REPORT 48

ARGESIM

ASIM Workshop GMMS/STS 2025 Tagungsband

ASIM WS GMMS/STS 2025

Workshop der ASIM Fachgruppen GMMS – Grundlagen und Methoden in Modellbildung und Simulation und STS – Simulation Technischer Systeme

DLR Oberpfaffenhofen
10.4. – 11.4.2025

Herausgeber: Walter Commerell, Umut Durak, Dirk Zimmer

ISBN ebook 978-3-903347-66-3
ARGESIM Report 48
www.argesim.org

DOI 10.11128/arep.48
ASIM Mitteilung 193
www.asim-gi.org

ASIM Books – ASIM Book Series – ASIM Buchreihen

Monographs

Simulation-based Optimization: Industrial Practice in Production and Logistics
Lothar März, Markus Rabe, Oliver Rose (Eds.); to appear; ASIM Mitteilung 191

Energy-related Material Flow Simulation in Production and Logistics.
S. Wenzel, M. Rabe, S. Strassburger, C. von Viebahn (Eds.); Springer Cham 2023, print ISBN 978-3-031-34217-2, eISBN 978-3-031-34218-9, DOI 10.1007/978-3-031-34218-9, ASIM Mitteilung 182

Kostensimulation - Grundlagen, Forschungsansätze, Anwendungsbeispiele
T. Claus, F. Herrmann, E. Teich; Springer Gabler, Wiesbaden, 2019; Print ISBN 978-3-658-25167-3; Online ISBN 978-3-658-25168-0; DOI 10.1007/978-3-658-25168-0; ASIM Mitteilung 169

Proceedings*

Tagungsband ASIM Workshop 2025 - GMMS/STS - ASIM Fachgruppenworkshop 2025, DLR Oberpfaffenhofen
W. Commerell, U. Durak, D. Zimmer (Hrgs.), ARGESIM Report 48; ASIM Mitteilung AM 193
ISBN ebook 978-3-903347-66-3, DOI 10.11128/arep.48, ARGESIM Verlag Wien, 2025

Tagungsband Kurzbeiträge ASIM SST 2024 -27. ASIM Symposium Simulationstechnik, Univ. Bundeswehr München,
München/Neubiberg, Sept. 2024, O. Rose, T. Uhlig (Hrgs.), ARGESIM Report 46; ASIM Mitteilung AM 189
ISBN ebook 978-3-903347-64-9, DOI 10.11128/arep.46, ARGESIM Verlag Wien, 2024

Tagungsband Langbeiträge ASIM SST 2024 -27. ASIM Symposium Simulationstechnik, Univ. Bundeswehr München
München/Neubiberg, Sept. 2024,O. Rose, T. Uhlig (Hrgs.), ARGESIM Report 47; ASIM Mitteilung AM 190
ISBN ebook 978-3-903347-65-6, DOI 10.11128/arep.47, ARGESIM Verlag Wien, 2024

Simulation in Production and Logistics 2023 – 20. ASIM Fachtagung Simulation in Produktion und Logistik
TU Ilmenau, September 2023; S. Bergmann, N. Feldkamp, R. Souren, S. Straßburger (Hrsg.);
ASIM Mitteilung 187; ISBN ebook 978-3-86360-276-5, DOI: 10.22032/dbt.57476, Universitätsverlag Ilmenau, 2023

Proceedings Langbeiträge ASIM Workshop 2023 - STS/GMMS/EDU - ASIM Fachgruppenworkshop 2023
Univ. Magdeburg, März 2023; C. Krull; W. Commerell, U. Durak, A. Körner, T. Pawletta (Hrsg.)
ARGESIM Report 21; ASIM Mitteilung 185; ISBN ebook 978-3-903347-61-8, DOI 10.11128/arep.21, ARGESIM Verlag, Wien, 2023

Kurzbeiträge & Abstract-Beiträge ASIM Workshop 2023 STS/GMMS/EDU - ASIM Fachgruppenworkshop 2023
Univ. Magdeburg, März 2023; C. Krull; W. Commerell, U. Durak, A. Körner, T. Pawletta (Hrsg.)
ARGESIM Report 22; ASIM Mitteilung 186; ISBN ebook 978-3-903347-62-5, DOI 10.11128/arep.22, ARGESIM Verlag, Wien, 2023

Proceedings Langbeiträge ASIM SST 2022 -26. ASIM Symposium Simulationstechnik, TU Wien, Juli 2022
F. Breitenecker, C. Deatcu, U. Durak, A. Körner, T. Pawletta (Hrsg.), ARGESIM Report 20; ASIM Mitteilung AM 180
ISBN ebook 978-3-901608-97-1, DOI 10.11128/arep.20, ARGESIM Verlag Wien, 2022

Proceedings Kurzbeiträge ASIM SST 2022 -26. ASIM Symposium Simulationstechnik, TU Wien, Juli 2022
F. Breitenecker, C. Deatcu, U. Durak, A. Körner, T. Pawletta (Hrsg.), ARGESIM Report 19; ASIM Mitteilung AM 179
ISBN ebook 978-3-901608-96-4, DOI 10.11128/arep.19, ISBN print 978-3-901608-73-5, ARGESIM Verlag Wien, 2022

Book Series Fortschrittsberichte Simulation – Advances in Simulation *

An Architecture for Model Behavior Generation for Multiple Simulators. H. Folkerts, FBS 42
ISBN ebook 978-3-903347-42-7, DOI 10.11128/fbs.32, ARGESIM Publ. Vienna,2024

Das Verhalten von Transuranelementen in Erdböden - Theorie, Beprobung und radiochemische Analysen. K. Breitenecker, FBS 41; ISBN ebook 978-3-903347-41-0, DOI 10.11128/fbs.41, 2024; ISBN print 978-3-901608-99-5, 2010/2024; ARGESIM Publ. Vienna

Aufgabenorientierte Multi-Robotersteuerungen auf Basis des SBC-Frameworks und DEVS. B. Freymann, FBS 40
ISBN ebook2020_978-3-903347-40-3, DOI 10.11128/fbs.40, ARGESIM Publ. Vienna, 2022

Cooperative and Multirate Simulation: Analysis, Classification and New Hierarchical Approaches. I. Hafner, FBS 39
ISBN ebook978-3-903347-39-7, DOI 10.11128/fbs.39, ARGESIM Publ. Vienna,2022

Die Bedeutung der Risikoanalyse für den Rechtsschutz bei automatisierten Verwaltungsstrafverfahren. T. Preiß, FBS 38
ISBN ebook 978-3-903347-38-0, DOI 10.11128/fbs.38, ARGESIM Publ. Vienna,2020

Methods for Hybrid Modeling and Simulation-Based Optimization in Energy-Aware Production Planning. B. Heinzl, FBS 37
ISBN ebook 978-3-903347-37-3, DOI 10.11128/fbs.37, ARGESIM Publ. Vienna,2020;

Konforme Abbildungen zur Simulation von Modellen mit verteilten Parametern. Martin Holzinger, FBS 36
ISBN ebook 978-3-903347-36-6, DOI 10.11128/fbs.36, ARGESIM Publ. Vienna, 2020

Fractional Diffusion by Random Walks on Hierarchical and Fractal Topological Structures. G. Schneckenreither, FBS 35
ISBN ebook 978-3-903347-35-9, DOI 10.11128/fbs.35, ARGESIM Publ. Vienna, 2024

A Framework Including Artificial Neural Networks in Modelling Hybrid Dynamical Systems. Stefanie Winkler, FBS 34
ISBN ebook 978-3-903347-34-2, DOI 10.11128/fbs.34, ARGESIM Publ. Vienna, 2020

Modelling Synthesis of Lattice Gas Cellular Automata and Random Walk and Application to Gluing of Bulk Material. C. Rößler, FBS 33
ISBN ebook 978-3-903347-33-5, DOI 10.11128/fbs.33, ARGESIM Publ. Vienna, 2021

Combined Models of Pulse Wave and ECG Analysis for Risk Prediction in End-stage Renal Desease Patients. S. Hagmair, FBS 32
ISBN ebook 978-3-903347-32-8, DOI 10.11128/fbs.32, ARGESIM Publ. Vienna, 2024

Mathematical Models for Pulse Wave Analysis Considering Ventriculo-arterial Coupling in Systolic Heart Failure. S. Parragh, FBS 31
ISBN ebook 978-3-903347-31-1, DOI 10.11128/fbs.31, ARGESIM Publ. Vienna, 2024

Variantenmanagement in der Modellbildung und Simulation unter Verwendung des SES/MB Frameworks. A. Schmidt, FBS 30; ISBN ebook 978-3-903347-30-4, DOI 10.11128/fbs.30, ARGESIM Verlag, Wien 2019

Classification of Microscopic Models with Respect to Aggregated System Behaviour. Martin Bicher, FBS 29
ISBN ebook 978-3-903347-29-8, DOI 10.11128/fbs.29, ARGESIM Publ. Vienna, 2020

*** Download** via ASIM **www.asim-gi.org** Open Access - Basic Version Member Access - Enhanced Version

ASIM Workshop GMMS/STS 2025

Tagungsband

10.4. – 11.4. 2025

DLR Oberpfaffenhofen

Workshop der ASIM Fachgruppen
GMMS – Grundlagen und Methoden in Modellbildung und Simulation und
STS – Simulation Technischer Systeme

Herausgegeben von
Walter Commerell (TH Ulm), Umut Durak (DLR Braunschweig),
und Dirk Zimmer (DLR Oberpfaffenhofen)

ISBN ebook 978-3-903347-66-3 DOI 10.11128/arep.48
ARGESIM Report 48 ASIM Mitteilung 193
ARGESIM Publisher, Wien, 2025
www.argesim.org, www.asim-gi.org

Bibliographic Data:

Title: ASIM Workshop GMMS/STS 2025 Tagungsband

Subtitle: Workshop der ASIM-Fachgruppen
GMMS – Grundlagen und Methoden in Modellbildung und Simulation und
STS – Simulation Technischer Systeme, 10.4.-11.4.2025, DLR Oberpfaffenhofen

Editors: Walter Commerell (TH Ulm), Umut Durak (DLR Braunschweig),
Dirk Zimmer (DLR Oberpfaffenhofen)

Series: ARGESIM Reports

Series Editors: Felix Breitenecker, Thorsten Pawletta, ASIM

Volume: ARGESIM Report AR 48

ISBN ebook: 978-3-903347-66-3, ARGESIM Verlag

DOI: 10.11128/arep.48

ASIM ID: ASIM Mitteilung AM 193

Publication Date: 8.4.2025

Number of Pages: v + 100 pages

Cover: “David”. David ist ein Humanoider Roboter mit geschickten Manipulationsfähigkeiten.
Credit: DLR (CC BY-NC-ND 3.0); www.dlr.de/de/rm/forschung/robotersysteme/humanoide/david

About ARGESIM

ARGESIM is a non-profit society generally aiming for dissemination of information on system simulation from research via development to applications of system simulation. **ARGESIM** is closely co-operating with **EUROSIM**, the Federation of European Simulation Societies, and with **ASIM**, the German Simulation Society. **ARGESIM** is an 'outsourced' activity from the Mathematical Modelling and Simulation Group of TU Wien, there is also close co-operation with TU Wien (organisationally and personally).
ARGESIM Publisher organizes publishing activities, with ISBN roots 978-3-901608-X and 978-3-903347-X, and DOI root 10.11128/xx…x.

ARGESIM’s activities are:

- Publication of the scientific journal **SNE** - Simulation Notes Europe (Membership Journal of EUROSIM, the Federation of European Simulation Societies) → *www.sne-journal.org*
- Organisation and Publication of the **ARGESIM Benchmarks** for Modelling Approaches and Simulation Implementations → *www.argesim.org/benchmarks/*
- Publication of the series **ARGESIM Reports** (for monographs in system simulation, and proceedings of simulation conferences and workshops) → *www.argesim.org/publications/*
- Publication of the special series **FBS Simulation** - Advances in Simulation / Fortschrittsberichte Simulation (monographs in co-operation with **ASIM**, the German Simulation Society
- Organisation of the Conference Series MATHMOD Vienna (triennial, in co-operation with **EUROSIM**, **ASIM**, and TU Wien) → *www.mathmod.at*
- Administration and support of **ASIM** (German Simulation Society → *www.asim-gi.org*) and of **EUROSIM** (Federation of European Simulation Societies → *www.eurosim.info*)

ARGESIM – Arbeitsgemeinschaft Simulation News – Working Committee Simulation News – SNE Publication
Mommsengasse 19/8, 1040 Vienna, Austria; Tel +43-1-58801-10111, -10115
Email: office@argesim.org, office@sne-journal.org; WWW: www.argesim.org, www.sne-journal.org
Incorporated Austrian Society ZVR No 213056164 – EU VAT ID No ATU 72054279
Bank Account: ARGESIM, IBAN AT07 2011 1828 9115 0800, BIC GIBAATWWXXX, ERSTE BANK VIENNA

Vorwort

Simulationstechnologie ist in vielen Bereichen und insbesondere im Entwicklungsprozess technischer Systeme fest verankert. Der jährliche Workshop der **ASIM/GI**-Fachgruppen **STS** (Simulation Technischer Systeme) und **GMMS** (Grundlagen und Methoden in Modellbildung und Simulation) bietet eine breite Plattform, um sowohl den aktuellen technologischen Stand als auch zukünftige Chancen in Theorie, Praxis und Ausbildung zu beleuchten.

Der **ASIM Workshop GMMS/STS** fand 2025 als Präsenzveranstaltung am DLR Institut für Systemdynamik und Regelungstechnik statt, und bot eine offene Atmosphäre für lebendige Diskussionen und den Informations- und Erfahrungsaustausch zwischen Fachleuten aus Hochschulen, Forschungseinrichtungen und der Industrie.

Den Rahmen bildete ein Besuch der Labore am DLR Institut und der Besuch des Satellitenkontrollzentrums des DLRs Oberpfaffenhofen.

Dank vieler engagierter Autorinnen und Autoren wurden hochqualitative Beiträge mit aktuellen Forschungsergebnissen eingereicht. Neben den traditionellen Inhalten des Workshops waren wieder aktuelle Themen aus den Bereichen Energietechnik, Thermodynamik, Fahrzeugtechnik, etc. vertreten.

Zwei interessante Keynote-Vorträge bildeten den Rahmen des Programms. Sie beleuchteten die Themen *Die Herausforderung hoher Frequenzen: neue Modellierungsmethodiken für moderne Rechnerarchitekturen* und *Mathematical Problems due to Oversimplification* und führten zu angeregten Diskussionen im Nachgang.

Der Tagungsband enthält die Beiträge des Workshops, die unter anderem folgende Themenschwerpunkte behandeln:

- Modellbasierte Mechatronikentwicklung
- Modellbildung und Simulation mechanischer, thermischer und pneumatischer Systeme
- Modellbildung und Simulation von Marktmodellen
- Mathematische Verfahren in Modellbildung und Simulation
- Grundlagen und Methoden in Modellbildung und Simulation
- Echzeitsimulation
- Simulation Technischer Systeme
- Angewandte Simulationen

Dieser Tagungsband, **ASIM Workshop GMMS/STS 2025 Tagungsband**, ISBN ebook 978-3-903347-66-3, DOI 10.11128/arep.48, ARGESIM Report 48, ASIM Mitteilung 193, publiziert die 14 Beiträge, die vom Programmkomitee angenommen wurden. Für Form und Inhalt der in den Tagungsbänden enthaltenen Beiträge sind die Autoren selbst verantwortlich.

Entsprechend ASIMs Publikationsstrategie sind Tagungsband und Einzelbeiträge als Open Access in Basisversion auf www.asim-gi.org verfügbar. ASIM-Mitgliedern stehen die Vollversionen sowie die Sammlung der Vortragsfolien mit individuellem Login zur Verfügung (für Workshop-Teilnehmer auf der Tagungswebsite www.asim-gi.org/ws2025 bis Ende 2025 mit Gruppenlogin).

Für die Unterstützung und Mithilfe bei der Organisation des diesjährigen Workshops möchten wir uns sehr herzlich bei *Madeleine Breitkreuz, Pascal Krenckel,* und *Thorsten Pawletta,* sowie bei allen weiteren Beteiligten vor Ort am Institut für Systemdynamik und Regelungstechnik der DLR bedanken. Außerdem gilt mein Dank den Reviewern für die tatkräftige Unterstützung bei der Auswahl und Verbesserung der Einreichungen, und dem Team von ARGESIM Publisher für das Layout.

Walter Commerell, Umut Durak, Dirk Zimmer *Oberpfaffenhofen, im April 2025*

Inhaltsverzeichnis – List of Content

ASIM Workshop GMMS/STS 2025

Beiträge und Abstracts Hauptvorträge

Autorenindex – Index of Authors

ASIM Workshop GMMS/STS 2025

Beiträge und Abstracts Hauptvorträge

Die Herausforderung hoher Frequenzen: neue Modellierungsmethodiken für moderne Rechnerarchitekturen

Dirk Zimmer

Deutsches Zentrum für Luft- und Raumfahrt (DLR), Institut für Robotik und Mechatronik, Oberpfaffenhofen.
82234 Weßling, Deutschland; *dirk.zimmer@dlr.de*

Abstract.
Die Frequenzen fundamentaler physikalischer Prozesse im mikroskopischen Bereich trennen oft zahlreiche Größenordnungen zu den Frequenzen der resultierenden Epiphänomene auf makroskopischer Ebene.
Diese inhärente Steifheit physikalischer Systeme stellt schon immer eine Herausforderung für ihre Simulation dar.
Dieses Problem gewinnt erneut an Relevanz, da moderne Rechenarchitekturen eine enorme Menge an Rechenleistung zu geringen Kosten bereitstellen - allerdings nur in begrenzter Frequenz.
Die sinnvolle Einschränkung des Frequenzraums für die Simulation ist daher eine der zentralen Aufgaben eines Modellierers.
Der Beitrag zeigt einige verschiedene Herangehensweisen für die Einschränkung des Frequenzraums auf.

Mathematical Problems due to Oversimplification

Peter Junglas

PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany;
peter@peter-junglas.de

Abstract.
An important step in modeling is simplification: A model should contain only those aspects of a system that are relevant for the modeling purpose.
But sometimes the exclusion of seemingly unimportant physical details leads to models that are mathematically ill defined and cannot be simulated. In such cases it can be helpful to include a rudimentary version of the omitted physics.
Using four examples from different application areas it will be shown, how this can be done efficiently and without the introduction of lots of additional parameters, for which proper values would have to be supplied.

Optimized operation of a multi-energy system for a small P-t-X facility in Ulm

Samuel Peifer[1*], Peter Renze[1]

[1]Institute for Energy Technology and Energy Economics, University of Applied Science, Albert-Einstein-Allee 55, 89081 Ulm, Germany; [*]*samuel.peifer@thu.de*

Abstract. One of the main challenges of renewable energies, such as photovoltaic (PV) and wind power, is their highly fluctuating nature. As a result, energy is often unavailable when demand arises. Therefore, an essential aspect of ensuring a reliable supply from renewable sources is the ability to store energy. A combination of hydrogen and a battery energy storage system presents a promising approach. This paper proposes a methodology to model an existing Power-to-X (P-t-X) facility at the system level in Python. The electrical load and heat demand of a household are considered. A core aspect of this work is the comparison of rule-based operation strategy with a more advanced approach that incorporates model predictive control (MPC) combined with an univariate Long Short-Term Memory (LSTM) neural network and a Mixed-Integer Linear Program (MILP). The study investigates whether such an advanced strategy can improve the operation of a P-t-X facility in terms of self-sufficiency and self-consumption.

Introduction

Hydrogen produced from renewable electricity is a zero-emission energy carrier and is expected to play a major role in a future climate-neutral economy [1]. As proposed by Kalchschmid et al. [2], hydrogen serves as a suitable energy storage solution from both an economic and environmental perspective. However, further development of operating strategies for individual plants remains necessary. Moreover, Kalchschmid et al. suggest that utilizing waste heat could be a valuable extension of potential use cases.

Building on these findings, this work proposes a methodology to model a small-scale P-t-X facility and apply an appropriate operating strategy. The paper is structured as follows: First, a brief system description of the P-t-X facility is given, then the methodology to physically model the involved systems and the different operation strategies are described. Subsequently, the case setup is described and the different operational strategies are compared to each other. Finally, conclusions are drawn.

1 System Description

In the following section, the systems used to model the P-t-X facility and the household are described. First the dependencies for the electrical infrastructure, Power-to-Gas (P-t-G) and Power-to-Power (P-t-P), are shown. Afterwards the dependencies for the thermal integration, Power-to-Heat (P-t-H), are described.

1.1 Electrical Integration

The P-t-G and P-t-P dependencies modeled in this work represent a small-scale energy system that integrates renewable electricity generation with different storage technologies. The system dependencies for the model are shown in Figure 1.

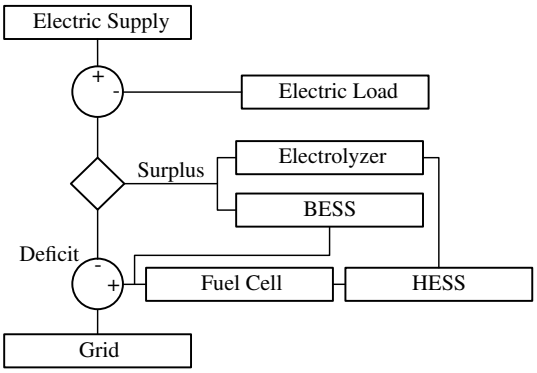


Figure 1: System Model of the P-t-G / P-t-P infrastructure [3]

The system consists of a photovoltaic (PV) facility, which is the main source of renewable energy. The electrical energy from the PV system can be used to directly

supply the household's electric load or can be stored in a battery energy storage system (BESS). Furthermore surplus electricity can be converted into hydrogen via an electrolyzer. The produced hydrogen is stored in a hydrogen energy storage system (HESS) and can be reconverted into electricity by a fuel cell during times of energy deficit.

1.2 Thermal Integration

Additionally to the electrical energy flows, the system model incorporates a thermal integration. The system dependencies for the P-t-H system are shown in Figure 2.

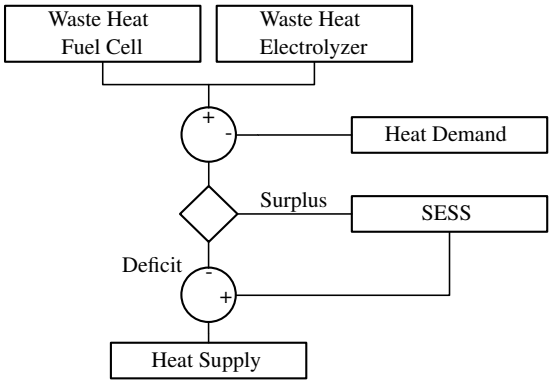


Figure 2: System Model of the waste heat utilization [3]

The heating demand of a representative household is considered and expressed as a function of the ambient temperature and the desired indoor setpoint temperature. The waste heat of the electrolyzer and the fuel cell can be utilized to meet the thermal demand directly or to store a possible surplus in an sensible energy storage system (SESS).

2 Methodology

2.1 Simulation Framework

The whole simulation is set up in python. The temporal resolution is hourly ($dt = 1h$) and one year with 8760 h are considered. Data input for the simulation is the photovoltaic power generation, the ambient temperature and the electrical demand.

2.2 Thermal Demand Modeling

The heat demand for each time step can be calculated by Equation 1[4].

$$\dot{Q}^t_{\mathrm{Loss}} = U_{\mathrm{Total}} \cdot A \cdot (T_{\mathrm{Set}} - T^t_{\mathrm{Amb}}) \tag{1}$$

Where A denotes the surface area through which heat is transferred, and U_{Total} represents the overall heat transfer coefficient, defined as follows [4]:

$$U_{\mathrm{Total}} = \frac{1}{\frac{1}{h_{\mathrm{in}}} + \frac{d_1}{k_1} + \frac{d_2}{k_2} + \frac{1}{h_{\mathrm{out}}}} \tag{2}$$

h_{in} and h_{out} denote the heat transfer coefficients for the inside and the outside of the wall. d_1 and k_1 are the thickness and the thermal conductivity of layer 1 and d_2 and k_2 the thickness and the thermal conductivity of layer 2. The heat provided by a heating system can be described as follows:

$$\dot{Q}^t_{\mathrm{Provided}} = \dot{m} \cdot c_{\mathrm{p}} \cdot (T_{\mathrm{supply}} - T_{\mathrm{return}}) \tag{3}$$

$\dot{m}$ is the massflow of the heating system, c_{p} the specific heat capacity, T_{supply} the supply temperature and T_{return} the return temperature. For this simulation we assume that the massflow is adjusted such that a constant temperature spread is achieved. To calculate the maximal amount of sensible heat that can be stored in a storage system, Equation 4 is used.

$$Q_{\mathrm{Usable}} = m_{\mathrm{Storage}} \cdot c_{\mathrm{p}} \cdot (T_{\mathrm{supply}} - T_{\mathrm{return}}) \tag{4}$$

m_{Storage} denotes the maximal storage mass, c_{p} the fluids specific heat capacity and T_{supply} and T_{return} are the inlet and outlet temperatures of the fluid, respectively

2.3 Storage System Equations

The used equations for the battery energy storage system (BESS), the hydrogen energy storage system (HESS) and the thermal energy storage system (TESS) are stated in Equation 5, 6 and 7:

$$\begin{aligned} E^{t+1}_{\mathrm{BESS}} &= E^t_{\mathrm{BESS}} \\ &+ \left(\eta_{\mathrm{BESS,ch}} \cdot P^t_{\mathrm{BESS,\,ch}} - \frac{P^t_{\mathrm{BESS,\,dis}}}{\eta_{\mathrm{BESS,dis}}} \right) \cdot dt \end{aligned} \tag{5}$$

$$E^{t+1}_{\mathrm{HESS}} = E^t_{\mathrm{HESS}} + \left(\eta_{\mathrm{Ely}} \cdot P^t_{\mathrm{Ely}} - \frac{P^t_{\mathrm{FC}}}{\eta_{\mathrm{FC}}} \right) \cdot dt \tag{6}$$

$$E^{t+1}_{\mathrm{TESS}} = E^t_{\mathrm{TESS}} + (\dot{Q}^t_{\mathrm{TESS,ch}} - \dot{Q}^t_{\mathrm{TESS,dis}}) \cdot dt \tag{7}$$

$P^t_{\text{BESS, ch}}$ and $P^t_{\text{BESS, dis}}$ denote the charge and discharge power of the BESS. The BESS can either be charged or discharged in one time step. P^t_{Ely} and P^t_{FC} are the electrolyzer and fuel cell power and both can be operated in parallel. The TESS can as well be charged and discharged at the same time. The discharge power $\dot{Q}^t_{\text{TESS,dis}}$ of the TESS is simply the drain rate when there is a demand. The charge power $\dot{Q}^t_{\text{TESS,ch}}$ depends on the electrolyzers and fuel cells power and is expressed in Equation 8.

$$\dot{Q}^t_{\text{TESS,ch}} = \eta_{\text{Ely,th}} \cdot P^t_{\text{Ely}} + \eta_{\text{FC,th}} \cdot P^t_{\text{FC}} \tag{8}$$

The lower and upper boundaries for the BESS, the HESS and the TESS are shown in equation 9, 10 and 11 respectivly. Those system boundaries hold true for the rule-based and MPC approach.

$$\text{E}_{\text{BESS,min}} \leq \text{E}^t_{\text{BESS}} \leq \text{E}_{\text{BESS,max}} \tag{9}$$

$$\text{E}_{\text{HESS,min}} \leq \text{E}^t_{\text{HESS}} \leq \text{E}_{\text{HESS,max}} \tag{10}$$

$$\text{E}_{\text{TESS,min}} \leq \text{E}^t_{\text{TESS}} \leq \text{E}_{\text{TESS,max}} \tag{11}$$

2.4 Rule-Based Control

The rule-based control strategy follows a simple heuristic approach based on predefined priority rules for energy distribution and storage operation. At each time step, the available photovoltaic power is first used to cover the electrical demand of the household. Surplus is then directed to charge the BESS, taking into account its maximum charging power and capacity limits. Additional surplus is then utilized to operate the electrolyzer for hydrogen production. On the contrary, in times of power deficit, the BESS is discharged first to meet the demand. If the battery is discharged or not able to meet the remaining load, the fuel cell is run to convert stored hydrogen into electricity. If there are still remaining deficits energy is drawn from the electrical grid. To meet the heat demand the recovered waste heat from the electrolyzer and fuel cell is used. The TESS is discharged if the waste heat utilization is insufficient. Afterwards, any remaining thermal demand is covered by an external supply.

2.5 Model Predictive Control

The core of the MPC-based operation strategy is the formulation of an optimization problem that minimizes the external energy supply from both the electrical and thermal grid. The formulation of the MPC is based on the work of Huang [5]. The optimization is solved as a Mixed-Integer Linear Program (MILP) over a finite prediction horizon, with only the first timestep being applied to the system. The MILP is set up in `Pyomo` and solved with the solver `Gurobi`.

Objective Function

The objective function for the optimization used in the MPC is the minimization of the electrical and the thermal grid supply and aims to minimize over the prediction horizon H_{Pred}:

$$\min \sum_{k=1}^{H_{Pred}} \left(P^k_{\text{Grid}} + \dot{Q}^k_{\text{Grid}} \right) \tag{12}$$

Constraints

The constraints, which describe the physical limits of the energy system, can be separated into power balance equations, operation limits of the energy storage systems and power boundaries for the system components (i.e. charging/discharging battery, running electrolyzer and fuel cell). Equation 13 denotes the electrical power balance and Equation 14 the thermal power balance of the system:

$$\begin{aligned} &P^k_{\text{Grid}} + P^k_{\text{PV}} + P^k_{\text{FC}} + P^k_{\text{BESS, dis}} \\ &= P^k_{\text{Load}} + P^k_{\text{Ely}} + P^k_{\text{BESS, ch}} \end{aligned} \tag{13}$$

$$\begin{aligned} &\dot{Q}^k_{\text{Grid}} + \dot{Q}^k_{\text{Ely}} + \dot{Q}^k_{\text{FC}} + \dot{Q}^k_{\text{TESS,dis}} \\ &= \dot{Q}^k_{\text{Loss}} + \dot{Q}^k_{\text{TESS,ch}} \end{aligned} \tag{14}$$

The system boundaries for the BESS, the HESS and the TESS are already shown in equations 9, 10 and 11. The charging and discharging constraints for the BESS are defined in equation 15 and equation 16.

$$0 \leq P^k_{\text{BESS,ch}} \leq (1 - \delta^k_{\text{BESS}}) \cdot P_{\text{BESS,ch,max}} \tag{15}$$

$$0 \leq P^k_{\text{BESS,dis}} \leq \delta^k_{\text{BESS}} \cdot P_{\text{BESS,dis,max}} \tag{16}$$

δ^k_{BESS} is the binary decision variable to prevent a simultaneous charging and discharging of the BESS during the optimization.

The power constraints for the electrolyzer and the fuel cell are stated in equation 17 and 18.

$$\delta^k_{\text{Ely}} \cdot P_{\text{Ely,min}} \leq P^k_{\text{Ely}} \leq \delta^k_{\text{Ely}} \cdot P_{\text{Ely,max}} \tag{17}$$

$$\delta^k_{\mathrm{FC}} \cdot P_{\mathrm{FC,min}} \leq P^k_{\mathrm{FC}} \leq \delta^k_{\mathrm{FC}} \cdot P_{\mathrm{FC,max}} \tag{18}$$

δ^k_{Ely} and δ^k_{FC} are the binary decision variables for the electrolyzer and the fuel cell to enforce power limits.

Time Series Forecasting

Similar to Kreuzer [6] a univariate Long Short-Term Memory (LSTM) neural networks is implemented for time series forecasting. The LSTM is used to enable the predictive control within the MPC framework. Two separate networks are trained, one for the photovoltaic power generation and one for the thermal demand. The same hyperparameters are used for both networks and are shown in Table 1. The data sets to train the networks range over four years from 2015 to 2018 and including training, validation, and testing of the models. To allow the model to capture temporal patterns the input of the network includes past values of the PV power or the thermal demand. The training data set for the photovoltaic power generation and the thermal demand, which is a function of the ambient temperature, are shown in Figure 3. The time series of photovoltaic power and ambient temperature used is from `www.renewables.ninja` and a description on how the data is generated can be found in [7] and [8].

Hyperparameter	Value
Forecast horizon	24 hours
Input sequence length	24 hours
# of LSTM layers	1
# of units (neurons)	32
Dropout rate	0.1
Activation function	tanh + 1
Loss function	Mean Squared Error (MSE)
Optimizer	Adam
Learning rate	0.001
Batch size	128
Number of epochs	100
Validation split	20%
Normalization	Min-Max Scaling (0–1)
Framework	TensorFlow/Keras

Table 1: Hyperparameters used for the LSTM models

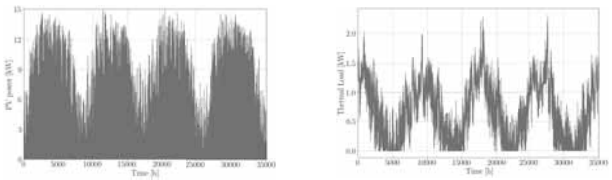


Figure 3: Training data used for the PV power generation and the thermal demand forecasting (2015–2018)

3 Case Setup

3.1 Data Sources

The photovoltaic power generation and the ambient temperature used for the simulation are from 2019 for the federal state of Baden-Württemberg and obtained from `www.renewables.ninja` [7][8]. The data has an hourly resolution and a total amount of 8760 data points.

Electrical Demand

The electrical demand is modeled using the standard load profile H0 for households proposed by the *Bundesverband der Energie- und Wasserwirtschaft*. An examplary week of the standard load profile used in this work is shown in Figure 4. The integral electric demand for this household is set to 4600 kWh.

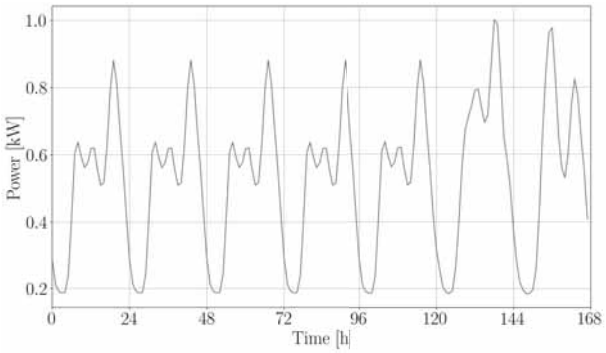


Figure 4: Exemplary week for the electrical demand of the household

Thermal Demand

The calculation of the thermal demand is accordingly to Equation 1. The surface area is set for a $10 \cdot 10\,m^2$ big room with a ceiling height of $2.5\,m$. Heat can be transferred through the vertical walls and the ceiling which leads to a total heat transfer surface of $200\,m^2$. The heat

tranfer coefficients h_{in} and h_{out} are calculated by correlations for natural convection at vertical walls and horizontal walls [9]. The wall parameters for layer 1 are set to $d_1 = 0.24\,m$ and $k_1 = 0.7\,\frac{W}{m \cdot K}$ and for layer 2 to $d_1 = 0.07\,m$ and $k_1 = 0.04\,\frac{W}{m \cdot K}$ [10]. By using a set temperature of $T_{\text{Set}} = 20°C$ the resulting thermal demand for 2019 is approximately 5781 kWh.

The annual time series of the considered year 2019 for the photovoltaic power generation, the thermal demand and the electrical demand are shown in Figure 5.

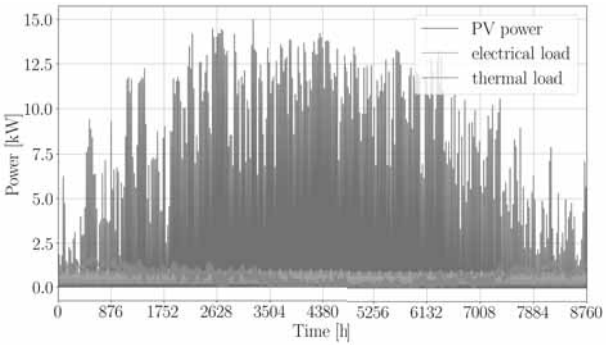


Figure 5: Time series for the PV-power, the electrical and the thermal load for 2019

3.2 System Configuration

Hydrogen Components

The dimensioning of the components of the hydrogen infrastructure in this simulation are based on the real-world configuration of the "Energiepark" at the University of Applied Sciences in Ulm. It contains an AEM-electrolyzer with a maximal power of 20 kW. The efficiency to convert electrical energy into hydrogen is assumed to be constant at 62.5 %. The fuel cell has a maximal electrical power of 8 kW with an assumed constant efficiency of 57 %. The facility features a low-storage (LPS) and a high-pressure storage (HPS), each consisting of 16 tanks with a volume of 50 liters per tank. To determine the usable energy content, the hydrogen mass is calculated using the respective storage pressures at a constant ambient temperature and the hydrogen gas density at those conditions. The lower heating value (LHV) of hydrogen is then used to convert this mass into chemical energy. With a storage pressure for the LPS of 35 bar, a storage pressure for the HPS of 300 bar and an ambient temperature of 20 °C this results in a maximal storage capacity of roughly 631 kWh.

Thermal Energy Storage

For the thermal energy storage a perfectly isolated vessel with a volume of 1000 l containing water is assumed. By using Equation 4 and constant temperatures of $T_{\text{supply}} = 45°C$ and $T_{\text{return}} = 30°C$ the maximal storage capacity of the TESS is roughly 17 kWh.

All for the simulation relevant parameters are summed up in Table 2.

Component	Parameter	Value
PV	$P_{\text{PV,peak}}$	15 kW
	E_{PV}	22874 kWh
BESS	$E_{\text{BESS,min}}$	2 kWh
	$E_{\text{BESS,max}}$	15 kWh
	$E_{\text{BESS,start}}$	0 kWh
	$P_{\text{BESS,ch,max}}$	5 kW
	$P_{\text{BESS,dis,max}}$	5 kW
	η_{BESS}	95%
HESS	$E_{\text{HESS,min}}$	5 kWh
	$E_{\text{HESS,max}}$	631 kWh
	$E_{\text{HESS,start}}$	0 kWh
Electrolyzer	$P_{\text{Ely,min}}$	2.5 kW
	$P_{\text{Ely,max}}$	20 kW
	η_{Ely}	62.5%
	$\eta_{\text{Ely,th}}$	37.5%
Fuel Cell	$P_{\text{FC,min}}$	0 kW
	$P_{\text{FC,max}}$	8 kW
	η_{FC}	57%
	$\eta_{\text{FC,th}}$	43%
TESS	$E_{\text{TESS,min}}$	0 kWh
	$E_{\text{TESS,max}}$	17 kWh
	$E_{\text{TESS,start}}$	0 kWh
Household	$E_{\text{demand,el}}$	4600 kWh
	$E_{\text{demand,th}}$	5781 kWh

Table 2: System parameters for all components used in the simulation

3.3 Considered Cases

Three simulation cases are defined and analyzed to evaluate the impact of the predictive control and the forecasting accuracy on the operation of the P-t-X system.

- **Case 1 – Rule-Based Control:** In the baseline case the simple rule-based control from subsection 2.4 is applied. Determination of the energy flows by predefined priorities are applied.
- **Case 2 – MPC with LSTM Forecasting:** The second scenario contains the case with applying the MPC in combination with the LSTM forecasts. The forecasts are done on the thermal load and the photovoltaic power generation. The prediction and optimization horizon is set to 24-hour. At each simulation step only the first optimized control parameters are used, following the receding horizon principle.
- **Case 3 – MPC with Perfect Forecast (Reference):** A third case is introduced to assess the theoretical potential of the MPC approach. Here the future PV generation and the thermal demand of the original time series are used as input for the optimization. This allows to measure the performance of the second proposed case.

4 Results and Discussion

4.1 Comparison of Strategies

The three in subsection 3.3 defined cases are compared in terms of self-consumption, self-sufficiency and grid dependency. The comparison is shown in Table 3.

Metric	Case 1	Case 2	Case 3
Self-consumption [%]	25.5	36.8	38.9
Self-sufficiency [%]	46.8	70.9	74.6
– Electrical [%]	96.2	81.7	96.1
– Thermal [%]	7.4	62.3	46.8
Grid usage [kWh]	5524.1	3020.4	2636.8
– Electrical [kWh]	173.0	841.9	179.7
– Thermal [kWh]	5351.1	2178.5	2457.1

Table 3: Comparison of performance indicators for the three simulated cases

The rule-based control strategy in Case 1 yields the lowest performance with only 25.5 % of the generated PV energy self-consumed and a self-sufficiency of 46.8 %. The total grid usage is about 5524.1 kWh with the main contribution of the thermal grid.

The in Case 2 introduced combination of MPC and the LSTM-based predictions of thermal load and PV production significantly improves the performance. The self-consumption-ratio is increased to 36.8 % and the self-sufficiency to 70.9 %. This demonstrates that the active use of available resources is beneficial for the autarky. Compared to Case 1, the total grid usage is reduced by over 45 %.

As expected, Case 3 does achieve the best performance overall. The use of the original time series enables the optimizer to find a better solution for the operation of the plant. A self-consumption ratio of 38.9 % and a self-sufficiency ratio of 74.6 % are achieved. A drop of the total grid usage to 2636.8 kWh can be seen. Highlighing the theoretical potential of the predictive control with more accurate forecasts. Furthermore, the difference between Case 2 and Case 3 highlights the influence of the forecast accuracy.

4.2 Utilization Profiles of Hydrogen System Components

In Figure 6 the utilization profile of the electrolyzer and the fuel cell is shown exemplary for Case 2. Figure 7 does show the H2 storage level for the said case. For

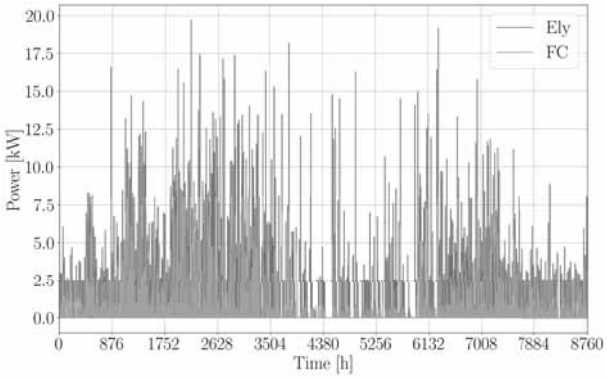


Figure 6: Electrolyzer / fuel cell utilization for Case 2

Figure 6 it is noticeable, that the electrolyzer and the fuel cell are operated frequently. The electrolyzer is utilized to store excess energy into hydrogen and make use of the waste heat. The fuel cell on the other hand is then used for the electrical or thermal demand not covered by PV or storage systems or to store energy in those storage systems. From Figure 7 it can be drawn, that the hydrogen storage level remains well below the maximum capacity for most of the year. It follows the operation of the electrolyzer and fuel cell, indicating a well-balanced charging and discharging pattern.

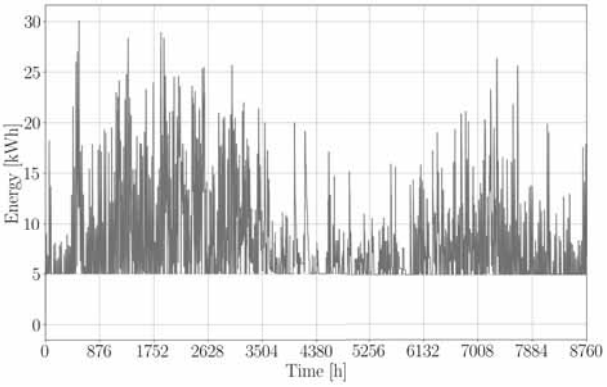


Figure 7: H2 storage level for Case 2

Table 4 shows a comparison of the full load hours. For Case 1 it can be noticed, that both of the components are rarely used. The electrolyzer only has 55 full load hours, and the fuel cell just 6.3 hours. In contrast to this, Case 2 and Case 3 show a significant increase in full load hours. The electrolyzer reaches 341.6 and 323.9 hours, respectively, while the fuel cell achieves 303.8 and 288.0 hours.

Component	Case 1	Case 2	Case 3
Electrolyzer [h]	55.0	341.6	323.9
Fuel Cell [h]	6.3	303.8	288.0

Table 4: Full load hours of the electrolyzer and the fuel cell for each simulated case

It can be seen that scheduling, enabled by the MPC approach, improves the utilization in general. Active usage of storage capacities and shifting does improve the utilization of waste heat. The higher values in full load hours of Case 2 are probably a result of inaccurate predictions of the LST models.

4.3 Performance of the LSTM-forecast

To evaluate the performance of the LSTM models both, qualitative and quantitative indicators are used. Figure 8 and Figure 9 show example predictions for photovoltaic power generation and thermal load, respectively. In case of the photovoltaik production it can be noticed that the bell-shaped profile and the timing of the solar production is captured well. However, a rapid change in PV production from one day to the other is not captured well. This might be caused by limited input features. For the thermal load prediction it can be noticed, that

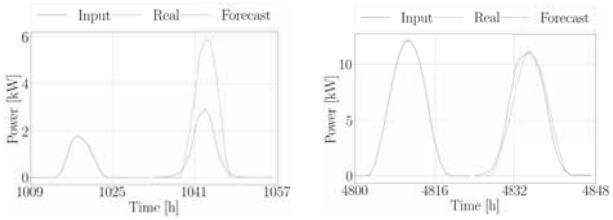


Figure 8: Example prediction for the photovoltaik power generation

the LSTM model does capture the general shape of the curve. But the model does struggle to predict abrupt changes in the time series. This is likely due to limited input features such as ambient conditions.

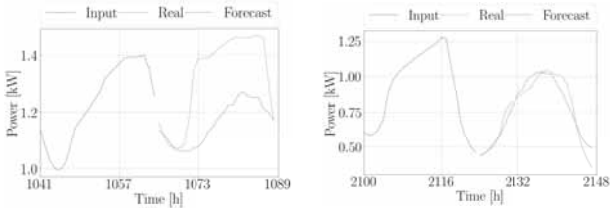


Figure 9: Example prediction for the thermal demand

The relative standard deviation of the prediction error is shown in Figure 10 to further quantify the forecast uncertainty. For the PV generation it can be noticed, that the relative error increases rapidly within the first 6 hours and then stabilizes at around 40 %. For the thermal load prediction, the relative error remains lower but increases more steadily. It reaches 27 % at the 24-hour mark. Overall these results indicate that temporal trends are captured well by LSTM models but with longer prediction horizons their accuracy decreases.

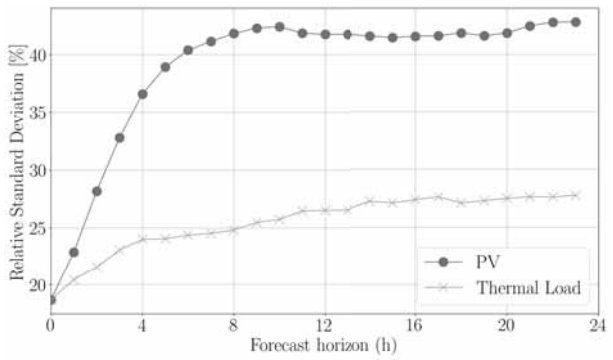


Figure 10: Relative standard deviation of the prediction error

5 Conclusion and Outlook

In this work a simulation-based approach to evaluate the operation of a small-scale sector-coupled P-t-X system is presented. The model includes electrical and thermal energy flows and is implemented in Python. A simple rule-based operation strategy is compared to a more advanced approach that incorporates model predictive control. For the model predictive control forecasts are done using a univariate LSTM neural network and the operation of the P-t-X facility is optimized with MILP. Even with imperfect forecasts, the model predictive approach improves the performance in terms of self-consumption and self-sufficiency.

However, the model proposed here is a strongly simplified representation of a real system. Future work could include a more detailled modeling of such a facility. This could include transient effects, startup behaviors of components like electrolyzer or fuel cell and a more detailed thermal modeling of components. On the forecasting side, prediction accuracy could be improved by using multivariate LSTM neural networks or even convolutional LSTM networks to further improve the proposed methodology.

References

[1] Gregor, Liselotte. Hydrogen as an energy carrier for a climate-neutral economy. *EU hydrogen policy*. 2021;.

[2] Kalchschmid V, Erhart V, Angerer K, Roth S, Hohmann A. Decentral Production of Green Hydrogen for Energy Systems: An Economically and Environmentally Viable Solution for Surplus Self-Generated Energy in Manufacturing Companies. *Sustainability*. 2023;15(4). URL https://www.mdpi.com/2071-1050/15/4/2994

[3] Dominic A, Giliomee J, Zenner T, Winter M, Schullerus G, Booysen M. Green Hydrogen for Future Energy Demand in Germany. In: *2024 IEEE 8th Energy Conference (ENERGYCON)*. 2024; pp. 1–6.

[4] Sterner M, Stadler I. *Energiespeicher - Bedarf, Technologien, Integration*. 2017. URL https://doi.org/10.1007/978-3-662-48893-5

[5] Huang C, Zong Y, You S, Træholt C. Economic model predictive control for multi-energy system considering hydrogen-thermal-electric dynamics and waste heat recovery of MW-level alkaline electrolyzer. *Energy Conversion and Management*. 2022;265:115697. URL https://www.sciencedirect.com/science/article/pii/S0196890422004939

[6] Kreuzer D, Munz M, Schlüter S. Short-term temperature forecasts using a convolutional neural network — An application to different weather stations in Germany. *Machine Learning with Applications*. 2020;2:100007. URL https://www.sciencedirect.com/science/article/pii/S2666827020300074

[7] Pfenninger S, Staffell I. Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data. *Energy*. 2016;114:1251–1265. URL https://www.sciencedirect.com/science/article/pii/S0360544216311744

[8] Staffell I, Pfenninger S. Using bias-corrected reanalysis to simulate current and future wind power output. *Energy*. 2016;114:1224–1239. URL https://www.sciencedirect.com/science/article/pii/S0360544216311811

[9] Stephan P, Kabelac S, Kind M, Mewes D, Schaber K, Wetzel T. *VDI-Wärmeatlas*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2019.

[10] Schmidt P, Windhausen S. *Lohmeyer Praktische Bauphysik*. Wiesbaden: Springer Fachmedien Wiesbaden. 2024.

[11] Drgoňa J, Arroyo J, Cupeiro Figueroa I, Blum D, Arendt K, Kim D, Ollé EP, Oravec J, Wetter M, Vrabie DL, Helsen L. All you need to know about model predictive control for buildings. *Annual Reviews in Control*. 2020;50:190–232. URL https://www.sciencedirect.com/science/article/pii/S1367578820300584

Modellgestützte Analyse des Betriebsverhaltens eines Diesel-Stromerzeugers im Inselbetrieb mit dynamischer Netzlast

Daniel Jörss[1], Maximilian Ringel[1], Bert Buchholz[2], Christian Fink[1*]

[1]Department of Mechanical / Process and Environmental Engineering, Wismar University of Applied Sciences, Philipp-Müller-Str. 14, 23966 Wismar, Germany
[*]*Christian.Fink@hs wismar.de*
[2]Chair of Piston Machines and Internal Combustion Engines, Rostock University, Albert-Einstein-Straße 2, 18059 Rostock, Germany

Abstract. In dem vorliegenden Bericht wird ein Simulationsmodell eines Stromerzeugers präsentiert, welches aus einem Verbrennungsmotor, einem Synchrongenerator und elektrischen Lasten besteht. Das Verbrennungsmotormodell basiert im wesentlichen auf thermodynamischen und mechanischen Modellen, die den Prozess innerhalb des Motors beschreiben. Für das Verbrennungsmodell wird ein phänomenologischer Modellansatz verwendet, bei dem der Einspritzverlauf aus einer gekoppelten Hydrauliksimulation als Eingangsgröße für das Verbrennungsmodell dient. Der Synchrongenerator wird mit Hilfe eines PU-basierten Simscape^{TM}-Modells beschrieben. Durch die Kopplung der beiden Modelle wird die Wechselwirkung zwischen dem Verbrennungsmotor und dem Synchrongenerator simuliert, um das Gesamtverhalten des Stromerzeugers unter verschiedenen Betriebsbedingungen zu analysieren. Das Gesamtmodell ermöglicht eine sehr gute Vorhersage des Verhaltens des Stromerzeugers und dient als Grundlage für die Optimierung der Betriebsstrategie und der Systemkonfiguration. Zur Validierung des Simulationsmodells wurden Messungen an einem Stromerzeuger mit variablen Lastaufschaltungen durchgeführt. Die Simulationsergebnisse weisen eine hohe Übereinstimmung mit den Messergebnissen im dynamischen Betrieb auf.

Einleitung

Im Zuge der Energiewende erhält die dezentrale Stromversorgung eine signifikant zunehmende Bedeutung. Dies betrifft auch Blockheizkraftwerke, die simultan Wärme und Strom generieren und daher eine hohe Effizienz aufweisen. Der Kern dieser Anlagen besteht in der Regel aus Verbrennungsmotoren, die bei einer zukünftig größeren Verfügbarkeit von Wasserstoff CO_2-neutral betrieben werden können. Der Einsatz von Stromerzeugern in Inselnetzen bei dynamischen Netzlasten stellt hohe Anforderungen an das Betriebsverhalten des antreibenden Verbrennungsmotors. Bei der herkömmlichen Bauweise, bei der der Verbrennungsmotor direkt mit einem Synchrongenerator verbunden ist, können Lastschwankungen zu Drehzahlschwankungen der Anlage und somit zu einer Schwankung der Netzfrequenz führen. Aufgrund der sensiblen Reaktion vieler elektrischer Verbraucher auf Netzfrequenzschwankungen ist es erforderlich, diese in engen Grenzen zu halten. Hersteller vor allem größerer Stromerzeuger sind daher bereits in einer frühen Auslegungsphase aufgefordert, die Auswirkungen möglicher Lastszenarien auf die durch das Aggregat bereitgestellte Netzfrequenz zu quantifizieren. Vor diesem Hintergrund wurde ein Simulationsmodell entwickelt, welches auf Grundlage physikalischer Zusammenhänge eine Berechnung des Betriebsverhaltens eines Verbrennungsmotors in Kombination mit einem Generator sowie eines variablen Netzlastszenarios zulässt. Das Modell basiert auf einer drehwinkelaufgelösten Prozessrechnung, um eine präzisere Aussage über das Systemverhalten zu erhalten und somit die Anlagenkonfiguration hinsichtlich einer Vielzahl von Parametern zu optimieren.

In dem vorliegenden Bericht wird das Modell eines Stromerzeugers beschrieben und die erlangten Simulationsergebnisse mit den Messergebnissen des Prüfstands verglichen.

1 Verbrennungsmotormodell

Für die Simulationsumgebung des Verbrennungsmotors wurde ein Berechnungsansatz auf Grundlage etablierter Teilmodelle gewählt. In diesem 0D-Modell wird der Brennraum zu jedem Zeitpunkt als ideal durchmischtes System betrachtet und die Zustandsgrößen Druck sowie Temperatur als Funktionen der Zeit bzw. des Kurbelwinkels dargestellt.

Über die Energiebilanz nach dem 1. Hauptsatz der Thermodynamik lässt sich die Änderung der inneren Energie dU im Brennraum beschreiben [1].

$$\frac{dU}{dt} = -p\frac{dV}{dt} + \frac{dQ_B}{dt} - \frac{dQ_W}{dt} + h_E\frac{dm_E}{dt} - h_A\frac{dm_A}{dt} \quad (1)$$

Der erste Term in der Gleichung beschreibt die am Kolben verrichtete Arbeit in Form der Volumenänderungsarbeit $-p\frac{dV}{dt}$. Des Weiteren beschreiben $\frac{dQ_B}{dt}$ die Wärmefreisetzung durch die Verbrennung, $\frac{dQ_W}{dt}$ den Wandwärmestrom, $h_E\frac{dm_E}{dt}$ den Enthalpiestrom über das Einlassventil sowie $h_A\frac{dm_A}{dt}$ den Enthalpiestrom über das Auslassventil [2].

Um die Massen- und Energiebilanz lösen zu können, wurden Modelle für die Wärmefreisetzung durch die Verbrennung (Brennverlauf), die Wärmeübertragung zwischen dem Gas im Brennraum und den Brennraumwänden, den Ladungswechsel sowie die Änderung der inneren Energie im Brennraum implementiert. Die Berechnung der inneren Energie im Brennraum erfolgt unter Anwendung der kalorischen Zustandsgleichung für ideale Gase.

$$du = c_v \cdot dT \quad (2)$$

Für die Berechnung der spezifischen Wärmekapazität c_v wird ein Komponentenmodell nach Grill [3] verwendet. Dabei wird angenommen, dass das Rauchgas ein Gemisch verschiedener Spezies idealer Gase ist. Grundvoraussetzung ist eine vorherige Berechnung der Gleichgewichtszusammensetzung des Rauchgases in Abhängigkeit von Temperatur, Luftverhältnis und Druck. Die aus diesem Ansatz berechnete spezifische innere Energie lässt sich die spezifischen Wärmekapazität c_v nach Gleichung 3 berechnen [3].

$$c_v = \left(\frac{\partial u}{\partial T}\right)_p + \left(\frac{\partial u}{\partial p}\right)_T \cdot \frac{p}{T} \quad (3)$$

1.1 Brennverlaufsmodell

Ein elementarer Bestandstandteil zur Modellierung des Motorinnenprozesses ist das Brennverlaufsmodell. Dieses Modell beschreibt den zeitlichen Verlauf der Wärmefreisetzung der im Kraftstoff gebundenen chemischen Energie (freiwerdende Reaktionsenthalpie aus dem Verbrennungsprozess). Zur Darstellung des Brennverlaufs können verschiedene Modellansätze, wie phänomenologische- oder sogenannte Ersatzbrennverlaufsmodelle, genutzt werden. In dem Simulationsmodell wird ein kombiniertes phänomenologisches Verbrennungsmodell nach Barba [4] und Chmela [5] verwendet. Die Eingangsgröße für dieses Modell ist der Einspritzverlauf, der mit einem implementierten Injektormodell berechnet wird [6].

Die Berechnung des Brennverlaufs erfolgt auf Basis zweier Teilmodelle, einerseits für die Voreinspritzung, andererseits für die Haupteinspritzung. Bei dem Verbrennungsmodell für die Voreinspritzung wird eine kugelkörmige Ausbreitung der Flammenfront von einem einzigen Zündort angenommen. Der Kugelradius ergibt sich dabei aus der turbulenten Flammengeschwindigkeit s_{turb}. Die Flammenoberfläche lässt sich mit Gleichung 4 beschreiben [7].

$$A_{Flamme} = 4 \cdot \pi \cdot r_{Flamme}^2 \quad (4)$$

Unter der Annahme eines homogenen Kraftstoff-Luft-Gemisches lässt sich die umgesetzte Rate bei einer turbulenten Verbrennung von einem Einzelzündort wie folgt beschreiben.

$$\dot{m}_{b,1} = k_1 \cdot \rho_{uv} \cdot s_{turb} \cdot A_{Flamme} \quad (5)$$

Es ist jedoch davon auszugehen, dass die Zündung nicht nur an einem lokalen Punkt des Brennraums erfolgt, insbesondere wenn die Verbrennung der Voreinspritzung bereits fortgeschritten ist. Wie in [4,7,8] wurde daher ein zweiter Ansatz zur Berechnung des Brennverlaufs implementiert.

$$\dot{m}_{b,2} = k_2 \cdot \frac{1}{\Lambda_{GW}^2} \cdot \frac{s_{turb}}{r_{GW}} \cdot m_{B,uv} \quad (6)$$

Der gesamte verbrannte Kraftstoff ergibt sich aus der Überlagerung der beiden Gleichungen 5 und 6 [8].

$$\dot{m}_{b,v} = \frac{\dot{m}_{b,1} \cdot \dot{m}_{b,2}}{\dot{m}_{b,1} + \dot{m}_{b,2}} \quad (7)$$

Das Modell für die Haupteinspritzung wird ebenfalls mittels zweier Modellansätze beschrieben: Einerseits für die vorgemischte Verbrennung und andererseits für die Diffusionsverbrennung. Im ersten Schnitt erfolgt die Berechnung der Strahleindringtiefe S des Sprays mithilfe eines Ansatzes nach Najar [9].

$$S_{\rho_g} = 265,7 \cdot \left(\frac{m_f}{\sqrt{Q_{hyd}}} \right)^{0,445} \cdot \rho_g^{-0,387} \tag{8}$$

In Abbildung 1 ist die schematische Darstellung für verschiedene Lambda-Bereiche des Einspritzstrahls im Verbrennungsmodell dargestellt. Der düsennahe Bereich stellt einen sehr kraftstoffreichen Bereich dar, in dem wenig Luft vorhanden ist. Je weiter das Spray in den Brennraum eindringt, desto größer wird das vorliegende Luft-Kraftstoff-Verhältnis. Über die Lambda-

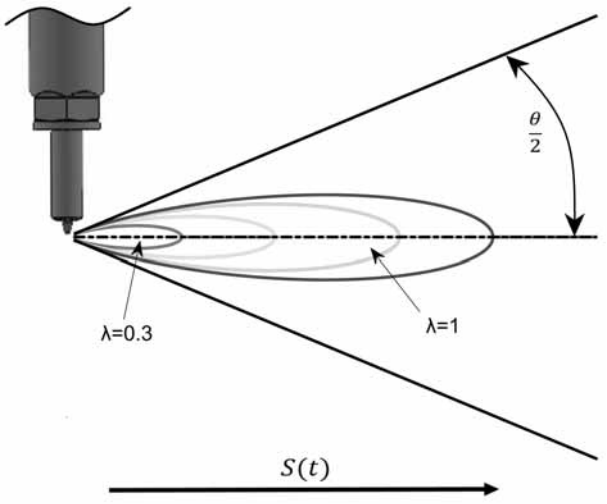


Abbildung 1: Schematische Darstellung der Lambda-Bereiche im Einspritzstrahl

Bereiche lässt sich ein Gemischvolumen V_{Gem} berechnen, in dem der verdampfte Kraftstoff und die durch das lokale Luftverhältnis definierte Luftmenge vorhanden sind [5].

$$V_{Gem} = m_B \left(\frac{1}{\rho_{K,d}} + \frac{\lambda \cdot L_{min}}{\rho_L} \right) \tag{9}$$

Im Modell erfolgt eine Unterscheidung in zwei Kraftstoffpools. Während die im Laufe des Zündverzugs eingespritzte Kraftstoffmenge der vorgemischten Verbrennung zugeordnet wird, wird die ab dem Brennbeginn eingespritzte Kraftstoffmenge der Diffusionsverbrennung zugerechnet. Der Zündverzug teilt sich in einen physikalischen und einen chemischen Teil auf. [4].

$$\tau_{ZV} = \tau_{ZV,phy} + \tau_{ZV,chem} \tag{10}$$

Die Anteile lassen sich nach den Gleichungen 11 und 12 berechnen.

$$\tau_{ZV,phy} = c_0 \cdot {u_{Tr0}}^{-1.68} \cdot {d_{D.eff}}^{0.88} \tag{11}$$

$$\tau_{ZV,chem} = c_1 \cdot \left(\frac{p_z}{p_0} \right)^{c_2} \cdot {\lambda_{Zn}}^{c_3} \cdot e^{\frac{T_A}{T_G}} \tag{12}$$

Um die zeitlichen Veränderungen von Druck und Temperatur während des Zündverzuges zu berücksichtigen, wird das folgende Integral verwendet.

$$1 = \int_{t_{EB}}^{t_{VB}} \frac{1}{\tau_{ZV}} dt \tag{13}$$

Das Zündereignis erfolgt, wenn das Integral den Wert 1 erreicht hat. Ab dem Zeitpunkt des Brennbeginns lässt sich der Brennverlauf für die vorgemischte Verbrennung nach Gleichung 14 berechnen [5].

$$\frac{dQ_{b,Pre}}{dt} = c_{pre} \cdot \frac{{m^2}_{B,Pre,uv}}{V_{Gem}} \cdot H_U \cdot (t - t_{BB})^2 \tag{14}$$

Die Modellierung der Diffusionsverbrennung erfolgt unter der Verwendung eines Frequenzansatzes nach Barba [4] und wird über die Gleichung 15 berechnet.

$$\frac{dQ_{B.Diff}}{dt} = \frac{u'}{l_{Diff}} \cdot m_{B.uv} \cdot H_U \tag{15}$$

Unter der Verwendung einer charakteristischen Mischungslänge l_{Diff} sowie der turbulenten Mischungsgeschwindigkeit u' ergibt sich folgende Gleichung.

$$\frac{dQ_{Diff}}{dt} = \frac{\sqrt{c_G \cdot {c_m}^2 + c_{Kin} \cdot k}}{\sqrt[3]{\frac{V_{Zyl}}{\lambda \cdot Anz_D}}} \cdot m_{B.uv} \cdot H_U \tag{16}$$

Für die Berechnung der spezifischen turbulenten kinetischen Energie k wird folgende Gleichung herangezogen, die auf einem k-ε-Modell basiert [10].

$$\frac{dk}{dt} = -\frac{2}{3} \cdot \frac{k}{V_{Zyl}} \cdot \frac{dV_{Zyl}}{dt} - \varepsilon_{Diss} \cdot \frac{k^{1,5}}{l} + \left(\varepsilon_q \cdot \frac{k_q^{1,5}}{l} \right)_{\phi > ZOT} + \varepsilon_E \cdot \frac{dk_E}{dt} + \varepsilon_D \cdot \frac{c_m^3}{l} \tag{17}$$

Es wird angenommen, dass die vorgemischte und die diffusionskontrollierte Verbrennung gleichzeitig beginnen. In diesem Fall wird eine Zeitverzögerungsfunktion $F_{Pre-Diff}$ verwendet, um den Brennverlauf während der diffusionskontrollierten Verbrennung zu verschieben und die verzögerten chemischen Reaktionen zu berücksichtigen, die die diffusionskontrollierte Verbrennung unmittelbar nach dem Brennbeginn steuern [8].

$$F_{Pre-Diff} = \left(\frac{Q_{B.v.Pre}}{Q_{B.zu.Pre}} \right)^{e} \tag{18}$$

Unter Berücksichtigung dieser Verzögerungsfunktion lässt sich der gesamte Brennverlauf mit folgender Gleichung beschreiben.

$$\frac{dQ_{b,v}}{dt} = \frac{dQ_{b,Pre}}{dt} + F_{Pre-Diff} \cdot \frac{dQ_{B.Diff}}{dt} \tag{19}$$

In Abbildung 2 sind die einzeln Brennverlaufsfunktionen sowie deren Überlagerung exemplarisch dargestellt.

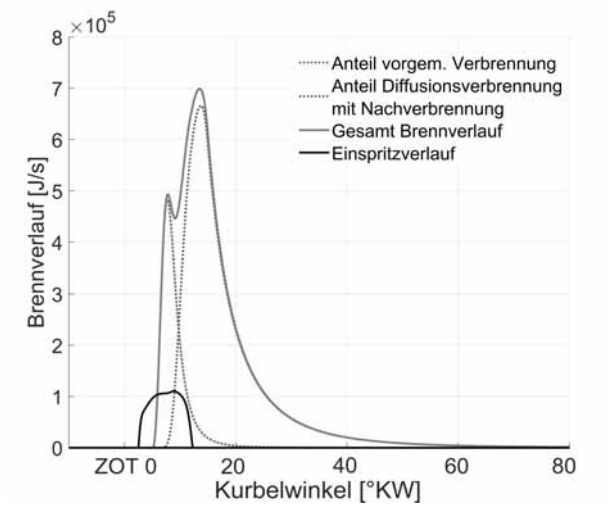


Abbildung 2: Modellierung Brennverlauf

1.2 Wandwärmemodell

Zu den komplexesten Modellen gehört die Beschreibung des Wärmeüberganges zwischen dem Arbeitsgas im Brennraum und den Brennraumwänden. Generell wird der Wandwärmestrom in der Arbeitsprozessrechnung nach dem Newton'schen Ansatz berechnet.

$$\dot{Q}_W = \alpha \cdot A(\varphi) \cdot (T_G - T_W) \tag{20}$$

In der Gleichung 20 ist A die wärmeübertragende Fläche, die sich aus Zylinderkopffläche, Kolbenbodenfläche und der momentanen Mantelfäche des Brennraumes je nach Kolbenposition zusammensetzt. Für die Berechnung des Wärmeübergangskoeffizienten α wird ein Modellansatz nach Bargende verwendet [11].

$$\alpha = 253{,}5 \cdot V_{Zyl}^{-0{,}073} \cdot T_m^{-0{,}477} \cdot p_z^{0{,}78} \cdot w^{0{,}78} \cdot \Delta \tag{21}$$

Der Term w steht für eine wärmeübergangsrelevante Geschwindigkeit, während der Verbrennungsterm Δ unter anderem die unterschiedlichen treibenden Temperaturdifferenzen zwischen verbranntem und unverbranntem sowie eine Durchbrennfunktion des Brennverlaufs X berücksichtigt. Die beiden Terme werden mit den Gleichungen 22 und 23 berechnet.

$$w = \frac{1}{2} \cdot \sqrt{\frac{8 \cdot k}{3} + c_K^2} \tag{22}$$

$$\Delta = \left[X \frac{T_v}{T_G} \frac{T_v - T_W}{T_G - T_W} + (1 - X) \frac{T_{uv}}{T_G} \frac{T_{uv} - T_W}{T_G - T_W} \right]^2 \tag{23}$$

In Gleichung 22 wird die spezifische kinetische Energie k ebenfalls über ein k-ε-Modell berechnet. Für die Berechnung der Temperaturdifferenzen erfolgt eine 1-zonige Modellannahme in dem keine explizite Unterteilung einer verbrannten und unverbrannten Zone durchgeführt wird. Nach Bargende [11] wird die Temperatur im unverbrannten T_{uv} mithilfe der Polytropenbeziehung ab Brennbeginn (BB) berechnet.

$$T_{uv} = T_{G,BB} \cdot \left(\frac{p_Z}{p_{Z,BB}} \right)^{\left(\frac{n-1}{n}\right)} \tag{24}$$

Die Temperatur im Verbrannten T_v lässt sich wie folgt bestimmen.

$$T_v = \frac{1}{X} \cdot T_G + \frac{X - 1}{X} \cdot T_{uv} \tag{25}$$

1.3 Zwei-Zonenmodell

Mit dem Ziel einer Vorhersage der entstehenden Stickoxidemissionen (NO), muss zusätzlich zum Ein-Zonenmodell ein Zwei-Zonenmodell für den Brennraum eingeführt werden. In diesem Modell wird der Brennraum in einer unverbrannten und einer verbrannten Zone unterteilt, die durch eine infinitesimal dünne Flamme voneinander getrennt sind. Eine Zumischung aus der unverbrannten in die verbrannte Zone erfolgt

in diesem Modell über einen phänomenologischen Gleichungsansatz nach Kožuch [10].

Nach Kožuch wird die unverbrannte Zone mit der Gleichung 26 beschrieben.

$$\left(\frac{V_{uv}}{V_{Zyl}}\right)^{\frac{2}{3}} \cdot \frac{dQ_w}{dt} - p\frac{dV_{uv}}{dt} + \frac{dm_{uv}}{dt} \cdot (u_{uv} + R_{uv} \cdot T_{uv}) = m_{uv} \cdot \left[\frac{\partial u_{uv}}{\partial T}\frac{dT_{uv}}{dt} + \frac{\partial u_{uv}}{\partial p}\frac{dp}{dt}\right] + u_{uv} \cdot \frac{dm_{uv}}{dt} \quad (26)$$

Analog zur Energiebilanz für die unverbrannte Zone lässt sich die verbrannte Zone wie folgt nach Gleichung 27 formulieren.

$$\left(1 - \frac{V_{uv}}{V_{Zyl}}\right)^{\frac{2}{3}} \cdot \frac{dQ_w}{dt} + \frac{dQ_B}{dt} - p\frac{dV_v}{dt} - \frac{dm_{uv}}{dt} \cdot h_{uv} = m_v \cdot \left[\frac{\partial u_v}{\partial T}\frac{dT_v}{dt} + \frac{\partial u_v}{\partial p}\frac{dp}{dt} + \frac{\partial u_v}{\partial \lambda}\frac{d\lambda_v}{dt}\right] + u_v \cdot \frac{dm_v}{dt} \quad (27)$$

Zusätzlich findet eine Zumischung aus der unverbrannten Zone statt, welche an der Flamme vorbeigeführt und direkt in die verbrannte Zone eingebracht wird. Dies ermöglicht die Regulierung der Temperatur und des Luftverhältnisses in der verbrannten Zone. Diese Zumischfunktion g wird nach Gleichung 28 berechnet [10].

$$g = c_g \cdot \rho_{uv} \cdot u_{Turb} \cdot V_v^{\frac{2}{3}} \cdot Anz_D + c_{ga} \cdot \frac{dm_B}{dt} \quad (28)$$

Zusammen mit einem weiteren Teilmassenstrom ergibt sich ein Gesamtmassenstrom aus dem Unverbrannten ins Verbrannte.

$$\frac{dm_{uv}}{dt} = -\left[g + (1 + x_{R,st}) \cdot \lambda_F \cdot L_{st} \cdot \frac{1}{H_U} \cdot \frac{dQ_B}{dt}\right] \quad (29)$$

Abbildung 3 zeigt exemplarisch die berechneten Temperaturverläufe der unverbrannten und verbrannten Zone im Vergleich zu der Massenmitteltemperatur aus der Einzonenrechnung.

Über den berechneten Temperaturverlauf in der verbrannten Zone und den aus der Gleichgewichtsrechnung ermittelten Konzentration vorhandener Spezies lässt sich die NO-Konzentration über die Reaktionskinetik bestimmen. Grundlage dieser Rechnung ist der erweiterte Zeldovich-Mechanismus [12,13], der die NO-Bildung nach Gleichung 30 beschreibt.

$$\frac{d[NO]}{dt} = k_{1,v}[O][N_2] + k_{2,v}[N][O_2] + k_{3,v}[N][OH] - k_{1,r}[NO][N] - k_{2,r}[NO][O] - k_{3,r}[NO][H] \quad (30)$$

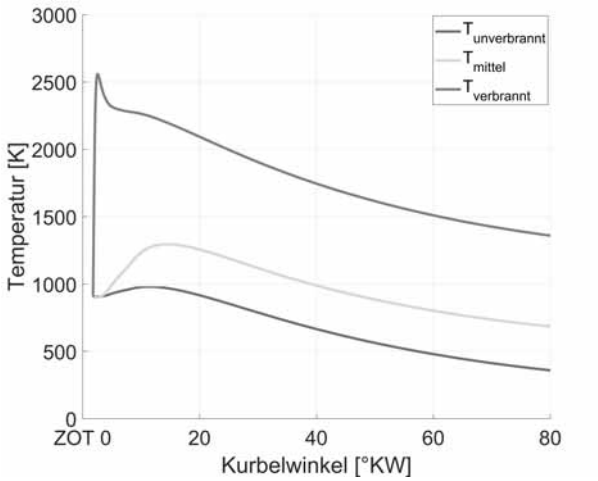


Abbildung 3: Temperaturverläufe Zweizonenmodell

2 Turboladermodell

Bei der Abgasturboaufladung wird die nach dem Arbeitstakt zur Verfügung stehende verbleibende Abgasenergie genutzt, um die Leistungsdichte des Motors zu erhöhen. Der Antrieb erfolgt durch den Abgasenthalpiestrom, der die Turbine radial durchströmt und das Turbinenrad mit der Welle in eine Drehbewegung versetzt. Dies überträgt sich auf das Verdichterrad, wodurch dem Verbrennungsmotor mehr Frischluft zur Verfügung gestellt werden kann. Betrachtet man die Leistungsbilanz für den statischen Fall zwischen Turbine und Verdichter sowie die Vernachlässigung der mechanischen Verluste, entsteht der Zusammenhang, dass die aus dem Abgasenthalpiestrom umgewandelte Turbinenleistung P_T auf der anderen Seite als Verdichterleistung P_V zur Verfügung gestellt wird.

$$P_T = P_V \quad (31)$$

Die Leistung des Verdichters und der Turbine ergibt sich jeweils aus dem Massenstrom $\dot{m}_i$, der spezifischen Enthalpiedifferenz $\Delta h_{i,is}$ und dem Isentropenwirkungs-

grad $\eta_{i,is}$ [14].

$$P_T = \dot{m}_T \cdot \Delta h_{T,is} \cdot \eta_{T,is} \tag{32}$$

$$P_V = \dot{m}_V \cdot \Delta h_{V,is} \cdot \frac{1}{\eta_{V,is}} \tag{33}$$

Der dynamische Betrieb des Abgasturboladers lässt sich über die Leistungsbilanz zwischen Verdichter und Turbine beschreiben. Aus der Leistungsdifferenz, dem Trägheitsmoment des Laufzeuges J_{ATL} und dem mechanischen Wirkungsgrad η_m resultiert eine Änderung der Winkelgeschwindigkeit ω_{ATL} bzw. Drehzahl des Turboladers. Durch Einbeziehung der Energieerhaltung für Drehbewegungen erhält man folgende Gleichung.

$$\frac{d\omega_{ATL}}{dt} = \frac{P_T \cdot \eta_m - P_V}{J_{ATL} \cdot \omega_{ATL}} \tag{34}$$

3 Synchrongenerator

Zur Simulation des Generators wird ein per Unit (PU) basiertes Synchrongenerator-Modell aus der Simscape^{TM}-Bibliothek herangezogen [15]. Das PU-Einheitensystem ist darauf ausgerichtet, alle relevanten Systemparameter auf Basiswerte zu beziehen, wodurch diese zu dimensionslosen Verhältniswerten werden. Die meisten Hersteller von Generatoren stellen die entsprechenden Parameter in diesem Einheitensystem zur Verfügung. Dadurch ist es möglich, das Simulationsmodell möglichst realitätsnah zu parametrieren.

Die Eingangsgröße für dieses Modell stellt der Drehzahlverlauf des Verbrennungsmotormodells dar. Als Ausgangsgröße dient das elektromagnetische Moment als Lastmoment für den Verbrennungsmotor.

4 Modellvalidierung

Zur Bewertung des transienten Lastverhaltens in der Simulation wurde das entwickelte Modell anhand von Messergebnissen am Prüfstand validiert. Abbildung 4 zeigt das transiente Lastverhalten zwischen Simulation und Experiment bei einer Lastaufschaltung von 2 kW auf 13 kW. Die Lastaufschaltung findet bei einer Simulationszeit von drei Sekunden statt. Dies hat zu Folge, dass die Drehzahl des Motors bzw. die Frequenz am Generator einbricht. Das Regelverhalten des Drehzahlreglers des Verbrennungsmotors bewirkt eine Anhebung der einspritzten Kraftstoffmasse, wodurch das innere Moment des Verbrennungsmotors ansteigt, um den erhöhten Leistungsbedarf zu kompensieren. Bei einer Simulationszeit von sieben Sekunden wird wieder die Sollfrequenz von 50 Hz erreicht und kann als gleichbleibend stabil betrachtet werden. Ausgehend vom Frequenzverlauf entspricht das Simulationsergebnis den Messwerten. Der Gradient des Drehmomentverlaufs der Messdaten weist hingegen einen nicht so starken Anstieg wie in der Simulation auf. Dennoch entspricht der gemessene Drehmomentverlauf in guter Näherung dem Verlauf der Simulation.

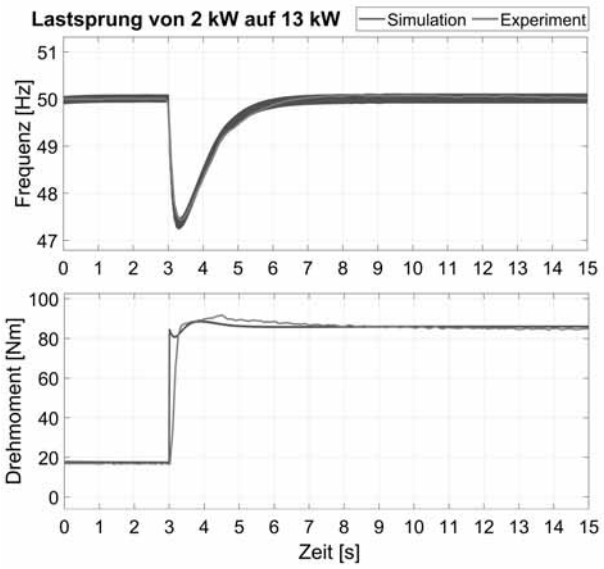


Abbildung 4: Berechneter und gemessener Frequenz- und Drehmomentverlauf während eines Lastsprunges

Zur weiteren Validierung der Modelle in diesem Lastszenario wurden die Simulationsergebnisse mit Messergebnissen des Dieselmotors am Prüfstand verglichen. In Abbildung 5 ist ein Vergleich der Zylinderdruckverläufe während der Hochdruckphase für die Lastpunkte 2 kW und 13 kW dargestellt. Darüber hinaus erfolgte die Ermittlung der dazugehörigen Brennverläufe mittels Druckverlaufsanalyse, um einen Vergleich mit den Simulationsergebnissen zu ermöglichen. Die Gegenüberstellung der ermittelten Zylinderdruck- und Brennverläufe zeigt eine sehr gute Übereinstimmung zwischen Simulation und Berechnung.

Abbildung 6 zeigt das dynamische Verhalten des Turboladers während des Lastsprunges mithilfe der Parameter Abgasdruck, Ladedruck und Turboladerdrehzahl. Die Anhebung der eingespritzten Kraftstoffmasse bei Lastaufschaltung führt zu einer Erhöhung

der Abgasenthalpie, wodurch sich der Druck im Abgaskrümmer erhöht und an der Turbine des Abgasturboladers mehr Arbeit verrichtet wird. Daraus resultiert eine steigende Drehzahl am Turbolader und bedingt durch den höheren Luftmassenstrom am Verdichter eine Anhebung des Ladedruckes im Saugrohr.

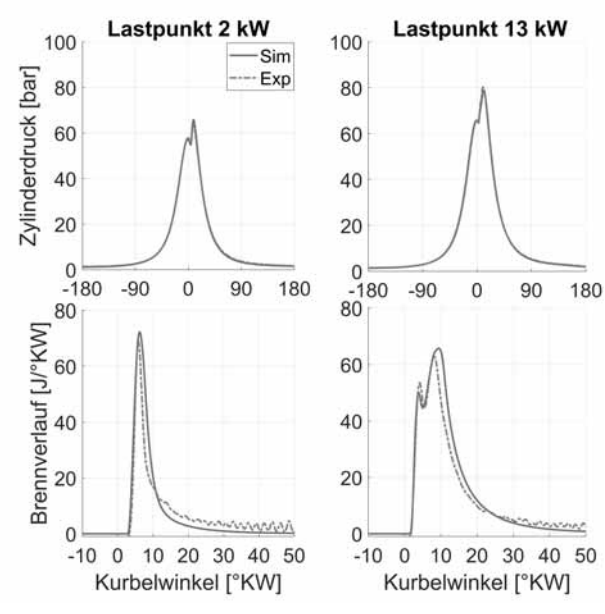


Abbildung 5: Vergleich der Zylinderdruck- und Brennverläufe vor und nach der Lastaufschaltung

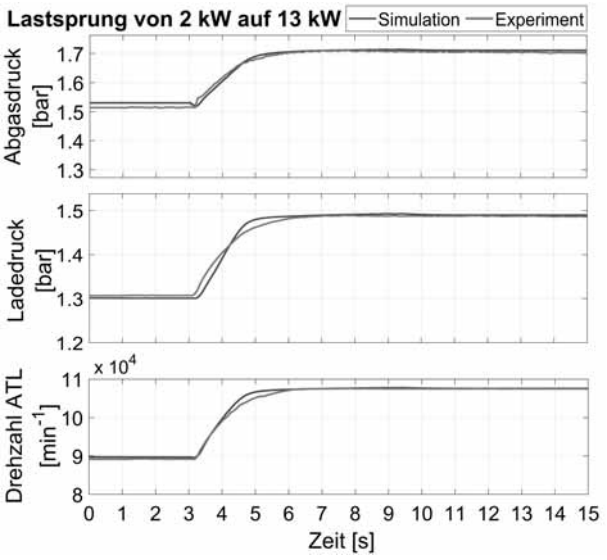


Abbildung 6: Exemplarisches Verhalten des Turboladers (ATL) im transienten Betrieb

Der Vergleich mit den Messdaten zeigt, dass die Simulationsverläufe des Abgasdruckes und der Turboladerdrehzahl eine hohe Übereinstimmung in der Verlaufscharakteristik aufweisen.

5 Zusammenfassung

Der in diesem Beitrag dargestellte Modellansatz ermöglicht die vollständige Simulation des Betriebsverhaltens von Stromerzeugern in Inselnetzen mit dynamischer Netzlast mittels zeitlich hoch aufgelöster Prozesssimulation. Im Gegensatz zu konventionellen Berechnungsansätzen erlaubt das hier vorgestellte Verbrennungsmodell aufgrund der Verwendung eines phänomenologischen Ansatzes auch die Berechnung von Brennverfahren mit Mehrfacheinspritzung. Zusätzlich bietet ein Zweizonenmodell auch die Möglichkeit enstehende Stickoxid-Emission in der Simulation zu berücksichtigen. Ein Vergleich der Simulationsergebnisse zeigt sehr gute Übereinstimmungen mit Messergebnissen eines Stromerzeugers am Prüfstand, was eine Anwendbarkeit dieses Simulationsansatzes unterstützt. Ein Ziel zukünftiger Arbeiten betrifft die Untersuchung des Modells bei der Anwendung verschiedener Kraftstoffeigenschaften und deren Auswirkungen auf das Betriebsverhalten des Stromerzeugers in dynamischen Betriebsszenarien.

Danksagung

Die hier vorgestellten Ergebnisse wurden im Rahmen des öffentlich geförderten Projekts SIDYN (FKZ:13FH043PX8) und des Promotionsprogramms der Hochschule Wismar erzielt. Die Autoren danken dem BMBF und der Hochschule Wismar für die gewährte finanzielle Unterstützung.

Nomenklatur

ε_{Diss} - Konstante für die Dissipation [-]
ε_{D} - Konstante für den Drall [-]
ε_{E} - Konstante für die Einspritzung [-]
ε_{q} - Konstante für die Quetschströmung [-]
λ - Verbrennungsluftverhältnis [-]
Λ_{GW} - Verbrennungsluftverhältnis in der Gemischwolke [-]
λ_{Zn} - Lokal vorherrschendes Luftverhältnis [-]
ρ_{g} - Gasdichte [kg/m^3]
$\rho_{K,d}$ - Kraftstoffdichte gasförmig [kg/m^3]

τ_{ZV} - Zündverzug [s]
Anz_D - Einspritzdüsen-Lochanzahl [-]
c - Allgemeine Konstante [-]
c_K - Kolbengeschwindigkeit [m/s]
c_m - Mittlere Kolbengeschwindigkeit [m/s]
c_{pre} - Abstimmparameter [$m^3/(kg \cdot s^3)$]
$d_{D.eff}$ - Effektive Düsendurchmesser [mm]
e - Exponent [-]
h - Spezifische Enthalpie [J/kg]
H_U - Unterer Heizwert [J/kg]
k - Kinetische Energiedichte [m^2/s^2]
$k_{1,2}$ - Abstimmparameter [-]
$k_{i,v,r}$ - Geschwindigkeitskonstante [$m^3/(mol \cdot s)$]
l_{Diff} - Mischungslänge [m]
l_l - Längenmass [m]
L_{st} - Mindestluftbedarf [kg/kg]
m_B - Kraftstoffmasse [kg]
m_f - Kumulierte Kraftstoffmasse [kg]
n - Polytropenexponent [-]
p_0 - Umgebungsdruck [bar]
p_z - Zylinderdruck [bar]
Q_B - Freigesetzte Energie (bei der Verbrennung) [J]
Q_{hyd} - Hydraulischer Spritzlochdurchfluss [cm^3/30s @100 bar]
R - Gaskonstante [J/(kg·K)]
r_{GW} - Radius Gemischwolke [m]
T_A - Aktivierungstemperatur [K]
t_{EB} - Zeit Einspritzbeginn [s]
$T_{G,m}$ - Mittlere Gastemperatur im Zylinder [K]
t_{VB} - Zeit Verbrennungsbeginn [s]
T_W - Wandtemperatur [K]
u - Spezifische innere Energie [J/kg]
u_{Tr0} - Ausgangstropfengeschwindigkeit [m/s]
u_{Turb} - Turbulenzgeschwindigkeit [m/s]
V_{Zyl} - Zylindervolumen [m^3]
$x_{R,st}$ - Stöchiometrischer Restgasanteil [-]

References

[1] Pischinger, R., Klell, M., Sams, T. *Thermodynamik der Verbrennungskraftmaschine* 3.Auflage, Wien, SpringerWienNewYork, 2009

[2] Tschöke, H., Mollenhauer, K., Maier, R. *Handbuch Dieselmotoren* 4.Auflage, Wiesbaden, Springer Vieweg, 2018

[3] Grill, M. *Objektorientierte Prozessrechnung von Verbrennungsmotoren*, Stuttgart, Universität Stuttgart, Diss., 2006

[4] Barba, C. *Erarbeitung von Verbrennungskennwerten aus Indizierdaten zur verbesserten Prognose und rechnerischen Simulation des Verbrennungsablaufes bei Pkw-DE-Dieselmotoren mit Common-Rail Einspritzung*, Zürich, ETH Zürich, Diss., 2001

[5] Pirker, G., Chmela, F., Wimmer, A. *ROHR Simulation for DI Diesel Engines Based on Sequential Combustion Mechanisms*, SAE Paper 2006-01-0654, 2006

[6] Jörss, D., Ringel, M., Buchholz, B., Fink, C. *Gekoppelte Simulation des Einspritz- und Verbrennungsvorgangs eines Industrie-Dieselmotors*, ASIM Workshop 2023 STS / GMMS / EDU, Magdeburg, 2023, DOI: 10.11128/arep21

[7] Rether, D., Grill, M., Schmid, A., Bargende, M. *Quasi-Dimensional Modeling of CI-Combustion with Multiple Pilot- and Post Injections*, SAE Paper 2010-01-0150, 2010

[8] Warth, M. *Comparative investigation of mathematical methods for modeling and optimization of Common-Rail DI diesel engines*, Zürich, ETH Zürich, Diss., 2005

[9] Najar, I., Fink, C., Harndorf, H., Pinker, F., Buchholz, B. *"Anwendungsorientierte Modelle zur Berechnung von Diesel-Sprays" in 10. Tagung Diesel- und Benzindirekteinspritzung 2016*, Wiesbaden, Springer Vieweg, 2017

[10] Kožuch, P. *Ein phänomenologisches Modell zur kombinierten Stickoxid- und Rußberechnung bei direkteinspritzenden Dieselmotoren*, Stuttgart, Universität Stuttgart, Diss., 2004

[11] Bargende, M. *Ein Gleichungsansatz zur Berechnung der instationären Wandwärmeverluste im Hochdruckteil von Ottomotoren*, Darmstadt, Technische Hochschule Darmstadt, Diss., 1991

[12] Zeldovich, Y.A. *The Oxidation of Nitrogen in Combustion and Explosions*, Acta Physicochimica, USSR, Vol.21 (577-628), 1946

[13] Lavoie, G. A., Heywood, J. B., Keck, J. C. *Experimental and theoretical study of nitric oxide formation in internal combustion engines*, Combustion science and technology 1970, Vol.1 (313–326)

[14] Pucher, H., Zinner, K. *Aufladung von Verbrennungsmotoren* 4.Auflage, Berlin Heidelberg, Springer Vieweg, 2012

[15] The MathWorks, Inc. *https://de.mathworks.com/help/sps/powersys/ref/synchronousmachinepufundamental.html*, letzter Zugriff: 14.01.2025

Design and Implementation of an Accessible Real-Time Simulation Framework Using Low-Cost Hardware and Open-Source Software

Mohd Azam Naqvi[1*], Siddhartha Gupta[1], Umut Durak[2], Sven Hartmann[1]

[1]Institute of Informatics, Technische Universität Clausthal, 38678 Clausthal-Zellerfeld, Germany;
[*]*mohd.azam.naqvi@tu-clausthal.de*

[2] German Aerospace Center (DLR), Institute of Flight Systems, 38108 Braunschweig, Germany

Abstract. Simulation plays an important role in modern engineering and research by providing an efficient method for developing and validating complex systems while minimizing errors. In real-time simulation, the simulation time is synchronized with the wall clock, ensuring that input providers and output consumers operating under time constraints adhere to the same timing requirements. However, commercial real-time simulation tools require extensive investment for software licenses and specialized hardware, creating a significant barrier to widespread adoption. The lack of affordable solutions further exacerbates this issue as it limits access for students, independent researchers and resource constrained organizations thus hindering innovation in fields such as control systems, robotics and automation. This paper explores cost-effective alternatives for real-time simulation by investigating the use of open-source software and low-cost hardware. A Raspberry Pi 4B running NixOS RT is evaluated as a platform to assess its feasibility for real-time applications. Our results demonstrate that this configuration effectively supports real-time tasks by enabling the loading and execution of Functional Mock-up Units (FMUs) while achieving a minimum system latency. This provides a viable solution for environments with limited resources. This framework represents a significant step toward democratizing access to real-time simulation, enabling cost-effective experimentation, and fostering innovation in engineering and research.

Introduction

Accurate prediction and analysis of complex system behavior are the foundation of engineering innovation and scientific progress. Simulation is an important tool in research and engineering, allowing the modeling analysis and optimization of complex systems without the risks and costs associated with physical prototyping[1]. By enabling virtual testing under controlled conditions, simulation helps engineers to refine designs, evaluate system performance, and anticipate potential failures before real-world implementation[2].

Among vast simulation methodologies, real-time simulation is essential for applications demanding precise temporal fidelity. Real-time simulation synchronizes execution with the real-world clock, ensuring that input signals and responses occur within strict timing constraints. This capability is essential for aerospace, power systems, automotive engineering, and industrial automation applications, where immediate system responses and precise control are necessary for operational safety and efficiency[3].

The development of real-time simulation dates to the mid-20th century when the first large and analog-scale digital computers were implemented for the military and aerospace sectors. The first of such systems was pioneering, but they were expensive and had limited access, allowing their applications only in specialized domains. The sharp advent of microprocessor technology in the seventies signified a significant turning point permitting the evolution of more sophisticated digital control systems, as those implemented with early industrial-grade robotics[3]. By the 1980s, Digital Signal Processors (DSP) and microcontrollers significantly advanced real-time simulation (RTS) by providing specialized processing capabilities for real-time computations. These developments facilitated more precise modeling of dynamic systems, including aircraft flight dynamics, power electronics, and real-time control systems, which became possible paving the way for real-time simulation to penetrate other engineering fields[18]. The rapid advancement of inte-

grated simulation platforms that emerged in the 1990s was marked by dSPACE[17]. This platform combined high-performance hardware with a convenient software interface to a glorious mix, far improved from their predecessor, fortifying real-time control and hardware-in-the-loop (HIL) testing, allowing an engineer to introduce fundamental physical components in the simulation environment for a more realistic validation. Furthermore, FPGAs introduced parallel processing capabilities to real-time simulation, leading to greater performance in real-time simulation and allowing for much more computationally heavy simulation tasks[18].

Real-time simulation frameworks are essential by enabling design verification before deployment. High-performance commercial solutions such as dSPACE offer excellent real-time capabilities, but their high cost makes them inaccessible to academic researchers, small-scale developers, and budget-constrained industries[4]. In contrast, low-cost alternatives based on open-source software and off-the-shelf hardware have emerged today, however almost all existing solutions still face challenges from scalability, computational performance, real-time determinism and interoperability with widely adopted model-exchange standards like the Functional Mock-up Interface (FMI). Despite these advances a significant gap remains, there is no widely available cost effective real time simulation frameworks that balance affordability with real time accuracy, particularly when paired with advanced frontend. Addressing this challenge requires low-cost solutions that utilize open-source software in conjunction with low-cost hardware whilst retaining sufficient accuracy and reliability needed for real-time applications. A well-designed frontend, such as a web-based React interface, can further enhance accessibility by simplifying setup and configuration, and real-time data visualization.

This research proposes the design and development of an affordable, user-friendly real-time simulation framework that uses Raspberry Pi 4B, NixOS-RT, and FMPy as an FMU execution engine to counteract such situations. The backend built with FastAPI ensures seamless communication between the FMU manager and the React-based front-end interface, creating a scalable, lightweight, and application-agnostic simulation environment. Lessening the cost constraints of real-time simulation, this framework achieves a balance between the hard real-time features, the openness of its tools, and the capability to be accessible via the web, thereby improving the accessibility of RT simulation to researchers and developers. The paper describes a structured methodology toward an accessible real-time simulation platform, representing a step toward a cost-effective alternative to existing proprietary solutions. A key outline of design aspects, cost-saving measures, and validation strategies to prove the performance of a system is made to utilize real-time applications.

The structure of the paper is as follows: Section 1 reviews related work, providing an overview of real-time simulation and cost-effective solutions. Section 2 outlines the methodology, detailing the design and implementation of the proposed framework, including hardware and software components. Section 3 presents the results and the discussion, analyzing the performance of the proposed framework and considering future work. Finally, Section 4 brief conclusion.

1 Related Work

Jacobitz, among others, proposed LoRra, a low-cost platform for rapid control prototyping, which would be running on Scilab/Xcos and allows for quite a range of simulations such as Model-in-the-loop, Software-in-the-loop, and Hardware-in-the-loop. The platform embodies model libraries, a code generator, a real-time interface, and real-time hardware, thus offering an economically attractive alternative to commercial tools such as dSPACE systems with Matlab/Simulink[2] . One way to show the usefulness of LoRra is to use it in a state-of-charge estimator for a battery management system[2]. However, LoRra, despite offering quite an interesting proposal for rapid control prototyping, has limitations on its widespread adoption owing to its dependence on Scilab/Xcos. In addition, the platform's effectiveness in other application domains remains largely unexplored and hence offers greater scope for adapting it into real-time simulation applications.

Lu et al. [5] propose the Virtual Test Bed Real-Time (VTB-RT) platform, a low-cost hardware-in-the-loop (HiL) testing approach designed for power electronics controls. Built on open-source software and off-the-shelf hardware, VTB-RT demonstrates a four-step design and testing process. Its application to a boost converter and H-bridge inverter control system achieves time steps of 250-300 µs, demonstrating effective performance with low-cost hardware. This platform, while promising for high-performance real-time

simulation on a budget, is constrained toward referring to power electronics only, thus painting itself less generalizable for other domains. In addition, the capabilities for scalability for more complex systems and performance evaluations for subsequent advanced hardware have been missing for a thorough analysis, marking potential areas for further scientific work and wider applicability.

Walter et al.[6] describe the use of low-cost devices, like Raspberry Pi, in hardware-reduced real-time simulation. The paper indicates how to obtain hard real-time behavior from real-time operating systems, such as Xenomai on the Raspberry Pi while creating integration into model-based design workflows. Through this, the potential of identifying long-term stability problems is highlighted. Their work presents a novel approach for low-cost real-time simulation, which is limited by the inheritance limitation of Raspberry Pi. In contrast, our work uses a similar open-source platform. Still, it leverages a different software stack and optimization techniques, making it more versatile and accessible for a wider range of applications.

Dávila Delgado et al.[4] propose an educational embedded platform realized through digital signal processors, allowing real-time simulations and rapid implementation of complex systems in Simulink. The platform utilizes a PIC32 microcontroller and the MPLAB Device Blocks for Simulink for derivatives generated from an automatic code generation procedure. They validated its capabilities using hardware-in-the-loop (HiL) testing of an engine control system and compared it with a commercial dSPACE system. This paper shows a possible approach to developing low-cost tools for real-time simulation, with performance close to that of the much more expensive available commercial systems, it relies heavily on proprietary tools (MPLAB and Simulink), which limits its flexibility and openness. For our work, we did not adapt their approach because it depends on a proprietary ecosystem; instead, we aimed for a solution that is more adaptable and open-source, providing flexibility for a wider range of applications and users. Further research is required to assess its performance in more complicated scenarios.

Aravena et al. [7] propose a low-cost and dependable real-time control platform based on the TMS320F28379D microcontroller family priced at about 400 dollars. The balance between price and performance-a cost-efficient platform; program in C/C++ or Matlab İt is verified through effective control of the 20 kVA power converter. It is a low-cost and reliable platform, but its applicability outside of power electronics remains an open question. The research gap lies in the narrow application focus and the absence of modern user interfaces or web-based visualization, limiting accessibility and scalability.

Hatledal et al. [8] propose FMU-proxy, an open-source framework that opens up the Functional Mock-up Unit (FMU) to configurable distributed access across various programming languages and platforms. FMU-proxy encapsulates FMUs using a server program that supports several RPC technologies and provides security, reusability, and cross-platform compatibility. However, server-based encapsulation may increase overhead and latency, jeopardizing real-time performance, particularly in highly dynamic simulation environments. More studies are required to evaluate its efficiency in real-time applications.

Legaard et al. [9] propose UniFMU, a tool that simplifies the implementation of FMUs in multiple languages, such as Python, C#, and Java. UniFMU relies ona modular architecture that separates the implementation of the FMI C-API from the model behavior, thereby contributing to greater flexibility and portability. RPC is employed for interoperability, although this approach involves performance trade-offs with direct C-ABI calls. More so, there might also be greater communication latency and extra pure timing costs from using RPC, creating limitations for real-time applications.

2 Methodology

The proposed framework addresses the growing need for affordable real-time control systems in embedded applications. By combining Raspberry Pi 4B's computational power with NixOS-RT's real-time capabilities, we achieve deterministic performance typically requiring specialized hardware. In addition to these core hardware and software components, the FMU Execution Engine (FMPy) is integrated into the system to support model-based design through Functional Mock-up Units (FMUs). FMUs allow for the exchange of simulation models between different tools and platforms, facilitating the interoperability of different simulation components. The Fast API backend is another key component responsible for handling communication between the simulation engine and the frontend interface. It handles real-time data transmission, ensuring a smooth

exchange of simulation results and configurations between the system components. Furthermore, React Front provides a user-friendly web platform that allows users to configure simulation settings, initiate simulations, and monitor results in real time. The modular design of the system ensures that users can easily modify simulation parameters and interact with the real-time data provided by the backend. The integration of FMU standards and modern web technologies creates a flexible platform for simulation and control tasks. The modular architecture of this approach allows for scalability, flexibility, and user-friendliness.

FMU Integration and Real-Time. The framework's scalability depends on its ability to handle FMUs, a widely recognized standard related to model-based design and simulation. Within the FMPy open-source library, the framework runs FMUs in real-time while keeping synchronized.

- **Real-time synchronization** will be maintained using NixOS-RT, which ensures that the time step of simulations adheres to timing constraints.

Frontend-Backend Architecture. A React-based frontend will serve as the user interface, acting as the user interface allowing in-system interaction through the web browser by users. The front end shall enable users to upload FMU models, provide simulation parameters, and visualize results in real-time. With this approach, it provides a flexible and easy-to-use platform for a variety of users.

The FastAPI backend will manage:

- **FMU management:** Handling the loading, execution, and configuration of FMUs.
- **Real-time data transmission:** Ensuring that simulation results are communicated to the front end in real-time.
- **Scalability:** Supporting multiple simultaneous simulations and configurations.

2.1 System Overview

The architectural layout is depicted in Fig.1. The real-time simulation system is designed for a model-based approach to design based on standards in FMIs and user interaction via a React-based frontend interface. The core components of the system are as follows:

- **Raspberry Pi 4B:** Computational power for real-time execution.
- **NixOS-RT:** A real-time operating system.
- **FMU Execution Engine (FMPy):** Enables the integration and execution of Functional Mock-up Units (FMUs)[14].
- **FastAPI Backend:** Handles communication between the simulation engine and the frontend interface, supporting integration with experiments and ensuring scalable performance.
- **React Frontend:** A user-friendly web interface allowing users to configure simulation settings, initiate simulations, and receive real-time data and feedback.

2.2 Modular Breakdown and Component Design

The system follows a modular architecture where each component plays a specific role in achieving real-time simulation goals. Below, we outline how the key components interact.

Raspberry Pi 4B The Raspberry Pi 4B provides much-needed computational resources to run real-time simulations. The powerful quad-core ARM Cortex-A72 CPU able to execute simulations is appropriate for several embedded applications. The Raspberry Pi interacts with NixOS-RT to ensure real-time behavior, which circumvents the performance bottlenecks known to affect low-cost hardware platforms.

- **The RaspberryPi 4B:** has a quad-core ARM Cortex-A72 CPU, which runs at 1.5 GHz.

NixOS-RT: NixOS-RT is a real-time operating system that works towards achieving time-deterministic execution. The real-time kernel allows precise timing in scheduling and guarantees that simulation time steps are in conformity with the timing constraints required for a proper simulation. NixOS-RT enables running both user-level applications and critical real-time processes. To achieve real-time operation performance on Raspberry Pi 4B, we installed NixOS-RT, a real-time operating system mounted on NixOS distribution. The integration of NixOS-RT in the Raspberry Pi was a major step to ensure that the system could meet the timing demands for a real-time simulation. Below are the main

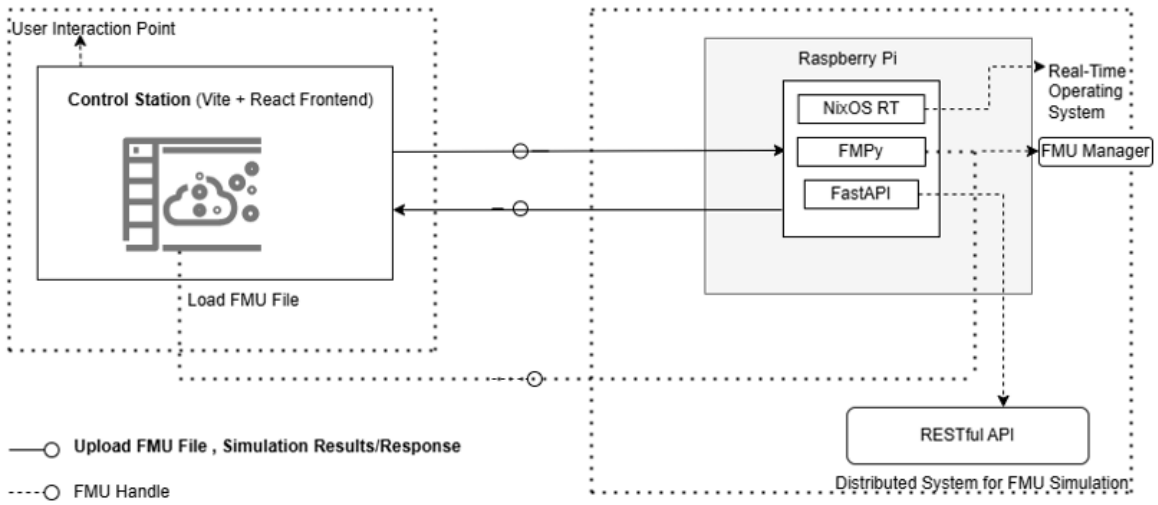


Figure 1: System Architecture.

steps we took in installing, configuring, and setting up NixOS-RT on Raspberry Pi 4B:

- **T**he standard NixOS was installed on the Raspberry Pi 4B, involving flashing the NixOS image onto a microSD card via Etcher.
- **A**fter displaying the form, the Raspberry Pi was booted into NixOS, where we accessed the system through an SSH connection for further configuration.
- **N**ext was the installation of a real-time kernel for NixOS-RT, a modification of the standard Linux kernel that would allow for preemptive multitasking, latency scheduling and precise timing control.
- In NixOS, the kernel configuration is declarative by nature and can then be defined in the system's configuration files. The configuration of the real-time kernel was added to the file /etc/nixos/configuration.nix under the boot.kernel Packages option specifying a real-time variant of the Linux kernel.

After installation and configuration of the real-time kernel, we performed several tests to verify the real-time behavior of the systems.

Integrating FMPy and FastAPI: The FMPy library was integrated into the back-end built on FastAPI with the aim of giving beneficial support to the handling and lightweight execution of FMUs in the real-time simulation framework. The delicate join will permit the use of all the capabilities of FMPy for FMU simulation, while FastAPI will host the communication integration for real-time data exchange between the back-end and front-end.

FastAPI is the backend framework responsible for coordinating simulation tasks with the FMPy execution while ensuring real-time processing to the front end. The backend is built to support the asynchronous functionality of the FastAPI library, allowing the execution of multiple concurrent simulation tasks without blocking or deteriorating real-time functionality. FastAPI also handles the loading, configuration, and execution of FMUs through the API entry points and WebSocket connections for seamless communication between the backend and the front end.

Vite and React for User Interaction This front end is built on React, a widely used JavaScript library for building user interfaces, combined with Vite, which is a modern build tool that vastly improves the development experience by providing fast on-demand compilation and hot-reloading. This combination creates a dynamite, powerful user interface that promotes efficient interaction between the end-user and the real-time sim-

ulation backend.

The front-end core is designed based on a modular component architecture, where around each component of the user interface exists an independent component. It encompasses

- Model Management is for uploading FMU models and configuration.
- Simulation Control to start and stop simulations.

3 Results and Discussion

3.1 Results

The primary goal of this project was to setup a low-cost real-time simulation system for Raspberry Pi 4B and NixOS-RT. The assessment sheds light on how the system could have deterministic execution for real-time simulation applications in tight coupling with the FMU-based models.

FMU Execution and Visualization The framework is tested with the FMU model. While running perfectly, it nevertheless demonstrated that, indeed, the Raspberry Pi 4B and NixOS-RT can manage real-time simulation tasks. Figures 2 and 3 show snapshots of the running FMU into the developed interface.

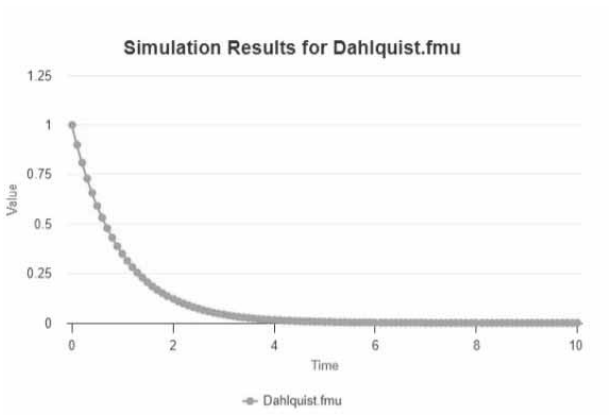


Figure 2: Dahlquist FMU Simulation.

Real-Time Execution and Determinism Its capability for executing simulation steps with predetermined time intervals defined its real-time behavior. The addition of the NixOS-RT enabled preemptive multitasking and increased predictability of task scheduling. The system showed consistent time-step execution, which is crucial for real-time applications. Furthermore, the real-time kernel formed a hard barrier concerning simulation timing and eliminated excessive latency and drift in execution timing from the simulation.

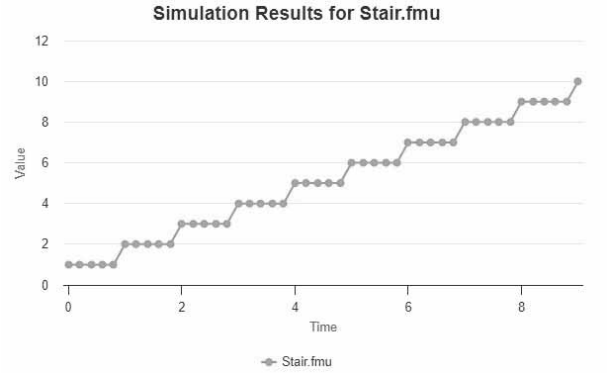


Figure 3: Stair FMU Simulation.

3.2 Discussion

The proposed real-time simulation framework incorporates low-cost hardware with open-source software in a flexible and modular way to execute Functional Mock-Up Units. Design choices include Raspberry Pi 4B, NixOS-RT and FastAPI with accessibility and scalability for the real-time simulation environment in mind. The topics of system architecture, trade-offs, and prospects for enhancement are considered.

System Architecture and Performance Considerations The framework is based on a Raspberry Pi 4B, which provides a promising equilibrium between computing performance and cost-effectiveness. NixOS-RT has been integrated into the framework and acts as the real-time operating system responsible for precise scheduling and executing a simulation produced via a real-time kernel for enhanced timing control. However, being an embedded system, any embedded computation can be performance-limiting because of certain environment constraints, software optimizations, or hardware limitations.

The backend invoking FastAPI synchronizes the communication between simulation processes and the

GUI while executing FMUs through FMPy.The current setup enables parallel execution of two FMUs using FMPy. Due to FastAPI being fully asynchronous in structure, the model is capable of running through multiple simulation instances at once, thereby exempting itself from blocking data migration.

3.3 Future Work

While the future presents multiple avenues of development to overcome the challenges identified above and extend the capabilities of the system, some of them are:

- **Optimizing Multi-FMU Execution:** Different mechanisms to enable execution across more than two FMUs are being explored, such as process-based isolation, lightweight containers, or alternative FMU execution engines.
- **Performance Profiling and Optimization:** An examination of CPU and memory usage from performance profiling could optimize the scheduling policies in order to minimize computational overhead.
- **Real-Time Synchronization Improvements:** Improvements in timing accuracy could be achieved by fine-tuning NixOS-RT configurations or exploring more suitable real-time scheduling algorithms.
- **Distributed Real-Time Simulation:** Implementation of multi-node executions on several Raspberry Pi devices that would work simultaneously to distribute computation load.
- **Hybrid Cloud-Edge Co-Simulation:** Processing high-complexity FMUs in a cloud environment and a real-time edge execution.
- **Edge AI for Real-Time Resource Management:** Using edge AI models that can run on specialized hardware like Raspberry Pi or FPGAs to make real-time scheduling decisions without compromising on timing constraints.

4 Conclusion

This paper outlines a preliminary procedure for developing a more straightforward framework for performing cost-effective real-time simulations of FMU-based model simulation using embedded hardware. The development, using Raspberry Pi 4B, NixOS-RT, and open source, FMPy, and FastAPI, allows flexibility and scalability for real-time simulations. The system synchronizes in real-time and presents an approachable environment for model-based design and co-simulation and hence has the potential to be embraced widely by researchers and developers with resource constraints.

Currently, the framework supports a maximum of two FMUs, with further work needed to enhance multi-FMU handling, scalability, and overall performance in the case of more complex simulations. Future work will focus on enhancing its functionality and on the exploration of distributed execution strategies. This work represents a step towards the provision of low-cost real-time simulations, which could further the development of platforms as cost efficient for high performance of real-time control systems in embedded applications.

References

[1] Maria A. Introduction to modeling and simulation. InProceedings of the 29th conference on Winter simulation 1997 Dec 1 (pp. 7-13).

[2] Jacobitz S, Liu-Henke X. The Seamless Low-cost Development Platform LoRra for Model based Systems Engineering. InMODELSWARD 2020 (pp. 57-64).

[3] Bélanger J, Venne P, Paquin JN. The what, where and why of real-time simulation. Planet Rt. 2010 Oct 4.

[4] Dávila Delgado E, Raygoza Panduro JJ, Becerra Álvarez EC, Espinoza Jurado FJ, Gutiérrez Frías EF. Low cost DSP-based educational embedded platform for real-time simulation and fast implementation of complex systems in Simulink. Computer applications in engineering education. 2019 Jul;27(4):955-70.

[5] Lu B, Wu X, Figueroa H, Monti A. A low-cost real-time hardware-in-the-loop testing approach of power electronics controls. IEEE Transactions on Industrial Electronics. 2007 Mar 19;54(2):919-31.

[6] Walter J, Fakih M, Grüttner K. Hardware-based real-time simulation on the raspberry pi. InProceedings of the 2nd Workshop on High Performance and Real-time Embedded Systems, Vienna, Austria 2014.

[7] Aravena J, Carrasco D, Diaz M, Uriarte M, Rojas F, Cardenas R, Travieso JC. Design and implementation of a low-cost real-time control platform for power electronics applications. Energies. 2020 Mar 24;13(6):1527.

[8] Hatledal LI, Zhang H, Styve A, Hovland G. Fmu-proxy: A framework for distributed access to functional mock-up units. InProceedings of the 13th International Modelica Conference 2019. Linköping University Electronic Press.

[9] Legaard CM, Tola D, Schranz T, Macedo HD, Larsen PG. A Universal Mechanism for Implementing Functional Mock-up Units. InSIMULTECH 2021 Jul 7

(pp. 121-129).

[10] González-Vargas AM, Serna-Ramirez JM, Fory-Aguirre C, Ojeda-Misses A, Cardona-Ordoñez JM, Tombé-Andrade J, Soria-López A. A low-cost, free-software platform with hard real-time performance for control engineering education. Computer applications in engineering education. 2019 Mar;27(2):406-18.

[11] Lee W, Yoon M, Sunwoo M. A cost-and time-effective hardware-in-the-loop simulation platform for automotive engine control systems. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering. 2003 Jan 1;217(1):41-52.

[12] Vogel S, Mirz M, Razik L, Monti A. An open solution for next-generation real-time power system simulation. In2017 IEEE Conference on Energy Internet and Energy System Integration (EI2) 2017 Nov 26 (pp. 1-6). IEEE.

[13] Magnani GF, Rocco GR. Real-time simulation of modelica models under Linux/RTAI. InPaper presented at the 4th International Modelica Conference 2005 Mar 7.

[14] FMPy Contributors. FMPy: A Python library for FMUs [Internet]. 2021 [cited 2025 Feb 01]. Available from: https://github.com/CATIA-Systems/FMPy

[15] Faruque MO, Strasser T, Lauss G, Jalili-Marandi V, Forsyth P, Dufour C, Dinavahi V, Monti A, Kotsampopoulos P, Martinez JA, Strunz K. Real-time simulation technologies for power systems design, testing, and analysis. IEEE Power and Energy Technology Systems Journal. 2015 Jun 29;2(2):63-73.

[16] Balda P. Real-time simulator of component models based on Functional Mock-up Interface 2.0. In2017 21st International Conference on Process Control (PC) 2017 Jun 6 (pp. 392-397). IEEE.

[17] Menghal PM, Laxmi AJ. Real time simulation: Recent progress and perspective. InConf. Rec. Int. Journal Of Electrical Engineering And Electrical Systems (IJEEES) 2010 (Vol. 3, No. 01, pp. 45-59).

[18] Mencer O, Allison D, Blatt E, Cummings M, Flynn MJ, Harris J, Hewitt C, Jacobson Q, Lavasani M, Moazami M, Murray H. The History, Status, and Future of FPGAs: Hitting a nerve with field-programmable gate arrays. Queue. 2020 Jun 30;18(3):71-82.

[19] Andreasson J, Machida N, Tsushima M, Griffin J, Sundström P. Deployment of high-fidelity vehicle models for accurate real-time simulation.

[20] Worschech N, Mikelsons L. A toolchain for real-time simulation using the OpenModelica compiler.

[21] Jiang Z, Hossain S, Huang H. A distributed, real-time simulation platform for aerospace power system design, testing and evaluation. In2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS) 2016 Jul 25 (pp. 9-15). IEEE.

Segmentation of bus driving data: A clustering-based approach to identify similar driving sections

Anna Schniertshauer[1*], Sven Angerer[2], Andreas Grabow[1], Michael Schlick[1]

[1]Institute for Automotive Systems Engineering, Technische Hochschule Ulm, Prittwitzstraße 10, 89075, Ulm; *anna.schniertshauer@thu.de*

[2] Ulm University, 89081 Ulm

Abstract. Driving cycles are required for a variety of applications including longitudinal dynamics simulations. For the generation of representative driving cycles, a driving data analysis is indispensable. This paper proposes a method to efficiently segmenting data and subsequently identifying typical trip sections. A first cluster analysis is performed on individual data points using the kmeans++ algorithm. Based on the results, the consecutive data points are segmented into microsegments. Subsequently, these microsegments are being clustered in a second cluster analysis. The results obtained reveal patterns of cluster formations that are similar to those observed in the cluster analysis of individual data points. Another segmentation, based on the minimum duration of standstill times between two driving sections, enables the identification of typical trips of longer durations. This is achieved by taking the proportions of the microsegments assigned to the same cluster as input variables for the third cluster analysis. Thereby, groups of similar trips can be identified with the typical distribution of microsegment proportions. Thus, the developed method yields representative trip sections for a driving dataset and thereby forms a basis for generating representative driving cycles both in the research and in the development of simulation-based technologies.

Introduction

To reduce greenhouse gas emissions related to public transit vehicles, such as buses, the transition from conventional internal combustion drive trains to alternative drive trains is necessary. When developing new drive trains, longitudinal dynamics simulations play an important role. In order to fully exploit the capabilities of these simulations, it is essential to use representative driving cycles for different application scenarios as stimuli. This facilitates the evaluation of the vehicle's ability to meet not only the requirements of standardized driving cycles for buses, such as the Standardized On-Road Tests (SORT), but also the requirements of real customer use. To obtain these representative driving cycles, it is necessary to identify typical operating trips by analyzing driving data from buses already operating in the application area. To this end, this paper proposes an approach to effectively segmenting the dataset and finding groups of similar trips using a clustering algorithm. This approach provides a basis for generating representative driving cycles, which can subsequently be used in longitudinal dynamics simulations and support the development of new drive trains.

There are several approaches for analyzing driving data, whereby the dataset is segmented into brief driving sections, hereby referred to as microsegments. One of these approaches involves the establishment of a fixed duration for the segments. Montazeri-Gh et al. use a duration of 150 s [1] whereas Brady and O'Mahony use a duration of 30 s [2]. The disadvantage of a fixed duration is that a single microsegment contains several different driving scenarios. Therefore, in this work, the driving data is segmented into microsegments of variable length which start or end when the driving conditions change. A similar approach is employed by Langner et al. to extract representative driving scenarios from real-world driving data [3]. The segmentation is based on significant changes in features such as speed limits, street types and curviness [3]. However, there is no detailed description of how these significant changes in the features are identified. The approach in this paper is to realize the segmentation by clustering individual data points and creating microsegments

from consecutive data points that are in the same cluster. Chetouane et al. also perform a cluster analysis of individual data points to identify similar driving episodes from an autonomous driving dataset [4]. However, the post-processing differs from the one chosen in this paper and the identified driving episodes are not used for further characterization of longer driving segments [4].
The methodology employed for the segmentation of driving data and the identification of groups of similar trips is examined first. Then a description of the dataset utilized to develop the method follows. The implementation of the approach is divided into two steps. Initially, data points are segmented into microsegments, which are then clustered. Subsequently, trips are segmented based on a minimum standstill time, followed by a cluster analysis. Finally, a discussion of the results precedes the conclusion of this work.

1 Methodology

With the aim of identifying similar trips from a driving dataset, the method shown in Figure 1 is developed. The raw dataset undergoes a series of preprocessing steps to obatin the desired data quality prior to segmentation. Two segmentation levels are employed for this purpose. On the first level, the data is segmented into microsegments, which are characterized by similar driving conditions. A new segment is created when the considered variables change significantly. To achieve this segmentation, all data points are categorized with a first cluster analysis. For all cluster analyses in this paper the kmeans++ algorithm by Arthur and Vassilvitskii is used [5]. It is an augmentation of the well-known cluster algorithm kmeans, which was developed by Lloyd [6]. Kmeans is a partitioning cluster algorithm and is commonly used to cluster driving data [1, 2, 4, 7]. The algorithm takes the number of clusters k as an input and chooses k cluster centers randomly from the dataset. In the next step the Euclidean distances from each point to the cluster centers are calculated and the data points are assigned to the cluster center with the minimal distance to the point. New cluster centers are calculated as the average of all assigned points and the procedure is repeated. Kmeans++ chooses the initial cluster centers based on the distribution of the input data, which leads to better results and a faster convergence [5]. To choose the number of clusters k, three metrics are considered. The Silhouette score compares the difference of the average distances between a given point and the points within the same cluster and the average distance between that point and the points in the nearest cluster with the maximum of these two values [8]. The values are in a range of -1 and 1, where a higher value indicates a better cluster result [8]. The Davies-Bouldin index compares the size of the clusters with the distance between them. Better partition and separation are achieved with a lower Davies-Bouldin index [9]. The third index used is the Calinski-Harabasz index, which validates the cluster validity by calculating the proportion obtained by dividing the total dispersion between clusters by the total dispersion within clusters across all clusters [10]. A higher index indicates better results [10]. All three metrics are also used by Chetouane and Wotawa to evaluate the results of clustering driving data [11].

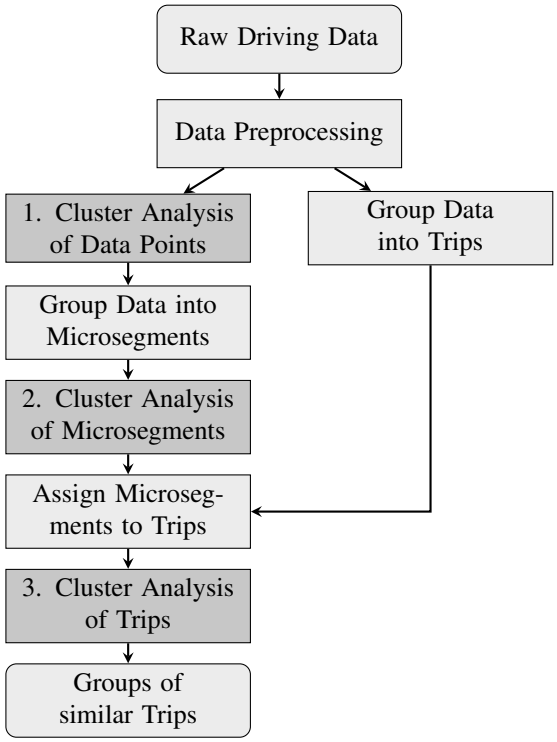


Figure 1: Segmentation and clustering process

The first cluster analysis assigns to each data point a cluster index, ci_p. It is imperative to ensure that the occurrence of single data points, possessing a different cluster index compared to their surrounding data points that share the same cluster index, does not result in the initiation of a new microsegment. To this end, a smoothing method is developed. Furthermore, driving sections are identified that consist of data points with highly divergent

cluster indices. They are grouped as self-contained microsegments. The values of the individual data points are aggregated so that each microsegment has one value for each cluster variable. The start and end times of each microsegment are taken from the first and the last data point of this microsegment. Subsequently, a second cluster analysis is performed on the aggregated microsegments, which assigns to each microsegment a cluster index ci_m.
On the second level of segmentation, the preprocessed dataset is segmented into trips based on a defined threshold for standstill time. If the vehicle stands still for a duration exceeding the threshold, the current trip ends and a new trip starts as soon as the vehicle moves off again. This results in larger segments than those extracted from the individual data point clustering. Therefore, each trip consists of several microsegments. The start and end time of each trip and microsegment facilitates the temporal assignment of the microsegments to the trips. For each trip, the proportions of time spent by the vehicle in a microsegment of the clusters ci_m are calculated and taken as new variables for the third cluster analysis at the trip level. The quantity of cluster variables at this level is defined by the cluster number that is derived from the second cluster analysis conducted at the microsegment level. The analysis yields groups of trips with similar proportions of microsegments assigned to the same cluster index ci_m.

2 Data Source and Preparation

The data source used for this paper is the Zurich Transit Bus dataset [12]. It contains driving data from two electric city buses over a period of 3.5 years. To prepare the data for the cluster analysis and to ensure a sufficient data quality, a variety of filtering and data preprocessing steps are carried out preceding the clustering. The signals used from the dataset are the time of recording, the latitude and longitude of the vehicle and the velocity of the vehicle.
To filter out incomplete data, the dataset is split into individual days. A day is only included in the analysis if the positional information *or* the velocity signal is not missing for more than 60 s. If the positional information *and* the velocity signal is missing, the day is included since this corresponds to a break. This results in a dataset of 7.5 million data points with a sampling rate of less than 10 s for 99.2% of the data points and a sampling rate of exactly one second for 93.1% of the data points. If the positional information is missing, it is interpolated linearly.
As described by Widmer et al., the dataset contains the raw measurements of the sensors [12]. To counteract on inaccuracies of the latitude and longitude signal introduced by the GNSS sensor, the Valhalla Map Matching API [13] is used to map the coordinates onto the road.
The altitude information for the matched coordinates is retrieved from OpenStreetMap [14] and has a resolution of 1 m. To smooth the signal, the average altitude of the preceding and succeeding 15 s of each data point is calculated and used for further processing.
The velocity of the vehicle is calculated by Widmer et al. based on the signal of rotational speed sensors mounted on the motor shafts of the vehicle and then multiplied with a transmission ratio γ [12]. To counteract slight deviations introduced through this estimation, all values with negative velocity values are set to zero. To calculate the slope, the difference in altitude is divided by the distance traveled between two data points. For the slope of one data point, the average slope of the preceding and succeeding data point is used. To avoid dividing by a small value of the traveled distance, the slope is set to zero if the velocity of a data point or the preceding or succeeding data point is less than $0.2\ \frac{m}{s}$.

3 Segmentation and Clustering of Microsegments

In order to identify similar microsegments, individual data points are clustered. For this, the slope and velocity of the vehicle are used, as these are typical input variables for longitudinal dynamics simulations. To filter out any outliers, the interpercentile range (ipr) between the 5 and 95 percentile is calculated. Values smaller than $p_5 - 1.5 \cdot ipr$ or larger than $p_{95} + 1.5 \cdot ipr$ are neglected. This filters out 0.2% of the values corresponding to 14 385 data points. The data is then scaled using a Standard Scaler to remove the mean and scaling it to unit variance [15]. Each data point is clustered using the kmeans implementation by Scikit learn [15] with the kmeans++ algorithm for choosing the initial clusters [5]. This results in each data point being assigned to one of k clusters. To determine the number of clusters, the scores in Table 1 are considered. As the Silhouette score and Calinski-Harabasz index indicate, the best results are achieved with $k = 4$. The cluster index a data point is assigned to is called ci_p. As can be seen in Figure 2, the clusters with $ci_p = 2$ and $ci_p = 4$ contain data points with

moderate slope but differ in velocity. The clusters with $ci_p = 1$ and $ci_p = 3$ contain data points with moderate to high absolute slope values regardless of the velocity. In

k	Silhouette	Davies–Bouldin	Calinski–Harabasz
2	0.370	1.099	398668
3	0.374	0.929	435960
4	0.405	0.876	470390
5	0.356	0.885	459099
6	0.360	0.866	447477

Table 1: Scores to determine the number of clusters

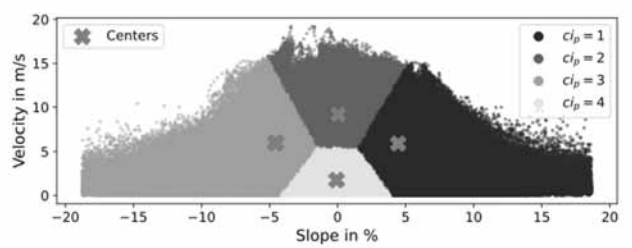


Figure 2: Clustering result of individual data points

the next step, consecutive data points with the same cluster indices ci_p are grouped together into microsegments. As can be seen in Figure 3(a), there are data points with a different cluster index than the cluster index that is dominating in this driving section. To address this issue a smoothing process is introduced. For this, each data point is first transformed into a binary vector representation v, where only the value at the index of the cluster (ci_p) is set to one.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_k \end{bmatrix} \qquad v_i = \begin{cases} 1 & \text{if } i = ci_p, \\ 0 & \text{otherwise.} \end{cases}$$

Since each data point can only be assigned to one cluster, each vector always only has one entry not equal to zero, which means that all cluster indices have the same Euclidean distance to each other and the smoothing can be performed. The average is calculated row-wise in a rolling window. The row with the highest average is then set to one and all other values are set to zero. In a last step the vector can be transformed back into the cluster assignment. An example of this process is demonstrated in Table 2 with a fixed window size of 3 data points. The actual implementation of the algorithm uses a window size of 9 s centered around the current data point. Since the majority of data points have a sampling rate of 1 s, this corresponds to a window size of 9 data points. As shown in Figure 3(b), this method successfully assigns the dominating cluster index to individual points that differ from the dominating cluster index. Depending on the size of the rolling window, even multiple deviating points can be adjusted this way. An important aspect which this method also fulfills is that an already clear transition between microsegments remains intact.

2	2	1	1	3	1	1

↓ Transformation in $\{0,1\}^k$ ↓

$$\begin{bmatrix}0\\1\\0\end{bmatrix}\ \begin{bmatrix}0\\1\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\1\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}$$

↓ Rolling Average ↓

$$\begin{bmatrix}0\\1\\0\end{bmatrix}\ \begin{bmatrix}\frac{1}{3}\\\frac{2}{3}\\0\end{bmatrix}\ \begin{bmatrix}\frac{2}{3}\\\frac{1}{3}\\0\end{bmatrix}\ \begin{bmatrix}\frac{2}{3}\\0\\\frac{1}{3}\end{bmatrix}\ \begin{bmatrix}\frac{2}{3}\\0\\\frac{1}{3}\end{bmatrix}\ \begin{bmatrix}\frac{2}{3}\\0\\\frac{1}{3}\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}$$

↓ Assignment ↓

$$\begin{bmatrix}0\\1\\0\end{bmatrix}\ \begin{bmatrix}0\\1\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}\ \begin{bmatrix}1\\0\\0\end{bmatrix}$$

↓ Transforming to Cluster ↓

2	2	1	1	1	1	1

Table 2: Example on how to smooth the cluster result with a rolling window of length 3.

(a) Before Smoothing (b) After Smoothing

Figure 3: Correcting missmatched points after clustering

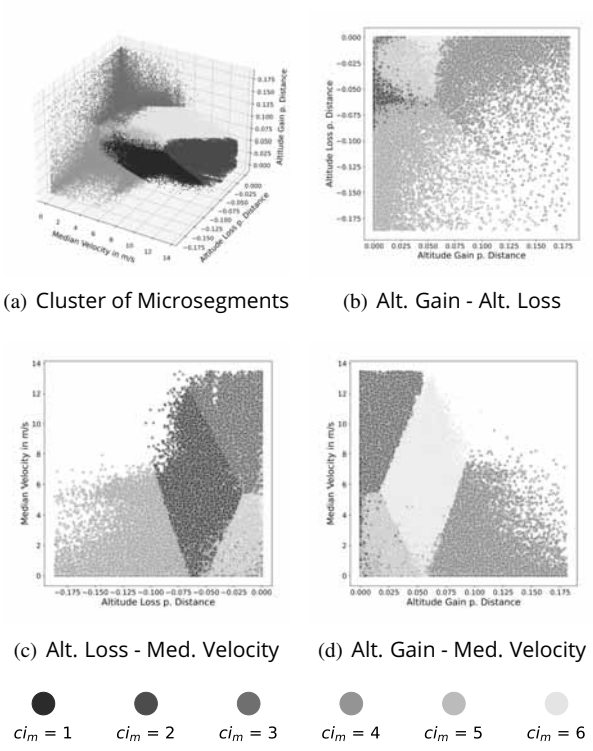
In a next step the driving sections are handeled in which a dominant cluster is not apparent. For this, the variance of the smooth clusters is calculated. If in a window of 9 s the cluster index changes more than four times or the

variance is smaller than 0.1 and the cluster index changes more than two times, the data point is labeled as diverse and assigned to an additional cluster index value. After the introduction of this additional cluster index value, the smoothing is performed again. If a microsegment has a duration of less than 5 s, it will be assigned to the microsegment closest to the start or end time of the current microsegment. If the time difference to the previous and next microsegment does not differ, the microsegment will be assigned to which ever has the shorter duration. The entire dataset now consists of 325 311 microsegments with a duration of at least 5 s. To uniquely identify each microsegment, a microsegment index is assigned to each microsegment. This index is employed to aggregate the data and calculate the metrics later used for clustering. The aggregated table also contains the start and end time of each microsegment, which is later used to match the microsegments to the corresponding trips.
To not only capture the average or median slope of a microsegment, the cumulative sum of altitude gain and altitude loss are calculated while aggregating. These variables are now used to calculate the altitude gain per distance and altitude loss per distance. Together with the median velocity of a microsegment these two variables are used to cluster the microsegments. To filter out any outliers, microsegments in which the velocity falls within the bottom or top 5% of values or the altitude loss per distance or altitude gain per distance falls within the top or bottom 3% of values, are not considered. To determine the number of clusters, the metrics in Table 3 are considered. Based on that, a cluster number of $k = 6$ is chosen.

k	Silhouette	Davies–Bouldin	Calinski–Harabasz
2	0.324	1.292	139185
3	0.378	1.008	179274
4	0.385	0.909	207663
5	0.379	0.942	198766
6	0.404	0.876	213039
7	0.391	0.908	202768
8	0.390	0.922	200846

Table 3: Scores to determine the number of clusters

As can be seen in Figure 4, the clusters are separated by the median velocity, as well as the altitude gain and altitude loss within a microsegment. The cluster with $ci_m = 2$ contains microsegments with the highest velocities but only little to moderate changes in altitude gain and loss. The clusters with $ci_m = 1$ and $ci_m = 6$ contain microsegments with moderate velocities and are separated by altitude gain and altitude loss. While the cluster with $ci_m = 1$ contains microsegments with moderate altitude loss and almost no altitude gain, the cluster with $ci_m = 6$ contains microsegments with moderate altitude gain and only very little altitude loss. The cluster with $ci_m = 5$ contains the microsegments with low velocity and almost no altitude gain or loss. In contrary to that, the clusters with $ci_m = 3$ and $ci_m = 4$ contain microsegments with velocities up to 6.5 $\frac{m}{s}$. While the cluster with $ci_m = 3$ contains segments with moderate to high altitude gain and moderate altitude loss, the cluster with $ci_m = 4$ contains segments with moderate altitude gain and moderate to high altitude loss.



(a) Cluster of Microsegments (b) Alt. Gain - Alt. Loss

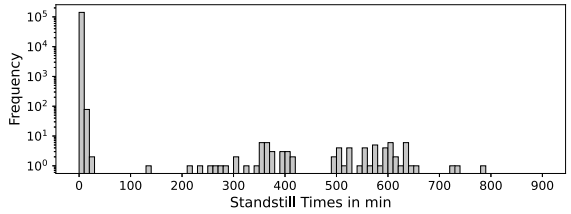
(c) Alt. Loss - Med. Velocity (d) Alt. Gain - Med. Velocity

Figure 4: Cluster Result Microsegments

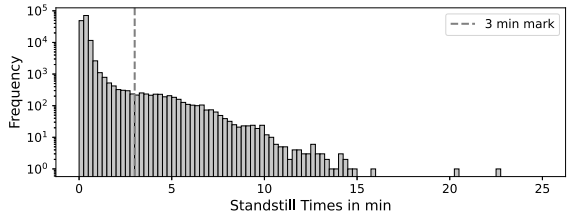
4 Segmentation and Clustering of Trips

In addition to segmenting the dataset based on changes of the velocity or the slope, segmentation is performed at a higher level based on a minimum time of standstill. A trip ends when the velocity signal is below 0.5 $\frac{m}{s}$ for

a certain amount of time. As soon as the velocity signal exceeds this value again, a new trip begins. Figure 5(a) shows a histogram of the standstill times up to a duration of 900 minutes. This exposes that most standstill times are below 20 minutes, which correspond to short breaks and traffic flow interruptions during daily operation. The standstill times between 200 and 800 minutes most probably reflect the nighttime breaks. However, choosing a minimum standstill time of 200 minutes would result in a total of only 229 trips because only 0.16 % of standstills have a minimum duration of 200 minutes. Consequently, the minimum standstill time is set at a lower threshold. As can be seen in Figrue 5(b) a high proportion of standstill times are below a duration of three minutes. Standstills of less than three minutes are therefore attributed to normal standstill times during operation, for example at bus stops or traffic lights. Standstills that last longer than three minutes account for 2.44% of all standstills and are defined as a segmentation criterion for the trips. This results in a total number of 3 475 trips, 99.42% of which have a duration of less than 300 minutes.



(a) Histogram of Standstill times until 900 minutes



(b) Histogram of Standstill times until 25 minutes

Figure 5: Histograms of Standstill times

A closer look reveals that 10.16% of the trips are shorter than 30 s and 11.08% are not longer than 5 minutes. However, a trip duration of less than 5 minutes is not reasonable. Therefore, all trips of less than 5 minutes are excluded from further analysis. This leaves 3 090 trips for the cluster analysis. After segmentation, all 325 311 microsegments are temporally assigned to the trip that includes the microsegment. Since trips with a duration of less than 5 minutes are neglected, 0.97% of microsegments cannot be assigned to trips. For each trip, the temporal proportions of microsegments with cluster index ci_m are calculated. As the best result within the cluster analysis of the microsegments is achieved when $k = 6$ is set, it leads to six new input variables for the cluster analysis of the trips. Outliers are identified and excluded by neglecting data points if the value of any of the used variables fall within the 1% lowest or highest values. From 3 090 trips before filtering there are 2 866 left after filtering for the cluster analysis.

To determine the number of clusters k with the best cluster results, the metrics in Table 4 are calculated for different values of k. As the Davies-Bouldin and Calinksi-Harabasz Index indicates best results for $k = 6$, a cluster number of six is chosen.

k	Silhouette	Davies–Bouldin	Calinski–Harabasz
2	0.240	1.804	731.8
3	0.175	1.740	631.8
4	0.183	1.625	598.5
5	0.185	1.534	559.7
6	0.189	1.441	540.9
7	0.174	1.450	504.3
8	0.163	1.459	479.6

Table 4: Scores to determine the number of clusters within cluster analysis on trip level

In order to show the results of the cluster analysis with six input variables, a radar chart is plotted in Figure 6. The chart visualizes the centers of the six clusters with the cluster indices $ci_t = 1$ to $ci_t = 6$ from the trip-level cluster analysis. Each axis of the radar chart represents one cluster variable. These variables represent the proportions of microsegments that are assigned to the cluster indices $ci_m = 1$ to $ci_m = 6$. The proportions range from 0% in the center of the chart to 50% in the outermost circle. Each cluster on trip-level is represented by one line. As an example, the trip corresponding to the cluster center of $ci_t = 6$ consists of 26% microsegments which are assigned to $ci_m = 6$ and 5% of microsegments which are assigned to $ci_m = 3$. Figure 6 also shows that the proportions for trips represented by $ci_t = 2$ and $ci_t = 4$ overlap significantly. Likewise $ci_t = 1$ and $ci_t = 5$ have similar curves, but differ in the axes $ci_m = 5$ and $ci_m = 6$. The cluster $ci_t = 6$ clearly stands out due to higher values for $ci_m = 6$ and the cluster $ci_t = 3$ differs by an overall flatter

distribution. The proportions of microsegments assigned to $ci_m = 2$ are the highest, ranging between 35% to 45%, whereas the proportions of $ci_m = 3$ are the lowest with proportions under 9%.

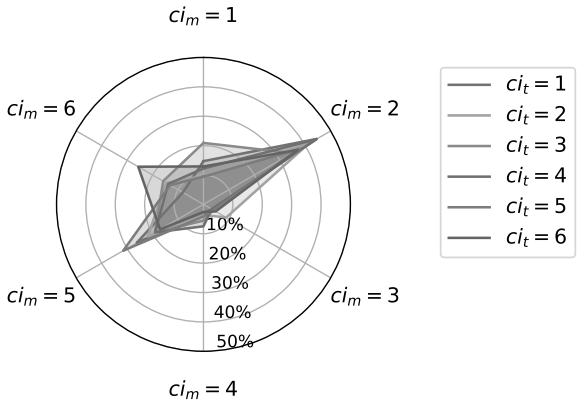


Figure 6: Radar chart of trip-level cluster centers of clusters $ci_t = 1,...,6$ representing temporal proportion of microsegment clusters $ci_m = 1,...,6$

5 Results and Discussion

The clustering of individual points of driving data from electric city buses leads to the differentiation of four driving scenarios. Cluster $ci_p = 4$ represents scenarios in which the vehicle is not moving or moving at a very slow velocity, for instance at bus stops. By contrast, $ci_p = 2$ indicates normal driving in urban traffic at velocities ranging from 6 to 19 $\frac{m}{s}$. The remaining two clusters contain scenarios in which the bus is either ascending or descending an incline, irrespective of its velocity. When consecutive data points are combined into microsegments based on the preceding cluster results and the microsegments are then clustered again, a similar result is obtained. Again we have a cluster containing microsegments with low median velocity and a cluster containing microsegments with high median velocity. Furthermore, there are two clusters, each predominantly containing microsegments with either altitude gain or loss. However, two additional clusters are identified, representing microsegments with high altitude gain or loss, or both, and low to moderate values for the median velocity. The additional clusters can be attributed to the replacement of slope with the altitude gain and loss and the fact that a driving section is now considered instead of individual points. Therefore sections with altitude gain as well as altitude loss can occur. Analyzing the results of the clustering concerning trips, it can be seen that the most common microsegments occurring in a trip are microsegments with $ci_m = 2$ and $ci_m = 5$ with over 30% for some trips. Those microsegments correspond to segments with only moderate values for altitude gain and loss. This aligns with the topography of Zurich, as the city center is predominantly flat, while the surrounding areas feature hilly terrain [14]. This also corresponds to the observation that all clusters contain less than 9% of microsegments from $ci_m = 3$ and $ci_m = 4$ which represent microsegments with high altitude difference.

The distribution of microsegments across trips is relatively balanced. One reason contributing to this finding is that trips have a duration of at least five minutes and can span over multiple hours and therefore increase the probability of many different microsegments being contained in one trip. Another reason might be that the dataset is limited to city buses within a single city, resulting in routes that are inherently similar. Because of the variables available and to mitigate the complexity of the clustering results only two or three variables, respectively, are considered in the first step. However, these variables are only capable of representing the actual driving sections to a limited extent.

6 Conclusion

The aim of this research is to develop an approach to effectively segmenting driving data and finding groups of similar driving sections using a clustering algorithm. For this purpose a dataset of electric city buses operating in Zurich is used. After data preparation, all data points are being clustered in a first cluster analysis. The categorization of the data points using the cluster results enables a segmentation of the dataset into microsegments. The dataset is aggregated on these microsegments and a second cluster analysis is conducted. The results show, that the characteristic patterns obtained in the first cluster analysis of points can also be seen in the second cluster analysis of microsegments. Additionally, the dataset is segmented into longer driving sections, referred to as trips, based on the duration of standstills between two consecutive driving sections. All microsegments are temporally assigned to the trips and the proportion of microsegments with the same cluster index are computed and used as an input variable for the cluster analysis on

trip-level. The results demonstrate minimal variation in the distribution of microsegments across the trips. Considering the setting of Zurich, the segmentation of microsegments and distribution of trips achieves plausible results.

In summary, this approach enables to identify representative driving sections by segmentation of driving data based on cluster analyses. Thereby a basis for generating representative driving cycles is provided, which are essential to fully exploit the capabilities of longitudinal dynamics simulations. In a further study, more variables, for example a congestion index could be taken into account, to potentially represent driving sections more accurately. Another subject of interest could be including driving data from other cities to gain information on the typical trips of city buses in general. Moreover, an investigation on data of intercity buses or coaches could discuss the transferability of the approach described in this paper to data of other bus classes.

References

[1] Montazeri-Gh M, Fotouhi A, Naderpour A. Driving patterns clustering based on driving features analysis. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*. 2011;225(6):1301–1317. doi: 10.1177/2041298310392599.

[2] Brady J, O'Mahony M. Development of a driving cycle to evaluate the energy economy of electric vehicles in urban areas. *Applied Energy*. 2016;177:165–178. doi: 10.1016/j.apenergy.2016.05.094.

[3] Langner J, Grolig H, Otten S, Holzäpfel M, Sax E. Logical Scenario Derivation by Clustering Dynamic-Length-Segments Extracted from Real-World-Driving-Data. In: *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*. INSTICC, SciTePress. 2019; pp. 458–467. doi: 10.5220/0007723304580467.

[4] Chetouane N, Klampfl L, Wotawa F. Extracting information from driving data using k-means clustering. In: *Proceedings - SEKE 2021*, Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE. Knowledge Systems Institute Graduate School. 2021; pp. 610–615. doi: 10.18293/SEKE2021-118.

[5] Arthur D, Vassilvitskii S. k-means++: the advantages of careful seeding. In: *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. 2007; pp. 1027–1035.

[6] Lloyd S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*. 1982; 28(2):129–137. doi: 10.1109/TIT.1982.1056489.

[7] Berzi L, Delogu M, Pierini M. Development of driving cycles for electric vehicles in the context of the city of Florence. *Transportation Research Part D-transport and Environment*. 2016;47:299–322. doi: 10.1016/j.trd.2016.05.010.

[8] Rousseeuw P. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65. *Journal of Computational and Applied Mathematics*. 1987; 20:53–65. doi: 10.1016/0377-0427(87)90125-7.

[9] Davies DL, Bouldin DW. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1979;PAMI-1(2):224–227. doi: 10.1109/TPAMI.1979.4766909.

[10] Caliński T, Harabasz J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*. 1974;3(1):1–27. doi: 10.1080/03610927408827101.

[11] Chetouane N, Wotawa F. On the application of clustering for extracting driving scenarios from vehicle data. *Machine Learning with Applications*. 2022; 9:100377. doi: 10.1016/j.mlwa.2022.100377.

[12] Widmer F, Ritter A, Onder CH. ZTBus: A large dataset of time-resolved city bus driving missions. *Scientific Data*. 2023;10(1). doi: 10.1038/s41597-023-02600-6.

[13] Valhalla contributors. Valhalla: Open Source Routing Engine for OpenStreetMap. `https://github.com/valhalla/valhalla`. Accessed: 2024-11-17.

[14] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`. 2017. Accessed from Geofabrik: 2024-11-22.

[15] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825–2830.

Acknowledgement

This research is part of the project "HyCo", which is funded by the Federal Ministry for Digital and Transport under the grant number 03B10305B.

Systemidentifikation eines omnidirektionalen Versuchsfahrzeug für die modellbasierte Funktionsauslegung

Björn Carstens[1*], Marian Göllner[1], Taihao Li[1], Xiaobo Liu-Henke[1]

[1]Fachgruppe Regelungstechnik und Fahrzeugmechatronik, Hochschule Ostfalia, Salzdalumer 46/48, 38302 Wolfenbüttel, Deutschland; *bj.carstens@ostfalia.de*

Kurzfassung. Die Systemidentifikation ist eine essenzielle Methode in der Regelungstechnik, die es ermöglicht, mathematische Modelle dynamischer Systeme zu erstellen und zu validieren. In diesem Beitrag wird ein detaillierter Überblick über die Prinzipien und Methoden der Systemidentifikation gegeben, mit besonderem Fokus auf die Validierung der Übertragungsfunktion eines omnidiektionalen Versuchsfahrzeugs mit Killough Fahrwerk. Besonderes Interesse besteht an der Identifikation der Verkopplungsstrukturen des überaktuierten Systems. Das identifizierte Modell bildet die Grundlage für die modellbasierte Funktionsauslegung.

Einleitung

Der multidisziplinäre Bereich der Systemidentifikation , der sich mit der Entwicklung mathematischer Modelle für dynamische Systeme auf der Grundlage experimentell ermittelter Daten befasst, spielt eine zentrale Rolle in der Ingenieurwissenschaft. Dieser Prozess wird in vielen Anwendungsbereichen, wie der Regelungstechnik, Signalverarbeitung und im Maschinenbau, genutzt [1]. Die Fähigkeit, präzise Modelle zu entwickeln, ermöglicht es Ingenieuren und Wissenschaftlern, das Verhalten komplexer Systeme zu verstehen, zu analysieren und zu kontrollieren.

Die grundlegende Idee der Systemidentifikation besteht darin, ein Modell zu finden, das das Verhalten eines Systems auf der Grundlage von Eingangs- und Ausgangsdaten beschreibt. Ein dynamisches System kann durch verschiedene mathematische Modelle dargestellt werden, darunter Differenzialgleichungen, Übertragungsfunktionen und Zustandsraumdarstellungen. Diese mathematischen Modelle ermöglichen es, das zeitliche bzw. frequenzabhängige Verhalten eines Systems zu beschreiben und Vorhersagen über dessen zukünftige Zustände zu treffen. Die Erfassung relevanter Daten ist der erste und einer der wichtigsten Schritte im Prozess der Systemidentifikation. Daten können durch experimentelle Verfahren, die gezielte Experimente zur Erzeugung aussagekräftiger Datensätze umfassen, oder durch passives Beobachten, bei dem vorhandene Daten aus dem Betrieb des Systems genutzt werden, gesammelt werden. Ziel dieser Arbeit ist es, die Grundlagen der Systemidentifikation darzulegen und eine Methodik zur Validierung von Modellen zu entwickeln, die auf experimentellen Daten basieren.

Hierbei wird eine im folgenden dargestellte Methodik genutzt und sukzessiv an einem in Kapitel 4.1 dargestellten Versuchsfahrzeug durchlaufen um die Verifikation und Validierung der Ergebnisse an einem realen System in Kombination mit einem iterativ zeitgleich aufgebauten Modell durchführen zu können.

1 Methodik

Der durchgängig modellbasierte und verifikationsorientierte Funktionsentwurf und die Absicherung vernetzter mechatronischer Systeme nach [2] hat sich in zahlreichen Anwendungen in Forschung und Industrie als zeit- und kosteneffizient erwiesen. Abbildung 1 zeigt den modellbasierten mechatronischen Entwicklungskreislauf, der zur Absicherung Model-in-the-Loop (MiL)-, Software-in-the-Loop (SiL)- und Hardware-in-the-Loop (HiL)-Simulationen sowie die Echtzeitrealisierung durch Prototypen umfasst.

Der Entwurfsprozess beginnt mit der Modellbildung basierend auf dem realen System, welches gemäß der Anforderungen reduziert bzw. vereinfacht wird, sodass sich zunächst ein physikalisches Modell ergibt. Dieses wird mithilfe physikalischer Gesetzmäßigkeiten in ein mathematisches Modell überführt, welches wiederum

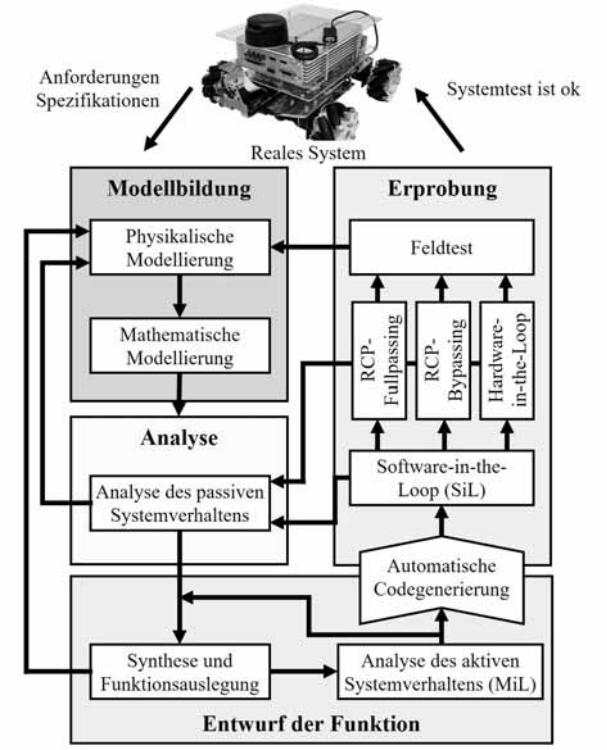


Abbildung 1: Mechatronischer Entwicklungskreislauf.

bspw. in Form von Signalflussplänen im Rechner abgebildet und mithilfe von CAE-Werkzeugen und entsprechender Numerik simuliert werden kann. Der Modellbildungsprozess umfasst Messungen am realen System. Dabei werden die Parameter des mathematischen Modells identifiziert und die Simulation validiert. Eine anschließende Analyse der Simulationsergebnisse lässt Schlussfolgerungen über die grundlegenden statischen und dynamischen Eigenschaften des realen Systems zu, auf dessen Grundlage die Konzeption der Funktionen erfolgt. Es werden sowohl die Funktionsarchitektur als auch Ansätze zu dessen Auslegung und Optimierung festgelegt. Komplexe Funktionen werden zudem nach dem verallgemeinerten Kaskadenprinzip in hierarchische Teilfunktionen zerlegt um die Funktionskomplexität zu reduzieren und so den Auslegeprozess handhabbar zu machen. Dazu werden bereits in einem frühen Entwicklungsstadium Hard- und Softwareanforderungen berücksichtigt und Schnittstellen für die funktionsübergreifende Kommunikation definiert. Der Erprobungsprozess wird parallel zur Entwicklung durchgeführt. Wenn eine Teilfunktion entwickelt worden ist, wird diese getestet und analysiert.

In dieser Arbeit werden zunächst mit Hilfe dieser Methode Anforderungen bestimmt. Durch die Analyse des physikalischen Modells, sowie der Mathematik, wird die Funktion modelliert. Jeder Teil der Funktion wird entsprechend getestet.

2 Stand des Wissens

Unter Systemidentifikation versteht man die Erstellung eines Modells für ein dynamisches System, das die Beziehung zwischen Eingangs- und Ausgangsdaten beschreibt. In diesem Beitrag werden die theoretischen Grundlagen der Systemidentifikation behandelt, mit besonderem Fokus auf die Identifikation mittels linearer Modelle. Es werden Methoden zur Bestimmung nichtparametrischer Modelle vorgestellt, wie beispielsweise die Ermittlung von Übertragungsfunktionen aus Sprungantworten und die Bestimmung des Frequenzgangs. Bei parametrischen Modellen ist die Parameterschätzung ein wesentlicher Bestandteil der Systemidentifikation. Die theoretischen Grundlagen der Parameterschätzung werden detailliert erklärt in [3, 4]. Anregungssignale spielen hierbei eine zentrale Rolle, da sie das Verhalten des zu regelnden Systems aufzeigen. In der Regelungstechnik ist es von besonderer Bedeutung, dass das Anregungssignal das relevante Frequenzspektrum abdeckt, um genaue Modelle zu entwickeln und eine effektive Regelung zu gewährleisten. Verschiedene Arten von Anregungssignalen, wie Impulse, Sprünge, Sinusschwingungen und Pseudozufallsbinärsignale (PRBS), werden verwendet, um das Systemverhalten umfassend zu charakterisieren. Die präzise Modellbildung auf Basis dieser Signale ermöglicht es Ingenieuren, das Regelverhalten von Systemen zu optimieren und die Stabilität sowie die Leistungsfähigkeit der Regelkreise zu verbessern. [4]

Die Frequenzganganalyse ist eine zentrale Methode zur Identifikation und Validierung von Modellen in der Regelungstechnik. Sie umfasst die Analyse der Systemantwort auf sinusförmige Eingangssignale bei verschiedenen Frequenzen. Diese Methode beschreibt, wie das System auf unterschiedliche harmonische Eingangsgrößen reagiert, wobei Amplituden- und Phasenverschiebungen gemessen und analysiert werden. Ein Bodediagramm stellt die Ergebnisse der Frequenzganganalyse grafisch dar. Darin werden die logarithmische Amplituden- und Phasenverschiebung als Funktionen der Frequenz dargestellt. Dadurch können nicht nur die Stabilität und die Dynamik eines Systems beurteilt, sondern auch Schwachstellen im Modell identifiziert wer-

den. Die praktischen Schritte einer Frequenzganganalyse umfassen:

1. Anlegen von Eingangssignalen an das System.
2. Messen der jeweiligen Ausgangssignale.
3. Berechnung der Amplitudenverhältnisse und Phasenverschiebungen zwischen Ein- und Ausgangssignalen.
4. Darstellung der Ergebnisse in Bodediagrammen.
5. Vergleich der experimentellen Frequenzgänge mit den theoretischen Modellen.

Durch den systematischen Vergleich von experimentellen und theoretischen Datensätzen kann die Genauigkeit der Modelle validiert und mögliche Modellierungsfehler identifiziert werden. [5, 6, 7]

3 Konzeption

Der Prozess der Systemidentifikation beginnt mit der Modellbildung, bei der das Verhalten des Systems durch physikalische Gesetze und Gleichungen beschrieben wird [8]. Anschließend werden Ein- und Ausgangssignale des Systems gemessen, die durch geeignete Anregungssignale erzeugt werden können, siehe Kap 2. Besonders für Fahrzeugsysteme eignen sich Signale mit einem breiten Frequenzspektrum [9]. Auf Basis der gemessenen Daten wird ein mathematisches Modell gewählt, häufig in Form von AR- (Autoregressive), MA- (Moving Average) oder ARMA-Modellen (Autoregressive Moving Average). Die Parameter des Modells werden mittels Verfahren wie der Methode der kleinsten Quadrate, der rekursiven Methode der kleinsten Quadrate oder der Vorhersagefehlermethode optimiert. Die Validierung erfolgt durch den Vergleich der Modellvorhersagen mit unabhängigen Datensätzen, um sicherzustellen, dass das Modell das Systemverhalten hinreichend genau abbildet. Herausforderungen wie Messfehler, Nichtlinearitäten und zeitliche Variationen des Systems erfordern ein ausgewogenes Verhältnis zwischen Modellkomplexität und Handhabbarkeit. Experimentelle Methoden basieren auf der Messung von Systemantworten auf definierte Testsignale, während theoretische Ansätze auf der Modellierung mittels physikalischer Gesetze beruhen. Ein wichtiger Schritt der Systemidentifikation ist die Frequenzgangmessung, die eine präzise Validierung der Übertragungsfunktion erlaubt. In unserem Fall wurde die Übertragungsfunktion $G(s)$, die das Verhältnis der Laplace-Transformierten des Ausgangs- zu Eingangssignalen beschreibt, wie folgt aufgestellt:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + b_1 s + \cdots + b_m s^m}{a_0 + a_1 s + \cdots + a_n s^n}$$

Zur Überprüfung der Übertragungsfunktion wurde das System harmonischen Eingangssignalen unterschiedlicher Frequenzen ausgesetzt, und die jeweiligen Ausgangssignale wurden gemessen. Die Frequenzgangmessung erfolgte mit einem Abacus 901, das präzise Amplituden- und Phasenänderungen analysiert. Die Validierung besteht im Abgleich der experimentellen Frequenzgangantwort mit der Modellvorhersage. Stimmt das gemessene Verhalten mit dem modellierten überein, gilt das Modell als bestätigt. Abweichungen können auf nicht-lineare Effekte, Messfehler oder unzureichende Modellannahmen hinweisen.

3.1 Kinematik

Das in Abbildung 2 dargestellte Mecanum-Rad zeigt die zur Beschreibung der Kinematik notwendigen Größen. Es wird die Winkelgeschwindigkeit ω dargestellt, welche im weiteren Verlauf auch häufig durch die Drehzahl n repräsentiert wird. Dazu steht im Schrägungswinkel α der Rollkörper die Geschwindigkeit des Rades in Sperrrichtung. Über den Radradius r und die Winkelgeschwindigkeit ω kann die Fahrzeuggeschwindigkeit V berechnet werden.

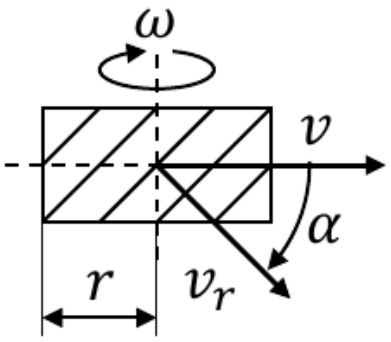


Abbildung 2: Kinematik eines Mecanum-Rads [10].

Ziel ist es, die in Gleichung 1 dargestellte Kinematikmatrix aufzustellen. Diese beschreibt die Raddrehzahlen n in Abhängigkeit von der Gesamtgeschwindigkeit des Versuchsfahrzeugs in seinen drei Freiheitsgra-

den v_{fx} ,v_{fy} und $\dot{\omega}_f$.

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix} = \underline{\underline{K}} \cdot \begin{bmatrix} v_{fx} \\ v_{fy} \\ \dot{\omega}_f \end{bmatrix} \tag{1}$$

3.2 Entkopplung

Die Entkopplung mechatronischer Systeme ist ein zentrales Konzept in der Regelungstechnik, das darauf abzielt, die Wechselwirkungen zwischen verschiedenen Komponenten eines Systems zu minimieren oder zu eliminieren. Dies ermöglicht eine unabhängige Steuerung der einzelnen Komponenten und verbessert die Gesamtleistung und Effizienz des Systems. In mechatronischen Systemen, die oft aus mechanischen, elektrischen und softwaregesteuerten Komponenten bestehen, kann die Entkopplung besonders herausfordernd sein, da diese Komponenten eng miteinander verknüpft sind. [11] Die Entkopplung erfolgt anhand der Kinematikmatrix $\underline{\underline{K}}$. Für ein Fahrzeug mit vier Mecanum-Rädern ergibt sich nach [12]:

$$\underline{\underline{K}} = \begin{bmatrix} \frac{\cos(\delta_1)}{\sin(\alpha_1)} & \frac{-\sin(\delta_1)}{\sin(\alpha_1)} & \frac{-\sin(\varepsilon_1)}{\sin(\alpha_1)} \cdot r_1 \\ \frac{\cos(\delta_2)}{\sin(\alpha_2)} & \frac{-\sin(\delta_2)}{\sin(\alpha_2)} & \frac{-\sin(\varepsilon_2)}{\sin(\alpha_2)} \cdot r_2 \\ \frac{\cos(\delta_3)}{\sin(\alpha_3)} & \frac{-\sin(\delta_3)}{\sin(\alpha_3)} & \frac{-\sin(\varepsilon_3)}{\sin(\alpha_3)} \cdot r_3 \\ \frac{\cos(\delta_4)}{\sin(\alpha_4)} & \frac{-\sin(\delta_4)}{\sin(\alpha_4)} & \frac{-\sin(\varepsilon_4)}{\sin(\alpha_4)} \cdot r_4 \end{bmatrix} \tag{2}$$

Mit dem Lauftonnenwinkel des Mecanumrades α_i, dem Positonswinkel der Radachse β_i zum Koordinatensystem des Fahrzeugs und dem Orientierungswinkel γ_i. Zusammen mit dem absoluten Abstand des Ursprungskoordinatensystems des Mecanumrades zum Ursprungskoordinatensystem des Fahrzeugs r_i ergeben sich die Polarkoordinaten ausgehend vom Koordinatenursprung.

$$\alpha_i = (-1)^i \cdot 45^\circ \tag{3}$$

$$\beta_i = \arcsin\left(\frac{b_i}{r_i}\right) - (i-1) \cdot 90^\circ \tag{4}$$

$$\gamma_i = \arcsin\left(\frac{b_i}{r_i}\right) + (i-1) \cdot 90^\circ \tag{5}$$

$$r_1 = \sqrt{b_i^{2^2} + b_n{}^2}, r_{i>1} = \sqrt{b_{i-1}^2 + b_i^2} \tag{6}$$

Die Konstruktionswinkel $\delta_i = \alpha_i - \beta_i - \gamma_i$ und $\varepsilon_i = \alpha_i - \beta_i$ ergeben sich aus diesen. Für das einsetzte Versuchsfahrzeug ergibt sich so die Kinematikmatrix $\underline{\underline{K}}$ zu:

$$\underline{\underline{K}} = \begin{bmatrix} 1 & -1 & -\frac{l_x+l_y}{2} \\ 1 & 1 & -\frac{l_x+l_y}{2} \\ 1 & -1 & \frac{l_x+l_y}{2} \\ 1 & 1 & \frac{l_x+l_y}{2} \end{bmatrix} \cdot \frac{1}{2 \cdot \pi \cdot r} \tag{7}$$

4 Verwendete Prüfstandsinfrastruktur

Die Identifikation eines Simulationsmodells erfordert wie bereits erwähnt zusätzlich Hard- und Software sowie entsprechende Anpassungen im Versuchsfahrzeug. Nachfolgend werde alle notwendigen Eingriffe vorgestellt. Zu ermitteln ist die Übertragungsfunktion der Geschwindigkeit des Versuchsfahrzeugs.

4.1 Aufbau des Versuchsfahrzeug

Das Versuchsfahrzeug bietet zahlreiche Funktionen, wobei hier nur die relevanten Systeme für diese Arbeit beschrieben werden. Weitere Details sind in [12] zu finden. Ein relevantes System ist der Mikrocontroller. Das genutzte Versuchsfahrzeug ist mit einem STM32H743 Board ausgestattet, das eine Dual-Core-Architektur aufweist. Der Mikrocontroller kann über eine LAN-Schnittstelle Anweisungen empfangen und Statusinformationen übermitteln. Das Antriebssystem des Versuchsfahrzeugs kann aus Swerve-, holonomen oder Mecanum-Systemen bestehen. Mecanum-Räder, entwickelt von Bengt Ilon, ermöglichen eine omnidirektionale Bewegung durch spezielle Rollen am Umfang des Rades, die eine unabhängige Steuerung der Bewegungsrichtung in alle Achsen erlauben. Swerve-Räder kombinieren Drehbewegungen und seitliche Verschiebungen, wodurch das Fahrzeug ebenfalls in alle Richtungen gesteuert werden kann, während holonome Räder eine ähnliche Bewegungsfreiheit bieten, jedoch auf eine ganz andere Art und Weise. Mecanum-, Swerve- und holonome Räder bieten Vorteile wie Platzersparnis, geringere Bau- und Instandhaltungskosten, hohe Präzision und den einfachen Austausch der Rollen. Mecanum-Räder bestehen aus schräg montierten Rollen, die es dem Fahrzeug ermöglichen, sich ohne Richtungseinschränkung zu bewegen. Swerve-Räder sind spezielle Lenk- und Fahrwerkskombinationen, die eine vollständige Bewegungsfreiheit in allen Richtungen ermöglichen, indem sie die Lenkung

und Geschwindigkeit jedes Rades unabhängig steuern. Holonome Räder bezeichnen Systeme, die es einem Fahrzeug ermöglichen, sich in alle Richtungen zu bewegen, ohne Einschränkungen hinsichtlich der Bewegungsrichtung. Sie sind ideal für die Intralogistik, Fahrerlose Transportsysteme und mobile Robotik [13]. Das Versuchsfahrzeug ist maßstäblich verkleinert, das Skalengesetz wird bei der Auswertung und Adaption berücksichtigt werden. In einem nächsten Schritt soll auf dem Raspberry Pi des Versuchsfahrzeugs ein Abstandsregler implementiert werden. Dieser Regler erfasst kontinuierlich die Distanz zu einem vorausfahrenden Objekt und berechnet darauf basierend eine geeignete Soll-Geschwindigkeit. Die berechneten Geschwindigkeitswerte werden anschließend an den Mikrocontroller des Fahrzeugs übertragen, der die Regelung der Antriebseinheit übernimmt.

4.2 Prüfsystem zur Messung

In Abbildung 3 wird das Prüfsystem der Messung dargestellt. Ein Host-PC mit der DP900 Software der Firma Data Physics wird über eine Ethernet-Verbindung mit dem Abacus 901 verbunden. Das Anregungssignal als Ausgang, sowie die zwei Eingänge werden mit dem Versuchsfahrzeug verbunden. Der Abacus 901 erzeugt analoge Anregungssignale, die durch die AD-Wandler der dSPACE Echtzeithardware in digitale Signale umgewandelt werden. Dieses digitale Anregungssignal wird sofort wieder zurück durch einen DA-Wandler an den Abacus zurückgeführt. Dieses Signal dient später als Referenzsignal. Dadurch ist es möglich, die möglichen Störeinflüsse, resultierend aus der AD- und DA-Wandlung, zu kompensieren. Der gewählte Signalpfad wurde bewusst so konzipiert, dass die Totzeiten, die durch die Signal- und Informationswandlungen auftreten, mit erfasst werden können. Das digitale Anregungssignal wird über die EtherNET-Schnittstelle des Mikrocontrollers als Soll-Geschwindigkeitssignal empfangen [12]. Dieser regelt nun mithilfe eines auf dem Mikrocontroller implementierten Geschwindigkeitsreglers die Soll-Geschwindigkeit ein. Die Antriebsmotoren sind mit Dreh-Encodern ausgestattet. Mit Hilfe dieser Encoder kann der Mikrocontroller die aktuelle Raddrehzahl und mit Hilfe der inversen Kinematik aus Gl.7 die Geschwindigkeit bestimmen. Die Geschwindigkeit wird dann über die EtherNet-Schnittstelle des Mikrocontrollers an die dSPACE Echtzeithardware zurückgesendet, die diese Geschwindigkeit als analoges Signal an den Abacus 901 übermittelt. Dieser Signalpfad soll identifiziert werden, da später der Raspberry Pi die Soll-Geschwindigkeit ebenfalls über die EtherNET-Schnittstelle an den Mikrocontroller senden wird. Der in Abbildung 3 gezeigte Aufbau stellt die Übertragung der Messungen in der Software dar. Bei der Bildung der Übertragungsfunktion wird nicht das Anregungssignal des Abacus als Systemeingang verwendet, sondern das durch die Echtzeithardware beeinflusste Signal.

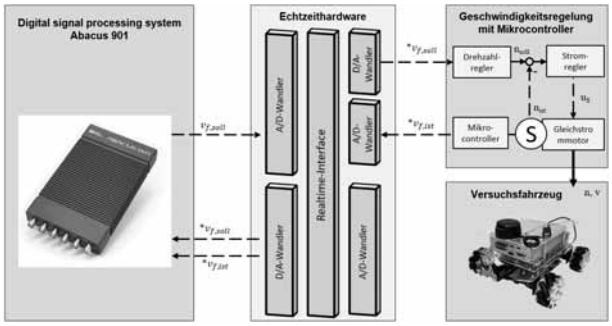


Abbildung 3: Übertragung der Anregung in der Software.

5 Ergebnisse

Die Ergebnisse der Frequenzgangmessungen wurden analysiert und mit dem mathematischen Modell verglichen. Anhand einer Parameteranpassung wurden die Parameter des Modells modifiziert und hinsichtlich der Abweichung zu des Messungen optimiert. So konnte die Übertragungsmatrix $\underline{\underline{G}}$ des Systems gefunden werden.

$$\underline{\underline{G}} = \begin{bmatrix} G_{v;x,x} & G_{v;x,y} & G_{v;x,\psi} \\ G_{v;y,x} & G_{v;v;y,y} & G_{v;y,\psi} \\ G_{v;\psi,x} & G_{v;\psi,y} & G_{v;\psi,\psi,} \end{bmatrix} \tag{8}$$

Die Hauptdiagonale beschreibt hierbei den direkten Wirkeinfluss des über die Kinematik zum jeweiligen Freiheitsgrad gestellten Anregegungssignals. Die Übertragungsfunktionen der Hauptdiagonalen wurden mit Hilfe der in GL. 7 dargestellte Transformationsmatrix über den Geschwindigkeitsregler angeregt und durch direkte Rückwärtstransformation der Messsignale auf den zu untersuchenden Freiheitsgrad bezogen.

Das in Abbildung 4 gezeigte Bodediagramm stellt die Übertragungsfunktion Geschwindigkeit in x-Richtung, bei einer Anregung in x-Richtung dar. Das Diagramm zeigt die Charakteristik eines PT1-Systems. Die Übertragungsfunktion $G_{v;x,x}$ ergibt sich entsprechend als Zeitverzögerungsglied erster Ordnung mit ei-

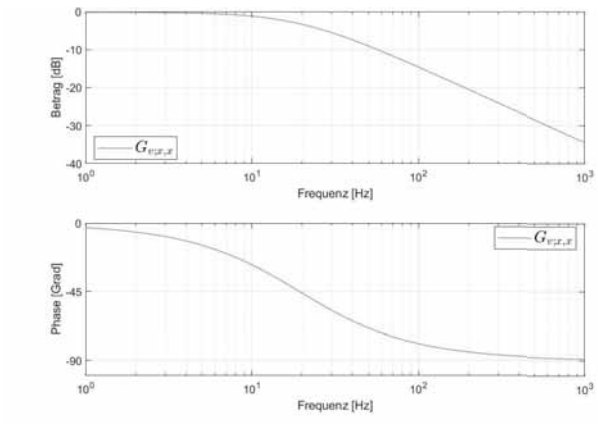


Abbildung 4: Übertragungsfunktion der Geschwindigkeit aus MatLab/Simulink.

ner Zeitkonstante von $\omega = 1,93 \cdot 10^1 \rightarrow T = \frac{1}{19,3} \approx 0,05$ und einem Verstärkungsfaktor $K = 1$.

Die Nebendiagonalen der Übertragunsgmatrix stellen die Verkopplung des Systems dar und wären bei einem relativ hohen Verstärkungsfaktor störend für die gewünschte Systemdynamik.

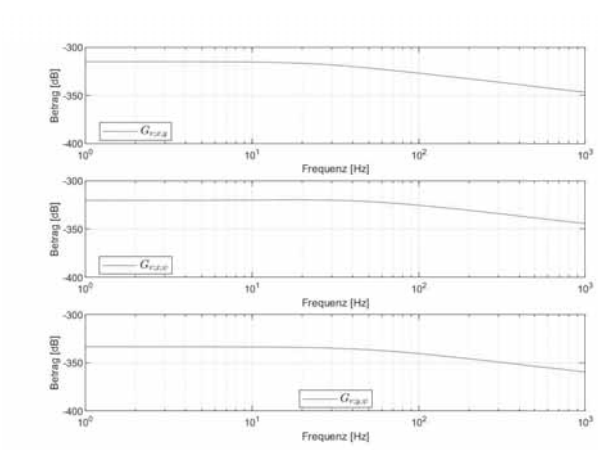


Abbildung 5: Übertragungsfunktionen einiger der Nebendiagonalen der Geschwindigkeit aus MatLab/Simulink.

Die in Abbildung 5 gezeigten Bodediagramme stellen die Übertragungsfunktionen der Nebendiagonalen dar. Das obere Bodediagramm ist die Übertragungsfunktion für die Geschwindigkeit in y-Richtung, bei einer Anregung in x-Richtung. Das mittlere Diagramm zeigt eine Anregung in x-Richtung und die Systemantwort in ψ, die Hochachse des Fahrzeugs. Das letzte Diagramm zeigt eine Anregung in y-Richtung und wieder die Systemantwort in ψ um die Hochachse des Fahrzeugs. Allen Antworten ist gemein, das der Verstärkungsfaktor der identifizierten PT1 Systeme bei $-320dB = 20\log_{10}(K) \rightarrow K \approx 10^{-16}$ liegt und somit der Einfluss der Verkopplung in Relation zu der gewünschten Systemdynamik der Hauptdiagonalen sehr gering ist. Auf diesem modifizierten Modell aufbauend können nun anhand des mechatronischen Entwicklungsprozesses die Reglersynthese und Optimierung durchlaufen werden.

Dazu wird ein Referenzmodell gleicher Ordnung und Struktur mit freien Parametern aufgebaut. Die Identifikation der optimalen Parameter erfolgt über die Minimierung des quadratischen Fehlers zwischen einem angenommen Übertragungsglied des Referenzmodels als Führungsübertragungsfunktion des separierten Übertragungsweges innerhalb der Übertragungsmatrix zu dem gemessenen Frequenzgang. Die Gesamtheit der Übertragungsglieder wird so in ihrem Verlauf dem gemessenen Frequenzgang optimal gefittet.

6 Zusammenfassung und Ausblick

Um ein Modell des dynamischen Verhaltens eines omnidirektionalen Intralogistikfahrzeug im verkleinerten Maßstab herzuleiten wurden die vorgestellten Methoden der Systemidentifikation angewandt. Die Omnidirektionalität des Fahrzeugs wird hier durch den Einsatz eines Killough-Fahrwerks mit vier Mecanum-Rädern realisiert.

Im Bereich der Kinematik beschreibt das Mecanum-Rad die Bewegung eines Fahrzeugs basierend auf der Winkelgeschwindigkeit und der Fahrzeuggeschwindigkeit. Die Entkopplung ist ein zentrales Konzept, um Wechselwirkungen zwischen Systemkomponenten zu minimieren. Das Versuchsfahrzeug ist zudem mit einem Mikrocontroller ausgestattet welcher die lokale Reglung des Fahrwerks zur Aufgabe hat. Die zur Messung zusätzlich benötigte Hardware beinhaltet überdies eine Verbindung zum Host-PC und dem Abacus 901 Signalanalysator; Notwendige Ergänzungen in der Software des Fahrzeugs umfassen die Verarbeitung und Rückübertragung der Soll- und Istwerte zu ebendiesen. Die Ergebnisse beinhalten die Darstellung von Übertragungsfunktionen für verschiedene Freiheitsgrade und

zeigen die Systemantworten auf unterschiedliche Anregungen.

Im weiteren Verlauf wird die selbe Methode zur Identifikation der Übertragungsfunktion bei einem größeren Versuchsfahrzeug durchgeführt. Der Versuchsträger AURONA dient als Plattform für aktuelle Forschungsvorhaben, daher ist es notwendig für dieses Fahrzeug grundlegende Fahrdynamikmodelle zu identifizieren und darauf basierend verschiedene autonome Fahrfunktionen weiter zu entwickeln, die mit Hilfe der kleinen Versuchsfahrzeuge erstmals umgesetzt wurden.

Danksagung

Gefördert vom Niedersächsischen Ministerium für Wissenschaft und Kultur unter Fördernummer ZN4068 im Niedersächsischen Vorab der VolkswagenStiftung.

Literatur

[1] Balakrishnan V. System identification: theory for the user (second edition). *Automatica*. 2002;38(2):375–378.

[2] Liu-Henke X. *Mechatronische Entwicklung der aktiven Feder-/Neigetechnik für das Schienenfahrzeug RailCab: Zugl.: Paderborn, Univ., Diss., 2005*, vol. 589 of *Fortschritt-Berichte VDI Reihe 12, Verkehrstechnik/Fahrzeugtechnik*. Düsseldorf: VDI-Verl., als ms. gedr ed. 2005.

[3] Bohn C. Systemidentifikation. In: *HÜTTE – Das Ingenieurwissen*, edited by Hennecke M, Skrotzki B, pp. 1–54. Berlin, Heidelberg: Springer Berlin Heidelberg. 2020;.

[4] Unbehauen H, ed. *Identifikation, Adaption, Optimierung: Mit 11 Tabellen*, vol. 3 of *Studium Technik*. Braunschweig: Vieweg, 6th ed. 2000.

[5] BLASS tH. Anwendung der Frequenzganganalyse beim praktischen Betrieb von Regelungseinrichtungen. *at - Automatisierungstechnik*. 1954;2(1-12):137–143.

[6] Pestel E, Kollmann E. Die Frequenzgangmethode. In: *Grundlagen der Regelungstechnik*, edited by Pestel E, Kollmann E, Regelungstechnik in Einzeldarstellungen, pp. 148–246. Wiesbaden and s.l.: Vieweg+Teubner Verlag. 1961;.

[7] Iyer SV. Frequenzganganalyse. In: *Digitales Filterdesign mit Python für Anwendungen in der Energietechnik*, edited by Iyer SV, pp. 125–161. Cham: Springer Vieweg. 2024;.

[8] Isermann R. *Identifikation dynamischer Systeme 1*. Berlin, Heidelberg: Springer Berlin Heidelberg. 1992.

[9] Isermann R. *Automotive Control*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2022.

[10] Dr-Ing X Liu-Henke, Sören Scherler, Marian Göllner, Johannes Maisik, Matthias Fritsch. Simulationsgestützte Konzeption der Antriebstopologie eines fahrerlosen Transportfahrzeugs. In: *Tagungsband ASIM Workshop STS/GMMS 2018*. ARGESIM. 2018; .

[11] Labisch D, Konigorski U. Verkopplungsbasierte Methode zur Entkopplung nichtlinearer Deskriptorsysteme. *at - Automatisierungstechnik*. 2014; 62(7):475–486.

[12] Jacobitz S, Göllner M, Zhang J, Liu-Henke X. Modellbasierte Entwicklung des Antriebsmoduls für vernetzte fahrerlose Transportfahrzeuge in einem cyber-physischen Labortestfeld;.

[13] Aktuelle Meldungen der Nabtesco Precision Europe GmbH.
URL `https://j2n.eu/r/wfeEa`

[14] Max van Haren, Lennart Blanken, Tom Oomen. Frequency Domain Identification of Multirate Systems: A Lifted Local Polynomial Modeling Approach. *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022;pp. 2795–2800.

[15] Tang W, Shi Z. Robust system identification of continuous-time model from frequency response function data. In: *2011 9th World Congress on Intelligent Control and Automation*. IEEE. 2011; pp. 661–665.

[16] Göllner M, Jacobitz S, Ferrara R, Liu-Henke X. Identifikation instabiler, unteraktuierter Systeme mit nicht-linearem dynamischen Verhalten. In: *ASIM SST 2024 Tagungsband Langbeiträge*, edited by Rose O, Uhlig T. ARGESIM Publisher Vienna. 4.– 6. September 2024; pp. 175–183.

Integration einer Funktion zur spurgenauen Lokalisierung für den Einsatz in einem realen Versuchsträger

Tianchen Hang[1*], Taihao Li[1], Marian Göllner[1], Xiaobo Liu-Henke[1]

[1]Fachgruppe Regelungstechnik und Fahrzeugmechatronik, Hochschule Ostfalia, Salzdalumer 46/48, 38302 Wolfenbüttel, Deutschland; *ti.hang@ostfalia.de*

Kurzfassung. Die Einführung digitaler Technologien treibt den gesellschaftlichen und wirtschaftlichen Wandel voran, insbesondere im Bereich der Mobilität, die durch autonomes Fahren in cyber-physischen Systemen (CPS) vorangetrieben wird. Für die Realisierung autonomer Fahrsysteme müssen Fahrzeuge in der Lage sein, sowohl ihre eigene Position als auch die anderer Verkehrsteilnehmer in komplexen Straßenumgebungen spurgenau zu lokalisieren. In dieser Arbeit werden ein Spurerkennungsmodell und ein Objekterkennungsmodell auf einem realen Versuchsträger implementiert. Diese Modelle ermöglichen es dem Fahrzeug, Straßeninformationen und andere Verkehrsteilnehmer präzise zu identifizieren. Die entwickelten Funktionen werden anhand eines real aufgezeichneten Videos validiert.

Einleitung

Gegenwärtig durchlaufen Gesellschaft und Wirtschaft einen tiefgreifenden Wandel, der durch den Einsatz neuer digitaler Technologien vorangetrieben wird. Eine Schlüsseltechnologie für die Mobilität der Zukunft stellt das autonome Fahren in vernetzten CPS-Umgebungen dar [1]. Aus [2] ist bewusst, dass hochpräzise Navigation und Lokalisierung eine der Schlüsseltechnologien für autonome Fahrzeuge sind, wobei die spurgenaue Lokalisierung eine Grundvoraussetzung für diese Lokalisierung darstellt. Eine weit verbreitete Methode zur spurgenauen Positionierung ist die Verwendung von GNSS Echtzeit-Kinematik (RTK), die jedoch häufig aufgrund schlechter Satellitengeometrie ausfällt und zu erheblichen Fehlern führt.

Das Hauptziel dieser Arbeit ist die Integration einer Funktion zur spurgenauen Lokalisierung in einen realen Versuchsträger. Die Lokalisierungsfunktion umfasst zwei wesentliche Komponenten: die Spurerkennung und die Objekterkennung. Zur Validierung der entwickelten Funktion werden reale, aufgezeichnete Videodaten verwendet, die Szenarien aus realen Straßenumgebungen repräsentieren. Die Validierung zielt darauf ab, die Robustheit und Genauigkeit der Funktion unter praxisnahen Bedingungen zu überprüfen. Die Integration und Validierung in einem realen Versuchsträger sind von zentraler Bedeutung, um die praktische Anwendbarkeit und Zuverlässigkeit der entwickelten Lokalisierungsfunktion zu gewährleisten.

Der weitere Aufbau dieser Arbeit ist wie folgt gegliedert: In Abschnitt 1 wird der strukturierte, modellbasierte und verifikationsorientierte Entwicklungsprozess des Rapid Control Prototyping (RCP) detailliert beschrieben. Abschnitt 2 behandelt die aktuellen Fortschritte und Herausforderungen in den relevanten Technologien. Die Konzeption der Lokalisierungsfunktion wird in Abschnitt 3 vorgestellt, gefolgt von der Integration dieser Funktion in einen realen Versuchsträger in Abschnitt 4. Die Ergebnisse der spurgenauen Lokalisierung und deren Validierung auf Basis realer aufgezeichneter Videodaten werden in Abschnitt 5 analysiert. Abschließend werden in Abschnitt 6 die wichtigsten Erkenntnisse zusammengefasst und ein Ausblick auf zukünftige Forschungsarbeiten gegeben.

1 Methodik

Der durchgängig modellbasierte und verifikationsorientierte Funktionsentwurf und die Absicherung vernetzter mechatronischer Systeme nach [3] hat sich in zahlreichen Anwendungen in Forschung und Industrie als zeit- und kosteneffizient erwiesen. Abbildung 1 zeigt den modellbasierten mechatronischen Entwicklungskreislauf, der zur Absicherung Model-in-the-Loop (MiL)-, Software-in-the-Loop (SiL)- und Hardware-in-

the-Loop (HiL)-Simulationen sowie die Echtzeitrealisierung durch Prototypen umfasst.

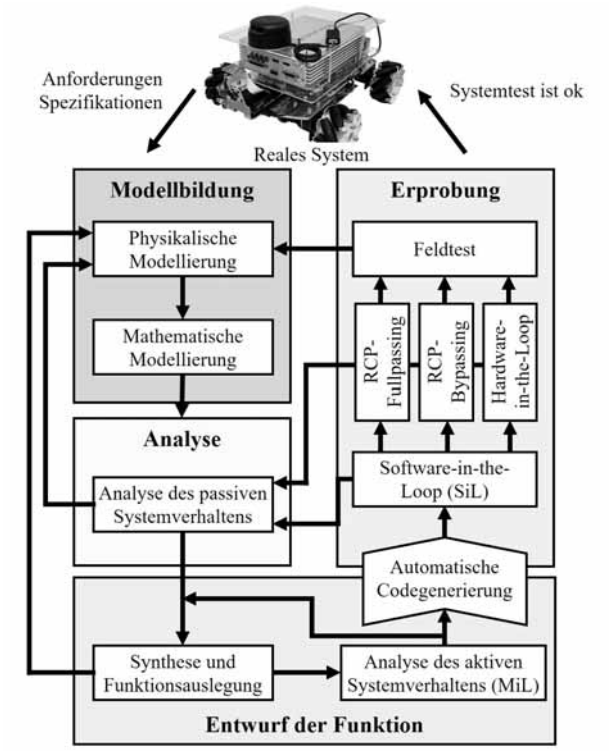


Abbildung 1: Mechatronischer Entwicklungskreislauf.

Der Entwurfsprozess beginnt mit der Modellbildung basierend auf dem realen System, das gemäß den Anforderungen reduziert und vereinfacht wird, um zunächst ein physikalisches Modell zu erstellen. Dieses wird unter Anwendung physikalischer Gesetzmäßigkeiten in ein mathematisches Modell überführt. Das mathematische Modell wird anschließend z. B. in Form von Signalflussplänen im Rechner abgebildet und mithilfe von CAE-Werkzeugen und numerischen Methoden simuliert. Dabei werden durch Messungen am realen System die Parameter des Modells identifiziert und die Simulation validiert. Die Analyse der Simulationsergebnisse ermöglicht Rückschlüsse auf die statischen und dynamischen Eigenschaften des realen Systems und bildet die Grundlage für die Konzeption der Funktionen.

Hierbei werden die Funktionsarchitektur sowie Ansätze zur Auslegung und Optimierung festgelegt. Komplexe Funktionen werden nach dem Kaskadenprinzip in hierarchische Teilfunktionen zerlegt, um die Komplexität zu reduzieren und den Entwicklungsprozess zu vereinfachen. Bereits in einer frühen Phase werden Hard- und Softwareanforderungen berücksichtigt und Schnittstellen für die funktionsübergreifende Kommunikation definiert. Die Erprobung erfolgt parallel zur Entwicklung, wobei jede Teilfunktion nach ihrer Entwicklung getestet und analysiert wird.

In dieser Arbeit werden mit Hilfe dieser Methode zunächst die Anforderungen definiert. Anschließend wird die Funktion basierend auf der Analyse des physikalischen und mathematischen Modells modelliert. Jeder Funktionsteil wird separat getestet, um die Entwicklung schrittweise zu validieren.

2 Stand des Wissens

Real-Time Multisensor Applications (RTMaps) ist ein Multisensor-Software-Framework für Datenerfassung und -wiedergabe, Softwareentwicklung sowie Echtzeitausführung. Sie findet häufig Anwendung in automatisierten Fahrsystemen und fortschrittlichen Fahrerassistenzsystemen (ADAS), um Echtzeitdaten von Kameras, Radar, Lidar und Ultraschallsensoren effizient zu verarbeiten [4]. Ein herausragendes Merkmal von RTMaps ist seine Fähigkeit, Multisensorsysteme nahtlos zu integrieren. Durch die Bereitstellung einer einheitlichen Schnittstelle kann RTMaps simultan Daten von verschiedenen Sensoren verarbeiten, diese innerhalb von Millisekunden synchronisieren und fusionieren. Diese Echtzeitfähigkeit ist entscheidend für automatisierte Fahrsysteme, die schnell reagieren müssen, um die Sicherheit im Straßenverkehr zu gewährleisten.

Traditionelle Methoden der Spurenerkennung basieren auf hochspezialisierten, handgefertigten Merkmalen und Heuristiken, um Fahrspursegmente zu identifizieren. Beliebte Beispiele für solche handgefertigten Merkmale sind farbbasierte Merkmale [5], der Balkenfilter und Rippenmerkmale, die möglicherweise mit einer Hough-Transformation und Partikel- oder Kalman-Filtern kombiniert werden [6, 7, 8]. Nach der Erkennung der Spurliniensegmente werden Nachbearbeitungsverfahren verwendet, um Fehlerkennungen herauszufiltern und Segmente zu kombinieren, um die endgültigen Fahrspuren zu bilden. Im Allgemeinen sind diese traditionellen Methoden anfällig für Robustheitsprobleme aufgrund von Variationen der Straßenszene, die von solchen modellbasierten Systemen nicht einfach modelliert werden können. Die neuen Methoden besteht darin, tiefe Netzwerke statt der handgefertigten Merkmalserkennung zu verwenden, um dichte Vorhersagen zu lernen. CNN wird mit RANSAC-Algorithmus

kombiniert, um Fahrspuren anhand von Kantenbildern zu erkennen. CNN wird in dieser Methode nur bei komplexen Straßenszenen zur Bildverbesserung eingesetzt [9]. Ein Multi-Task Deep Convolutional Network, das sich auf die Ermittlung geometrischer Fahrspurattribute wie Lage und Ausrichtung konzentriert, wird zusammen mit einem Recurrent Neural Network (RNN) kombiniert, um die Fahrspure zu erkennen [10]. Das Modell LanetNet [11] zeichnet sich durch seine hohe Leistungsfähigkeit aus. Es ist ein End-to-End-Deep-Learning-Modell, das die gleichzeitige Erkennung und Segmentierung von Fahrspurlinien, ein Merkmalsextraktionsnetzwerk und ein integriertes Clusternetzwerk umfasst. Das Merkmalsextraktionsnetz verwendet eingefaltete Neuronales Netz, um Merkmale aus dem Bild zu extrahieren. Das Clusternetzwerk hingegen gruppiert die extrahierten Merkmale, um die verschiedenen Fahrspurlinien zu segmentieren. Zu den Hauptvorteilen zählen eine verbesserte Echtzeitleistung und eine erhöhte Robustheit. Im LaneNet-Modell werden häufig vortrainierte Modelle als Encoder eingesetzt. Diese vortrainierten Modelle bieten den Vorteil, die Merkmalsextraktion zu optimieren und den Trainingsprozess erheblich zu beschleunigen.

Die auf Deep Learning basierenden Objekterkennungsalgorithmen werden hauptsächlich in zwei Kategorien unterteilt: One-Stage- und Two-Stage-Verfahren. Das Two-Stage-Verfahren R-CNN [12], Fast R-CNN [13] und Cascade R-CNN [14] lösen dieses Problem, in dem sie die Bildregionen in kleine Regionen aufteilen und separat klassifizieren. Zuerst werden den Erkennungsbereich abgegrenzt, dann erfolgt die Merkmalsextraktion und Klassifizierung. R-CNN zeichnet sich durch die Verwendung von Faltungsnetzen zur Merkmalsextraktion aus, wodurch die Schwächen herkömmlicher Erkennungsmethoden ausgeglichen werden, und verwendet eine selektive Suche zur Abgrenzung vom Bereich von Interesse (ROI), wodurch der Rechenaufwand reduziert wird. Fast R-CNN baut auf R-CNN auf und verbessert die Erkennungsgeschwindigkeit durch die Einführung einer neuen Verlustfunktion und erreicht eine erste Implementierung der End-to-End-Erkennung [15]. Two-Stage-Verfahren zeichnen sich durch eine hohe Erkennungsgenauigkeit aus, sind jedoch langsamer und für Echtzeitanwendungen weniger geeignet. Die YOLO-Serie [16] folgt einem völlig anderen Ansatz als zweistufige Detektoren: Die One-Stage-Verfahren verzichten auf die Begrenzung des Erkennungsbereichs und liefern direkt die Objektkategorie-Wahrscheinlichkeiten sowie die Positionsdaten. Dies führt zu einer deutlich erhöhten Erkennungsgeschwindigkeit.

Für Algorithmen zur Zielverfolgung lassen sich die derzeit verwendeten Verfolgungsschritte in zwei Hauptteile unterteilen: 1. Bewegungsmodell und Zustandsschätzung, die hauptsächlich dazu dienen, die Position von Objekten in nachfolgenden Frames vorherzusagen; 2. die Assoziation neuer Frame-Erkennungsergebnisse mit der aktuellen Trajektorie. Die derzeit führenden Tracking-Algorithmen sind DeepSORT [17] und Bytetrack [18]. Ersterer ist eine Weiterentwicklung des SORT-Algorithmus und verwendet die Mahalanobis-Distanz und Erscheinungsinformationen, um die Tracking-Frames des aktuellen Frames mit den Erkennungs-Frames abzugleichen. Allerdings sind der Rechenaufwand und die Verzögerung relativ hoch, so dass es für Anwendungen, die eine sofortige Reaktion erfordern, ungeeignet sein kann. Bytetrack verfügt über eine starke Assoziationsstrategie, die durch ein Sekundärer-Matching von Trajektorien mit hohen und niedrigen Vertrauensrahmen sowohl die Genauigkeit beibehält als auch die Geschwindigkeit erhöht und eine gute Robustheit bietet.

3 Konzeption

Im vorliegenden Abschnitt wird eine Konzeption für die Integration einer Funktion zur spurgenaue Lokalisierung in einem realen Versuchsträger dargestellt. Abbildung 2 zeigt den realen Versuchsträger AURONA. AURONA ist ein Forschungsfahrzeug, das mit vier Radnabenantrieben und einem by-Wire-Steuerungssystem für Antrieb, Bremse und Lenkung ausgestattet ist. Die Echtzeitdatenverarbeitung wird durch den dSPACE Autera-Rechner und eine Scalexio-Autobox ermöglicht. Zur Umfelderfassung sind verschiedene Sensoren integriert, darunter LiDAR, Radar und hochauflösende Kameras. Ein GNSS-IMU-System liefert präzise Positions- und Bewegungsdaten, während 5G-V2X-Technologie die Kommunikation und Vernetzung fördert. AURONA dient als Plattform für die Erforschung autonomer Mobilität und unterstützt die Entwicklung und Erprobung innovativer Fahrtechnologien in realen Verkehrsszenarien. Abbildung 3 zeigt die Konzeption des der spurgenaue Lokalisierungsfunktion. Um eine spurgenaue Lokalisierung zu ermöglichen, muss die Funktion folgende Anforderungen erfüllen: Das Fahrzeug kann die Spurlinien erkennen und die Anzahl

der Fahrspuren bestimmen, auf denen es sich befindet. Das Fahrzeug kann andere Verkehrsteilnehmer erkennen und die Fahrspuren der anderen Verkehrsteilnehmer bestimmen.

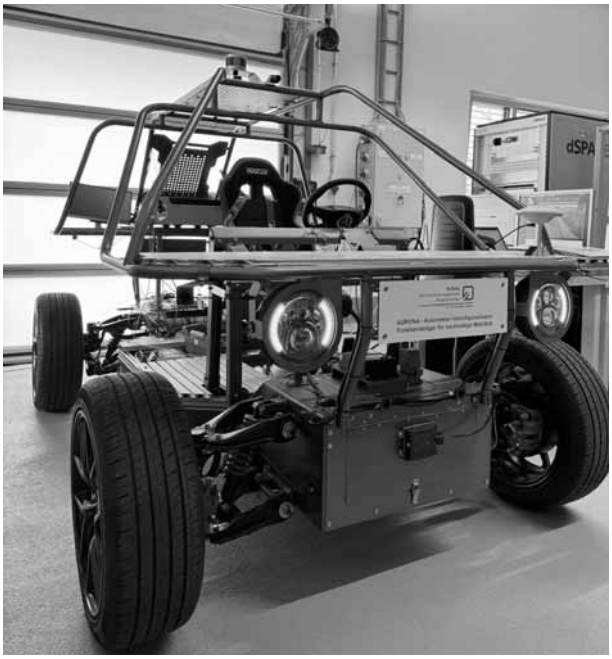


Abbildung 2: Realer Versuchsträger AURONA.

In dieser Arbeit wird ein Video V als eine zeitlich geordnete Sammlung von Einzelbildern betrachtet. Jedes Einzelbild wird als Matrix dargestellt und mit $\underline{\underline{B}}(t)$ bezeichnet. Die Bildrate des Videos wird mit r angegeben.

- Spurerkennung: Ziel dieses Teils ist es, die Spurlinien aus den Daten zu erkennen. Jede Spurtrennlinie $l_i(x)$ kann mit einem Polynom beschrieben werden (siehe Gleichung 1), wobei i die Nummer der Spurtrennlinie und x die Position entlang der Fahrtrichtung ist:

$$l_i(x) = a_n x^n + \ldots + a_1 x + a_0 \tag{1}$$

  Das Modell verwendet eine Kodierungs-/Dekodierungsstruktur. Der Kodierer extrahiert die wichtigen Merkmale, und der Dekodierer erstellt ein binäres Segmentierungsbild $\underline{\underline{B}}_b(t)$ und ein Instanzsegmentierungsbild $\underline{\underline{B}}_i(t)$. Das binäre Segmentierungsbild teilt die Pixel in Spurlinie und Hintergrund ein. Die Instanzsegmentierung unterscheidet die einzelnen Spurlinien. Um die Erkennung zu verbessern, wird das Rauschen aus den Bildern entfernt. Die bereinigten Bilder werden dann genutzt, um die Koordinaten der Spurlinienpunkte $C_i(t)$ zu berechnen. Schließlich werden die Punkte mit einem Polynom verbunden, um jede Spurlinie genau zu beschreiben.
- Objekterkennung und Lokalisierung: Ziel dieses Teils ist es, Fahrzeuge, Fußgänger oder andere Verkehrsteilnehmer zu erkennen und ihre Position zu bestimmen. Jedes erkannte Fahrzeug erhält eine eindeutige ID id. Zusätzlich werden die Position $\underline{P}_{id}(t)$ des Fahrzeugs und die Fahrspur $l_{id}(t)$, auf der es fährt, aufgezeichnet.

4 Entwurf

In diesem Kapitel wird der Entwurf der spurgenauen Lokalisierung unter Verwendung der Python Bridge in RTMaps vorgestellt. In dieser Arbeit wird das Design der Lokalisierung beschrieben, das durch die Integration der Python Bridge in RTMaps ermöglicht wird. Als Eingabe wird ein Video verwendet, das die Verkehrsinformationen der Fahrspuren aus der Ego-Perspektive zeigt. Wie bereits erwähnt, umfasst die Informationsauswertung sowohl die Identifizierung der Fahrspurlinien als auch die Lokalisierung, die durch die Python Bridge in RTMaps verarbeitet wird. Abbildung 4 zeigt den Entwurf der Integration.

4.1 Spurerkennung

In diesem Modell wird das Spurerkennungsmodell LaneNet verwendet. Die Aufgabe dieses Teils besteht darin, die Merkmale der Fahrspur zu extrahieren, was aufgrund der Encoder-Decoder-Struktur von LaneNet rechnerisch effizient ist. Nach der Dekodierung werden die binäre Abbildung und die Instanzabbildung erzeugt. Es erfolgt keine Anwendung von morphologischen Operationen wie Schließoperationen, da LaneNet primär auf die Segmentierung und die Instanzermittlung der Fahrspuren ausgerichtet ist. Die verarbeiteten binären und Instanzabbildungen dienen als Eingaben, um die Fahrspurmerkmale zu extrahieren. Die erhaltene Ausgabe sind die Koordinaten der Fahrspurpunktmengen. Das DBSCAN-Clustering wird auf die Koordinaten der Fahrspurlinienpunkte angewendet, um die Anzahl der Cluster, die Clusterbeschriftungen und die Clusterzentren zu ermitteln. Um die Genauigkeit der Fahrspurberechnung sicherzustellen, erfolgt eine Anpassung der Punktkoordinaten. Hierbei werden die Clu-

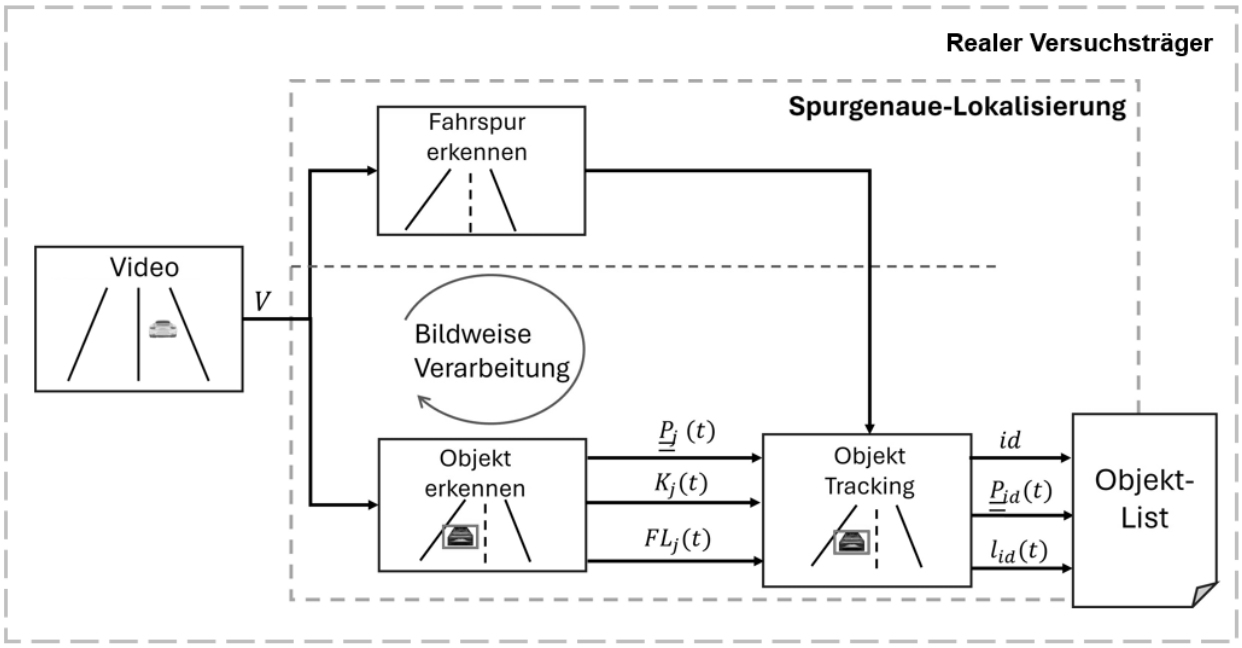


Abbildung 3: Konzeption der spurgenauen Lokalisierung.

sterergebnisse und die Koordinaten der Fahrspurpunkte kombiniert, um die spezifischen Punktkoordinaten jeder Fahrspur zu berechnen. Diese Punktmengen repräsentieren den Verlauf der Fahrspur und dienen als die endgültige Ausgabe des Modells.

4.2 Objekterkennung und Lokalisierung

Die Objekterkennung und Lokalisierung beginnt mit der genauen Erkennung von Fahrzeugen im Video. Aufgrund der hohen Anforderungen an Verarbeitungszeit und Genauigkeit wurde der YOLO-Algorithmus ausgewählt, da er Geschwindigkeit und Präzision optimal vereint. Nach der Erkennung wird jedem Fahrzeug eine Box $\underline{P}_j(t)$ zugeordnet, die vier Koordinatenpunkte enthält. Dabei steht j für die Nummer des erkannten Fahrzeugs. Die Box umgibt das Fahrzeug und ermöglicht die Bestimmung der Fahrzeugposition anhand des Mittelpunkts der unteren Seite. Zudem wird ein Konfidenzwert $K_j(t)$ berechnet, der die Wahrscheinlichkeit angibt, dass das erkannte Objekt der Fahrzeugkategorie $FL_j(t)$ entspricht. Dieser Wert dient sowohl zur Bestätigung der Erkennungsgenauigkeit als auch als entscheidender Parameter für die Zuordnung des Objekts an den Tracker, um eine präzise Verfolgung zu gewährleisten.

Nach der erfolgreichen Erkennung von Fahrzeu-

Prozedur 1 Erkennung der Fahrspur

1: INPUT: V
2: OUTPUT: $l_i(x)$
3: __init__: Initialisierung
4: Dynamic: Definition der Ein- und Ausgänge
5: Birth: Initialisierung des Komponentenstarts
6: Core: Hauptprozess
7: **for** $\underline{\underline{B}}(t)$ in V **do**
8: Kodieren: $\underline{\underline{F}}(t) = \text{LaneNet\_Encoder}(\underline{\underline{B}}(t))$
9: Dekodieren: $\underline{\underline{B}}^b(t), \underline{\underline{E}}(t) = \text{LaneNet\_Decoder}(\underline{\underline{F}}(t))$
10: Cluster: $C_i(t) = \text{DBSCAN}(\underline{\underline{E}}(t))$
11: Polynomanpassung: $l_i(x) = \text{Fit}(C_i(t))$
12: Ausgabe der Punktesätze: $l_i(x)$
13: **end for**
14: Death: Beende den Prozess

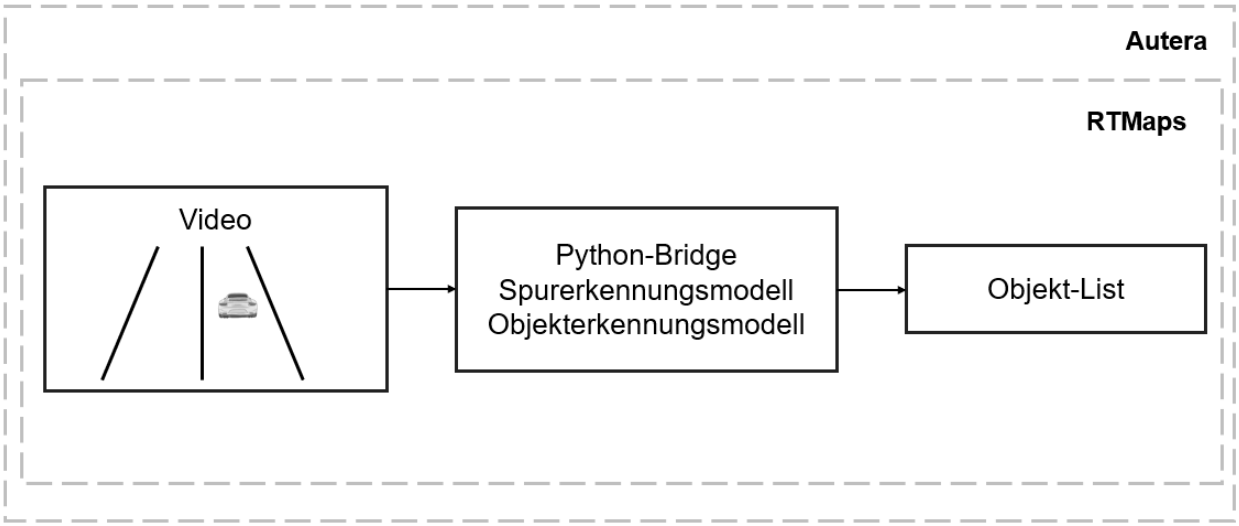


Abbildung 4: Konzeption der spurgenauen Lokalisierung.

gen wird in dieser Arbeit der ByteTrack-Algorithmus zur Verfolgung eingesetzt. ByteTrack weist jedem erkannten Fahrzeug eine eindeutige Identifikationsnummer (ID) zu, basierend auf der Position und den Konfidenzwerten des erkannten Fahrzeugs. Diese ID ermöglicht die Verfolgung desselben Fahrzeugs in aufeinanderfolgenden Videobildern. Zudem wird durch die Analyse der Fahrzeugposition in Bezug auf die Fahrspur ermittelt, auf welcher Spur sich das Fahrzeug befindet. ByteTrack speichert die Position jedes Fahrzeugs in jedem Bild, um den Fahrweg zu rekonstruieren. Dieses Verfahren verbessert die Kontinuität und Genauigkeit der Verfolgung und liefert wichtige Daten über die dynamischen Informationen der Fahrzeuge. Die Vorgehensweise ist in Prozedur 2 dargestellt.

5 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Spurgenaue Lokalisierung dargestellt. Diese Funktion wird mit einem aufgezeichten Video in RTMaps validiert.

Abbildung 5 zeigt die Validierung dieser Objekterkennungsfunktion. Die Fahrzeuge in dieser Umgebung werden erkannt. Das Fahrzeug in dem rechteckigen Box stellt das erkannte Fahrzeug dar. Auf dem Kasten befindet sich eine Nummer, die der Identifikationsnummer des erkannten Fahrzeugs bedeutet.

Abbildung 6 zeigt die erkannten Fahrspuren nach

Abbildung 5: Identifizierte Fahrzeuge

der Spurerkennung. Die Linien stellen die angepassten Fahrspuren dar, die mit den tatsächlichen Spurlienen im Bild übereinstimmen.

Abbildung 7 zeigt die Integartion von den identifizierten Fahrzeugen und den erkannten Fahrspuren.

Tabelle 1 zeigt die Lokalisierungsinformationen sowie die jeweilige Fahrspur, auf der sich das Fahrzeug befindet.

6 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden sowohl die Selbstlokalisierung auf Fahrspurebene als auch die Lokali-

Prozedur 2 Lokalisierung

INPUT: V, $l_i(x)$, r

OUTPUT: $t, id, \underline{P}_{id}(t), l_{id}(t))$

__init__: Initialisierung

Dynamic: Definition der Ein- und Ausgänge

Birth: Initialisierung des Komponentenstarts

Core: Hauptprozess

for $\underline{\underline{B}}(t)$ in V **do**

  $f = f + 1$

  $\underline{\underline{B}}(t) = addline(\underline{\underline{B}}(t), l_i(x))$

  $\underline{P}_j, K_j(t), FL_j(t) = YOLO(\underline{\underline{B}}(t))$

  $\underline{P}(t)^{j,B}$, $ID = Track(\underline{P}_j, K_j(t), FL_j(t))$

  $\underline{P}_{id}(t) = Box\_Mittelpunktder\_Unterseite(\underline{P}(t)^{j,B}, id)$

  if $\underline{P}_{id}(t)[x] > l_i^{-1}(\underline{P}_{id}(t)[y])$ **then**

    $l_{id}(t)$ ist i

  end if

  $Save(t, id, \underline{P}_{id}(t), l_{id}(t))$

end for

Death: Beende den Prozess

Abbildung 6: Erkannte Fahrspuren

sierung anderer Fahrzeuge auf Fahrspurebene in einem realen Versuchsträger entwickelt. Diese Arbeit besteht aus zwei Hauptaspekten: Erstens wird LaneNet verwendet, um die Fahrspuren zu identifizieren. Zum anderen wird YOLO eingesetzt, um Fahrzeuge auf der Straße zu erkennen und deren Position zu verfolgen. Die Funktion wurde auf einem realen Versuchsträger validiert, wobei ein vorab aufgenommenes Video als Eingabe in RTMaps verwendet wurde, um die Genauigkeit und Effizienz der Lokalisierung in einer realen Umgebung zu überprüfen.

Abbildung 7: Integration von Fahrsuprerkennung und Lokalisierung

	$\underline{P}_{id}$	l_{id}
id54	(566,99 , 362,96)	Fahrspur 1
id53	(953,83, 284,06)	Fahrspur 4
id66	(837,83, 279,95)	Fahrspur 3
id111	(803,43, 261,97)	Fahrspur 3

Tabelle 1: Ergebnisse von Ausgaben

Zukünftige Arbeiten werden sich auf die Optimierung der Straßenerkennungsfunktion zur Anpassung an verschiedene Umgebungen konzentrieren. Außerdem soll das Spurerkennungsmodell optiniert werden. Darüber hinaus wird die Fusion weiterer Sensoren wie LiDAR sowie die Übertragung der erkannten Informationen über V2X-Kommunikation an andere Verkehrsteilnehmer untersucht werden.

Danksagung

Gefördert vom Niedersächsischen Ministerium für Wissenschaft und Kultur unter Fördernummer ZN4172 im Niedersächsischen Vorab der VolkswagenStiftung.

VolkswagenStiftung Niedersächsisches Ministerium für Wissenschaft und Kultur

Literatur

[1] Francini M, Chieffallo L, Palermo A, Viapiana MF. Systematic Literature review on smart mobility: A framework for future “quantitative” developments. *Journal of Planning Literature*. 2021;36(3):283–296.

[2] Rabe J, Necker M, Stiller C. Ego-lane estimation for lane-level navigation in urban scenarios. 2016; pp. 896–901.

[3] Liu-Henke X. *Mechatronische entwicklung der aktiven Feder-, Neigetechnik für das schienenfahrzeug RailCab*. VDI Verlag. 2005.

[4] Katare D, El-Sharkawy M. Autonomous Embedded System Enabled 3-D Object Detector: (with Point Cloud and Camera). In: *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2019; .

[5] Chiu KY, Lin SF. Lane detection using color-based segmentation. In: *IEEE Proceedings. Intelligent Vehicles Symposium*. IEEE. 2005; .

[6] Loose H, Franke U, Stiller C. Kalman Particle Filter for lane recognition on rural roads. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2009; .

[7] Liu G, Wörgötter F, Markelić I. Combining Statistical Hough Transform and Particle Filter for robust lane detection and tracking. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2010; .

[8] Teng Z, Kong DJ, Kang DJ. Real-time lane detection by using multiple cues. In: *ICCAS*. 2010; .

[9] Kim J, Lee M. Robust Lane Detection Based On Convolutional Neural Network and Random Sample Consensus. In: *International Conference on Neural Information Processing*. 2014; .

[10] Li J, Mei X, Prokhorov D, Tao D. Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene. In: *IEEE Transactions on Neural Networks and Learning Systems*. 2017; pp. 690–703.

[11] Wang Z, Ren W, Qiu Q. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:180701726*;.

[12] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014; pp. 580–587.

[13] Girshick R. Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015; .

[14] Cai Z, Vasconcelos N. Cascade R-CNN: Delving Into High Quality Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018; .

[15] Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017; pp. 2117–2125.

[16] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016; pp. 779–788.

[17] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017; pp. 3645–3649.

[18] Zhang Y, Sun P, Jiang Y, Yu D, Weng F, Yuan Z, Luo P, Liu W, Wang X. Bytetrack: Multi-object tracking by associating every detection box. In: *European conference on computer vision*. Springer. 2022; pp. 1–21.

Fahrsimulator-in-the-Loop – Simulationsgestützte Fahrfunktionsentwicklung unter Einbezug menschlicher Verhaltensmuster

Paul Ole Flender[1], Xiaobo Liu-Henke[1], Marian Göllner[1]

[1]Institut für Mechatronik Ostfalia Hochschule für angewandte Wissenschaften, Salzdahlumer Str. 46/48, 38302 Wolfenbüttel, Deutschland;

Abstract. *Die Entwicklung hochautomatisierter Fahrfunktionen erfordert eine präzise Abstimmung zwischen technischer Leistungsfähigkeit und der Akzeptanz potenzieller Nutzer. Ein innovativer Ansatz zur Verbesserung der Entwicklungsprozesse ist die Driving-Simulator-in-the-Loop (DSiL)-Simulation. Diese Methode integriert einen modularen Closed-Loop-Fahrsimulator, der es ermöglicht, Fahrmanöver frühzeitig unter realitätsnahen Bedingungen zu testen und die Reaktionen der Nutzer in immersiven virtuellen Umgebungen zu analysieren.*

Das vorgestellte System kombiniert fortschrittliche Technologien wie virtuelle Realität, dynamische Fahrsimulation und Echtzeitdatenverarbeitung, um Fahrszenarien realitätsnah abzubilden und gleichzeitig Biosignale der Testpersonen zu messen. Parameter wie Herzfrequenz, elektrodermale Aktivität und Augenbewegungen werden erfasst, um psychophysiologische Zustände wie Stress, Angst oder Vertrauen zu analysieren.

Die Integration dieses Systems in den Entwicklungsprozess ermöglicht eine frühzeitige Evaluation und Optimierung von Fahrzeugkonzepten, was sowohl die Akzeptanz bei den Nutzern fördert als auch Entwicklungszeit und Kosten reduziert. Das DSiL-Konzept bietet damit eine innovative Grundlage für die Entwicklung zukunftssicherer, gesellschaftlich akzeptierter Fahrzeugtechnologien.

Einleitung

Das automatisierte Fahren ist ein bedeutender Trend in der Automobilindustrie. Mit der Entwicklung hochautomatisierter Fahrfunktionen, die es Passagieren ermöglichen, die Rolle des Fahrers vollständig abzugeben, rücken technologische Innovationen und gesellschaftliche Fragestellungen gleichermaßen in den Fokus. Ein zentraler Aspekt ist die Akzeptanz dieser neuen Technologien durch die NutzerInnen. Ohne Vertrauen und Akzeptanz der Mitfahrenden besteht das Risiko, dass die Funktionen abgelehnt und deaktiviert werden, wodurch alle potenziellen Vorteile des automatisierten Fahrens verloren gehen.

Um diese Herausforderungen zu bewältigen, wurde das Projekt VEAL (Virtuelle Entwicklung und Evaluation der Akzeptanz von automatisierten Level-4-Fahrzeugkonzepten) ins Leben gerufen. Ziel des Projekts ist die Entwicklung einer ganzheitlichen, integrativen Entwicklungsplattform, die alle relevanten Parameter zur Untersuchung und Förderung der Nutzerakzeptanz integriert. Ein zentraler Bestandteil dieser Plattform ist der Driving-Simulator-in-the-Loop (DSiL)-Prüfstand, der weiterentwickelt wird, um die Nutzerakzeptanz frühzeitig mittels geeigneter Nutzerdaten in den Entwicklungsprozess zu integrieren [1].

Die bestehende modellbasierte Funktionsentwicklung, die auf Model-in-the-Loop, Software-in-the-Loop und Hardware-in-the-Loop-Simulationen basiert, wird dabei durch DSiL-Simulationen ergänzt [1], [2]. Dieses Verfahren ermöglicht es, immersive Testszenarien zu schaffen, die sowohl objektive als auch subjektive Bewertungen von Fahrfunktionen erlauben. Bereits in frühen Entwicklungsstadien können so wichtige Erkenntnisse über Aspekte wie Komfort, Sicherheitsempfinden und Vertrauen in die automatisierten Fahrfunktionen gewonnen werden.

Das Projekt vereint die Expertise der TU Braunschweig (Institut für Fahrzeugtechnik, Institut für Konstruktionstechnik), TU Clausthal (Institute for Software and Systems Engineering) sowie der Ostfalia HaW (Institut für Mechatronik) Wolfenbüttel. Durch die Kombination modernster Technologien wie virtueller Simulationsumgebungen, künstlicher Intelligenz und dynamischer Fahrsimulatoren wird eine durchgängige Methodik entwickelt, die die Nutzerakzeptanz nachhaltig in den

Entwicklungsprozess integriert.

1 Motivation

Die Entwicklung hochautomatisierter Fahrfunktionen erfordert nicht nur technische Präzision, sondern auch die Akzeptanz zukünftiger Nutzer. Im Rahmen des VEAL-Projekts steht die virtuelle Entwicklung und Evaluation der Akzeptanz von Level-4-Fahrzeugkonzepten im Mittelpunkt. Ein kritisches Ziel besteht darin, Vertrauen in das Fahrzeug aufzubauen, indem realitätsnahe Erlebnisse in einer frühen Entwicklungsphase ermöglicht werden. Hierbei spielt der Closed-Loop-Fahrsimulator eine zentrale Rolle.

Der Fahrsimulator dient als Brücke zwischen der rein virtuellen Entwicklung und der realen Fahrzeugsimulation. Er ermöglicht es, immersive und reproduzierbare Testszenarien zu schaffen, die die Gemütszustände der Passagiere wie Stress, Angst oder Unaufmerksamkeit präzise analysieren können. Durch die Integration modernster Technologien wie virtueller Realität (VR), künstlicher Intelligenz (KI) und dynamischer Fahrzeugsimulation werden realistische Fahrsituationen simuliert, ohne dabei Sicherheitsrisiken oder hohe Kosten eines realen Prototyps in Kauf nehmen zu müssen.

Eine der größten Herausforderungen besteht darin, ein Gleichgewicht zwischen Sicherheit und Aufmerksamkeit der Passagiere zu schaffen. Der Fahrsimulator ermöglicht es, gezielt verschiedene Szenarien zu testen, um zu verstehen, welche Faktoren das Sicherheitsgefühl und die Akzeptanz beeinflussen. Gleichzeitig trägt er dazu bei, Schwachstellen in den Fahrfunktionen frühzeitig zu identifizieren und zu beheben.

Mit diesem Ansatz wird nicht nur die Akzeptanz hochautomatisierter Fahrfunktionen gefördert, sondern auch ein bedeutender Beitrag zur Ressourcenschonung und Kostensenkung geleistet [3]. Die Ergebnisse des Projekts bieten eine fundierte Grundlage für die Entwicklung zukunftssicherer und gesellschaftlich akzeptierter Fahrzeugkonzepte.

2 Stand des Wissens

2.1 Messung menschlicher Reaktion

Um den Gemütszustand einer Person subjektiv bewerten zu können, ist zunächst eine individuelle Anpassung der Messmethoden erforderlich. Eine zentrale physische Reaktion, die Aufschluss über den psychologischen Zustand einer Testperson gibt, ist die Herzfrequenz. Diese kann durch die Erfassung der elektrischen Herzerregung (Elektrokardiogramm, EKG) oder durch die Messung der peripheren Sauerstoffsättigung (SpO_2) mittels Pulsoximetrie bestimmt werden. Eine Analyse der EKG-Signale im Frequenzbereich zeigt, dass entspannte Personen eine niedrigere Herzfrequenz im Hochfrequenzbereich aufweisen, während gleichzeitig die Grundfrequenz höher ist als in einem angespannten Zustand [4]. Diese Werte werden auf Basis einer zuvor festgelegten individuellen Grundlinie interpretiert.

Ein weiterer Indikator ist die elektrodermale Aktivität (EDA). Ein kurzfristiger Abfall des Hautwiderstands deutet auf eine starke emotionale Reaktion, wie beispielsweise Stress, hin. Auch hier ist es notwendig, einen durchschnittlichen Hautwiderstand im Ruhezustand zu bestimmen, um eine verlässliche Basislinie zu schaffen. Abweichungen, die als kurze Spitzen auf dieser Basislinie auftreten, können als Zeichen emotionaler Belastung interpretiert werden [5].

Beide Parameter – Herzfrequenz und EDA – werden durch das vegetative Nervensystem gesteuert und entziehen sich der bewussten Kontrolle [6]. Besonders das Gleichgewicht zwischen dem sympathischen und dem parasympathischen Nervensystem spielt eine wichtige Rolle. Während das sympathische Nervensystem in Stresssituationen die Notfallreaktionen des Körpers aktiviert und die Handlungsbereitschaft erhöht, fördert das parasympathische Nervensystem in entspannten Zuständen Regeneration und Erholung. Ein biologisches Gleichgewicht besteht, wenn beide Systeme in einer ausgeglichenen Balance agieren [7].

Zusätzlich liefert die Atemaktivität wichtige Hinweise auf den Erregungszustand. Stresssituationen führen häufig zu einer erhöhten Atemfrequenz. Allerdings können sowohl bewusste Atemsteuerung als auch externe Faktoren, wie Luftqualität oder gesundheitliche Beeinträchtigungen, die Messung erschweren und die Ergebnisse verfälschen. Dadurch wird die Zuordnung emotionaler Zustände allein durch Atemparameter problematisch [8].

Auch Augenbewegungen können wertvolle Informationen liefern, insbesondere über die Auslöser emotionaler Erregung. Diese lassen sich entweder durch Eye-Tracking oder über elektrische Impulse der Augenmuskulatur mittels Elektrookulografie (EOG) erfassen [9].

Weitere Indikatoren wie die Konzentration von Glukose, Adrenalin oder Etilefrin im Blut können ebenfalls

Aufschluss über den psychologischen Zustand geben. Allerdings ist ihre Messung invasiv und nicht in Echtzeit möglich, was ihre Anwendung in Fahrsimulator [10].

2.2 Simulation der Fahrumgebung

Für die Bewertung emotionaler Zustände und subjektiver Wahrnehmungen im Zusammenhang mit hochautomatisierten Fahrfunktionen ist eine glaubwürdige Simulation des Fahrerlebnisses von entscheidender Bedeutung. Um aussagekräftige Daten zu erhalten, müssen die sensorischen Systeme der Testpersonen mit realistischen und synchronisierten Reizen angesprochen werden. Dabei sind insbesondere die audio-visuellen und vestibulären Eindrücke entscheidend, um ein hohes Maß an Immersion in der Simulation zu gewährleisten.

Der visuelle Eindruck bildet die Grundlage für eine überzeugende Fahrsimulation. Fortschritte in der Videospieltechnologie ermöglichen es heute, realistische visuelle Darstellungen von Fahrzeuginnenräumen, Außenumgebungen und sogar detaillierte Nachbildungen von Strecken zu generieren. Tools wie Unity und die Unreal Engine bieten effiziente Methoden zur schnellen Erstellung von visuellen Inhalten auf Basis von Entwicklungsdaten [11], [12] und [13]. Dennoch bleibt die korrekte Darstellung von Geschwindigkeit eine Herausforderung: Selbst bei physikalisch korrekten Simulationen wird die Geschwindigkeit oft langsamer wahrgenommen als sie tatsächlich simuliert wird [14]. Techniken wie Tunnel-Effekte oder Bewegungsunschärfe aus der Videospielindustrie können hier zur Verbesserung der Immersion beitragen [15].

Neben den visuellen Reizen spielen akustische Signale eine entscheidende Rolle. Durch Mehrkanal-Audiosysteme lassen sich Klangquellen präzise lokalisieren und in Relation zur Testperson setzen. Software zur Berechnung von Schallreflexionen und -projektionen sorgt dafür, dass Objekte in der virtuellen Umgebung realistisch positioniert werden, wobei auch Kopfbewegungen der Testperson durch Head-Tracking berücksichtigt werden.

Ein weiteres Kernelement ist der vestibuläre Eindruck, der das Gefühl von Fahrzeugbewegungen und Beschleunigungen vermittelt. Bewegungsplattformen können impulsartige Bewegungen und Positionierungsänderungen realitätsnah darstellen. Für konstante oder rampenartige Beschleunigungen über längere Zeiträume sind jedoch zusätzliche Simulationen erforderlich, beispielsweise durch Überlagerung der Bewegung mit Gravitationsbeschleunigung. Um kleinere, hochfrequente Bewegungen darzustellen, können Shaker direkt in das Mock-up oder an der Testperson angebracht werden.

Eine Herausforderung bei der Simulation ist die Vermeidung von Bewegungskrankheit, die auftreten kann, wenn Bewegungen und visuelle Eindrücke nicht synchronisiert sind oder der künstliche Horizont unnatürlich wirkt [14]. Die Isolierung der Testperson von der Außenwelt durch geschlossene Projekträume oder Head-Mounted Displays mit integrierten IMU-Sensoren und Head-Tracking trägt wesentlich dazu bei, solche Effekte zu minimieren [16].

Ein letzter Aspekt für eine vollständige Immersion ist die Integration von Rückstellkräften (Force Feedback) an den Mensch-Maschine-Schnittstellen wie Lenkrad und Pedalen. Diese Kräfte basieren auf Werten aus dem Fahrzeugdynamikmodell und können über Aktuatoren realistisch zurückgeführt werden, um ein authentisches Fahrerlebnis zu gewährleisten.

3 Methodik

Um eine Simulation zu entwickeln, die für Testpersonen glaubwürdig ist und realistische emotionale Reaktionen während des (automatisierten) Fahrens in einem frühen Entwicklungsstadium eines Fahrzeugs erfasst, ist ein hochkomplexes System erforderlich. Dieses System wird durch ein „Top-Down“-Ansatz in mehrere Subsysteme unterteilt, die aus mechatronischen Komponenten bestehen und als Funktionsmodule bezeichnet werden.

Die Entwicklung jedes dieser Funktionsmodule folgt dem „Bottom-Up“-Prinzip des mechatronischen Entwicklungszyklus. Eine ausführliche Darstellung der dabei eingesetzten Methoden findet sich in [1].

4 Konzeption

Um die Anforderungen an eine immersive Simulation zu erfüllen, die menschliche Gemütszustände während des (automatisierten) Fahrens in frühen Entwicklungsstadien bewertet, ist eine nahtlose Integration verschiedener Systeme erforderlich. Diese Systeme müssen synchron arbeiten, Daten austauschen und speichern können. Ein zentraler Bestandteil ist das Simulationssystem, das auf miteinander vernetzten Computern basiert und die Simulationsaufgaben parallel bearbeitet [3]. Zur Steigerung der Immersion wird dieses System um ein

Visualisierungssystem erweitert, das ein Head-Mounted Display (HMD) und einen zusätzlichen Visualisierungsrechner integriert.

Die folgenden Funktionen sind entscheidend für die Aufgaben des Simulationssystems:

- **Bewegungsstimulation:** Ein Hexapod mit linearen Aktuatoren simuliert Fahrzeugbewegungen und Beschleunigungen in sechs Freiheitsgraden. Die Steuerung erfolgt über ein lokales Kontrollsystem.
- **Mensch-Maschine-Schnittstelle (HMI):** Haptische Bedienelemente ermöglichen eine realistische Nachbildung der Fahrerumgebung, die durch spezielle HMI-Controller koordiniert wird.
- **Echtzeitdatenverarbeitung:** Mithilfe leistungsstarker Rapid Control Prototyping-Systeme (RCP) wird die Fahrzeugdynamik simuliert und eine Echtzeitkommunikation mit anderen Systemen ermöglicht.
- **Visualisierung:** Zwei dedizierte Visualisierungsrechner übernehmen die grafische Darstellung. Die gerenderten Bilder werden entweder auf Bildschirme oder ein Head-Mounted Display übertragen, während ein gespiegeltes Bild zusätzlich auf externe Monitore projiziert werden kann.
- **Koordination:** Ein zentraler Master-Computer synchronisiert alle Module und steuert die Kommunikation über eine Ethernet-Schnittstelle. Zudem ermöglicht er Benutzereingriffe während der Simulation.

Zusätzlich wird Sensortechnologie integriert, um die Reaktionen der Testpersonen zu erfassen. Dabei stehen drei Hauptparameter im Mittelpunkt, die Aufschluss über den Gemütszustand geben:

- **Optisches Oculogramm:** Die Muskelaktivität der Augen wird durch kamerabasierte Verfahren gemessen. Dies ermöglicht die Erkennung von Erregungs- und Angstzuständen und deren Rückführung auf visuelle Auslöser [17]. Aspekte wie Konvergenzbewegungen, reflexartige Anpassungen und sakkadische Bewegungen werden ebenfalls berücksichtigt [18].
- **Elektrodermale Aktivität:** Veränderungen der Hautleitfähigkeit und des elektrochemischen Potenzials, ausgelöst durch Schweißdrüsenaktivität, sind sensitive Indikatoren für psychophysische Aktivierung und Stress [5].
- **Elektrokardiogramm (EKG):** Durch die elektrische Erregung des Herzens können Herzrhythmen aufgezeichnet werden, die emotionale Zustände wie Stress und erhöhte Handlungsbereitschaft widerspiegeln [19].

Diese nicht-invasiven Messmethoden bieten eine solide Grundlage für die Analyse emotionaler Zustände. Durch die Kombination der gemessenen Daten wird die Genauigkeit der Erkennung verbessert, was eine differenzierte Bewertung ermöglicht. Die Simulations- und Messsysteme werden synchronisiert, und die gewonnenen Daten strukturiert gespeichert und ausgewertet, um präzise Rückschlüsse auf das Verhalten der Testpersonen zu ziehen.

5 Realisierung des Simulationssystem

Das Simulationssystem umfasst wesentliche Elemente wie Aktuatoren, die vestibuläre, auditive und visuelle Reize stimulieren. Der Gleichgewichtssinn der Testperson wird durch eine hexapodische Bewegungsplattform von E2M angeregt. Diese trägt eine einfache Mock-up-Struktur als Tragkonstruktion. Auf der Plattform sind Bildschirme zur Visualisierung und Lautsprecher für die Audiowiedergabe angeordnet, um einen Surround-Effekt rund um den Fahrersitz zu erzeugen (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**). Alternativ kann ein HTC Vive Pro Eye Head-Mounted Display (VR-Headset) genutzt werden, das mit einem Sensorcluster ausgestattet ist. Dieses misst die Beschleunigung und Winkelgeschwindigkeit des Kopfes und bestimmt durch Datenfusion dessen Position und Orientierung im Raum.

Um Bewegungen der Erregungsplattform zu kompensieren, wird die Position des Head-Mounted Displays (HMI) mittels eines Messgeräts mit zwei Infrarotsensoren erfasst. Die relativen Verschiebungen des Kopfes in der virtuellen Umgebung im Verhältnis zum simulierten Fahrzeugkoordinatensystem werden durch die Kombination folgender Daten berechnet: die aktuelle Position des Mock-ups (ermittelt durch die Odometrie der linearen Hexapod-Erregungseinheiten), die Messwerte des HMI-Sensorclusters und die absoluten Messwerte der Infrarotsensoren (siehe auch [16]).

Die Steuerung der Sensoren und Aktuatoren erfolgt zunächst lokal über untergeordnete mechatronische Steuerungssysteme. Die Kommunikation mit den übergeordneten Simulatorkomponenten läuft über einen

zentralen Router, der ein UDP-Ethernet-Protokoll verwendet. Zur Visualisierung werden zwei parallel arbeitende Computer eingesetzt: Ein Rechner versorgt die Bildschirme rund um den Fahrer, während der andere das HMI mit grafischen Daten beliefert, die durch eine

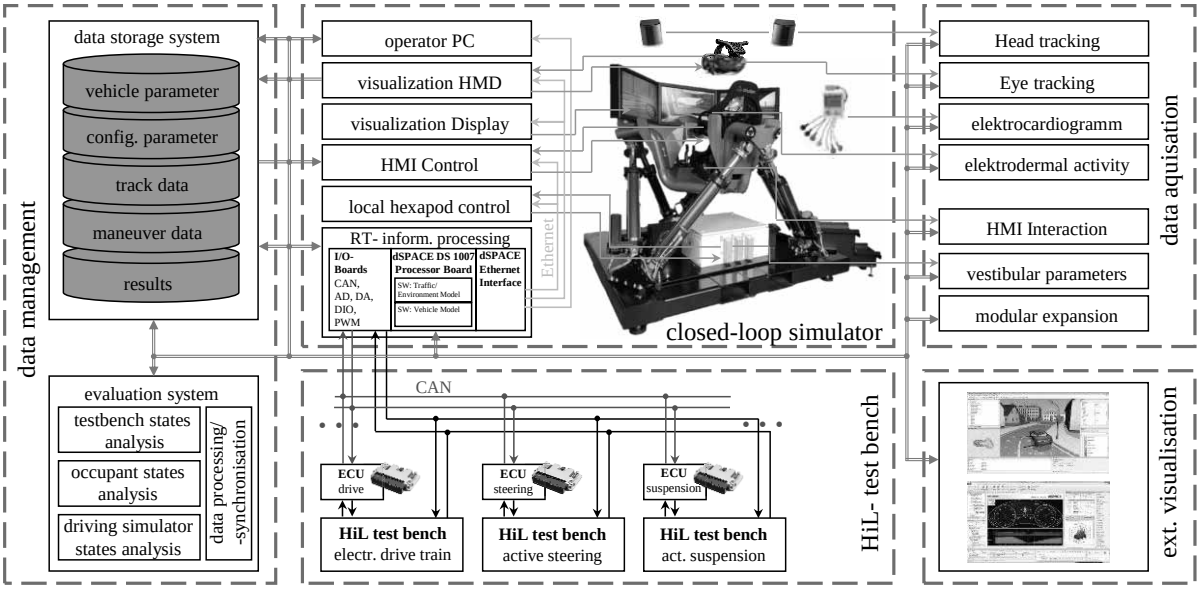


Figure 1: Struktur der Simulationsumgebung.

angepasste Unity Engine generiert werden. Diese Aufteilung gewährleistet die hohe Qualität der Visualisierung, die für eine immersive Simulation erforderlich ist.

Das Fahrzeugdynamikmodell wird auf modularer Echtzeithardware berechnet, die über flexible Schnittstellen (z. B. CAN-Bus) für externe HiL-Prüfstände verfügt. Dadurch können Fahrzeugdynamikmodelle direkt aus Matlab/Simulink kompiliert und im Simulator verwendet werden (weitere Details in [3]).

Ein Kontrollrechner verwaltet den gesamten Datenstrom, koordiniert die untergeordneten Rechner und sammelt alle relevanten Informationen. Diese werden in einem zentralen Datenmanagementsystem gespeichert, das als Datenbank des Simulators dient. Hier sind sowohl kompilierte als auch native Fahrzeugdynamikmodelle samt Fahrzeugparametern, Fahrmanövern und Streckendaten abgelegt. Visuelle Modelle von Fahrzeugen und Strecken, Kalibrierdaten der Hardware (z. B. des HMI oder des Hexapods) sowie Testergebnisse werden ebenfalls im Datenmanagementsystem gespeichert.

Zusätzlich umfasst das System ein Ergebnisauswertungsmodul, das die Simulationsdaten, Biodaten der Testperson und Zustandsdaten der Prüfstandskomponenten synchronisiert und analysiert. Die analysierten Ergebnisse können über das Ethernet-Netzwerk an externe Visualisierungssysteme gesendet und dort zu Demonstrationszwecken angezeigt werden.

6 Zusammenfassung

In dieser Arbeit wurde ein umfassender Ansatz zur Untersuchung des menschlichen Verhaltens in Fahrsimulationen vorgestellt. Der Fokus lag auf der Erzeugung realistischer virtueller Eindrücke, darunter die visuelle Simulation der Fahrumgebung, die Nachbildung der Fahrzeugbewegung durch vestibuläre Reize sowie die auditive und ortsbezogene Wahrnehmung von Objekten innerhalb der Simulation. Darüber hinaus wurde beschrieben, wie durch diese Reize ausgelöste emotionale Reaktionen von Testpersonen mithilfe von Biosignalen erfasst und analysiert werden können. Zu diesem Zweck wurde ein technisches System entworfen, das die gestellten Anforderungen erfüllt.

Das vorgestellte Projekt ist Bestandteil einer fortlaufenden Forschungsinitiative. Das Simulationssystem wird kontinuierlich erweitert, um neuen Anforderungen gerecht zu werden. Der aktuelle Entwicklungsstand zeigt eine vollständige Funktionalität aller Hardwarekomponenten und eine betriebsbereite Simulation, die sowohl auf Bildschirmen als auch mit einem VR-Headset genutzt werden kann. Inhalte für die Simulation können mit dem Tool RoadRunner erstellt und direkt im Simulator abgespielt werden. Derzeit erfolgt die Auswertung der erfassten Biosignale manuell, und ein Feedbacksystem ist noch nicht integriert. Zukünftig sollen jedoch automatische Auswertungsalgorithmen auf Basis heuristischer Methoden implementiert werden, um die Effizienz und

Präzision der Datenauswertung zu erhöhen.

Danksagung

Dieser Beitrag wurde im Rahmen des Forschungsprojektes VEAL (*Virtuelle Entwicklung und Evaluation der Akzeptanz von automatisierten Level-4-Fahrzeugkonzepten*) durch den Europäischen Fonds für regionale Entwicklung (EFRE) unter dem Förderkennzeichen ZW 7-87012033 gefördert. Die Verantwortung für den Inhalt liegt bei den Autoren.

7 References

[1] X. Liu-Henke, H. Tao, S. Jacobitz. Konzeption des Entwicklungsprozesses zur Berücksichtigung von Genderaspekten in fahrzeugmechatronischen Systemen. Heilbronn, Germany: Workshop of ASIM/GI; 2018.

[2] Schäuffele J. Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge Effizient Einsetzen. 4th ed. Wiesbaden: Springer Vieweg. in Springer Fachmedien Wiesbaden GmbH; 2010.

[3] X. Liu-Henke, M. Göllner, R. Buchta, F. Quantmeyer, H. Tao. Systemkonzept eines modularen HiL-Systems für modellbasierte Funktionsentwicklung fahrzeugmechatronischer Systeme. Ulm, Germany: ASIM/GI STS/GMMS Workshop 2017; 2017.

[4] Taelman J, Vandeput S, Spaepen A, van Huffel S. Influence of Mental Stress on Heart Rate and Heart Rate Variability. In: Magjarevic R, Nagel JH, Vander Sloten J, Verdonck P, Nyssen M, Haueisen J, editors. 4th European Conference of the International Federation for Medical and Biological Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009, p. 1366–1369.

[5] R. Mertens. Aussagekraft der elektrodermalen Aktivität in Laborexperimenten mit Schwerpunkt Lärm – Literaturstudie zu wichtigen Einflussfaktoren und gesundheitlichen Implikationen: Dissertation. Düsseldorf: Medizinischen Fakultät, Heinrich-Heine-Universität Medizinischen Fakultät, Heinrich-Heine-Universität; 2016.

[6] Hjemdahl P, Freyschuss U, Juhlin-Dannfelt A, Linde B. Differentiated sympathetic activation during mental stress evoked by the Stroop test. Acta Physiol Scand Suppl 1984;527:25–9.

[7] Hamer M, Steptoe A. Association between physical fitness, parasympathetic control, and proinflammatory responses to mental stress. Psychosom Med 2007;69(7):660–6. https://doi.org/10.1097/PSY.0b013e318148c4c0.

[8] Bernardi L, Wdowczyk-Szulc J, Valenti C, Castoldi S, Passino C, Spadacini G et al. Effects of controlled breathing, mental activity and mental stress with or without verbalization on heart rate variability. J Am Coll Cardiol 2000;35(6):1462–9. https://doi.org/10.1016/s0735-1097(00)00595-7.

[9] N. Döring JB. Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften. In: pp. 522-524.

[10] Wetherell MA, Crown AL, Lightman SL, Miles JNV, Kaye J, Vedhara K. The four-dimensional stress test: psychological, sympathetic-adrenal-medullary, parasympathetic and hypothalamic-pituitary-adrenal responses following inhalation of 35% CO2. Psychoneuroendocrinology 2006;31(6):736–47. https://doi.org/10.1016/j.psyneuen.2006.02.005.

[11] J. Murray. Unity Engine Polish and Optimization. In: Building Virtual Reality with Unity and SteamVR, 2020.

[12] Matej J. VIRTUAL REALITY AND VEHICLE DYNAMICS IN UNREAL ENGINE ENVIRONMENT. MM SJ 2016;2016(04):1141–4. https://doi.org/10.17973/MMSJ.2016_10_201688.

[13] A. Tarantola. Epic Games teases its new, nearly-photorealistic Unreal Engine 5. In: Engadget, Retrieved.

[14] B. Aykent, Z. Yang, F. Merienne, A. Kemeny. Simulation sickness comparison between a limited field of view virtual reality head mounted display (Oculus) and a medium range field of view static ecological driving simulator (Eco2). Driving Simulation Conference Europe 2014 Proceedings 2014:pp.65-71.

[15] J.-P.Guertin, M. McGuire, D. Nowrouzezahrai. A Fast and Stable Feature-Aware Motion Blur Filter. Computer Graphics Forum 2014 2014.

[16] B. Hartfiel, A. Kroys, N. Kruithof, R. Stark. Driving Simulator with VR Glasses for Evaluation of New Interior Concepts. In: ATZ - intenartional, pp. 16 – 23.

[17] M. Schwalm. Pupillometrie als Methode zur Erfassung mentaler Beanspruchungen im automotiven Kontext. Dissertation. Philosophische Fakultäten, Universität des SaarlandesPhilosophische Fakultäten, Universität des Saarlandes; 2008.

[18] B. Hesse. Wechselwirkung von Fahrzeugdynamik und Kfz-Bordnetz unter Berücksichtigung der Fahrzeugbeherrschbarkeit. Dissertation. Fakultät für Ingenieurwissenschaften, Universität Duisburg-Essen; 2011.

[19] C.V. Bauch. Minimalsensorisches Konzept zur multimodalen Fahrerzustandsüberwachung auf Basis physiologischer Daten - Validierung einer neuen Methode. Dissertation. Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn; 2010.

Dynamic System Simulation with Hybrid Thermal Storage Tanks

Sarah Jäger[1*], Peter Renze[1]

[1]Institute for Technology and Energy Economics, University of Applied Sciences Ulm, 89075 Ulm, Germany,
*sarah.jaeger@thu.de

This paper is focused on recent advancements in the integration of thermal storage systems with phase change material (PCM) packed beds for energy storage applications. A simulation environment is presented that combines detailed PCM modeling with dynamic load profiles from consumers and producers. By incorporating individual building and user parameters, this approach provides the foundation for simulating and analyzing application scenarios and their impact on the thermal performance of storage systems with different PCM configurations, compared to a conventional stratified water tank.

The model is based on a one-dimensional mathematical approach that represents temperature distributions in two zones (Jäger et al. 2024). It is adaptable to arbitrary tank geometries and various PCM layer configurations and is currently being expanded to investigate storage behavior under dynamic operating conditions.

The thermal dynamics within the water zone of the storage tank during charging or discharging processes are modeled as an open system with energy and mass exchange. This is described through differential equations for discrete tank layers, accounting for heat loss to the environment, conduction between layers using an effective conductivity, mass flow during charging and discharging processes, and heat transfer between water and PCM capsules. For the PCM, a temperature-dependent heat transfer coefficient between the water and the PCM capsules is included, and an enthalpy model is applied to describe its behavior during phase change. Zones within the tank are determined based on PCM placement and inlet and outlet locations, and if no PCM is present, the corresponding heat transfer term is omitted. A schematic representation visualizing the described model is provided in Figure 1.

Developed in MATLAB, the model was validated using experimental data from a hardware in the loop test facility, successfully simulating continuous charging scenarios. This integration of simulation and experimental validation enables a comprehensive evaluation of storage system performance under realistic conditions.

The recent modifications enabling predefined charging and discharging profiles to be implemented by converting hourly energy demands of both consumers and heat generators, defined at specified hot water temperature levels, into corresponding volume flow rates entering or leaving the tank, thus directly affecting the energy balance calculations of the discretized tank layers.

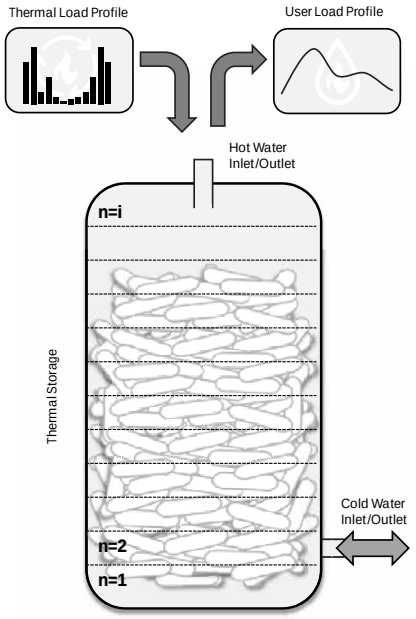


Figure 1: Schematic representation of the hybrid thermal storage model, illustrating the surrounding system modeled as load profiles.

A 2 m² domestic hot water tank was top loaded at 60 °C and discharged following a standard family load profile (Bundesverband Wärmepumpe (BWP) e.V. 2023). The study compared three tank configurations without losses: an ideal stratified tank, a tank filled with PCM capsules with a melting point of 58 °C, and one with capsules at 37 °C.

A 24-hour segment of a simulation cycle is presented

in Figure 2. The first column illustrates the hourly temperature distribution along the tank height, while the second column depicts the stored heat distributions at each 2 cm layer of height. The PCM58 capsules did not fully melt in this scenario. In contrast, the PCM37 configuration demonstrated a higher energy density spreading downward until hour 6, indicating significant latent heat absorption.

Despite storing 109 kWh (PCM37) at its peak or immediately after the last charging hour, compared to 83.5 kWh (PCM58), the temperature in the upper section of the PCM37 tank decreased more rapidly during discharge, reaching 38 °C by hour 21. At the same time, the PCM58 tank retained 52 °C, while the stratified tank provided 57 °C. After discharge (after hour 22), both PCM tanks exhibited slight reheating, stabilizing near the melting temperature of the respective PCM. For this scenario, the stratified tank maintained a stable hot water temperature longer, proving to be the better option when a minimum temperature of 55 °C is required.

While PCM integration did not yield an energetic advantage in this case, further evaluation is required for systems with indirect or multiple charging options through various thermal units. On a higher system level, the efficiency of the heat generator should also be considered, as it is significantly influenced by temperature levels, as seen in the case of heat pumps. Additionally, long-term thermal storage solutions that operate beyond hourly fluctuations should be analyzed to assess their impact on overall system efficiency. This study offers a practical tool for evaluating PCM-integrated storage systems under dynamic conditions.

References

Bundesverband Wärmepumpe (BWP) e.V. (2023): Leitfaden Trinkwassererwärmung. Hg. v. BWP. Online verfügbar unter https://www.waermepumpe.de/verband/publikationen/fachpublikationen/, zuletzt geprüft am 29.01.2025.

Jäger, Sarah; Pabst, Valerie; Renze, Peter (2024): Multizone Modeling for Hybrid Thermal Energy Storage. In: *Energies* 17 (12), S. 2854. DOI: 10.3390/en17122854.

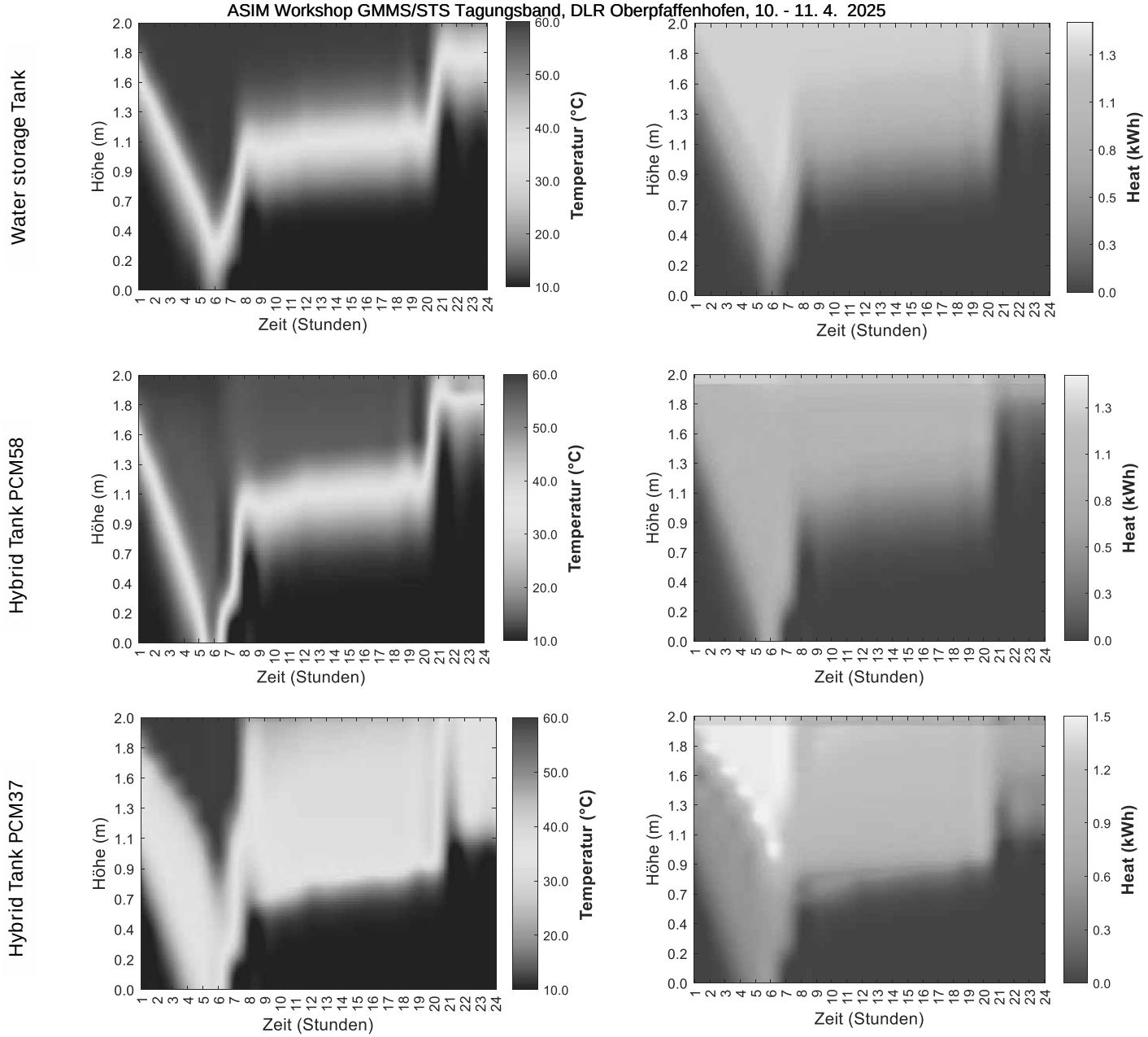


Figure 2: Temperature Distribution and Stored Heat for each of 100 Layers across a 24-Hour Segment of a Simulation

Simulating a Pneumatics Network using the DLR ThermoFluidStream Library

Peter Junglas[1*], Raphael Gebhart [2]

[1]PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany; *peter@peter-junglas.de*

[2]Institute of System Architectures in Aeronautics, German Aerospace Center (DLR), Münchener Str. 20, 82234 Weßling, Germany;

Abstract. Modeling and simulation of pneumatics networks is still a challenging task, plagued by initialization problems even in sophisticated environments such as the Modelica Fluid Library. The recently proposed DLR ThermoFluid Stream Library uses a promising new approach to cope with such problems. Therefore, it should be a convenient basis for a more specialized pneumatics library.

The essential concepts and components of such a library are presented, with a special focus on the notorious tee branch components. Their dynamic behaviour is very complex, since it couples the effects of dynamic pressure changes and friction losses, and often leads to stability problems. Results of systematic tests as well as more realistic models are discussed. They show that even though some problems with stability remain in special examples, the new library generally allows for the simulation of pneumatics networks using realistic tee branch models, which are more accurate than previous implementations.

Introduction

Modeling and simulation of pneumatic systems is a non-trivial endeavour, since it combines the turbulent flow of a compressible medium in a usually large pipe network with the highly non-linear behaviour of components such as actuators and valves [1]. A starting point for the mathematical description could be a non-linear partial differential equation describing the fluid flow, coupled with a set of ordinary differential equations modeling the mechanical behaviour of the corresponding components. Of course, this direct approach is usually unfeasible not only due to high computational demands, but because it requires a lot of fine-grained parameters to describe the model and provides much more data than is necessary for typical applications.

Applying a divide-and-conquer strategy, different modeling approaches are used according to the complexity of the system or component under study: A simple tee branch can be analyzed using the full power of a CFD simulation [2], while for more complex situations a coarse grained finite volume approach is employed, using discretizations in one or two dimensions [3]. To cope with very complex systems, one even reduces the description of many components to a zero-dimensional model, using ordinary differential or even purely algebraic equations to describe their behaviour, disregarding any spatial resolution. This approach is adopted in the Modelica Fluid library ("MFL") [4] for most of its components.

For the modeling of large pneumatic networks the MFL has been used in [5]. Unfortunately, most models studied there didn't run in standard Modelica environments, unless the behaviour of some components – especially the tee branches – had been simplified drastically. This is due to the structure of the model: For a pipe network the MFL approach leads to a large system of nonlinear equations, which needs very precise starting values to make the initialization converge.

The recently presented DLR ThermoFluidStream Library ("TFS") [6] has been invented to address these problems. For this purpose, it adds the inertial pressure of the fluid, promoting the mass flows to state variables. Additionally, it uses a clever approximation scheme that decouples the equations of the components, without destroying the correct behaviour in static or quasi-static models [7]. Furthermore, the flows generally have fixed directions, which simplifies the modeling. As a consequence, the initialization usually works, even starting with vanishing mass flow, which should make it a suitable approach for the modeling of pneumatic pipe networks. It is the basis of the specialized PneuBibTFS library presented here, which is freely available from [8].

To show that the TFS library is up to this task, we will closely follow the lines of [5]: After a short introduction to the library and its basic components, a special focus will be on the modeling of tee branches,

where several alternatives will be presented and extensively tested. Finally several variants of complete networks containing time-varying consumers will be analyzed and compared to the simplified versions presented in [5].

1 Using the DLR ThermoFluidStream Library

The problems with the initialization of models in different application areas are well-known, and a solution based on interpolating between the complete model and a simplified version has been proposed [9] and applied to thermofluid models [10]. Unfortunately, it only works in very special cases, especially not for pneumatic network models [5]. The TFS library addresses the initialization problem by introducing two major changes to the usual description of thermofluid models. They will be described briefly in the following, more details and motivations can be found in [7].

Integrating the Euler equation along a stream line leads to the "Newton's law like" pressure balance

$$\Delta r = \Delta q + \Delta p + \Delta p_{ext},$$

where Δq is the dynamic pressure difference due to change of velocity, Δp the pressure difference at the end points of the stream line, Δp_{ext} the pressure due to additional forces such as gravity or friction and Δr the pressure difference due to the inertia of the fluid, given by

$$\Delta r = -L\frac{d\dot{m}}{dt}.$$

The *inertance* L is independent of the thermodynamical state of a fluid and very small for gases. Since one is usually interested only in quasi-static processes, the inertial pressure difference Δr is neglected in the MFL library. In the TFS library, models include this term, where L is generally defined as a globally set small value – since one is not really interested in the transient behaviour –, but can be set for each component individually.

The second ingredient of the TFS library is the introduction of the *steady mass flow pressure* $\hat{p}$, which is defined by splitting the total pressure as

$$p = \hat{p} + r.$$

Its change $\Delta\hat{p}$ along a stream line generally depends on the total pressure and the mass flow. In the steady state r vanishes, therefore the approximation

$$\Delta\hat{p} = f(p,\dot{m}) \approx f(\hat{p},\dot{m})$$

is generally sufficient for quasi-static simulations. It leads to a decoupling of the component equations along the stream direction. This reduces the large set of nonlinear equations for the complete system to small-sized equations inside the components, thereby making the initialization problem feasible.

An important element in the design of a Modelica library is the connector. Instead of the stream connector used in the MFL library [11] the TFS library defines different connectors for ingoing and outgoing flows. They both use the mass flow $\dot{m}$ as flow variable and the initial pressure r as normal (*potential*) variable. Additionally they contain the thermodynamic state as input or output variable, respectively. It is usually given by the pressure, the specific enthalpy and a set of mass fractions. Here, the steady mass flow pressure $\hat{p}$ is used instead of the total pressure, thereby implementing the approximation scheme described above.

Based on these ideas, the freely available TFS library contains many of the components that are needed for pneumatics simulations. The specialized pneumatics library PneuBibTFS mainly just contains wrappers around the TFS counterparts, which reduce the number of parameters to the few needed here, and fix the medium to SimpleAir. This further reduces possible non-linearities in the medium model, leading to enhanced stability.

Basic elements of PneuBibTFS generated in this way are:

- `Pipe`: a straight pipe with pressure loss according to Cheng [12].
- `Bend`: a curved pipe with pressure loss from the MFL dissipation library.
- `Tank`: an isothermal pressure tank with explicit inflow and outflow ports.
- `PressureSource`, `PressureSink`: simple source and sink with given pressure.
- `MassFlowSource`, `MassFlowSink`: source and sink that define an input or output mass flow. This is non-trivial in TFS, since the mass flow is a state variable. The components work by combining a pressure source or sink with a control valve

from TFS that uses a PT1 dynamic to obtain the given mass flow.

- `MassFlowSourceLin`: source that uses a linear valve component to obtain a given input mass flow.
- `CVActuatorLin`: actuator using a linear valve to obtain a given output mass flow.

The linear mass flow source/sink components are simpler than their controlled counterparts and can lead to more stable models. Furthermore, they are used here to make results comparable to those of [5]. The critical tee branch components have to be created from scratch, they will be studied extensively in the following.

2 Modeling Tee Branches

As has been shown in [5], the modeling of the tee branch components is crucial for the stability of pneumatic network models. This is mainly a consequence of their complex behaviour, combining pressure drops due to internal friction with dynamic pressure changes caused by the changed cross sections of the fluid flow. In the case of splitting flows the division of the mass flows depends on incoming and outgoing pressures, which leads to a nonlinear coupling across the complete model. Using MFL-based components, even simple models did only run – i. e. survive the initialization phase –, when the tee branches were simplified drastically by completely disregarding all dynamical pressure changes.

Using the TFS approach instead, the mass flows become state variables, which breaks most of such loops. Since the flow directions are generally fixed in TFS, one now needs two different components: a splitter `TeeBranchS` and a junction `TeeBranchJ`, which join or split along the straight direction (cf. Fig. 1). For simplicity, we will only consider tee branches with a 90° angle and identical cross sections A at all three ports. This is a common situation in many pneumatics networks.

The basic equations to describe the behaviour of a tee branch have been formulated in [13] and are widely used in applications. They rely on two functions ζ_{cs} and ζ_{cb} that describe the pressure losses across the straight and branch directions. Since they contain a part of the dynamical pressure effects, they can be negative in certain cases, giving an actual pressure rise. Unfortunately, their concrete form varies largely in the literature [5]; we will use simple polynomials that fit published data.

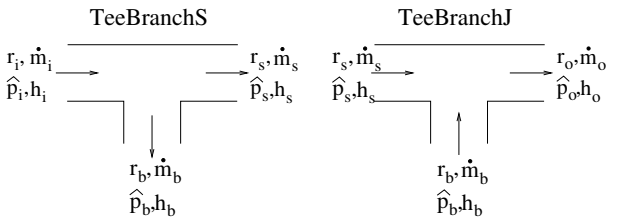


Figure 1: Tee Branch components.

The basic component `TeeBranchS` is simplified further by assuming constant temperature and density, using the density of the incoming flow everywhere. This avoids additional nonlinear loops inside the component and leads to the following equations:

$$
\begin{aligned}
0 &= \dot{m}_i + \dot{m}_s + \dot{m}_b \\
\rho &= \rho(\hat{p}_i, h_i) \\
\Delta p_s &= -\frac{1}{2\rho A^2}\zeta_{cs}\left(\frac{\dot{m}_b}{\dot{m}_i}\right)\dot{m}_i^2 \\
\Delta p_b &= -\frac{1}{2\rho A^2}\zeta_{cb}\left(\frac{\dot{m}_b}{\dot{m}_i}\right)\dot{m}_i^2 \\
\Delta p_{dyn,s} &= \frac{1}{2\rho A^2}(\dot{m}_i^2 - \dot{m}_s^2) \\
\Delta p_{dyn,b} &= \frac{1}{2\rho A^2}(\dot{m}_i^2 - \dot{m}_b^2) \\
\hat{p}_s &= \hat{p}_i + \Delta p_{dyn,s} + \Delta p_s \\
\hat{p}_b &= \hat{p}_i + \Delta p_{dyn,b} + \Delta p_b \\
h_s &= h_i \\
h_b &= h_i
\end{aligned}
$$

It is important to note that the equations to calculate the dynamic pressure differences Δp_{dyn} along the straight or branch direction are using the total input mass flow, while only a part of this mass flow reaches the corresponding output. This formulation is used, because the mass flow split is unknown beforehand. The error introduced here is made up for by including the difference to the correct dynamical pressure in the ζ-functions – which makes clear, why they can have negative values.

These are the same equations that have been used in [5] for the split case, if one identifies $\hat{p}$ and p. In the TFS context, one needs additional equations describing the behaviour of the r variables. They can be derived from results in [7] or directly read off the component `SplitterN` provided in the TFS library:

$$L\dot{m}_i = r_i - r_{mix}$$
$$L\dot{m}_s = r_s - r_{mix}$$
$$L\dot{m}_b = r_b - r_{mix}$$

where r_{mix} is an internal variable that is defined implicitely by the component equations.

A different approach to the modeling of a tee branch splitter uses the `DynamicSplitter` that is provided by the TFS library (cf. Fig. 2). It contains `DynamicPressureInflow/Outflow` components that compute dynamic pressure differences from the cross section area and the inlet/outlet velocity, which are given as parameter values. This leads exactly to the dynamic pressure differences from above.

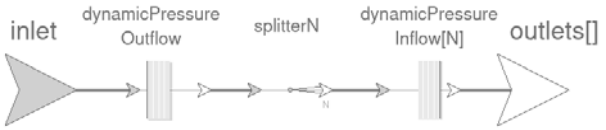


Figure 2: DynamicSplitter component.

The complete `TeeBranchS1` component adds a `SplitterPressureLoss` that computes the pressure loss caused by friction and the correction of the dynamical pressure, again using the ζ-functions (cf. Fig. 3). Basically, it reproduces the equations from above, with two small differences: The `DynamicPressureInflow/Outflow` include the temperature changes that are due to the – usually adiabatic, not isothermal – pressure change, and the frictional pressure computation uses the density at the outputs of the `DynamicSplitter`, not at the inlet. This corrects a part of the approximations that are made in the simpler `TeeBranchS`. Furthermore, its approach is more modular and easier to understand. On the other hand, its Modelica implementation consists of 147 equations altogether, compared to only 28 equations for the simpler component. Luckily, the Modelica preprocessing usually gets rid of this overhead.

To get even better results, one can use the component `TeeBranchS2`, which calculates the dynamic pressure differences by correctly using the densities of the input and output streams instead of using the input density everywhere. The price is the addition of two nonlinear equations inside the component. A similar approach can be used with the

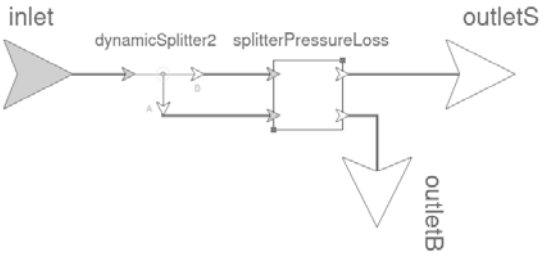


Figure 3: Alternative component TeeBranchS1.

`SplitterPressureLoss`, which would lead to two more nonlinear equations.

The construction of corresponding joining elements `TeeBranchJ`, `TeeBranchJ1` and `TeeBranchJ2` completely follows the lines above. The basic difference lies in the handling of the r variables when mixing input streams. The proper equations again can be found in [7] or in the component `JunctionN` from the TFS library.

3 Testing Tee Branches

The various tee branch components have been tested thoroughly using similar models as in [5], a typical example for the joining case is shown in Fig. 4. Here, the mass flows at the inflows and the pressure at the outflow are given explicitely.

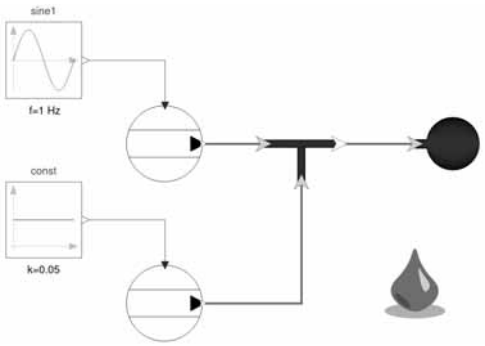


Figure 4: Model for testing a TeeBranchJ component.

The results for this example using the three different `TeeBranchJ` components and the `TeeBranch1` component from [5] are shown in Fig. 5. The plots for the basic MFL and TFS based components are almost identical, which is expected, since they use basically the same equations. The deviation at the beginning is due to the different initialization methods: MFL starts with

a given value of $\dot{m}$, while TFS starts here with $\dot{m} = 0$ and winds it up using the inertance equation. The slight phase difference is not caused by the inertance, but by the PT1 dynamic of the mass flow controller used in TFS. Much larger are the pressure differences between the three TFS components, especially for the straight branch. At this point, this seems to indicate that a better modeling of the density changes could be useful.

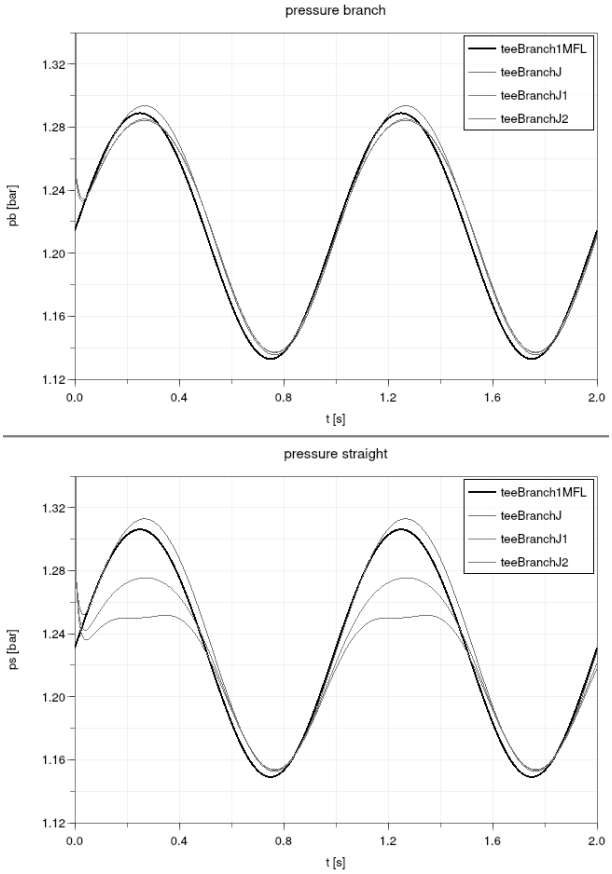


Figure 5: Comparison of the pressure drops in join mode.

More important than the exact results – which depend on the choice of the ζ-functions anyhow – is the question of stability: Do the models run immediately, only with special initial values or doesn't the initialization converge? To check this, test models similar to Fig. 4 have been analyzed, using different kinds of boundary conditions:

- a: pressure given at inflow, mass flow at outflow
- b: mass flow given at inflow, pressure at outflow
- c: pressure given at inflow and outflow

The results are unexpected: Only in the simple case, where two mass flows are given (case a for the splitter, case b for the joiner), all four components work. For the other cases, the MFL models work always, the basic TFS components in most cases, the more advanced TFS components never. The problem here is not the initialization, all models start and run for a (very) short time. Then the pressure values diverge rapidly. Obviously, the differential equations used here are highly unstable. In some cases, the problem can be fixed by using non-zero initial conditions for the mass flows, but often even very good starting points – coming from the working MFL model! – don't lead to a stable solution.

In additional tests a small pipe has been added either at the incoming or the outgoing straight branch. For the stable cases this leads to a problem with an MFL model: The splitter doesn't run with a pipe in the output [5]. The corresponding TFS models are not affected, they all work with the additional pipe on either side. In the unstable cases, the situation is more complicated, but generally, the situation gets worse in the MFL case, while in the TFS case several models that didn't run before, get stabilized by the additional pipe.

In conclusion, the tests show that the TFS approach does not solve all problems, due to the inherent instability of the basic equations. This apparently gets worse when the change in density is included. But at least it works in many cases, and the addition of pipes sometimes stabilizes a model.

4 Modeling Pneumatic Networks

To check the performance of the PneuBibTFS library in more realistic situations, the basic example model from [5] has been studied, which contains one `TeeBranchJ` and four `TeeBranchS` components, together with several pipes and curves, a pressure source, a few consumers and an auxiliary tank. Since the tank uses dedicated inflow and outflow ports, it is connected to the network via a loop consisting of a splitter and a joiner (cf. Fig. 6).

Starting with an empty tank (i. e. $p = p_0$), running the model works without problems and leads to results that are similar to those from [5] (cf. Fig. 7). If one replaces all tee branches by their more sophisticated versions, the models still run and reproduce the results of Fig. 7 within the plot accuracies. But in the MFS case, the model didn't run at all, unless one replaced the basic

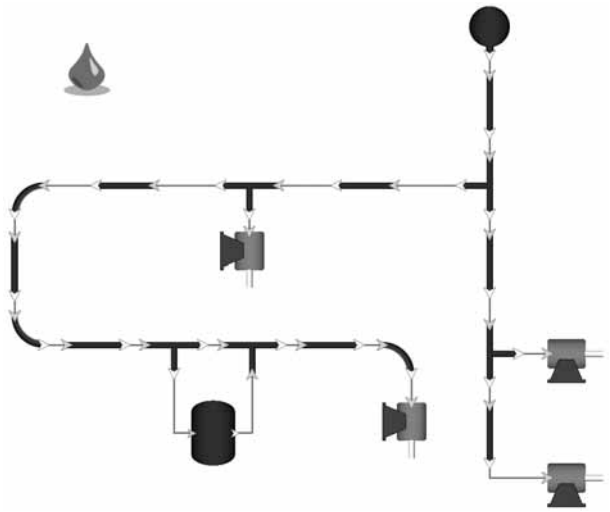


Figure 6: Model of the simple pneumatics network 1.

tee branch model by a very simplistic model based on substitutional pipe lengths. This shows that the somewhat unconvincing conclusions from the teebranch test results are much clearer in larger models: While the initialization problems in the MFL case get much more serious for larger models, in the TFS approach, the instabilities are largely mitigated.

A slightly extended example has been studied in [5] that contains an auxiliary tank between the two consumers on the right side. Building this model with PneuBibTFS, the simulation stops immediately with the error message

```
Positive mass flow rate at Volume outlet.
```

Apparently, in the initial phase of the simulation the medium flows into the tank through the outflow port, which is caught by an assertion. The TFS library includes a variable – hidden inside the `DropOfCommons` component – to reduce the assertion level from *error* to *warning*. Doing this, the model runs fine and produces the expected results. The backflow issue is a minor initialization problem and can be safely ignored here.

Increasing the simulation time one runs into another problem: The simulation stops at $t = 80$ s, one has hit the instability region. Taking a closer look at the model, one finds, that at this moment the consumer near the first tank is switched on for the first time. To increase the stability, a small pipe has been added between the splitter and the joiner that form the loop containing the tank (cf. Fig. 8). This works fine and the model now runs for long simulation times. Fig. 9 displays the pressure curves at the two consumers at the right side for the MFL and TFS variants. It shows clearly that the simplifications, which had been necessary to make the MFL model run, lead to significant deviations in the results.

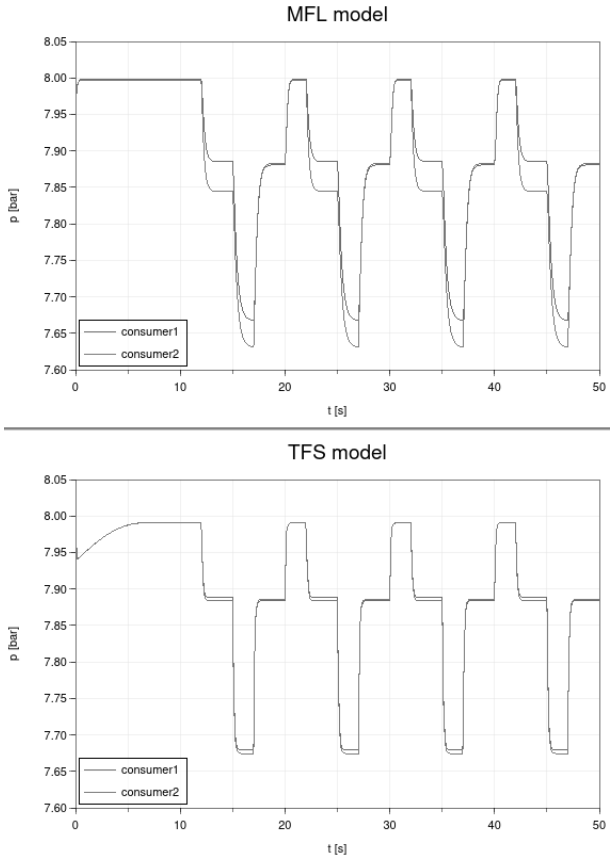


Figure 7: Simulation results of example network 1.

Finally, one of the real-world models from [5], coming from an industrial partner, has been ported to PneuBibTFS. It contains almost 60 components, among them three pumps, one tank, 12 consumers and 17 tee branches. The MFL version only contains the simplistic tee branch component and has about 4500 equations. For the port to TFS the flow directions have to be specified everywhere. Furthermore, the tank again has to be included via a small loop, and its initial pressure has been set to the (identical) pump pressures. The final model has only 1750 equations, since the TFL library has a much simpler structure than the MFL library.

The simulation of the TFS-based model stopped after 1 s with the usual blowup of all pressures. Additionally, several flows had the wrong direction, which is due to the identical pressures of all pumps. To ensure the

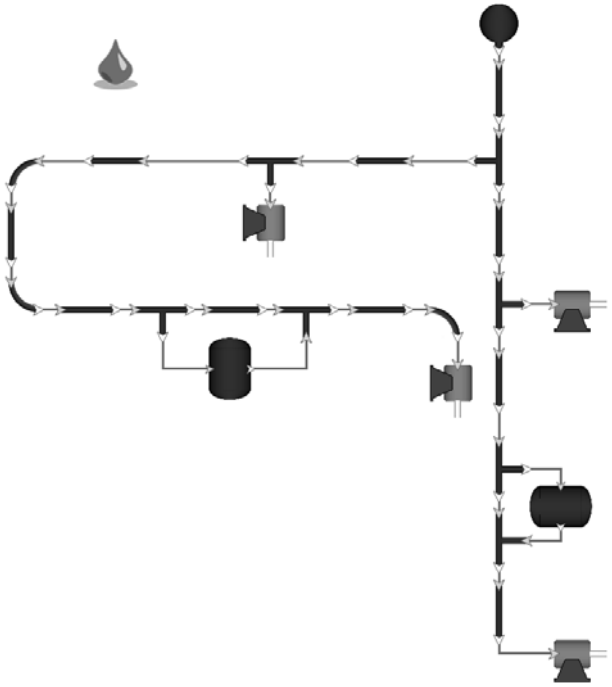


Figure 8: Model of the enlarged pneumatics network 2.

correct flow directions, the pressure of one pump has been increased marginally. With this change, the model runs immediately and qualitatively reproduces the results of the MFL version.

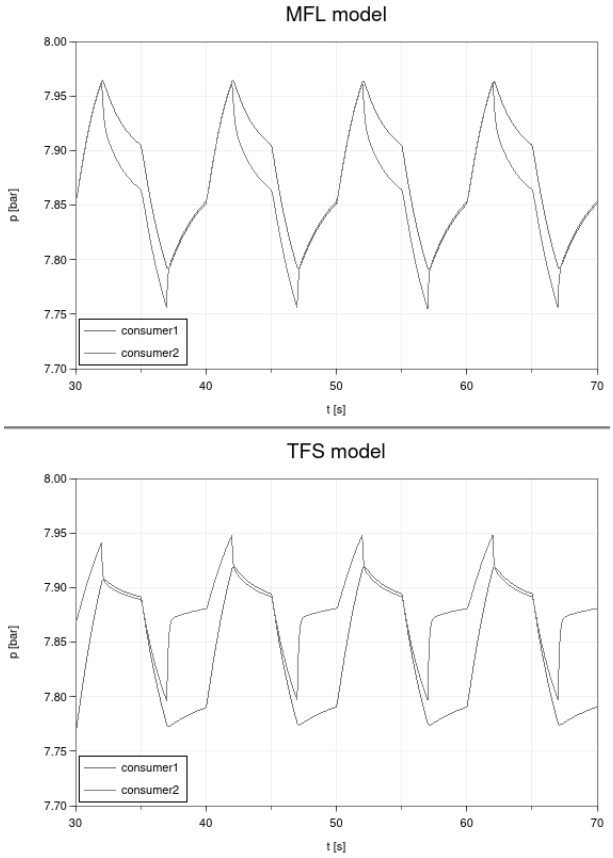


Figure 9: Simulation results of example network 2.

5 Conclusions

Though the PneuBibTFS library still has problems with stability in special examples, it allows for the simulation of pneumatics networks using a realistic tee branch model. In many cases, the TFS-based methods work much better than an MFL-based approach, especially for larger models. If problems appear, they can often be cured by insertion of auxiliary pipes. Comparison with the MFL-based results show significant differences, which are due to the very crude tee branch models used there. Apparently, the omission of the proper dynamic pressure changes introduced considerable errors.

Using the more detailed tee branch models that take into account local variations in density and temperature changed the results only marginally. Since these components reduce the general stability of the model, one should stick to the basic `TeeBranchS` and `TeeBranchJ` components.

In [5] the use of OpenModelica [14] as a modeling and simulation tool introduced additional problems. This has changed completely, all PneuBibTFS models that run in Dymola [15], work in OpenModelica as well, and vice versa. This is due to two effects: On the one hand, the OpenModelica simulator has been enhanced considerably in the last years [16], on the other hand, the new models are much simpler conceptionally, since they don't lead to huge monolithic nonlinear equations.

An interesting point for improvement is the modeling of the tanks: In reality, a tank is often connected to the pipe network using a simple port. It works as a buffer, the flow direction changes according to the pressure differences between the tank and the network. To model such a tank, one also needs a tee branch model that works with different flow directions. For such purposes, the TFS library has been enhanced to allow for bidirectional flows [17]. This leads to more complex components that are more tightly coupled. Whether such models deliver significantly better results and –

more importantly – are more stable, is an interesting question.

Clearly, the most important open point is the question of stability. Probably, the difficulties in solving the MFL-based nonlinear equations and the instability of the TFS-based differential equations are related. It would be interesting to study the instability of the basic tee branch equations in more detail and to find out, whether there exist more stable formulations, as well as how the stabilization in larger models actually works.

A basic conclusion from [5] with respect to the MFL library was:

> The fundamental problem of initialization seems to be still far from being solved.

In the light of the results presented here, it seems to be justified to claim that the TFS library has solved the initialization problem, at least for the class of models that have been studied here.

References

[1] Beater P. *Pneumatic drives*. Berlin Heidelberg New York: Springer. 2007.

[2] Aigner D. Gleichung zur Berechnung der hydraulischen Verluste der Rohrvereinigung - kalibriert mit Ergebnissen numerischer und physikalischer Modelle. *3R-international*. 2008;47(1).

[3] Fischer F, Schmitz K. Distributed Parameter Pneumatics. In: *Proc. 15th Int. Modelica Conference*. Aachen, Germany. 2023; pp. 85–44.

[4] Franke R, Casella F, Sielemann M, Proelss K, Otter M. Standardization of thermo-fluid modeling in Modelica.Fluid. In: *Proc. 7th Int. Modelica Conference*. Como, Italy. 2009; pp. 122–131.

[5] Drente P, Junglas P. Simulating a simple pneumatics network using the Modelica Fluid library. *SNE Simulation Notes Europe*. 2015;25(2):85–92.

[6] Zimmer D, Weber N, Meißner M. The DLR ThermoFluid Stream Library. *Electronics*. 2022;11(22).

[7] Zimmer D. Robust object-oriented formulation of directed thermofluid stream networks. *Mathematical and Computer Modelling of Dynamical Systems*. 2020; 26(3):204–233.

[8] Junglas P. *Pneumatics Network library in Modelica*. URL `https://www.peter-junglas.de/fh/simulation/pneubib.html`

[9] Sielemann M, Casella F, Otter M, Clauß C, Eborn J, Mattsson SE, Olsson H. Robust Initialization of Differential-Algebraic Equations Using Homotopy. In: *Proc. 8th Int. Modelica Conference*. Dresden, Germany. 2011; pp. 75–85.

[10] Casella F, Michael Sielemann LS. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. In: *Proc. 8th Int. Modelica Conference*. Dresden, Germany. 2011; pp. 1–11.

[11] Franke R, Casella F, Otter M, Sielemann M, Elmqvist H, Mattson SE, Olsson H. Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena. In: *Proc. 7th Int. Modelica Conference*. 2009; pp. 108–121.

[12] Cheng NS. Formulas for friction factor in transitional regimes. *Journal of Hydraulic Engineering*. 2008; 134(9):1357–1362.

[13] Miller DS. *Internal Flow Systems*. Cranfield, UK: The British Hydromechanics Research Association, 2nd ed. 1990.

[14] Fritzson P, Pop A, Abdelhak K, Ashgar A, Bachmann B, Braun W, Bouskela D, Braun R, Buffoni L, Casella F, Castro R, Franke R, Fritzson D, Gebremedhin M, Heuermann A, Lie B, Mengist A, Mikelsons L, Moudgalya K, Ochel L, Palanisamy A, Ruge V, Schamai W, Sjölund M, Thiele B, Tinnerholm J, Östlund P. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*. 2020;41(4):241–285.

[15] Brück D, Elmqvist H, Mattsson SE, Olsson H. Dymola for multi-engineering modeling and simulation. In: *Proc. 2nd Int. Modelica Conference*. Oberpfaffenhofen, Germany. 2002; pp. 55-1–55-8.

[16] Pop A, Östlund P, Casella F, Sjölund M, Franke R. A new OpenModelica compiler high performance frontend. In: *Proc. 13th Int. Modelica Conference*. Regensburg, Germany. 2019; pp. 689–698.

[17] Zimmer D, Weber N, Meißner M. Robust Simulation of Stream-Dominated Thermo-Fluid Systems: From Directed to Non-Directed Flows. *SNE Simulation Notes Europe*. 2021;31(4):177–184.

A simple supplier-based market model offering rich dynamic potential

Christian Iniotakis

Institute for Management and Entrepreneurship, TH Ulm, Prittwitzstraße 10, 89075 Ulm; *christian.iniotakis@thu.de*

Abstract. This work introduces a (toy) market model that relies on a simple, individual decision-making behaviour of suppliers: Which fraction of the current sales to spend for the next production? Within the model, this behaviour - together with characteristic properties of both the market and the particular cost setting - is decisive for the supplier's long-term existence on the market as well as the resulting market dynamics in general. Eventually, such a market can vanish, become stationary, exhibit market cycles or even go chaotic.

Introduction

Markets exist in an abundance of variations and are prime examples of complex systems in economy, e.g. [1, 2, 3]. As a basic feature, real markets as well as typical approaches for modelling them involve suppliers, demanders, and some type of interactions resulting in trading activities, e.g. [3]. From the simulation perspective, it is quite common to work with rather simple (toy) market models, which - despite their simplicity - still allow for qualitative insights into market behaviours, and are also helpful in the course of student education, e.g. [2, 4]. This work introduces a model which is particularly simple, while offering a suprising variety of inherent dynamic market patterns. The model only applies to some specific types of markets relying on certain assumptions and restrictions as briefly described in the following.

1 Model Assumptions

The market under consideration occurs in discrete time steps t and enables the trade of one particular, indifferentiable, commodity-like good. In other words, units from different suppliers are all valued to be of the same quality from the demanders' perspective. Moreover, once produced all units need to be directly sold, e.g. they are perishing goods or there are only negligible buffering or storage options. All market participants are fully aware of the overall situation, such that the price P per sold unit is the same for every unit and solely depends on the total number (or quantity) Q of units sold at the market, also referred to as market size in the following. The suppliers on this market are explicitly modelled, with each supplier n producing an individual number q_n of units. The production of each supplier is assumed to be fully scalable, just based on individual, but constant production costs c_n per unit, and without relevant dead time, i.e. any delay caused by actual production times can be neglected. In contrast to the suppliers, the demanders are only taken into account in an aggregated way in form of a linearized demand curve, determining the resulting unit price $P(Q)$. The demand curve has a negative slope - the larger Q, the lower the unit price P - and is fully characterized by two parameters denoted as Q_{max} and P_{max} in the following. Q_{max} is the particular market size for which the unit price even drops to zero, commonly referred to as saturation quantity. The prohibitive price P_{max} is the maximum price per unit to be achieved in the limit of minimum supply $Q \to 0$. Probably the most crucial feature of the model is how to implement the dynamical behaviour: How do suppliers decide about their next, future productions $q_n(t+1)$ based on the current market situation at time t or even the past? For example, a rather simple, supposedly rational approach of an individual production increase or decrease - depending on the current market being profitable or not - leads to a closed simulation model with typical characteristics also found in real markets of that type, but is not in the scope here [5]. The model presented in this work, however, relies on an approach even simpler. After selling all units, a supplier just uses an individually chosen 're-investment' percentage ε_n of the sales for the production of the next units and simply takes the remainder as his or her personal profit, as sketched in Fig. 1. This model assumption certainly is debatable, nevertheless some particular

properties should be pointed out. Firstly, such a supplier accepts a profit that directly scales with the current sales, which is not as far-fetched as it might seem at first glance. Secondly, following this behaviour offers the attractive side aspect of never bearing an individual loss, at least in principle, which is rather hard to achieve on such a type of market otherwise. Moreover, such a simple procedure of decision-making neither requires effort nor access to particular market or competitor information.

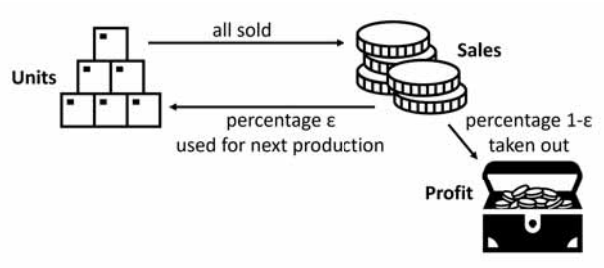


Figure 1: Sketch of simple supplier behaviour: After selling all units, a fixed percentage ε of the sales is re-used for producing the next units, while the remainder is taken out as profit.

2 Model Equations

Each supplier n starts with an initial number of units $\mathbf{q}_n(0)$. Here and in the following, quantities depicted in bold style are normalized to Q_{max} for convenience and therefore dimensionless. The resulting dynamic behaviour is determined by the time stepping equations

$$\mathbf{q}_n(t+1) = r_n \cdot \mathbf{q}_n(t) \cdot (1 - \mathbf{Q}(t)) \tag{1}$$

together with the total market size

$$\mathbf{Q}(t) = \sum_n \mathbf{q}_n(t). \tag{2}$$

The individual parameter r_n is dimensionless, assumed to be constant over time and simply given by:

$$r_n = \varepsilon_n \cdot \frac{P_{max}}{c_n}.$$

Here, as introduced above, ε_n is the 're-investment' percentage, c_n are the production costs per unit, and P_{max} is the prohibitive price. While the latter factor is identical for all suppliers and represents a property of market and good, the production costs depend on various aspects, such as purchasing and labour costs, the involved production technology as well as the excellence in applying it, etc.; typically, this factor is not too different between similar supplier settings. In contrast, the factor ε_n is a fully individual parameter that can simply and freely be chosen without any restriction. In the following, the constraint $0 < r_n < 4$ is applied for all involved suppliers which ensures the total market size to stay within regular bounds $0 < \mathbf{Q} < 1$ at all times. As a consequence, the market can go on forever with both unit prices and quantities always positive.

3 Some Results

Competition

The model turns out to be highly competitive in the sense of a tough selection of suppliers surviving on the market in the long run. As a key result, this existential issue is purely determined by the r_n-values. As long as all $r_n \leq 1$, the market will just die out completely. In any other case, it is exactly the (group of) supplier(s) with the maximum r to prevail over time. In other words, if the indiviual r_n of a given supplier is not the largest one, it inevitably means the long-term exit from the market. Some sample simulations illustrating this aspect are depicted in Fig. 2.

Long-term Market Dynamics

The model approaches a monopolistic or oligopolistic scenario in the long run. Exploiting the situation of only the supplier(s) with the (same) maximum r to be still alive and active allows to derive a simpler equation directly describing the long-term dynamics of the overall market:

$$\mathbf{Q}(t+1) = r \cdot \mathbf{Q}(t) \cdot (1 - \mathbf{Q}(t)). \tag{3}$$

This step-wise dynamics turns out to be identical to the so-called logistic map, presumably the most prominent equation in the history of chaos, allowing for an overwhelming variety of dynamical behaviours just determined by the actual value of r [6]. As mentioned above, for $0 < r \leq 1$, no market will exist in the long run. For $1 < r \leq 3$, the system converges to a finite and stationary market of size $\mathbf{Q}_\infty = 1 - \frac{1}{r}$, as can be seen in Fig. 2, (top). For r slightly larger than 3, the potential stationary solution - in form of the fixed point men-

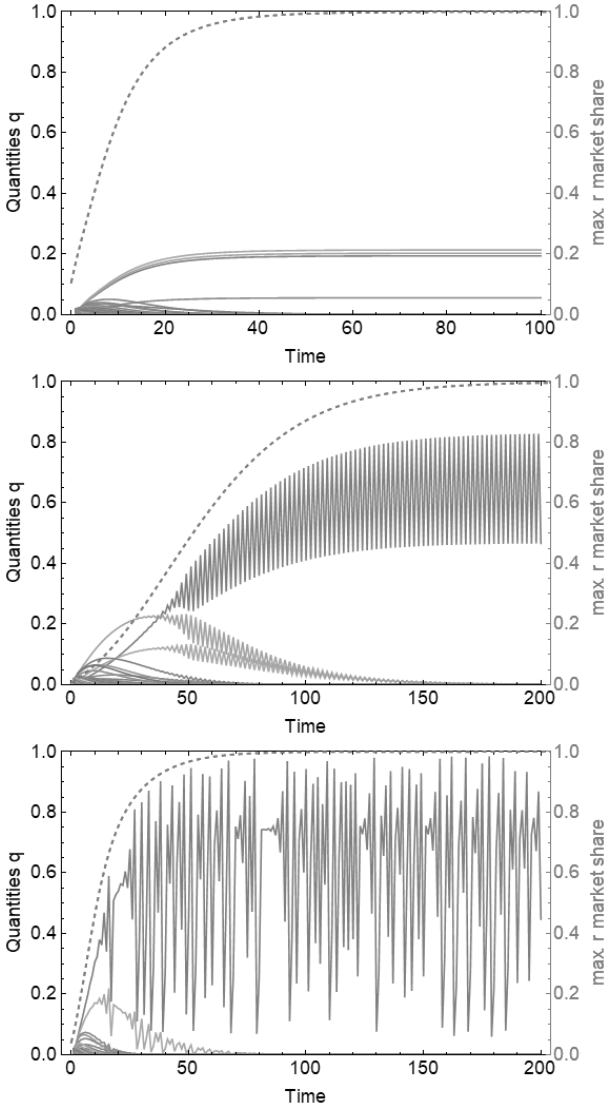


Figure 2: Dynamics in a competitive setting of 50 suppliers starting with initial quantities $\mathbf{q}_n(0)$ both random and small, as well as randomly assigned r_n. The red dashed line indicates the market share of the (group of) supplier(s) with maximum r.
(top) 4 suppliers with maximum $r = 3.0$, the market becoming stationary with $\mathbf{Q}_\infty = 2/3$.
(middle) 1 supplier with maximum $r = 3.34$, the market becoming cyclic with period 2.
(bottom) 1 supplier with maximum $r = 3.94$, the market becoming chaotic.

tioned above - becomes unstable, and the system shows a bifurcation leading to a dynamical pattern of period 2, cf. Fig. 2, (middle). Generally, the interval $3 < r < 4$ contains periodic market cycles for any natural period and plenty of chaotic dynamics, as for the sample case depicted in Fig. 2, (bottom). Analogous to [6], and for a better understanding of the overall situation, the long-term market sizes are indicated in Fig. 3 for the full range $0 < r < 4$ (top), as well as for some exemplary sub-range (bottom). Clearly, the long-term market sizes are restricted to specific values or confined to certain bands, with rather stable islands to be located in close vicinity to chaotic regions. Thus, a small change in r can drastically affect the resulting market dynamical behaviour as typical for complex systems.

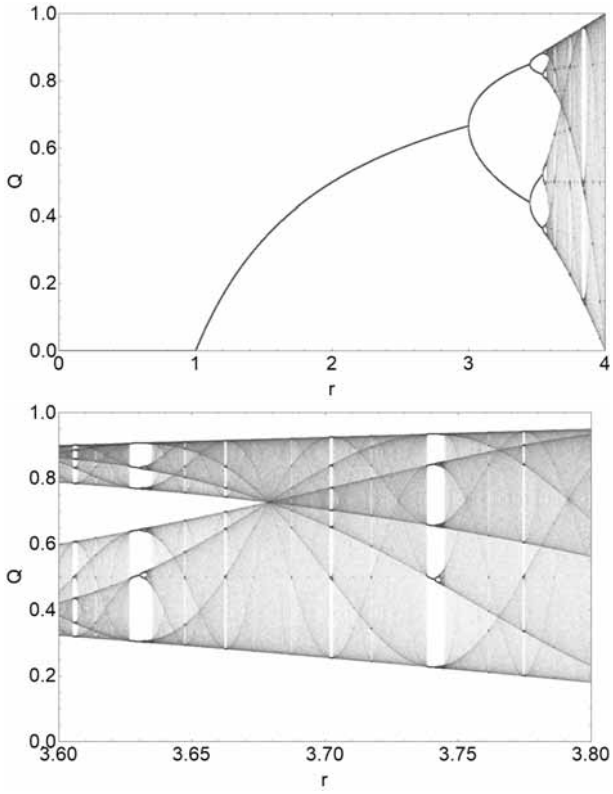


Figure 3: Long-term values for the total market size $\mathbf{Q}$ depending on r in the full relevant range **(top)** and as an exemplary zoom-in **(bottom)**, analogous to [6]. Illustrations indicate the last 100 market sizes after 10.000 time steps starting from $\mathbf{Q}(0) = 0.5$, for 20.000 different r-values along the x-axis.

4 Further Aspects

This work shows that even simple, constant decision-making behaviours of suppliers can result in an abun-

dance of different market dynamics. In particular, a market according to this model could even follow cycles of any period - at least in principle. This is in no contradiction to other time-related mechanisms and aspects also known to cause market cycles, such as dead times or supplier memories. Despite its simplicity and limitations, the model might help to indicate, in which markets to expect chaos - or not, cf. e.g. [7]. It should be noted, that even though this work sets the focus on total market sizes, the dynamics of the dimensionless unit price $\mathbf{P}$, normalized to P_{max}, is directly correlated via $\mathbf{P} = 1 - \mathbf{Q}$ for all times. While a profit consideration is straightforward to achieve, e.g. [5], allowing a change of $r_n(t)$ over time is far beyond the scope of this work: On top of the inherent complexity already contained in the model for constant r_n, other layers of complexity are expected to emerge. For example, suppliers simply adapting $\varepsilon_n(t)$ based on any strategy or investing parts of sales in cost-cutting activities in order to reduce $c_n(t)$ go along with all kinds of interactions assumed to lead to overall market patterns far from trivial. From a supplier's perspective, there is also the immanent dilemma between the basic need for ensuring long-term existence on the market, while still somehow receiving (or even maximizing) profit.

References

[1] Arthur WB. Foundations of complexity economics. *Nat Rev Phys*. 2021; 3: 136-145. doi: 10.1038/s42254-020-00273-3

[2] Bouchaud J-P. The (unfortunate) complexity of the economy. *Phys. World*. 2009; 22(04): 28-32. doi: 10.1088/2058-7058/22/04/39

[3] Farmer JD, Foley D. The economy needs agent-based modelling. *Nature*. 2009; 460: 685-686. doi: 10.1038/460685a

[4] Sayama H. *Introduction to the Modeling and Analysis of Complex Systems*. Open SUNY Textbooks, Milne Library; 2015. 478 p.

[5] Iniotakis C. Unpublished

[6] May RM. Simple mathematical models with very complicated dynamics. *Nature*. 1976; 261: 459-467. doi: 10.1038/261459a0

[7] Faggini M, Parziale A. More than 20 years of chaos in economics. *Mind Soc*. 2016; 15: 53-69. doi: 10.1007/s11299-015-0164-1

Verallgemeinerte Virtuelle Stochastische Sensoren – Training und Rekonstruktion auf unterschiedlichen Ein-Personen Apartments

Vishwajeet Karumuri, Claudia Krull

Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Deutschland; *claudia.krull@ovgu.de

Abstract

Virtuelle stochastische Sensoren (VSS) wurden entwickelt um für partiell-beobachtbare stochastischen Systeme anhand von Systemausgaben das nicht beobachtete Verhalten des Systems zu rekonstruieren. Dies kann als inverses Problem im Bereich Simulation und Modellierung betrachtet werden, da basierend auf Systemausgaben auf Systemverhalten geschlossen wird, und nicht umgekehrt. Die betrachteten Systeme haben ein verborgenes stochastisches Zeitverhalten, und erzeugen beobachtbare Ausgaben durch einen zweiten stochastischen Prozess basierend auf dem verborgenen Systemverhalten. Anhand von Protokollen unterschiedlicher Systemausgaben schätzen VSS ab, was system-intern wahrscheinlich abgelaufen ist, um diese Ausgaben zu erzeugen. [1]

Ein vielversprechender Anwendungsbereich von VSS ist Ambient Assisted Living, insbesondere die Rekonstruktion der Aktivitäten von Bewohnern eines Apartments anhand von Bewegungs- und Türsensoren. Die Methoden können perspektivisch genutzt werden, um Notfälle zu erkennen oder Abweichungen von der Routine, die Hinweise auf degenerative Erkrankungen sein können. Dadurch könnte älteren Personen eine längere Selbständigkeit in einem gewohnten Umfeld ermöglicht werden. [2]

In den bisherigen Anwendungen wurden die VSS Modelle mit Daten eines Apartments parametriert und dann zur Rekonstruktion des Bewohner Verhaltens des gleichen Systems genutzt. Um VSS praktisch anwendbar zu machen, müssen portable und skalierbare Methoden erarbeitet werden. In einer ersten Untersuchung wurde die Verallgemeinerung der Aktivitäten untersucht, und die Auswirkungen auf die Genauigkeit der Rekonstruktion. [3]

Aufbauend darauf wurde in einer Masterarbeit [4] untersucht, ob sich durch Verallgemeinerung der Sensorkonfiguration ein generalisiertes Modell auf Daten mehrerer Apartments trainieren lässt, und dann die Rekonstruktion für ein System ermöglicht auf dem nicht trainiert wurde. Es wurden verschiedene Verallgemeinerungen der Sensoren getestet und mit unterschiedlichen verallgemeinerten Aktivitäten kombiniert. Insbesondere die Wechselwirkung zwischen den verschiedenen Abstraktionsebenen wurden untersucht. Außerdem werden zwei verschiedenen Modelltypen untersucht.

Die Arbeit zeigt, dass eine Stufe der Verallgemeinerung sowohl der Aktivitäten als auch der Sensoren notwendig ist, um ein generalisiertes Modell zu erhalten. Eine weitere Verallgemeinerung führt allerdings wieder zur Verschlechterung der Rekonstruktionsgüte. Die Ergebnisse der Masterarbeit sind durchwachsen, auch wegen der geringen Datenmenge, bieten aber Ansätze für Verbesserungen.

Perspektivisch ist die Nutzung von generalisierten Modellen im Bereich AAL notwendig, da sonst für jeden Haushalt ein neues Aktivitäten Modell trainiert werden müsste. Die Vorgehensweise ist interessant, da es in der Regel große Ähnlichkeiten gibt zwischen den Aktivitäten von Bewohnern in Seniorenwohnanlagen gibt. Durch AAL kann langfristig eine längere Selbständigkeit ermöglicht werden, was bei einer alternden Bevölkerung unumgänglich ist.

Die Nutzung von VSS für inverse Fragestellungen in Simulation und Modellierung ist interessant, wenn die in realen Systemen verfügbaren Daten quantitativ nicht ausreichen um klassische KI Methoden zu trainieren, und gleichzeitig Systemwissen vorhanden ist.

References

[1] C. Krull, Virtual Stochastic Sensors: Formal Background and Example Applications, Magdeburg: Shaker Verlag Düren, 2021.

[2] L. Fialho Müller. Feasibility and Applicability of Virtual Stochastic Sensors for Human Activity Recognition in the Context of Ambient Assisted Living, Master Thesis, Otto-von-Guericke Universität 2021

[3] V. Karumuri, C. Krull , Virtual Stochastic Sensors for Ambient Assisted Living - Analyzing the Effect of Generalized Resident Behavior, ASIM SST, Universität der Bundeswehr München, 2024.

[4] V. Karumuri. Generalization of Virtual Stochastic Sensors in the Field of Ambient Assisted Living for Multiple Single-Resident Apartments, Master Thesis, Otto-von-Guericke Universität 2024

Automated Generation of Simulation Models Based on Plant Engineering Data

Malte Ramonat[1], Marius Wieduwilt[1], Luca von Roenn[1], Felix Gehlhoff[1*]

[1]Department of Automation Technology, Helmut Schmidt University / University of the Federal Armed Forces Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany; *felix.gehlhoff@hsu-hh.de*

Abstract. The increasing complexity of process plants and their automation systems requires simulation models for efficient optimization, and operational decision-making. Manual model generation is time-consuming and prone to errors, necessitating automated approaches. This paper presents a method for the automated generation of simulation models from plant engineering data, leveraging AutomationML for topology modeling and OpenModelica for simulation execution. The method enables an automated translation of plant structure and component information as well as control logic into simulation-formats, reducing modeling effort while ensuring accuracy. Evaluations on a pilot-scale process plant demonstrate the feasibility and benefits of the proposed approach.

Introduction

Simulation models play a crucial role in modern engineering, supporting design, validation, and operational decision-making [1]. Especially in process plants, where complex interdependencies and dynamic behaviors must be accounted for, simulations enable scenario testing without risking damage to physical assets [2]. However, generating simulation models remains a labor-intensive and expertise-dependent task. The high manual effort and associated costs limit the widespread adoption of simulation-based approaches in industrial settings [3].

Automated simulation model generation not only accelerates the model development process but also enhances model accuracy and consistency [4]. By systematically extracting and processing plant engineering data, the method ensures that essential details are retained while eliminating common manual errors [5]. This is particularly beneficial in industries where safety and precision are paramount, such as chemical processing, power generation, and manufacturing automation. The ability to rapidly adapt and update simulation models further supports agile engineering workflows and continuous process improvement [6].

Moreover, integrating automation in model generation aligns with the broader trend of digital transformation in industrial engineering [7]. With the increasing adoption of Industry 4.0 technologies, the need for seamless interoperability between design, simulation, and operational systems becomes more critical [8]. Automated model generation serves as a key enabler for this integration, facilitating improved decision-making through enhanced simulation fidelity and accessibility.

Existing research has explored various methods for automating simulation model generation, particularly for early engineering phases [9]. However, these approaches often lack the necessary detail for accurate operational-phase simulation models [8]. To bridge this gap, a novel method is needed to extract plant engineering data systematically and transform it into detailed simulation models that remain usable throughout the plant lifecycle [10].

The proposed approach builds on the established data exchange standard AutomationML [11] and employs OpenModelica [12] as a target simulation environment. By automating the mapping of topological and parametric data, this method reduces the reliance on domain expertise and minimizes the risk of inconsistencies in manually created models [13]. This paper outlines the method's development, implementation, and evaluation, demonstrating its effectiveness in a real-world laboratory-scale process plant [2].

The remainder of this paper is structured as follows: Section 2 provides an overview of relevant background and fundamental concepts. Section 3 reviews the state of the art in automated simulation model generation. Section 4 presents the proposed method, including the extraction and processing of plant engineering data. Section 5 details the implementation of the method and its evaluation on a laboratory-scale process plant. Section 6

discusses the findings and implications, while Section 7 concludes the paper and outlines directions for future research.

1 Background and Fundamentals

The increasing use of simulation in process plant engineering has led to the need for automated methods that streamline the generation of simulation models [1]. Traditionally, simulation models are manually created, requiring substantial engineering effort and domain-specific knowledge [9]. With the introduction of standardized data exchange formats such as AutomationML, new opportunities arise to automate the transformation of engineering data into simulation-ready formats [2].

1.1 Data Representation and Standardization

A key aspect of simulation model generation is the representation of plant topology [10]. Process plants typically consist of interconnected components, such as tanks, pipes, pumps, and control elements, all of which must be accurately represented in a simulation environment [12]. AutomationML provides a hierarchical structure that allows these components to be systematically defined and interconnected, serving as a basis for automatic model generation [6].

Furthermore, AutomationML enables the representation of different engineering domains within a unified data format, facilitating seamless integration between mechanical, electrical, and control system models [8]. However, a significant challenge remains in ensuring the consistency of data across these domains. Variations in how plant components are modelled and described in different engineering tools can introduce inconsistencies that need to be resolved before generating an accurate simulation model.

To bridge this gap, mapping techniques and transformation rules must be developed that can standardize heterogeneous engineering data into a coherent simulation representation. This includes the extraction of component parameters, connection definitions, naming conventions, and operational constraints from various design sources. The use of structured templates and validation rules can help ensure the completeness and correctness of the converted data [4].

1.2 Challenges in Automation Model Generation

Despite advancements in data standardization, several challenges persist in automating the simulation model generation process [14]. One major issue is data inconsistency, as engineering data is often scattered across multiple sources and tools [15]. Extracting relevant information and ensuring its accuracy requires rule-based data transformation approaches that can reconcile differences in nomenclature, data structures, and parameter units [16].

Another significant challenge is the handling of incomplete engineering data [17]. In many cases, plant models miss certain information, such as sensor calibration values or detailed fluid properties, which are crucial for accurate simulations [18]. One approach to addressing this problem is the use of heuristic inference techniques that estimate missing values based on available data trends and best-practice assumptions [19].

Additionally, the computational efficiency of automated model generation must be optimized to handle large-scale plant models [20]. As process plants often consist of thousands of interconnected components, model generation algorithms must be scalable and capable of efficiently processing complex topologies. Advances in machine learning and AI-driven data integration methods hold promise for further improving the robustness and adaptability of automated model generation systems [21]. The efficiency of model generation is especially important, as plants regularly undergo changes and reconfigurations. In order for the simulation model to stay relevant and to prevent a drift in model accuracy, the model generation process must be repeated regularly, thus necessitating computational efficiency

1.3 Integration of Control Logic into Simulation Models

A significant advancement in simulation model generation is the integration of process control logic. Traditionally, control strategies are designed independently from simulation models, leading to potential mismatches between the simulated and real-world plant behavior [22]. By embedding control logic directly into simulation models (Model in the Loop, MIL) or creating the control logic in e.g. 61131-3 code on an emulated programmable logic controller (PLC) via Software in the Loop (SIL) engineers can achieve more accurate and comprehensive system validation [23].

The use of PLCopenXML as a data exchange standard for control logic enables the seamless transfer of control strategies into simulation environments [24]. This approach allows control sequences, setpoints, and interlocking mechanisms to be represented as structured data that can be interpreted by simulation tools [25]. Through this integration, engineers can test control algorithms under realistic operating conditions before deploying them to physical systems [26].

Furthermore, closed-loop simulation methodologies, where control logic interacts dynamically with process models, offer enhanced validation capabilities [27]. These simulations enable engineers to analyze system responses to disturbances, optimize control parameters, and identify potential failure modes. This approach significantly reduces commissioning time and minimizes risks associated with control system implementation [28].

By addressing these challenges and leveraging standardized data formats, automated simulation model generation can become a crucial tool in process plant engineering, improving efficiency, accuracy, and reliability.

2 State of the Art in Automated Simulation Model Generation

The automation of simulation model generation has evolved significantly in recent years. Various methodologies have been developed to address the challenges associated with manual modelling efforts, aiming to enhance accuracy, efficiency, and scalability [10].

One prominent approach is data-driven modelling, which leverages historical process data to construct and refine simulation models [4]. Machine learning and artificial intelligence techniques have been integrated to automate parameter estimation, identify behavioral patterns, and optimize simulation performance [1]. However, the quality of these models is highly dependent on the availability and reliability of training data [9]. A key challenge in data-driven approaches is the generalizability of models across different plant configurations. Although AI-based methods improve efficiency, they require significant computational resources and domain-specific training data, which limits their immediate applicability in highly heterogeneous industrial environments [2].

Another widely used method is rule-based model transformation, where predefined transformation rules convert engineering schematics into structured simulation models [3]. This approach ensures consistency across engineering and simulation domains while maintaining adaptability to various process plant configurations [8]. A major advantage of rule-based transformation is its interpretability, as domain experts can validate the applied rules and make necessary adjustments. However, one limitation is the difficulty of maintaining rule sets for highly complex or frequently changing process plants [5]. To mitigate this issue, adaptive rule-based systems have been introduced, incorporating feedback loops that refine transformation rules based on discrepancies between simulated and actual plant behavior [29].

Hybrid techniques, which combine data-driven and rule-based methodologies, offer a balanced solution by leveraging the strengths of both approaches [30]. These hybrid approaches enable dynamic adjustments to model parameters based on real-time operational data, enhancing the fidelity of automated simulation models [6]. An example of this hybrid strategy is the integration of sensor data streams with rule-based transformation processes, allowing models to self-adjust in response to changing operational conditions [12]. This significantly improves long-term accuracy and usability of simulation models in dynamic industrial environments [7].

The emerging concept of the digital twin further expands the scope of automated model generation, by continuously generating new simulation models whenever changes occur in the real plant enabling real-time [13]. Digital twins provide an advanced simulation framework by continuously integrating live sensor data with engineering models, facilitating real-time scenario analysis and predictive maintenance [20]. These systems are becoming increasingly valuable for process optimization and operational decision-making [22]. However, challenges remain in ensuring data accuracy, computational efficiency, and secure integration with existing IT infrastructures [31]. Recent advancements in edge computing and cloud-based simulations are addressing these challenges, enabling scalable and efficient implementations of digital twin-based simulation models [22].

Additionally, automated model generation is increasingly influenced by advancements in knowledge-based systems and ontology-driven modelling [25]. These approaches aim to formalize domain knowledge into machine-readable structures, enabling simulation tools to interpret and apply expert knowledge autonomously [26]. By leveraging semantic technologies, simulation models can be dynamically adapted to represent new engineering

concepts without extensive manual reconfiguration. However, the complexity of developing and maintaining such knowledge representations remains a challenge, requiring collaborative efforts between domain experts and software engineers.

In summary, the state of the art in automated simulation model generation continues to evolve through the integration of AI, rule-based methods, digital twins, and knowledge-driven modelling. The combination of these approaches holds significant promise for improving accuracy, efficiency, and adaptability in industrial simulation processes [32]. However, existing research lacks the ability to automatically generate sub-models based on AutomationML topology models. Furthermore, previous work has focused on generating physical systems, while the generation or integration of control logic has not been a priority.

3 Proposed Methodology

To address the challenges identified in previous sections, this paper proposes a systematic method for automated simulation model generation based on structured plant engineering data. The methodology builds on established data exchange formats and integrates key advancements in model transformation, control logic integration, and data-driven adaptation.

3.1 Extraction of Engineering Data

The foundation of the proposed methodology lies in the structured extraction of plant engineering data from existing sources. This data typically resides in engineering tools such as CAD software, automation databases, and process design documents. AutomationML serves as a standardized and widespread exchange format, allowing for the systematic representation of plant topologies, component attributes, and interconnections.

To ensure consistency, the extracted data undergoes a pre-processing step, where inconsistencies, missing attributes, and redundant elements are identified and corrected. This step utilizes rule-based data validation techniques to ensure that all required parameters for simulation model generation are available and accurate. Furthermore, data normalization techniques are applied to standardize different formats and ensure interoperability between engineering and simulation environments.

3.2 Transformation into Simulation Models

Once extracted and pre-processed, the engineering data is transformed into a simulation-ready format. The transformation process follows a structured workflow:

- Component Mapping: Identifies and maps plant components (e.g., valves, pumps, sensors) to corresponding simulation elements using predefined templates.
- Topology Construction: Establishes connectivity between components, ensuring accurate representation of material and signal flows.
- Parameter Assignment: Assigns operational values such as flow rates, pressure levels, and control setpoints based on engineering specifications.
- Control Logic Integration: Incorporates PLC logic and automation rules using PLCopenXML to ensure realistic simulation behavior.

By leveraging these transformation steps, the proposed method ensures that simulation models accurately reflect the physical plant configuration while maintaining adaptability for different plant scenarios.

3.3 Model Validation and Optimization

To verify the accuracy and reliability of the generated simulation models, a validation phase is performed. The validation process involves comparing simulated behavior against historical process data and conducting sensitivity analyses to evaluate model robustness. Key steps include:

- Benchmarking Against Historical Data: Simulated process outputs are compared with historical operational data to ensure fidelity. This has been described in detail in [33].
- Analysis of Deviation Root Causes: Identifies the parameters in the simulation model responsible for occurring deviations across different operational scenarios. This has been described in detail in [34].
- Optimization Techniques: Employs automated parameter tuning to enhance model precision and stability. This has been described in detail in [35].

Through iterative validation and refinement, the proposed methodology achieves high-fidelity simulation models that can be confidently utilized for process optimization, predictive analysis, and operational training.

4 Implementation

This section details the implementation of the proposed methodology based on structured plant engineering data.

The implementation follows the outlined steps for automated simulation model generation and is applied to a modular process plant setup. The model validation and optimization steps have been demonstrated in [33–35]. The focus on the implementation and evaluation thus lies on the model generation process itself.

4.1 System Setup

The implementation was carried out using a laboratory-scale modular process plant equipped with industrial sensors, actuators, and a PLC. The plant consists of interconnected fluid storage tanks, pumps and valves, as well as control instrumentation. The automation system, i.e., the control logic, follows the IEC 61131-3 standard. The plant has been described in detail in [33, 34].

AutomationML was chosen as the primary data exchange format to facilitate structured data representation. The engineering data, including piping and instrumentation diagrams (P&IDs), control logic, and process parameters, was extracted from existing plant documentation and converted into an AutomationML-based format for further processing.

4.2 Model Generation Workflow

The extracted engineering data was processed through a structured transformation workflow to generate the simulation models. This workflow involved:

- Data Parsing and Pre-processing: The extracted AutomationML data was validated for consistency and completeness. Missing attributes were inferred using predefined engineering rules.
- Component Mapping: Key process components, i.e., pumps, pipes, and tanks, were mapped to their corresponding simulation elements within the OpenModelica environment.
- Topology Reconstruction: The connectivity between components was reconstructed to ensure accurate material and signal flow representation.
- Parameter Integration: Process-specific operational parameters were assigned to simulation models to ensure realistic behavior.
- Control Logic Implementation: The extracted PLC programs were integrated into the simulation environment (MIL) and an emulated PLC (SIL), allowing for closed-loop process control simulation.

By following these steps, a high-fidelity simulation model reflecting the real plant configuration was generated.

4.3 Integration with OpenModelica

The generated models were implemented in OpenModelica, a widely used open-source simulation platform. OpenModelica's component-based modelling approach allowed for seamless integration of process and control system models. The system's dynamic behavior was simulated under different operating conditions, validating the model's performance in replicating real-world plant operations.

5 Evaluation

This section evaluates the effectiveness of the proposed methodology by analyzing the generated simulation models in terms of accuracy, efficiency, and applicability in industrial settings. The evaluation is based on a set of predefined metrics and real-world validation tests.

5.1 Accuracy Assessment

To assess the accuracy of the generated models, simulated process outputs were compared with real operational data from the laboratory-scale process plant. The key performance indicators (KPIs) considered in this evaluation include flow rates, pressure levels, and temperature deviations. The comparison showed a deviation of less than 5% for most process variables, demonstrating high model fidelity. Additionally, transient behavior was evaluated by applying the real plant's control code onto the simulation model MIL and SIL test configurations and analyzing the system's dynamic response.

For the MIL configuration, the control code of the plant was exported as PLCOpenXML and parsed into the simulation model. It was compared with the real laboratory setup to assess its accuracy. The method generates both the physical plant using components of the Modelica Standard Library and the control logic in form of a StateGraph. Fig. 1 illustrates the state-graph-based control in OpenModelica.

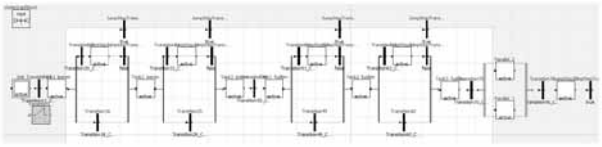


Figure 1: StateGraph-based Control (MIL).

At the top, a root component manages the overall process. The sequence begins with an initialization step, followed by timed transitions and conditional branches

guiding the system through different states. Some transitions ensure consistency, while others determine process flow. Alternative and parallel branches handle different conditions and visualization elements. The cycle concludes with a loop returning to the initial state, ensuring continuous operation. While not all connections are visible, they are integrated within the model.

The simulated responses closely matched real-world plant behavior, further validating model accuracy. The results of the controlled filling levels with MIL are displayed in Fig. 2.

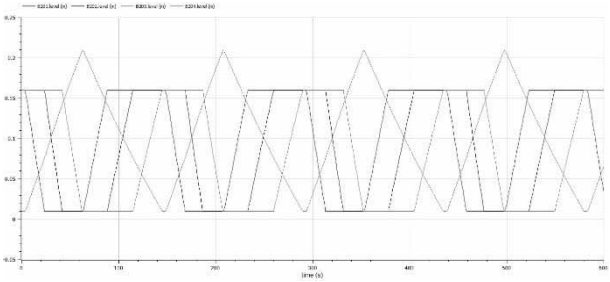


Figure 2: Results of the MIL simulation.

The x-axis represents time in seconds, and the y-axis shows values, with fill levels recorded in meters. The fill levels for "B201" (red), "B202" (blue), "B203" (green), and "B204" (violet) start at 0.16 m, with the dosing tanks emptying sequentially while "B204" fills evenly. The graphs are nearly linear with minimal fluctuations, reaching a maximum of 0.21m for "B204." The dosing tanks are refilled immediately without idle times. The lower limit for all tanks is 0.01m, where the low-level sensor stops signaling, ending the emptying process. This cycle repeats four times, showing consistent behavior. Compared to the real system, the behavior is nearly identical, with differences only in fill levels and cycle durations. Unlike the Open-Loop control, this Closed-Loop system exhibits smooth, repeating cycles without step changes.

To complement the KPI-based accuracy assessment, a series of case studies were conducted to examine how well the simulation models represented different plant configurations. The studies involved modifying specific process components, such as adjusting pump capacities and altering pipeline diameters. The resulting simulations accurately reflected expected process variations, confirming that the generated models were sufficiently detailed and adaptable to different operational conditions.

Furthermore, sensitivity analyses were conducted to determine the robustness of the models under varying operational conditions. The sensitivity tests examined the impact of sensor noise, parameter uncertainties, and control system delays. The results indicated that the models reliably captured dynamic system behavior and control interactions, ensuring their suitability for predictive analysis and system optimization. The robustness of the methodology was further demonstrated by testing various fault scenarios, such as sensor failures and actuator malfunctions. The models successfully replicated fault-induced deviations in plant behavior, highlighting their potential application in fault detection and predictive maintenance.

5.2 Computational Efficiency

The computational performance of the generated models was evaluated by analyzing simulation runtime, memory consumption, and model complexity. The OpenModelica-based implementation demonstrated efficient execution, with simulation runtimes remaining within acceptable limits.

Model generation times were also assessed to determine efficiency gains compared to manual modelling approaches. In particular, the structured data extraction and transformation process enabled seamless integration between engineering and simulation environments, eliminating the need for manual model adjustments. The reduction in modelling time is especially beneficial for complex process plants, where traditional model development often involves extensive iteration and debugging.

To further evaluate computational efficiency, scalability tests were conducted by increasing the model complexity and analyzing performance impacts. The methodology successfully handled larger process models with multiple interconnected units, demonstrating linear scalability with respect to computational resource requirements.

5.3 Applicability in Industrial Scenarios

The structured automation-based approach was found to be highly beneficial for reducing engineering workload and improving model maintainability. Furthermore, the integration with AutomationML ensured interoperability with existing engineering tools, enhancing usability in industrial workflows.

The ability to rapidly generate and update simulation models based on real-time plant data is a key advantage in dynamic industrial settings. Key improvements include reduced human intervention, enhanced data consistency, and seamless integration with control logic.

One of the identified challenges in industrial application was the need for standardized interfaces between different engineering platforms. While AutomationML provided a solid foundation for data exchange, further efforts are required to enhance compatibility with proprietary engineering tools used in specific industries and also the compatibility with other simulation tools. Future research could focus on extending the methodology to support additional industrial standards and integrating machine learning techniques for automated model refinement.

In conclusion, the evaluation results confirm that the proposed methodology provides a reliable and efficient approach for automated simulation model generation. The methodology demonstrated high accuracy, computational efficiency, and broad applicability, making it a valuable tool for process engineers and automation specialists.

6 Conclusion and Future Work

This paper presented a methodology for the automated generation of simulation models based on structured plant engineering data. By leveraging AutomationML for data representation and OpenModelica for simulation execution, the approach successfully addressed the challenges associated with manual model generation, including time consumption, inconsistencies, and dependency on expert knowledge.

The proposed method demonstrated high accuracy, efficiency, and scalability. The evaluation confirmed that the generated models closely matched real plant behavior, required significantly less development effort, and integrated seamlessly into industrial workflows. Additionally, the automated transformation process ensured a consistent and standardized model structure, enhancing reusability and interoperability with existing engineering tools.

Despite these achievements, several areas for future research remain. One key challenge is improving the integration of automated simulation models with proprietary engineering tools and industrial software ecosystems. Further work is needed to expand compatibility with different engineering standards and to develop adaptive interfaces that facilitate seamless data exchange between simulation environments and industrial control systems.

Another promising direction is the application of machine learning techniques to enhance model accuracy and adaptability. By incorporating AI-driven parameter estimation and real-time model refinement, future implementations could further reduce the need for manual intervention and improve predictive capabilities. Additionally, the integration of real-time plant data into digital twins could enable continuous model updates, supporting predictive maintenance and process optimization.

Furthermore, scalability remains an important aspect for large-scale industrial applications. While the current methodology has demonstrated its ability to handle modular process plants, further research could focus on optimizing computational performance for highly complex systems with thousands of interconnected components.

In conclusion, the proposed methodology provides a reliable and efficient solution for automating the generation of process simulation models. By addressing remaining challenges and integrating emerging technologies, future developments could further enhance its applicability, making it an essential tool for engineering, process optimization, and industrial automation.

7 Acknowledgements

This contribution is funded as part of the research project AVEDAS. This project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.

References

[1] M. Barth and A. Fay, "Automated generation of simulation models for control code tests," *Control Engineering Practice*, vol. 21, no. 2, pp. 218–230, 2013.

[2] M. Oppelt, G. Wolf, O. Drumm, B. Lutz, M. Stöß, and L. Urbas, "Automatic Model Generation for Virtual Commissioning based on Plant Engineering Data," in *IFAC Proceedings Volumes*, 2014, pp. 11635–11640.

[3] P. P. Schmidt *et al.*, "Validierung von Steuerungscode mit Hilfe automatisch generierter Simulationsmodelle," *at - Automatisierungstechnik*, vol. 63, no. 2, pp. 111–120, 2014.

[4] M. Azangoo *et al.*, "A Methodology for Generating a Digital Twin for Process Industry: A Case Study of a Fiber Processing Pilot Plant," *IEEE Access*, vol. 10, pp. 58787–58810, 2022.

[5] A. Deuter and F. Pethig, *The Digital Twin Theory. Eine neue Sicht auf ein Modewort* (in de). [Online]. Available: https://publica.fraunhofer.de/handle/publica/257277

[6] R. Drath, *AutomationML*. Berlin/Boston: De Gruyter Oldenbourg, 2022.

[7] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. New Jersey: John Wiley & Sons, 2011.

[8] S. Boschert and R. Rosen, "Digital Twin—The

Simulation Aspect," in *Mechatronic Futures*, 2016, pp. 59–74.
[9] M. Barth, "Automatisch generierte Simulationsmodelle verfahrenstechnischer Anlagen für den Steuerungstest," Helmut-Schmidt-Universität, Universität der Bundeswehr Hamburg, Hamburg.
[10] E. Arroyo, M. Hoernicke, P. Rodríguez, and A. Fay, "Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams," *Computers & Chemical Engineering*, vol. 92, pp. 112–132, 2016.
[11] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML - the glue for seamless automation engineering," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Hamburg, Germany, 2008, pp. 616–623, doi: 10.1109/ETFA.2008.4638461.
[12] P. Fritzson, "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development," *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 41, no. 4, pp. 241–295, 2020.
[13] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*, 2017, pp. 85–113.
[14] C. Härle, "Automatisierte Generierung, Adaption und Rekonfiguration von Co-Simulationen für modulare Produktionsanlagen," Karlsruher Institut für Technologie (KIT), Karlsruhe.
[15] T. Holm, L. Christiansen, M. Goring, T. Jager, and A. Fay, "ISO 15926 vs. IEC 62424 – Comparison of plant structure modeling concepts," *IEEE ETFA Conference*, pp. 1–8, 2012.
[16] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.
[17] M. Oppelt, M. Barth, M. Graube, and L. Urbas, "Enabling the integrated use of simulation within the life cycle of a process plant," *IEEE Industrial Informatics Conference*, pp. 49–55, 2015.
[18] P. Novak, P. Douda, J. Vyskocil, and B. Wally, "PyAML: Enhancing AutomationML for Advanced Virtualization," in *IEEE ETFA Conference*, 2021, pp. 1–8.
[19] P. Novak, F. J. Ekaputra, and S. Biffl, "Generation of Simulation Models in MATLAB-Simulink Based on AutomationML Plant Description," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7613–7620, 2017.
[20] M. Oppelt, L. Urbas, and G. Wolf, Eds., *Towards an integrated use of simulation within the life cycle of a process plant*, 2015.
[21] Open Source Modelica Consortium. "OpenModelica User's Guide." [Online]. Available: https://openmodelica.org/doc/OpenModelicaUsersGuide/OpenModelicaUsersGuide-latest.pdf
[22] S. Sierla *et al.*, "Roadmap to semi-automatic generation of digital twins for brownfield process plants," *Journal of Industrial Information Integration*, vol. 27, p. 100282, 2022.
[23] M. Hoernicke, A. Fay, and M. Barth, "Virtual plants for brown-field projects," in *IEEE ETFA Conference*, 2015, pp. 1–8.
[24] B. Thiele, "Towards a Modelica OPC UA Library for Industrial Automation," in *Proceedings of the 14th Modelica Conference*, 2021, pp. 205–213.
[25] N. Striffler and T. Voigt, "Concepts and trends of virtual commissioning – A comprehensive review," *Journal of Manufacturing Systems*, vol. 71, pp. 664–680, 2023.
[26] Technische Hochschule Nürnberg Georg Simon Ohm. "DigiZ." [Online]. Available: https://www.th-nuernberg.de/einrichtungen-gesamt/wissenschaftliche-und-forschungskooperationen/nuremberg-campus-of-technology/automatisierungstechnik/projekte/digiz/
[27] *Virtuelle Inbetriebnahme Einführung der virtuellen Inbetriebnahme in Unternehmen*, VDI/VDE 3693, 2016.
[28] *Modellierung und Simulation*, VDI 4465-1, 2016.
[29] *Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme*, DIN EN IEC 62714, 2019.
[30] PLC open, *PLCopen XML now available as IEC 61131-10*, 2019.
[31] M. Shafto and M. Conroy, *DRAFT Modeling, Simulation, Information Technology & Processing Roadmap*. NASA, 2010.
[32] VDMA, *Leitfaden Virtuelle Inbetriebnahme: Handlungsempfehlungen zum wirtschaftlichen Einstieg*, 2020.
[33] M. Ramonat and A. Fay, "Method for Automatic Simulation Model Calibration and Maintenance for Brownfield Process Plants," in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, Helsinki, Finland, 2023, pp. 1–6, doi: 10.1109/ISIE51358.2023.10227947.
[34] M. Ramonat, F. C. Kunze, F. Gehlhoff, and A. Fay, "Identifying Root-Causes of Deviations between Simulation and Real Plant Data based on an Adaptive Causal Directed Graph," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Padova, Italy, 2024, pp. 1–8, doi: 10.1109/ETFA61755.2024.10710690.
[35] M. Ramonat, J. Hujer, F. Gehlhoff, and A. Fay, "An Assistance System for Simulation Model Adaptation and Parameter Validation," in *3rd IEEE Industrial Electronics Society Annual Online Conference (ONCON)*, 2024, pp. 1–6.

A drilling force model for use in multibody system simulation environments

Robert Reiser[1*]

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR),
Münchener Straße 20, 82234 Weßling, Germany; *firstname.lastname@dlr.de

Abstract. In this work, a drilling force model for the use in multibody system simulations is presented. The model is based on drilling force calculations, collision detection, a contact model, and a method for applying the forces to the multibody environment. The model can be used in two ways. First, it can be used to calculate drilling forces based on a given velocity and angular velocity of the drill. Second, it can be integrated into a model to interact as a body in a multibody scenario and be subjected to an external force and angular velocity.

Introduction

Drilling processes are widely used and occur in many different areas, such as manufacturing or construction. Several models exist to calculate the forces during a drilling process. However, they usually cannot be used for body interaction in a multibody context (e.g. if the drilling forces are to be applied to the robot in order to optimize the robot-assisted drilling process).

Therefore, in this work, a drilling force model is presented that can be used in multibody simulations. The goal is to use it in system simulation environments. This allows the creation of multi-domain models. For example, the drilling model can be combined with the dynamics model of a robot and other models such as the control system or the motors of the robot. This can then be used to analyze the influence of the drilling process on the robot dynamics or energy consumption. The object-oriented and multi-domain modeling language *Modelica* [7] is used as system simulation environment. Models are built based on the multibody components [9] from the *Modelica Standard Library* (MSL) [8].

The developed drilling force model is based on drilling force calculations, collision detection, a contact model, and a method for applying the forces to the multibody environment.

In the next section, the state of the art is introduced. In Section 2, the developed drilling force model is presented. Examples are given in the following example. Finally, the results are discussed and future developments for the model are considered.

1 State of the art

In this section, the basics are introduced. These include the forces in the drilling process and collision detection and resolution for multibody simulation environments.

1.1 Forces in the drilling process

Drilling is a machining process. Because the geometry of the cutting edges is known, it is classified as a *machining process with a geometrically defined cutting edge* (similar to turning and milling) [5]. There are several models for calculating the forces in the drilling process. Examples are the works of Dietrich (2016) [2] and Fritz and Schulze (2015) [5]. The most suitable parameters for a drilling process are often determined using table values [3]. In this work the model of Dietrich (2016) is applied [2]. It is used to calculate the cutting force F_c and the feed force F_f from a given rotational frequency n and feed per rotation f.

First, the feed per rotation per cutting edge f_z as well as the stress thickness h, the stress width b, and the stress cross-section A are calculated. The inputs are the number of cutting edges z_E, the tip angle of the drill σ, and the drill diameter d (see Figure 1) [2].

$$f_z = \frac{f}{z_E} \tag{1}$$

$$h = f_z \cdot \sin(\frac{\sigma}{2}) \tag{2}$$

$$b = \frac{d}{2 \cdot \sin(\frac{\sigma}{2})} \tag{3}$$

$$A = h \cdot b \tag{4}$$

The specific cutting force k_c depends on the stress thickness h and the material dependent parameters z and $k_{c1.1}$ [2]. The latter is the specific cutting force of a given material for $h = 1$ mm. There are also correction factors which are not considered in this work.

$$k_c = \frac{0.001^z}{(f_z \cdot \sin(\frac{\sigma}{2}))^z} \cdot k_{c1.1} \tag{5}$$

The cutting force per cutting edge F_{cz}, the cutting force (for the entire drill) F_c, and the feed force F_f are then calculated as follows [2].

$$F_{cz} = \frac{d \cdot f_z}{2} \cdot k_c \tag{6}$$

$$F_c = z_E \cdot \frac{d \cdot f_z}{2} \cdot k_c \tag{7}$$

$$F_f = z_E \cdot F_{cz} \cdot \sin(\sigma) \tag{8}$$

1.2 Collision detection

The basis for calculating drilling forces in a multibody environment is the interaction between the bodies. This requires collision detection. Common algorithms for collision detection are the Gilbert-Johnson-Keerthi distance algorithm (GJK) [6] and the Minkowski Portal Refinement algorithm (MPR) [12]. Both algorithms provide the penetration depth of two colliding bodies.

There have been several approaches to enabling collision detection for the system simulation environment *Modelica*. Most of them extend *Modelica* with an external library for collision detection. An overview can be found in Reiser and Reiner (2023) [11].

1.3 Multibody contacts

Based on the collision detection, contact forces are now calculated between the bodies to prevent them from penetrating each other. This is also known as collision response. There are two main types: impulse-based and penalty-based collision response. Only the latter will be considered in this work. Contact forces are calculated based on the penetration depth. [4]

A common model is the Kelvin-Voigt contact force model. It uses a spring-damper element to calculate the normal force F_N based on the penetration s [4]:

$$F_N = k \cdot s + d \cdot \dot{s} \tag{9}$$

where k is the spring stiffness and d the damping factor. In addition, a friction force dependent on F_N acts in the tangential direction (not considered in this work).

Multiple works have dealt with contact forces between bodies in Modelica multibody environments. One example is Buse et al. (2023) [1]. Further works are listed in Reiser and Reiner (2023) [11].

2 A drilling force model for multibody environments

The developed model for drilling forces in multibody simulation environments is presented in this section.

2.1 Overview

A key component of the model is collision detection. This is used to determine the drill diameter and the penetration depth. The model has two parts.

In the **kinematic model**, the rotational frequency and the feed per rotation are provided and the model calculates the drilling forces. These are not used for body interaction. However, they can be used in a simulation to check if certain forces are exceeded.

In contrast, the drilling forces in the **dynamic model** are applied to the bodies in the multibody environment. This allows, for example, force-controlled robot-assisted drilling processes to be simulated together with robot dynamics models.

2.2 Determination of the drilling diameters

The first step is to determine the drilling diameters during the drilling process. If the drill does not come out of the material on the opposite side during drilling, only the outer diameter d_O is relevant. Otherwise, the inner diameter d_I is also important.

Collision detection is used to determine the drilling diameters during the process. The MPR algorithm is used in the developed model. It is connected to the multibody model in Modelica via an external library, similar to the method in Buse et al. (2023) [1].

Figure 1 shows the resulting drilling diameters for several states during the process. Collision detection is used to determine both the penetration depth of the drill and if the drill cone is penetrating the workpiece, is within the workpiece, or is leaving the workpiece. Depending on the state, different calculations are used to determine the diameters (see Figure 1). In addition, it is necessary to distinguish if the material is thicker than the height of the drill cone.

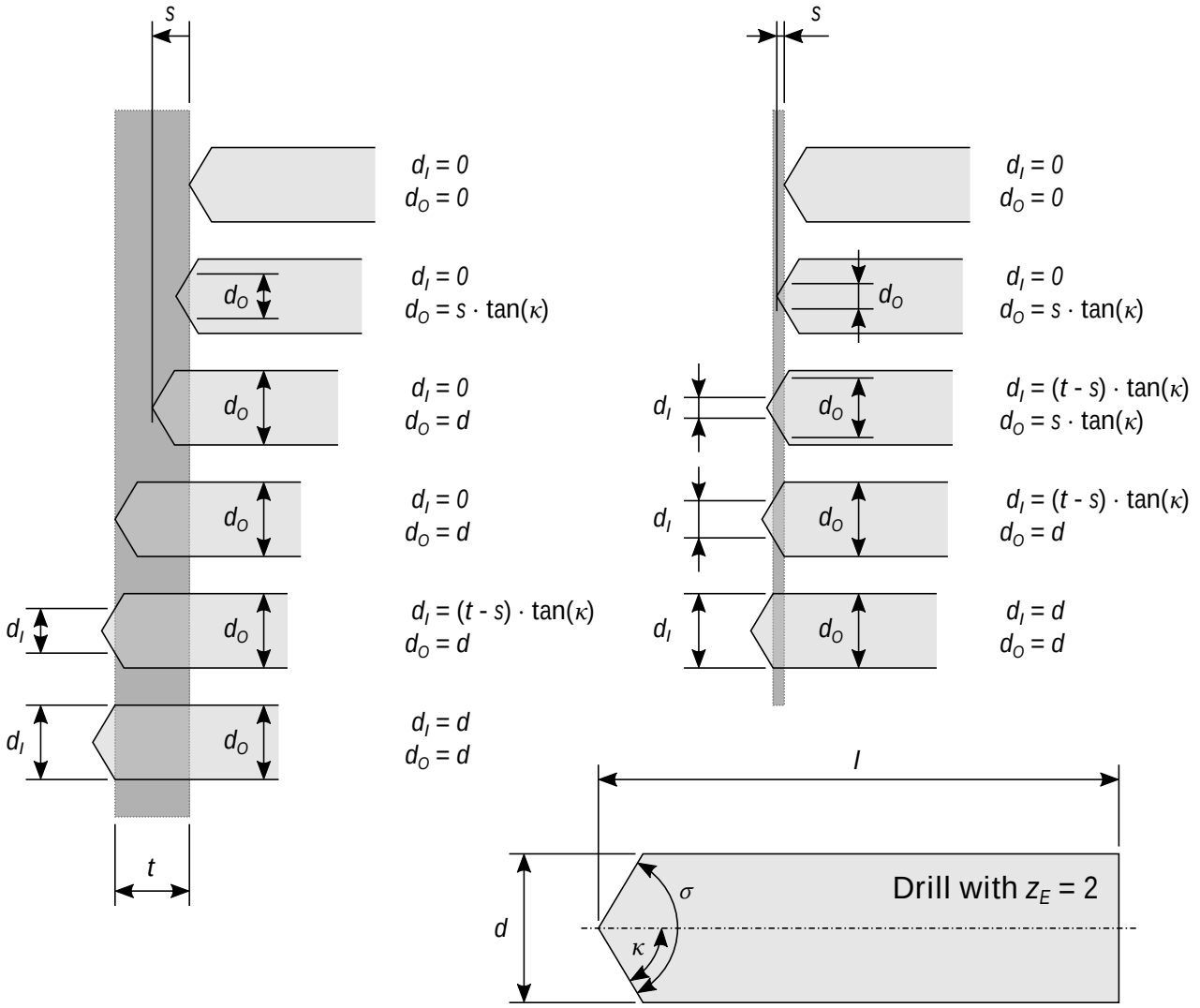


Figure 1: Drilling diameters for several states during the drilling process. A distinction is made as to whether the material is thicker than the height of the drill tip (left) or not (right). The dimensions of a drill are shown in the bottom left.

2.3 Kinematic model

The kinematic model uses the calculated drilling diameters from Section 2.2. Taking into account the drilling diameters, the cutting force per cutting edge F_{cz} is calculated with a modification of Equation 6 [2]:

$$F_{cz} = \frac{(d_O - d_I) \cdot f_z}{2} \cdot k_c \tag{10}$$

Based on the cutting force, the feed force F_f is calculated using Equation 8. The kinematic part of the developed drilling force model has already been used in the *MFlex 2025* project [10].

2.4 Dynamic model with applied forces

A different approach is used for the dynamic model. The drilling forces are not applied directly to the bodies in the model. Instead, a contact model is applied to the drill body. This allows the drill to be pressed against the table even when it is not rotating. The collision detection provides the penetration depth s for the contact model.

The idea now is to make the feed force for the drilling F_f equal to the contact normal force F_N (see Equation 9). The cutting force per cutting edge F_{cz} can

then be calculated from this:

$$F_f = F_N \quad (11)$$
$$\Rightarrow z_E \cdot F_{cz} \cdot \sin(\sigma) = k \cdot s + d \cdot \dot{s} \quad (12)$$
$$\Rightarrow F_{cz} = \frac{k \cdot s + d \cdot \dot{s}}{z_E \cdot \sin(\sigma)} \quad (13)$$

Next, the feed per cutting edge f_z is calculated from the cutting force per cutting edge. This is done by combining Equation 10 and Equation 5:

$$F_{cz} = \frac{(d_O - d_I) \cdot f_z \cdot 0.001^z}{2 \cdot (f_z \cdot \sin(\frac{\sigma}{2}))^z} \cdot k_{c1.1} \quad (14)$$
$$\Rightarrow F_{cz} = \frac{(d_O - d_I) \cdot f_z^{1-z} \cdot 0.001^z}{2 \cdot \sin(\frac{\sigma}{2})^z} \cdot k_{c1.1} \quad (15)$$
$$\Rightarrow f_z = (\frac{F_{cz} \cdot 2 \cdot \sin(\frac{\sigma}{2})^z}{(d_O - d_I) \cdot 0.001^z \cdot k_{c1.1}})^{\frac{1}{1-z}} \quad (16)$$

The feed per rotation f is then calculated based on f_z and the number of cutting edges z_E:

$$f = z_E \cdot f_z \quad (17)$$

This feed per rotation is now used to calculate the target velocity v_{tar} of the drill, based on the angular velocity ω of the drill, which is derived from the rotational frequency n of the drill.

$$v_{tar} = f \cdot n = f \cdot \frac{\omega}{2 \cdot \pi} \quad (18)$$

Finally, the target drill depth s_{tar} can be calculated by integrating the target velocity v_{target}:

$$s_{tar} = \int v_{tar} \, \mathrm{d}t \quad (19)$$

This target drill depth is used as the reference depth for the calculation of the contact force. Equation 9 is modified to calculate the contact normal force F_N:

$$F_N = k \cdot \Delta s + d \cdot \Delta \dot{s} \quad (20)$$
$$\Rightarrow F_N = k \cdot (s - s_{tar}) + d \cdot v \quad (21)$$

The penetration depth s of the drill and the drill velocity v are still used, but now the contact surface of the workpiece moves based on the drilling process. Its position is defined by s_{tar}.

A substitute model of the developed drilling force model is shown in Figure 2. It can be seen as a series connection of a spring-damper element and a damper element. The former describes the contact normal force and the latter the movement of the drill as a function of the rotational frequency and the drilling diameters.

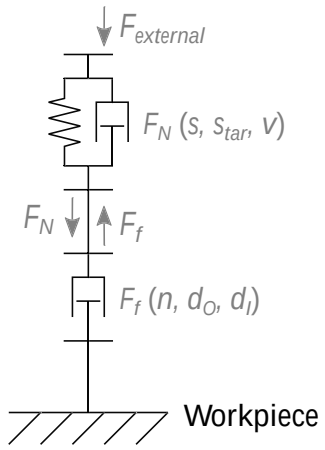


Figure 2: A substitute model of the drilling force model. An external force is applied to the spring-damper element representing the contact model. The contact force is equal to the feed force of the drilling force model, here represented by a damper element, with the damping force dependent on the rotational frequency and the drill diameters.

Figure 3 shows a more detailed representation of the drilling force model. It consists of three parts:

- A *DrillGeometry* model to calculate the drilling diameters and the penetration depth of the drill based on an external collision detection library (see Section 1.2 and Section 2.2).
- The *DrillDepthCalculation* model to calculate the target position of the drill. The calculation is based on Equations 11 to 19.
- A *ContactForce* model to calculate the contact force based on the actual and target position and velocity of the drill (see Equation 20 and 21).

As mentioned above, the inputs to the drilling force model are the velocity v and the rotational frequency n of the drill. In addition, the position r and orientation T of the drill are required for the collision detection in the *DrillGeometry* model. The same applies to the workpiece, described by r_W and T_W. Collision detection also requires the geometry of the drill and the workpiece.

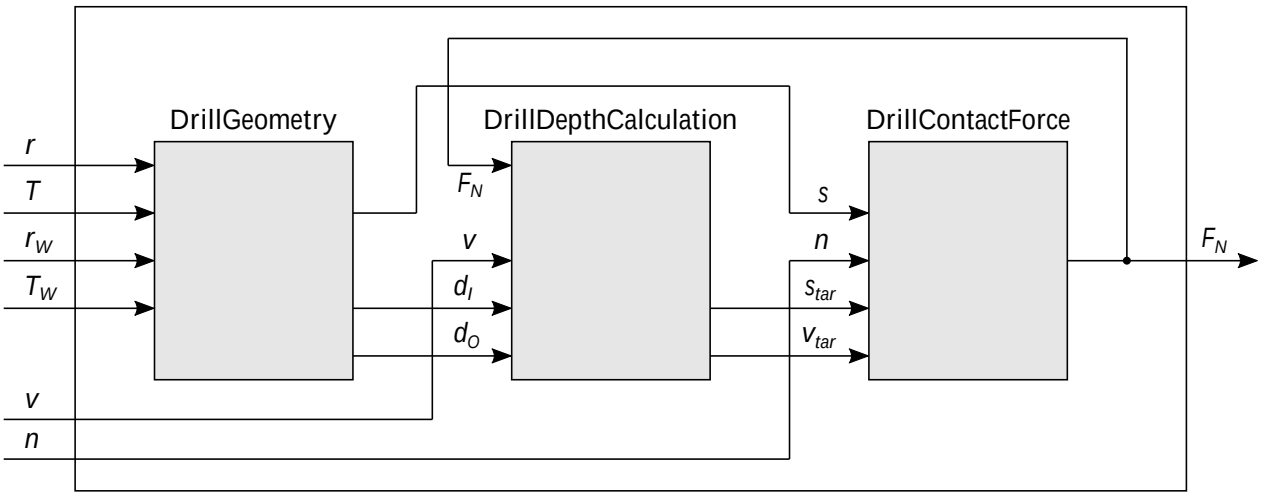


Figure 3: The developed drilling force model with all models included. The drilling diameters and the drill penetration are calculated in the *DrillGeometry* model. The *DrillDepthCalculation* model provides the target position for the drill. The contact force is calculated in the *ContactForce* model based on the actual and target position of the drill.

3 Applications

In this section, two drilling processes are shown. One process consists of drilling in a thin plate. The kinematic model is used to calculate the forces involved. In the other example, the dynamic model is applied for a drilling process. This involves drilling into a material with a high material thickness.

3.1 Kinematic drilling in a thin plate

In the first example, the kinematic model is used for the drilling process of a thin plate. The plate is thinner than the drill cone height of the drill. The material *34 CrMo4* is used for the workpiece (see [2, p. 19]):

$$k_{c1.1} = 2240 \frac{\mathrm{N}}{\mathrm{mm}^2} \tag{22}$$

$$z = 0.21 \tag{23}$$

A rotational frequency of $n = 9.28 \frac{1}{\mathrm{s}}$ and a velocity of $v = 1.672 \frac{\mathrm{mm}}{\mathrm{s}}$ are used. The drill diameter is $d = 16$ mm with an angle of 118 degree. From this, the feed per cutting edge per rotation f_z can be calculated, resulting in a specific cutting force k_c of:

$$f_z = \frac{f}{z_E} = \frac{v}{z_E \cdot n} = \frac{1.672 \frac{\mathrm{mm}}{\mathrm{s}}}{2 \cdot 9.28 \frac{1}{\mathrm{s}}} \approx 0.090 \text{ mm} \tag{24}$$

$$\Rightarrow k_c = \frac{0.001^z}{(f_z \cdot \sin(\frac{\sigma}{2}))^z} \cdot k_{c1.1} \approx 3836 \frac{\mathrm{N}}{\mathrm{mm}^2} \tag{25}$$

The calculated maximum cutting force is:

$$F_{c,\,max} = z_E \cdot \frac{d \cdot f_z}{2} \cdot k_c \approx 5524 \text{ N} \tag{26}$$

However, this value is not reached because the difference between the inner and outer drilling diameter is much smaller. The results are presented in Figure 4. Both the cutting force and the feed force and

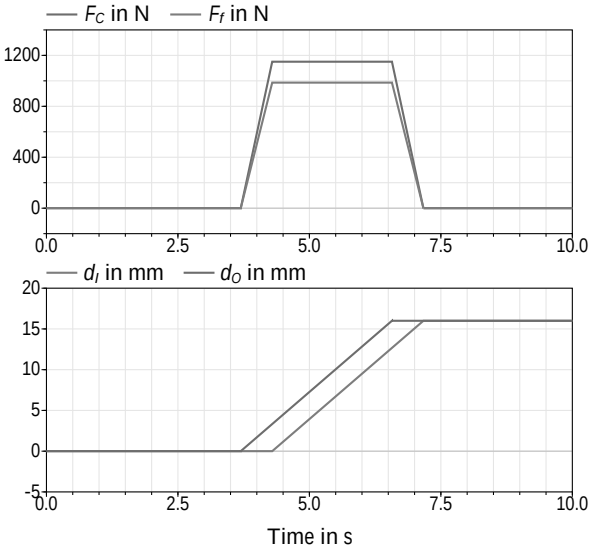


Figure 4: Results for the drilling process of a thin plate. The cutting force and the feed force are shown above. The drilling diameters are shown at the bottom.

both drilling diameters are shown. The maximum cutting force is $F_c = 986$ N and the maximum feed force $F_f = 2917$ N (see Figure 4).

The simulation was run on a desktop computer with an Intel Core i7-11700K processor. A *Rkfix2* fixed step solver with a fixed step size of 0.001 seconds was used. The computation time was 0.11 seconds for a simulation time of 10 seconds (real-time factor of 0.011). The simulation is real-time capable.

3.2 Drilling process using the dynamic model

The second example shows the dynamics model. A *Modelica* model has been created for a drilling process. As in Section 3.1, the material used is *34 CrMo4*. The model is shown in Figure 5. An external force and angular velocity are generated with source blocks from the MSL. The drilling force model is integrated into one block. This block is connected to a revolute joint, which in turn is connected to a prismatic joint. This allows a linear motion and a rotation. The drilling forces are applied directly to the frame of the model.

The workpiece in the form of a plate is represented by a second block. Both the *DrillGeometry* model of the drilling force model and this workpiece are modeled

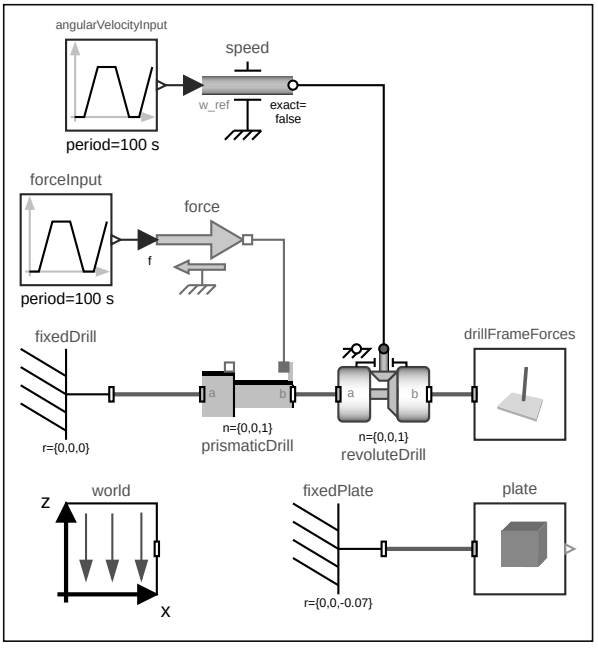


Figure 5: The *Modelica* model of the drilling process based on the dynamics model. The drilling model applies the forces directly to the frame. An external force and angular velocity are also applied to the frame.

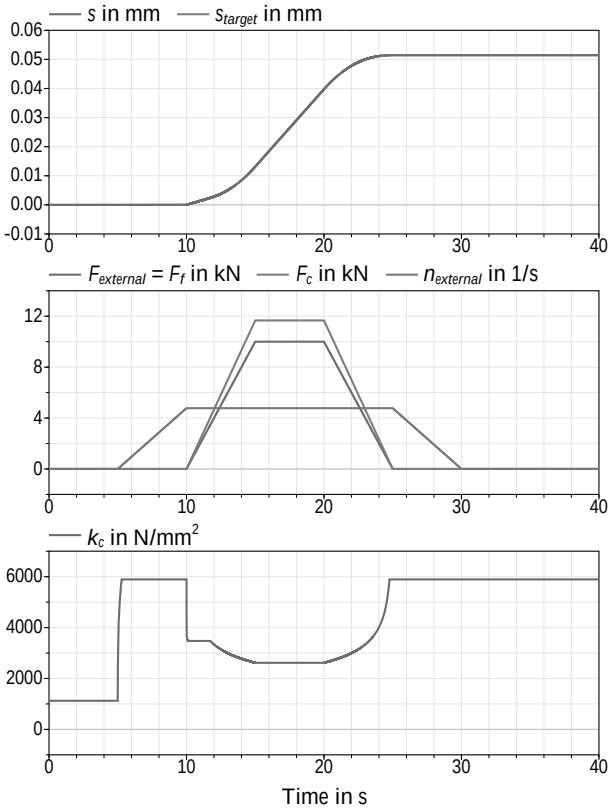


Figure 6: The results for the drilling process based on the dynamics model. The image shows the actual and target position for the drill (top), the external force, the cutting force and the rotational frequency (middle), and the specific cutting force (bottom).

with components from a contact detection library. This allows multiple workpieces to be drilled in a multibody scenario without the need for predefined contact pairs.

The results are shown in Figure 6. The variable step solver *Dassl* was used with a tolerance of 1e-8. It took 0.33 seconds to run the 40 seconds simulation. The simulation was also run on a desktop computer with an Intel Core i7-11700K processor. Fixed step solvers are not suitable because the contact between the drill and the workpiece is very stiff and therefore very small step sizes are required at some points.

The actual and target position of the drill fit together well. The feed force and cutting force results match the manually calculated values. The specific cutting force is not defined for the full range of feeds (see Equation 5). It is therefore limited. When the feed is less than 1 mm, the specific cutting force does not increase any further. This increases the stability of the model.

4 Conclusion

In this work, a drilling force model has been developed. It consists of two parts. A kinematic model can be used to calculate the forces of the drilling process without applying them. The drilling forces in the dynamics model are applied to the bodies in the multibody environment and can be used for body interaction.

The drilling force model is based on drilling force calculations, collision detection, a contact model, and a method for applying the forces to the multibody environment. It can be used in system simulation environments such as *Modelica*. The capability of the developed model has been successfully demonstrated in two example drilling processes. However, the model has some limitations: Holes must be drilled at a 90 degree angle to the workpiece without tangential motion. And friction forces are not taken into account (for both the contact force and the drilling forces).

Possible further developments include the integration of a friction model. This would allow forces in the tangential direction to be considered for the contact model. Friction forces could also be included in the drilling force calculation. Torques from the drilling process could also be considered, for example to model the drive of the drill spindle. In addition, the combination of the drilling force model with force controlled robot models could be investigated.

Acknowledgement

I would like to thank my colleagues Fabian Buse, Bernhard Thiele, Tobias Treichl, and Dirk Zimmer for fruitful discussions regarding the drilling force model.

Parts of this model have been developed in the *MFlex 2025* project funded by the German Federal Ministry for Economic Affairs and Climate Action (Förderkennzeichen 20X1720B).

References

[1] Buse, Fabian, Antoine Pignède, and Stefan Barthelmes (2023). “A Modelica Library to Add Contact Dynamics and Terramechanics to Multi-Body Mechanics”. In: *Proceedings of the 15th International Modelica Conference*, pp. 433–442. DOI: 10.3384/ecp204433.

[2] Dietrich, Jochen (2016). *Praxis der Zerspantechnik. Verfahren, Werkzeuge, Berechnung.* 12th ed. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-14052-6.

[3] Fischer, Ulrich et al. (2008). *Tabellenbuch Metall.* 44th ed. Haan-Gruiten: Verlag Europa-Lehrmittel. ISBN: 978-3-8085-1724-6.

[4] Flores, Paulo (2022). “Contact mechanics for dynamical systems: a comprehensive review”. In: *Multibody System Dynamics* 54.2, pp. 127–177. DOI: 10.1007/s11044-021-09803-y.

[5] Fritz, A. Herbert and Günter Schulze (2015). *Fertigungstechnik.* 11th ed. Berlin and Heidelberg: Springer Vieweg. ISBN: 978-3-662-46554-7.

[6] Gilbert, Elmer G., Daniel W. Johnson, and S. Sathiya Keerthi (1988). “A fast procedure for computing the distance between complex objects in three-dimensional space”. In: *IEEE Journal on Robotics and Automation 4.2*, pp. 193–203. DOI: 10.1109/56.2083.

[7] Modelica Association (2017). *Modelica - A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4. Tech. Rep.* Linköping: Modelica Association. URL: https://modelica.org/documents/ModelicaSpec34.pdf

[8] Modelica Association (2020). *Modelica Standard Library. Version 4.0.0.* Modelica Association. URL: https://github.com/modelica/ModelicaStandardLibrary

[9] Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003). “The New Modelica MultiBody Library”. In: *Proceedings of the 3th International Modelica Conference*, pp. 311–330.

[10] Reiser, Robert et al. (2022). “Real-time simulation and virtual commissioning of a modular robot system with OPC UA”. In: *ISR Europe 2022. 54th International Symposium on Robotics (Munich)*. Munich: VDE Verlag. ISBN: 978-3-8007-5891-3.

[11] Reiser, Robert and Matthias J. Reiner (2023). “Modeling and simulation of dynamically constrained objects for limited structurally variable systems in Modelica”. In: *Proceedings of the 15th Int. Modelica Conference*, pp. 151–158. DOI: 10.3384/ecp204151.

[12] Snethen, Gary (2008). “Xenocollide: Complex collision made simple”. In: *Game programming gems 7*. Ed. by Scott Jacobs. Boston, MA: Charles River Media, pp. 165–178. ISBN: 978-1-58450-527-3.

Development of a digital twin application for an industrial robot with ROS2 and OPC UA

Ahmed Ibrahim Almohamed[1], Lisa Ollinger[1*]

[1]Institute for Manufacturing Technology and Material Sciences, Ulm University of Applied Sciences, Prittwitzstraße 10, 89075 Ulm, Germany; **lisa.ollinger@thu.de*

Abstract.
A digital twin is defined as a virtual representation of a physical object or system that is utilized to simulate, monitor, and optimize its real-life counterpart. The establishment of a connection between the real entity and its digital representation constitutes a significant challenge in the context of Industry 4.0. In the present study, a digital twin of a 6DoF industrial robotic arm is created using vendor-independent, openly-available standards for robot control and communication. Therefore, ROS2 is used for the purposes of simulation, control, and optimization of both twins of the system. The utilization of OPC UA plays a pivotal role in establishing connections with other components within the automation environment. The developed system is then used for virtual commissioning use cases that are supported with a specifically designed graphical user interface.

Introduction

In the contemporary era, we find ourselves in the midst of Industry 4.0, a period characterized by the digital transformation of systems. The advent of Industry 4.0 has given rise to the development of concepts such as Digital Twins (DT) [1]. The importance of DT, characterized by cyber-physical integration, is increasingly emphasized by both academia and industry. Primary applications of DTs across various sectors include design/planning, optimization, maintenance, safety, decision-making, remote access, and training, among others. DTs serve as an effective tool for companies to enhance their competitiveness, productivity, and efficiency. [1]

The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications [2]. It encompasses a wide range of components, including drivers, state-of-the-art algorithms, and powerful developer tools. Its current version ROS2 is an essential component for any robotics project, providing open-source tools necessary for successful execution [2]. Additionally, ROS2 can be regarded as a substitute for conventional proprietary robot programming environments [3].

The integration of ROS2 and DT signifies a substantial advancement in the domains of robotics and Industry 4.0. ROS2 plays a pivotal role in creating and operating DTs in robotics, offering a foundational framework for real-time data exchange, synchronization, and control. A DT, leveraging ROS2 communication protocols such as publishers, subscribers, and services, can replicate the physical robot's state and behavior, enabling real-time data flow. Enhanced real-time capabilities in tools such as Gazebo, Rviz, and ROS2 further facilitate DT development by enabling simulation, testing, and optimization in a virtual environment before deployment to the physical system [4][5].

While ROS2 excels in building robot systems, the requirements of Industry 4.0 pose additional challenges. First, robot systems must adhere to industrial standards, ensuring seamless collaboration with other industrial devices. Second, a robot system must serve as a flexible, shared experimental foundation for multiple Industry 4.0 technologies, use cases, and projects. This allows robots to seamlessly join and exit testbeds for efficient task execution [6].

In this context, OPC UA (Open Platform Communications Unified Architecture) acts as a digital thread to standardize and manage the ROS2 layer across different system layers [7]. This integration ensures secure, reliable, and interoperable communication, making it a cornerstone for robotics systems within Industry 4.0.

The paper introduces first the conceptual framework for integrating ROS2 and OPC UA in a digital twin application. This is followed by a detailed system architecture, including the physical robot setup, simulation environment, and communication interfaces. The realization section describes the practical implementation of the digital twin, including path planning and graphical user

interface (GUI) functionalities. Finally, the application in visual commissioning use cases and evaluation results are presented.

1 Concept

The foundational objective of this concept is to facilitate interoperability within robotic DT applications in two ways. Firstly, it ensures the capacity of the system to control the robot via ROS2 and simulate its movement, in addition to facilitating bidirectional data exchange. Secondly, it involves exposing the ROS2's internal functionality to the external environment through the utilization of OPC UA, thereby enhancing the interface between ROS2 and other components within the industry. The implementation of this concept facilitates the enhancement of system modularity and facilitates seamless integration with external systems.

1.1 System Overview

The system comprises three primary components: a robot (KR3 R540) that utilizes the KRC4 controller; an OPC UA server that operates on a host machine and runs the ROS2-KR3 Core in the server's background; and a dashboard, which also runs on the host machine and acts as an OPC UA client (see Figure 1).

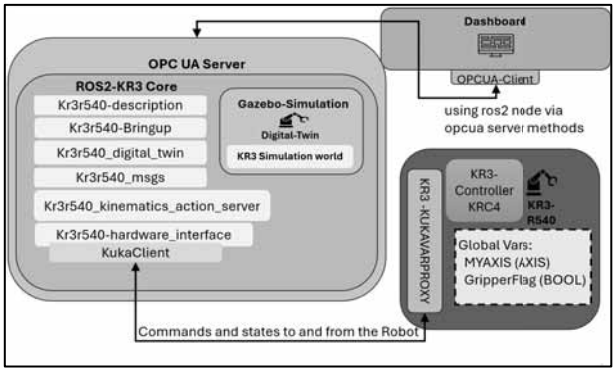


Figure 1: System overview

The KRC4 controller is equipped with KukaVarProxy (KVP), a lightweight TCP/IP server that facilitates the reading and modification of global variables stored on the robot [8]. On the KRC4 controller, two global variables were defined in KRL: GripperFlag, a Boolean (BOOL) flag used to control the state of the gripper, and MYAXIS, an AXIS-typed variable representing the robot's joint positions. The MYAXIS variable is used in PTP motion commands for joint-space control, while GripperFlag toggles the open or closed state of the gripper.

It is important to note that the robot does not include an OPC UA server by default. Instead, the OPC UA server was fully developed and implemented as part of this work. It runs externally on the host machine and exposes key ROS 2 Core functionalities as UA methods, enabling external clients to interact with the robot through a standardized interface.

Communication between the robot and the OPC UA server is handled by a dedicated library, developed as part of this work, based on Boost.Asio [9]. This library performs the serialization and deserialization of messages exchanged with the robot using KukaVarProxy. It is integrated into the kr3r540-hardware-interface, which acts as a bridge between the ROS2-KR3 Core and the physical robot, allowing seamless control and feedback within the ROS 2 ecosystem.

The ROS2-KR3 Core itself is composed of several packages responsible for the robot's main functionalities. These include:

- kr3r540-bringup, which contains the launch scripts to initialize system nodes
- kr3r540-description, which defines the robot's physical configuration in URDF format [10]
- kr3r540-digital-twin, which synchronizes the states of the real and simulated robots
- kr3r540-msgs, which holds custom-defined message and service types [11]
- kr3r540_kinematics_action_server, which implements an inverse kinematics algorithm using the Kinematics and Dynamics Library (KDL) to compute target joint positions for the robot's JointTrajectoryController [12][13]

In addition to real-robot communication, the ROS2-KR3 Core also launches and manages the simulated robot environment using Gazebo Ignition Fortress, allowing for full digital twin operation [14].

The final component of the system is the Dashboard, a cross-platform OPC UA Client application written in C# using the Avalonia UI Framework [15]. This dashboard provides a graphical interface for controlling and monitoring both the real and simulated KR3 R540 robots. It connects to the custom-developed OPC UA Server to invoke ROS2 Core functionalities exposed as UA methods. The system in its entirety, along with all associated documentation, has been made available to the public on our git repository [16].

2 Realization for Kuka KR3

The realization of the DT for the Kuka KR3 robot involves the integration of the physical robot, its simulation model and the necessary control and communication framework to achieve seamless interaction. The system is built using the following components.

2.1 Physical Robot Integration

The kr3r540-hardware-interface transmits read and write commands to the robot via the TCP protocol, leveraging the KukaClient layer (see Figure 2).

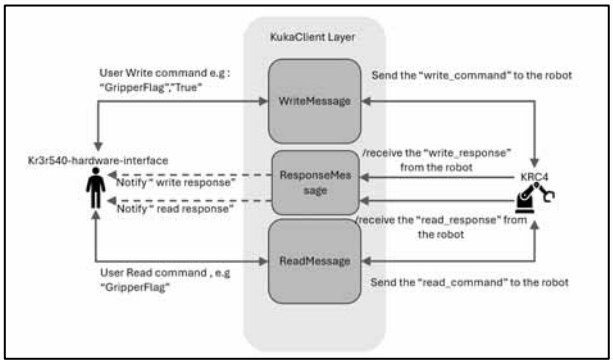


Figure 2: Kr3r540-Hardware-interface connection to Robot via KVP

Upon the issuance of a write command (e.g., setting GripperFlag to True), it undergoes serialization into a WriteMessage and is directed towards the robot. The robot, through KukaVarProxy, processes the command and responds with a ResponseMessage, which is deserialized and returned asynchronously to the hardware interface as a write response notification.

A similar process occurs for a read command, such as requesting the current value of GripperFlag, which is sent as a ReadMessage. Subsequent to the robot's response, the resulting ResponseMessage undergoes deserialization and is delivered to the hardware interface as a read response notification.

2.2 Integration with ROS2

ROS2 has been integrated into the system through the establishment of nodes (publisher/subscriber, server/client) and the management of data flow between them (see Figures 3 & 4). The robot description commences with the definition of the robot's physical characteristics and variables. In the context of a DT, the creation of two distinct robots under different namespaces results in a shared physical joint description yet a distinction in controller interface (see Figure 5).

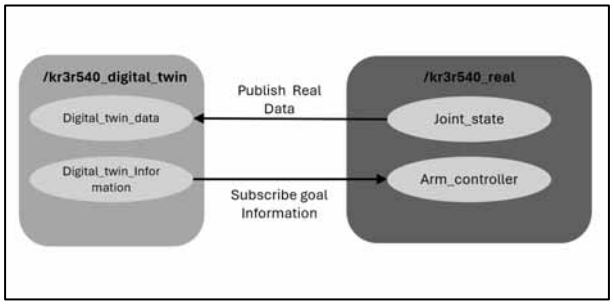


Figure 3: ROS2 nodes - Digital Twin real

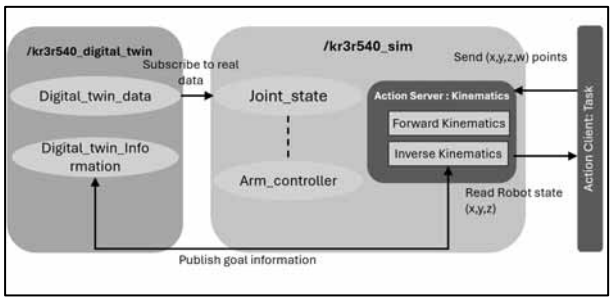


Figure 4: ROS2 nodes - Digital Twin simulation

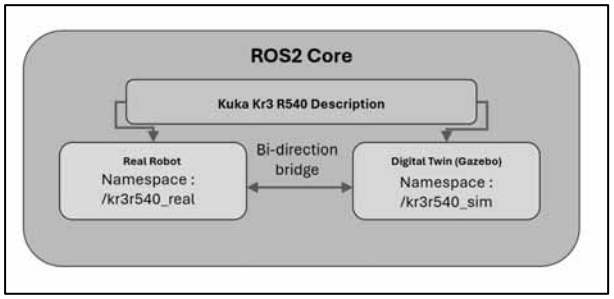


Figure 5: ROS2 system namespaces

It has been determined that each robot is equipped with its own controller and utilizes a distinct hardware interface. The primary objective is to establish a conduit between the two robots to facilitate the exchange of data. To this end, a novel ROS2 package has been developed to serve as a liaison between the two entities.

2.3 Path Planning Sequence

In the context of ROS 2, robotic manipulators are often controlled using third-party libraries such as MoveIt2, which provide built-in motion planning and trajectory generation [17]. However, in this work, MoveIt2 was not used. Due to the presence of distinct ROS 2 namespaces and custom controllers within the system architecture, integrating MoveIt2 was found to be limiting and overly complex for the intended control structure. As a result, a custom path planning sequence was developed,

independent of MoveIt2's motion planning stack and naming conventions.

The sequence of actions implemented in this work is as follows:

1. Initialization of pose points in Cartesian space
2. Path sampling and inverse kinematics computation
3. Conversion of pose points to joint space
4. Interpolation of joint positions into a time-parameterized trajectory using the Joint Trajectory Controller (JTC)
5. Generation of a geometric path in joint space
6. Execution of the path using PTP commands on the physical robot

This approach enables the conversion of Cartesian points into joint commands through inverse kinematics using the kr3r540_kinematics_action_server, which internally uses the KDL (Kinematics and Dynamics Library). The resulting joint trajectories are executed by the Joint Trajectory Controller in the Gazebo simulation or translated into *MYAXIS* commands for the real robot, which executes them using its KRC4 PTP motion commands. While the simulation environment, based on Gazebo (Ignition Fortress) and ros2_control [18], provides precise and predictable timing for trajectory execution, the real robot's behavior may vary due to internal smoothing algorithms, acceleration limits, or external load conditions. Consequently, the physical execution of motion may differ slightly in accuracy compared to the simulated environment. To ensure consistency and reliability across both environments, position thresholds were explicitly implemented in both the inverse kinematics solver and the hardware interface layer. These thresholds restrict the robot's movements to within its safe operational range and prevent infeasible or unsafe joint configurations. By applying the same safety constraints to both the simulation and the real robot, consistent and safe execution of planned motions is achieved without relying on additional calibration procedures.

2.4 Integration with OPC UA

The OPC UA Server establishes a connection between the ROS 2 core and the external environment, thereby generating ROS 2 nodes through UA methods (see Figure 6). These methods enable external clients to initiate or terminate various system components, such as the DT, simulation, and robot control nodes. They also support joint state subscription, pose retrieval, and goal command execution. The server was implemented in Python using the asyncua library, which supports asynchronous communication and dynamic node management [19].

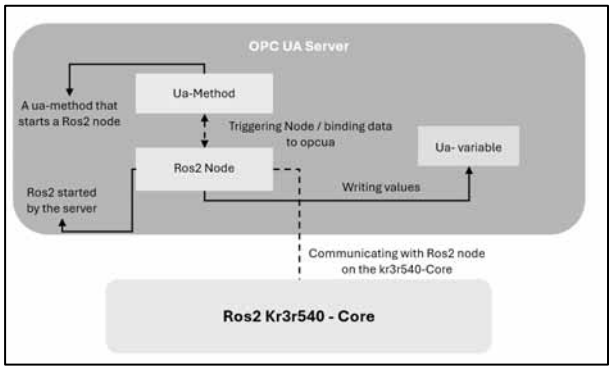


Figure 6: ROS2 OPC UA environment

An overview of the available OPC UA methods and their corresponding functionalities is presented in Table 1.

Method Name	Description
LaunchDigitalTwin	Launches the digital twin.
ShutdownDigitalTwin	Shuts down the digital twin.
LaunchRos2Simulation	Starts the ROS 2 simulation.
KillRos2Simulation	Stops the ROS 2 simulation.
LaunchRqt	Launches RQT.
SubscribeToJointState	Subscribes to ROS 2 joint states.
UnsubscribeToJointState	Unsubscribes from ROS 2 joint states.
GetRealCartesianPose	Retrieves the real robot's cartesian pose.
ShutdownRealCartesianPose	Stops tracking the real robot's cartesian pose.
GetSimCartesianPose	Retrieves the simulated robot's cartesian pose.
ShutdownSimCartesianPose	Stops tracking the simulated robot's cartesian pose.
SendGoal	Sends a movement goal to the robot.

Table 1: OPC UA - UA methods

2.5 Dashboard

The Dashboard is a desktop GUI application developed in C#. It functions as a full-featured OPC UA Client, enabling users to interact with both the simulated and real robot through OPC UA server methods. The client-side OPC UA implementation is based on the OPC UA .NET Standard Library, developed by the OPC Foundation

(Opc.Ua, Opc.Ua.Client), which provides support for secure and asynchronous UA communication over the network [20].

The application is structured into six main views, each responsible for a specific task in the user workflow (see Figure 7). The *Login* View allows users to authenticate, while new users can register through the *Register View*. Upon successful login, the user is taken to the *Home View*, which serves as the central interface for managing system startup and navigating to subsequent functionalities.

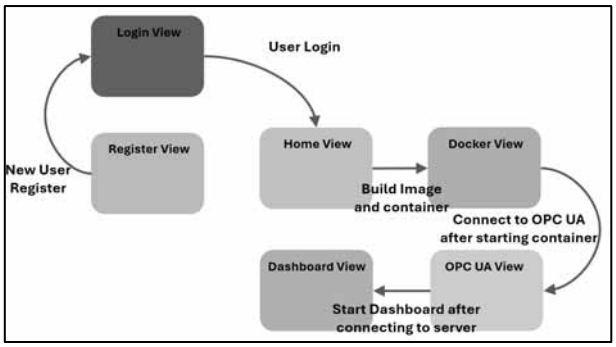


Figure 7: Dashboard overview

From the *Home View*, users can access the *Docker View*, where they can build Docker images and run containers for the ROS 2 and OPC UA server environment. After successfully launching the containerized backend, the *OPC UA View* allows the user to establish a connection with the running OPC UA server. Once connected, control is passed to the *Dashboard View*, which serves as the main interface for robot interaction. In this view users can launch or shut down the Digital Twin, send robot motion goals, retrieve cartesian poses from either the real or simulated robot, control the gripper, and monitor the states of the robot systems. These interactions are made possible through OPC UA method calls such as *SendGoal*, *LaunchDigitalTwin*, *GetRealCartesianPose*, etc. (see Table 1). To support persistence and user-specific configuration, the Dashboard uses MongoDB to store login credentials and robot point data. This enables session continuity and personalized robot control across restarts.

3 Application and Evaluation

3.1 Use case: Visual Commissioning

The system has been integrated into a visual commissioning use case (see Figure 8). This use case is employed in a showcase at our institute, wherein students create pathways, execute them in the simulation, launch the digital twin, and verify the congruence of the positions. A feedback collection process was initiated among students to identify potential vulnerabilities or software defects.

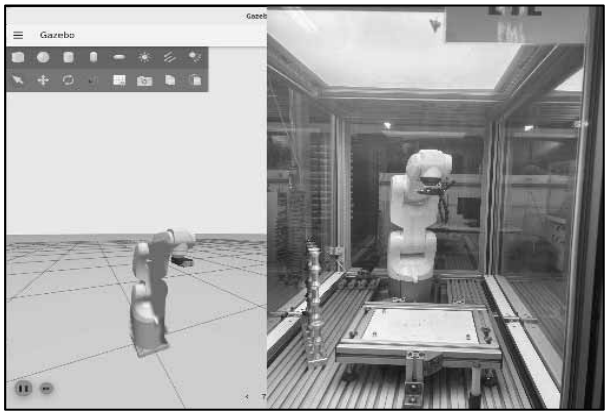


Figure 8: Digital twin use case

3.2 Evaluation

The evaluation process entails the collection and analysis of student feedback. Generally, the system functions adequately; however, its User Interface (UI) exhibits room for enhancement. Specifically, the integration of a task overview that displays all stored tasks is necessary. Additionally, the system's path planning sequence necessitates refinement, such as the incorporation of interpolation in Cartesian space to generate a Cartesian path, thereby eliminating the need for manual input. This process could be automated, enhancing the system's efficiency and user-friendliness. Additionally, the scope of the simulation could be expanded to encompass the robot cabin and the objects within it. Subsequently, a collision detection algorithm could be integrated into the system.

4 Conclusion

This work contributes to the expanding body of research on digital twins by demonstrating practical applications in robot simulation, control, and monitoring. It serves as a case study for integrating real-time synchronization between physical and digital systems. Within the context of research or academic settings, this work aligns with efforts to create flexible and reproducible robotics experiments, complementing other projects in robotics education by providing a platform for hands-on learning and research. The use of protocols like OPC UA and frameworks like ROS2 aligns this work with broader efforts in

robotics and industrial automation, ensuring compatibility with other systems and applications in the field.

The system's modular architecture and reliance on open frameworks (e.g., ROS2) facilitate its integration with advancements in the Internet of Things (IoT) and smart manufacturing. Consequently, it can function as a testbed for implementing AI-driven optimization or predictive maintenance.

References

[1] Juarez MG, Botti VJ, Giret AS. Digital twins: Review and challenges. In: Computing and Information Science in Engineering editors. *Journal of Computing and Information Science in Engineering*; 2021 Mar. New York: ASME. doi: 10.1115/1.4050244.

[2] OSRF**.** Robot Operating System (ROS) Documentation: Jazzy. In: OSRF editors. *Robot Operating System (ROS) Documentation*; 2025 Jan; San Francisco: OSRF. Available: https://docs.ros.org/en/jazzy/index.html.

[3] Rantanen A. *Robot Operating System: overview and case study* [Bachelor's thesis]. Turku, Finland: University of Turku; 2024.

[4] Saavedra Sueldo C, Perez Colo I, De Paula M, Villar SA, Acosta GG. ROS-based architecture for fast digital twin development of smart manufacturing robotized systems. *Annals of Operations Research*; 2022 Feb. New York: Springer. p. 1–14. doi: 10.1007/s10479-022-04759-4.

[5] Open Source Robotics Foundation, *RViz User Guide — ROS 2 Documentation (Humble)*. [Online]; accessed 2025 Mar. Available: https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz/RViz-User-Guide/RViz-User-Guide.html.

[6] Nguyen Q-D, Dhouib S, Huang Y, Bellot P. An approach to bridge ROS 1 and ROS 2 devices into an OPC UA-based testbed for industry 4.0. In *Proceedings of 1st Industrial Electronics Society Annual On-Line Conference (ONCON)*; 2022 Jan; New York: IEEE. p. 1–6. doi: 10.1109/ONCON56984.2022.10126783.

[7] OPC Foundation, OPC UA Online Reference Documentation. https://reference.opcfoundation.org/. [Online]; accessed 2025 Mar. Available: https://reference.opcfoundation.org/

[8] Arbo MH, Eriksen I, Sanfilippo F, Grødahl JT. Comparison of KVP and RSI for controlling KUKA robots over ROS. In: Findeisen R, Hirche S, Janschek K, Mönnigmann M., *21st IFAC World Congress*; 2020 Jul; Berlin. Elsevier. vol. 53, no. 2, p. 9841–9846. doi : 10.1016/j.ifacol.2020.12.2688.

[9] Koranne S, Koranne S. Boost C++ libraries. In: Open Source Tools editors. *Handbook of Open Source Tools*; 2011 Apr; Berlin. Springer. p. 127–143.doi: 10.1007/978-1-4419-7719-9.

[10] Open Source Robotics Foundation, *URDF - Unified Robot Description Format*. https://wiki.ros.org/urdf. [Online]; accessed 2025 Mar. Available: https://wiki.ros.org/urdf.

[11] Open Source Robotics Foundation, *Creating Custom ROS 2 Interfaces*. https://docs.ros.org/en/crystal/Tutorials/Custom-ROS2-Interfaces.html. [Online]; accessed 2025 Mar. Available: https://docs.ros.org/en/crystal/Tutorials/Custom-ROS2-Interfaces.html.

[12] O. Project, *KDL - Kinematics and Dynamics Library*. https://www.orocos.org/kdl. [Online]; accessed 2025 Mar. Available: https://www.orocos.org/kdl.

[13] ROS 2 Control Working Group, *Joint Trajectory Controller - User Documentation*. [Online]; accessed 2025 Mar. Available: https://control.ros.org/rolling/doc/ros2_controllers/joint_trajectory_controller/doc/userdoc.html.

[14] Open Source Robotics Foundation, *Gazebo Ignition Fortress - Getting Started*. [Online]; accessed 2025 Mar. Available: https://gazebosim.org/docs/fortress/getstarted/

[15] AvaloniaUI Community, *Avalonia UI Framework*. https://avaloniaui.net/. [Online]; accessed 2025 Mar. Available: https://avaloniaui.net/.

[16] Github repository of I. A. Ahmed, *MPA_KR3_Digital_Twin*. [Online]; accessed 2025 Mar. Available: https://github.com/aialmohamed/MPA_KR3_Digital_Twin.

[17] MoveIt Maintainers, *MoveIt Documentation (Main Branch)*. PickNik Robotics. [Online]; accessed 2025 Mar. Available: https://moveit.picknik.ai/main/index.html.

[18] ROS 2 Control Working Group, *ros2_control Documentation Index*. [Online]; accessed 2025 Mar. Available: https://control.ros.org/rolling/index.html.

[19] FreeOpcUa Project, *Minimal OPC UA Server — asyncua Documentation*. [Online]; accessed 2025 Mar. Available: https://opcua-asyncio.readthedocs.io/en/latest/usage/getstarted/minimal-server.html.

[20] OPC Foundation, *OPC UA .NET Standard Library Documentation*. https://opcfoundation.github.io/UA-.NETStandard/. [Online]; accessed 2025 Mar. Available: https://opcfoundation.github.io/UA-.NETStandard/.

SNE SIMULATION NOTES EUROPE

Simulation Notes Europe (**SNE**) provides an international, high-quality forum for presentation of new ideas and approaches in simulation - from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation - in context of the simulation process.

SNE seeks to serve scientists, researchers, developers and users of the simulation process across a variety of theoretical and applied fields in pursuit of novel ideas in simulation and to enable the exchange of experience and knowledge through descriptions of specific applications. **SNE** puts special emphasis on the overall view in simulation, and on comparative investigations, as benchmarks and comparisons in methodology and application. Additionally, **SNE** welcomes also contributions in education in / for / with simulation.

SNE is the official membership journal of **EUROSIM**, the Federation of European simulation societies and simulation groups, and the scientific membership journal of **ASIM**, the German simulation society. **SNE** is open for post-conference publications and for special issues organized by **EUROSIM** societies, e.g. **ASIM** thematic special issues or **ASIM** post-conference special issues.

SNE is primarily an electronic journal and follows an open access strategy, with free download in basic layout. Members of EUROSIM societies, as **ASIM**, **SIMS**, e.g. are entitled to download **SNE** in an elaborate and extended layout. Print **SNE** is available for specific groups of **EUROSIM** societies.

www.sne-journal.org

Schlüsselrolle Simulation: Wandel gestalten.
Herausforderungen meistern.

ASIM Fachtagung

24. bis 26. Sept.

2025

„Haus der Kirche“

Dresden

21. ASIM Fachtagung „Simulation in Produktion und Logistik“

Als größte europäische Tagung zum Thema Simulation im Bereich Produktion und Logistik gibt die ASIM Fachtagung alle zwei Jahre einen Überblick der zukunftsweisenden Trends, aktuellen Entwicklungen und erfolgreichen Projekte. Präsentiert und diskutiert werden wissenschaftliche Arbeiten sowie interessante Anwendungen aus der Industrie.

Themen der Tagung

Supply Chain Simulation & Logistik

- Produktionsnetzwerke & -logistik
- Intralogistik; Lieferketten
- Transport & Verkehr
- Innovative Materialflusstechnik
- Produktionsplanung & -steuerung
- Systemresilienz; (Cyber-)Sicherheit

Simulationsanwendungen in Industrie und Dienstleistung

- Automobilindustrie
- Maschinen- & Anlagenbau
- Halbleiterindustrie
- Simulation as a Service
- Energieeffizienz; Nachhaltigkeit

Methoden, Werkzeuge & Simulationstechnik

- Virtuelle Inbetriebnahme; Digitaler Zwilling; Digitaler Schatten
- Simulationsbasierte Optimierung
- Verifikation & Validierung
- Data Science; Visual Analytics; Virtual Reality; Augmented Reality
- Künstliche Intelligenz; Maschinelles Lernen
- Ereignisdiskrete Simulation; Agentenbasierte Simulation
- Interoperabilität; verteilte Simulation; Cloud-basierte Simulation

Teilnahme

Ausstellung

Die begleitende Ausstellung zeigt den aktuellen Stand von Simulationswerkzeugen, verwandter Software und Dienstleistungen im Simulationsumfeld. Falls Sie sich als Aussteller auf der Fachtagung präsentieren möchten, kontaktieren Sie uns bitte.

Teilnahmegebühr (pro Person, inkl. Mehrwertsteuer)

Im Preis enthalten sind die Teilnahme an Vorträgen, der Ausstellung, Mittagessen, Erfrischungsgetränke mit Pausensnack, ein Get-Together sowie eine Abendveranstaltung. Ein gedruckter Tagungsband kann gegen einen Aufpreis von 50,00 € optional erworben werden.

Grundpreis		1190,00 €
Frühbucherrabatt (bis 01.07.2025)	abzgl.	50,00 €
Mitglieder von ASIM, GI, EUROSIM	abzgl.	140,00 €
Einreichung Beitrag (1 x pro Beitrag und Autor)	abzgl.	170,00 €
ab dem 2. Teilnehmer einer Institution	abzgl.	140,00 €

Hochschulangehörige erhalten 40 % Nachlass auf den Endpreis.

Tagungsleitung

Prof. Dr.-Ing. habil.
Thorsten Schmidt

Professur für
Technische Logistik
Fak. Maschinenwesen
TU Dresden

Tagungsort

„Haus der Kirche“
Hauptstraße 23
01097 Dresden

Anmeldung

Alle Informationen zur Anmeldung finden Sie auf unserer Website:

[https://asim-gi.org/spl2025]

GI | ASIM | ASIM Fachtagung | Institut für Technische Logistik und Arbeitssysteme | TECHNISCHE UNIVERSITÄT DRESDEN