications 43(1), 13-25, 1996.

[5] Hairer, E., Wanner, G.: *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer Verlag, Berlin, 1991.

[6] Kecskeméthy, A.: *Objektorientierte Modellierung der Dynamik von Mehrkörpersystemen mit Hilfe von Übertragungselementen*, Fortschritt-Berichte VDI, Reihe 20 Nr. 88, VDI Verlag, Düsseldorf, 1993b.

[7] Mosterman P. J., Biswas, G.: *A Formal Hybrid Modeling Scheme for Handling Discontinuities in Physical System Model*, Proceedings of AAAI-96, pp. 905-990, Aug. 2.-4., Portland, OR, 1996.

[8] Otter, M., Elmqvist, H., Mattsson, S. E.: *Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle*, CACSD'99, Hawaii, USA, 1999.

[9] Pffeifer, F., Glocker, C.: *Multibody Dynamics with Unilateral Contacts*, John Wiley, 1996.

[10] Scherf, H. E.: *Modellbildung und Simulation dynamischer Systeme*, Oldenbourg Wissenschaftsverlag, Auflage: 3., 2007.

FunctionalDMU – ein Ansatz für die kooperative, virtuelle Analyse mechatronischer Produkte

Peter Schneider, André Stork, Thomas Bruder, Eckhard Moeller
Fraunhofer-Gesellschaft, Dresden, Darmstadt und Berlin
Peter.Schneider@eas.iis.fraunhofer.de

Kurzfassung

Mechatronische Systeme entstehen durch die enge funktionale und räumliche Integration von Mechanik, Elektronik und Software. In den letzten Jahren sind sowohl Anteil als auch Komplexität mechatronischer Systemlösungen in Produkten signifikant gestiegen. Simultan dazu steigen auch die Anforderungen hinsichtlich der Reduktion von Entwicklungszeit und –kosten bei gleichzeitiger Erhöhung von Qualität und Zuverlässigkeit. Im diesem Spannungsfeld kommen effizienten Entwicklungsprozessen eine stetig wachsende Bedeutung zu.

Derzeit werden bei der Mechatronikentwicklung für die einzelnen Teilgebiete eine ganze Reihe von rechnergestützten Methoden und Werkzeugen eingesetzt. Einzelne Simulationswerkzeuge decken in der Regel jeweils nur einen Teil der mechatronischen Domänen in der für übergreifende Entwicklungsprozesse notwendigen Art und Weise ab. Zur Integration virtueller (Teil-)Produkte auf rein geometrischer Ebene hat sich in den letzten Jahren das Konzept des Digital Mock-Up (DMU) in der industriellen Praxis etabliert. DMU ist heute weitgehend beschränkt auf die geometrische Integration und Analyse verschiedener Teilmodelle. Die Industrie sucht derzeit nachdrücklich nach Softwarewerkzeugen und Methoden zur funktionalen Integration virtueller mechatronischer Produkte.

Im Rahmen des Vortrages wird eine Methodik und ein Software-Framework zur schnellen, komfortablen Integration von Geometrie- und Verhaltensmodellen aus unterschiedlichen mechatronischen Domänen vorgestellt, mit deren Hilfe die virtuelle Analyse funktionaler Prototypen komplexer mechatronischer Produkte ermöglicht wird. Durch diese Lösung wird das „klassische“ DMU um funktionale Aspekte hin zu einem FunctionalDMU erweitert. Die Integration von Verhaltensmodellen und Simulationswerkzeugen der Domänen Mechanik, Elektronik und Software sichert die Kompatibilität mit den etablierten Entwicklungsprozessen

Anhand eines Beispiels aus dem Automotive-Umfeld werden Anwendungsaspekte der Softwarelösung diskutiert. Hierzu wird u.a. auf die flexible Ankopplung typischer Simulatoren, die Vorgehensweise zur Modellbildung und die Durchführung von Berechnungen innerhalb des Frameworks eingegangen. Die Unterschiede und Vorteile der gekoppelten Simulation zur konventionellen Vorgehensweise werden verdeutlicht und Potentiale zur Produktoptimierung aufgezeigt. Der Vortrag schließt mit einem Ausblick auf zukünftige Anwendungen.

Literatur

[1] Clauß, Ch.; Reitz, S.; Schwarz, P.: Simulation mechanisch-elektrischer Wechselwirkungen am Beispiel eines sensorischen Mikrosystems. SIM2000 – Simulation im Maschinenbau, Dresden, 24.-25. März 2000, S. 183 ff.

[2] Schneider, P.; Huck, E.; Schwarz, P.: A Modeling Approach for Mechatronic Systems -Modeling and Simulation of an Elevator System. XI. Intern. Symposium in Theoretical Electrical Engineering, Linz, August 19-22. 2001

[3] Bruder, T.; Wallmichrath, M.; Jöckel, M.: VTL - Virtuelle Prüfumgebung als Bindeglied zwischen Simulation und Prüftechnik; Tagungsband des DVM-Workshops Prüfmethodik für Betriebsfestigkeitsversuche in der Fahrzeugindustrie; 2006

FunctionalDMU

ein Ansatz für die kooperative,
virtuelle Analyse mechatronischer Produkte

http://www.FunctionalDMU.org

Überblick

- Motivation und Zielsetzung
- Methodik
- Anwendungsszenarien und Demonstrator
- Zusammenfassung und Ausblick

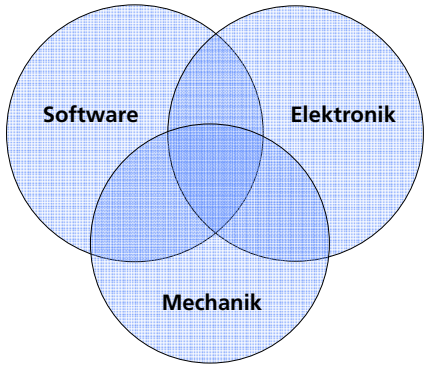
Aktueller Stand des DMU-Ansatzes

Ziele von DMU

- Ersetzen von physischen Versuchsmodellen (Physical Mock-Up - PMU)
- Bereitstellung verschiedener, aktueller und konsistenter Sichtweisen auf die Gestalt und Funktion eines Produktes

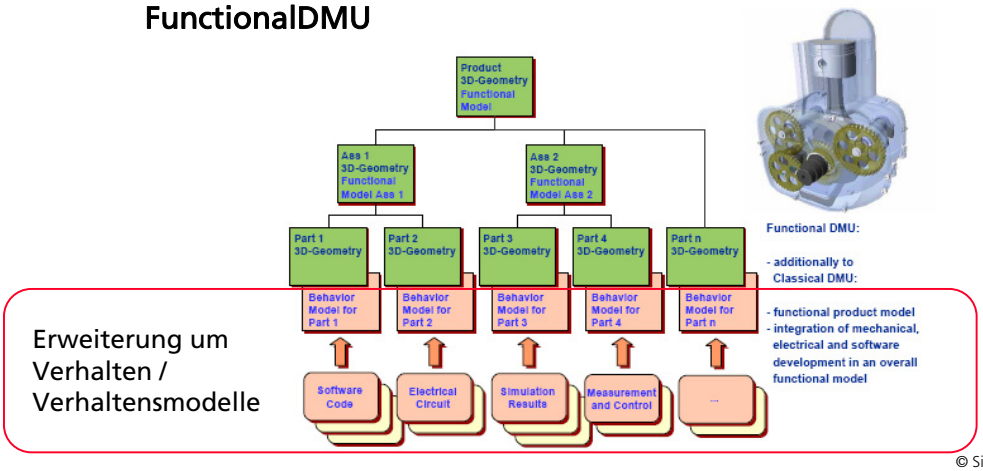
Grenzen von DMU

- Simuliert werden lediglich quasistatische, mechanische Zusammenhänge
- Elektronik- und Softwareanteile von mechatronischen Komponenten können (noch) nicht simuliert werden
- Zur realen Überprüfung der virtuellen Ergebnisse steht am Ende eines DMU-Prozesses meist immer noch ein Physical Mock-Up

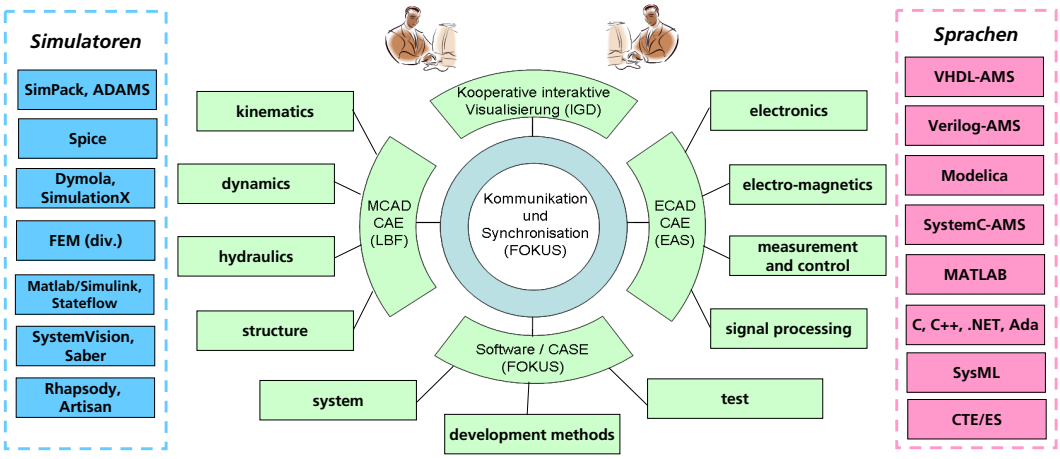


Innovation: DMU -> FunctionalDMU

FunctionalDMU



FunctionalDMU

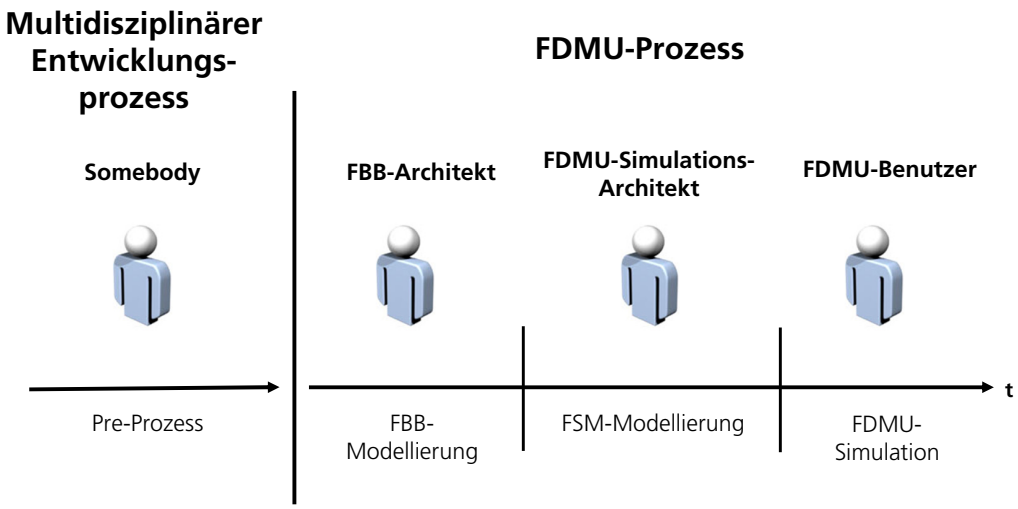


Softwaretechnische Zielstellung

- Software-Plattform
  - Interaktion zwischen Simulationswerkzeugen
  - Integration von existierenden Werkzeugen, Verhaltensmodellen, Sprachen
  - Herstellerunabhängigkeit
  - Offenheit
  - Skalierbarkeit
    - Modularer und dienstorientierter Aufbau (Web Services, SOA)
    - Erweiterbarkeit (auch über Unternehmensgrenzen hinweg)

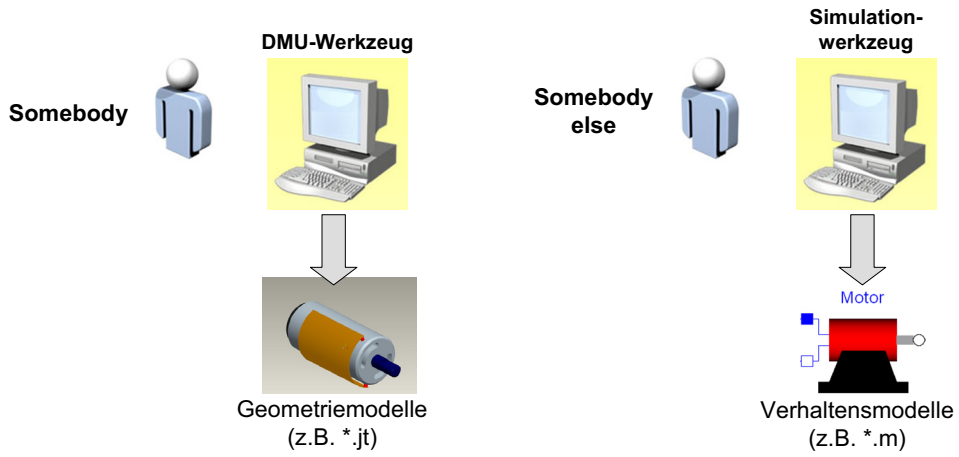
- Aufbau interdisziplinärer Simulationsmodelle
  - FDMU-Modellierungsumgebung zur Komposition von Verhaltensmodellen
  - Interaktive Visualisierung und Steuerung der Simulation

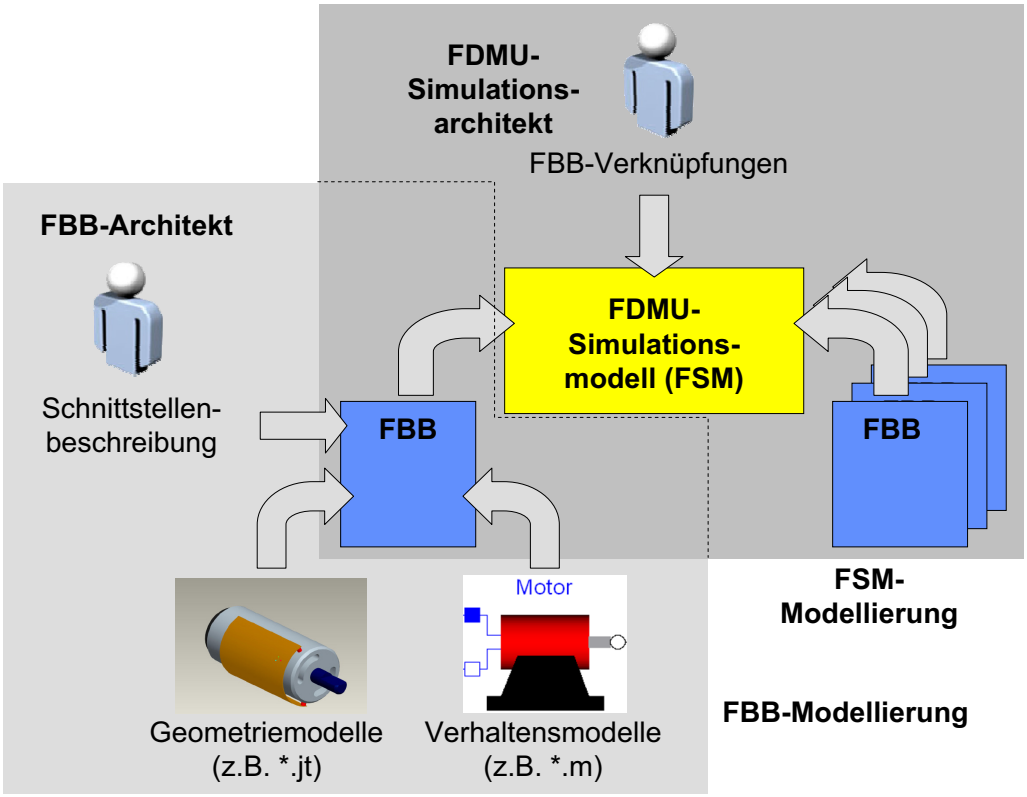
FDMU-Methodik



FDMU-Methodik

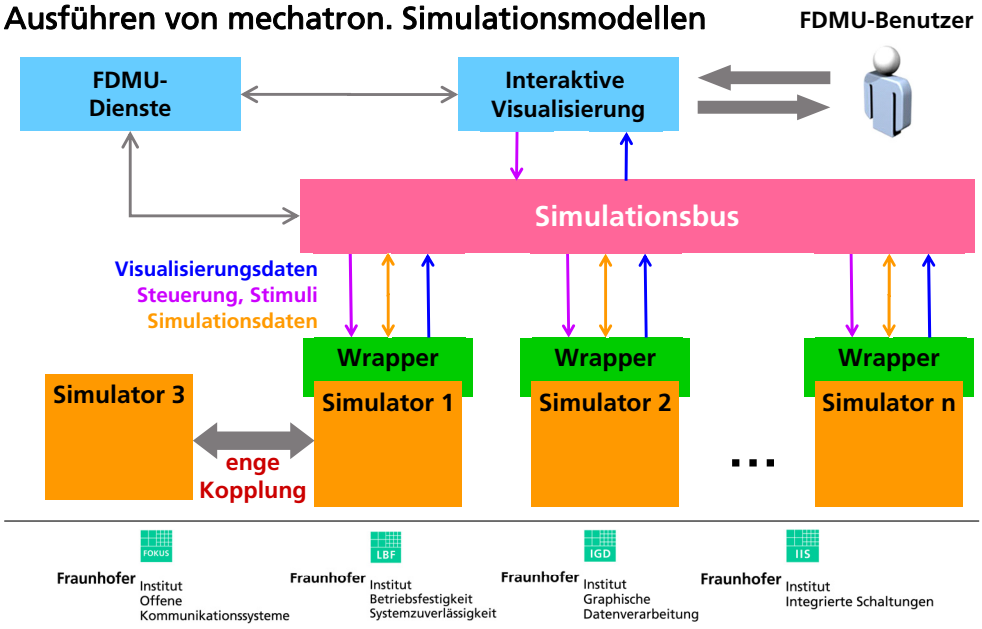
Ausgangssituation





FDMU-Methodik

Ausführen von mechatron. Simulationsmodellen

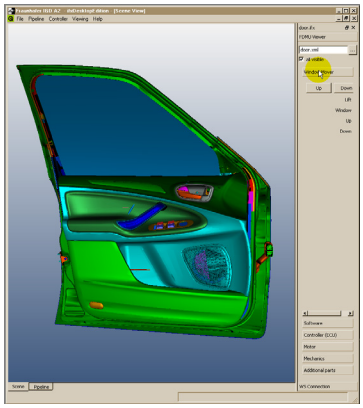


Anwendungsszenarien

- 6 Szenarien ausgearbeitet - zur Evaluierung des FDMU Frameworks
  - Fensterheber
  - Lenkungsprüfstand Funktionstest
  - Lenkungsprüfstand Austausch
  - E-Motor
  - Lane Departure Warning
  - Crash Funktion

- 1. Demonstrator: „Fensterheber“

Szenario Fensterheber



- Scherenmechanik bewegt PKW-Türfensterscheibe.
- Angetrieben wird sie über ein Getriebe durch einen Elektromotor.
- Motorsteuerung (ECU) gibt die Motor-spannung vor und überwacht gleichzeitig den Motorstrom.
- Steigt dieser über einen Schwellwert an, schaltet die Steuerung die Spannung ab.
- Die ECU erhält außerdem von einem Schalter (Taster) ein Signal zum Öffnen und Schließen der Fenster.
- Die Auswertung der Signale erfolgt innerhalb der ECU durch Software.

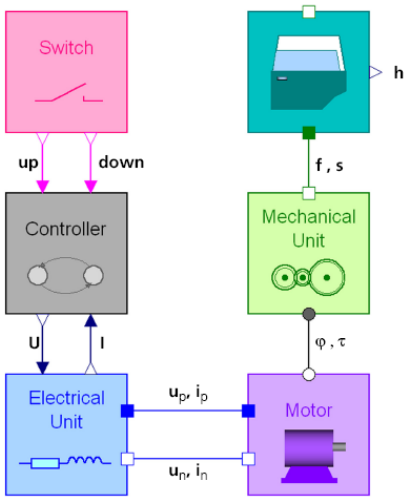
Szenario Fensterheber

WindowRegulator

Die Teilmodelle sind über die folgenden Schnittstellen miteinander verbunden:

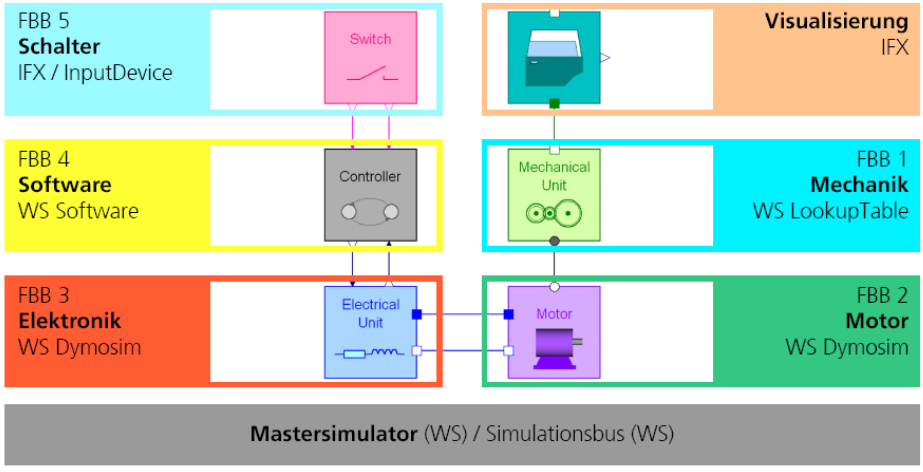
Hochfahren	up	Boolean
Runterfahren	down	Boolean
Stellgröße	U	Real
Regelgröße	I	Real
Spannung *	u	Real
Strom *	i	Real
Winkel *	φ	Real
Drehmoment *	τ	Real
Kraft *	f	Real
Weg *	s	Real
Scheibenhöhe	h	Real

* *ungerichtet*

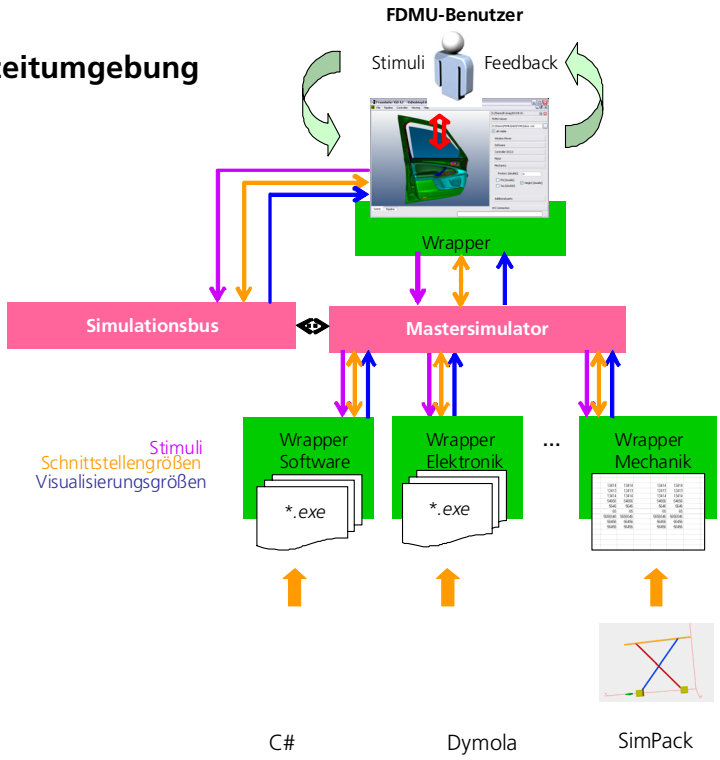


Szenario Fensterheber

Abbildung der Modellteile auf FBB's



Laufzeitumgebung



Fensterheber-Szenario

1. **Welche Aufgabe soll erfüllt werden?**

Auf die Tastersignale hin soll die Fensterbewegung erfolgen.

2. **Welche Fragen sollen beantwortet werden?**

- Wie schnell bewegt sich das Fenster?
- Wird ein Hindernis die Bewegung des Fensters blockieren?
- Funktioniert das System bei veränderten Reibungswerten?
- Wie ist der Verlauf des Drehmomentes an der Motorachse?
- Welcher Stromverlauf ergibt sich?
- Funktioniert die Steuerung?

3. **Welche Modellparameter sollen verändert werden?**

- Reibungswert in Abhängigkeit von der Position
- Nominalspannung und Ankerwiderstand des Motors
- Schwellwert des Stromes für das Abschalten

4. **Welche Daten sollen visualisiert werden?**

- Position des Fensters
- Verlauf des Drehmomentes (Motorachse)
- Strom- und Spannungsverlauf (Motor)

5. **Welche Interaktionen sollen erfolgen?**

Betätigen des Fensterhebers (Taster) durch graphische Interaktion

6. **Welche Einsichten gewinnt der Nutzer?**

Einhaltung von Grenzwerten (max. Strom, max. Drehmoment) in der Bewegung

Zusammenfassung

Kundennutzen

- Funktionales Design Review komplexer Mechatronik wird beherrschbar
- Verkürzung der Entwicklungszeit mechatronischer Produkte
- Bessere Produkte zu verringerten Kosten (domänen-übergreifende Optima)

Stärken der Lösung

- Hersteller-unabhängiges und -übergreifendes, offenes Framework (SOA)
- Visualisierung von komplexen funktionellen Wechselwirkungen und graphische Interaktion mit Modellen
- Methodikentwicklung für FunctionalDMU

Herausforderungen

- Vielfalt der Tools, Datenformate, Schnittstellen
- Usability und Akzeptanz
- Performance der Co-Simulation

Ausblick

- **Integrierte Simulatoren – Stand heute**
  - Software (C-Code)
  - Elektronik/Multi-Physik (Dymola)
  - Mechanik/MKS (SimPack)
- **Anbindung weiterer Simulatoren (voraussichtlich):**
  - AdvanceMS
  - Matlab/Simulink
  - Rhapsody od. Artisan
  - Abaqus, ANSYS
- und weiteres mehr, um die **Plattform effizienter und einfacher benutzbar** zu machen!

Fraunhofer Institut Offene Kommunikationssysteme
Fraunhofer Institut Betriebsfestigkeit Systemzuverlässigkeit
Fraunhofer Institut Graphische Datenverarbeitung
Fraunhofer Institut Integrierte Schaltungen

Ferndiagnose dynamischer technischer Systeme

Dipl.-Ing. Thorsten Schlage, Prof. Dr.-Ing. Jan Lunze
Lehrstuhl für Automatisierungstechnik und Prozessinformatik
Ruhr-Universität Bochum
Schlage@atp.rub.de, Lunze@atp.rub.de

Zusammenfassung

Dieser Beitrag stellt einen neuen Ansatz zur Ferndiagnose technischer Systeme vor, bei dem die Diagnoseaufgabe auf eine systemnahe und eine systemferne Komponente aufgeteilt wird. Während auf der systemnahen Komponente die Fehlerdetektion erfolgt, werden auf der systemfernen Komponente die Aufgaben der Fehlerisolation und der Fehleridentifikation gelöst. Die vorgeschlagene Aufteilung der Diagnoseaufgabe verringert durch die Verwendung systemferner Ressourcen die Anforderungen an die systemnahe Komponente hinsichtlich Rechenleistung und Speicherkapazität. Die entwickelte Diagnosemethode beruht auf dem Prinzip der konsistenzbasierten Diagnose, die hier für ereignisdiskrete Systeme angewendet und weiterentwickelt wird. Das ereignisdiskrete System wird durch zeitbewertete E/A-Automaten beschrieben. Die Kommunikation zwischen der systemnahen und der systemfernen Komponente unterliegt netzwerkspezifischen Effekten wie Zeitverzögerungen und Datenverlusten, welche bei der Aufteilung der Diagnoseaufgabe berücksichtigt werden. In diesem Beitrag werden die Modellbildung mittels zeitbewerteter Automaten anhand eines Beispielprozesses veranschaulicht, die methodischen Grundlagen der Ferndiagnose erläutert und experimentelle Ergebnisse zur konsistenzbasierten Diagnose vorgestellt.

1 Einleitung

In technischen Systemen treten Fehler auf, die zu Ausfallzeiten oder zur Gefährdung von Mensch und Maschine führen können. Die Aufgabe der Diagnose ist es, diese Fehler zu erkennen und zu identifizieren. Das Lösen der Diagnoseaufgabe benötigt enorme Systemressourcen hinsichtlich Rechenaufwand und Speicherkapazität, die oftmals in Automatisierungseinrichtungen nicht zur Verfügung stehen. Bei dem in diesem Beitrag vorgestellten Ansatz der Ferndiagnose erfolgt eine Aufteilung der Diagnoseaufgabe auf eine systemnahe (onboard) und eine systemferne (offboard) Komponente, um die Anforderungen an die systemnahe Komponente zu verringern. Abbildung 1 zeigt die Aufteilung der Diagnoseaufgabe anhand eines Fahrzeugbeispiels. Die Übertragung der Daten erfolgt zwischen den Komponenten über moderne Kommunikationsmedien, wie z.B. das Internet.

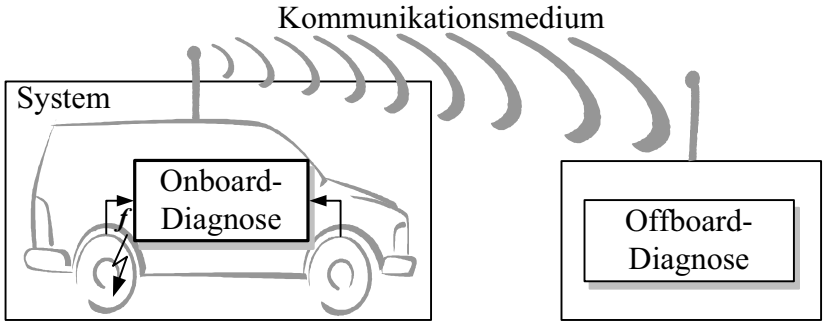


Abbildung 1: Struktur der Ferndiagnose

Im Abschn. 2 werden die Modellform der zeitbewerteten E/A-Automaten und die konsistenzbasierte Diagnose ereignisdiskreter Systeme erläutert. Die Struktur der Ferndiagnose wird in Abschn. 3 vorgestellt. Modellbildung und Diagnose werden in Abschn. 4 anhand eines Beispiels veranschaulicht. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick in Abschn. 5.

2 Konsistenzbasierte Diagnose ereignisdiskreter Systeme

2.1 Zeitbewertete E/A-Automaten

Betrachtet werden ereignisdiskrete Systeme [1] mit den Eingangsgrößen $v \in \mathcal{N}_\mathrm{v} = \{1, 2, ..., M\}$, Ausgangsgrößen $w \in \mathcal{N}_\mathrm{w} = \{1, 2, ..., Q\}$ und Zustandsgrößen $z \in \mathcal{N}_\mathrm{z} = \{1, 2, ..., N\}$ (Abb. 2). Das Verhalten $\mathcal{B}$ des ereignisdiskreten Systems wird durch die vom

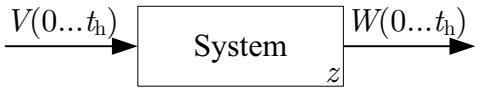


Abbildung 2: Ereignisdiskretes System

System erzeugbaren zeitbewerteten Eingangs- und Ausgangsfolgen

$$\begin{aligned} V(0...t_\mathrm{h}) &= (v_0, t_0; v_1, t_1; ...; v_\mathrm{h}, t_\mathrm{h}), \\ W(0...t_\mathrm{h}) &= (w_0, t_0; w_1, t_1; ...; w_\mathrm{h}, t_\mathrm{h}) \end{aligned}$$

beschrieben, wobei der Eingang v_i und Ausgang w_i zum Zeitpunkt t_i auftreten. Das ereignisdiskrete System kann durch zeitbewertete Eingangs/Ausgangs-Automaten (E/A-Automaten) $\mathcal{A}_\mathrm{T} = (\mathcal{N}_\mathrm{z}, \mathcal{N}_\mathrm{v}, \mathcal{N}_\mathrm{w}, L_\mathrm{T})$ [4] beschrieben werden, deren Verhaltensrelation

$$L_\mathrm{T} : \mathcal{N}_\mathrm{z} \times \mathcal{N}_\mathrm{w} \times \mathcal{N}_\mathrm{z} \times \mathcal{N}_\mathrm{v} \times \mathbb{R}^+ \rightarrow \{0, 1\}$$

den Wert eins besitzt

$$L_\mathrm{T}(z', w, z, v, \tau) = 1,$$

wenn der Automat den Zustand z zum Zeitpunkt t angenommen hat, die Eingabe v erhält, in den Nachfolgezustand z' zum Zeitpunkt $t+\tau$ übergehen kann und dabei die Ausgabe w erzeugt. Ist der Zeitpunkt eines möglichen Übergangs nicht exakt bestimmbar, so können die Zeitpunkte, zu denen der Automat vom Zustand z zu z' wechseln kann, durch ein Intervall

$$\tau(z', w, z, v) \in [\tau_{\min}(z', w, z, v), \tau_{\max}(z', w, z, v)]$$

beschrieben werden. Der zeitbewertete E/A-Automat ist durch einen gerichteten Automatengrafen darstellbar (Abb. 3).

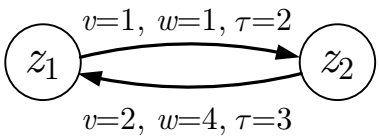


Abbildung 3: Graf eines zeitbewerteten E/A-Automaten

2.2 Konsistenzbasierte Diagnose

Der Grundgedanke der konsistenzbasierten Diagnose besteht in der Konsistenzprüfung der gemessenen E/A-Folge $(V(0...t_\mathrm{h}), W(0...t_\mathrm{h}))$ mit dem Modell $\mathcal{A}_\mathrm{T}$.

Definition 1 (Konsistenz [5]) *Ein Modell $\mathcal{A}_\mathrm{T}$ ist mit der gemessenen E/A-Folge $B(t_\mathrm{h}) = (V(0...t_\mathrm{h}), W(0...t_\mathrm{h}))$ konsistent, wenn $B(t_\mathrm{h}) \in \mathcal{B}$ gilt.*

Neben dem Modell $\mathcal{A}_{\mathrm{T},f_0}$ des fehlerfreien Systems werden bei der konsistenzbasierten Diagnose auch Modelle $\mathcal{A}_{\mathrm{T},f_i}, i = 1..s$, aufgestellt, die das jeweilige Systemverhalten $\mathcal{B}(f_i)$ unter Einfluss der Fehler $f_i \in \mathcal{N}_f$ beschreiben. Alle Modelle $\mathcal{A}_{\mathrm{T},f_i}, i = 0...s$, werden auf Konsistenz mit der gemessenen E/A-Folge $B(t_\mathrm{h})$ überprüft. Die Fehler, für die das jeweilige Systemverhalten $\mathcal{B}(f_i)$ mit der gemessenen E/A-Folge konsistent ist, bilden die Menge der Fehlerkandidaten $\mathcal{F}^*$.

Definition 2 (Fehlerkandidaten) *Die Fehler $f_i \in \mathcal{F}^*$ mit $\mathcal{F}^*(t_\mathrm{h}) = \{f | B(t_\mathrm{h}) \in \mathcal{B}(f_i)\}$ werden Fehlerkandidaten genannt.*

Da mehrere Modelle mit der E/A-Folge konsistent sein können, kann die Menge der Fehlerkandidaten mehr als einen Fehler beinhalten. Das Vorgehen der konsistenzbasierten Diagnose ist im Algorithmus 1 zusammengefasst.

Algorithmus 1 (Konsistenzbasierte Diagnose [6])

Gegeben: - *Menge der Modelle* $\mathcal{N}_A = \{\mathcal{A}_{\mathrm{T},f_0}, ..., \mathcal{A}_{\mathrm{T},f_s}\}$
- *Gemessene E/A-Folge* $(V(0...t_\mathrm{h}), W(0...t_\mathrm{h}))$

Gesucht: - *Menge von möglichen Fehlern* $\mathcal{F} \subseteq \mathcal{N}_f$, $f \in \mathcal{F}$

Wiederhole für alle $\mathcal{A}_{\mathrm{T},f_i} \in \mathcal{N}_A$*:*

Ist das Modell $\mathcal{A}_{\mathrm{T},f_i}$ *konsistent mit der gemessenen E/A-Folge?*

- Ja: $f_i \in \mathcal{F}$

- Nein: $f_i \notin \mathcal{F}$

Ergebnis: - Menge von möglichen Fehlern $\mathcal{F}$*, mit* $\mathcal{F} = \mathcal{F}^*$

Der Algorithmus 1 liefert das ideale Diagnoseergebnis $\mathcal{F} = \mathcal{F}^*$.

Definition 3 (Vollständigkeit des Diagnoseergebnis) *Ein Diagnoseergebnis* $\mathcal{F} \subseteq \mathcal{N}_f$ *heißt vollständig, wenn die Beziehung* $\mathcal{F}^* \subseteq \mathcal{F}$ *gilt.*

3 Struktur der Ferndiagnose

Im Allgemeinen unterteilt sich die Aufgabe der Diagnose in die drei Schritte [2]:

- **Fehlerdetektion:** Überprüfung ob ein Fehler aufgetreten ist.
- **Fehlerisolation:** Untersuchung in welcher Komponente der Fehler aufgetreten ist.
- **Fehleridentifikation:** Entscheidung welcher Fehler aufgetreten ist.

Für die Ferndiagnose sind diese Diagnoseschritte geeignet auf die systemnahe und systemferne Komponente aufzuteilen. Bei der Aufteilung muss beachtet werden, dass der systemnahen Komponente zwar sämtliche Messinformationen zur Verfügung stehen, sie aber nur eine geringe Rechenleistung und Speicherkapazität besitzt. Dem hingegen stehen der systemfernen Komponente nahezu unbegrenzte Systemressourcen zur Verfügung, jedoch nur eingeschränkte Informationen über das System.

Die Aufteilung der Diagnoseaufgabe auf die systemnahe (Onboard-Diagnose) und die systemferne (Offboard-Diagnose) Komponente ist in Abb. 4 dargestellt.

- **Onboard-Fehlerdetektion:** Die systemnahe Komponente löst die Aufgabe der Fehlerdetektion. Da zur Fehlerdetektion nur das Modell des nominellen Systemverhaltens notwendig ist, wird nur eine geringe Rechenleistung und Speicherkapazität benötigt.
- **Offboard-Fehlerisolation und -Fehleridentifikation:** Auf der systemfernen Komponente erfolgen die Fehlerisolation und Fehleridentifikation. Während zur Fehlerisolation ein komponentenorientiertes Modell notwendig ist, werden zur Fehleridentifikation Modelle des fehlerbehafteten Systemverhaltens benötigt.

Aufgrund der Bandbreitenbeschränkung des Kommunikationsmediums muss eine Auswahl der zur Offboard-Diagnose notwendigen Daten $p_{\mathrm{off}} \subset p$ erfolgen. Bei der Auswahl der Daten wird zudem festgelegt, zu welchen Zeitpunkten die entsprechenden Daten zur

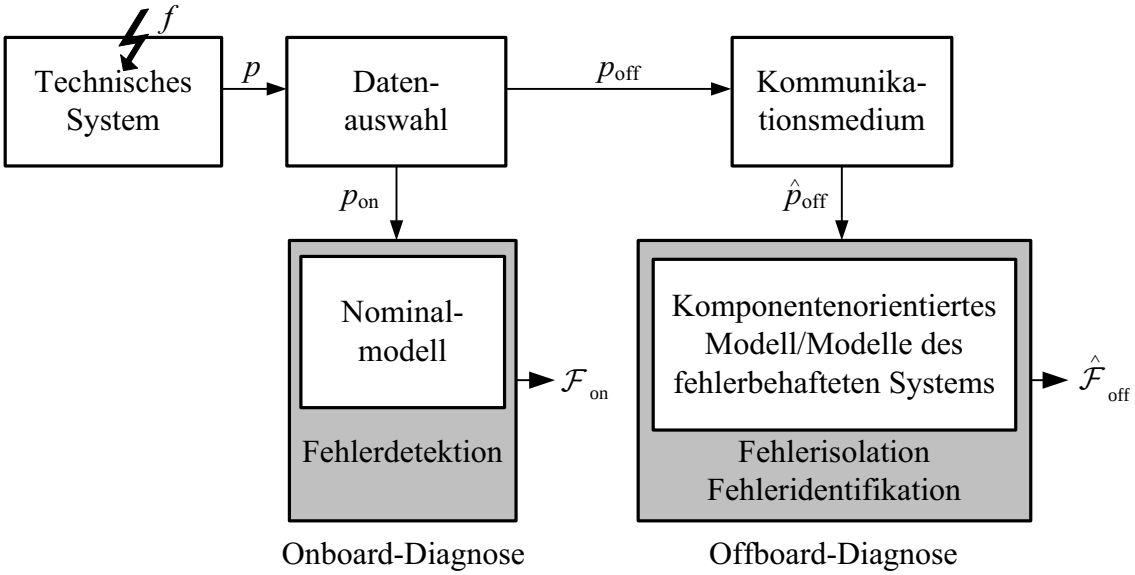


Abbildung 4: Aufteilung der Diagnoseaufgabe

systemfernen Komponente versendet werden. Des Weiteren kann eine Auswahl der zur Onboard-Diagnose verwendeten Daten $p_{\text{on}} \subseteq p$ vorgenommen werden.

Das Versenden der Daten p_{off} unterliegt aufgrund netzwerkspezifischer Eigenschaften u.a. variablen Zeitverzögerungen, Datenverlusten und Datenvertauschungen. Der Offboard-Diagnose stehen somit nur die veränderten bzw. unvollständigen Daten $\hat{p}_{\text{off}}$ zur Verfügung. Trotz der eingeschränkten Messinformationen wird gefordert, dass das Ergebnis der Offboard-Diagnose $\hat{\mathcal{F}}_{\text{off}}$ vollständig entsprechend Definition 3 ist.

Wird der in Abschn. 2.2 vorgestellte Algorithmus der konsistenzbasierten Diagnose unverändert auf der systemfernen Komponente implementiert, so führt dies aufgrund der eingeschränkten Messinformationen nicht zu einem vollständigen Diagnoseergebnis $\mathcal{F} \not\supseteq \mathcal{F}^*$. Der Offboard-Diagnosealgorithmus ist folglich so anzupassen bzw. zu erweitern, dass er den Einfluss des Kommunikationsmediums auf den Datenversand einbezieht und zu einem vollständigen Diagnoseergebnis $\mathcal{F} \supseteq \mathcal{F}^*$ führt. Einzelheiten sind in [3] beschrieben.

4 Beispielprozess

Abbildung 5 zeigt im Grundriss einen Ausschnitt aus einer Fertigungszelle. In dem Metallblock, der Kühlung und der Heizung können bis zu vier zylindrische metallische Elemente in Vertiefungen abgelegt werden. Die Elemente können mittels einer Transporteinheit zwischen den Komponenten transportiert werden, so dass ein Wärmeaustausch zwischen der Heizung bzw. Kühlung und dem Metallblock mit Hilfe von Elementen möglich ist.

Betrachtet wird ein Prozess an der Fertigungszelle mit dem Ziel, den Metallblock zwischen zwei vorgegeben unteren und oberen Temperaturgrenzen T_{u} und T_{o} aufzuheizen und abzukühlen. Das Heizen und Kühlen erfolgt durch je vier Heiz- und Kühlelemente, die in der Heizung bzw. Kühlung gelagert und abwechselnd zum Metallblock hin und wieder zurück transportiert werden. Die Transportvorgänge der jeweils gesamten vier Heiz- bzw. Kühlelemente werden durch das Erreichen des unteren bzw. oberen Grenzwerts ausgelöst.

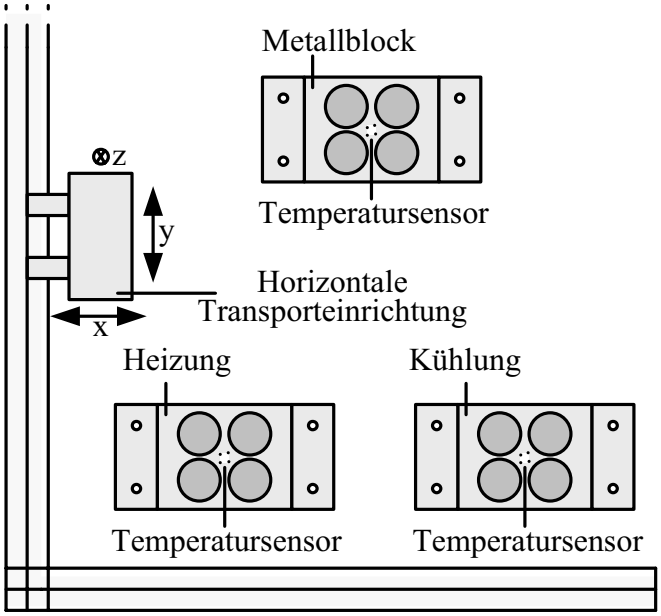


Abbildung 5: Grundriss der Fertigungszelle

Der zeitliche Ablauf des Prozesses wird im Wesentlichen durch die Zeitpunkte bestimmt, zu denen die Metallblocktemperatur die vorgegebenen Grenzwerte erreicht. Bei einer ereignisdiskreten Betrachtungsweise des Prozesses entspricht das Erreichen des jeweiligen Grenzwerts einem Ereignis, das eine diskrete Zustandsänderung des Systems auslöst. Der Systemzustand wird hierbei nicht durch den kontinuierlichen Verlauf der Metallblocktemperatur beschrieben, sondern durch die Temperaturen, die der Metallblock beim Auftreten eines Ereignisses besitzt und folglich den vorgegebenen Temperaturgrenzwerten entsprechen. Zur ereignisdiskreten Modellierung des Prozesses werden die in Tabelle 2 angegebenen diskreten Zustände z_1 und z_2 definiert.

Des Weiteren werden jeweils zwei diskrete Eingaben und Ausgaben entsprechend Tabelle 2 definiert. Die Eingaben v_h und v_k, die das Aufheizen bzw. Abkühlen des Metallblocks einleiten, dienen zur Steuerung des Prozesses. Da der Metallblock wechselweise aufgeheizt und abgekühlt wird, erfolgen die Eingaben abwechselnd. Wird der obere bzw. untere Temperaturgrenzwert erreicht, so wird die Ausgabe w_o bzw. w_u ausgegeben und anschließend ein neuer Abkühl- bzw. Aufheizvorgang gestartet.

Eingabe	**Ausgabe**	**Zustand**
v_h : Metallblock aufheizen	w_o : T_o ist erreicht	$z_1 : T_\mathrm{M} = T_\mathrm{u}$
v_k : Metallblock abkühlen	w_u : T_u ist erreicht	$z_2 : T_\mathrm{M} = T_\mathrm{o}$

Tabelle 2: Definition der Eingaben, Ausgaben und Zustände

Neben dem fehlerfreien Fall f_0 wird der Fehlerfall f_1 betrachtet, bei dem eines der nominellen Heizelemente durch ein Heizelement mit wesentlich geringerer Masse und Wärmekapazität ersetzt wird. In Abb. 6 sind die resultierenden Temperaturverläufe des Metallblocks für die beiden Fälle f_0 und f_1 abgebildet. Ein Vergleich zeigt einen deutlich unterschiedlichen Verlauf der Metallblocktemperaturen im fehlerfreien und fehlerbehafteten Fall. Die Unterschiede sind nicht nur im kontinuierlichen Verlauf, sondern auch

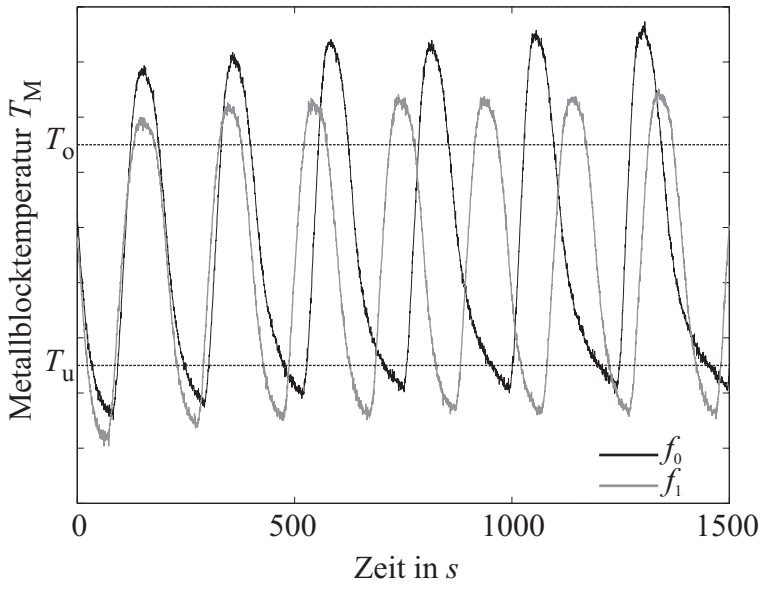


Abbildung 6: Verlauf der Metallblocktemperaturen für die Fehlerfälle f_0 und f_1

in den Zeitpunkten des Erreichens der Temperaturgrenzen zu erkennen. Aufgrund der Verwendung eines Heizelements mit geringerer Wärmekapazität, werden im Fehlerfall f_1 geringere maximale und minimale Temperaturen als im nominellen Prozess erreicht.

In Abb. 7 sind die experimentell ermittelten Automatengrafen des fehlerfreien (a) und fehlerbehafteten (b) Prozesses dargestellt. Aufgrund der unterschiedlichen Erwärmung

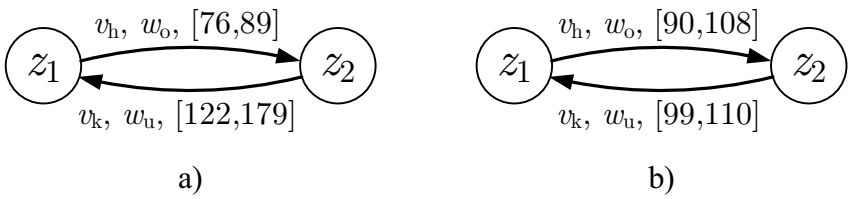


Abbildung 7: Automatengraf für den fehlerfreien (a) und fehlerbehafteten (b) Prozess

des Metallblocks in dem fehlerfreien und fehlerbehafteten Prozess resultieren abweichende Verweilzeitintervalle in den Zuständen. Da sich die Verweilzeitintervalle nicht überschneiden, kann sichergestellt werden, dass die Fehlerfälle f_0 und f_1 unterscheidbar sind.

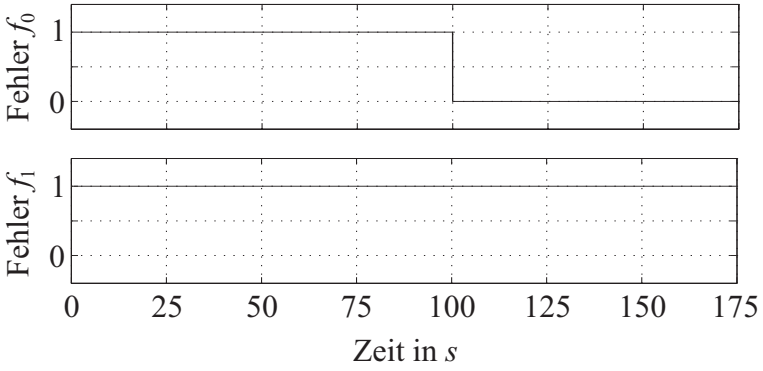


Abbildung 8: Fehler f_0 und f_1 (1: $f_i \in \mathcal{F}$ (Konsistenz), 0: $f_i \notin \mathcal{F}$ (Inkonsistenz))

Wird die Eingangs- und Ausgangsfolge

$$(V(0\text{s}...100\text{s}) = (v_\text{h}, 0\text{s}; v_\text{k}, 100\text{s}), W(0\text{s}...100\text{s}) = (w_\text{o}, 0\text{s}; w_\text{u}, 100\text{s}))$$

gemessen, so wechselt das System zum Zeitpunkt $t = 0$s in den Zustand z_2 und verbleibt dort eine Verweilzeit von $\tau = 100$s. Das Modell des fehlerfreien Prozessverhaltens ist somit inkonsistent mit der gemessenen E/A-Folge und der fehlerfreie Fall f_0 kann zum Zeitpunkt $t = 100$s aus der Menge der möglichen Fehler $\mathcal{F}$ ausgeschlossen werden (Abb. 8). Da das Modell des fehlerbehafteten Prozessverhaltens mit der Messung konsistent ist, ist der Fehlerfall f_1 identifiziert.

5 Zusammenfassung und Ausblick

Die in diesem Beitrag vorgeschlagene Aufteilung der Diagnoseaufgabe auf eine systemnahe und eine systemferne Komponente reduziert die auf der systemnahen Komponente erforderlichen Systemressourcen hinsichtlich Rechenaufwand und Speicherkapazität. Die durch das Kommunikationsmedium hinzukommenden Effekte wie Zeitverzögerungen, Datenverluste und Datenvertauschungen müssen jedoch bei den auf der systemfernen Komponente verwendeten Diagnosealgorithmen berücksichtigt werden. Weitere Arbeiten müssen sich folglich mit der Einbeziehung des Kommunikationsmediums in die Offboard-Diagnose beschäftigen. Hierzu sind Modelle des Kommunikationsmediums in Form zeitbewerteter E/A-Automaten aufzustellen und in den Algorithmus der konsistenzbasierten Diagnose auf der systemfernen Komponente einzubeziehen.

Literatur

[1] Lunze, J.: *Ereignisdiskrete Systeme*, Oldenbourg Verlag, München Wien, 2006.

[2] Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, J.: *Diagnosis and Fault-Tolerant Control*, Springer Verlag, Berlin Heidelberg, 2. Auflage, 2006.

[3] Fritsch, C., Lunze, Schwaiger, M., Krebs, V.: *Remote diagnosis of discrete-event systems with on-board und off-board components*, Proceedings of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (Safeprocess), Beijing, China, 2006.

[4] Supavatanakul, P.: *Modelling and Diagnosis of Timed Discrete-Event Systems*, Shaker-Verlag, Aachen, 2004.

[5] Neidig, J.: *An Automata Theoretic Approach to Modular Diagnosis of Discrete-Event Systems*, Books on Demand, Norderstedt, 2007.

[6] Fritsch, C., Lunze, J.: *Complexity Reduction of Remote Diagnosis of Discrete-Event Systems*, Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006.

Entwurfsumgebung für hallbasierte 3D-Positionssensorsysteme

Jörg Bretschneider, Andreas Wilde
Fraunhofer Institut für Entwurfsautomatisierung
denvi@eas.iis.fraunhofer.de

Zusammenfassung

Magnetische Sensorsysteme zur Bestimmung von Relativpositionen haben vielfältige Anwendungsmöglichkeiten, u. a. in der Automobilindustrie. Die neue Fraunhofer HallinOne® Technologie eröffnet die Möglichkeit, zuverlässige und preiswerte räumliche Positionsmeßsysteme auf Basis von 3D-Hallsensoren zu entwickeln. Der Beitrag gibt einen Überblick über die Entwurfs-Problematik solcher Systeme und stellt einen Ansatz zur Entwurfsunterstützung vor, der verschiedene Simulationsansätze, die Integration von Messdaten und eine problemangepasste Visualisierung umfasst. Dadurch werden verschiedene Stufen bei der Entwicklung solcher Systeme unterstützt, von Machbarkeitsstudien über Variantenvergleiche bis hin zu Toleranzabschätzungen und zur Validierung von Messungen.

1 Magnetische Positionssensorsysteme

Für die Messung des Magnetfeldes von Permanentmagneten werden heute vor allem halleffektbasierte Sensoren eingesetzt, d.h., es wird der physikalische Zusammenhang $U_H \sim B$ zwischen der magnetischen Flußdichte B und der Hall-Spannung U_H ausgenutzt. Stand der Technik in Standard-CMOS-Technologie sind 1D-Hall-Sensor-ICs, welche die Magnetfeldkomponente B_z senkrecht zur Chipoberfläche messen. Eine Mehrzahl der Applikationen nutzt die Magnetfeldmessung zur Bestimmung einer axialen Position und/oder Lage eines beweglichen Bauteils, an diskreten Arbeitspunkten (z.B. Schalter) oder kontinuierlich.

Mit 3D-Hallsensoren können alle drei Feldkomponenten, mit der neuartigen Fraunhofer HallinOne®-Technologie [1] außerdem deren räumlichen Änderungen parallel zur Chipebene direkt an einem Punkt gemessen werden (Abbildung 1). Hierdurch eröffnet sich eine Vielzahl neuer Anwendungsmöglichkeiten. Gerade im Automotive-Bereich ist es oft notwendig, mehrere der sechs räumlichen Freiheitsgrade eines Starrkörpers (drei Richtungen, drei Winkel) zuverlässig zu bestimmen und/oder die Auswirkungen magnetischer und elektronischer Störungen zu kompensieren. Diese Probleme sind durch die zusätzlichen Messgrößen prinzipiell

lösbar, gleichzeitig steigt jedoch die Komplexität der zugehörigen Algorithmen zur Positionsbestimmung. Die Eindeutigkeit der Abbildung Position → Magnetfeld und damit deren für die Positionsbestimmung notwendige Invertierbarkeit ist im 3D nicht immer gegeben, die erzielbare Genauigkeit ist im invertierbaren Arbeitsbereich nicht gleichmäßig. Diese Eigenschaften sind beim Entwurf zuverlässiger 3D-Positionssensorsysteme zu berücksichtigen.

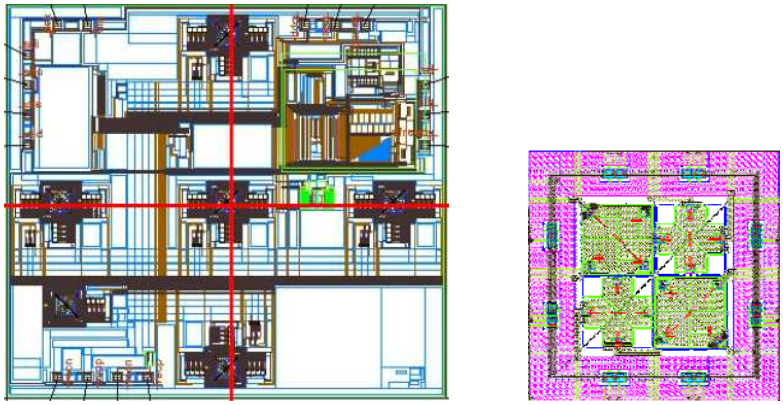


Abbildung 1 Fraunhofer HallinOne®-3D-Sensorchip (links), Detail 3D-Sensor (rechts)

2 Entwurfsaufgaben für hallbasierte 3D-Positionssensorik

Die wesentlichen System-Komponenten magnetischer Positionssensorsysteme sind die Hyperflächen der zu messenden Relativbewegung und -verdrehung, die Form und Größenordnung des Magnetfeldes sowie die Anzahl und Charakteristik der Messgrößen des Sensors. Für diese Elemente stellen sich Entwurfsaufgaben auf mehreren Ebenen der Produktentwicklung:

- Machbarkeits-/Konzeptstudien
- Konstruktion & Verifikation
- Systemoptimierung

Der erste Schritt von der Idee zu einem oder mehreren erfolgversprechenden Konzepten erfordert weniger detaillierte Modelle als schnelle Aussagen zu grundsätzlichen Fragestellungen. Nichtsdestotrotz sind bereits hier durchaus komplexe Zusammenhänge zu beleuchten. Im Fall eines hallbasierten Positionssensorsystems sind Fragen zu beantworten wie:

- Welche Messgrößen (B-Feldkomponenten, -Gradienten) sind notwendig, um die gestellte Aufgabe (2D-6D-Positionsbestimmung) eindeutig lösen zu können?
- Welche verfügbaren Sensortypen liefern diese Messgrößen mit welcher Genauigkeit?
- Welche Magnete liefern ein für die geforderte Aufgabe geeignetes Magnetfeld?
- Welche grundlegenden konstruktiven Varianten (Lage von Magnet und Sensor im Gesamtsystem und relativ zueinander) sind machbar und zielführend?
- Welche Auflösung ist prinzipiell mit einem bestimmten Konzept erreichbar?

Zur Beantwortung dieser Fragen benötigt der Entwerfer Flexibilität bei der Konfiguration der Relativbewegung, einfache Feldmodelle für die Flußdichte bei Standard-Magnetformen und stark vereinfachte Sensormodelle. Er wird zunächst auf die Berücksichtigung von Störfeldern verzichten und auch bei den Messgrößen schrittweise vom reinen B-Feld hin zu den zugehörigen Sensorsignalen nach den einzelnen Signalverarbeitungsstufen des Sensors vorgehen.

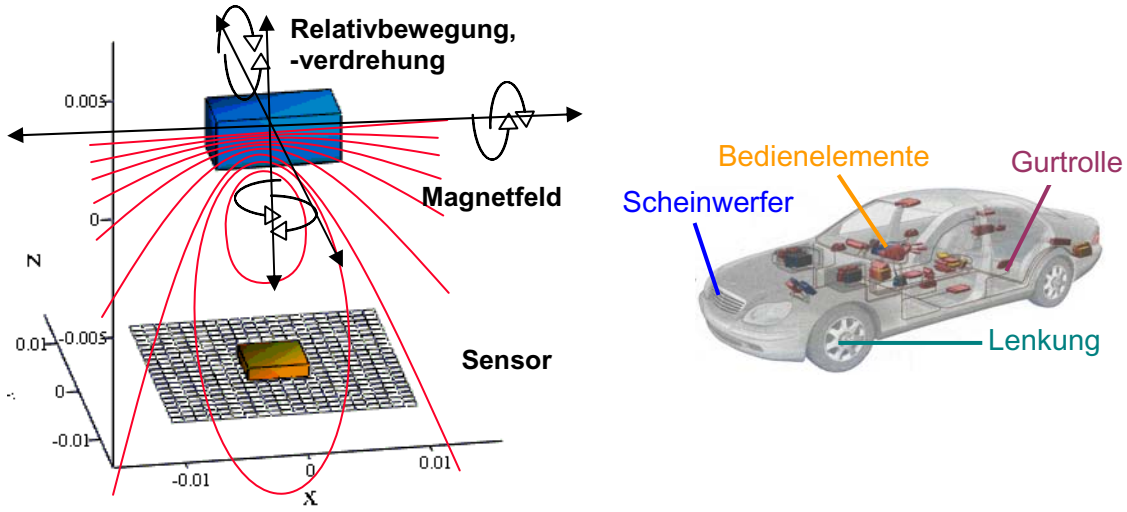


Abbildung 2 Entwurfs-Systemkomponenten für 3D-Hall-Positionssensorik (links), Ausgewählte mögliche Automotive-Applikationen (rechts)

Im zweiten Schritt sind die ausgewählten, erfolgversprechenden Konzepte zu detaillieren und im Detail zu verifizieren. Dies erfordert genauere, konzeptspezifische Modelle und in der Regel einen höheren Rechenaufwand. Die Ergebnisse dieser Modellierung sollen belastbare Aussagen zur Leistungsfähigkeit des gewählten Designs liefern. Gegebenenfalls sind spezielle Geometrien und permeable Materialen im Arbeitsbereich zu berücksichtigen, da diese Störfelder verursachen können, welche die Positionsberechnung zusätzlich erschweren.

Für die Entwicklung von Serienprodukten sollten in einem dritten Entwurfsschritt Feinabstimmungen und Optimierungen der Systemkomponenten hinsichtlich bestimmter Anforderungen erfolgen, etwa hinsichtlich geforderter Genauigkeit und/oder Kosten.

3 Entwurfumgebung für 3D-Hall-Sensorsysteme

Das Fraunhofer Institut für Entwurfsautomatisierung (IIS-EAS) hat eine Entwurfsumgebung entwickelt [2], die vor allem die erste und zweite Phase beim Entwurf halleffektbasierter 3D-Sensorsysteme unterstützt und im Folgenden in Grundzügen vorgestellt wird .

Die Entwurfsumgebung besteht aus drei Komponenten – der grafischen Benutzeroberfläche, einer Ablaufsteuerung sowie einem oder mehreren eingebundenen Simulatoren. Die beiden ersten Komponenten wurden zum Großteil in Java implementiert, für die Visualisierung im Rahmen der GUI wurde auf das Visualization Toolkit (VTK) zurückgegriffen. Die Komponenten kommunizieren untereinander über Java-basierte Datei-Schnittstellen.

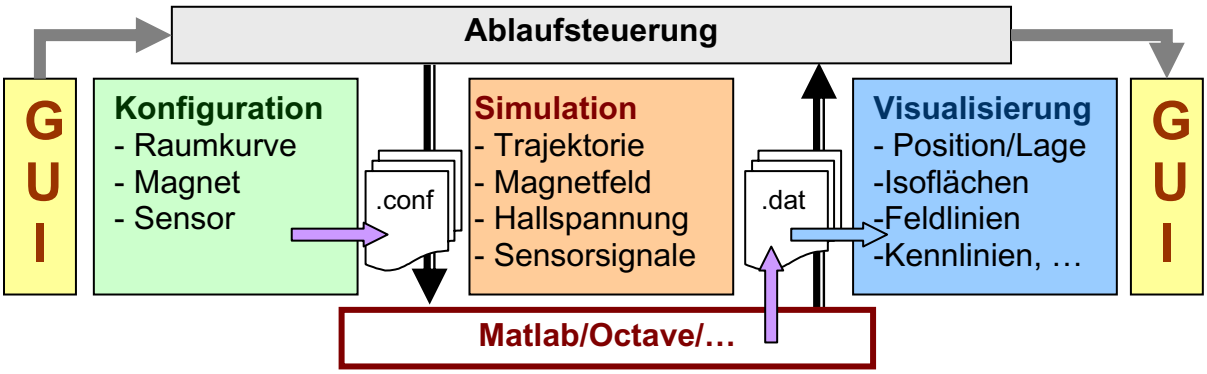


Abbildung 3: Ablaufsteuerung der Entwurfsumgebung

3.1 Experimentkonfiguration

Mit der grafischen Benutzeroberfläche lassen sich die zu entwerfenden Systemkomponenten – Raum der Freiheitsgrade, Magnetfeld und Hallsensor – auswählen und entsprechend der Anforderungen der speziellen Entwurfsaufgabe variieren. Außerdem werden die Simulatoren für jede Systemkomponente ausgewählt und konfiguriert.

Bei der Definition der Freiheitsgrade (Position & Lage) sind Aspekte der Modellierung und Beschreibung von Mehrkörpersystemen von Bedeutung. Grundsätzlich lassen sich Positionen in kartesischen oder anderen Koordinatensystemen in Kombination mit Eulerwinkeln (körperfeste Drehachsen) oder Roll-Pitch-Yaw-Winkeln (erdfeste Drehachsen) beschreiben, auch Quaternionen sind denkbar. Wichtig ist, dem Entwerfer Werkzeuge anzubieten, die sein Abstraktionsvermögen nicht bereits beim Mehrkörpersystem vollständig in Beschlag nehmen. Gleichzeitig soll eine große Flexibilität bei der Wahl der Bezugssysteme unterschiedlichen Gewohnheiten und Kenntnissen der Entwerfer entgegenkommen. In der Entwurfsumgebung wurden zunächst Eulerwinkel in einem magnetfesten Koordinatensystem implementiert, die Freiheitsgrade der beweglichen Systemkomponente sind vom Nutzer definierbar.

Zur Modellierung des Magnetfeldes stehen für Konzept- und Machbarkeitsstudien zunächst Funktionen zur analytischen oder quasi-analytischen Berechnung der räumlichen magnetischen Flußdichte für Permanentmagnete in Standardbauformen – Kugel, Quader, Zylinder – mit bis zu vier Polen zur Verfügung. Für die Konstruktions- und Validierungsphase des Systementwurfs können diese durch detailliertere FEM-Magnetfeldsimulationen ersetzt werden, die eine genauere Modellierung des Magnetfeldes erlauben. Ebenso können direkte Magnetfeldmessungen als Datenquelle eingebunden werden.

Das Sensormodell beinhaltet verschiedene Signalebenen, von der Hallspannung einzelner Hallelemente über diverse Signalverarbeitungsschritte, die auf dem realen Sensor unvermeidbar sind, jedoch die Signalqualität und damit die erzielbare 3D-Auflösung unmittelbar beeinflussen. Der Einfluss dieser Signalverarbeitung – Vorverstärker, A-D-Wandler – kann für die Modelle bestimmter, real vorliegender Sensor-ICs durch Vorgabe von Rausch- und Ansteuer-

parametern untersucht werden. Darüber hinaus ist es im gewissen Rahmen möglich, neue Sensormodelle zu generieren, um ggf. Entwurfsspielräume auch für anwendungsspezifische Sensoren zu untersuchen.

3.2 Simulation

Grundlage der Entwurfsumgebung ist ein mathematischer Formelinterpreter, der eine flexible Konfiguration und ggf. Neuerstellung von Varianten aller Systemkomponenten gestattet. Hierfür wurden MatLab™ und GNU-Octave ausgewählt, weitere sind möglich.

Nach Abschluss der Experimentkonfiguration werden die Simulatoren von der Entwurfsumgebung aus aktiviert. Sie generieren entsprechend der konfigurierten Entwurfsaufgabe Daten der Systemkomponenten, z.B. eine diskrete Repräsentation des Raums der Freiheitsgrade, zugehörige Magnetfeldwerte, entsprechende Sensormesswerte und jeweils davon abgeleitete Kennwerte, mit deren Hilfe der konkrete Entwurf bewertet werden kann.

Die Simulatoren werden über eine Ablaufsteuerung gekoppelt, welche auch die Interaktion über Datei-Schnittstellen sicherstellt. Durch diesen modularen Aufbau und definierte Datenschnittstellen (z.B. VTK) können die Ergebnisse sämtlicher Entwurfsschritte auch außerhalb der Entwurfsumgebung genutzt und weiterverwendet werden.

3.3 Visualisierung

Der Visualisierung der Simulationsergebnisse kommt für den Entwurfsprozess eine entscheidende Rolle zu. Sie soll dem Entwerfer das entworfene System entsprechend seiner Erfahrungsstufe auf verschiedenen Abstraktionsebenen transparent machen und ihm ermöglichen, die Charakteristik seines Systems schnell und fundiert kennenzulernen. Darüber hinaus soll sie erfahrenen Entwerfern erlauben, kritische Bereiche schnell zu identifizieren.

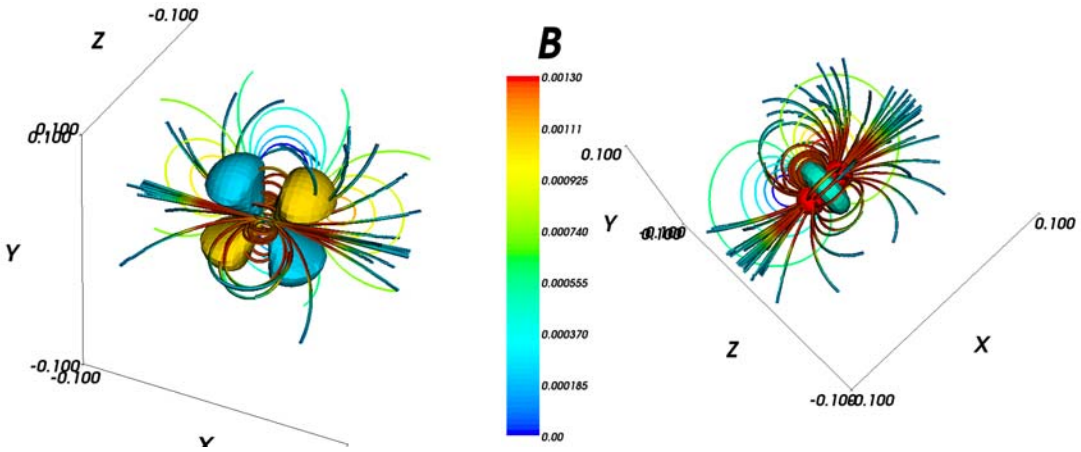


Abbildung 4: Magnetfeld-Visualisierung durch Stromlinien und Isoflächen

Zur Visualisierung des Magnetfeldes wie auch der Sensormesswerte kann der Entwerfer zwischen verschiedenen, anpassbaren Darstellungsarten wählen. Abbildung 5 zeigt dies beispielhaft. Links oben ist zunächst die Lage und Ausrichtung der beweglichen Systemkomponente (Magnet oder Sensor) an diskreten Punkten im Raum auf der Hyperfläche der Relativbewe-

gung ablesbar - hier einer Viertel-Kugelschale. Rechts daneben kann sich der Entwerfer anhand gleichmäßig verteilter Isolinien einen Überblick über die Stärke der Variation aller drei Flußdichtekomponenten verschaffen, die einen ersten Aufschluss über die Invertierbarkeit der Position anhand des gemessenen Magnetfeldes erlaubt. Unten sind dieselben Komponenten im Raum der Freiheitsgrade dargestellt und entsprechend ihrer Werte eingefärbt. Daraus kann der Entwerfer u. a. ablesen, in welchen Bereichen er die Einhaltung der Genauigkeitsanforderungen zu überprüfen ist.

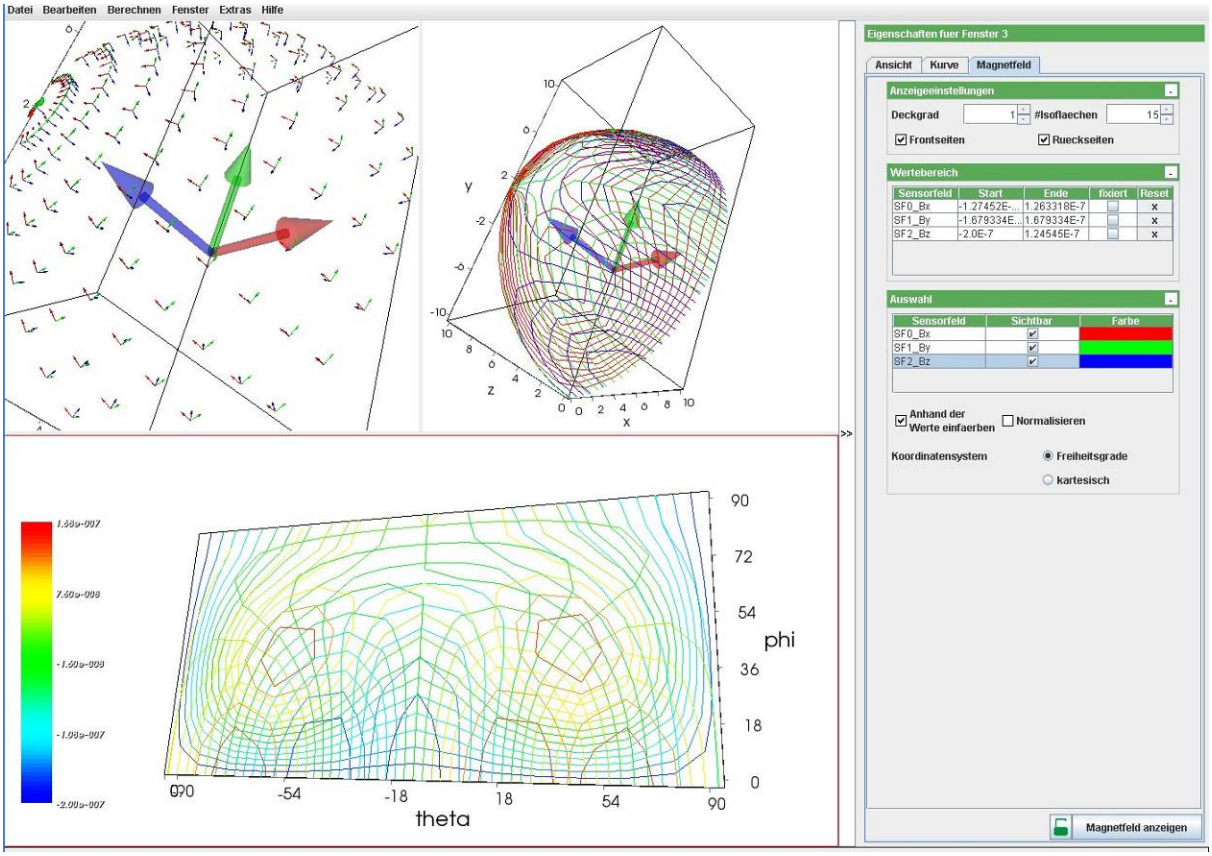


Abbildung 5: Entwurfsunterstützung durch Simulationsergebnisse

Die vorgestellte Entwurfsumgebung entsteht am Fraunhofer Institut für Integrierte Schaltungen, Institutsteil Entwurfsautomatisierung Dresden im Rahmen eines unter dem Themenschwerpunkt „Mikrosystemtechnik für Fahrerassistenz-Systeme“ vom BMBF geförderten Forschungsprojekts.

Literatur

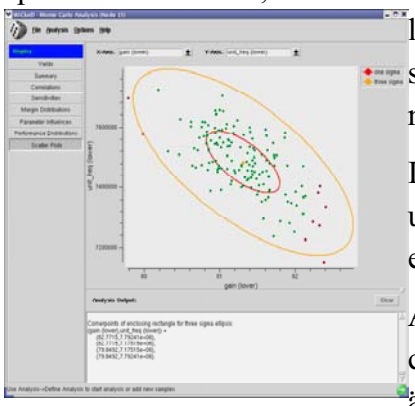
[1] Sauerer, Josef & Hohe, Hans-Peter: *Positionssensorik mit mehrdimensionalen Hallelementen*, Sensorik Aktuell II/2006, AMA Fachverband für Sensorik, 2006

[2] Wilde, A. et al.: *Design of Position Sensor Systems based on 3D Hall probe*. Proc. Sensor+Test 2007, Nürnberg, 2007

Statistische Analyse Verfahren für integrierte Analog und Mixed-Signal Schaltungen

Matthias Sylvester – MunEDA GmbH

Noch vor wenigen Jahren reichten selbst bei Analogdesigns oft Cornersimulationen aus, um Schaltungen mit hohem Yield zu erzeugen. Mit zunehmender Miniaturisierung der Prozesse spielen Phänomene, die früher ignoriert werden konnten, eine große Rolle. Insbesondere



lokale Prozeßschwankungen (Mismatch) wirken sich umso stärker aus, je kleiner die Prozesse werden. Diese lassen sich nur mit statistischen Methoden überprüfen.

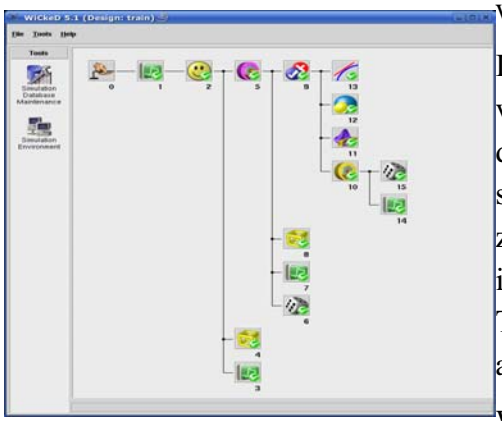
Ist es sinnvoll, Analogblöcke auf Ihre Zuverlässigkeit zu untersuchen, wo doch meist viel mehr digitale Elemente auf einem Chip sind?

Auf einem SoC (System-on-Chip) sind heute üblicherweise ca. 10% analoge Transistoren und 90% digital, in der Fläche ändert sich das Verhältnis auf 30% : 70%, aber bei den Ursachen für Redesigns ist das Verhältnis 50% : 50%. Es lohnt sich also, Analogschaltungen auf ihre Zuverlässigkeit hin genauer zu untersuchen.

Eine Schaltung wird dann als zuverlässig erachtet, wenn die Wahrscheinlichkeit sehr hoch ist, daß diese Schaltung bei realen Prozeßschwankungen (global und lokal) sowie über den vollen Bereich der Umgebungsbedingungen (z.B. Temperatur und VDD) die Spezifikationen erfüllt. Oft wird ein 3-Sigma-Design als hinreichend zuverlässig betrachtet, d.h. das Design erfüllt auf dem Silizium mit einer Wahrscheinlichkeit von 99,9% die Spezifikationen.

Kleine Prozeßstrukturen erlauben auch, eine wesentlich größere Anzahl von Schaltungen auf einem Chip unterzubringen. Oft werden beispielsweise 1000 oder sogar 10000 Senseamps in Speichern benötigt. Bei einem 3-Sigma-Design ist der erzielbare parametrische Yield (Prozentsatz guter Chips in der Produktion) bei 1000 Senseamps bei ca. 36%, bei 10000 Senseamps sogar bei nur 45ppm. Eine Erhöhung der Wahrscheinlichkeit auf 99,997% (4-Sigma-Design) für einen Senseamp resultiert bei 10000 Senseamps in einem Yield von 74%. Erst bei 5-Sigma-Designs oder gar 6-Sigma-Designs ist der Yield zufriedenstellend. Wollte man so ein Design mit Monte-Carlo-Methoden verifizieren, würde man für eine Corner bei 3-Sigma mindestens 3000 Simulationen benötigen, 4-Sigma benötigen 100000 Simulation und 6-Sigma liegen bei 3000000000 Simulationen.

Bislang gab es keine wirklich gute und schnelle Möglichkeit außerhalb von Redesigns, den Yield frühzeitig vorherzusagen. Schlechte Yields versuchte man durch Prozeßtuning in den Griff zu bekommen, aber wenn das funktioniert, dann höchstens bei homogenen Strukturen wie RAMs, nicht jedoch bei SoC's.



In diesem Vortrag werden kurz die Grundlagen vorgestellt, mit denen MunEDA's Toolset WiCkeD in der Lage ist, den parametrischen Yield einer Schaltung schon nach dem Schematic Entry (!) zu analysieren und zuverlässig vorherzusagen. Darüber hinaus ist WiCkeD in der Lage, die Designparameter (z.B. Transistorlängen und -weiten) der Schaltungen automatisch zu optimieren.

WiCkeD ist intuitiv bedienbar und kann sowohl aus dem Cadence ADE®, Mentor ICStudio® als auch im Standalonemode auf Basis der Netzliste betrieben werden. Derzeit werden die Spice-Simulatoren Spectre®, Eldo®, Hspice® sowie Titan® unterstützt. WiCkeD läuft auf SUN® und auf Linux. WiCkeD wird normalerweise über ein GUI bedient, kann aber auch mit Skripten betrieben werden.

Laut Aussage von Designern kann die Optimierungsphase eines Analogmoduls von typischen 4 - 12 Wochen oft auf 1 – 5 Tage reduziert werden. Entscheidend trägt dazu die Worst Case Analyse bei, die eine zuverlässige Yieldabschätzung erlaubt, unabhängig davon, wieviele Sigma Yield nachgewiesen werden sollen.

Die Yieldabschätzungen wurden vielfach durch Silizium bestätigt.

Einsatz von Leveled-Noise-Verfahren beim Computer Aided Robust Design

Andreas Burblies, Fraunhofer IFAM
andreas.burblies@ifam.fraunhofer.de
Natalia Reichert, Fraunhofer IFAM
natalia.reichert@ifam.fraunhofer.de
Matthias Busse, Fraunhofer IFAM
matthias.busse@ifam.fraunhofer.de

Zusammenfassung

Um den Einfluss von nicht kontrollierbaren Variationen und Unbestimmtheiten bei der numerischen Simulation technischer Systeme zu berücksichtigen, können zur Versuchsplanung Orthogonal Arrays eingesetzt werden. Es wird gezeigt, wie man spezielle Leveled-Noise-Verfahren für Zufallsvariablen, Zufallsprozesse und Zufallsfelder verwendet, um Versuchspläne zu realisieren. Anhand praktischer Beispiele aus der Mechanik werden die Vorteile der eingesetzten Methoden gegenüber Monte-Carlo-Verfahren verdeutlicht und die Anwendung der Taguchi-Methode zum Robust Design dargestellt.

1 Einführung

1.1 Computer Aided Robust Design

In der Realität gleicht kein Produkt oder Prozess dem anderen. Um dieses Realitätsmerkmal auf numerische Simulationsmodelle zu übertragen, müssen stochastische Methoden eingeführt werden. Diese berücksichtigen Unbestimmtheiten, Variationen und Fehler, so dass auch kein Simulationsergebnis dem anderen exakt gleicht. Für Simulationsexperimente sind dann Versuchsplanung und statistische Auswertungen notwendig. Weiterhin lässt sich ein virtuelles Produkt oder ein virtueller Prozess auf statistische Zielgrößen, wie zum Beispiel das Signal-Rausch-Verhältnis, optimieren. Taguchi [1] spricht dabei vom sogenannten Robust Design und wendet zur Versuchsplanung Orthogonal Arrays an. Nimmt man an, dass ein System durch eine stetige, reellwertige Funktion einer endlichen Anzahl k von Variablen oder Systemfaktoren x_i modelliert werden kann:

$$f : R^k \to R, (x_1,...,x_k) \to f(x_1,...,x_k) \tag{1}$$

so gibt Phadke [2] drei unterschiedliche Methoden zur Simulation varierender Systemfaktoren an. Dabei werden die Systemfaktoren als reelle Zufallsvariablen X_i mit einer Verteilungsfunktion $P_{xi}(x)$ aufgefasst.

1.2 Monte-Carlo-Simulation

Bei der Monte-Carlo-Simulation erfolgt zunächst die Realisation der Systemfaktoren X_i durch entsprechende Verfahren, zum Beispiel mittels der Inversionsmethode [3]:

$$X_i = P_{Xi}^{-1}(U) \tag{2}$$

wobei U die Realisation einer gleichförmig verteilten Zufallsvariable zwischen 0 und 1 ist, die mit Hilfe eines Zufallszahlengenerators erzeugt werden kann. Mit diesen Realisationen erfolgt die Berechnung der Systemantwort für ein Simulationsexperiment. Die Monte-Carlo-Simulation erfordert jedoch mehr als hundert Simulationsexperimente, um eine vernünftig auswertbare Statistik zu erzeugen.

1.3 Taylor-Reihen-Ansatz

Für die First-Order-Taylor-Methode wird angenommen, dass die Systemfaktoren normalverteilt und durch entsprechende Mittelwerte μ_i und Varianzen σ_i^2 charakterisiert sind. Der Mittelwert der Systemantwort ist dann:

$$\mu = f(\mu_1,...,\mu_k) \tag{3}$$

und die Varianz der Systemantwort lässt sich abschätzen durch:

$$\sigma^2 = \sum_{i=1}^{k}\left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_i^{\,2} \tag{4}$$

Die partiellen Ableitungen der Systemfunktion können numerisch bestimmt werden. Dafür sind in der Regel nur *k+1* Berechnungen der Systemantwort notwendig. Die First-Order-Taylor-Methode kann allerdings nur angewendet werden, wenn die Korrelationen zwischen den Systemfaktoren vernachlässigbar und die Varianzen sehr klein sind. Auch bei starker Nichtlinearität der Systemfunktion ist die Methode nicht hinreichend genau. Dann müssen höhere Talyor-Ansätze verwendet werden.

1.4 Orthogonal Array basierte Simulation (Leveled Noise Verfahren)

Bei der Orthogonal Array basierten Simulation werden die Systemfaktoren über zwei oder drei Level realisiert:

Zwei-Level-Methode $\quad X_i^0 = \mu_i - \sigma_i, \quad X_i^1 = \mu_i + \sigma_i$

Drei-Level-Methode $\quad X_i^0 = \mu_i - \sqrt{3/2}\sigma_i, \quad X_i^1 = \mu_i, \quad X_i^2 = \mu_i + \sqrt{3/2}\sigma_i$

Wählt man eine der beiden Methoden, so ergibt sich bei einer beliebigen Zusammenstellung der gleichen Anzahl von Level für einen Systemfaktor X_i immer ein Mittelwert von μ_i und eine Varianz von σ_i^2.
Anschließend wird ein spezieller statistisch fundierter Versuchsplan gewählt. Die Anzahl der Experimente N hängt von der Anzahl der Systemfaktoren k, der Anzahl der Level s und der Anzahl der Systemfaktorinteraktionen t ab. Diese sogenannten Orthogonal Arrays haben die Eigenschaft, dass eine bestimmte Level-Kombination von t beliebigen Systemfaktoren gleich häufig auftritt, zum Beispiel für das Orthogonal Array $OA(N,k,s,t)=OA(8,4,2,2)$:

Experiment #	Faktor 1	Faktor 2	Faktor 3	Faktor 4
1	0	0	0	0
2	0	0	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	1	0
6	1	0	1	1
7	1	1	0	0
8	1	1	0	1

Tabelle 1: Orthogonal Array OA (8,4,2,2)

Untersucht man beliebig paarweise ($t=2$) Spaltenkombinationen, ergibt sich, dass jede Level-Kombination (0,0), (0,1), (1,0) und (1,1) genau zweimal auftritt. Für die Level der Faktoren in einem Orthogonal Array werden anschließend die Level-Werte der Systemfaktoren eingesetzt (s.o.). Man erhält dann einen statistisch gut ausbalancierten Versuchsplan. Viele Orthogonal Arrays sind vorberechnet und können aus der Fachliteratur [2,4] entnommen werden.

2 Neue Verfahren

2.1 Kombination von Monte-Carlo und Leveled-Noise-Verfahren

In der Tabelle 1 erkennt man, dass der statistische Mittelwert und die Varianz für jeden Systemfaktor der Vorgabe entspricht, die Realsisation der Level aber trotzdem immer diskret und gleichverteilt sind. Um die statistischen Verteilungen der Systemfaktoren besser zu berücksichtigen, ist es möglich, für die Level gleichflächige (gleichwahrscheinliche) Bereiche unter der Wahrscheinlichkeitsverteilung zu wählen und diese später entsprechend zu realisieren (s. Abbildung 1).

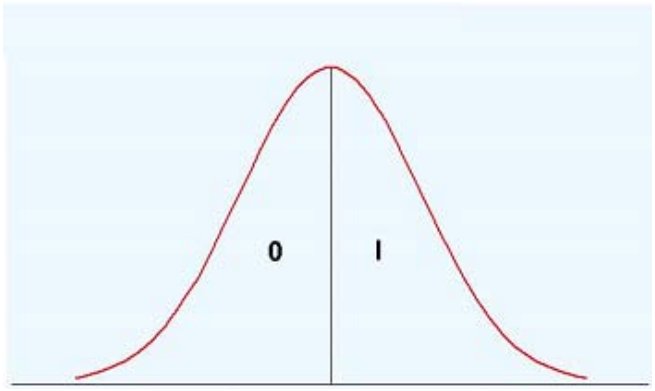


Abbildung 1: Level-Bildung (s=2) einer normalverteilten Zufallsvariablen

Definiert man für jeden Systemfaktor X_i der Tabelle 1 eine normalverteilte Zufallsvariable mit $\mu_i = 0$ und $\sigma_i = 1$, so ergibt sich zum Beispiel folgende Monte-Carlo-Realisation des Versuchsplans:

Experiment #	Faktor 1	Faktor 2	Faktor 3	Faktor 4
1	-1,209	-0,71301	-0,80034	-1,4833
2	0,86593	-0,2745	-1,3732	0,6766
3	-1,211	1.6293	-1,0405	0,1771
4	-1,1455	-2,5204	0,34927	-0,039061
5	0,68524	1,0095	-0,2332	-0,56265
6	0,093389	-0,074291	0,90197	1,4694
7	-0,74187	0,51531	2,205	0,61847
8	1,5494	1,0374	0,22802	-1,9776

Tabelle 2: Monte-Carlo-Realisation des Orthogonal Array OA (8,4,2,2)

Level 0 entspricht hier den negativen und Level 1 den positiven Realisationen der Systemfaktoren. Wegen des Einsatzes der Monte-Carlo-Realisation sollte man die Anzahl der Experimente erhöhen. Das kann durch Ergänzung einer wiederholten Realisation des Versuchsplans oder des komplementären Versuchsplans (Level $L^* = (s\text{-}1)\text{-}L$) geschehen.

2.2 Behandlung von Zufallsprozessen und Zufallsfeldern

Unbestimmte Randbedingungen wie Temperaturschwankungen oder die örtliche Variation von Materialeigenschaften lassen sich nicht durch einfache Zufallsvariablen realisieren. Bei einem Zufallsprozess (zeitliche Folge einer Zufallsgröße) oder einem Zufallsfeld (örtliche Verteilung einer Zufallsgröße) hängt die Wahrscheinlichkeit häufig von der näheren zeitlichen bzw. örtlichen Umgebung ab. Dies kann durch Markov-Prozesse oder Markov-Felder beschrieben werden [5]. Sind X_i Zufallsvariablen die örtlich im Raum verteilt sind, so ist die Wahrscheinlichkeit, dass die Zufallsvariable am Ort x den Wert X annimmt, $P_X(x)$ abhängig von den nächsten Nachbarn $c \in C$:

$$P_X(x) \propto \prod_{c \in C} f_c(x_c) \tag{5}$$

$P_X(x)$ lässt sich dann formulieren als:

$$P_X(x) = \frac{1}{Z} \exp(-\sum_c V_c(x_c)) \tag{6}$$

Es ist die unter Physikern bekannte Gibbs- oder Boltzmann-Verteilung mit dem Wechselwirkungspotenzial $V_c(x_c)$, der Wechselwirkungsenergie $U(x) = \sum_c V_c(x_c)$ und der Partitionsfunktion Z. Wählt man zum Beispiel:

$$U(x) = -\beta_1 \sum_{i \perp j} X_i X_j - \beta_2 \sum_{i \| j} X_i X_j \tag{7}$$

wobei $X=\{-1,1\}$[1] ist, so ergeben sich für verschiedene Parameter $\beta_1=\beta_2$ folgende Realisationen

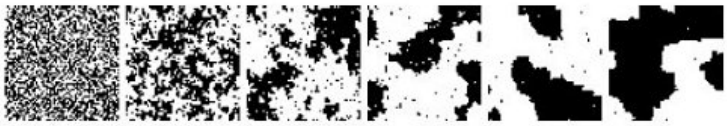


Abbildung 2: Realisationen eines 2D-Ising-Modells [5]

Wie in 2.1 beschrieben können auch hier für die örtlichen Verteilungen gleichwahrscheinliche Level gebildet und Versuchspläne auf Basis von Orthogonal Arrays realisiert werden.

3 Anwendungsbeispiele

3.1 Steifigkeit einer Tragstruktur bei Variation von Geometrie und Werkstoff

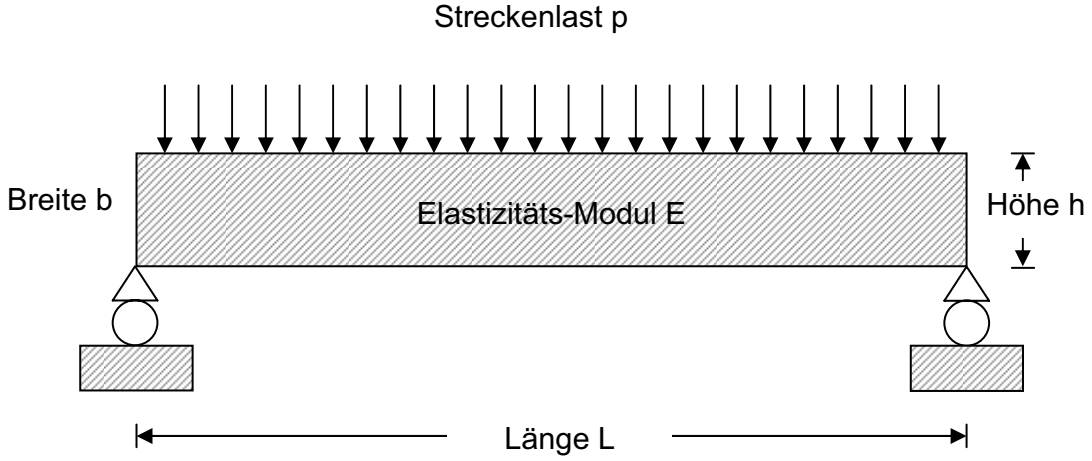


Abbildung 3: Lose gelagerter Träger unter Streckenlast

[1] Ising-Modell

Die maximale Durchbiegung w_{max} eines lose gelagerten Trägers (Abbildung 3) kann theoretisch berechnet werden:

$$w_{\max} = \frac{60 \cdot p \cdot L^4}{384 \cdot E \cdot b \cdot h^3} \tag{8}$$

Eine Stahlplatte hat eine Abmessung von $L = 5000\ mm$, $b = 2000\ mm$ und $h = 20\ mm$. Der E-Modul beträgt $E = 210000\ MPa$. Sie ist durch eine Streckenlast $p = 2\ Nmm^{-1}$ belastet. Für diese Werte ergibt sich eine maximale Durchbiegung von $w_{\max} = 58{,}1287\ mm$.

Um den Einfluss von Variationen in Geometrie und Werkstoff zu untersuchen, werden die Abmessungen L, b und h sowie der E-Modul E durch normalverteilte Zufallsvariablen ersetzt:

$\mu_L = 5000\ mm$	$\sigma_L = 1\ mm$
$\mu_h = 2000\ mm$	$\sigma_h = 1\ mm$
$\mu_b = 20\ mm$	$\sigma_b = 0{,}1\ mm$
$\mu_E = 210000\ MPa$	$\sigma_E = 5000\ MPa$

Folgende Methoden wurden angewendet, um Mittelwert und Varianz der maximalen Durchbiegung zu bestimmen:

Methode	μ_{wmax}	σ_{wmax}	Anzahl Experimente
Monte Carlo	58,3 mm	1,6737 mm	1000
Monte Carlo	58,4 mm	1,7097 mm	100
Taylor Series Expansion	58,1 mm	1,6346 mm	5
Orthogonal Array (2 Level)	58,2 mm	1,7530 mm	8
Orthogonal Array + Monte Carlo (5 Level)	58,2 mm	1,7057 mm	25

Tabelle 3: Vergleich verschiedener Simulationsmethoden

Die Ergebnisse zeigen, dass die First-Order-Methode und das Orthogonal-Array-Verfahren mit einer äußerst geringen Anzahl von Simulationsexperimenten zu einer guten statistischen Abschätzung der Systemantwort führen. Probleme gibt es, wenn die Systemantwort stark nichtlinear ist, oder die Eingangsvariationen sehr groß sind (Six-Sigma). Hier empfiehlt sich ein gemischtes Verfahren aus Orthogonal Array und Monte Carlo.

3.2 Robust Design einer Tragstruktur mittels ANOM-Verfahren

Am Fraunhofer-Institut für Angewandte Materialforschung wurde in den letzten Jahren ein Programm MPTO (Multi Phase Topology Optimisation) entwickelt [6,7], das für mechanische Strukturen unter Belastung verschiedene Werkstoffe so verteilt, dass die Steifigkeit maximal bzw. die innere elastische Energie minimal wird. Die Ergebnisse können mit der Verteilung von unterschiedlich porösem Knochenschwamm (Spongiosa) in mechanisch belasteten Knochenstrukturen verglichen werden. Die Optimierung kann jedoch nur für einen

repräsentativen Lastfall erfolgen. Interessant ist die Frage, welche Struktur ist optimal für variierende Lastfälle. Dazu wurde die Streckenlast der Tragstruktur in Abbildung 3 in vier gleichartige Wirkungsbereiche aufgeteilt und für die vier Lasten ein statistischer Versuchsplan für eine Orthogonal Array basierte Simulation mit der 3-Level-Methode ermittelt. In Tabelle 4 sind die Ergebnisse der Topologieoptimierung dargestellt. Die 9 Topologien wurden nun anhand aller Lastfälle mittels Finite-Elemente-Analyse analysiert und der Mittelwert E und die Varianz σ^2 der inneren elastischen Energie berechnet. Zur Bestimmung der optimalen Struktur kann das Signal-Rauschverhältnis E^2/σ^2 als Robustheitsmaß [1] betrachtet werden. Robust Design bedeutet, die maximale Robustheit zu finden.

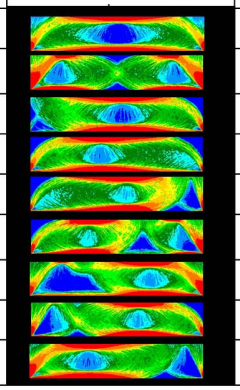
Exp.	Last 1	Last 2	Last 3	Last 4		Statistische Auswertung	Energie E	Varianz σ^2	Robustheit
1	1	1	1	1		9 exp.	1663	3707021	0,75
2	1	0	0	-1		9 exp.	4212	32157740	0,55
3	1	-1	-1	0		9 exp.	2321	8120940	0,66
4	0	1	0	0		9 exp.	1820	4543947	0,73
5	0	0	-1	1		9 exp.	2103	7136125	0,62
6	0	-1	1	-1		9 exp	2681	12156119	0,59
7	-1	1	-1	-1		9 exp.	2271	8250974	0,63
8	-1	0	1	0		9 exp.	2389	9550714	0,60
9	-1	-1	0	1		9 exp.	2531	11612738	0,55

Tabelle 4: Versuchsplan für das Robust Design einer Tragstruktur

Offensichtlich ist für den vorliegenden Versuchsplan die Robustheit des Experiments Nr. 1 maximal. Da es aber sein kann, dass außerhalb des Versuchsplans noch eine Lastvariation existiert, die zu einer Struktur mit höherer Robustheit führt, muss dies untersucht werden. Dies kann mit der ANOM-Methode (Analysis of Means) erfolgen. Dazu wird für jede Last und für jeden Last-Level der Mittelwert der zugehörigen Robustheitswerte berechnet und tabellarisch dargestellt:

Faktor/Level	-1	0	1
Last 1	0,59	0,65	0,64
Last 2	0,6	0,59	0,70
Last 3	0,64	0,61	0,65
Last 4	0,59	0,66	0,64

Tabelle 5: Auswertung der Faktoreffekte durch die ANOM-Methode

Daraus ergibt sich, dass die Lastfallvariation (0,1,1,0), den größten Effekt auf die Robustheit hat. Dieser Lastfall ist im ursprünglichen Versuchsplan nicht enthalten und muss zusätzlich analysiert werden. Abbildung 4 zeigt das Ergebnis.

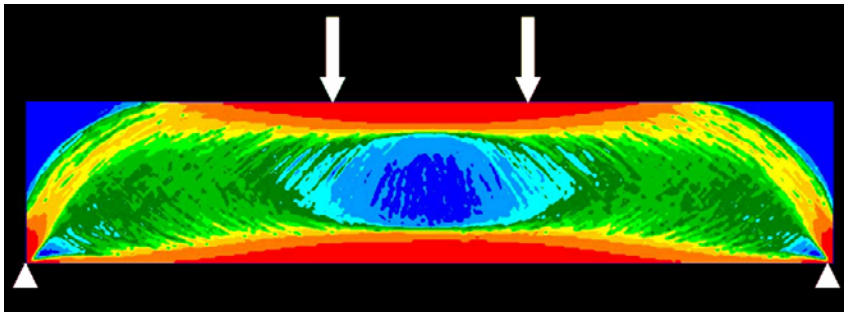


Abbildung 4: Optimale Topologie der Tragstruktur durch Robust Design

Auch die zusäztliche Struktur wurde mit allen 9 ursprünglichen Lastfällen getestet. Die Analyse ergibt folgende Werte:

$$\textit{Energie } E = 1782, \qquad \textit{Varianz } \sigma^2 = 4082728, \qquad \textit{Robustheit } R = E^2/\sigma^2 = 0{,}78$$

Die Topologie der Abbildung 4 ist demnach am unempfindlichsten gegenüber äußerer Lastfallvariationen und hat damit die größte Robustheit. Robustheit ist eine statistische Größe und kann damit virtuell nur unter Verwendung stochastischer Simulationsmethoden ermittelt werden.

Literatur

[1] Taguchi, Wu, Yu-In: *Introduction to Off-Line Quality Control*, American Supplier Institute, Dearborn, 1979

[2] Phadke: *Quality Engineering Using Robust Design*, Prentice-Hall, Englewood Cliffs, 1989

[3] Devroye: *Non-Uniform Random Variate Generation*, Springer, New York, 1986

[4] Hedayat, Sloane, Stufken: *Orthogonal Arrays*, Springer, New York, 1999

[5] Pérez: *Markov Random Fields and Images,* CWI Quarterly, Vol. 11 (4), 1998

[6] Burblies: *Topology Optimisation with Metallic Foams,* Cellular Metals and Metal Foaming Technology, Int. Conference, MIT-Verlag, Bremen, 2001

[7] Burblies, Busse: *Computer Based Porosity Design by Multi Phase Topology Optimisation*, Proceedings of Multiscale & Functionally Graded Materials Conference, Honolulu, 2006

Yield prediction in an automotive MEMS application with help of statistical analysis packages SAE J2748

Dirk Dammers, Daniel Schollän, Lars M. Voßkämper
Dolphin Integration GmbH
mems@dolphin-integration.com

Abstract

This paper shows a methodology for the yield prediction of Micro Electro Mechanical Systems (MEMS). Variations of process properties during the production of MEMS are considered in system simulations. Yield prediction based on the distribution functions provided by statistical analysis packages SAE J2748 is demonstrated on a capacity based accelerometer system.

Key words: MEMS, statistical distribution, Monte Carlo analysis, yield prediction

1 Introduction

The fabrication of MEMS underlies manufacturing process variations like geometry dimension variations of structures i.e. due to under etching, material property variation i.e. due to doping/temper temperature variations, etc. So the yield depends on the nominal - the optimal – design within its parameter tolerance ranges and distribution functions. Today there is a strong demand on decreasing structure sizes of micro-systems like MEMS. Due to this the tolerance range of the fabrication process decreases too. The production has to become more accurate. The tolerance range of each process step has to be well defined. Running a Monte Carlo simulation of the complete MEMS with realistic variations of the parameters considering their distributions helps to determine the yield of the production.

2 The MEMS model

An accelerometer was chosen as demonstrator for the methodology. Figure 1 shows a scheme of a typical single-axis airbag sensor. A seismic mass m is connected through a meander spring to the housing and so builds a spring-mass-damper system. Conductive plates are attached to the substrate on one side. With the counter plates on the other side attached to the movable mass, they build capacitors. These capacitors change capacitance according to the position of the seismic mass in non-linear relation to an external acceleration applied to the device. The resulting change in capacitance is detected and processed by some connected electronics.

For multi-domain or mechatronic simulations, the hardware description language VHDL-AMS (Very high speed Hardware Description Language – Analog and Mixed Signal) is best suited. With VHDL-AMS, it is not only possible to model and simulate time discrete systems but also to model and simulate systems with time continuous behavior; mixed signal circuits like ADC, DAC, PLL or systems of multiple domains like MEMS, MOEMS, MST, µLab. Furthermore, the language offers constructs to handle discontinuities, like they often occur in mechatronic systems. So this language was chosen for implementing the model of the accelerometer.

The SAE J2748 distribution functions, which are also implemented in VHDL-AMS, are used to provide parameter variations with defined stochastic system functions for the geometrical dimensions of the MEMS structure and its material properties. Furthermore, the same method can be applied to the parameters of the electrical circuit for considering their variations in system simulation.

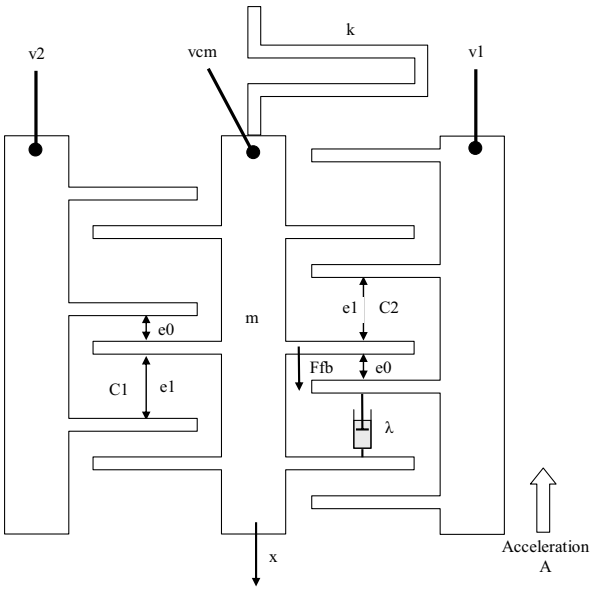


Figure 1: accelerometer scheme, x-axis

2.1 Basic physical effect implementation

The accelerometer is modeled with the basic physical effect approach, i.e. those basic physical effects like the inertia of masses in translational moved bodies, the spring effect of a beam or the electro-static attraction effect between two conductive plates, are modeled in VHDL-AMS and made available in the library EMBLEM-Mecha. To build the model of the complete MEMS device, the effect models have to be connected together (see Figure 2). For this, strict port compatibility has to be used for the “through” and “across” variables (see Table 1).

Type of variable	electrical network	rotational	translational
"through" variable	current i	torque T	force F
"across" variable	voltage v	angle φ	displacement x

Table 1: Variable types of the implemented ports for each domain

The implemented effects are:

- The inertia effect of a translational moved mass. This effect model is instantiated 5 times: for the seismic mass and for the movable and stationary fingers of each of the two capacitors. Equation (1) shows the equation to calculate the mass of a cubic body from the geometrical dimensions and material constants. Equation (2) shows the resulting inertia force of a translational moved mass. With: length l, width w, depth d, specific weight ρ, mass/hole ratio r.

$$m = l * w * d * \rho * \frac{r}{r+1} \tag{1}$$

$$f = m * a = m * \ddot{x} \tag{2}$$

- The spring effect equation to calculate the spring constant of a meander beam from the geometrical dimensions and material constant Emod (elastic modulus of the meander beams) is shown in equation (3). Equation (4) is the corresponding equation for the resulting spring force.

$$k = \frac{d * w^3 * E_{\mathrm{mod}} * n_m}{l^3 * n_b} \tag{3}$$

$$f = k * x \tag{4}$$

- The damping constant d can be determined by the reciprocal value of the quality factor Q, equation (5). Q itself can be calculated with the resonance frequency f0 and the bandwidth B, which can be captured by measurements. The damping force can be calculated with the equation (6); d is the damping constant.

$$d = \frac{1}{Q} = \frac{B}{f_0} \tag{5}$$

$$f = d * v = d * \dot{x} \tag{6}$$

- The capacity of the asymmetrical comb structure can be calculated with help of equation (7) and the geometry dimensions (etch depth d, gap_1, gap_2, overlap length o of the facing fingers and the number of comb fingers n and material parameter (relative permittivity of the material ε_r).

$$C = n * d * o * \varepsilon \left(\frac{1}{gap_1 - x} + \frac{1}{gap_2 + x} \right) \tag{7}$$

- The electrostatic attraction force of the asymmetrical comb drive can be calculated with help of the geometrical dimensions and material constants (etch depth d, gap_1, gap_2, overlap length o of the facing fingers and the number of comb fingers n and material parameter (relative permittivity of the material ε_r) and applied voltage V see equation (8).

$$f = \frac{1}{2} * n * d * o * \varepsilon * V^2 (gap_1 + gap_2) \left(\frac{gap_1 - gap_2 - 2x}{gap_1 * gap_2 + (gap_1 - gap_2) * x^2} \right)^2 \tag{8}$$

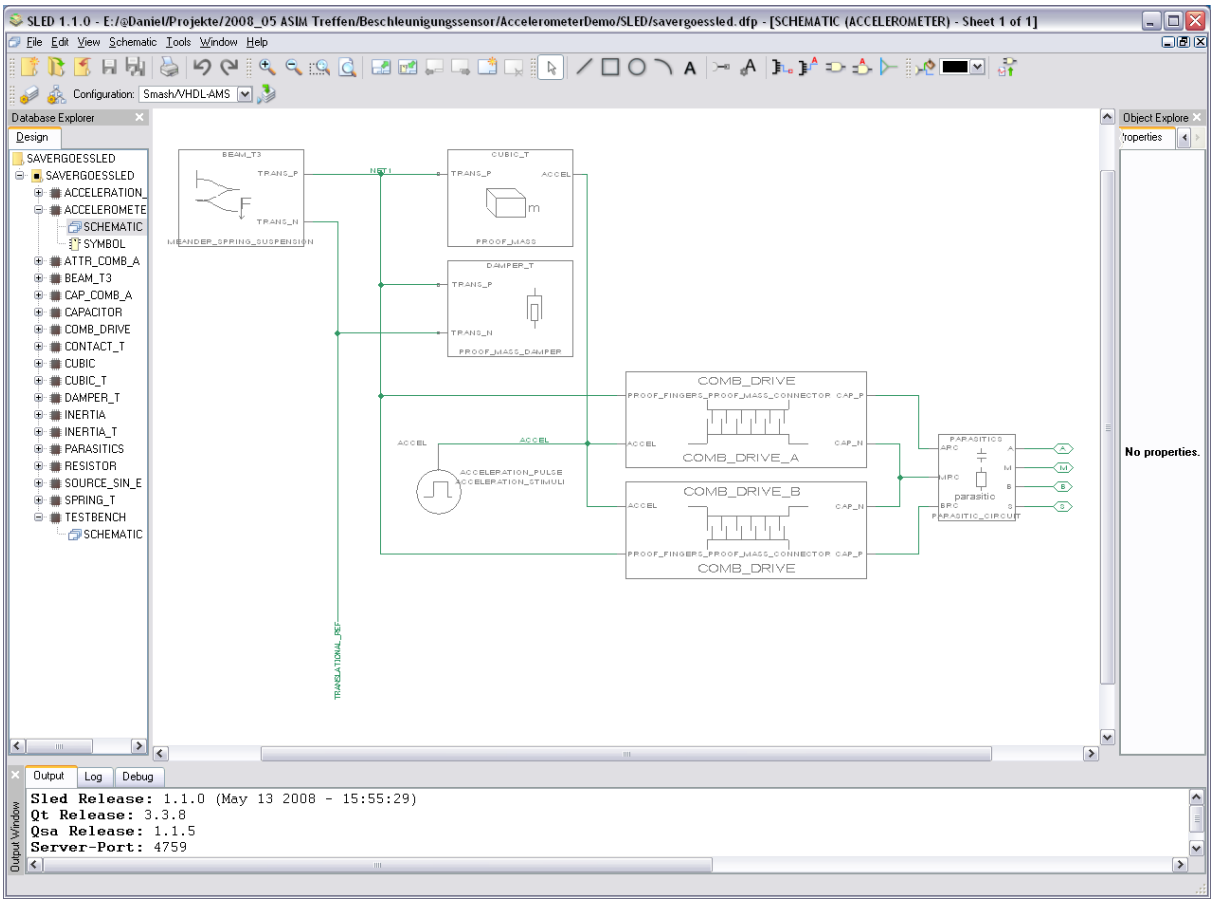


Figure 2: accelerometer scheme, x-axis

2.2 Applying the distributions

The variation of the MEMS structures geometry elements in x,y-direction is applied with Normal (Gaussian) distribution and different tolerances. Since the design of the chosen MEMS structure is insensitive to thickness variation of the structures, no distribution function is applied here. For simplification, the material parameters specific weight ρ, the dielectric constant ε_r and the elastic modulus of the beam E_{mod} are supposed to be well controllable in the process and are considered constant in this simulation. The Normal distribution function of the SAE J2748 packages is applied on the parameters in the accelerometer architecture, which are then passed to the effect model instances through the generic maps.

3 Monte Carlo simulation

The above described model is used to determine the resonance behavior of the MEMS device. For this, the accelerometer model is instantiated in a simple test-bench and a small signal Monte Carlo analysis was initialized. Figure 3 shows the small signal Monte Carlo analysis with 50 random runs. With this simulation, it is possible to detect if the processed accelerators perform in an acceptable range or not. For each of the random runs it is possible to extract the applied device parameters for further design optimizations. This helps significantly to value the fabrication process for an expected yield or if a design optimization becomes necessary.

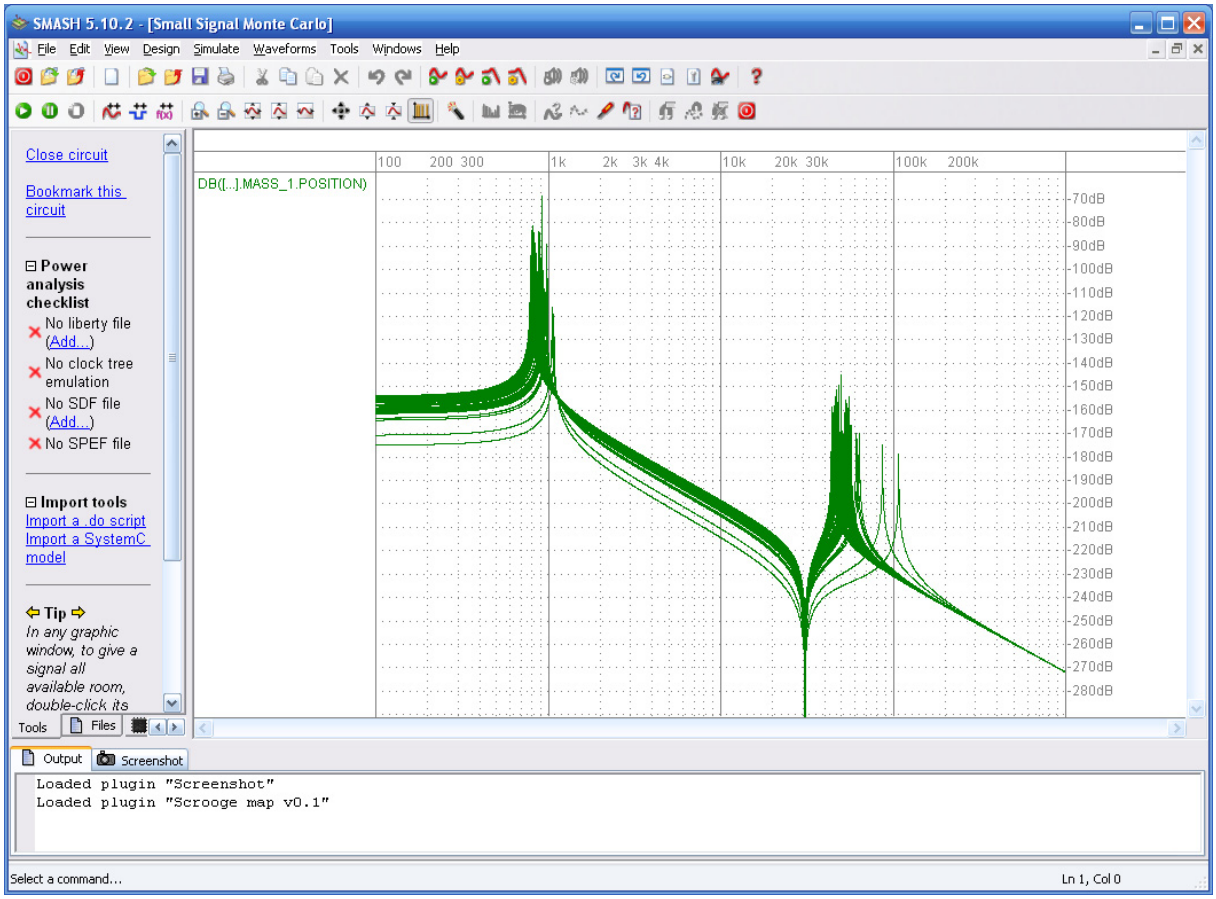


Figure 3: Monte Carlo analysis

4 Conclusion

Applying distribution functions on the parameters of simulation models helps to determine where design or process optimizations become necessary to get the system behavior in an acceptable range. Simulation runs of the optimized design with parameters that match the typical process variations of the production unit allow predicting the yield of the final MEMS devices.

The bundle comprised of the EMBLEM-Mecha basic effects library, the SLED schematic link editor and the SMASH single-kernel mixed-signal multi-language simulator enables the graphic modeling and simulation of such multi-domain systems.

Literature

[1] Voßkämper, L.M., Schmid, R., and Pelz G.: *Combining Models of Physical Effects for Describing Complex Electromechanical Devices*, IEEE/ACM International Workshop on Behavioral Modeling and Simulation, BMAS Orlando, Florida, USA, 2000

[2] Voßkämper, L.M., Schmid, R., and Pelz G.: *Modeling Micro-Mechanical Structures for System Simulations*, Anne Mignotte, Eugenio Villar and Leslie Spruiell (Eds.) System on Chip Design Languages" (best of FDL'01 and HDLCon'01) CHDL Series Kluwer, 2002

[3] Voßkämper, L.M, Domingues, Chr., Engels, L., Sibeud, L.: *Calibration in a MEMS based measurement system or how to make a sensor-ASIC loop of high resolution work*, Smart system Integration 2008, Barcelona, Spain, April 9-10, 2008, T. Gessner (Ed.) VDE Verlag, Page 546-549

[4] Pelz, G., Decker, Chr., Metzner, D., Dammers, D., Voßkämper, L.M.: *Simulating Microsystems in the Context of an Automotive Drive Application*, 10th International Forum on Advanced Microsystems for Automotive Applications (AMAA), April 25-27, 2006, Berlin, Germany

[5] Pelz, G., *Mechatronic Systems: Modeling and Simulation with HDLs*, John Wiley & Sons, 2003.

[6] Vudathu, S. P., Duganapalli, K. K., Laur, R., Kubalińska, D., Bunse Gerstner, A.: *PARAMETRIC YIELD ANALYSIS OF MEMS VIA STATISTICAL METHODS,* DTIP of MEMS and MOEMS, Stresa, Italy, 26-28 April 2006

[7] Schenato, L., Wu, W. C., El Ghaoui, L., Pister, K.*: Process Variation Analysis for MEMS design*, SPIE Symposium on Smart Materials and MEMS, Melbourne, Australia, December 2000

[8] Product flyer of Bosch SMB05x/06x accelerometer

Automatischer Back-to-Back-Test von Modellen und Code

Präsentation zum ASIM Workshop 2008

von

Dr. Joachim Wegener und Dr. Harmen Sthamer
Berner&Mattner Systemtechnik GmbH, Berlin

Überblick

- Motivation
- Evolutionärer Strukturtest
- Automatisierung von Back-to-Back-Tests
- Erste Ergebnisse
- Zusammenfassung, Ausblick

Motivation

- Standards wie ISO 26262 stufen Back-To-Back-Tests zwischen Modell und Code
  - als „highly recommended“ ein,

  sofern kein zertifizierter Code-Generator für die Generierung der Software eingesetzt wird.
- Bei einer separaten Entwicklung von Spezifikationsmodell und Implementierung sind Back-to-Back-Tests ebenfalls sinnvoll.
- Obwohl Back-to-Back-Tests in der Literatur häufig zu den Testmethoden gezählt werden, lassen sie die Frage der Testfallermittlung offen.

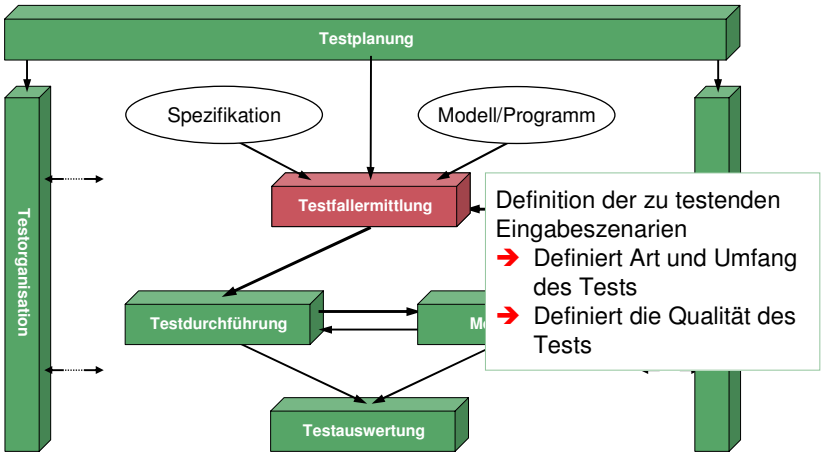
Motivation

- Testfallermittlung erfolgt üblicherweise manuell
  - aufwendig
  - unsystematisch
  - Qualität stark abhängig von der Leistung des einzelnen Testers

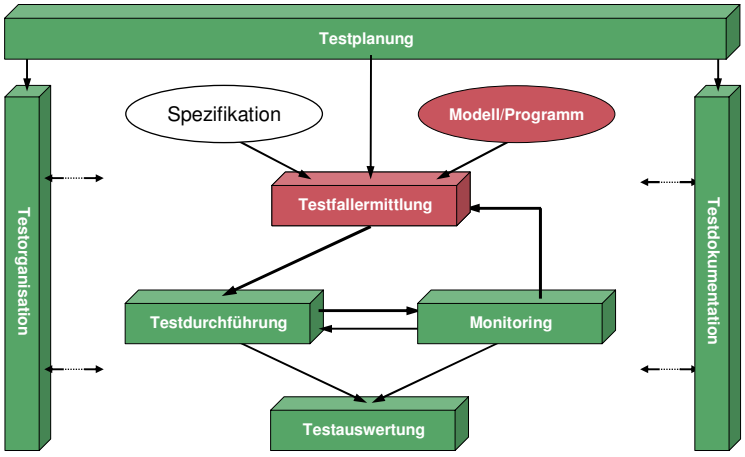
  ➢ Automatische Generierung von Testfällen für den Back-To-Back-Test aus dem generierten Code

  ➢ Ziel ist es, eine möglichst hohe Überdeckung von Modell und Code während des Tests zu erreichen

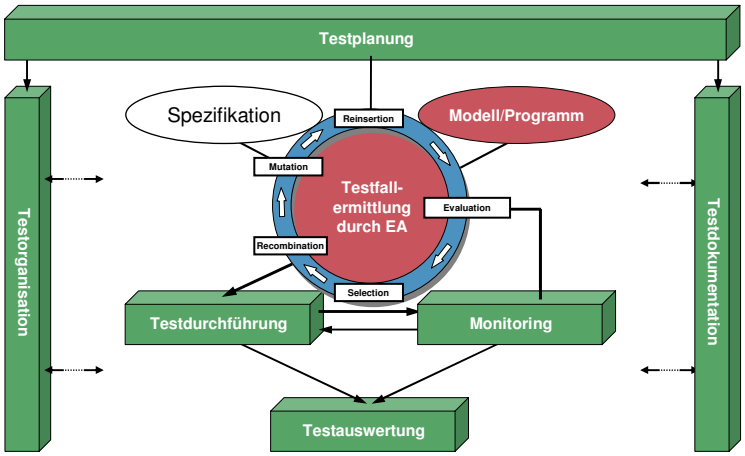
Evolutionärer Strukturtest - Grundlagen



Evolutionärer Strukturtest - Grundlagen



Evolutionärer Strukturtest - Grundlagen



berner & mattner
optimizing your development

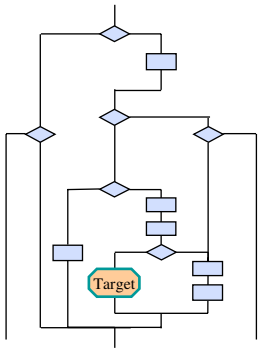
Evolutionärer Strukturtest - Grundlagen

Für evolutionäre Tests

- muss das Testziel in ein Optimierungsproblem überführt werden (Definition einer geeigneten Zielfunktion),
- bildet der Eingabedatenraum des zu testenden Systems den Suchraum, in dem nach passenden Testdaten gesucht wird,
- kommen meta-heuristische Suchverfahren zum Einsatz,
- erfolgt die Zielfunktionsbewertung anhand der Monitoring-Ergebnisse,
- werden interessante Testdaten in einem iterativen Prozess miteinander kombiniert, um bessere Testdaten zu erhalten.

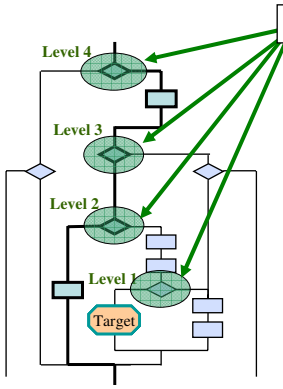
Evolutionärer Strukturtest

- Testziel: Generiere eine Menge von Testdaten, die eine möglichst hohe Strukturüberdeckung erreichen



Evolutionärer Strukturtest

- Testziel: Generiere eine Menge von Testdaten, die eine möglichst hohe Strukturüberdeckung erreichen



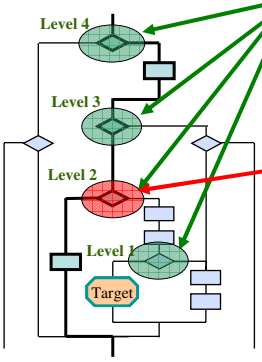
1. Approximation level

- Identifikation der Verzweigungsanweisungen, die zu einem Verfehlen des Zielknotens führen können
- Je größer der Abstand vom Zielknoten ist, desto schlechter wird der Zielfunktionswert

berner & mattner
optimizing your development

Evolutionärer Strukturtest

- Testziel: Generiere eine Menge von Testdaten, die eine möglichst hohe Strukturüberdeckung erreichen



1. Approximation level

- Identifikation der Verzweigungsanweisungen, die zu einem Verfehlen des Zielknotens führen können
- Je größer der Abstand vom Zielknoten ist, desto schlechter wird der Zielfunktionswert

2. Entfernungsmessung in den unerwünscht verzweigenden Bedingungen (Local_Distance)

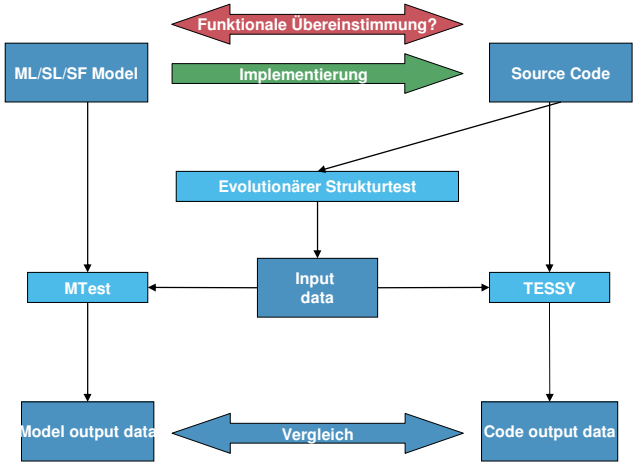
- Die Prädikate der Verzweigungsbedingungen werden für die Entfernungsmessung herangezogen, z.B.
  if A = B ⟹ Entfernung = | A - B |

Zielfunktionswert (Testdatum) =
Approximation_Level(Testdatum) +
Local_Distance(Testdatum)

Seite 11

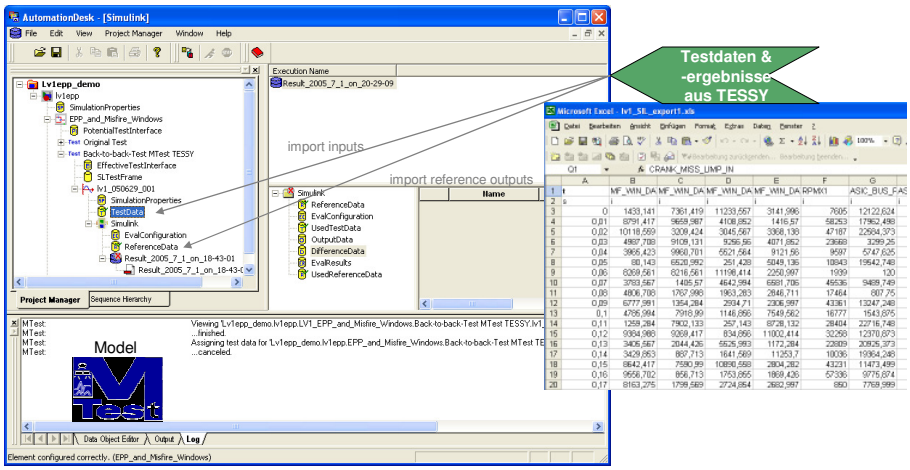
berner & mattner
optimizing your development

Back-to-Back-Test - Setup

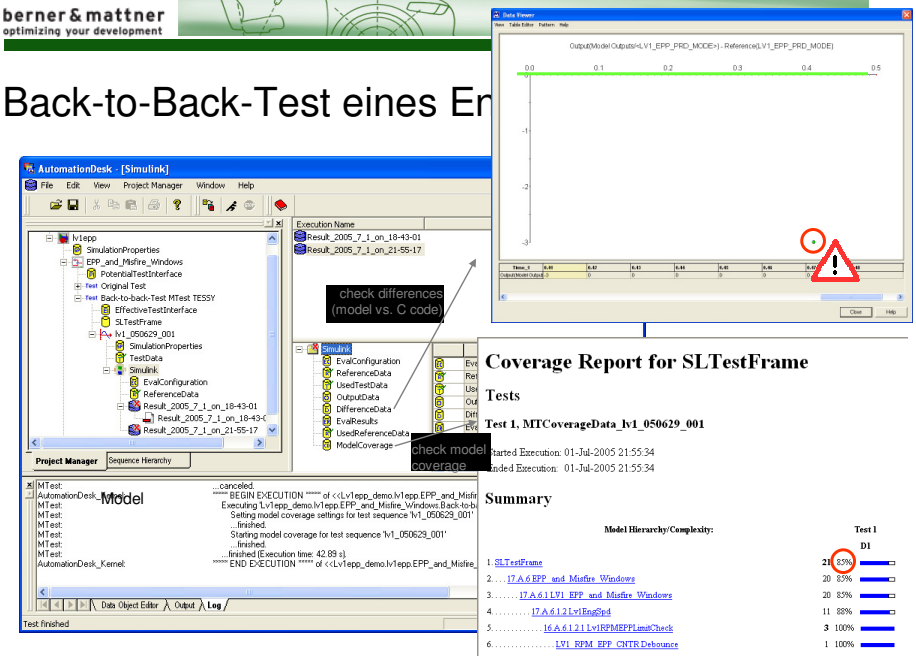


24.04.2008, Automatischer Back-To-Back-Test, Seite 12

Back-to-Back-Test eines Engine Controller



Back-to-Back-Test eines En



Erste Ergebnisse

- Modellüberdeckung* Überdeckungsgrad
  - Für das Modell entworfener manueller Test 61%
  - Mit evolutionärem Test erzeugter Test 85%

- Vergleich des Istverhaltens:
  - 8 Signale stimmen exakt überein
  - 1 signal: minor deviation

➢ Vollautomatisch höhere Modellüberdeckung auf Basis von evolutionärem Strukturtest erreicht

➢ Anomalie zwischen Modell und Code aufgedeckt

Zusammenfassung und Ausblick

- Mit der zunehmenden Verbreitung der modellbasierten Entwicklung steigt der Bedarf nach Back-To-Back-Tests
- Mit Hilfe von evolutionären Tests können Testfälle für den Back-To-Back-Test automatisch generiert werden
- In ersten Experimenten konnten mit entsprechenden Tests Abweichungen zwischen Modell und Implementierung aufgedeckt werden

- Zukünftig können Testfälle auch auf Basis des Modells generiert werden
- Gezielte Suche nach Unterschieden zwischen Modell und Implementierung

Modellierung des CAI-Brennverfahrens und die Anforderungen an die Motorsteuerung eines Versuchsfahrzeuges

Jens Ladisch, IAV GmbH, Gifhorn
Dr.Jens.Ladisch@iav.de
Wolfram Gottschalk, IAV GmbH, Gifhorn
Dr.Wolfram.Gottschalk@iav.de
Dr. Olaf Magnor, IAV GmbH, Gifhorn
Olaf.Magnor@iav.de
Matthias Schultalbers, IAV GmbH, Gifhorn
Matthias.Schultalbers@iav.de

Zusammenfassung

Das CAI-Brennverfahren verspricht einen erheblichen Beitrag zur Verringerung der CO_2-Emission von Kraftfahrzeugen. Dessen Umsetzung in Serienfahrzeugen scheiterte bislang an dem schwierig zu steuernden Verbrennungsprozess, der zwischen Motorstillstand einerseits und Motorschaden andererseits nur ein relativ kleines einzuregelndes Fenster zulässt. Motorsteuergerätefähige Modelle verhalfen zum Durchbruch, so dass mittlerweile zwei Prototypenfahrzeuge mit dem CAI-Brennverfahren im realen Straßenbetrieb bewegt werden können. Das dafür notwendige Motormanagement mit seinen speziellen Anforderungen an die Steuerung und Regelung wurde in der IAV GmbH entwickelt und auf den beiden Versuchsträgern angewendet. Insbesondere standen dabei die exakte Regelung und Steuerung des Brennverfahrens sowie die Beherrschung der Betriebsarten-Umschaltung im Fokus.

1 Einleitung

Getrieben durch die CO_2-Diskussion wird neben der Optimierung bestehender Brennverfahren, Antriebs- und Gesamtfahrzeugkonzepte die homogene Kompressionszündung diskutiert, auch CAI –Controlled Auto Ignition – genannt:

Im ottomotorischen Prozess erfolgt die Gemischaufbereitung i.d.R. bereits während der Ansaugphase, also sehr früh, so dass es zu einer guten Homogenisierung des Luft-Kraftstoff-

Gemisches kommt. Die Entflammung des Gemischs wird durch die Zündkerze initiiert, von der aus eine relativ langsam fortschreitende Flammenfront den Rest des Gemischs erreicht. Während die gute Homogenisierung Partikel- bzw. Rußbildung weitestgehend verhindert, führt die nur langsam stattfindende, einphasige Verbrennung dazu, dass ein großer Teil der Wärme an die Brennraumwände abgegeben wird und für den Antrieb des Fahrzeuges nicht mehr zur Verfügung steht. In der Flammenfront selbst führen hohe Spitzentemperaturen zu einer nennenswerten NO_X-Bildung.

Im Gegensatz dazu komprimiert der Dieselmotor zunächst die Luft und spritzt sehr spät Kraftstoff mit hohem Druck in den Brennraum ein. Der selbstentzündungsfreudige Treibstoff durchläuft dabei eine zweiphasige Verbrennung; die Gemischbildung und die Energieumsetzung gehen direkt ineinander über. Hohe Verdichtungsverhältnisse und ein nennenswerter Luftüberschuss führen zu hohen thermischen Wirkungsgraden. Die nur kurze zur Homogenisierung verfügbare Zeit führt jedoch zu lokal fetten als auch mageren Zonen und damit zur Produktion von Rußpartikeln und Stickoxiden.

CAI-Motoren vereinen die Vorteile beider Brennverfahren: Die Gemischbildung findet wie beim Ottomotor sehr früh statt, wodurch übermäßige Partikelbildung unterbunden wird. Im Gegensatz zum Dieselmotor findet nun aber die Kompression von Luft und Kraftstoff statt, bis die Aktivierungsphase durchlaufen ist und sich das Gemisch selbsttätig entzündet. Im Idealfall erfolgt die Entzündung zeitgleich im ganzen Brennraumvolumen und lässt nur sehr wenig Zeit für einen Wärmetransport zu den Brennraumwänden zu. Insgesamt niedriges Temperaturniveau der Verbrennung und Entflammbarkeit auch bei Luftüberschuss ergeben eine äußerst geringe NO_X-Emission bei niedrigem Kraftstoffverbrauch.

So vielversprechend der CAI-Motor auch ist, verhinderte bis vor kurzem dessen schwierige Regelung den Einsatz im Fahrzeug: Im Gegensatz zum Otto- und Dieselmotor gibt es beim CAI keinerlei direkten Zugriff auf die zeitliche Gestaltung der Energieumsetzung. Ein zu hoher Energieinhalt im Brennraum würde zu übermäßig hohen Zylinderdrücken und Zylinderdruckgradienten führen – im ungünstigsten Fall bis hin zum mechanischen Versagen brennraumbegrenzender Teile. Bei zu geringem Energieinhalt kann die Entzündung gänzlich ausbleiben. Das sinnvollerweise einzuhaltende Toleranzband ist mit nur etwa ±5° Kurbelwinkel sehr klein. Die Situation wird durch realen Fahrbetrieb und variierende Umweltbedingungen zudem zum Teil deutlich erschwert.

Als zielführend hat sich die Modellierung der CAI-Verbrennung erwiesen: Durch die Abbildung des Verbrennungsvorganges im Motorsteuergerät ist der Zylinderdruckverlauf und damit der aus thermodynamischer Sicht noch bedeutsamere 50%-Energieumsatzpunkt berechenbar. Auch das für die Rückkopplung wichtige Signal der sich tatsächlich einstellenden Verbrennungshalbwertzeitlage wird modellbasiert aus der Kurbelwellenbewegung des Motors gewonnen, so dass auf teure Zylinderdrucksensoren verzichtet werden kann.

In den nachfolgenden Kapiteln werden die Besonderheiten der CAI-Verbrennung aus Sicht der Modellierung und der Umsetzung in die Motorsteuerung beschrieben. Die Modelle

sind maßgeblich daran beteiligt gewesen, den weltweit ersten, im realen Fahrbetrieb verwendbaren PKW mit CAI-Antrieb darzustellen.

2 Modellbasierte Vorsteuerung und Regelung des Brennverfahrens

2.1 Charakterisierung des Prozesses

Die Zielstellung des CAI-Projektes war die Durchbildung und Darstellung eines Versuchsfahrzeuges, dessen Antriebsaggregat als selbstzündender Benzinmotor arbeitet. Dieses Versuchsfahrzeug wurde auf der Basis eines aktuell handelsüblichen PKW mit einem 1.6l-SI-Motor aufgebaut. Dabei wurde der Serienstand des 1.6-Liter-Aggregates weitestgehend beibehalten. Eine Ausnahme bildet jedoch der Ventiltrieb im Zylinderkopf. Dieser wurde zu einem zweistufigen mechanischen Ventilumschaltsystem konstruktiv umgestaltet. Bedingt durch den mechanisch einfachen und kostengünstigen Aufbau dieses Ventilumschaltsystems konnten im Hinblick auf eine eventuelle Serieneinführung die hardwareseitigen Mehraufwendungen relativ gering gehalten werden. Der IAV-seitige Forschungsgegenstand dieses Projektes bestand in der Bereitstellung einer entsprechend erweiterten und den erhöhten Anforderungen angepassten Steuergeräte-Hardware sowie der Entwicklung der Steuer- und Regelungsfunktionen, die für das CAI-Brennverfahren einschließlich des Ventilumschaltsystems erforderlich sind. Weiterhin waren entsprechende Applikationen am Versuchsträger vorzunehmen, um diesen hinsichtlich der Fahrbarkeit sowie dem Bestehen von standardisierten Emissionstests zu optimieren.

Wie bereits in der Einleitung erwähnt, bietet das HCCI-Brennverfahren prinzipbedingt keine Stellgröße, die einen direkten Zugriff auf die Entzündung des Luft-Treibstoff-AGR-Gemisches zulässt. Entzündungsbestimmend ist der Kurbelwinkel, zu dem der erforderliche Aktivierungszustand erreicht wird. Dieser wiederum wird im wesentlichen durch das Luft-Kraftstoff-Verhältnis (im Folgenden mit Lambda bezeichnet), den Druck, die Temperatur und den chemischen Zustand der Zylinderladung bestimmt. Um den Serienaspekt nicht aus den Augen zu verlieren, wurde das Fahrzeug nicht mit teuren Sensoren bestückt, die eine Rückinformation über den tatsächlich erreichten Lambda- und Temperaturzustand geben könnten. Ohnehin wäre diese Information wertlos, da bei Vorliegen eines Sensorsignals über keinerlei Eingriffsmöglichkeit in den anschließend ablaufenden Prozess mehr bestünde.

Die Lösung des Problems konnte dadurch erreicht werden, dass das Brennverfahren zunächst einmal in hohem Maße eigenstabil gestaltet werden konnte: Bei Fixierung der brennverfahrensrelevanten Parameter ist auch über einen längeren Zeitraum – mehrere Minuten – keine nennenswerte Betriebspunktdrift erkennbar. Dies wiederum ist die Voraussetzung gewesen, um ein Brennverfahrensmodell aufzustellen, welches eine in allen Betriebspunkten ausreichend genaue Vorsteuerung der Aktuatoren erlaubt. Die begrenzten Ressourcen im

Motorsteuergerät verlangen Abstriche an die Qualität des Modells. Als brauchbare Lösung wurde daher eine Kombination aus Kennfeldern und physikalisch basierten Rechenalgorithmen gefunden. Damit wird sowohl der reale, oft hochdynamische Fahrbetrieb als auch die Umschaltung von konventioneller fremdgezündeter Betriebsart in die CAI-Betriebsart realisiert.

Kriechende Effekte können jedoch nicht allein durch die Vorsteuerung kompensiert werden, weil sie entweder schwer erfassbar sind – verschiedene Kraftstoffsorten, Ablagerungen im Brennraum usw. – oder bislang noch nicht in ihrer Auswirkung hinreichend genau untersucht werden konnten – wie etwa Serienstreuung, Alterung und Verschleiß. Auch an dieser Stelle helfen Modelle im Motorsteuergerät: Die Drehbewegung der Kurbelwelle setzt sich durch die diskontinuierliche Verbrennung des Hubkolbentriebes aus einem Gleich- und einem Wechselanteil zusammen. Aus dem Wechselanteil und der Kenntnis kinematischer Parameter des Kurbeltriebes kann auf einen verursachenden Zylinderdruck zurückgeschlossen werden. Vom Zylinderdruck wiederum ist die Bestimmung der so wichtigen Verbrennungshalbwertzeitlage möglich.

Das Modell zur Berechnung des Zylinderdruckes aus der Drehunförmigkeit der Kurbelwellenbewegung basiert auf rein kinematischen und geometrischen Gesetzen und kann mit wenigen mathematischen Gleichungen beschrieben werden kann. Die Bestimmung der Verbrennungshalbwertzeitlage erfolgt mittels eines kennfeldbasierten Verbrennungsmodells. Das Ergebnis sei beispielhaft in der Abbildung 1 gezeigt: In der oberen Hälfte ist das Ergebnis des Modellausgangs zu sehen, in der unteren das Ergebnis einer Heizverlaufsanalyse auf Basis des Zylinderdrucksignals. Zur besseren Vergleichbarkeit der beiden Signale ist jeweils der gleitende Mittelwert des Rohsignals über fünf Arbeitsspiele dargestellt. Der rein visuelle Eindruck wird durch den rechnerischen Vergleich bestätigt: Im Allgemeinen liegt die Abweichung des Modellwerts vom Messwert unter 1° Kurbelwinkel.

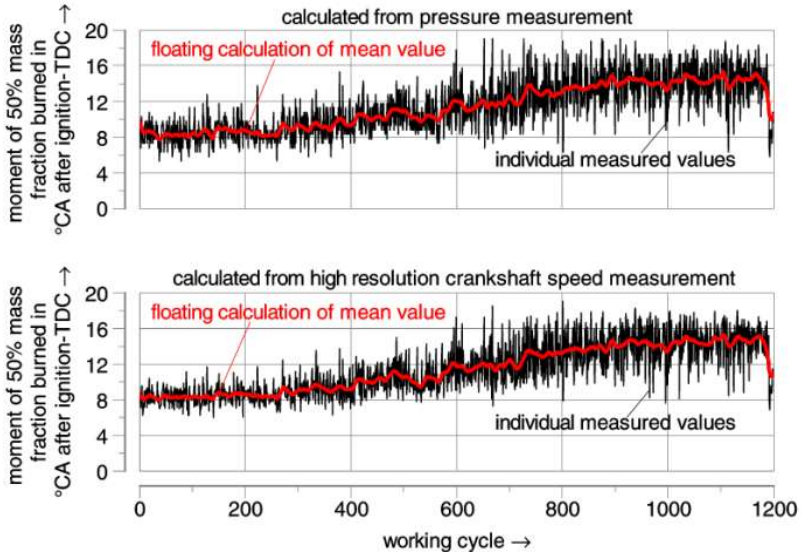


Abbildung 1: Vergleich der modellierten Verbrennungsschwerpunktlage (oben) mit der gemessenen (unten), jeweils als Rohwert und gleitender Mittelwert

2.2 Die technischen Arbeitspakete

Für die zu entwickelnden bzw. zu applizierenden Regel- und Steuerfunktionen sind eine Reihe von Arbeitspaketen aufgestellt worden. Dazu zählen die CAI-spezifischen Funktionen wie die Füllungserfassung, die Füllungssteuerung über den Drosselklappensollwert, der Treibstoffpfad, die Lambdaregelung, die Nockenwellensollwertsteuerung, die Zündungssteuerung sowie die Umschaltfunktionen vom konventionellen Verbrennungsmodus in den CAI-Modus und umgekehrt. Darüber hinaus sollte auch bei Lastwechseln und Drehzahländerungen in begrenztem Umfang ein Mindestniveau an Fahrkomfort gewährleistet sein.

Für den CAI-Betrieb benötigt man im Vergleich zum Serienstand des Motors veränderte Nockenwellenprofile. Um jedoch den Motor mit beiden Brennverfahren betreiben zu können, sind demnach zweistufig variable Nockenkonturen auf der Einlass- und auf der Ausslassseite vorzusehen. Diese Forderung wurde mittels des bereits erwähnten zweistufigen, mechanischen Ventilumschaltsystems realisiert.

Die Umschaltung erfolgt durch 16 elektromagnetische Aktuatoren, davon 8 auf der Einlassseite sowie 8 auf der Auslassseite. Je Zylinder werden 4 Aktuatoren verwendet, davon 2 für die Einlassventile und 2 für die Auslassventile. Durch die jeweils beiden Aktuatoren wird der erste für die Umschaltung von dem konventionellen Verbrennungsmodus in den HCCI-Modus und der zweite für die Rückschaltung vom HCCI-Modus in den konventionellen Verbrennungsmodus benötigt. Die Aktuatoren sind über einen zusätzlichen Kabelbaum an ein spezielles IAV-Steuergerät ($FI^2RE^®$) angeschlossen.

2.3 Die Motorsteuerung

Das besondere Charakteristikum der CAI-Verbrennung ist die kontrollierte Selbstzündung eines relativ warmen und vergleichsweise hochgradig restgasverdünnten Gemisches. Die Gemischtemperatur und die Gemischkomposition bestimmen dabei in besonderem Maße die Zündwilligkeit, wobei der erforderliche Restgasanteil mit steigender Motorlast abnimmt, weil die Abgastemperatur mit der Motorlast kontinuierlich steigt. Das bedeutet u.a., dass die optimale Lage des 50%-Energieumsatzpunktes nicht mehr vollständig durch den Zündfunken gesteuert werden kann, sondern vielmehr durch die Ladungskomposition des Gemisches mit seinen sich entsprechend einstellenden Eigenschaften. Ein weiteres charakteristisches Merkmal ist die nahezu vollständig vorliegende Gleichverteilung der Verbrennungsnester und die fast gleichzeitig einsetzenden Zündvorgänge im Brennraum, wodurch das Gemisch im Vergleich zum konventionellen Benzinmotor deutlich schneller bei gleichzeitig niedrigeren Temperaturen umgesetzt wird.

Eine der besonderen Herausforderungen an die Regelung des CAI-Brennverfahrens besteht darüber hinaus darin, den hohen Genauigkeitsanforderungen auch unter den

hochdynamischen Randbedingungen des Alltagsbetriebes zu entsprechen, zumal der Stellbereich zu kleinen Lasten hin durch das Auftreten von Verbrennungsaussetzern begrenzt ist. Zu großen Lasten hingegen nehmen die Druckgradienten sehr schnell hohe Werte an, was sich in deutlich vernehmbaren Verbrennungsgeräuschen bemerkbar macht und zudem zu Bauteilschäden am Brennraum sowie am Kurbeltrieb führen kann.

Deswegen wird vielfach zur geeigneten Regelung einer stabilen Verbrennung die sensorische Erfassung des Zylinderdruckes favorisiert. Allerdings spricht dagegen, dass bedingt durch die sehr steilen Wirkungsgradkennlinien der homogenen Kompressionszündung eine Vorsteuerung der brennverfahrensrelevanten Aktuatoren bereits derart grob erfolgen würde, dass in einem Abstand von nur zwei aufeinanderfolgenden Arbeitszyklen bereits das Brennverfahrensfenster verlassen wäre, und demnach auch die Regelung zur Optimierung des Motorbetriebes nicht mehr adäquat eingreifen kann. Vielmehr besteht die Aufgabe der Motorregelung darin, eher niederfrequente Variationen wie Ansauglufttemperatur, Umgebungsdruck, Alterung, Ablagerungen oder selbst Exemplarstreuungen zu kompensieren. Innerhalb des Brennverfahrensfensters, welches in der Vorsteuerung abgebildet ist, wird der Betriebspunkt zugunsten von Verbrauch, Emissionen und Akustik optimiert und eingestellt. [1]

Die vorgesteuerten motorischen Betriebsgrößen bilden sich im Zylinderdruckverlauf ab, der selbstverständlich über einen entsprechenden Sensor gemessen werden könnte. Allerdings kann diese Information auch aus der zyklisch variierenden Winkelbeschleunigung der Kurbelwelle extrahiert werden. Die sich in einem Betriebspunkt nur langsam ändernden Druckverlaufskurven erlauben zusammen mit einer geeigneten Signalverarbeitung der Kurbelwellensignale die fortwährende Selbsteinstellung des Verfahrens. [2]

Die variable Ventilsteuerung ist, wie erwähnt, in Gestalt des zweistufigen Ventilumschalt-systems vorgesehen worden, welches mittels des speziellen FI²RE®-Steuergerätes über die 16 Aktuatoren angesteuert wird. Die Anforderung der Schaltung erfolgt über den CAN-Bus vom Motorsteuergerät. Anhand der Signale des Kurbelwellendrehzahlsignals und des Nockenwellendrehzahlsignals errechnet das FI²RE®-Steuergerät mit seiner internen Winkeluhr die drehwinkelsynchrone und zündfolgekonforme Ansteuerreihenfolge der Aktuatoren

Da jedoch das Zeitfenster zwischen der Anforderung der Umschaltung von der Motorsteuergerätefunktion bis zur tatsächlichen Umschaltung durch FI²RE® nicht determiniert, andererseits diese Information für das Brennverfahren aber wichtig ist, wird eine schnelle Rückleitung mit definierter Übertragungszeit benötigt. Hierfür wurde eine analoge Datenleitung vorgesehen, die vom Seriensteuergerät mit einer Abtastrate von 1 ms eingelesen wird. Vom FI²RE®-Steuergerät soll diese Leitung in der Art bereitgestellt werden, dass die aktuell anliegen Ausgangssignale des FI²RE® an die 16 Aktuatoren gesamthaft in ein analoges Spannungssignal umgesetzt werden. Anhand dieser Information kann das Motorsteuergerät

dann zylinderindividuell die Parameter für den Wechsel von dem einen in den anderen Betriebsmodus verstellen.

Da im CAI-Modus deutlich höhere Restgasraten gefahren werden und die optimalen Steuerzeiten stark betriebspunktabhängig sind, ist darüber hinaus in der Motorsteuerung eine modifizierte Sollvorgabe für die Nockenwellenphasensteller vorzusehen.

Wegen der starken Verdünnung durch den hohen Restgasanteil verbleibt die Temperatur während der Verbrennung auf einem vergleichsweise niedrigen Niveau und läuft bei niedriger bis mittlerer Last unterhalb der Schwelle nennenswerter NO_X-Bildung ab. Aus den Prüfstandsversuchen war darüber hinaus zu entnehmen, dass der CAI-Betrieb bei stöchiometrischen sowie bei überstöchiometrischen Betrieb möglich ist. Aus diesem Grund wurde die serienmäßige Lambdaregelung hinsichtlich eines dynamischen Führungsverhaltens modifiziert und die Reglerparameter neu bemessen. Mit Hilfe des relativ dynamischen Verhaltens des Lambdaregelkreises, welcher über den Treibstoffpfad die Gemischkomposition beeinflusst, konnte ein angemessener Dynamikaufbau für das Fahrverhalten realisiert werden.

Der Lambdaregler ist als PI-Regler ausgelegt worden, dessen Reglerparameter im Hinblick auf den totzeitbehafteten Prozess der Ansaugstrecke in Kennfeldern, welche über die relative Luftfüllung und die Aggregatedrehzahl adressiert werden. Da dieser Lambdaregler nur für die Fahrten im CAI-Modus aktiv geschaltet ist, wurde er mit einer Hold-Funktion und einer Wind-up-Funktionalität für den I-Anteil erweitert, um ein unkontrolliertes Aufladen zu vermeiden. Das Wiederaufnehmen des Betriebes des CAI-Lambdareglers wurde über eine Rampenfunktion vorgenommen, um weiche und ruckfreie Umschaltungen erzielen zu können.

Im Unterschied zu einem fremdgezündeten Motor, bei dem die schnelle Momentenvariation über den Zündwinkelpfad möglich ist, existiert eine solche Stellmöglichkeit bei einem CAI-Aggregat nicht. Um dennoch eine schnelle Momentenvariation realisieren zu können, wurde eine sogenannte sollmomentenabhängige Lambdavariation entwickelt, mit der bei entsprechend höheren Lambdawerten das abgegebene Moment reduziert und bei niedrigeren Lambdawerten das Moment erhöht werden kann.

Ein weiteres Merkmal der CAI-Verbrennung ist der deutlich entdrosseltere Betrieb wodurch zusätzlich die Verluste im Motor reduziert werden können. Daher mussten auch die Füllungssteuerung und die Füllungserfassung im Vergleich zur Serie neu entwickelt beziehungsweise modifiziert werden. Die Füllungssteuerung weist dabei eine wesentlich höhere Dynamik als die Serienausführung auf, um insbesondere einen schnellen Füllungsaufbau für die Umschaltung vom konventionellen Verbrennungsmodus in den CAI-Modus zu gewährleisten. Für eine momentenneutrale Umschaltung ist es daher erforderlich, dass bei einem dynamischen Füllungsaufbau zugleich über die Momentensteuerung des Motorsteuergerätes der Zündwinkel zur Momentenreduktion in Richtung Spätwinkellagen zurückgefahren wird. Erst nach der erfolgten Umschaltung wird dann die Zündwinkelvorgabe von der CAI-Zündwinkelsteuerung vorgenommen. Der Zündzeitpunkt hat im Gegensatz zum

konventionellen Verbrennungsprozess im CAI-Betrieb nicht die direkte Funktion der Entflammung über ein Zündplasma, zumal eine Entflammung eines homogenen Gemisches bei einer AGR von rund 50 % auch gar nicht möglich wäre. Vielmehr wird durch den Energieeintrag des Zündfunkens zum Zündfunkenzeitpunkt das Brennverfahren robuster und sicherer gestaltet. Denn in dem durch den Zündfunken gebildeten Zündplasma werden freie Radikale produziert, welche die Entzündungsstabilität besonders im Bereich kleiner Lasten verbessern. In Abhängigkeit des Zündwinkels wird die Radikalenbildung beeinflusst, wodurch sich die Verbrennung in einem allerdings nur relativ geringen Umfang steuern lässt. [3]

Die Zündwinkelsteuerung hat in Anlehnung an die Vorgabe der Lambda-Sollwerte ebenfalls eine lastabhängige Interpolation, um eine dynamischere Momentenvariation zu erzielen, wobei jedoch bedingt durch die CAI-typisch sehr geringe Stellwirkung des Zündwinkels auch der Einfluss auf den Momentenaufbau eher klein ist.

Eine weitere wichtige Steuerungsaufgabe besteht in der eigentlichen Freigabe der Umschaltung in den CAI-Modus. Dazu werden zunächst in Form von Kennlinien die Grenzen des Betriebskennfeldes hinsichtlich der Last und der Drehzahl vorgegeben und mit dem aktuell vorliegenden Betriebszustand verglichen. Weitere Kriterien für eine Freigabe sind die Kühlwassertemperatur, die Batteriespannung im Bordnetz, der eingelegte Gang sowie die vollständig eingerückte Kupplung.

Um an den Grenzen des Betriebskennfeldes ein ruhiges und schwingungsfreies Umschalten der Betriebsmodi zu realisieren, wurde eine Schalthysterese vorgesehen. Auch bei dynamischen Momentenanforderungen mit hohen Drehzahlgradienten sowie einem nur kurzzeitigem Tangieren des Betriebskennfeldes innerhalb gewisser Grenzlasttoleranzen und Toleranzdrehzahlen erfolgt keine Umschaltung, um den Fahrkomfort auf einem angemessenen Niveau zu halten. Ein häufiges Umschalten würde sich auch wegen der dann ständig auftretenden Füllungsüberhöhung mit einhergehender Zündwinkelrücknahme negativ auf den Verbrauch auswirken. Die Hysteresebreite und die Auswertung der Dynamik der Momentenanforderung lassen sich applizieren und wurden im Hinblick auf den MVEG-Zyklus optimiert.

Bei optimaler Bedatung erfolgen mit der vorliegenden Motorsteuerung nahezu ruckfreie Umschaltvorgänge und es konnte ein insgesamt alltagstaugliches Fahrverhalten erzielt werden. In Abbildung 2 sind die entsprechenden Verläufe der relevantesten Größen für die Umschaltvorgänge dargestellt.

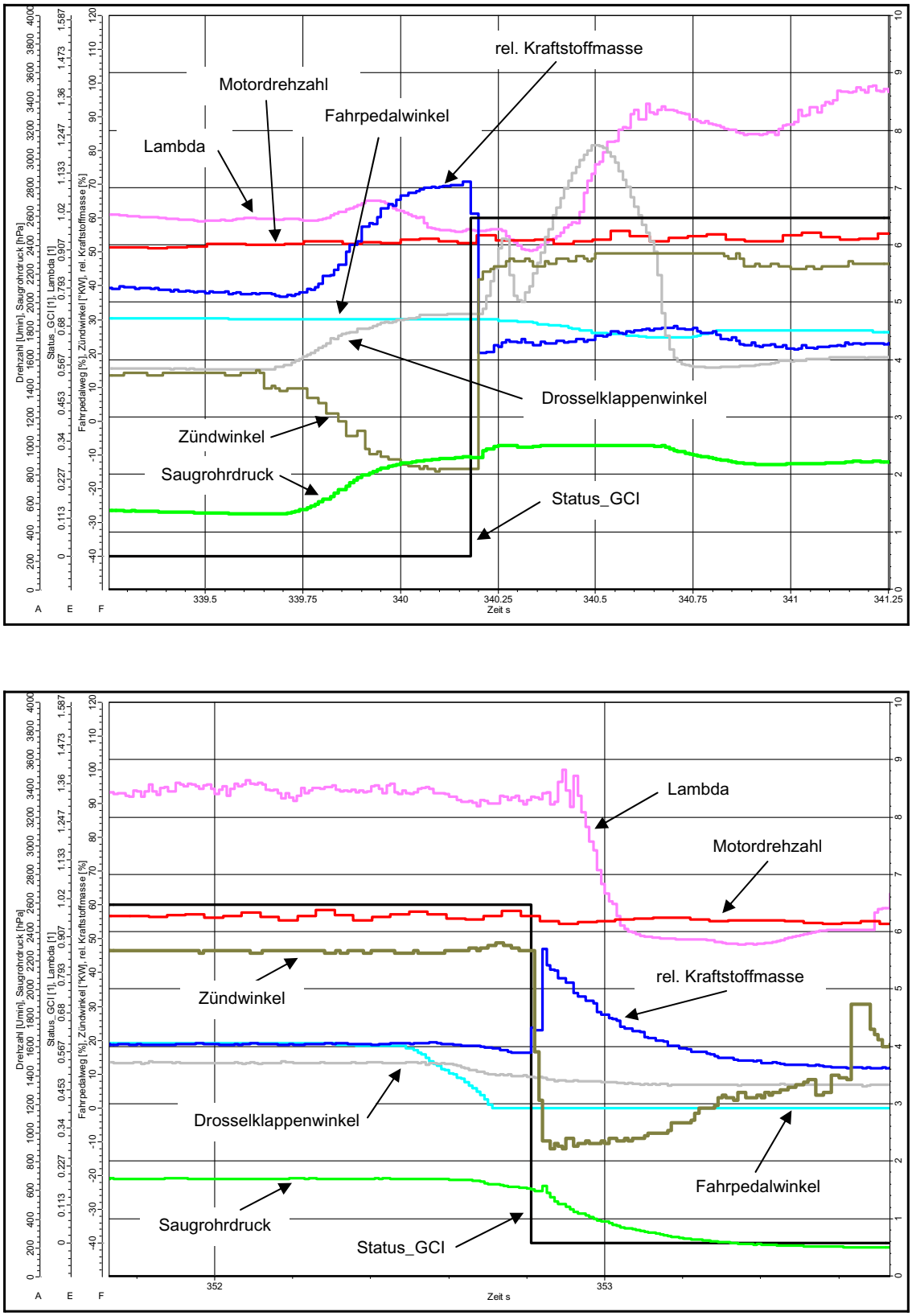


Abbildung 2: Umschaltvorgänge von SI- in den CAI-Modus und zurück

2.4 Übertrag ins Fahrzeug und Ergebnisse

Die am Prüfstand zunächst im stationären Betrieb erzielten Ergebnisse waren möglichst vollständig auf den Versuchsträger zu überführen. Da am Prüfstand jedoch die Optimaleinstellungen der Brennverfahrensparameter zunächst betriebspunktindividuell ermittelt wurden, mussten jeweils ausgehend von diesen Punkten die noch tolerierbaren Parameterstrahlen bis hin zu den Brennverfahrensgrenzen vermessen werden. Das bedeutet zum Beispiel, dass für den Betriebspunkt mit fester Drehzahl-Last-Kombination sowie fester Steuerzeit und somit fester AGR die Lambdavariation von der Magergrenze bis zur Fettgrenze durchgemessen worden ist. Im realen Fahrbetrieb stellen sich Betriebspunkttrajektorien ein, entlang derer die Brennverfahrensparameter mit unterschiedlichem Verzug nachgeführt werden könnten. Deswegen werden in einem zweiten Schritt die Optimalkennfelder innerhalb der ermittelten Brennverfahrensgrenzen so modifiziert, dass auch bei hoher Fahrdynamik noch alle relevanten Aktuatoren nachgeführt werden können. Auch hier kann das bereits genannte Beispiel des Luftverhältnisses herangezogen werden, welches sich betriebspunktabhängig in einem Bereich von λ = 1,0 ... 1.5 variieren lässt. Dieser zusätzliche Freiheitsgrad kommt der Brennverfahrensführung entgegen. In einzelnen ausgewählten Konstantfahrpunkten konnte im Vergleich zum konventionellen Serienstand eine Verbrauchsreduktion von bis zu 17 Prozent ermittelt werden, was im wesentlichen auf die weitgehende Entdrosselung sowie die schnellere Energieumsetzung zurückzuführen ist. Im MVEG-Zyklus, in dem der CAI-Modus noch vergleichsweise wenig gefahren wird, sind dennoch bis zu 4 % Treibstoffersparnis nachgewiesen worden.

Literatur

[1] Jelitto, Christian, et. al.: *Herausforderungen für einen Gasoline Compression Ignition-Demonstrator*, 16. Aachener Kolloquium Fahrzeug- und Motorentechnik, Aachen, 2007

[2] Jeinsch, Torsten et. al.: *Die Anforderungen neuer Brennverfahren an die Motorsteuerung*, 6. Symposium: Steuerungssysteme für den Antriebsstrang von Kraftfahrzeugen, Berlin, Juni 2007

[3] Jelitto, Christian et. al.: *Herausforderungen für einen Gasoline Compression Ignition-Demonstrator,* 5. VDI-Tagung Innovative Fahrzeugantriebe, Dresden, November 2006

Simulation der Auswirkungen einer gesteuerten Motormomentenvorgabe auf das Schwingungsverhalten einer Sattelzugmaschine

Carsten Joachim, Universität Stuttgart, IVK
Prof. Dr.-Ing. H.-C. Reuss, Universität Stuttgart, IVK
Jochen Horwath, Daimler AG

Zusammenfassung

Triebstrangschwingungen treten als Phänomen bei jeder Art von Kraftfahrzeugen auf. Bei schweren Nutzfahrzeugen im 40t Bereich sind sie aufgrund der hohen Massen und niedrigen Gangübersetzungen besonders ausgeprägt. Sie können sich negativ auf die Bauteilbelastung, das Komfortempfinden und den Schaltablauf auswirken. Moderne Nutzfahrzeuge sind fast ausschließlich mit automatisiertem Antriebsstrang ausgestattet, was über die Elektronik die Möglichkeit einer gezielten Beeinflussung der Schwingungen bietet. Eine geeignete Maßnahme ist eine intelligent gesteuerte Motormomentenvorgabe für den Momentenabbau bzw. -aufbau und wird im Rahmen dieses Beitrages vorgestellt. Der Einfluss auf die vom Fahrer gefühlte Beschleunigung (Fahrkomfort) kann jedoch anhand dem der Steuerung zugrundelegenden Zweimassenmodell nicht untersucht werden. Hierfür wird eine Modellbeschreibung für eine Sattelzugmaschine mit mehreren Freiheitsgraden entwickelt. Anhand dieser werden zum Einen die Auswirkungen der gesteuerten Momentenvorgabe untersucht und schließlich theoretisch erforderliche Momentenverläufe für optimierten Fahrkomfort hergeleitet.

1 Motivation

1.1 Entstehen von Triebstrangschwingungen

Bei der Momentenübertragung vom Motor zu den Rädern sind eine Vielzahl von Komponenten mit ihren Trägheiten und Steifigkeiten beteiligt. Zusammen bilden sie ein komplexes schwingungsfähiges System. Abbildung 1 zeigt ein vereinfachtes Modell mit den wesentlichen Komponenten.

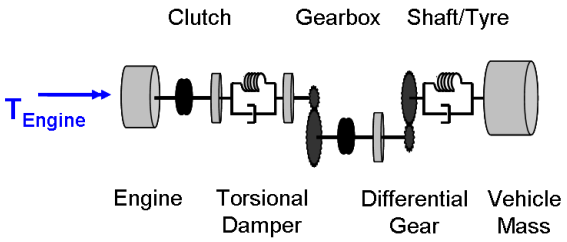


Abbildung 1: Triebstrangmodell

Das System wird u. a. vom Motormoment angeregt. Transiente Anregungen sind z. B. Lastwechsel, wie sie beim Beschleunigen oder Verzögern vorliegen. Bei automatisierten Schaltgetrieben treten solche schnellen Momentenänderungen auch vor und nach einer Schaltung auf. Die auftretenden Schwingungen werden nach ihren Eigenformen klassifiziert, wobei hier hauptsächlich die erste Eigenform, das Ruckeln, relevant ist. Dessen Frequenz beträgt bei vollbeladenen LKW in den Anfahrgängen 0.5 bis 1 Hz.

1.2 Auswirkungen

Die niederfrequenten Ruckelschwingungen können in zweierlei Hinsicht störend wirken. Zum Einen können sie den Schaltablauf beeinträchtigen. Nutzfahrzeuggetriebe sind heutzutage meist als automatisierte Schaltgetriebe mit Klauenkupplung ausgeführt. Eine elektronische Synchronisation durch Software und Aktuatorik ersetzt hierbei die mechanische Synchronisiereinrichtung. Bei einer Schaltung muss der Gang ausgelegt werden, d.h. die Klaue im Getriebe geöffnet werden. Das Motormoment muss vorher abgebaut werden, damit der Motor während der Schaltung nicht hochdreht. Jedoch liegt nicht das Motormoment, sondern das in der Wellensteifigkeit wirksame Radmoment bei vernachlässigbaren Trägheiten der Abtriebswellen am Getriebe an. Die Abhängigkeit des Radmoments vom Motormoment lässt sich über Differentialgleichungen beschreiben. Es kann folglich bei voll abgebautem Motormoment noch ein Radmoment und damit eine Restverspannung im Triebstrang vorhanden sein, vgl. Abbildung 2. Wird der Gang bei vorhandener Restverspannung ausgelegt, d.h. die Klaue geöffnet, können Oszillationen auf der Getriebewelle bis zum Extremfall Getriebeschlag auftreten. Dies erschwert die folgende elektronische Synchronisation und kann zu einer ungewünscht langen Schaltdauer und Zugkraftunterbrechung führen.

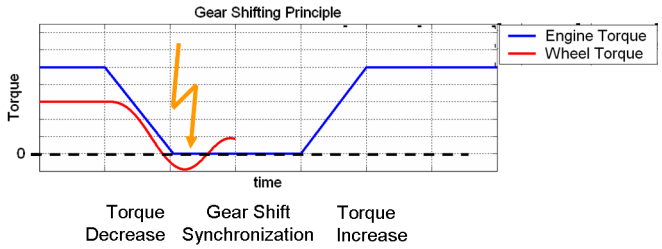


Abbildung 2: Prinzipieller Ablauf eines Schaltvorgangs

Zum Anderen wirken sich Triebstrangschwingungen auf den Fahrkomfort in Form der vom Fahrer gefühlten Beschleunigung aus. Das Radmoment beschleunigt die Räder, die wiederum das Fahrzeug antreiben. Bei starrem Fahrzeugaufbau und starrer Rad-Fahrbahn-Kopplung ist das Radmoment direkt der Fahrzeuglängsbeschleunigung proportional. Schwingungen im Radmoment übertragen sich auf den Fahrer als oszillierende Beschleunigung und werden als unangenehm empfunden (vgl. [1]), zumal die Ruckeleigenfrequenzen im für den Menschen deutlich wahrnehmbaren Bereich liegen. Bei einer Sattelzugmaschine ist die Annahme eines starren Aufbaus jedoch eine deutlich stärkere Vereinfachung im Vergleich zum PKW. Insbesondere tritt hier neben die Längsbeschleunigung noch eine Vertikalkomponente, die den gefühlten Komfort dominiert. In Kapitel 3 wird ein Modell vorgestellt, das dieses Verhalten nachbilden kann.

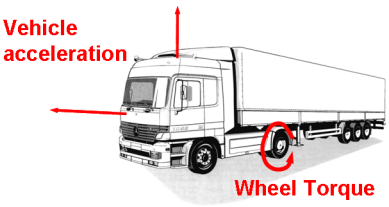


Abbildung 3: Komfortrelevante Beschleunigungskomponenten beim LKW

2 Verfahren zur Schwingungsunterdrückung

2.1 Feedforward - Strategie

Für eine schnelle Synchronisation sollte das an der Klaue anliegende Radmoment zum Zeitpunkt des Gangauslegens Null betragen. Somit stellt das Radmoment die zu regelnde Größe dar. Es bietet sich hier die Möglichkeit, durch einen Feedforward-Ansatz den Verlauf des Radmomentes direkt vorzugeben. Dies kann erreicht werden, indem ein invertiertes Streckenmodell in der Steuerung gerechnet wird, vgl. Abbildung 4.

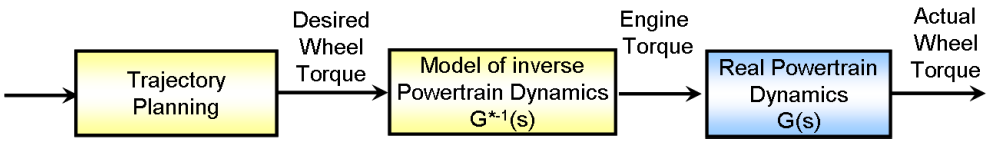


Abbildung 4: Feedforward-Strategie

Das Streckenmodell wird zwangsläufig eine Vereinfachung der realen Strecke darstellen, um für eine praktische Umsetzung handhabbar zu bleiben. Ein Zweimassenschwinger

ist das einfachste Modell, das die Ruckeleigenform abbilden kann (vgl. Abbildung 5). Die Form der Sollmomentenvorgabe ist entscheidend für den sich ergebenden Motormomentenverlauf. Eine angepasste Trajektorienplanung ist in [2] vorgestellt.

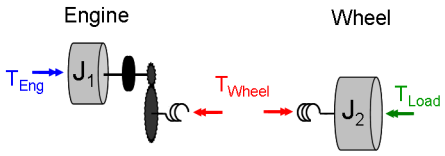


Abbildung 5: Zweimassenmodell zur Abbildung der Ruckeleigenform

2.2 Wirkungsweise beim Schaltvorgang

Zur Verifikation der Wirksamkeit dieser Strategie wurde mittels eines Rapid Control Prototyping-Systems ein Softwareprototyp erstellt und in einem nächsten Schritt in ein Serienfahrzeug des Typs Actros integriert. Abbildung 6 zeigt Ergebnisse für einen Momentenabbau innerhalb von 0.4 sec, durchgeführt in der linken Grafik mit einem rampenförmigen Verlauf und rechts mit der Feedforward Strategie. Im oberen Teil der Grafiken ist jeweils der Motormomentenverlauf, im unteren die Drehzahlen von Motor und Getriebewellen dargestellt. Nach dem Momentenabbau wird bei 0.9 s die Kupplung geöffnet und die Synchronisationsphase schließt sich an.

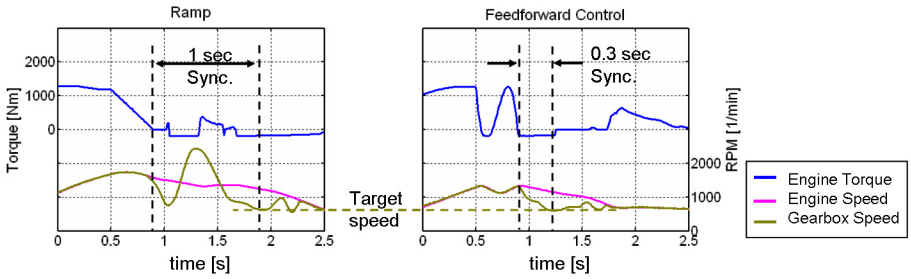


Abbildung 6: Vergleich Messergebnis mit/ohne Feedforward Strategie

Es ist deutlich erkennbar, dass im linken Bild die Restverspannung im Triebstrang zu starken Oszillationen der Drehzahlen führt. Dies erschwert die Synchronisation auf die neue Zieldrehzahl. Im rechten Bild ist der Abfall der Getriebedrehzahl allein durch die Aktuatorik (Vorgelegenwellenbremse) bedingt, eine überlagerte Schwingung ist kaum noch erkennbar. Durch die Maßnahme der gesteuerten Vorgabe kann in diesem Beispiel eine um 0.7 sec verkürzte Schaltzeit im Vergleich zu einem rampenförmigen Abbau erreicht werden. Es interessiert nun die Frage, wie sich solche Vorgaben auf die vom Fahrer gefühlten Beschleunigungen auswirken. Das nächste Kapitel widmet sich der Untersuchung dieser Auswirkungen.

3 Schwingungsverhalten einer Sattelzugmaschine

3.1 Modellbildung

Die luftgefederten Achsen, das luftgefederte Fahrerhaus und der verwindungsweiche Rahmen bilden zusammen mit dem Triebstrang ein schwingungsfähiges Gesamtsystem höherer Ordnung. Abbildung 7 zeigt ein Modell mit neun Freiheitsgraden, das die relevanten Bewegungen nachbilden kann. Aus Gründen der einfachen Handhabung werden konzentrierte, lineare Parameter angenommen. Die Motormasse ist über Getriebe, Hinterachsdifferential und die torsionsweichen Seitenwellen mit den Rädern verbunden. Der Reifen-Fahrbahnkontakt wird vereinfacht als starr angenommen. Der Rahmen ist verwindungsweich und deshalb in zwei Teile, die mit einer Feder verbunden sind, aufgeteilt. An Vorder- und Hinterachse sind die Nachgiebigkeiten und Dämpfungen der Achsfederung dargestellt. Das Fahrerhaus als Starrkörper ist auf vier Feder-Dämpferelementen auf dem vorderen Teil des Rahmens gelagert. Die Anhängermasse wird konzentriert in der Sattelkupplung angenommen.

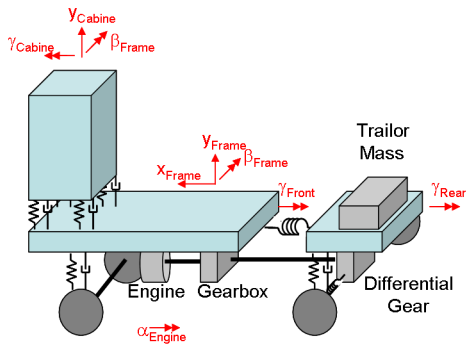


Abbildung 7: Schwingungsmodell einer Sattelzumaschine

Zur Generierung der für die Simulation notwendigen mathematischen Beschreibung müssen die Bewegungsgleichungen aufgestellt werden. Die Elemente der Steifigkeitsmatrix erhält man aus Einheitsverschiebungszuständen, die Dämpfungsmatrix aus Einheitsgeschwindigkeitszuständen und die Massenmatrix aus Einheitsbeschleunigungszuständen. Wichtig ist die korrekte Berücksichtigung der Lagerkräfte von Motor, Getriebe und Hinterachsdifferential. Das Motormoment wirkt als Stellgröße auf den Triebstrang und als Gegenmoment auf den vorderen Rahmen. Die Lagerkräfte von Getriebe und Hinterachsdifferential sind bei der Aufstellung der Steifigkeitsmatrix als innere Momente zu berücksichtigen. Das Getriebe stützt sich mit $M_G = (i_G - 1) * M_{Getriebeeingang}$ ab. Das Hinterachsdifferntial stützt sich mit $M_{HA} = (i_{HA} - 1) * M_G$ um die Fahrzeuglängsachse und mit dem vollen Moment der Seitenwellen (Radmoment) um die Querachse auf dem hinteren Rahmenteil ab. Die Bewegungsgleichung ergibt sich demnach in Matrixform:

$$\underline{M}^{(9x9)}\,\underline{\ddot{q}}^{(9x1)} + \underline{D}^{(9x9)}\,\underline{\dot{q}}^{(9x1)} + \underline{K}^{(9x9)}\,\underline{q}^{(9x1)} = \underline{f}^{(9x1)}$$

$$\underline{q} = \begin{bmatrix} \alpha_{Engine} & x_{Frame} & y_{Frame} & \beta_{Frame} & y_{Cabine} & \beta_{Cabine} & \gamma_{Cabine} & \gamma_{FrameFront} & \gamma_{FrameRear} \end{bmatrix}^T$$

$$\underline{f} = \begin{bmatrix} T_{Engine} & 0 & 0 & 0 & 0 & 0 & 0 & -T_{Engine} & 0 \end{bmatrix}^T$$

Abbildung 8 zeigt den Vergleich der so erhaltenen Modellbeschreibung mit einer Fahrzeugmessung, bei der ein Beschleunigungssensor im Fahrerhaus eingesetzt wurde. Der dargestellte Vorgang ist ein Motormomentenabbau vor einer Schaltung in einer Zeitdauer von 0.4 sec. Reihe 1 zeigt den Motormomentenverlauf. In Reihe 2 ist die im Fahrerhaus gemessene Längsbeschleunigung sowie die an der Rahmenkoordinate x_{Frame} wirksame Beschleunigung dargestellt. Letztere kann durch Ableiten der gemessenen Raddrehzahlen ermittelt werden. In Reihe 4 ist die aus der Fahrerhausvertikalbeschleunigung durch Integration errechnete Vertikalposition aufgetragen.

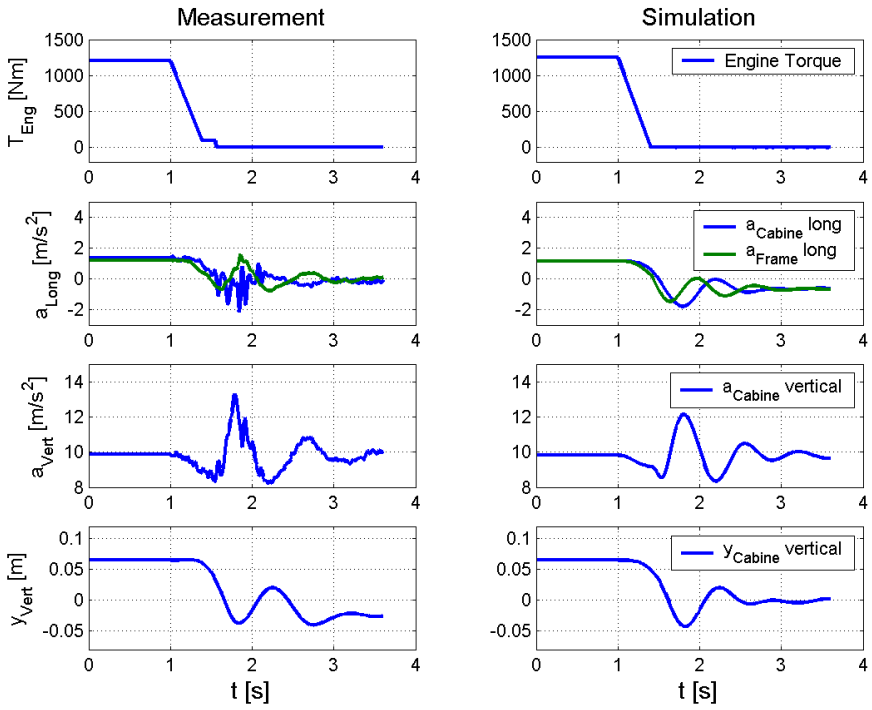


Abbildung 8: Vergleich Simulation - Messung bei rampenförmigen Momentenabbau

Die Verläufe zeigen eine gute Übereinstimmung von Modell und Messung für die charakteristischen Verläufe der Signale. Es ist zu erkennen, dass die Beschleunigung des Rahmens der Beschleunigung des Fahrerhauses vorauseilt, was von der Simulation bestätigt wird. Der Durchtaucher in der Vertikalposition repräsentiert den unteren Scheitelpunkt, in dem das Fahrerhaus nahezu auf den Anschlägen aufsetzt. Dies ist für den Fahrer im Fahrversuch deutlich spürbar.

3.2 Auswirkung gesteuerter Motormomentenverläufe

Die nach Kapitel 2 basierend auf einem Zweimassenmodell ausgelegten Steuerfunktionen werden nun zusammen mit dem Mehrkörpermodell simuliert. Für den Komfort relevant sind die vom Fahrer gefühlten Beschleunigungen im Fahrerhaus an der Position des Sitzes. Für die Funktion (Synchronisation) relevant ist das am Getriebe anliegende Radmoment zum Zeitpunkt des Auslegens des Ganges. In Abbildung 9 sind Simulationsergebnisse für den Verlauf dieser Größen sowie der Faherhausvertikalposition nach einem Momen-

tenabbauvorgang gezeigt. Links ist eine Motormomentenrampe, rechts ein Verlauf durch Feedforward Strategie - jeweils in 0.6 sec Abbauzeit - aufgeführt.

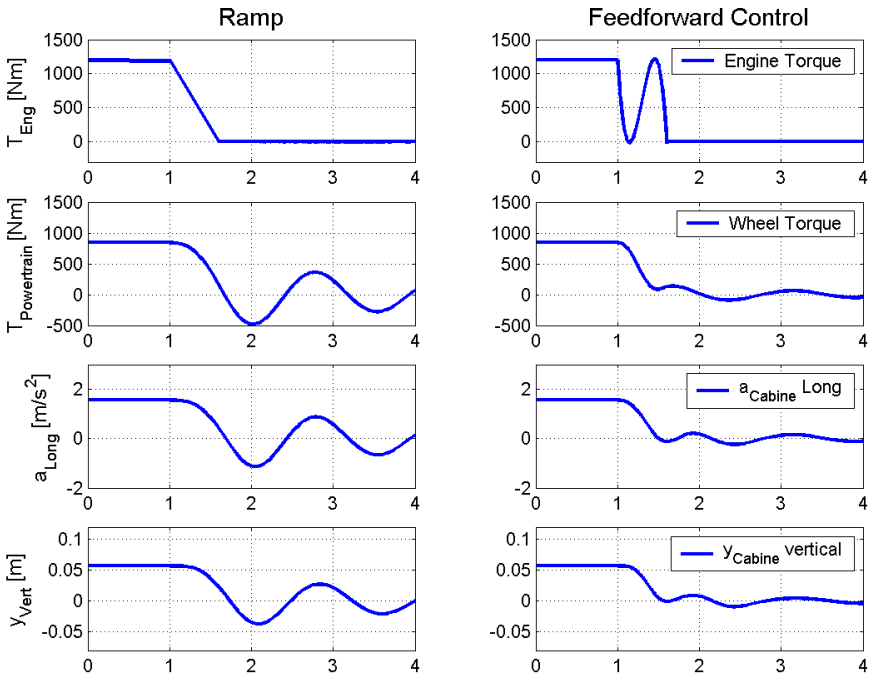


Abbildung 9: Vergleich Simulationsergebnisse mit/ohne Feedforward Strategie

Man erkennt unschwer die Verbesserung durch die gesteuerte Vorgabe auch bei Simulation mit dem komplexen Modell. Das Radmoment kann zwar nicht exakt auf Null geführt werden, liegt aber deutlich besser als bei einer Rampe. Auch der Überschwinger der Fahrerhauslängsbeschleunigung ist stärker gedämpft. Zur Veranschaulichung ist die Vertikalposition des Fahrerhauses aufgetragen. Hier ist bei einer Rampe ein stärkeres Absacken erkennbar. Zusammengefasst lässt sich eine positive Auswirkung der Feedforward-Strategie auch auf den Fahrkomfort in der Simulation erkennen. Das subjektive Empfinden bei entsprechenden Fahrversuchen bestätigt dies.

3.3 Ideale Steuerfunktionen

Die Verläufe bei der auf dem Zweimassenmodell basierenden Feedforward-Strategie sind theoretisch noch nicht ideal. Es treten noch Überschwinger bei Längsbeschleunigung und Vertikalposition bzw. eine Restverspannung im Triebstrang auf. Daher stellt sich die Frage, wie ideale Steuerkurven aussehen müssten, um theoretisch exakte Verläufe der Beschleunigungen bzw. dem Radmoment zu erhalten. Die mathematische Modellbeschreibung ermöglicht es, diese aus der Simulation zu ermitteln. Zunächst wird dazu eine Zustandsraumdarstellung aus den Bewegungsgleichungen abgeleitet (vgl. [3]):

$$\frac{d}{dt}\begin{bmatrix}\underline{q}\\ \underline{\dot{q}}\end{bmatrix} = \begin{bmatrix}\underline{0} & \underline{I}\\ -\underline{M}^{-1}\underline{K} & -\underline{M}^{-1}\underline{D}\end{bmatrix}\begin{bmatrix}\underline{q}\\ \underline{\dot{q}}\end{bmatrix} + \begin{bmatrix}\underline{0}\\ \underline{M}^{-1}\end{bmatrix}\underline{f} \;\Leftrightarrow\; \underline{\dot{x}} = \underline{A}\underline{x} + \underline{B}\underline{u}$$

$$\begin{bmatrix}\underline{q}\\ \underline{\dot{q}}\\ \underline{\ddot{q}}\end{bmatrix} = \begin{bmatrix}\underline{I} & \underline{0}\\ \underline{0} & \underline{I}\\ -\underline{M}^{-1}\underline{K} & -\underline{M}^{-1}\underline{D}\end{bmatrix}\begin{bmatrix}q\\ \dot{q}\end{bmatrix} + \begin{bmatrix}\underline{0}\\ \underline{0}\\ \underline{M}^{-1}\end{bmatrix}\underline{f} \;\Leftrightarrow\; \underline{y} = \underline{C}\underline{x} + \underline{D}\underline{u}$$

Damit sind alle Freiheitsgrade, deren erste und zweite Ableitung sowie beliebige Linearkombinationen derselben als Ausgangsgrößen verwendbar. So lassen sich für die Fahrerhauslängsbeschleunigung, die Fahrerhausvertikalbewegung und das Radmoment aus der Zustandsraumdarstellung Übertragungsfunktionen ableiten. Durch geeignete Modellreduktion und -erweiterungen kann ein inverses System für die Simulation entworfen werden. In Abbildung 10 sind so erhaltene ideale Motormomentenverläufe dargestellt. Abbildung 11 zeigt die Auswirkungen dieser Steuergrößenverläufe auf die drei komfort- und funktionsrelevanten Größen.

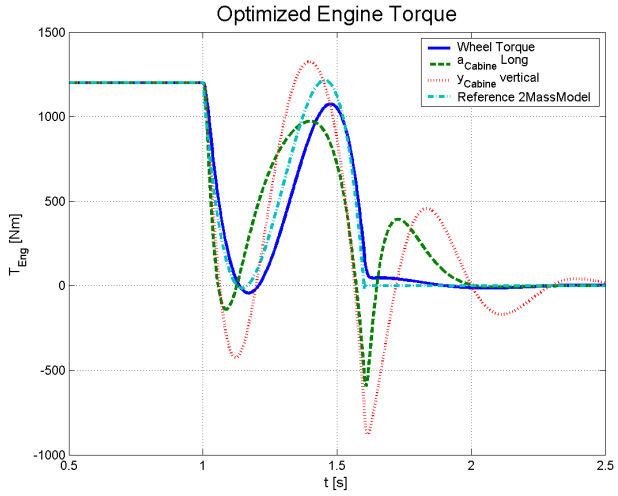


Abbildung 10: Ideale Steuerkurven für das Motormoment

Die Längsbeschleunigung, Vertikalposition und das Radmoment verlaufen auch bei optimierten Vorgaben hier nicht ganz exakt ohne Überschwingen. Die Ursache hierfür liegt in den vorgenommenen Modellreduktionen in Form von Vernachlässigung schneller Dynamik. Man erkennt einen engen Zusammenhang zwischen den Verläufen für optimale Fahrerhauslängsbeschleunigung und Fahrerhausvertikalbewegung. Die Steuerkurve für idealen Radmomentenverlauf weicht davon ab und ähnelt stark der aus dem Zweimassemodell gewonnenen Referenzkurve. Es ist festzustellen, dass die Stellgröße Motormoment bei theoretisch idealen Verläufen nicht in einer vorgegebenen Abbauzeit auf einen stationären Wert eingestellt ist. Systemtheoretisch erfüllen die Fahrerhauslängsbeschleunigung, die Faherhausvertikalposition und das Radmoment bei einem Mehrkörpermodellansatz für die Sattelzugmaschine nicht die Bedingungen für einen „flachen“ Systemausgang (vgl. [4]). Wegen der nicht definierbaren Zeitdauer sind solche idealen Vorgaben zumindest für den Momentenabbau vor einer Schaltung praktisch nicht umsetzbar, da hier eine möglichst kurze und definierte Dauer erforderlich ist. Anders kann es bei Lastwechseln

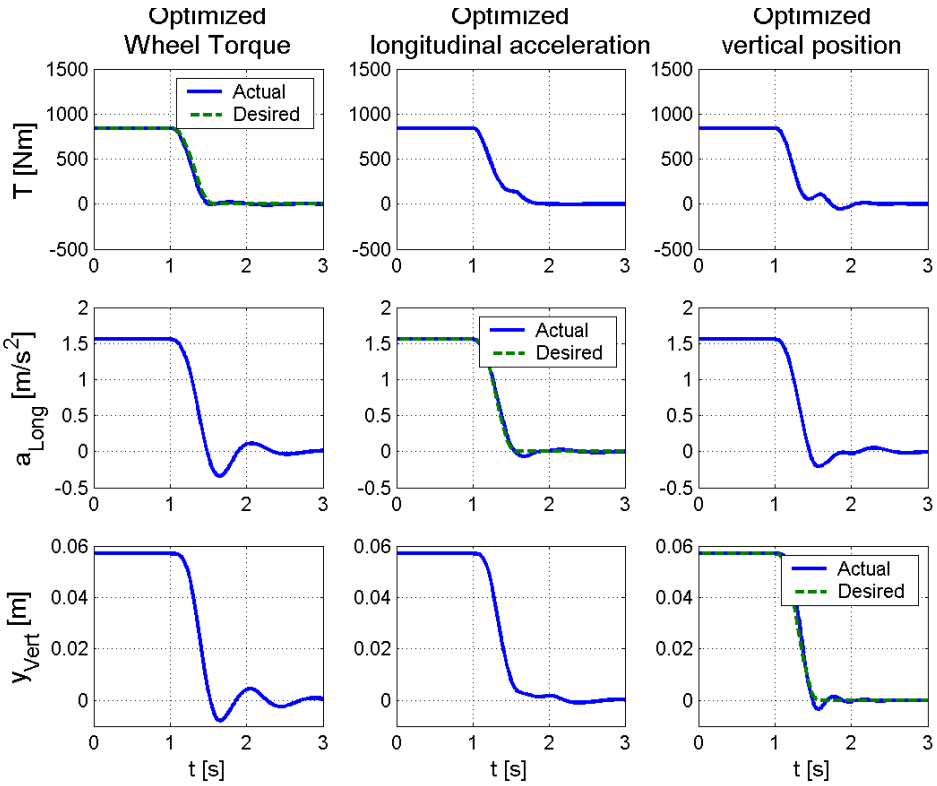


Abbildung 11: Systemreaktion auf ideale Steuerkurven

infolge Fahrpedaländerungen aussehen. Weiterhin ist zu berücksichtigen, dass die erforderlichen Stellmomente für idealen Fahrkomfort insbesondere im negativen Bereich sehr hoch und praktisch nicht realisierbar sind. Generell ist eine gleichzeitig theoretisch optimale Auslegung der Steuerkurven für idealen Fahrkomfort und Radmomentenverlauf nicht vollständig möglich. Die aus dem einfachen Zweimassenmodell erhaltenen Steuerkurven zeigen jedoch schon eine sehr gute Performance, die für die Praxis akzeptabel ist.

Literatur

[1] Hagerodt, Arnd: *Automatisierte Optimierung des Schaltkomforts von Automatikgetrieben*, Dissertation, Shaker Verlag, 2003.

[2] Joachim, Carsten: *Ein Verfahren zum Unterdrücken von Triebstrangschwingungen bei schweren Nutzfahrzeugen mit automatisiertem Schaltgetriebe*, 8th Stuttgart International Symposium, 2008.

[3] Nordmann, Rainer: *Umdruck zur Vorlesung Mechatronische Systeme im Maschinenbau*, TU Darmstadt, MiM, 2000.

[4] Rothfuß, Ralf: *Anwendung der flachheitsbasierten Analyse und Regelung nichtlinearer Mehrgrößensysteme*, Dissertation, VDI Fortschrittsberichte Reihe 8, Nr. 664, 1997.