Multi-Fidelity Modeling of Dynamic Systems for Operation-Parallel Simulation

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der KIT-Fakultät für

Elektrotechnik und Informationstechnik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Christoph Bergs, M.Sc.

geb. in Würselen

Tag der mündl. Prüfung:	29.06.2020
Hauptreferent:	Prof. Dr.-Ing. Michael Heizmann, KIT
Korreferent:	Prof. Dr. rer. nat. Oliver Niggemann, HSU

Acknowledgment

The present work was accomplished while being a research assistant at the Institute of Industrial Information Technologies (IIIT) in Karlsruhe as well as being an industrial PhD candidate at Siemens AG in Munich. Over the past years, this work has benefited from several people and my gratitude is especially to those mentioned below.

At first, I want to thank Prof. Dr.-Ing. Michael Heizmann for all his input and every discussion which has helped this work to become more valuable. Thanks for letting this work grow as a combination of my personal ideas and your expertise. Additionally, I am really honored that Prof. Dr. rer. nat. Oliver Niggemann agreed on being the second examiner of this thesis.

Within Siemens, I especially want to thank Dr. Harald Held and Dr. Dirk Hartmann for all the support and valuable input they provided to this work. Furthermore, I want to thank Dr. Thomas Baudisch, Dr. Christoph Heinrich and Theodoros Papadopoulos for giving me the chance to drive this topic further within Siemens.

I very much appreciated the support of all my fellow PhD students for the great working atmosphere, content challenging and sport activities. A special thanks goes to Christoph Ernst Ludwig, Mohamed Khalil, Jakob Zietsch, Stefan Gavranovic, Beate Dutschk, David Uhlig and Norbert Mitschke.

During my time at Siemens, I got the chance to personally grow by co-leading the global Siemens PhD Network. In this context, I especially want to thank Dr. Fabian Rhein, Dr. Jan-Carl Grager, Martin Küppers, Iris Figalist and Philipp Krämer for the great time we had together.

My deepest respect, gratitude and love is due to my wife Susanne for her mental support and her acceptance during the intense time of this work. Furthermore, I want to express my gratitude and love to my parents and grandparents who mentally and financially supported me throughout this time and my whole life. Lastly, I want to thank my deceased uncle Heinz who was the earliest supporter of my academic career and always believed in me.

Herzogenrath, August 2020 Christoph Bergs

Zusammenfassung

Unter der Begrifflichkeit „Digitaler Zwilling“ werden aktuell im industriellen Kontext viele verschiedene Dinge verstanden. Im Rahmen dieser Arbeit wird die Erstellung eines solchen aus unterschiedlichen Artefakten des gesamten Produktlebenszyklus mit dem Ziel der Verwendbarkeit im Rahmen der betriebsparallelen Simulation betrachtet. Insbesondere wird die Verwendung von digitalen Zwillingen im Anwendungsbereich der prädiktiven Wartung und Zustandsüberwachung von industriellen Systemen untersucht.

Der in dieser Arbeit erforschte Ansatz der Modellierung dynamischer Systeme mit dem Verwendungszweck der betriebsparallelen Simulation besteht darin, die Systeme durch Kombination von theoretischen und datengetriebenen Modellierungsansätzen im Rahmen des Konzeptes der Multi-fidelity Modellierung zu beschreiben. Dabei liegt der methodische Hauptfokus darauf, sogenannte High-fidelity Modelle, Low-fidelity Modelle und Beobachtungen des realen Systems durch Methoden der Informationsfusion optimal zu kombinieren. Dazu wird eine Methodik zur Bestimmung sowie Parametrisierung der optimalen Fusionsmethode aus einem Satz möglicher Methoden hergeleitet, welche sowohl die Lösung eines Optimierungsproblems sowie ein Formalisierungskonzept für Fusionsmethoden beinhaltet. In dieser Arbeit werden zudem die Fusionsmethoden der multivariaten Regression (quadratisches Modell mit Interaktionen), neuronaler Netze (Multi-layer Perzeptron), Gaußprozesse (Co-Kriging) sowie die im Rahmen dieser Arbeit entwickelte hybride Fusionsmethodik moSAIc näher betrachtet.

Zur theoretischen Validierung der entwickelten Methodik werden drei unterschiedliche Typen von Verwendungszwecken eingeführt, welche alle als mögliche Anwendungsfälle im Kontext der betriebsparallelen Simulation vorstellbar sind. Für diese Verwendungszwecke werden dazu Benchmark Systeme eingeführt, anhand derer die Methodik für den speziellen Fall evaluiert wird. Der erste Verwendungszweck ist die Dimensionalitätserhöhung bei der Zustandsüberwachung von industriellen Systemen. Dabei werden durch Anwendung der Multi-fidelity Modellierung sogenannte Soft-Sensoren erstellt, welche betriebsparallel eingesetzt werden können. Als zweiter Verwendungszweck wird die Modellierung zur Verbesserung der Modellgenauigkeit dargestellt, welche darauf abzielt, detailliertes Wissen aus der Entwicklungsphase des dynamischen Systems in dessen Betriebsphase zu verwenden. Der dritte betrachtete Verwendungszweck ist die Übertragung von Wissen modellierter industrieller Systeme auf nicht-modellierte ähnliche Systeme, von denen lediglich Beobachtungen zur Verfügung stehen.

Nach der theoretischen folgt die praktische Evaluierung der entwickelten Methodik anhand von drei industriellen Fallstudien, welche jeweils einem der drei Verwendungszwecke zugeordnet sind. Für den Verwendungszweck der Dimensionalitätserhöhung wird die thermische Zustandsüberwachung von Asynchronmaschinen betrachtet. Dazu wird ein betriebsparallel einsetzbares Simulationsmodell aus einem numerischen FEM-Modell, einem analytischen Drei-Körper-Modell sowie aus Beobachtungen von Temperatursensoren hergeleitet, welches Temperaturen an Gehäuse, Stator, Rotor sowie innerhalb der Wicklungen schätzt. Der Verwendungszweck der Verbesserung der Modellgenauigkeit wird anhand der Druckabfallsteuerung eines Klappenventils analysiert. Dabei wird ein Simulationsmodell anhand der Methodik entwickelt, welches ein numerisches CFD-Modell mit einem analytischen physikalisch-basierten Modell kombiniert. Zuletzt wird der Verwendungszweck der Übertragbarkeit anhand einer Fallstudie untersucht, welche das Ziel hat, die strukturmechanische Lebensdauer von Elektromotoren dynamisch basierend auf dessen aktuellem Zustand zu prognostizieren. Dabei zielt die Verwendung der Methodik darauf ab, die Restlebensdauer vieler unterschiedlicher Derivate von Asynchronmaschinen basierend auf den Prognosen weniger Modelle zu schätzen, da die Modellierung jedes einzelnen Derivates sehr kostenintensiv wäre.

Zusammenfassend werden der in dieser Arbeit entwickelte Modellierungsansatz zur Erstellung von betriebsparallelen Simulationsmodellen, die entwickelte Methodik zur Ermittlung der optimalen Fusionsmethode sowie die Anwendung der vier beschrieben Fusionsmethoden auf Benchmark Systeme sowie industrielle Fallstudien beschrieben.

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Related Work 5
1.3 Contributions 7
1.4 Thesis Structure 9

2 Modeling of Dynamic Systems **11**
2.1 System Definitions 11
2.2 Modeling Approaches 15
2.2.1 Theoretical Models 15
2.2.2 Data-driven Models 17
2.2.3 Knowledge-based Models 19
2.2.4 Hybrid Models 20
2.3 Multi-fidelity Modeling 24
2.3.1 Model Fidelity 24
2.3.2 Development Work Flow 27

3 Multi-fidelity Modeling for Operation-Parallel Simulation **31**
3.1 Model Development 31
3.2 Multi-Model Data Fusion 37
3.2.1 Fusion operator training 39
3.2.2 Explored Methods as Fusion Operator 42

4 Purpose-driven Model Design **59**
4.1 Model Design for Dimensionality Increase 59
4.1.1 Work Flow 60
4.1.2 Benchmark System 62
4.2 Model Design for Accuracy Enhancement 73
4.2.1 Work Flow 73
4.2.2 Benchmark System 76

4.3 Model Design for Transferability 83
4.3.1 Work Flow 84
4.3.2 Benchmark System 86

5 Industrial Case Studies on Multi-fidelity Modeling 93
5.1 Temperature Estimation for Electric Motors 93
5.1.1 Case Study Description 93
5.1.2 Case Study Results 98
5.2 Pressure Drop Control of a Flapper Valve 101
5.2.1 Case Study Description 101
5.2.2 Case Study Results 106
5.3 Motor Fleet Condition Monitoring 109
5.3.1 Case Study Description 109
5.3.2 Case Study Results 117

6 Conclusion 123
6.1 Summary 123
6.2 Outlook 125

Bibliography 129
List of Publications 134
List of Supervised Theses 135

Scalars are given by lower or upper case letters ($x, X, \alpha, \in \mathbb{R}$).
Vectors are given by bold lower case letters ($\boldsymbol{x}, \boldsymbol{\alpha} \in \mathbb{R}^n$).
Matrices are given by bold upper case letter ($\boldsymbol{X} \in \mathbb{R}^{\mathrm{n} \times \mathrm{m}}$).
Mapping Operators are given by calligraphic upper case letters ($\mathcal{L}$).

Notations

a	Constant coefficient I
A	System attribute according to [Rod12]
b	Constant coefficient II
c	Specific training sample from training data set
C	Descriptive system features
C_{th}	Thermal capacity
d_{b}	Damping of a PT2-System
d^{m}	Measurable disturbance
D	Valve lift
E	Error function of backpropagation algorithm
E_{M}	Elastic modulus
f	General mathematical function
g	Weighing function
g^c	Co-Kriging function
h	Multi-layer perceptron function
H	High-fidelity model superscription index
i	Index variable
I	Electrical current
j	Index variable
k	Discrete time step
k^G	Co-variance function of a Gaussian process
K	Co-Kriging co-variance matrix
l	Length scale of radial basis function kernel
l_{h}	Housing length of electric motor
l_{r}	Rotor length of electric motor
L	Low-fidelity model superscription index
m	Method under consideration for fusion operator training

M_h	Housing material of electric motor
N	Number of samples
N_p	Number of allowed parameters for the fusion problem
n	Motor speed
N_L	Number of load cycles
N_Q	Number of variables of a multivariate regression problem
o	Output of a multi-layer perceptron
p	Pressure drop
P_el	Electrical power
P_V	Thermal losses in an asynchronous motor
Q_H	Heat quantity
Q_V	Volume flow
R	Electrical resistance
S_eq	Equivalent stress
S_f	Fatigue strength
t	Time variable
T	Motor torque
T_b	Time constant of a PT2-System
u	System input
v	Input of a multi-layer perceptron
x	System state
y	System output
z	Fatigue coefficient
α	Parametric model vector
$\alpha_{20^\circ\mathrm{C}}$	Temperature coefficient
β	Parametric model coefficent
$\mathit{\Gamma}$	General fusion function set
δ	Parallel misalignment
Δ	Laplace operator
ϵ	Residuals
ζ	Specific layer of a multi-layer perceptron
η	Learning rate
η_M	Motor efficiency
ϑ	Temperature
θ_G	Hyper-parameters of a Gaussian process

θ_h	Parameter vector of a multi-layer perceptron
κ	Target output of backpropagation algorithm
F_f	Selected fusion function set
Λ_th	Thermal conductivity
ν	Poisson ratio
Ξ	Input domain of a Gaussian process
μ	Mean function of a Gaussian process
ξ	Sum of all layers of a multi-layer perceptron
ρ	Autoregressive Co-Kriging coefficient
ϱ	Parameters of method under consideration for fusion operator training
σ	Amplitude of radial basis function kernel
τ	Time delta
Υ	Mapping function set according to [Rod12]
φ	Angular misalignment
Φ	Training set matrix
χ	Actual output of backpropagation algorithm
ψ	Multi-index of multivariate regression model
Ψ	Multi-index set
ω	Non-parametric model coefficient
Ω	General function set
$\mathcal{L}$	Low-fidelity model operator
$\mathcal{H}$	High-fidelity model operator
$\mathcal{H}^\mathrm{r}$	Reduced-order high-fidelity model operator
$\mathcal{O}$	General model operator
$\mathcal{S}$	Mapping operator according to [FB09]
$\mathcal{Z}$	State machine
$\circledcirc$	Fusion Operator

Abbreviations

ANN	Artificial neural network
ARCK	Autoregressive Co-Kriging
ARMA	Autoregressive model with moving average
ARX	Autoregressive model with exogenous input
CAD	Computer-aided design
CAM	Computer-aided manufacturing
CFD	Computational fluid dynamics
CBM	Condition-based maintenance
DCS	Distributed control system
DMGP	Deep multi-fidelity Gaussian processes
DoE	Design of experiments
ED	Euclidean distance
FEM	Finite-element method
FPGA	Field-programmable gate array
FVM	Finite-volume method
GP	Gaussian process
GPR	Gaussian process regression
HFM	High-fidelity model
IIoT	Industrial internet-of-things
IPC	Industrial personal computer
LFM	Low-fidelity model
LSE	Least-squares estimation

LTI	Linear-time-invariant system
MAPD	Mean absolute percentage deviation
MFM	Multi-fidelity model
MFHM	Multi-fidelity hierarchical model
MFMopS	Multi-fidelity for operation-parallel simulation
MFSM	Multi-fidelity surrogate model
MLP	Multi-layer perceptron
MLE	Maximum-likelihood estimation
MLP	Multi-layer perceptron
MOR	Model order reduction
MPR	Multiple polynomial regression
MRE	Mean relative error
PCA	Principle component analysis
PLC	Programmable logic controller
PLM	Product life cycle management
PLS	Partial least squares
QMI	Quadratic model with interactions
RMSE	Root-mean-square error
RUL	Remaining useful life
SVR	Support vector regression
SCK	Simple Co-Kriging
SNC	Stress-cycle-curve
VHDL	Very high speed integrated circuit hardware description language
WAAF	Weighed arithmetic average filter

1 Introduction

This chapter introduces the subject of operation-parallel simulation using digital twins and its usage in industrial service solutions, which are the main motivating aspects for this work. The chapter closes by outlining the main contributions as well as explaining the organization of this thesis.

1.1 Motivation

The digitization of industrial systems is rapidly increasing and impacts every phase of the product life cycle of an industrial system. Starting in the design phase of a product, the digitization helps design engineers to create a virtual representation of their future product. Mainly all functionalities of the future product can be explored in a virtual environment today, e.g. using a Computer-aided design (CAD) tool in combination with different simulation solutions. The information which is gained in the design phase mainly serves the purpose of guaranteeing the fulfillment of the end-product requirements without having to build expensive prototypes. Furthermore, design changes and their influences can be determined rather quickly in comparison to building prototypes and can be later leveraged for product optimization purposes. Therefore, the models which are developed in the design phase of a product are typically very detailed and high-dimensional and consequently require huge computational resources.

Having the product functionality guaranteed and the final design determined, the production process for the final product can be started. Nowadays, many digital solutions support production plant operators in creating the production process automatically, like Computer-aided manufacturing (CAM) tools. During the production process itself highly-instrumented machines allow to collect product individual information which provide insights about the quality of the manufactured product, e.g.

whether the specified tolerances were reached. To gain this product transparency, huge digital infrastructures and solutions like Manufacturing Execution Systems (MES) are utilized.

During the operation phase of the product, for example the Industrial internet-of-things (IIoT) allows operators to acquire asset-specific information. Hereby, many upcoming digital ecosystems are the driver to handle the large amounts of data which are generated during operation. Concepts of edge computing are becoming more and more important, especially to pre-process the data from sensors and just transmit the relevant features to the digital ecosystem. The large database of operational data opens the door for the usage of data-intensive methods to gain system insights, like machine learning algorithms. In the context of big data analytics, the usage of machine learning algorithms to create data-driven models from operational data is leveraged to create services like predictive maintenance. Models which are created in this product life cycle phase are typically low-dimensional, because they are developed to to fulfill a specific purpose based on the setup provided. Furthermore, the possible insight highly depends on the accuracy which is provided by the installed sensors.

Concluding this, Figure 1.1 shows the different Product life cycle management (PLM) phases and the different types of digital information which is available in the operation phase of the product. Up to now, these phases are mainly decoupled due to the fact that different stakeholders are responsible for the different life cycle phases, meaning that the design engineer is probably not developing models for the operation phase of the system. This thesis is motivated by the vision, that models from the design phase of a product can be automatically used during the operation phase of the product to transfer the knowledge which is already available from the deep analysis during the product design to its operation.

According to [Gar19], the concept of a Digital Twin has proven to be one of the most recent research topics in this area and will be introduced in the following. The term *Digital Twin* has been mentioned and defined in many publications over the last years. One common definition is given by [GS12] who described it as an "integrated multiphysics, multiscale, probabilistic simulation of an as-built system which uses best available physical models, sensor updates, and fleet history to mirror the life of its corresponding flying twin". It was used in the context of vehicles. A more

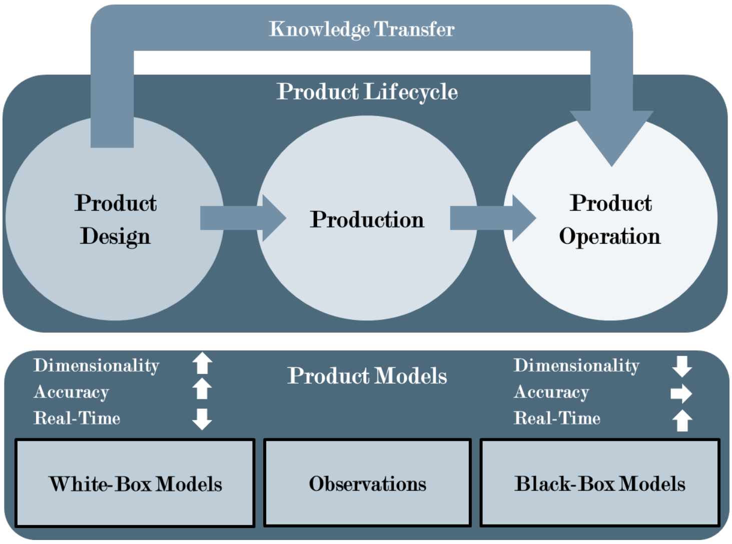


Figure 1.1 Product life cycle phases and available product models.

detailed definition is given by Boschert, Rosen et al. [BRH18] who include the IIoT perspective and separate the digital twin into a digital product twin, a digital production twin and a digital performance twin. The separation between product twin, production twin and performance twin is aligned with the corresponding product life cycle phases (see Figure 1.1), meaning a digital product twin corresponds to the design phase, the digital production twin to the production phase and the digital performance twin to the operation phase. As example, a digital performance twin of a point switching machine for trains allows the operator to monitor the equipment using an executable online-capable model during operation. Having such a model available also enables operators to forecast the wear and damages of the product based on given and future-expected operating conditions to schedule maintenance more efficiently.

The main motivation of this thesis is to transfer models from the design phase of the product efficiently to the operation phase of the product. In digital twin categories spoken: transfer already existing knowledge provided by the digital product twin efficiently to the digital performance twin of the product. The economic aim is to reduce the development effort for digital twins which can be used for predictive maintenance activities. Figure 1.2 gives an overview of the different maintenance types according to [Eur18].

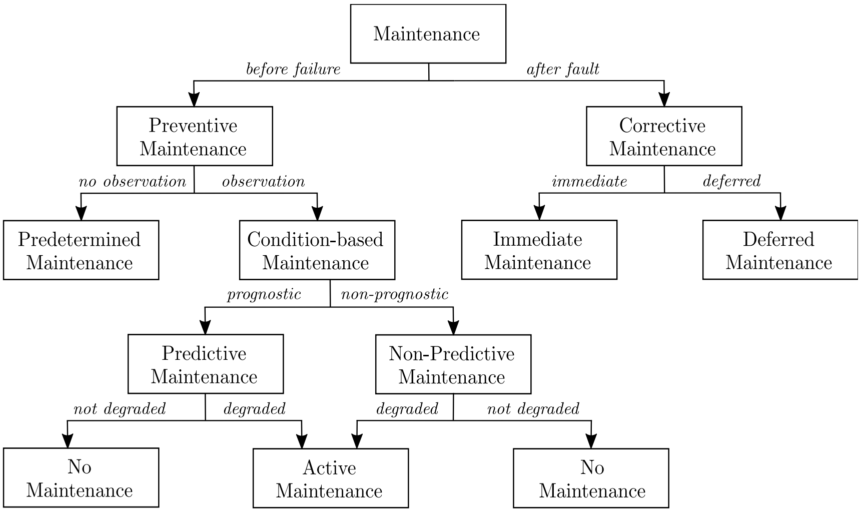


Figure 1.2 Types of maintenance according to [Eur18].

Nowadays, models which are used for predictive maintenance are mainly data-driven and base on the sensor data which is acquired during operation [NW18]. This is due to the fact, that it is relatively cheap, in particular in terms of manual effort, to develop data-driven models for fault detection or degradation prediction compared to development of models which base on natural science laws, for example equation-based laws. The concept of the digital performance twin is to provide a possibility to add knowledge to the operation phase and combine the data-driven models, the models transferred and adapted from the design phase and the observations made during operation to a joint digital performance twin.

The concepts of multi-fidelity modeling and hybrid modeling are established in their specific domains, e.g. for product design optimization [Böh15] and for process control of chemical plants [GS18]. In this thesis, these concepts are going to be evaluated and utilized as enabler for digital twin transfer between different PLM phases. The goal is to create a work flow including methodologies to provide digital performance twins for operation-parallel simulation improving accuracy and speed of non-predictive maintenance services (like condition monitoring) as well as predictive maintenance services (like state and output prediction).

1.2 Related Work

In this section, a brief review of the most relevant work for this thesis in the fields of digital twins for different PLM phases, of multi-fidelity and hybrid modeling concepts for dynamic systems as well as of predictive maintenance for industrial systems is given.

Related to approaches for the transfer of digital twins between PLM phases, different work has been done in the past. [Rie+19] describes the concept of using a digital shadow including the digital twin over different PLM phases by making use of an information model which includes the specific properties for each phase. A more detailed approach is introduced by [Qi+18], who addresses the necessity of creating digital services out of digital twins in form of their provision as simulation services using also an information model to make use of a digital twin in different PLM phases. The partition of the usage of a digital twin not just for interoperability between PLM phases, but also for scalability, expansibility and fidelity [Sch+17] or even simulation domains, simulation methods, applications, disciplines, users, or infrastructures [Sch+18] is also addressed in recent research. [Sch+17] introduces the terminology of *twinning*, which means the adaption of one digital twin for the purpose it is going to be used for. Actual realizations of such theoretical approaches have been done in the field of monitoring and diagnosis of manufacturing processes on the one hand and reconfigurability of manufacturing processes on the other hand [KST10; Kap11]. The work of [Kap11] focuses on time-discrete operation-parallel simulation on manufacturing system level for dynamic processes.

In the context of using different modeling approaches and data sources to enhance predictive maintenance services, [GMZ16] introduced a hybrid prognostics approach in the context of health management for mechanical systems. [GMZ16] states the advantages and disadvantages of hybrid prognostics approaches and evaluates them in different case studies mainly focusing on one specific mechanical system component. [GS18; Sto+14] evaluate a similar hybrid modeling approach for biochemical and petrochemical processes. It is underlined, that hybrid modeling approaches have a high benefit-to-cost ratio for the modeling of complex systems. Both [GMZ16] and [GS18] consider the combination of physics-based models and data-driven models as well as available observations, but do not mention the re-usage of already existing design models. Further case studies dealing with traffic flow queing [Hof13] or temperature estimation in a pipe [Kic17] came to similar results where hybrid models outperform their pure individual components. A further approach of hybrid modeling and prediction of dynamical systems is described by [HLF17] in the context of computational biology, who combines (similar to [GS18]) parametric and non-parametric models to a hybrid prediction model which also outperforms the individual models.

Extending the approach hybrid modeling leads to the consideration of multi-fidelity modeling. Related to this topic, a lot of research has been done in the area of design optimization and simulation acceleration. In the context of design optimization, multi-fidelity modeling has proven to deliver qualitatively equivalent results in a shorter time compared to a full-scale high-fidelity modeling approach. The case studies which have been considered vary from airfoil shape optimization [LW11] over extreme loads on wind turbines [Abd+15] to semiconductor manufacturing systems [Hua+15]. A systematical approach of using multi-fidelity modeling for design optimization was introduced by [Böh15]. Generally, the concept of multi-fidelity information fusion has been described mathematically by [PVK16]. Research regarding the usage of multi-fidelity modeling to join observations and model prediction has also been done, for example by [Bab+16] who modeled mixed convection using experimental data and numerical simulations.

1.3 Contributions

The previously introduced related work and the introduction show the research which has been conducted in the specific domains of digital twin development, hybrid modeling, multi-fidelity modeling, and predictive maintenance services. One of the main aims of this thesis is the integration of methodologies from different domains to develop a joint concept for operation-parallel simulation of industrial systems using multi-fidelity modeling with a focus on predictive maintenance. Another important proposal is the differentiation among the application of multi-fidelity modeling for operation-parallel simulation, meaning the focus on model design for accuracy enhancement, model design for dimension enhancement or model design for similarity estimation. For the identification of advantages and drawbacks of each model design approach, benchmark data sets are developed. In detail, the main contributions of this thesis are:

- Design of a model development work flow for digital performance twins combining models of different PLM phases. The novelty of this work flow is its application in the context of digital twin development using the concept of multi-fidelity modeling (see section 3.1).

- Development of a methodology how to determine the optimal fusion method for a data set provided by low-fidelity models, high-fidelity models and observations including its parameterization out of a predefined set of possible methods. An optimization problem as well as a formalization concept for fusion methods are introduced to find the most suitable fusion method based on a training and validation set (see section 3.2).

- Development of a method to transfer knowledge from systems with corresponding high-fidelity models to similar systems without corresponding high-fidelity model. The combination of observations and existing high-fidelity models to predict the remaining useful life of an electric motor without available high-fidelity model is presented to show the capability of multi-fidelity modeling (see section 3.2.2.3, 5.3).

- Exploitation of co-kriging approaches with the purpose of operation-parallel simulation to enhance model accuracy (see section 3.2.2.4, 5.2).
- Validation of multi-fidelity modeling concepts on industrial case studies for operation-parallel simulation. The functionality of the developed concept is validated using several different industrial applications (see section 5).

Publications

Several partial results and methodological concepts have been presented in different publications during the elaboration of this work. The concept of hybrid modeling for industrial applications and determination of optimal fusion methods is explained in [BHH18]. A first application showing the capabilities of multi-fidelity modeling by combining different types of temperature distribution simulations of electric motors is presented in [BH18]. Following this, a deeper investigation of the previous case study was done and the new insights were presented in [BH19b]. How to combine different high-fidelity simulations of electric motors to provide condition monitoring services for unknown but similar motors in explained in [Hil+19]. Furthermore, a deep dive of the method *moSAIc* which was presented in the previous publication was done and the improved methods were introduced in [BH19a]. The integration of these algorithms in a digital ecosystem as well as the interactive visualization of condition monitoring services were presented in [Kha+19]. Additionally, an investigation of multi-fidelity modeling in the domain of centrifugal pumps was done and the results, how to estimate the wear of a centrifugal pump during its operation, were presented in [Ber+19a].

Furthermore, several publications in the domain of energy flexible production systems have been done, such as [Ber+19b], [Pre+18] and [Tur+19].

Finally, one final thesis in the domain of multi-fidelity modeling [Ram18] and one final thesis in the domain of energy flexible production systems [Tur18] were supervised.

1.4 Thesis Structure

The thesis consists of six consecutive chapters and argues the topic of multi-fidelity modeling of dynamic systems for operation-parallel simulation. **Chapter 2** reviews existing modeling approaches for dynamic systems, including different system definitions and types of modeling approaches such as theoretical modeling, data-driven modeling, knowledge-based modeling and especially hybrid modeling. Furthermore, the concept of multi-fidelity modeling is introduced including the relevant approaches for this thesis. In **Chapter 3**, the theoretical approach of multi-model data fusion as well as fusion method formalization and optimal criteria for a trained fusion method are explained. Additionally, the modeling work flow to develop multi-fidelity models based on low-fidelity models, high-fidelity models and observations for operation-parallel simulation purposes is described. **Chapter 4** is split into the three different purposes for operation-parallel simulation: dimensionality increase, accuracy enhancement and transferability. The different modeling work flows and benchmark system concepts are defined as well as benchmark results are explained. In **Chapter 5**, industrial case studies for each purpose are introduced, comprising thermal and mechanical investigations on electric motors as well as on industrial flapper valves. Each case study uses the approach of multi-fidelity modeling for operation-parallel simulation to develop a digital performance twin. The case studies are described in detail and the results of applying multi-fidelity modeling are evaluated and contextually ranked. Finally, **Chapter 6** concludes the thesis, summarizes the advantages and drawbacks of multi-fidelity modeling for operation-parallel simulation and gives a proposal where future research should be conducted.

2 Modeling of Dynamic Systems

This chapter introduces different modeling approaches for dynamic systems, partitioned in theoretical, data-driven, knowledge-based, and hybrid modeling approaches as well as the system definition for this thesis. Furthermore, the concept and development work flow of multi-fidelity modeling are explained in detail.

2.1 System Definitions

An operation-parallel digital twin of a dynamic system in context of an application like condition monitoring or predictive maintenance requires a representation of the system which should be observed, the behavior of which should be predicted or whose performance should be evaluated. The base for providing system insights is its assumed representation. The definition of the representation differs depending on the domain which it is defined for. Therefore, the definitions of a system in general and a dynamic system in particular are discussed in the following, based on the work of [Kro16], [FB09] and [Rod12]. [Kro16] derives the definition for a dynamic system by using both the system and process definition based on the international dictionary for control technology [Int14]:

Definition 2.1 (System definition)
A system is a set of interrelated elements that are seen as a whole in a particular context and are regarded as separate from the environment.

Definition 2.2 (Process definition)
A process is the totality of interacting processes in a system through which matter, energy or information is transformed, transported or stored.

Considering this, [Kro16] defines a dynamic system in general in form of a block diagram characterizing the input values as $u(t)$, the output values as $y(t)$, the actual system state as $x(t)$ and the initial state as $x(0)$ in dependency on the time variable t. A main differentiation factor for systems is the fact if a system is static or dynamic. A static system exclusively relies on the input $u(t)$, a dynamic system relies on the initial system state $x(0)$, the actual system state $x(t)$ and the input $u(t)$. A similar

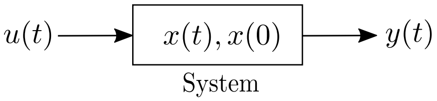


Figure 2.1 System definition according to [Kro16].

system representation is introduced by [FB09], which however differs in the details, because the system state is covered by a mapping operator $\mathcal{S}$ between the system input $u(t)$ and system output $y(t)$. Since typically not only time-continuous system representations are required, the mapping operator $\mathcal{S}$ is used for both, time-continuous systems with system input $u(t)$ and system output $y(t)$ as well as time-discrete systems with system input u_k and system output y_k, where k indicates the time step of the respective discretized signal.

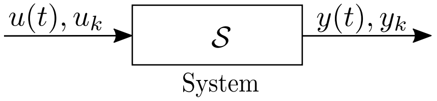


Figure 2.2 System definition according to [FB09].

Another considered system definition is used in the context of mechatronics: [Rod12] introduces, that each system shows certain characteristics, features and properties to its environment, which are called attributes. Every attribute A, which is neither a system input nor a system output is described as a state. Using these definitions, a function set Υ is derived, which describes the relationship between attributes.

All given definitions are similar in describing the system representation, because all include an input, an output and a function or operator which however realizes a mapping between the input and the output. The

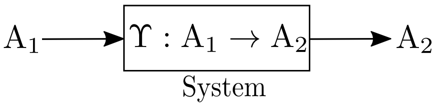


Figure 2.3 System definition according to [Rod12].

introduced definitions already cover two characteristics of a system, if it is static or dynamic and if it is time-continuous or time-discrete. In general, there are many characteristics of systems from a theoretical point of view. Some of them are listed in Table 2.1.

Table 2.1 System characteristics according to [FB09].

Definition Space	Time-continuous	Time-discrete
Value Range	continuous in value	discrete in value, discrete in amplitude
	limited (in value)	not limited
More Properties	linear	non-linear
	time-invariant	time-variant
	causal	non-causal
	stable	unstable
	dynamic	static
	deterministic	stochastic
	observable	non-observable
	differentiable	non-differentiable

For this work, the introduced methods are valid for linear or non-linear, time-invariant, causal or non-causal, stable or unstable, deterministic, observable and differentiable or non-differentiable systems. Considering the reduced characteristics, another system representation for a dynamic system is required which is introduced by [Kro16] and extends the general system definition (see Figure 2.1) by measurable disturbances $\boldsymbol{d}^{\mathrm{m}}$ and non-measurable disturbances $\boldsymbol{d}^{\mathrm{nm}}$. These disturbances are directly influencing the system state $\boldsymbol{x}(t)$ and can have a deterministic as well as a stochastic behavior (see Figure 2.4, 2.5).

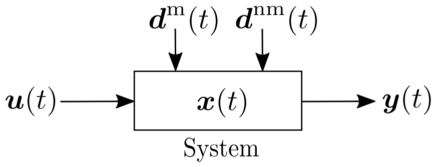

$\boldsymbol{d}^{\mathrm{m}}(t)$
$\boldsymbol{d}^{\mathrm{nm}}(t)$
$\boldsymbol{u}(t)$
$\boldsymbol{x}(t)$
$\boldsymbol{y}(t)$
System

Figure 2.4 Time-continuous dynamic system definition by [Kro16].

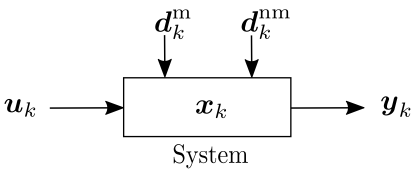

$\boldsymbol{d}_k^{\mathrm{m}}$
$\boldsymbol{d}_k^{\mathrm{nm}}$
$\boldsymbol{u}_k$
$\boldsymbol{x}_k$
$\boldsymbol{y}_k$
System

Figure 2.5 Time-discrete dynamic system definition by [Kro16].

Since the focus of this work is the modeling for operation-parallel simulation of dynamic systems, the system representation is simplified based on the definitions by [Kro16], [FB09] and [Rod12]. Therefore, the system state is modeled using a mapping operator $\mathcal{O}$ which maps the input vector $\boldsymbol{u}(t)$ or $\boldsymbol{u}_k$ to the output vector $\boldsymbol{y}(t)$ or $\boldsymbol{y}_k$. Furthermore, the combined modeling and measurement error $\boldsymbol{w}(t)$ or $\boldsymbol{w}_t$ for the mapping operator $\mathcal{O}$ which also includes the disturbances $\boldsymbol{d}^{\mathrm{m}}$ and $\boldsymbol{d}^{\mathrm{nm}}$ is added to the system representation. The operator $\mathcal{O}$ can be modeled using different modeling approaches (see section 2.2). Concluding this,

$$\boldsymbol{y}(t) = \mathcal{O}\{\boldsymbol{u}(t), \boldsymbol{w}(t)\} \tag{2.1}$$

and

$$\boldsymbol{y}_k = \mathcal{O}\{\boldsymbol{u}_k, \boldsymbol{w}_t\} \tag{2.2}$$

describe the relationship between the system input, the combined modeling and measurement error and the system output by applying the model operator $\mathcal{O}$. The respective system representations for this work are shown in Figure 2.6 for a time-continuous system and 2.7 for a time-discrete system.

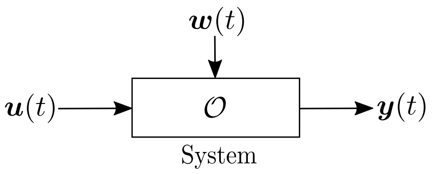


Figure 2.6 Time-continuous dynamic system definition.

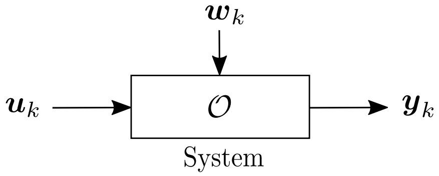


Figure 2.7 Time-discrete dynamic system definition.

2.2 Modeling Approaches

Generally, modeling approaches can be separated into three classes. The first model class are theoretical models (see section 2.2.1) that base on the laws of natural science. In contrast to this class, the class of data-driven models are based on observations and estimations. The third class of models uses the knowledge of an expert, therefore it is called knowledge-based modeling [HP16]. These different approaches will be introduced in detail in the following.

2.2.1 Theoretical Models

Theoretical models are based on natural laws and can have different mathematical representations, for example equation-based representations. They are mainly developed in the design phase of a system and are called white-box-models. Due to the use of domain-specific properties, an expert is necessary to develop them. In the context of this work, white-box-models describe the physical laws of a dynamic system by providing rules for the relationship between system input and system output as well as the system state. These physical laws are formulated in a mathematical model which can be solved analytically or numerically. Analytical models of dynamic systems depend on the system properties: an analytical model of a dynamic system in a stationary case can have a different mathematical description than an analytical model for a dynamic system in a transient case. Typically, a set of equations describing the relation between the system input and system output is used to describe a dynamic system in a stationary case. For example, the relationship between electrical current I and electrical power P_{el} of an electrical resistor R considering constant temperature ϑ conditions can be described by

$$P_{\mathrm{el}}(t) = I^2(t) \cdot R. \tag{2.3}$$

Taking the change of the resistance in dependency of the temperature into account, adds the consideration of the transient phase and changes the mathematical description from a simple equation (see Equation 2.3) to a linear differential equation (see Equation 2.6, based on [Ném18]). These equations also include the material's resistance $R_{20^\circ\mathrm{C}}$ and temperature coefficient $\alpha_{20^\circ\mathrm{C}}$ at ambient temperature, its thermal conductivity Λ_{th} as well as its thermal capacity C_{th}:

$$P_{\mathrm{el}}(t) = I^2(t) \cdot R(\vartheta), \tag{2.4}$$

$$R(\vartheta) = R_{20^\circ\mathrm{C}} \left(1 + \alpha_{20^\circ\mathrm{C}}(\vartheta(t) - \vartheta_{20^\circ\mathrm{C}}(t))\right), \tag{2.5}$$

$$\frac{d}{dt}\,\vartheta(t) = \frac{I^2(t)\; R_{20^\circ\mathrm{C}}\; \alpha_{20^\circ\mathrm{C}} - \Lambda_{\mathrm{th}}}{C_{\mathrm{th}}} \vartheta(t) + \frac{I^2(t)\; R_{20^\circ\mathrm{C}}}{C_{\mathrm{th}}}. \tag{2.6}$$

Considering the definitions made in the previous section, the respective white-box-model is displayed in figure 2.8, with $u(t) = I(t)$, $y(t) = P_{\mathrm{el}}(t)$ and $\mathcal{O}$ can be described by the equation set.

$$\begin{tikzcd} I(t) \arrow[r] & \boxed{y(t) = R(\vartheta(t)) \cdot u^2(t)} \arrow[r] & P_{\mathrm{el}}(t) \\ & \text{Resistor} & \end{tikzcd}$$

Figure 2.8 Example for a white-box-model.

Since the time-continuous progression of the resistor temperature can be described by a linear first order differential equation, it can be solved analytically with an exponential function and a time constant. The property of being solvable analytically is however rarely given in a real live case.

This leads to much more complex computations, which can only be solved numerically. For the given example this could be the computation of the spatial temperature distribution inside the electric resistor. The problem is becoming high dimensional and needs to be separated into multiple single computations. Typical methods to solve such a problem numerically are the Finite-volume method (FVM) or the Finite-element method (FEM) (according to [ZZT13]). Complex physical problems can be solved using these methods, because they are able to reduce problems to sub-problems, which can be linearized. Typically, these kinds of methods are used in the design phase of a system to guarantee the functionality of a system before it is actually built. Another numerical method to solve complex physical systems is the domain of Computational fluid dynamics (CFD). Typically, numerical solutions require high computational costs, which can sum up to days or weeks for single computations (based on [Bun+13]).

Definition 2.3 (Theoretical or white-box-model)
A theoretical or white-box-model is based on physical laws, can be described by a mathematical model and can be solved analytically or numerically in a finite time. The mathematical model can consist of integral-, (partial) differential- and algebraic equations.

2.2.2 Data-driven Models

As the name implies, the base of every data-driven model is any kind of data. The source of this data can either be observations of the real system (measurements) or observations using a white-box-model simulating the real system (simulations). Data-driven models do not rely on physical laws and they cannot be interpreted in a physical manner. Therefore, data-driven models are also called black-box-models. Mathematical methods are used to describe the relationship between the system input, system output and system state.

Data-driven modeling can be separated in parametric modeling approaches and non-parametric modeling approaches. According to [UB16], there is no sharp border between parametric and non-parametric models, but generally non-parametric models can be defined as models with an infinite number of parameters. An example for a non-parametric model of a linear, causal, time-invariant system can be explained using the convolution integral for a time-continuous system [UB16]

$$y(t) = \int_{-\infty}^{t} g(t-\tau)u(\tau)d\tau = \int_{0}^{t} g(\tau)u(t-\tau)d\tau \tag{2.7}$$

or the convolution sum for a time-discrete system [UB16]

$$y_k = \sum_{i=0}^{\infty} g_i \, u_{k-i}. \tag{2.8}$$

The respective model is described by the weighing function $g(t)$ or g_i, where τ is a time delta and i is a counting index. In contrast, a parametric model of a linear, causal, time-invariant and time-discrete system can be described by a linear difference equation [UB16])

$$y_k = -\sum_{i=1}^{\beta_y} a_i \, y_{k-i} + \sum_{i=0}^{\beta_u} b_i \, u_{k-i}, \tag{2.9}$$

where a and b are constant coefficients and β_y and β_u are parametric model coefficient which need to be identified using the observations.

For this identification, often parameter estimation methods like Least-squares estimation (LSE) or Maximum-likelihood estimation (MLE) are used. An overview of the common parameter estimation methods is given by [KRS11]. On the one hand, typical methods for parametric modeling are Multiple polynomial regression (MPR) or auto-regressive modeling like Autoregressive model with exogenous input (ARX) or Autoregressive model with moving average (ARMA). On the other hand, typical methods for non-parametric modeling are Support vector regression (SVR), Gaussian process regression (GPR) or Artificial neural network (ANN). Applying the given explanations to the example of the electric resistor, a mathematical model would be identified which describes the relation f between $I(t)$, $P_\mathrm{el}(t)$ and ϑ without knowing the physical principle of thermal conduction and convection (see Figure 2.9). The development of such a model requires observations of current, power and temperature without understanding the physics.

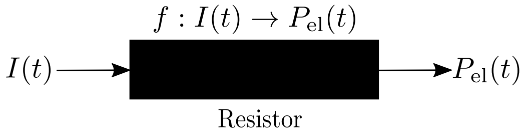


Figure 2.9 Example for a black-box-model.

One big advantage of data-driven models is the small development time. Many toolboxes are available for different development environments to create data-driven models and the developer does not necessarily have to be domain expert. Furthermore, data-driven models highly depend on the data that has been used for training. The approximation capability is limited by the observations that are provided by the system behavior. Depending on the method, the interpolation and extrapolation capabilities vary. Typically, all kinds of data-driven methods have their strength in interpolating in data ranges which are included in the observation space.

Definition 2.4 (Data-driven or black-box-model)
A data-driven or black-box-model is based on observations provided either by measurements or simulations, approximates the system behavior using a parametric or non-parametric mathematical model and can be solved analytically or numerically.

2.2.3 Knowledge-based Models

Another type of dynamic system modeling is based on expert knowledge and is called knowledge-based modeling. Basically, these models are developed by domain experts who know their systems for a long time and can describe their behavior by rules. According to [Kro16], a knowledge-based model consists of a rule base, a data base and evaluation rules. The rule base consists of a premise and a conclusion, comparable to an if-then relationship. Multiple rules are combined using inference methods which are specified in the evaluation rule, e.g. a MIN-MAX inference. Typical modeling methods for knowledge-based systems are models based on Fuzzy-Logic or lookup tables. Applying this to the electric resistor example, the linear differential equation for the temperature development (see Equation 2.6) could be simplified to Table 2.2. Equation 2.3 could be used as simple analytical equation and the complex part would be simplified by a lookup table.

Table 2.2 Lookup table for electric resistance depending on the temperature.

Temperature Range	**Resistance** $R(\vartheta)$
0°C – 50°C	$100\ k\Omega$
51°C – 100°C	$90\ k\Omega$
101°C – 150°C	$80\ k\Omega$
151°C – 200°C	$70\ k\Omega$
$\dots$	$\dots$

Definition 2.5 (Knowledge-based model)

A knowledge-based model is based on the experience of a domain expert and approximates the system behavior by rules and best practices.

2.2.4 Hybrid Models

The notion *hybrid* in terms of modeling can have several different meanings, but all have in common, that at least two models describing either the same system or subsystems are combined. Generally, the most common hybrid model definitions distinguish between continuous dynamics and discrete dynamics, parametric approximation functions and non-parametric functions as well as theoretical models and data-driven models. According to [SHH17], a hybrid model combining continuous and discrete system dynamics can be described by combining a state machine $\mathcal{Z}$ and a set of dynamic systems $\boldsymbol{x}^i$ to model a process which consists of continuous as well as discrete dynamics (see Figure 2.10). A switching signal is generated by the state machine $\mathcal{Z}$ based on the inputs u and decides which system dynamic $\boldsymbol{x}^i$ is used to predict the output y. In the example of [SHH17], all dynamic systems $\boldsymbol{x}^i$ are modeled using an ARX modeling approach, but of course the modeling method can be a different one without changing the architectural concept of the hybrid model. The active system is selected using the state machine $\mathcal{Z}$ which decides based on the input $\boldsymbol{u}$ and the output $\boldsymbol{y}$ which system is used to predict the dynamics.

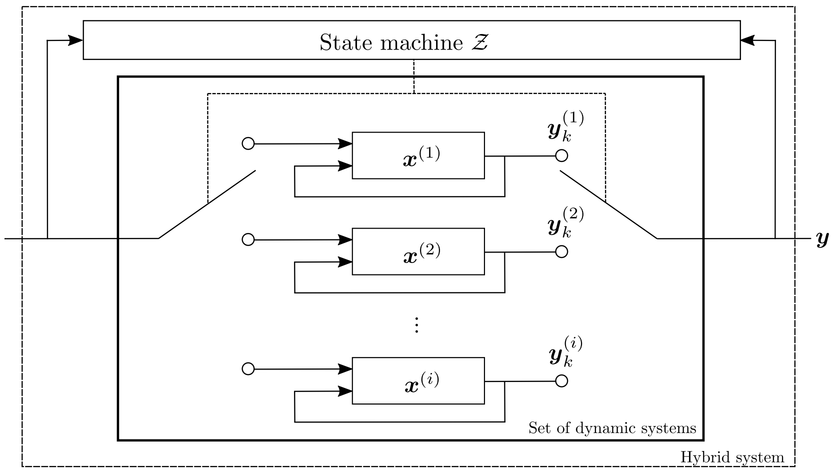


Figure 2.10 Hybrid system according to [SHH17].

Another type of hybrid modeling is introduced by [GS18] and [HLF17]. They differentiate between parametric and non-parametric methods which can be combined to a hybrid model using different architectures. Figure 2.11 a) shows a parallel architecture: a parametric model, in this case mechanistic[1], is used in parallel to a non-parametric model, in this case an ANN and they are combined to predict the output $\boldsymbol{y}$. Figure 2.11 b) and c) show serial architectures to create a hybrid model. In Figure 2.11 b) a mechanistic model feeds an ANN whereas in Figure 2.11 c) an ANN feeds a mechanistic model. Furthermore, [GS18] differentiates between a static hybrid model which is computing the output $\boldsymbol{y}(t_0)$ based on the input $\boldsymbol{u}(t_0)$ and a predictive model which is predicting the output $\boldsymbol{y}(t_1)$ based on the input $\boldsymbol{u}(t_0)$ and output $\boldsymbol{y}(t_0)$. In the context of process industry, these models are often called grey-box-models.

A respective mathematical model description of a parallel architecture

[1] Models being derived from conservation laws (material, momentum, and energy), thermodynamic, kinetic, and/or transport laws

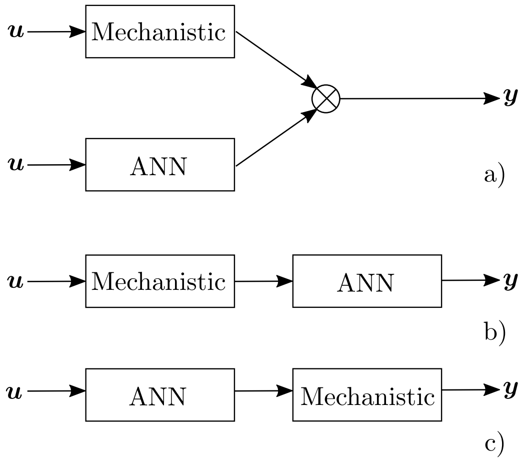


Figure 2.11 Architectures for hybrid models according to [GS18].

combining a parametric and non-parametric model is

$$\boldsymbol{y}_k = \beta \cdot \boldsymbol{x}_k + g(\boldsymbol{x}_k, \omega) + \epsilon. \tag{2.10}$$

The description for a serial architecture combining a parametric and a non-parametric model based on [GS18] is

$$\boldsymbol{y}_k = g(\beta \cdot \boldsymbol{x}_k, \omega) + \epsilon. \tag{2.11}$$

Hereby, ω describes the hyper-parameter of the non-parametric model, β describes the parameter of the parametric model and ϵ describes the residuals. Seeing the parametric and non-parametric modeling approach of [GS18] in a bigger scope, it can be interpreted as combining theoretical models and data-driven models as they were introduced before. This is exactly, how [GMZ16] defines the different architectures of hybrid prognostic approaches using dynamic systems for predictive maintenance purposes. Figure 2.12 a) shows the serial approach whereas Figure 2.12 b) shows the parallel approach. The architectures are similar to the ones which are introduced by [GS18], but they differ by especially focusing on data-driven and theoretical models.

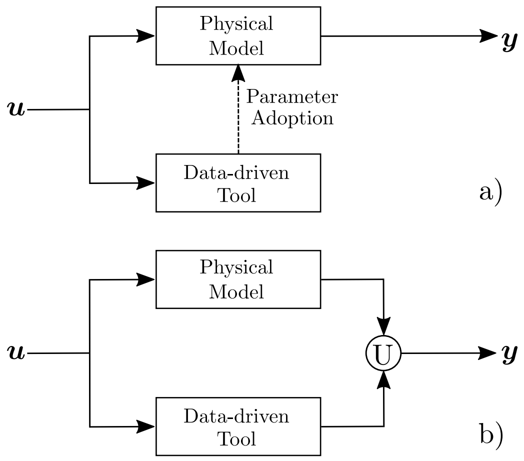


Figure 2.12 Architectures for hybrid models according to [GMZ16].

Concluding the introduced definitions of hybrid modeling approaches, a hybrid model can be identified by two main types: On the one hand, the sub-models which are combined describe different system dynamics by separating the system into sub-systems which have different system characteristics (see Table 2.1). On the other hand, the sub-models which are combined describe either a sub-system or the whole system, but they vary in the modeling type which has been used to create the sub-model. Applying this modeling approach to the example of the electric resistor, a hybrid model may be structured as follows: On the one hand, the thermal differential equation (see Equation 2.6) may be unknown due to a very complex physical process, but the temperature might be measurable. A data-driven model could be trained to learn the dynamics of the temperature and predict future temperatures, for example an ARX model. On the other hand, the physical law of electric power depending on the electrical resistance R and the electrical current I might be modeled as shown in equation 2.3. Therefore, a serial hybrid model could be constructed which describes the system dynamics both by a parametric and a non-parametric sub-model.

Definition 2.6 (Hybrid model)

A hybrid model is a serial or parallel combination of at least two sub-models which vary in their system characteristics or in the modeling approach which has been used to create the sub-model.

2.3 Multi-fidelity Modeling

Multi-fidelity modeling is a mathematical concept which is typically used to accelerate cost-intensive computations without loosing too much accuracy. According to [RK16], it is also extremely useful for solving inverse problems, which are ubiquitous in science. In the following sections, an overview will be given about model fidelity, model simplification methods and architectures of an Multi-fidelity model (MFM).

2.3.1 Model Fidelity

One of the main challenges for multi-fidelity modeling is the definition of the term *fidelity*. According to [CLK14], the *fidelity* of a model is defined as the degree to which the model reproduces the system properties being modeled. On the one hand, an High-fidelity model (HFM) has a high accuracy, but its computation speed is relatively slow. On the other hand, an Low-fidelity model (LFM) is relatively fast in computation speed, but lacks accuracy. On the right side of Figure 2.13, the accuracy when feeding an input $\boldsymbol{u}$ in both an HFM and an LFM and comparing the system output y to the reference or expected output is visualized. Figure 2.13 shows the definition of [CLK14] by visualizing the mention of accuracy feeding an input $\boldsymbol{u}$ in both an HFM and LFM. The HFM has a smaller deviation from the reference system output than the LFM.

Another similar but extended definition was made by [Fer+16], who defined that an HFM usually represents the behavior of the system to acceptable accuracy for the application intended. HFMs are usually expensive and multiple realizations often cannot be afforded. LFMs are cheaper and less accurate. This definition adds another important aspect of the fidelity of the model: the perspective of the application. Models which are realized for a design phase of a system may have another accuracy than models which are realized for the production or operation phase. [PWG18]

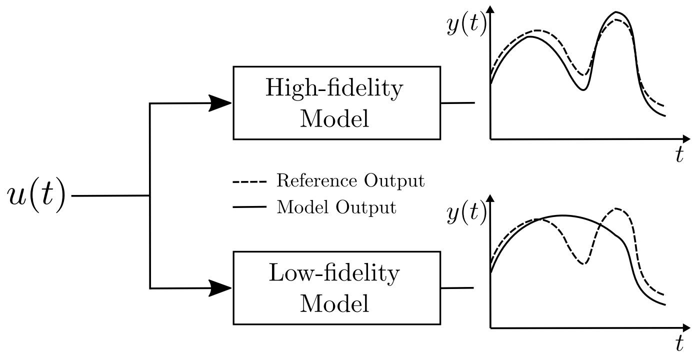


Figure 2.13 Example for accuracy-based HFM and LFM.

underline the given definitions and extend the application perspective by introducing different types of outer-loop applications which vary in their purpose. Figure 2.14 shows the block diagram of single- and multi-fidelity approaches with different outer-loop applications. According to [PWG18], outer-loop applications can be

- Optimization,
- Uncertainty Propagation,
- Data Assimilation,
- Control,
- Sensitivity Analysis.

The term *accuracy* needs to be subdivided into two main characteristics: The **amount of dimensions** which are taken into account for the simulation problem and the **approximation quality** of each of those dimensions. This differentiation is done due to the fact that the simulation time is mainly influenced by these criteria and therefore becomes a key topic for operation-parallel simulation. Concluding the introduced definitions, an HFM and an LFM in the context of different life cycle phases and operation-parallel simulation can be defined as:

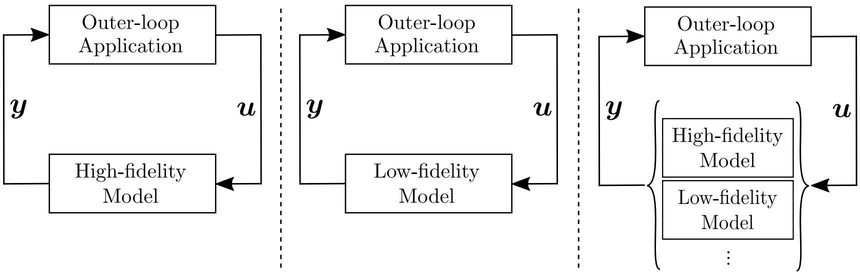


Figure 2.14 Outer-loop application for different fidelities according to [PWG18].

Definition 2.7 (High-fidelity model)
Theoretical model which estimates the system output with the dimension and approximation quality which is necessary for an outer-loop application from the design phase of the product life cycle.

Definition 2.8 (Low-fidelity model)
Theoretical or data-driven model which estimates the system output with a lower approximation quality or dimension than the high-fidelity model typically in favor of lower computational costs for an outer-loop application from the operation phase of the product life cycle.

In the following, the different modeling approaches will be represented as follows: $\mathcal{H}$ stands for a high-fidelity model operator with its corresponding superscription index H, $\mathcal{L}$ represents a low-fidelity model operator with its superscription index L. The computation of the system state $\boldsymbol{x}$ is assumed to be covered by the corresponding model operator. The resulting model equations also include the respective model error $\boldsymbol{w}_k^{\mathrm{H,L}}$:

$$\boldsymbol{y}_k^{\mathrm{H}} = \mathcal{H}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}}), \tag{2.12}$$

$$\boldsymbol{y}_k^{\mathrm{L}} = \mathcal{L}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{L}}). \tag{2.13}$$

The outer-loop applications which are addressed for operation-parallel simulation need to be modified in comparison to the introduced applications by [PWG18]. Outer-loop applications for operation-parallel simulation are purpose-driven, meaning the application fulfills a specific purpose in combination with the MFM. The outer-loop applications in an industrial context can be

- Descriptive Analytics: clarifying "what happened",
- Diagnostic Analytics: clarifying "why it happened",
- Predictive Analytics: clarifying "what will happen",
- Prescriptive Analytics: strategically controlling "what will happen".

2.3.2 Development Work Flow

Similar to hybrid models, MFM can be constructed in different ways and with different architectures and for different life cycle phases. This work focuses on the usage of an MFM during operation; therefore, development work flows for the design phase of a system, e.g. introduced by [Böh15], are not further discussed. A common work flow for the development of an MFM which is capable to be executed in parallel to the operation was introduced by [CLK14] and is visualized in Figure 2.15. This architecture describes the development of a combination of a time-continuous and a time-discrete system as introduced by [SHH17]. Additionally, the models for the respective system parts have various fidelities. [CLK14] separates the development of a multi-fidelity model into four main phases:

1. Target model selection and interest region definition
2. Low-fidelity model development
3. Multi-fidelity model composition
4. Selected target model substitution

First of all, the sub-model which should be replaced by an MFM needs to be selected and the interest region for which the MFM should be valid needs to be specified. Generally, this interest region should be smaller than the overall simulation region. Secondly, the development of the LFM needs to be done. In general, the LFM construction based on an HFM can be done by many different simplification methodologies. In Figure 2.16, the most common methodologies based on [Fer+16] are shown.

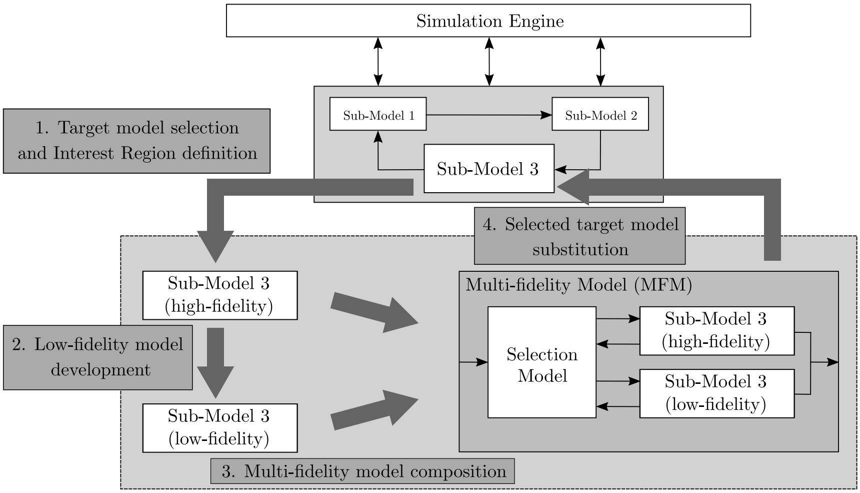

Simulation Engine
Sub-Model 1
Sub-Model 2
Sub-Model 3
1. Target model selection and Interest Region definition
4. Selected target model substitution
Sub-Model 3 (high-fidelity)
2. Low-fidelity model development
Sub-Model 3 (low-fidelity)
Multi-fidelity Model (MFM)
Selection Model
Sub-Model 3 (high-fidelity)
Sub-Model 3 (low-fidelity)
3. Multi-fidelity model composition

Figure 2.15 MFM methodology according to [CLK14].

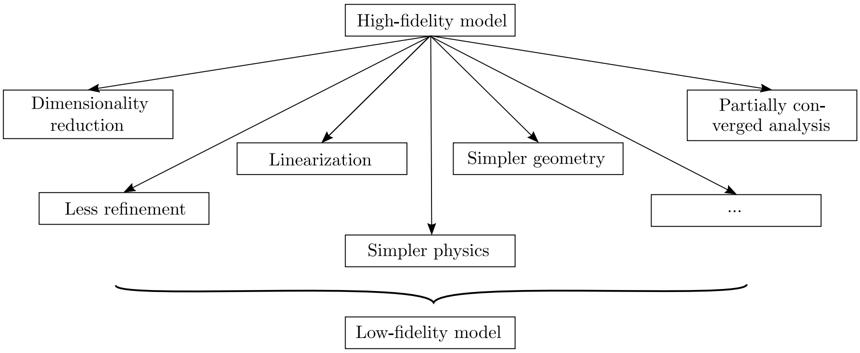


Figure 2.16 Model simplification according to [Fer+16].

Furthermore, an LFM is interpreted independently of the modeling approach which was used to construct it, meaning it can be both a simplified theoretical model or a data-driven model. An LFM which is still a theoretical model can be developed either by simplifying the model characteristics, e.g. using an early stopping criteria to get faster simulation results taking a worse accuracy into account. Additionally, projection-based models can be developed by using mathematical Model order reduction (MOR) methods like Krylov subspace methods. A survey about state-of-the-art MOR methods is given by [BGW15] and is not further discussed. The third type of creating an LFM are data-fit models, typically called surrogate models. The methods which are used to develop data-driven surrogates are parametric and non-parametric interpolation or regression methods, ANNs or support vector machines. In Figure 2.17, the different types of LFM development are visualized in an overview.

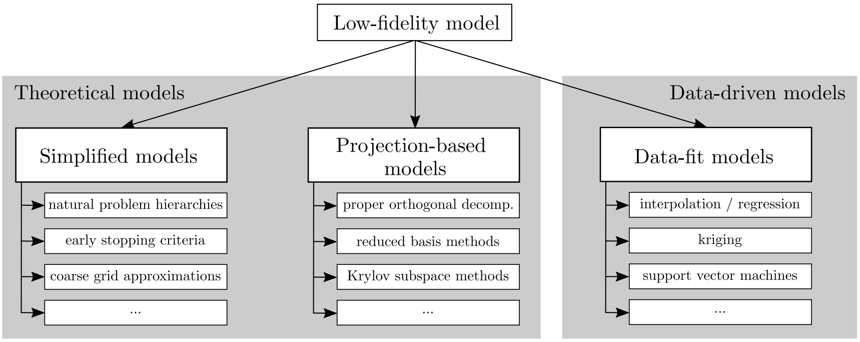


Figure 2.17 LFM types according to [Fer+16].

The third step of the architecture of [CLK14] is the composition of the MFM combining the HFM and newly developed LFM using a selection model for the different regions of interest. This selection model decides, which model is used to predict the output depending on the actual input. In the last step, the selected target model from the first step gets replaced by the newly developed MFM. This development work flow is the base for the further work which is going to be discussed in the following sections. It will be shown, that the development work flow can be modified such that other architectures of MFMs can be applied using the modified methodology. The main part of the research discussed in the following focuses on the third part of the development work flow, the multi-fidelity model composition.

3 Multi-fidelity Modeling for Operation-Parallel Simulation

This chapter focuses on the methodology to develop multi-fidelity models for operation-parallel simulation using low-fidelity models, high-fidelity models and observations. The concept of multi-model data fusion as well as the formalization of fusion methods and optimality criteria for most suitable fusion methods will be introduced. Finally, the methods which are explored as fusion operators are presented in detail.

3.1 Model Development

This section introduces the general modeling work flow to develop an operation-parallel simulation model. Based on [Lep+20], Figure 3.1 visualizes the different phases of data and model processing towards prescriptive analytics, which is the aim for operation-parallel simulation. First of all, the available models and the available observations need to be acquired from the respective data stocks. For simplification purposes, these two data stocks will be called *Operational Data Stock* and *Model Stock* in the following. The operational data stock includes observations acquired by sensors directly connected to the assets, extracted features from the fetched sensor data, simulation results using the extracted features as well as all information which is necessary for user interaction. Such a data stock could be part of an IIoT system, a manufacturing execution system or a process control system. Contrary to this, the model stock includes functional simulation models from different life cycle phases. Typically, such a model stock would be a PLM system.

The first step of the modeling work flow is the data and model preprocessing. On the operational data pre-processing side, this means especially to examine the data quality which is required for later usage

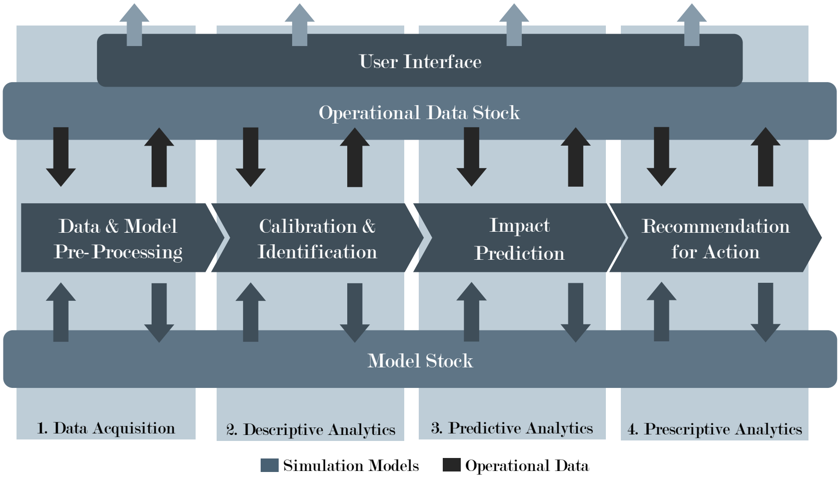


Figure 3.1 Data & model processing work flow for operation-parallel simulation.

including data consistency checks, time stamp adjustments or disturbance filtering. On the model pre-processing side, this includes the adaption of the model to guarantee the requirements of its later purpose, meaning for example the model order reduction from a high-fidelity model to a low-fidelity model. The pre-processed data streams and simulation models are stored back in their respective stocks after having done the necessary modifications.

In the second stage, the operational data is analyzed and the developed simulation models are calibrated to gain insight on the data. In detail, features are extracted from the operational data to describe the actual state of the observed system as well as pre-processed models from the first stage are calibrated using this operational data. Consequently, this allows a system operator to answer the question *what happened to the system* in case of a fault or anomaly. Furthermore, the extracted features along with the detected faults and observed data can be used to train fully data-driven models by using machine learning methods or system identification methods. The gained insights along with extracted features, calibrated models and new trained models are stored back to their respective stock afterwards.

Subsequent to stage two, the focus switches from gaining insight to predicting impact. This means, the type of question which should be answered changes from *what happened* to *what will happen*. Therefore, the models which have been calibrated in stage two are used now to predict future system behavior based on a-priori knowledge for the system input $\boldsymbol{u}$. Both the physics-based models from the design phase as well as the data-driven models trained during operation phase will predict the intended insights with different fidelities. This is the stage where the concept of multi-fidelity modeling for operation-parallel simulation is anchored. It is becoming more and more necessary to use the advantages of both data-driven as well as physics-based models to optimize the accuracy or dimension for the impact prediction stage.

Concluding the work flow of 3.1, in the last stage the question which should be answered changes from *what will happen* to *how can that what will happen be prevented*. This is the stage where only having a prediction-capable model is not enough to prevent events that will happen and damage the system. Domain expertise and deep system knowledge is required to influence the system behavior, e.g. to know which parameters need to be changed or how the system will react on a change of boundary conditions.

The developing work flow for operation-parallel multi-fidelity models is mainly located in the stages two and three of the work flow which is shown in Figure 3.1. Therefore, the development work flow for the multi-fidelity model itself will be introduced in detail in the following. It is based on the work of [PWG18] and [CLK14] (see section 2.3.2), but it is adapted to the needs of operation-parallel simulation. Figure 3.2 visualizes the elaborated development work flow to come up with a multi-fidelity model which can be used for operation-parallel simulation.

First of all, the sub-model describing the phenomena which shall be observed during operation as well as the region of interest regarding the later usage in the outer-loop application need to be defined. In contrast to the original model, a sub-model from the design phase is going to be reused to gain maximal value from the effort which has been spent to develop this model. For example, an FEM model is not just leveraged in the design phase but also the base to develop a model for the operation phase. Furthermore, the interest region definition is important, since

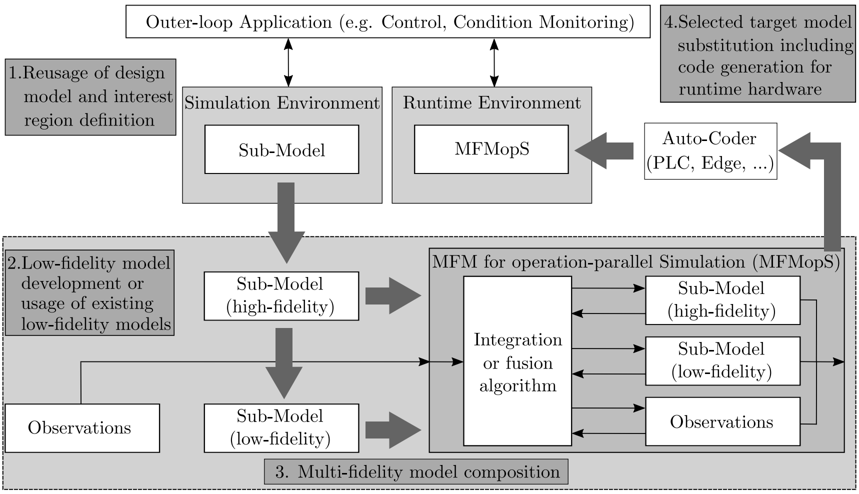

Outer-loop Application (e.g. Control, Condition Monitoring)
4.Selected target model substitution including code generation for runtime hardware
1.Reusage of design model and interest region definition
Simulation Environment
Runtime Environment
Sub-Model
MFMopS
Auto-Coder (PLC, Edge, ...)
2.Low-fidelity model development or usage of existing low-fidelity models
Sub-Model (high-fidelity)
MFM for operation-parallel Simulation (MFMopS)
Integration or fusion algorithm
Sub-Model (high-fidelity)
Sub-Model (low-fidelity)
Observations
Observations
Sub-Model (low-fidelity)
3. Multi-fidelity model composition

Figure 3.2 Multi-fidelity modeling work flow for operation-parallel simulation.

the downstream work flow will be to optimize the model regarding this region. According to [CLK14], the interest region can be defined as shown in Figure 3.3 for a two-dimensional problem. As an example, the variables A and B could be speed and torque of a motor. For the operation-parallel simulation with an outer-loop application of condition monitoring just the area of a speed in the range of 500 rpm - 1000 rpm and the torque in the range of 10 Nm - 20 Nm is relevant. The selection or development of a low-fidelity model highly depends on the interest region definition.

Following up with this, the next step after having identified the reusable model and the interest region is the development of a new or selection of an existing low-fidelity model. Considering Figure 3.1, this means that either an existing low-fidelity model is part of the model stock or a new model needs to be developed. How to develop an LFM from an existing HFM is already subject of discussion in section 2.3, hence it is not further discussed here. Again, it is important that the LFM meets the requirements of the defined interest region. Hereby, the concepts of data fusion can be used to construct the specific LFM to be part of the MFM, thus they will be introduced in detail. According to [RP06], data fusion concepts can be divided into three fundamental types: *Competitive Integration*, *Comple-*

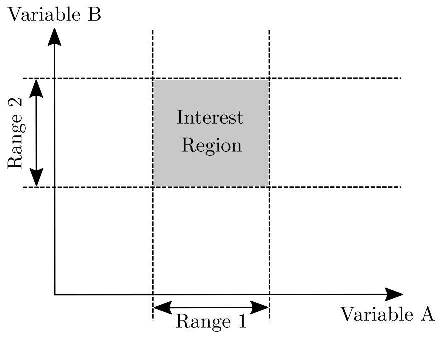


Figure 3.3 Definition of interest region according to [CLK14].

mentary Integration and *Cooperative Integration*. The concepts differ in the available information sources and have different purposes. By applying *Competitive Integration*, input sources with similar information are fused to reduce uncertainty. Relating this to the interest region definition, this transfers to two models, the LFM and HFM, which are both covering the full interest region. *Complementary Integration* is used to close information gaps by integrating similar input sources which have, for example, different resolutions. Again, setting this into relationship to the interest region definition, the LFM or HFM would extend each other. Translating this to the earlier used example of the interest region definition for a motor, the LFM covers the full speed and torque range but the HFM just covers a speed in the range of 600 rpm - 900 rpm and a torque in the range of 12 Nm - 18 Nm. If it is necessary to integrate information from input sources of different information and the target is to conclude to a measure of interest, this is called *Cooperative Integration*.

The third step of developing a multi-fidelity model for operation-parallel simulation is the model composition. In terms of the introduced concept of data fusion, the observations acquired from the system during operation serve as additional information source for both, calibrating the existing models and developing an MFM. One of the biggest differences to the work flow according to [CLK14] is the model composition. Hereby, no selection model is trained to decide which model, the LFM or HFM, is used based on boundary and operating conditions, similar to [SHH17].

Instead an algorithm is used to either integrate the different model types to a joint model or fuse the different models to a new surrogate model. In the following, this methodology will be called *Multi-Model Data Fusion*, analogously to *Multi-Sensor Data Fusion*. Based on the definitions of a Multi-fidelity hierarchical model (MFHM) and a Multi-fidelity surrogate model (MFSM) given by [Fer+16], Figure 3.4 shows the types of multi-fidelity models, either constructed by an integration algorithm or a fusion algorithm. An MFM is of the type MFHM, if information provided by different sub-models is combined by evaluating the models and integrating the evaluated output with either methods for redundant integration or methods for complementary integration. The models themselves are not going to be replaced for the purpose of operation-parallel simulation. Contrary, an MFM is of the type MFSM, if information is provided by sub-models which are combined to one surrogate model which is used for the purpose of operation-parallel simulation. As result of the training process, which will be introduced in section 3.2, an MFM for operation-parallel simulation (MFMopS) is developed.

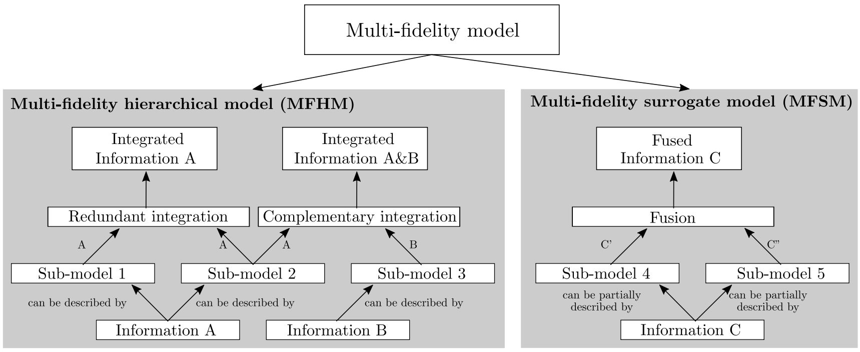


Figure 3.4 Types of multi-fidelity models based on [Fer+16].

A further differentiation to the work flow according to [CLK14] is the fourth step, the deployment of the developed Multi-fidelity for operation-parallel simulation (MFMopS) to a runtime environment, such that the outer-loop application can make use of the model during operation. Therefore, the developed MFMopS needs to be converted by the respective Auto-Coder for the target hardware to create an executable model which can be seamlessly integrated in an existing automation infrastructure. The Auto-Coder is not in the scope of this thesis, but it is required for the whole work flow. Auto-Coders are code converters which are able to deploy code to a specific hardware based on a script written in another programming language. After this has been done, the outer-loop application is able to communicate with the newly developed digital performance twin in form of a multi-fidelity model.

3.2 Multi-Model Data Fusion

This section addresses the theoretical concept of multi-model data fusion which will be introduced in detail. It will be differentiated between the two types of multi-fidelity models, the MFHM and MFSM, which are already introduced and shown in Figure 3.4. The mathematical representations $\mathcal{H}$ and $\mathcal{L}$ for an HFM and LFM are already introduced in section 2.3.1 and will be used in the following. Furthermore, a third type of model operator will be introduced: $\mathcal{H}^{\mathrm{r}}$ is considered to be executed operation parallel and therefore be an order-reduced model like introduced in Figure 2.16. In contrast, $\mathcal{H}$ is considered to be not executable during runtime, has a high dimension and requires a lot of computational effort to be solved. In Figure 3.5 the general block diagram for multi-model data fusion is shown. The input vector $\boldsymbol{u}_k$ is processed by the system operators $\mathcal{O}_i$ and the system outputs $\boldsymbol{y}^{O_i}$ are fused using the fusion operator $\circledcirc$ to predict the fused output vector $\boldsymbol{y}^{\mathrm{F}}$.

The main element of the block diagram is the fusion operator $\circledcirc$, which is a data fusion or integration method able to create MFHM or MFSM. The time-independent data fusion of the results of two general system outputs can be written as

$$\boldsymbol{y}^{\mathrm{F}} = \boldsymbol{y}^{O_1} \circledcirc \boldsymbol{y}^{O_2}. \tag{3.1}$$

To transfer the introduced concept to the combination of an LFM, an HFM and observations, the general system operators $\mathcal{O}_i$ need to be replaced by the respective system operator $\mathcal{L}$ or $\mathcal{H}^{\mathrm{r}}$. This results in different scenarios which are realizable. For the outer-loop application *Diagnostic Analytics* (see section 2.3), which can also be called *Condition Monitoring*,

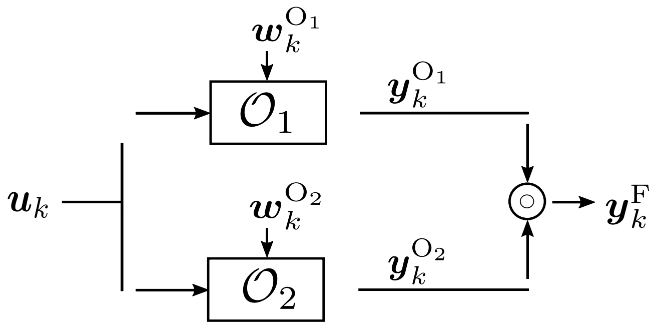

$\boldsymbol{w}_k^{\mathrm{O}_1}$
$\mathcal{O}_1$
$\boldsymbol{y}_k^{\mathrm{O}_1}$
$\boldsymbol{u}_k$
$\boldsymbol{w}_k^{\mathrm{O}_2}$
$\circledcirc$
$\boldsymbol{y}_k^{\mathrm{F}}$
$\mathcal{O}_2$
$\boldsymbol{y}_k^{\mathrm{O}_2}$

Figure 3.5 General block diagram of multi-model data fusion.

the block diagram looks like displayed in Figure 3.6 assuming that the operation-parallel model is an LFM.

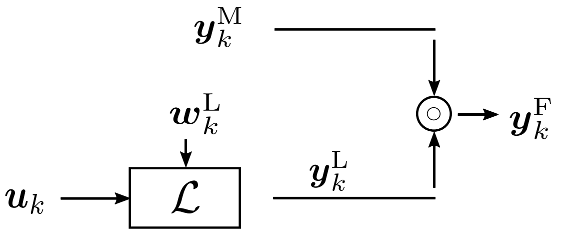
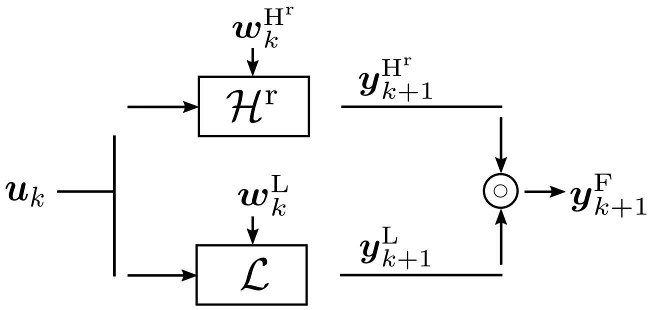

$\boldsymbol{y}_k^{\mathrm{M}}$
$\boldsymbol{w}_k^{\mathrm{L}}$
$\circledcirc$
$\boldsymbol{y}_k^{\mathrm{F}}$
$\boldsymbol{y}_k^{\mathrm{L}}$
$\boldsymbol{u}_k$
$\mathcal{L}$

Figure 3.6 Block diagram for diagnostic analytics.

Concluding the equations from section 2.3 a fusion or integration for diagnostic analytics can be written as

$$\boldsymbol{y}_k^{\mathrm{F}} = \boldsymbol{y}_k^{\mathrm{M}} \circledcirc \mathcal{L}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{L}}). \tag{3.2}$$

Another scenario considering the outer-loop applications *Predictive Analytics* and *Prescriptive Analytics* combines an HFM and an LFM, which typically bases on observations to predict future behavior based on known future input vectors. The resulting architecture is shown in Figure 3.7.



$\boldsymbol{w}_k^{\mathrm{H}^\mathrm{r}}$
$\mathcal{H}^\mathrm{r}$
$\boldsymbol{y}_{k+1}^{\mathrm{H}^\mathrm{r}}$
$\boldsymbol{u}_k$
$\boldsymbol{w}_k^{\mathrm{L}}$
$\circledcirc$
$\boldsymbol{y}_{k+1}^{\mathrm{F}}$
$\boldsymbol{y}_{k+1}^{\mathrm{L}}$
$\mathcal{L}$

Figure 3.7 Block diagram for predictive and prescriptive analytics.

Concluding again the equations from section 2.3, a fusion or integration for predictive or prescriptive analytics can be written as

$$\boldsymbol{y}_{k+1}^{\mathrm{F}} = \mathcal{H}^\mathrm{r}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}^\mathrm{r}}) \circledcirc \mathcal{L}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{L}}). \tag{3.3}$$

3.2.1 Fusion operator training

Selecting which integration or fusion method is the most suitable for a given application is the topic which will be discussed in the following. To find the best configuration to integrate or fuse model outputs or even different models is a very time-intensive challenge. Therefore, a methodology will be described in the following which allows an engineer to automatically find the most suitable method for a given problem based on a set of possible methods. First of all, a formalization to create a comparable base of different methods will be introduced.

Let Ω_b^a be a set of functions which are able to map the input domain $\mathbb{R}^a$ to the output domain $\mathbb{R}^b$, where a and b are the required input and output dimension for the fusion or integration. For the condition monitoring architecture, $a = \dim(\boldsymbol{y}^{\mathrm{M}}) + \dim(\boldsymbol{y}^{\mathrm{L}})$ applies and analogously for the output prediction architecture $a = \dim(\boldsymbol{y}^{\mathrm{H}^\mathrm{r}}) + \dim(\boldsymbol{y}^{\mathrm{L}})$ applies. Furthermore, $b = \dim(\boldsymbol{y}^{\mathrm{F}})$ applies for both architectures. The majority of integration or fusion methods can be parameterized, so let the set ${}^{p}\Gamma_b^a$ be a set of all functions with p parameters which are able to map to Ω_b^a.

$$\Omega_b^a := \{f | f : \mathbb{R}^a \to \mathbb{R}^b\} \tag{3.4}$$

$$^{p}\Gamma_b^a := \{f | f : \mathbb{R}^p \to \Omega_b^a\} \tag{3.5}$$

To illustrate the definitions, an example of the fusion method Weighed Arithmetic Average Filter (WAAF) with $p = 4$ parameters will be explained more in detail. A WAAF is an element of the set $^{4}\Gamma_1^1$ because this function is able to map $\mathbb{R}^1$ to $\mathbb{R}^1$ with 4 parameters, here α_i:

$$\bar{x} = \frac{\sum\limits_{i=1}^{4} \alpha_i \cdot x_i}{\sum\limits_{i=1}^{4} \alpha_i}. \tag{3.6}$$

For the training of the fusion operator, different sets are going to be introduced: For the condition monitoring architecture, let P_{C} be a set which consists of N_1^{P} tuples of the form $\{\boldsymbol{y}_k^{\mathrm{M}}, \boldsymbol{y}_k^{\mathrm{L}}, \boldsymbol{y}_k^{\mathrm{F}}\}$. For training purposes, either observations of the expected fusion results and dimension of $\boldsymbol{y}^{\mathrm{F}}$ are available and used as ground truth or the ground truth has to be generated artificially by running high-fidelity simulations and adapt the results to the expected application, $\boldsymbol{y}_k^{\mathrm{F}} \sim \mathcal{H}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}})$, considering measurement uncertainties and disturbances. The training architecture for the condition monitoring architecture is shown in Figure 3.8 and ca be formulated as

$$\mathcal{H}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}}) = \boldsymbol{y}_k^{\mathrm{M}} \circledcirc \mathcal{L}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{L}}). \tag{3.7}$$

For the output prediction architecture, let P_{O} be a set which consists of N_1^{O} tuples of the form $\{\boldsymbol{y}_k^{\mathrm{L}}, \boldsymbol{y}_k^{\mathrm{H^r}}, \boldsymbol{y}_k^{\mathrm{F}}\}$. Again, either observations of $\boldsymbol{y}^{\mathrm{F}}$ are available and can be used as ground truth or adapted high-fidelity simulations $\boldsymbol{y}_k^{\mathrm{F}} \sim \mathcal{H}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}})$ are used as ground truth. The training diagram for the output prediction architecture is visualized in Figure 3.9 and can be formulated as

$$\mathcal{H}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H}}) = \mathcal{H}^{\mathrm{r}}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{H^r}}) \circledcirc \mathcal{L}(\boldsymbol{u}_k, \boldsymbol{w}_k^{\mathrm{L}}). \tag{3.8}$$

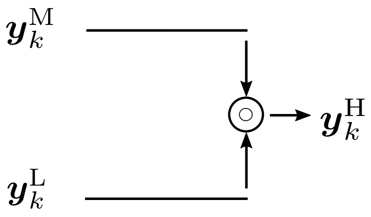

$\boldsymbol{y}_k^{\mathrm{M}}$
$\boldsymbol{y}_k^{\mathrm{H}}$
$\boldsymbol{y}_k^{\mathrm{L}}$

Figure 3.8 Block diagram for diagnostic analytics training.

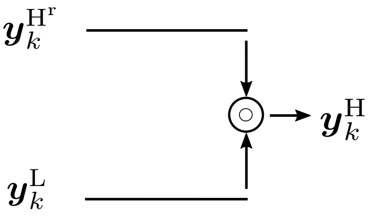

$\boldsymbol{y}_k^{\mathrm{H^r}}$
$\boldsymbol{y}_k^{\mathrm{H}}$
$\boldsymbol{y}_k^{\mathrm{L}}$

Figure 3.9 Block diagram for predictive analytics training.

Additionally, let F_{f} be a pre-selected subset of ${}^{\mathrm{P}}\Gamma_b^a$ including the possible fusion or integration methods and let m be the method under consideration for the training algorithm. Furthermore, let ϱ be the parameters of the specific method m under consideration. Consulting the definitions given so far, the fusion operator $\circledcirc$ for the condition monitoring architecture can be determined through the hierarchical minimization

$$\circledcirc = \underset{m \in F_{\mathrm{f}} \subset \Gamma}{\arg\min} \min_{\varrho} \left\{ \left\| m[\varrho](\boldsymbol{y}^{\mathrm{M}}, \boldsymbol{y}^{\mathrm{L}}) - \boldsymbol{y}^{\mathrm{H}} \right\|_2 \right\}, \qquad \forall (\boldsymbol{y}^{\mathrm{M}}, \boldsymbol{y}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{H}}) \in P_{\mathrm{C}} \tag{3.9}$$

and the fusion operator for the output prediction architecture can be determined through the hierarchical minimization

$$\circledcirc = \underset{m \in F_\mathrm{f} \subset \mathit{\Gamma}}{\arg\min} \min_{\varrho} \left\{ \left\| m[\varrho](\boldsymbol{y}^{\mathrm{H}^\mathrm{r}}, \boldsymbol{y}^{\mathrm{L}}) - \boldsymbol{y}^{\mathrm{H}} \right\|_2 \right\}, \qquad \forall (\boldsymbol{y}^{\mathrm{H}^\mathrm{r}}, \boldsymbol{y}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{H}}) \in P_\mathrm{O} \,. \tag{3.10}$$

3.2.2 Explored Methods as Fusion Operator

Since many fusion and integration methods are suitable to solve the considered problems, this work focuses on the deeper investigation of four different methods for the purpose of multi-fidelity modeling for operation-parallel simulation:

- The parametric feature-based method of multi-variate regression assuming a specific regression model in the background which is here a *Quadratic model with interactions (QMI)*
- The parametric method *Multi-layer perceptron (MLP)*
- The combined parametric method *moSAIc*, a combination of an MLP and a Weighed Arithmetic Average Filter, which is one result of this work
- The non-parametric method *Co-Kriging* which is based on Gaussian Process regression

These methods are selected because they fit well for regression purposes and especially for multi-dimensional problems. In Figure 3.10 an overview of different fusion methods is visualized and the selected methods are highlighted in bold letters.

3.2.2.1 Multivariate Regression

Regression analysis in general is used to approximate functions based on a given observation set using a predefined model. One commonly

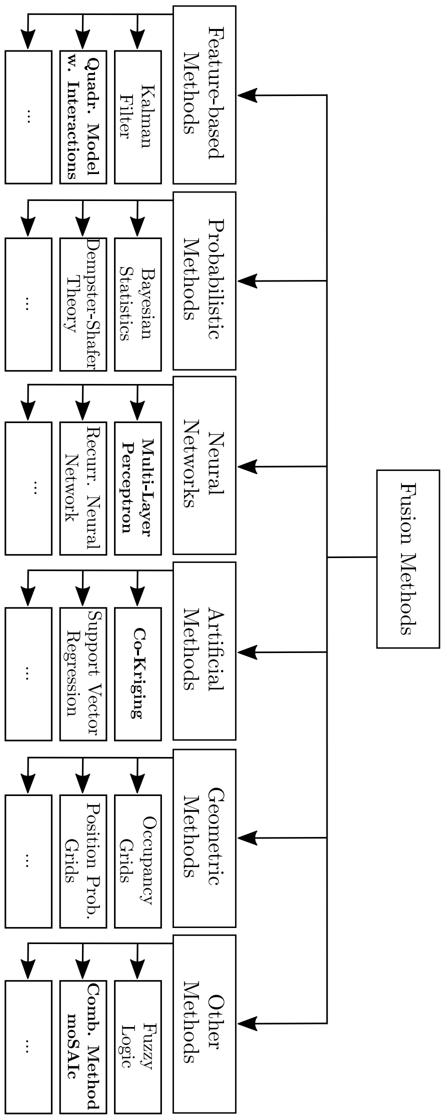


Figure 3.10 Possible methods for fusion operators adapted from [RP06].

used approximation for a mapping $\mathbb{R}^1 \rightarrow \mathbb{R}^1$ is the polynomial regression, where a polynomial of the form

$$y(\mathrm{x}) = \sum_{i=0}^{n} \alpha_i \mathrm{x}^i \tag{3.11}$$

is used to approximate an input-output-relationship using the parameter vector $\boldsymbol{\alpha}$. In this work, multi-dimensional regression problems are considered such that a model is required which is able to map $\mathbb{R}^a \rightarrow \mathbb{R}^b$. Therefore, $b = \dim(\boldsymbol{y}^{\mathrm{F}})$ stand-alone regression models are trained which have $a = \dim(\boldsymbol{y}^{\mathrm{M}}) + \dim(\boldsymbol{y}^{\mathrm{L}})$ variables. Every single regression model is able to map $\mathbb{R}^a \rightarrow \mathbb{R}^1$ and accumulated all models together are able to map from $\mathbb{R}^a \rightarrow \mathbb{R}^b$. Assuming a fusion problem with a dimension $\dim(\boldsymbol{y}^{\mathrm{F}}) = 1$, $\dim(\boldsymbol{y}^{\mathrm{M}}) = 1$ and $\dim(\boldsymbol{y}^{\mathrm{L}}) = 1$, this translates in a model function

$$y^{\mathrm{F}} = \alpha_0 + \boldsymbol{\alpha}_1 \cdot y^{\mathrm{M}} + \boldsymbol{\alpha}_2 \cdot y^{\mathrm{L}} + \boldsymbol{\alpha}_3 \cdot y^{\mathrm{M}} \cdot y^{\mathrm{L}} + \boldsymbol{\alpha}_4 \cdot (y^{\mathrm{M}})^2 + \boldsymbol{\alpha}_2 \cdot (y^{\mathrm{L}})^2. \tag{3.12}$$

This model function is of the shape *QMI*. For the derivation of the general model function for this regression model type, the multi-index ψ according to [Sai91] is introduced:

$$\psi = (\psi_1, \psi_2, \cdots, \psi_{N_{\mathrm{Q}}}), \tag{3.13}$$

$$N_{\mathrm{Q}} = \dim(\mathbf{X}), \tag{3.14}$$

$$\mathbf{X}^{\psi} = \prod_{i=1}^{N_{\mathrm{Q}}} \mathrm{x}_i^{\psi_i} = \mathrm{x}_1^{\psi_1} \cdot \mathrm{x}_2^{\psi_2} \cdots \mathrm{x}_{N_{\mathrm{Q}}}^{\psi_{N_{\mathrm{Q}}}}, \tag{3.15}$$

$$|\psi| = \sum_{i=1}^{n} |\psi_i|. \tag{3.16}$$

Exemplary, a multiplication $a \cdot \mathrm{x}^{(1,1,0)}$ is equal to $a \cdot \mathrm{x}_1 \cdot \mathrm{x}_2$ for $\psi = (1, 1, 0)$ and a multiplication $a \cdot \mathrm{x}^{(0,0,2)}$ is equal to $a \cdot \mathrm{x}_3^2$ for $\psi = (0, 0, 2)$. To formulate the general model equation for a *QMI*, the possible shapes of ψ need to be restricted. In general, the maximum value of the multi-index is just allowed to be 2 because just constant, linear and quadratic terms are part of the model. It is also required, that if one element of a multi-index has the value 2 every other element need to have the value 0, since quadratic terms are just considered single by single. Therefore, let Ψ be the set of all possible multi-indices ψ, for which applies:

$$\Psi = \left\{ \psi \in \mathbb{N}_0^{N_\mathrm{Q}} \,|\, \max(\psi) \leq 2 \wedge \max(\psi) = 2 \iff |\psi| = 2 \right\}. \tag{3.17}$$

Using the introduced definitions for the formulation of the general model equation for a *QMI*, this results in

$$y(\mathbf{X}) = \sum_{\psi \in \Psi} \alpha_\psi \, \mathbf{X}^\psi. \tag{3.18}$$

Transferring this to the fusion problem arithmetic, the fusion of the i-th element of $\boldsymbol{y}^\mathrm{F}$ with the purpose of condition monitoring, where

$$\boldsymbol{y}^\mathrm{I} = \begin{bmatrix} \boldsymbol{y}^\mathrm{M} \\ \boldsymbol{y}^\mathrm{L} \end{bmatrix}^\mathrm{T} \tag{3.19}$$

is the input vector consisting of the stacked vectors $\boldsymbol{y}^\mathrm{M}$ and $\boldsymbol{y}^\mathrm{L}$, can be written as

$$\boldsymbol{y}_i^\mathrm{F} = \sum_{\psi_i \in \Psi} \alpha_{\psi_i} (\boldsymbol{y}^\mathrm{I})^{\psi_i}. \tag{3.20}$$

Analogously, this is valid for the purpose of output prediction by changing $\boldsymbol{y}^\mathrm{M}$ to $\boldsymbol{y}^{\mathrm{H}^\mathrm{r}}$. The *QMI* is trained using a Least-Squares-Estimator to determine the parametric model vector $\boldsymbol{\alpha}$. Assuming a training data set with N samples, the approximation approach, according to [Pue15], can be described as

$$\begin{aligned} \boldsymbol{y}_i^\mathrm{F} = \begin{bmatrix} y_i^\mathrm{F}(k_0) \\ \vdots \\ y_i^\mathrm{F}(k_{N-1}) \end{bmatrix} &= \begin{bmatrix} y_1^\mathrm{I}(k_0) & \dots & (y_{N_\mathrm{Q}}^\mathrm{I})^2(k_0) & \alpha_0 \\ \vdots & \ddots & \vdots & \vdots \\ y_1^\mathrm{I}(k_{N-1}) & \dots & (y_{N_\mathrm{Q}}^\mathrm{I})^2(k_{N-1}) & \alpha_0 \end{bmatrix} \begin{bmatrix} \alpha_{\psi_i,1} \\ \vdots \\ \alpha_{\psi_i,N_\mathrm{Q}} \end{bmatrix} \\ &= \qquad\qquad \Phi \qquad\qquad \boldsymbol{\alpha}_{\psi_i}, \end{aligned} \tag{3.21}$$

where Φ is the training set matrix. Additionally, [Pue15] derives, that the parameter vector $\boldsymbol{\alpha}$ can be determined by using the sum of the squared approximation errors as quality measure resulting in the equation

$$\boldsymbol{\alpha}_{\psi_i} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\boldsymbol{y}_i^{\mathrm{F}} \tag{3.22}$$

for the computation of the parametric model vector $\boldsymbol{\alpha}$. In the form the method has been introduced so far, it is able to map $\mathbb{R}^a \rightarrow \mathbb{R}^1$. To be considered as a possible fusion operator $\circledcirc$, it needs to be able to map $\mathbb{R}^a \rightarrow \mathbb{R}^b$, which leads to the final method definition:

$$\boldsymbol{y}^{\mathrm{F}} = \begin{bmatrix} \sum\limits_{\psi_1 \in \Psi} \alpha_{\psi_1} \, (\boldsymbol{y}^{\mathrm{I}})^{\psi_1} \\ \vdots \\ \sum\limits_{\psi_b \in \Psi} \alpha_{\psi_b} \, (\boldsymbol{y}^{\mathrm{I}})^{\psi_b} \end{bmatrix} = q[\boldsymbol{\alpha}](\boldsymbol{y}^{\mathrm{I}}). \tag{3.23}$$

The resulting overall parameter vector $\boldsymbol{\alpha}$ for the *quadratic model interactions* is defined as $\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_{\psi_i} & \dots & \boldsymbol{\alpha}_{\psi_b} \end{bmatrix}^{\mathrm{T}}$. Concluding all the introduced definitions and applying the formalization introduced in 3.2, the fusion operator can be determined by

$$\circledcirc = \min_{\boldsymbol{\alpha}} \left\{ \left\| q\left[\boldsymbol{\alpha}\right](\boldsymbol{y}^{\mathrm{M}}, \boldsymbol{y}^{\mathrm{L}}) - \boldsymbol{y}^{\mathrm{H}} \right\|_2 \right\}. \tag{3.24}$$

3.2.2.2 Multi-Layer Perceptron

An MLP is a specific type of a neural network and is commonly used for regression and classification problems nowadays. In the following, a brief introduction on neural networks will be given. For deeper knowledge the work of [Kub17], [Kru+15] and [Kro16] can be considered. Figure 3.11 shows the basic concept of a neuron according to [Kro16], including the respective input or output signal o, the weight matrix $\boldsymbol{W}$, the propagation function and its result net, the activation function f_{act}, the neuron threshold and the indices i and j.

The MLP consists of many neurons structured in layers and neurons per layer and has no recursive part, the actual state of a neuron does not

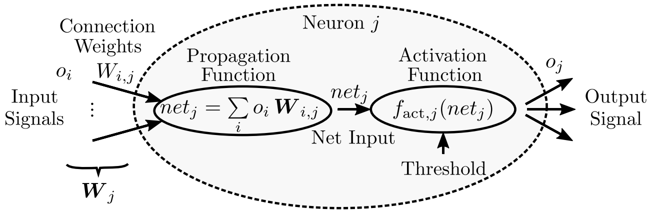


Figure 3.11 Data processing in a neuron according to [Kro16].

depend on the previous state because an MLP is a feed-forward network and has no memory. One small MLP with a 2-3-1 architecture is illustrated in Figure 3.12.

The corresponding forward path equation for this architecture with an assumption of a non-linear activation function $f_{\text{act}} = \tanh()$ is

$$o_1 = f_{\text{act, 1}} \left[\sum_{i=1}^{3} W_{i,1} \cdot f_{\text{act, } i} \left(\sum_{j=1}^{2} W_{j,i} \cdot v_j \right) \right]. \tag{3.25}$$

An MLP is typically trained using the delta rule or using the concept of back-propagation. The latter is now explained in more detail. Let E be the error function which should be minimized in the training process, for example a simple quadratic error function (see [Kro16]), then

$$E(c) = \frac{1}{2} \sum_{\zeta=1}^{\xi} (\kappa_\zeta(c) - \chi_\zeta(c))^2 \tag{3.26}$$

applies where ζ is the index of the layer which should be optimized, ξ is the number of all layers, $\chi_\zeta(c)$ is the layer output of layer ζ, $\kappa_\zeta(c)$ is the target value and c are the samples which are used for training. For the target value $\kappa_\zeta(c)$, it has to be differentiated if the weights to be trained are connected to the MLP output o or connected to a hidden layer. For the

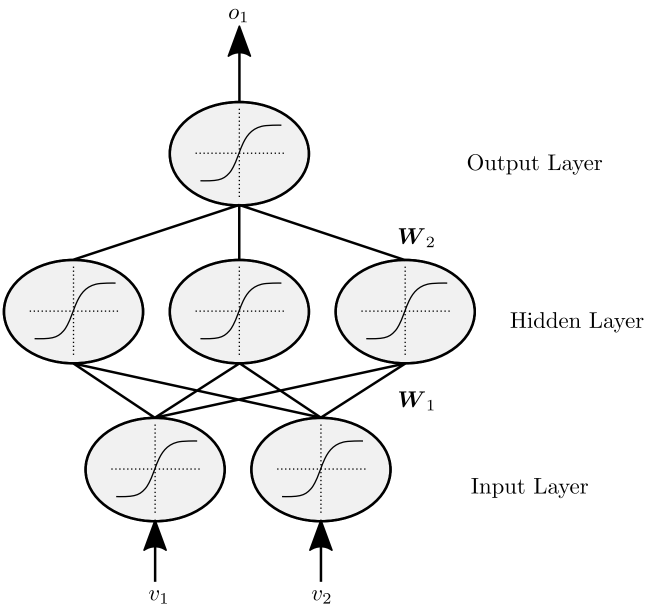


Figure 3.12 MLP example with 2-3-1 architecture.

first case, it can be shown [Kro16] that

$$\frac{\partial E(c)}{\partial W_{i,j}} = \left(\frac{\partial E(c)}{\partial o_j}\right) \cdot \left(\frac{\partial o_j(c)}{\partial W_{i,j}}\right), \tag{3.27}$$

$$\frac{\partial E(c)}{\partial W_{i,j}} = -(\kappa_j(c) - \chi_j(c)) \cdot f'_{\text{act}, j}\,(net_j) \cdot \chi_i(c) \tag{3.28}$$

and for the second case, it can be shown that

$$\frac{\partial E(c)}{\partial W_{i,j}} = (\kappa_\zeta(c) - \chi_\zeta(c)) \cdot f'_{\text{act},l}(net_l(c)) \cdot W_{j,\zeta} \cdot f'_{\text{act},j}(net_j(c)) \cdot o_i(c). \tag{3.29}$$

The updated weights are computed by multiplying the learning rate η with the computed gradient of the specific layer and adding this to the existing weights, resulting in

$$W_{i,j}^{\text{new}} = W_{i,j}^{\text{old}} + \eta \frac{\partial E(c)}{\partial W_{i,j}}, \tag{3.30}$$

$$W_{i,j}^{\text{new}} = W_{i,j}^{\text{old}} + \eta \left(-(\kappa_j(c) - \chi_j(c)) \cdot f'_{\text{act},\, j}\left(net_j(c)\right) \cdot \chi_i(c) \right) \tag{3.31}$$

for the first case and

$$\begin{aligned} W_{i,j}^{\text{new}} &= W_{i,j}^{\text{old}} + \\ \eta &\left((\kappa_\zeta(c) - \chi_\zeta(c)) \cdot f'_{\text{act},\, \zeta}\left(net_\zeta(c)\right) \cdot W_{j,\zeta} \cdot f'_{\text{act},\, j}\left(net_j(c)\right) \cdot \chi_i(c) \right) \end{aligned} \tag{3.32}$$

for the second case. Let θ_h be the vector which includes all parameters of the MLP, so every weight and every threshold which is necessary to propagate an input signal to become an output signal and let h be the function which computes a forward path of the trained MLP, such that $h\left[\theta_h\right](\boldsymbol{v}) = \boldsymbol{o}$ is valid. Transferring this now to the formalization which has been introduced in 3.2.1, an MLP with an input dimension of $a = \dim(\boldsymbol{y}^{\text{I}}) = \dim(\boldsymbol{v})$, an output dimension of $b = \dim(\boldsymbol{y}^{\text{H}}) = \dim(\boldsymbol{o})$ and $N_{\text{p}} = \dim(\theta_h)$ parameters could be used for a condition monitoring fusion problem, with $\boldsymbol{y}^{\text{I}} = \begin{bmatrix} \boldsymbol{y}^{\text{M}} & \boldsymbol{y}^{\text{L}} \end{bmatrix}^{\text{T}}$. For example, the MLP shown in Figure 3.12 is part of the set ${}^{15}\mathit{\Gamma}_1^2$. Therefore, the inner optimization problem of the hierarchical optimization problem to determine the best fusion or integration method considering an MLP as method can be written as

$$\circledcirc = \min_{\theta_h} \left\{ \left\| h\left[\theta_h\right](\boldsymbol{y}^{\text{I}}) - \boldsymbol{y}^{\text{H}} \right\|_2 \right\}. \tag{3.33}$$

3.2.2.3 moSAIc

As part of this work, the combined parametric fusion method *moSAIc* has been developed. *moSAIc* is combination of an MLP and a Weighed arithmetic average filter (WAAF), where the weights $\boldsymbol{\alpha}$ of the WAAF are determined by an MLP based on the system input $\boldsymbol{u}$ and static descriptive system information $\boldsymbol{C}$. The method is designed to fuse different dynamic system model outputs to a joint system output. The different systems need to be describable by the same features $\boldsymbol{C}$. To clarify the sense of

$$\begin{tikzcd}\boldsymbol{u}(t) \arrow[r] & \boxed{\dfrac{C_1}{C_2\, s^2 + C_3\, s + C_4}} \arrow[r] & \boldsymbol{y}(t)\end{tikzcd}$$

Figure 3.13 PT2 system in Laplace space.

the static descriptive system information $\boldsymbol{C}$, the PT2-System in Figure 3.13 is used. The assumed phenomena behind the model may be based on a physical relationship which can be described by a second-order differential equation, for example Newton's equation of motion. Hereby, a white-box-model approach is used because the law lying under the physics is known, such that the parameters of the white-box-model just need to be identified for the specific object, for which the equation of motion should be modeled. Once the parameters have been identified and validated, the parameters are fixed. The fixed parameters C_1 to C_4 are then defined as the static descriptive system information $\boldsymbol{C}$, because they describe Newton's equation of motion for this specific object. For another object, other parameters $\boldsymbol{C}$ may be valid and would need to be identified, but the features C_1 to C_4 are the same. Transferring this to a later introduced industrial application, the rotor length, housing length and housing material are influencing the lifetime of a motor. The features are static and differ in value depending on the motor derivative which is observed, but the physical impact is the same.

moSAIc is a hybrid fusion approach since an MLP and a WAAF are combined. The fused result for the i-th element of $\boldsymbol{y}^{\mathrm{F}}$ can be computed by

$$y_i^{\mathrm{F}} = \boldsymbol{\alpha}_i\, \boldsymbol{y}^{\mathrm{I}} \quad \text{with} \quad \boldsymbol{\alpha}_i = h_i[\theta_h](\boldsymbol{C}, \boldsymbol{u}). \tag{3.34}$$

A novelty of this approach is the joint training process of the MLP and WAAF. For the MLP $h : \mathbb{R}^{\dim(\boldsymbol{u})} \times \mathbb{R}^{\dim(\boldsymbol{C})} \to \mathbb{R}^{\dim(\boldsymbol{y}^{\mathrm{I}})}$ applies, because the parameters $\boldsymbol{\alpha}_i$ have to be determined. During the training process the aim is to fit the parameters θ_h of h such that

$$\boldsymbol{y}^{\mathrm{F}} \approx < \boldsymbol{\alpha}, \boldsymbol{y}^{\mathrm{I}} > \tag{3.35}$$

where $< .,. >$ is the inner product. This leads to the modified partial derivative

$$\frac{\partial E(c)}{\partial W_{i,j}} = -\left(\frac{y_i^{\mathrm{F}}(c) - \mathbb{L}_{\alpha}^{\beta}}{y_{\beta}^{\mathrm{I}}(c)} - \chi_j(c) \right) \cdot \left(f'_{\text{act},\, j}\, (net_j(c)) \cdot \chi_i(c)\right), \tag{3.36}$$

for $y^{\mathrm{I}}_{\beta}(c) \neq 0$ and with

$$\mathbb{L}^{\beta}_{\alpha} := \sum_{\substack{i=0 \\ i\neq\beta}}^{\dim(\boldsymbol{y}^{\mathrm{I}})} \alpha_i \, y^{\mathrm{I}}_i. \tag{3.37}$$

Hereby, the weights should not adapt to the target value but to the dividend $(y^{\mathrm{F}}_i - \mathbb{L}^{\beta}_{\alpha})/y^{\mathrm{I}}_{\beta}$. Hereby, $\mathbb{L}^{\beta}_{\alpha}$ indicates the sum of all products of the non-relevant elements for the training process.
The update function of the last layer therefore changes to

$$W^{\text{new}}_{i,j} = W^{\text{old}}_{i,j} + \eta \left(- \left(\frac{y^{\mathrm{F}}_i(c) - \mathbb{L}^{\beta}_{\alpha}}{y^{\mathrm{I}}_{\beta}(c)} - \chi_j(c) \right) \cdot f'_{\text{act, } j} \left(net_j(c) \cdot \chi_i(c) \right) \right). \tag{3.38}$$

In Figure 3.14, *moSAIc* can be seen as part of a block diagram to understand the relationships between the models and the composition of $\boldsymbol{u}$, what would be $\boldsymbol{u} = [\boldsymbol{u}_1 \; \boldsymbol{u}_2 \; \boldsymbol{u}_3]^{\mathrm{T}}$ in this case.

Some modifications have to be made to generalize the method to be used as fusion operator $\circledcirc$, because it has to be able to map $\mathbb{R}^a \to \mathbb{R}^b$, where $a = \dim(\boldsymbol{y}^{\mathrm{I}})$ and $b = \dim(\boldsymbol{y}^{\mathrm{F}})$. One possibility is to create a single model for each output dimension, like it is done for the multi-variate regression model described in 3.2.2.1. But this leads to a really high amount of parameters N_{p} and a time-intensive training phase due to a separate MLP for every output dimension. Therefore, only one MLP is considered to be used for the dimension extension. Let $\Upsilon^{\mathrm{I}} \in \mathbb{R}^{D\times J}$ be a matrix which consists of $D = \dim(\boldsymbol{y}^{\mathrm{F}})$ lines and J columns, where J is the amount of different models on the input side. Additionally, let $A^{\mathrm{m}} \in \mathbb{R}^{J\times D}$ be the corresponding mapping matrix. Every element of $\boldsymbol{y}^{\mathrm{I}}_i$ needs to be mapped to the respective entry in Υ^{I}. If the dimension of $\boldsymbol{y}^{\mathrm{I}}_i$ and for example $\boldsymbol{y}^{\mathrm{I}}_{i+1}$ is different, this results in inconsistencies for the mapping, because some dimensions are provided on the input side and some are not. The dimensions which are not available for in a model $\boldsymbol{y}^{\mathrm{I}}_i$ but are required for Υ^{I} may be filled with a 0. For the training of the MLP, the target output layer is assembled by $\mathrm{vec}(A^{\mathrm{m}})$ with $\mathrm{vec}(\cdot)$ being the matrix vectorization function. The multidimensional *moSAIc* function can be described by

$$\boldsymbol{y}^{\mathrm{F}} = A^{\mathrm{m}} \, \Upsilon^{\mathrm{I}} = m^{\mathrm{s}}[\theta_h](\boldsymbol{y}^{\mathrm{I}}) \quad \text{with} \quad A^{\mathrm{m}} = h[\theta_h](\boldsymbol{C}, \boldsymbol{u}). \tag{3.39}$$

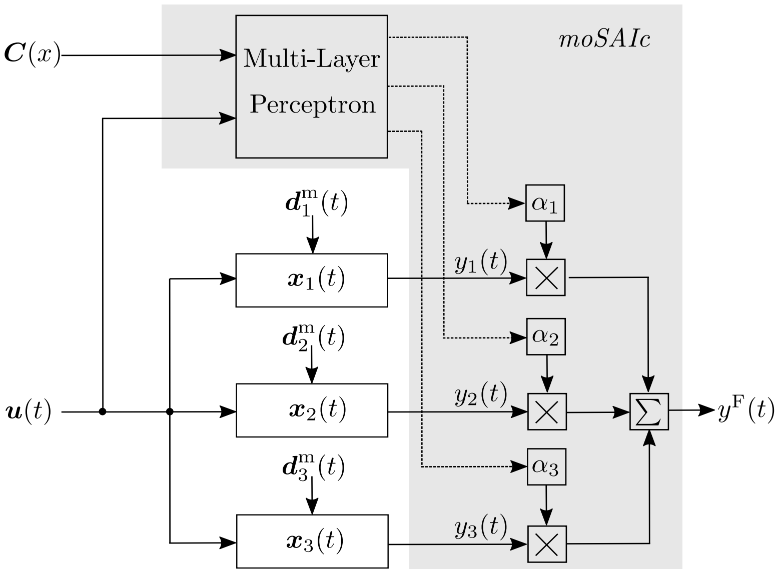


Figure 3.14 *moSAIc* as part of a block diagram.

Overall, *moSAIc* differentiates itself in comparison with similar supervised learning methods like a MLP with linear output layers or ensemble methods by combining the advantages of them. A classical MLP with a linear output layer or a parallel linear layer determines a weight change directly according to the label which is assumed to be ground truth during the training process (see Eq. 3.2.2.2). In summary, the label is directly presented to the output layer of the MLP. Applying *moSAIc*, the label is never presented directly to the output layer of the MLP, but the individual weights $\boldsymbol{\alpha}_i(\boldsymbol{y}_k^{\mathrm{F}})$ are used as output labels for the MLP. This results in an indirect learning approach for the MLP (see Eq. 3.36). Compared to ensemble methods like introduced by [HYW10], [BM15] or [Hon+19], *moSAIc* combines the fusion output $\boldsymbol{y}^{\mathrm{F}}$ considering both the system input $\boldsymbol{u}$ as well as the individual predictions of the base models y_1, y_2 and y_3. Ensemble methods in this case are meant as employing multiple methods simultaneously and aggregate different predictions, also known as gating when using a neural network for the aggregation. Doing so, these methods

either lack to integrate helpful information like the system input $\boldsymbol{u}$ due to its focus on determining weights of a linear regression model just based on base model predictions using an optimization algorithm [BM15; HYW10] or create additional complexity by applying a non-linear aggregation using a neural network [Hon+19]. Taking into account, that the base models are similar to the target model, bringing in additional complexity by applying a neural network as aggregation method does not perform better than an average filter (see [Hil+19]).
Finally, the *moSAIc* function can be written as a fusion operator:

$$\circledcirc = \min_{\theta_h} \left\{ \left\| m^s[\theta_h](\boldsymbol{y}^{\mathrm{I}}) - \boldsymbol{y}^{\mathrm{H}} \right\|_2 \right\} . \tag{3.40}$$

3.2.2.4 Co-Kriging

Originally, Co-Kriging is a concept which was developed in the domain of geo-statistics and is used for weather predictions (see [ALH97]). Mathematically, kriging uses GPR to inter- and extrapolate information which is not available based on observations of an available information source. The approach of Co-Kriging utilizes information from different sources, meaning the information provided by multiple models or observation sources like sensors can be combined. Co-Kriging is especially useful to solve fusion problems which require methods of the general fusion function set Γ_a^a. The methodology of Co-Kriging is based on the concept of GPR. According to [RW06], a Gaussian process (GP) is "a collection of random variables, any finite number of which have a joint gaussian distribution". The following introduction of Gaussian Processes is also based on the work of [RW06]. Generally, a GP can be described by an input domain Ξ, a mean function $\mu(\mathrm{x})$ and a positive semi-definite symmetric covariance function $k^G(\mathrm{x}, \mathrm{x}')$:

$$f(\mathrm{x}) \sim \mathcal{GP}(\mu, k^G) \tag{3.41}$$

$$\mathbb{E}[f] := \mu : \Xi \to \mathbb{R} \tag{3.42}$$

$$\mathbb{E}[(f(\mathrm{x}) - \mu(\mathrm{x}))(f(\mathrm{x}') - \mu(\mathrm{x}'))] := k^G : \Xi \times \Xi \to \mathbb{R} \tag{3.43}$$

In this work, the covariance function k^G also known as kernel function is a squared exponential kernel with automatic relevance determination and can be described by

$$k^G(\mathrm{x}, \mathrm{x}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{\mathrm{x}_d - \mathrm{x}'_d}{l_d}\right)^2\right), \tag{3.44}$$

where $D = \dim(\mathbf{X})$ and the hyper-parameters of the squared exponential covariance function are σ, the amplitude, as well as l_d, the length scale. Let $\mathbf{X} \subseteq \Xi$ be a set consisting of a finite number of samples and let $\boldsymbol{y}$ be the corresponding function values $f(\mathbf{X})$, then

$$p(\boldsymbol{y}|\mathbf{X}) = \mathcal{N}(\mu(\mathbf{X}), k^G(\mathbf{X}, \mathbf{X})) \tag{3.45}$$

applies. Adding now observations to the method, consider a finite set of N tuples $\mathbf{I} = (\mathbf{X}, \boldsymbol{y})$ as observational data and consider $\mathbf{X}_*$ as unknown points of interest. Then the joint distribution between the given and unknown data points can be described by

$$p(\boldsymbol{y}, \boldsymbol{y}_*) = \mathcal{N}\left(\begin{bmatrix}\mu(\mathbf{X})\\ \mu(\mathbf{X}_*)\end{bmatrix}, \begin{bmatrix}k^G(\mathbf{X}, \mathbf{X}) & k^G(\mathbf{X}, \mathbf{X}_*)\\ k^G(\mathbf{X}_*, \mathbf{X}) & k^G(\mathbf{X}_*, \mathbf{X}_*)\end{bmatrix}\right). \tag{3.46}$$

Having specified the joint distribution, the posterior of the Gaussian Process can be computed by

$$p(\boldsymbol{y}_*|\mathbf{X}_*, \mathbf{I}) = \mathcal{N}(\mu_{\mathrm{x}|\mathbf{I}}(\mathbf{X}_*), k^G_{\mathrm{x}|\mathbf{I}}(\mathbf{X}_*, \mathbf{X}_*)), \tag{3.47}$$

$$\mu_{\mathrm{x}|\mathbf{I}}(\mathrm{x}) = \mu(\mathrm{x}) + k^G(\mathrm{x}, \mathbf{X})(k^G(\mathbf{X}, \mathbf{X})^{-1}(\boldsymbol{y} - \mu(\mathbf{X})), \tag{3.48}$$

$$k^G_{\mathrm{x}|\mathbf{I}} = k^G(\mathrm{x}, \mathrm{x}') - k^G(\mathrm{x}, \mathbf{X}')(k^G(\mathbf{X}, \mathbf{X})^{-1}k^G(\mathbf{X}, \mathrm{x}'). \tag{3.49}$$

In the following, the abbreviations $K := k^G(\mathbf{X}, \mathbf{X})$, $K_* := k^G(\mathbf{X}, \mathbf{X}_*) = k^G(\mathbf{X}_*, \mathbf{X})$ and $K_{**} := k^G(\mathbf{X}_*, \mathbf{X}_*)$ will be used. During the training process of a Gaussian Process, the hyper-parameters $\boldsymbol{\theta}_{\mathrm{G}} := [\sigma, l]$ are going to be optimized. The typical cost function for a Gaussian Process is the negative marginal log likelihood function

$$\mathcal{C}(\boldsymbol{\theta}_{\mathrm{G}}) = -\frac{1}{2}\,\boldsymbol{y}^{\mathrm{T}}\,K^{-1}\,\boldsymbol{y} + \frac{1}{2}\log|K|\; + \frac{N}{2}\log 2\pi. \tag{3.50}$$

[RK16] shows, that the partial derivative of $\frac{\mathcal{C}(\theta_{\mathrm{G}})}{\partial K}$ is given by

$$\frac{\partial \mathcal{C}(\boldsymbol{\theta}_{\mathrm{G}})}{\partial K} = -\frac{1}{2}\,K^{-1}\,\boldsymbol{y}\,\boldsymbol{y}^{\mathrm{T}}K^{-1} + \frac{1}{2}\,K^{-1}, \tag{3.51}$$

which leads to the required partial derivative for the gradient descent optimization

$$\frac{\partial \mathcal{C}(\boldsymbol{\theta}_{\mathrm{G}})}{\partial \boldsymbol{\theta}_{\mathrm{G}}} = -\frac{\partial \mathcal{C}(\boldsymbol{\theta}_{\mathrm{G}})}{\partial K}\frac{\partial K}{\partial \boldsymbol{\theta}_{\mathrm{G}}}. \tag{3.52}$$

The basics of Gaussian Process regression have now been introduced and the different types of Co-Kriging, which are relevant for this work, are going to be introduced. Hereby, let the tuple $\{\boldsymbol{u}_1, \boldsymbol{y}_1\}$ be information provided by the first information source and the tuple $\{\boldsymbol{u}_2, \boldsymbol{y}_2\}$ be information provided by the second information source. For simplification purposes, the Co-Kriging types are introduced for two information sources, but the methods can be extended for more information sources analogously. In context of the purpose of multi-fidelity modeling, the two information sources can also be described as sources of different fidelities, for example an LFM and HFM. Let's assume, the tuples $\{\boldsymbol{u}_2, \boldsymbol{y}_2\}$ have a higher fidelity than the tuples $\{\boldsymbol{u}_1, \boldsymbol{y}_1\}$, meaning the information which is classified as higher fidelity is more valuable than the information which is classified with a lower fidelity. Let's further assume, the tuples are stacked to one joint input domain $\mathbf{U} := [\boldsymbol{u}_1 \ \boldsymbol{u}_2]^{\mathrm{T}}$ and one joint output domain $\mathbf{Y} := [\boldsymbol{y}_1 \ \boldsymbol{y}_2]^{\mathrm{T}}$, the low-fidelity function is given by $f_1(u_1) = y_1$ and the high-fidelity function is given by $f_2(u_2) = y_2$.

Autoregressive Co-Kriging (ARCK)
Based on the work of [Ken00], the autoregressive model for the Co-Kriging approach can be described by

$$f_2(u) = \rho f_1(u) + \delta_2(u), \tag{3.53}$$
$$\delta_2(u) \sim \mathcal{GP}(0, k_2^G(u, u')), \tag{3.54}$$
$$f_1(u) \sim \mathcal{GP}(0, k_1^G(u, u')), \tag{3.55}$$

which results in a new description, that includes two information sources

$$\mathbf{Y} = \begin{bmatrix} f_1(\boldsymbol{u}) \\ f_2(\boldsymbol{u}) \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_1^G(\boldsymbol{u}, \boldsymbol{u}') & \rho k_1^G(\boldsymbol{u}, \boldsymbol{u}') \\ \rho k_1^G(\boldsymbol{u}, \boldsymbol{u}') & \rho^2 k_1^G(\boldsymbol{u}, \boldsymbol{u}') + k_2^G(\boldsymbol{u}, \boldsymbol{u}') \end{bmatrix} \right). \tag{3.56}$$

The computation of the posterior and the training using the negative marginal log likelihood function are done similarly than before, the definition of K and K_* just change to

$$K = \begin{bmatrix} k_1^G(\boldsymbol{u}_1, \boldsymbol{u}_1) & \rho\, k_1^G(\boldsymbol{u}_1, \boldsymbol{u}_2) \\ \rho\, k_1^G(\boldsymbol{u}_2, \boldsymbol{u}_1) & \rho^2 k_1^G(\boldsymbol{u}_2, \boldsymbol{u}_2) + k_2^G(\boldsymbol{u}_2, \boldsymbol{u}_2) \end{bmatrix}, \tag{3.57}$$

$$K_* = \begin{bmatrix} \rho k_1^G(\boldsymbol{u}_*, \boldsymbol{u}_1) & \rho^2 k_1^G(\boldsymbol{u}_*, \boldsymbol{u}_2) + k_2^G(\boldsymbol{u}_*, \boldsymbol{u}_2) \end{bmatrix}, \tag{3.58}$$

and the hyper-parameters change to $\boldsymbol{\theta}_{\mathrm{G}} = [\rho, \sigma_1, l_1, \sigma_2, l_2]$.

Simple Co-Kriging (SCK)
Simple Co-Kriging is a generalization of ARCK, or saying it the other way around, ARCK is a special case of SCK. Every element of the co-variance matrix is generalized with the correlation parameters α_1, α_2, β_1 and β_2. This changes the covariance matrices to

$$K^i_{j,k} = k_i^G(\boldsymbol{u}_j, \boldsymbol{u}_k), \tag{3.59}$$

$$K = \begin{bmatrix} \alpha_1^2 K^1_{1,1} + \alpha_2^2 K^2_{1,1} & \alpha_1\beta_1 K^1_{1,2} + \alpha_2\beta_2 K^2_{1,2} \\ \alpha_1\beta_1 K^1_{2,1} + \alpha_2\beta_2 K^2_{2,1} & \beta_1^2 K^1_{2,2} + \beta_2^2 K^2_{2,2} \end{bmatrix}, \tag{3.60}$$

$$K_* = \begin{bmatrix} \alpha_1\beta_1 K^1_{*,1} + \alpha_2\beta_2 K^2_{*,1} & \beta_1^2 K^1_{*,2} + \beta_2^2 K^2_{*,2} \end{bmatrix}, \tag{3.61}$$

and the hyper-parameters to $\boldsymbol{\theta}_{\mathrm{G}} = [\alpha_1, \alpha_2, \beta_1, \beta_2, \sigma_1, l_1, \sigma_2, l_2]$. Thus, in comparison to ARCK, SCK is able to morph to even more non-linear function shapes at the cost of extended training time due to more hyper-parameters.

Deep multi-fidelity Gaussian processes (DMGP)
The concept of DMGP has been introduced by [RK16] and differs from ARCK and SCK in the way, that the input domain is transformed into several lateral domains before being used as input values for the covariance functions. The input transformation is done using an MLP and extends the methodology to be capable of approximating discontinuous function. The ARCK approach is used as Co-Kriging backbone. Like introduced in 3.2.2.2, the MLP function h can be specified by its parameter vector θ_h,

which leads to the definitions of the covariance matrices

$$K = \begin{bmatrix} k_1^G(\mathrm{h}_1, \mathrm{h}_1) & \rho k_1^G(\mathrm{h}_1, \mathrm{h}_2) \\ \rho k_1^G(\mathrm{h}_2, \mathrm{h}_1) & \rho^2 k_1^G(\mathrm{h}_2, \mathrm{h}_2) + k_2^G(\mathrm{h}_2, \mathrm{h}_2) \end{bmatrix}, \tag{3.62}$$

$$K_* = \begin{bmatrix} \rho k_1^G(\mathrm{h}_*, \mathrm{h}_1) & \rho^2 k_1^G(\mathrm{h}_*, \mathrm{h}_2) + k_2^G(\mathrm{h}_*, \mathrm{h}_2) \end{bmatrix}, \tag{3.63}$$

with $\mathrm{h}_i = h(\boldsymbol{u}_i)$ and $\boldsymbol{\theta}_\mathrm{G} = [\rho, \sigma_1, l_1, \sigma_2, l_2, \theta_h]$ as definition of the hyperparameters. Comparing DMGP to ARCK and SCK, it is able to also approximate discontinuous functions at the cost of the most expensive training process. The training of DMGP is done in a joint training process, a combination of gradient descent and stochastic gradient descent, and can be deeper investigated in [RK16].

Interpreting the methodology for multi-fidelity modeling, the input and output domain tuples consist of $\boldsymbol{u}^\mathrm{L}, \boldsymbol{y}^\mathrm{L}$ and $\boldsymbol{u}^\mathrm{M}, \boldsymbol{y}^\mathrm{M}$ for the condition monitoring case and of $\boldsymbol{u}^\mathrm{L}, \boldsymbol{y}^\mathrm{L}$ and $\boldsymbol{u}^\mathrm{H}, \boldsymbol{y}^\mathrm{H}$ for the output prediction case. The Co-Kriging function itself is named g^c in the following and evaluates the predictive distribution

$$p(\boldsymbol{y}_*^\mathrm{M} | \boldsymbol{u}_*, \boldsymbol{u}^\mathrm{L}, \boldsymbol{y}^\mathrm{L}, \boldsymbol{u}^\mathrm{M}, \boldsymbol{y}^\mathrm{M}) \tag{3.64}$$

for the condition monitoring case and

$$p(\boldsymbol{y}_*^\mathrm{H} | \boldsymbol{u}_*, \boldsymbol{u}^\mathrm{L}, \boldsymbol{y}^\mathrm{L}, \boldsymbol{u}^\mathrm{H}, \boldsymbol{y}^\mathrm{H}) \tag{3.65}$$

for the output prediction use case. Overall, Co-Kriging can be written as fusion operator:

$$\circledcirc = \min_{\theta_G} \left\{ \left\| g^c[\boldsymbol{\theta}_\mathrm{G}](\boldsymbol{y}^\mathrm{I}) - \boldsymbol{y}^\mathrm{I} \right\|_2 \right\}. \tag{3.66}$$

4 Purpose-driven Model Design

Operation-parallel simulation can have different sub-purposes, which are introduced in this chapter. The different sub-purposes are introduced separately by defining the prerequisites, an individual model design work flow and an exploration of the previously introduced methods. These methods are evaluated by applying them to specifically developed benchmark systems for each sub-purpose.

4.1 Model Design for Dimensionality Increase

In context of the digital performance twin, the model design for dimensionality increase has its purpose in creating so-called *Soft-Sensors*. This kind of sensors is especially used in process industries to model dynamic system behavior based on observations of processes which are complex to model. [KGS09] as well as [SAM16] introduce commonly used methods for data-driven modeling of soft-sensors in process industries like Principle component analysis (PCA), Partial least squares (PLS), ANNs or SVR. Additionally, [KGG11] extend the review by integrating adaption mechanisms for data-driven soft-sensors like Fast Moving Window PCA, Recursive PCA, Recursive PLS or ensemble-based methods based on incremental local learning. Operation-parallel soft-sensors created by multi-fidelity modeling can be used for both condition monitoring and output prediction. They are especially useful in cases where the real system is restricted to physical boundaries like geometric restrictions. For example, they are useful for systems which do not allow to acquire observations from certain unreachable spots which actually require monitoring. The model design for dimensionality increase requires the prerequisites listed in Table 4.1. Soft-sensoring using multi-fidelity modeling is applicable in cases where

- an accurate computational intensive high-dimensional model,

- a fast and therefore operation-parallel applicable low-dimensional model which includes the system dynamics,
- low-dimensional observations

are available. This might be often the case for systems which are modeled using FEM or CFD in the design phase to lay out a production process which requires condition monitoring during the operation phase.

Table 4.1 Prerequisites for dimensionality increase using MFM.

Source	Dimension	Accuracy	Rel. Acquisition Time
Observation	- -	+ +	+ +
LFM	-/o	-/o	+ +
HFM	+ +	+ +	- -

- -: very bad/small | -: bad/small | o: sufficient | +: good/big | ++: very good/big

4.1.1 Work Flow

Consulting Figure 3.2, the first step of the multi-fidelity modeling work flow is the reusage of the design model and the interest region definition. Interpreting this for the model design for dimensionality increase, first of all the HFM needs to be given as an executable simulation model in the simulation environment which was used during the design phase. Just having a CAD model at hand is insufficient in this case, because the model needs to be exploitable for different boundary and operating conditions. The definition of the interest region also includes the configuration of the simulation tool's granularity, for example the mesh size and discrete time step size. The model designer needs to define the dimensions which should be used as soft-sensors, which could be geometric positions on a system component where no sensor can be placed or complex physical phenomena which cannot be measured (typical use case in process industries) for example. The configuration of the HFM exploitation influences significantly the development time for a multi-fidelity model which is going to be used for dimensionality increase, because a small mesh size in combination with a small time step size translates quickly into computational intensive simulations.

The second step according to Figure 3.2 is the selection or development of the LFM. In the context of soft-sensoring and dimensionality increase, LFMs are typically low-dimensional model-based or data-driven simulation models. They are often developed by using simpler physics or dimensional downsizing, meaning the neglect of minor relevant physical influences or the accumulation of sub-systems to one joint system. Hereby, it is important to notice, that such an LFM as well as corresponding observations need to be developed or rather selected before the training of the MFM starts, because the simulation results of the LFM are an essential part of the training process. The same applies for the case that a data-driven model based on observations shall be trained and used as LFM.

The three main elements, the HFM, LFM and observations are afterwards used for the MFM composition. To generate training data and test data as well as to acquire necessary observations, a Design of experiments (DoE) has to be developed. The general work flow to develop a multi-fidelity model for dimensionality increase can be defined as:

1. Set up a DoE including every dimension of the system input $\boldsymbol{u}$ and every initialization of the system state $\boldsymbol{x}$. Take the conditions into account in which the MFM will be applied and make sure that all of them are included in the DoE.

2. Execute the DoE by applying it to the HFM and the LFM. Operate the system under consideration like defined in the DoE, for example in a laboratory environment, and acquire the necessary measurements.

3. Make sure the simulation results and the observations are synchronized in time and registered in space.

4. Combine the LFM output $\boldsymbol{y}^{\mathrm{L}}$ and observations $\boldsymbol{y}^{\mathrm{M}}$ into the fusion input $\boldsymbol{y}^{\mathrm{I}}$ and split the data set into training, validation and test data. Do these steps analogously with the HFM simulation results $\boldsymbol{y}^{\mathrm{H}}$ to create the desired fusion output $\boldsymbol{y}^{\mathrm{F}}$.

5. Specify the selected fusion function set F_{f} (typically $a < b$) and execute the fusion operator training algorithm to determine the

most suitable method m using the training and validation data set.

6. Assemble the MFM by determining the most suitable fusion operator according to Figure 3.6 for condition monitoring or Figure 3.7 for output prediction. Use the test data set to exploit the newly designed MFM and check if the prediction accuracy fits the need.

The last and fourth step of the MFM work flow is the code generation to execute the developed MFMopS on the target hardware. In the case of model design for dimensionality increase, a typical runtime environment for soft-sensors is the shop floor of a production. Depending on the real-time requirements, Table 4.2 visualizes a possible target hardware.

Table 4.2 Runtime environments for soft-sensor applications.

Real Time Req.[a]	Execution Env.	Runtime
Hard	PLC	Embedded Runtime
Soft	IPC, DCS	RT Linux, Windows Embedded
No Req.	Cloud Service	IIoT Operating System

[a] Liu09.

4.1.2 Benchmark System

In the following, the introduced work flow will be explained using a benchmark system specially developed for this purpose. Similar to [Küh13], different dynamic systems are utilized to show the capabilities of the modeling work flow and fusion methodology. Hereby, the dynamic systems out of which the benchmark system is designed can be characterized as Linear-time-invariant systems (LTIs) and are visualized in Figure 4.1. The system response y of the respective system is considered to be observed in terms of the resulting fusion problem and shall be predicted. The first system is a standard first-order system which can be defined by its parameters C_1, C_2 and C_3 and will be abbreviated $A(C_1, C_2, C_3)$ from now on. It is part of the benchmark set because typically first-order systems can be approximated sufficiently by exponential functions and therefore be part of an LFM. The second included system is a second-order system defined by its parameters C_1, C_2, C_3 and C_4 and from now on written as $B(C_1, C_2, C_3, C_4)$. Many physical phenomena can be described by second order systems, but often their time-continuous model requires intensive computational resources, especially when multiple second-order systems are coupled. The feedback loop is part of the benchmark set because the system input u gets amplified and smoothed and is characterized by the parameters C_1, C_2 and C_3. It will be abbreviated by $C(C_1, C_2, C_3)$ in the following. Analogously, the feed through system is utilized because the system output y is influenced by a direct and time-delayed impact of the system input u. The abbreviation for the feed through system is $D(C_1, C_2, C_3, C_4)$. The last considered system is a second-order system with a time delay, which means that the system output y will react on a system input u with a time delay of T_d. This system type will be abbreviated by $E(C_1, C_2, C_3, C_4)$ with $T_\mathrm{d} = 1\,\mathrm{s}$ in the following.

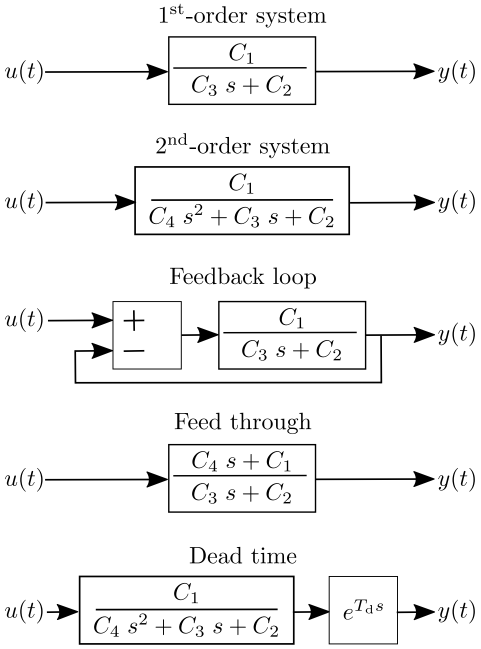

1st-order system
$u(t)$
$\frac{C_1}{C_3\ s + C_2}$
$y(t)$
2nd-order system
$u(t)$
$\frac{C_1}{C_4\ s^2 + C_3\ s + C_2}$
$y(t)$
Feedback loop
$u(t)$
+
−
$\frac{C_1}{C_3\ s + C_2}$
$y(t)$
Feed through
$u(t)$
$\frac{C_4\ s + C_1}{C_3\ s + C_2}$
$y(t)$
Dead time
$u(t)$
$\frac{C_1}{C_4\ s^2 + C_3\ s + C_2}$
$e^{T_\mathrm{d} s}$
$y(t)$

Figure 4.1 Utilized dynamic systems.

For the use case of dimensionality increase, the HFM is defined as a parallel ensemble of these different dynamic systems, where each system output y_i of dynamic system i represents one dimension of the HFM. Overall, the HFM consists of 25 dimensions[1] which react on the single system input u. The mean task execution time for the HFM based on five measurements for one simulation run with a sample rate of $0.01\,\mathrm{s}$ and overall simulation time of $15\,\mathrm{s}$ is $114.4\,\mathrm{ms}$. Table 4.3 shows the definition of the HFM.

Table 4.3 High-fidelity benchmark model for dimensionality increase.

Type	**System**	**Dim.**	**Type**	**System**	**Dim.**
1st-order	$A(10, 3, 5)$	1	2nd-order	$B(1, 8, 2, 9)$	6
	$A(6, 1, 3)$	2		$B(7, 2, 7, 4)$	7
	$A(3, 9, 8)$	3		$B(9, 9, 5, 7)$	8
	$A(8, 4, 10)$	4		$B(2, 6, 8, 2)$	9
	$A(4, 6, 1)$	5		$B(5, 3, 10, 6)$	10
Feedback loop	$C(3, 4, 6)$	11	Feed through	$D(2, 1, 6, 4)$	16
	$C(7, 9, 1)$	12		$D(7, 8, 7, 2)$	17
	$C(1, 2, 8)$	13		$D(1, 3, 5, 8)$	18
	$C(5, 10, 3)$	14		$D(9, 7, 2, 10)$	19
	$C(9, 6, 10)$	15		$D(3, 5, 9, 6)$	20
Dead time	$E(4, 5, 2, 8)$	21			
	$E(8, 2, 6, 3)$	22			
	$E(10, 9, 9, 1)$	23			
	$E(1, 7, 5, 6)$	24			
	$E(3, 4, 7, 10)$	25			

The LFM is designed by assuming to be able to simulate the dimensions 3 and 7 with a lower accuracy, resulting in an LFM definition which is shown in Table 4.4, having a mean task execution time of $68.8\,\mathrm{ms}$. Observations are generated by applying normally distributed uncorrelated random noise to the high-fidelity dimensions 14 and 20, emulating the fact that sensors could be placed on the real system to acquire observations which

[1] Each system response is considered to be an individual output dimension for the joint benchmark system.

are acting as independent dimensions. The unit step responses of the HFM and LFM as well as the acquired observations are shown in Figures 4.2, 4.3, and 4.4.

Table 4.4 Low-fidelity benchmark model for dimensionality increase.

Type	System	Dimension
1st-order	$A(2.8, 9.1, 8.1)$	Approximation of HF3
2nd-order	$B(7.3, 2.2, 6.9, 3.9)$	Approximation of HF7

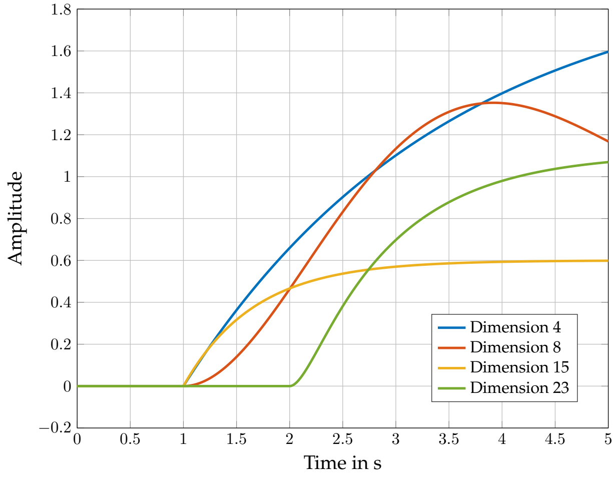


Figure 4.2 Exemplary HFM output dimensions of benchmark system.

Having introduced the different models, the work flow for the MFM composition can start. For the DoE, a setup has to be chosen to generate training, validation and test data. Therefore, three different input signals, the step function $u_{\text{Step}}(t)$, the ramp function $u_{\text{Ramp}}(t)$ and the sinusoidal function $u_{\text{Sin}}(t)$ are used:

$$u_{\text{Step}}(t) = \begin{cases} 0 & t < t_{\text{S}} \\ a & t \geq t_{\text{S}} \end{cases}, \tag{4.1}$$

$$u_{\text{Ramp}}(t) = a \cdot t, \tag{4.2}$$

$$u_{\text{Sin}}(t) = a \cdot \sin(b\,t). \tag{4.3}$$

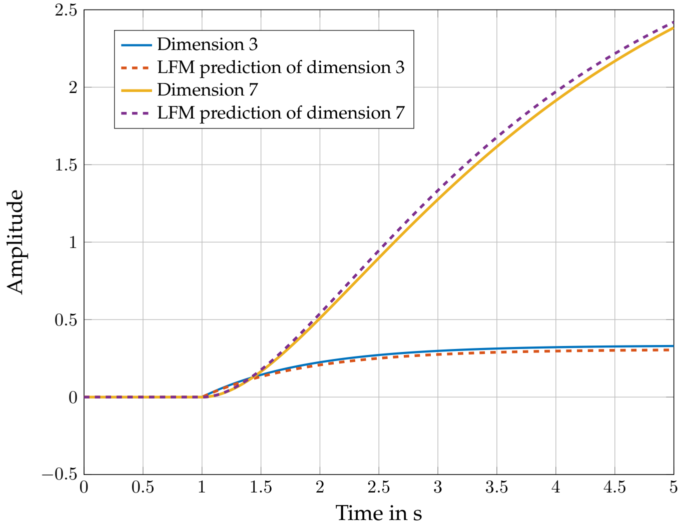


Figure 4.3 Exemplary LFM output dimensions of benchmark system.

The full-factorial DoE which is used afterwards to create the training and validation data is shown in Table 4.5. It results in 15 different input signals, which are processed with a simulation duration $t_{\text{sim}} = 15\,\text{s}$ and are sampled with a sample rate of $t_{\text{sr}} = 0.1\,\text{s}$. Aggregating all simulations together, the training and validation data set size sums up to 2265 samples of the LFM system output $\boldsymbol{y}^{\text{L}}$, HFM system output $\boldsymbol{y}^{\text{H}}$ and observations $\boldsymbol{y}^{\text{M}}$.

After the first three steps of the multi-fidelity modeling work flow are done, the fusion operator can be trained using the test and validation

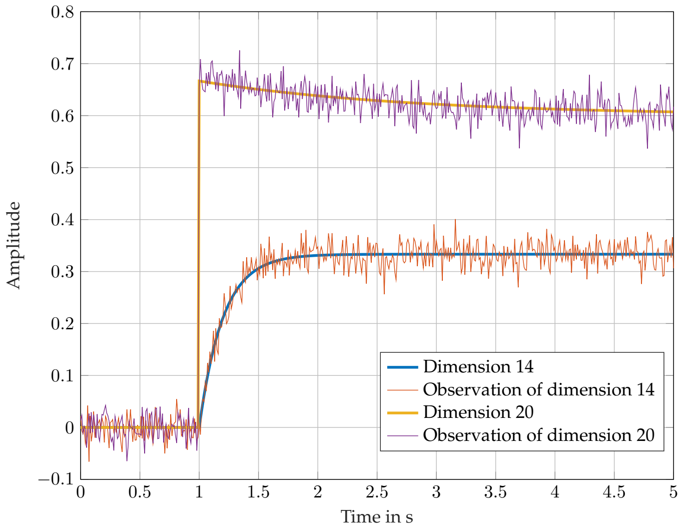


Figure 4.4 Example for acquired observations of benchmark system.

set. Therefore, the LFM output $\boldsymbol{y}^{\mathrm{L}}$ and the observations $\boldsymbol{y}^{\mathrm{M}}$ are stacked to a joint fusion input $\boldsymbol{y}_k^{\mathrm{I}} = \left[y_{1,k}^{\mathrm{L}}\ y_{2,k}^{\mathrm{L}}\ y_{1,k}^{\mathrm{M}}\ y_{2,k}^{\mathrm{M}} \right]^{\mathrm{T}}$, the HFM output is assumed to be the desired fusion output $\boldsymbol{y}_k^{\mathrm{F}} = \boldsymbol{y}_k^{\mathrm{H}}$. The investigated fusion methods which are applicable for the use case of dimensionality increase are the QMI, the MLP and moSAIc. Co-Kriging is not really applicable for the modeling purpose of dimensionality increase because it has its strength in problems which require a Γ_a^a mapping. Therefore, the fusion

Table 4.5 DoE of benchmark model for dimensionality increase.

Function	Parameter Range	Step Size
$u_{\text{Step}}(t)$	$0.5 \leq a \leq 2.5$	0.5
	$t_{\text{S}} = 1.0$	
$u_{\text{Ramp}}(t)$	$0.5 \leq a \leq 2.5$	0.5
$u_{\text{Sin}}(t)$	$0.5 \leq a, b \leq 2.5$	0.5

function set $F_\mathrm{f} = \{q[\alpha], h[\theta_h], m^\mathrm{s}[\theta_h]\}$ will be used for the fusion operator training. All specified methods need to be part of the general fusion function set Γ_{25}^4. Since many different topologies and training configurations are suitable to design and train an MLP, three pre-selected models $(h_1[\theta_{h,1}], h_2[\theta_{h,2}], h_3[\theta_{h,3}])$ are compared from now on. The models and training parameters are described in Table 4.6 and 4.7.

Table 4.6 Standard MLP models 1 and 2.

Parameter	**MLP 1**	**MLP 2**
Hidden Layers	None	2
Neurons per Layer	$[\mathbb{R}^{\boldsymbol{y}^\mathrm{I}}, \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}]$	$[\mathbb{R}^{\boldsymbol{y}^\mathrm{I}}, 3 \cdot \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}, 2 \cdot \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}, \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}]$
Activation Function	Identity	ReLu
Epochs	100	500
Learning rate	$1 \cdot 10^{-3}$	$5 \cdot 10^{-3}$

Table 4.7 Standard MLP model 3.

Parameter	**MLP 3**
Hidden Layers	3
Neurons per Layer	$[\mathbb{R}^{\boldsymbol{y}^\mathrm{I}}, 3 \cdot \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}, 3 \cdot \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}, 3 \cdot \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}, \mathbb{R}^{\boldsymbol{y}^\mathrm{F}}]$
Activation Function	$\tanh$
Epochs	1000
Learning rate	$1 \cdot 10^{-4}$

The applied moSAIc architecture is shown in Figure 4.5. The static descriptive features $\mathbf{C}(x)$ are the time constants C_1, C_2, C_3 and C_4 which are shown in Table 4.3. A non-existing time constant C_4 is replaced with a 0 for the training set.

Finally, the fusion operator training is executed with the three different methods which are part of F_f including also the three different MLP topologies. Table 4.8 shows the ED for the training and test set. Hereby, the test set consists of 10 simulations with an individual simulation duration of $t_\mathrm{s} = 30\,\mathrm{s}$ and a random system input $0.0 \leq u(t) \leq 2.5$ which is changing its

Table 4.8 Euclidean distance (ED) on training and test set.

Model	ED Training	ED Test
MLP 1	2576.90	497.22
MLP 2	309.67	103.92
MLP 3	2617.25	275.35
QMI	308.66	85.37
moSAIc	1074.96	246.11

value every 5 seconds. Therefore, only the interpolation case is addressed using this benchmark system, the extrapolation performance will be discussed in the industrial applications section 5.1. It can be seen, that the QMI method performs best on the training set, because its ED is the lowest. According to equation 3.2.1, the QMI is selected to be the fusion operator for this fusion problem. The performance of the different methods on the test data set underlines the fusion operator selection, because QMI also performs the best on the test set. In Figure 4.6, the Root-mean-square error (RMSE) distribution (test set) of the three best models are purified for each output dimension. Concluding from this figure, some output dimensions are approximated pretty well and others are approximated less well. The system outputs are predicted based on the system input $u(t)$ shown in Figure 4.7. For both, a good fit and a bad fit example are shown in Figure 4.8 and Figure 4.9. The selected output dimensions of $\boldsymbol{y}^{\mathrm{F}}$ are dimension 9, because all three models perform good on it, and dimension 21, because its challenging for all three models.

Summarizing the benchmark system evaluation for the model design for dimensionality increase, it was shown, that soft-sensoring applications can be realized using the multi-fidelity modeling approach and work flow. The capability of estimating dimensions of an HFM based on LFM simulations and observations can be utilized. An important remark is the capability to capture the dynamics of the output dimensions by the LFM and the observations. For example, this means it is respectively hard to estimate second-order behavior just based on first-order observations or simulations. The soft-sensoring is done by scaling the predictions of the LFM and observations to the wanted HFM dimension of a similar type on the one hand and learning dependencies to other dimensions on the other hand. Hence, the simplest method delivers the best results considering this benchmark system, because QMI is able to weigh the interactions between each low- and high-fidelity dimension without being forced to stay able to generalize like MLPs. Overall, the multi-fidelity modeling work flow can be used for soft-sensoring applications, but engineering knowledge is required to select the important LFM dimensions and the essential observations to design a robust and accurate soft-sensor.

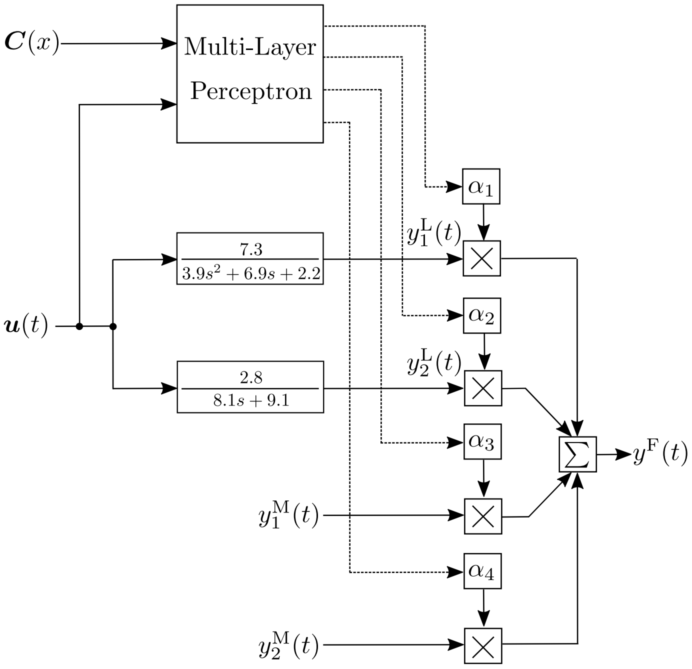


Figure 4.5 moSAIc architecture for dimensionality increase benchmark system.

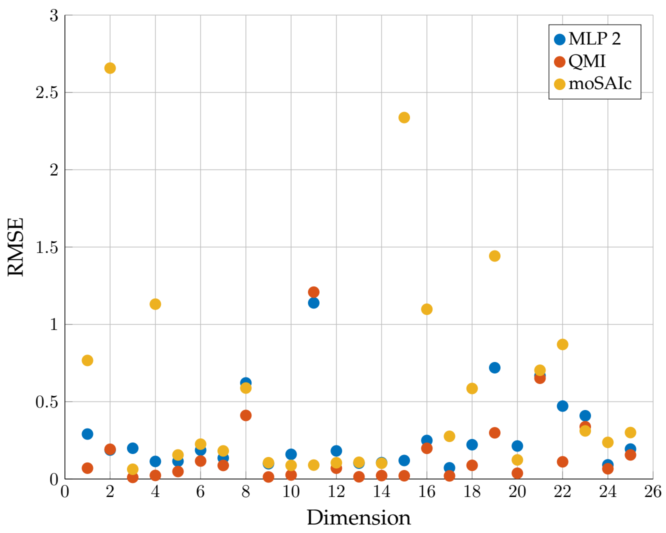


Figure 4.6 RMSE Distribution of benchmark system evaluation.

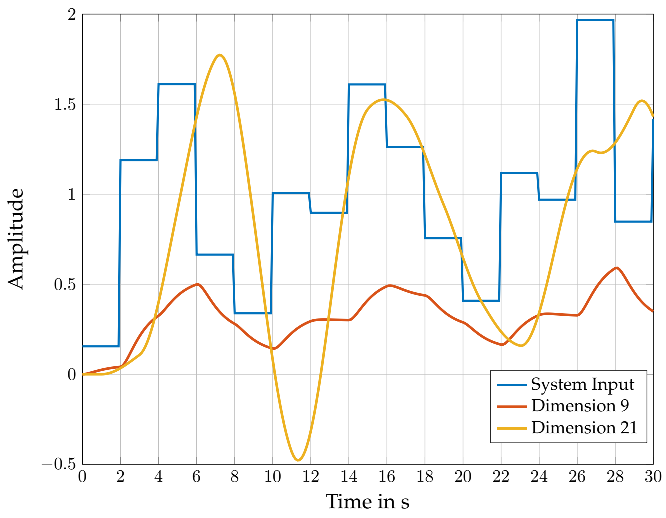


Figure 4.7 System input for benchmark system evaulation.

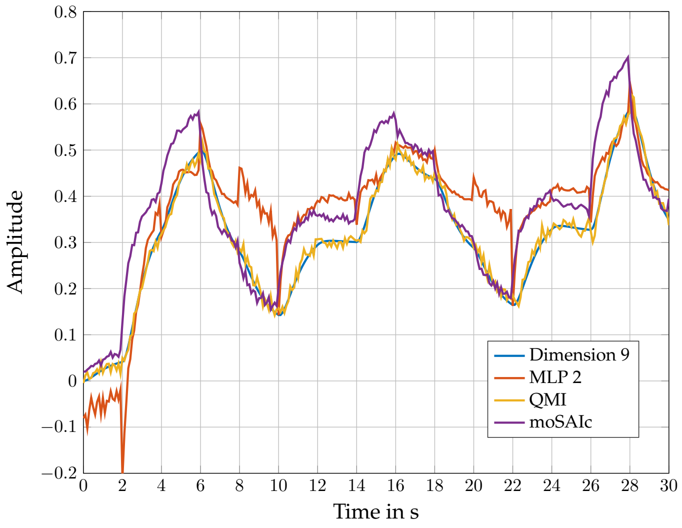


Figure 4.8 Fusion result of dimension 9 (good fit).

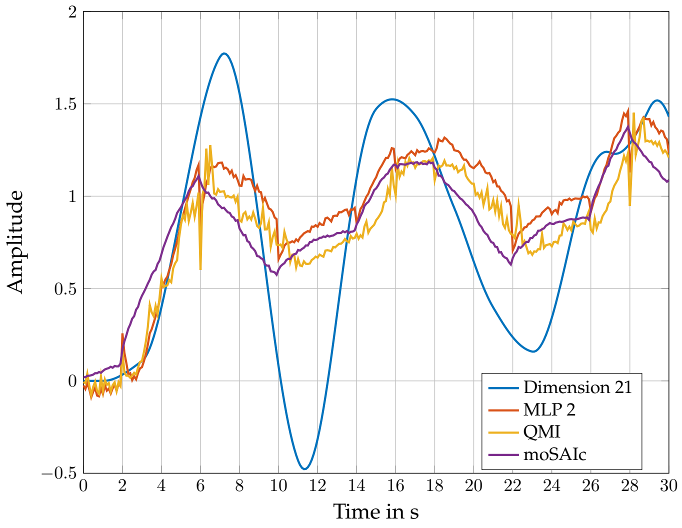


Figure 4.9 Fusion result of dimension 21 (bad fit).

4.2 Model Design for Accuracy Enhancement

In contrast to the model design for dimensionality increase, the model design for accuracy enhancement has its purpose in improving accuracy of LFMs by combining them with evaluations of HFMs. Thus, the main difference lies in the number of required observations and HFM evaluations to increase the accuracy of the LFM. Just a few, in this case computationally intensive, HFM evaluations or physically expensive observations are combined with many LFM evaluations to create a fast MFM with an accuracy which is close to the HFM evaluations or observations. Hereby, both the LFM as well as the HFM or observations are able to describe the same parameters of the interest region which are just differentiated by the respective model output accuracy against the ground truth. This kind of setup is typically requested for modeling tasks where computationally expensive model evaluations are somehow required in the operation phase of the system. The model design for accuracy enhancement focuses on dynamic systems in stationary cases, because typically the transient behavior of models used for this purpose is either to cost-intensive in observing it or to cost-intensive in simulating it. Conceptually, transient behavior could also be considered using this methodology. In conclusion, feature-based mapping algorithms without modeling the time-continuity will be used.

Another purpose of the model design for accuracy enhancement is the reduction of the model development time until an MFM for operation-parallel simulation can be used. Nowadays, methods like response surfaces or polynomial regression approaches (see [Aro16; Jam15]) are used to develop a purely data-driven surrogate model based on time-intensive evaluations of HFMs, mainly for the purpose of optimal and probabilistic design. The required simulations for such an analysis are predefined in a DoE, often even full-factorial DoEs, for example introduced by [Mon91]. Overall, the prerequisites to develop an MFM with the purpose of accuracy enhancement are shown in Table 4.9.

4.2.1 Work Flow

Again, consulting Figure 3.2, as the first step the reusage of the design model as well as the interest region definition have to be specified. For the accuracy enhancement modeling purpose, the HFM, for example an FEM

Table 4.9 Prerequisites for accuracy enhancement using MFM.

Source	Dimension	Accuracy	Rel. Acquisition Time
Observation	o	+ +	- -
LFM	o	-/o	+ +
HFM	o	+ +	- -

- -: very bad/small | -: bad/small | o: sufficient | +: good/big | ++: very good/big

or CFD model, or in this case also expensive observations, for example caused by time-intensive measurement series, need to be prepared. The HFM needs to be executable such that specifically required model evaluations under certain boundary conditions can be simulated. In case, that observations are expensive and used as data with a higher fidelity than the fidelity of the LFM, the DoE for the measurement series needs to be prepared as well. Again, just having descriptive models like CAD models at hand is not sufficient for this modeling purpose. In terms of data fusion concepts (see section 3.2), the purpose of accuracy enhancement can be defined as complementary integration, where either HFM evaluations or observations are extended by LFM evaluations. Therefore, it is important for the model designer, that all models are able to describe the same parameters of the input and output domain. This step is done in the interest region definition. For example, when the electric power of an electric motor shall be described by the electric voltage and current, all models and observations need to include these parameters. The intrinsic model accuracy, mathematical equations or evaluation conditions, for example the range of electric voltage and current which is evaluated must differ, but the input and output domain parameters need to be the same.

As second step of the multi-fidelity modeling work flow, the LFM has to be developed. Typically, simple and fast 1D models are used as LFM for the purpose of accuracy enhancement. The intrinsic model accuracy can be much lower than the intrinsic model accuracy of the HFM. Thus, the LFM needs to be able to describe the typically non-linear system behavior based on physical laws. Often, a roughly calibrated model is sufficient to serve as LFM for the purpose of accuracy enhancement.

The MFM composition is the third step which needs to be done. The big difference to the purpose of dimensionality increase is the number of required samples due to the fact that just dynamic systems in stationary cases are considered. Hence, the selection of the right HFM samples or observations is even more important. Since the executable MFM will be a fully data-driven surrogate model, this selection can vary over time and the surrogate model can be retrained if better system insights are available. The general work flow to develop a multi-fidelity model for accuracy enhancement can be defined as:

1. Set up a DoE for the HFM or measurement series planning to get insights especially from the non-linear system responses. Make sure that $\forall\, \mathbf{v} \in \{\boldsymbol{y}^{\mathrm{H}}, \boldsymbol{y}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{M}}\}\, \exists\, i \in \mathbb{N} : \mathbf{v}_i = \boldsymbol{y}_i^{\mathrm{F}}$, meaning all system outputs need to include the parameter which shall be described in the fusion output $\boldsymbol{y}^{\mathrm{F}}$ based on the system input $\boldsymbol{u}$.

2. Execute the DoE by applying it to the HFM. Furthermore, if required, operate the system in a laboratory or shop floor environment and acquire the necessary measurements.

3. Analyze the evaluated HFM results or acquired measurements. Based on this, set up the DoE for the LFM by defining the range for $\boldsymbol{u}$. Make sure this range is as big as it is defined in the interest region definition.

4. Execute the DoE by applying it to the LFM.

5. Combine the system input $\boldsymbol{u}$, the HFM output $\boldsymbol{y}^{\mathrm{H}}$, the LFM output $\boldsymbol{y}^{\mathrm{L}}$ and the acquired observations $\boldsymbol{y}^{\mathrm{M}}$ into training, validation and test data.

6. Specify the selected fusion function set F_{f} (typically $a = b$) and execute the fusion operator training algorithm to determine the most suitable method m using the training and validation data set.

7. Assemble the MFM by determining the most suitable fusion operator for condition monitoring according to the architecture shown in Figure 3.6. Use the test data set to exploit the newly designed MFM and check if the prediction accuracy fits the need.

Finally, as fourth step of the MFM work flow the code generation for the target hardware needs to be done. In the case of model design for accuracy enhancement, typically shop floor target hardware will be used. Depending on the real time requirements, Table 4.10 shows possible target hardware.

Table 4.10 Runtime environments for accuracy enhancement applications.

Real Time Req.	**Execution Env.**	**Runtime**
Hard	PLC, FPGA	Embedded Runtime, VHDL
Soft	IPC	RT Linux, Windows Embedded

4.2.2 Benchmark System

The MFM modeling work flow for accuracy enhancement will be explained using a benchmark function. Therefore, the Himmelblau function [Him72] is used because it has non-linear properties which are still visualizable. The Himmelblau function is a common benchmark function for optimization algorithms due to its four different minima. The previously introduced work flow will be used to present the concept of accuracy enhancement in an intuitive manner. As first step of the modeling work flow, the HFM has to be prepared and the interest region has to be defined. Thus, the HFM is given by

$$y^{\mathrm{H}} = (u_1^2 + u_2 - 11)^2 + (u_1 + u_2^2 - 7)^2 \tag{4.4}$$

and the interest region for this use case is defined to be $u_1, u_2 \in [-4.0, 4.0]$. Hence, Figure 4.10 shows the HFM Himmelblau function as well as selected samples which are later on used for the development of the MFM.

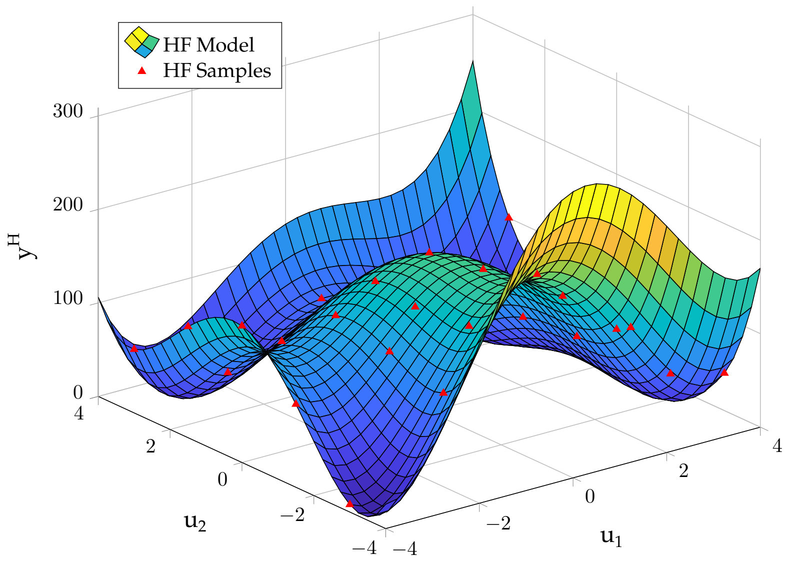


Figure 4.10 HFM Himmelblau function including selected samples.

The second step of the modeling work flow is the development of the LFM. Thus, the LFM is given by

$$y^{L} = (u_1^2 + 0.8 \cdot u_2 - 10.5)^2 + (0.9 \cdot u_1 + u_2^2 - 7.5)^2. \tag{4.5}$$

The HFM is considered to be more accurate and computationally intensive in comparison to the LFM, although it is a constructed example in this case. Overall, the Mean relative error (MRE) of the LFM in comparison to the HFM is 15.91%. For this benchmark example, the LFM is able to describe the non-linear system behavior with a lower accuracy, but still similar to the ground truth which is the HFM in this case. Figure 4.11 shows the LFM Himmelblau function and again the selected LFM samples which are required later on for the MFM development.

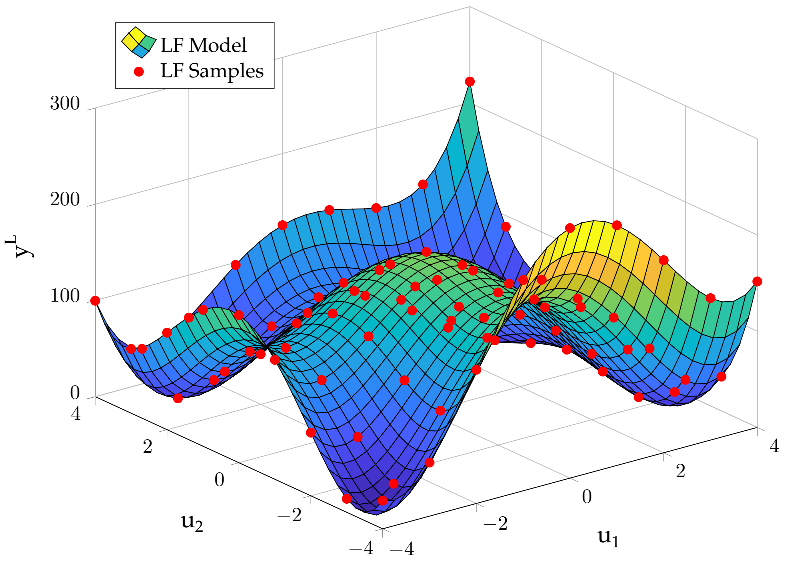


Figure 4.11 LFM Himmelblau function including selected samples.

Having both models available and an interest region defined, the MFM composition can be done. Therefore, the previously defined DoE of the HFM is executed to create HFM samples. For the benchmark example, 25 different HFM samples are generated and used for the MFM composition. Hereby, the HFM samples do not cover the whole interest region such that a complementary integration problem arises. Afterwards, 81 different LFM samples are generated which cover the whole interest region, including especially the boundaries of the interest region. Having done the DoEs, the fusion function set $F_{\mathrm{f}} = \{q[\alpha], h[\theta_h], g^c[\theta_G]\}$ is defined. All specified methods need to be part of the general fusion function set Γ_1^3 (Input: u_1, u_2, y^{L}, Output: y^{F}). moSAIc is not used as fusion function because typically not many different and similar HFMs or LFMs are available which could be leveraged as base models and the purpose is not to increase the generalization capability but the accuracy. Three different Co-Kriging approaches as well as the MLPs and QMI introduced in section 4.1 are compared as fusion methods. Overall, just 106 samples splitted in 25 HFM samples and 81 LFM samples are used to train the fusion operator. The goal is to reach the HFM accuracy by combining a few expensive HFM samples with many cheap LFM samples, which means to reduce the resulting MRE of the composed MFM in comparison to the LFM compared to the ground truth which is the HFM. The training and test results are shown in Table 4.11. Hereby, the test set consists of overall 1089 equidistantly evaluated samples of the HFM. For the DMGP approach, the neural network structure is selected as 2-6-4.

Table 4.11 ED on training and test set.

Model	ED Training	ED Test
MLP 1	509.38	3697.07
MLP 2	395.85	2297.27
MLP 3	503.76	3604.67
QMI	231.12	1599.08
ARCK	0.0078	3.46
SCK	0.0079	542.04
DMGP	0.0097	4487.75

According to the results, ARCK is the best method based on the training data and will be selected as fusion operator $\circledcirc$. The ED on the training data is computed just using the 25 known HFM samples (2.3% of the totally available samples) to demonstrate the efficiency of Co-Kriging for this modeling purpose. By contrast, the ED on the test data is computed using the whole 1089 HFM samples. All Co-Kriging approaches perform really well on the training data, but especially the DMGP performs bad on the test data. This is a result of the small training data base for the neural network. This is also the reason why all MLPs are performing pretty bad. Having just this small number of training samples available, results in a big dependency on the initialization of the weights and biases. Furthermore, the QMI does not perform really well on the training and test data, but still better than the approaches which are using a neural network. The ARCK is doing better on the test set in comparison to the generalization of ARCK which is SCK because less hyper-parameters need to be determined during the training phase. Again, the small number of training samples is the reason for this. Selected error plots compared to the ground truth are shown in the following figures. The overall MRE using ARCK can be reduced to 0.4% based on just 106 training samples.

In summary, the benchmark system for accuracy enhancement shows especially the capabilities of different Co-Kriging approaches in terms of estimating non-linear functions which are describing, for example, stationary behavior of dynamic systems. Again, the very little data base which is required to conclude from simple LFMs to detailed HFMs or observations is the key to apply MFM for accuracy enhancement. An additional advantage of Co-Kriging is its very fast evaluation time during runtime which is basically due to simple matrix multiplications that can be calculated quickly.

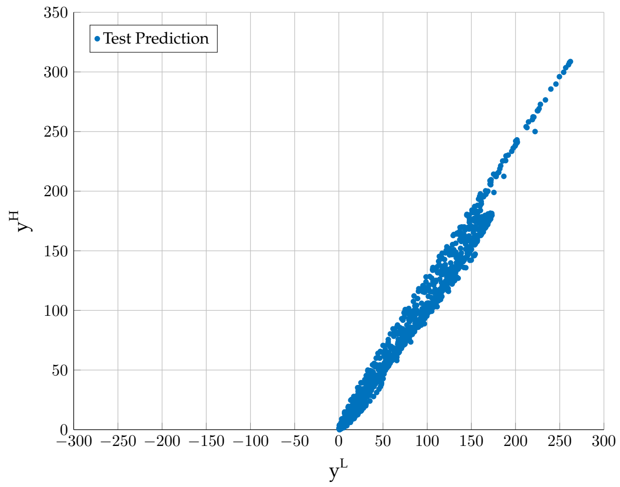


Figure 4.12 Results of accuracy enhancement: HFM-LFM comparison.

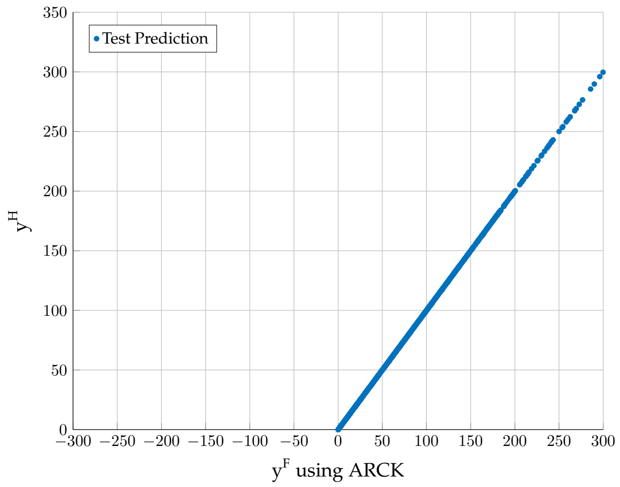


Figure 4.13 Results of accuracy enhancement: HFM-ARCK comparison.

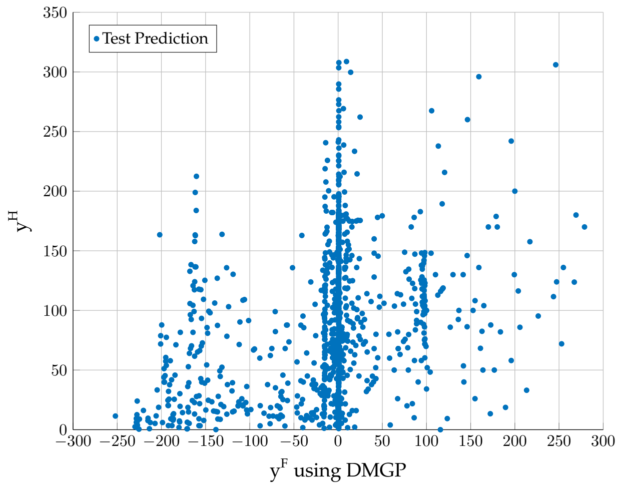


Figure 4.14 Results of accuracy enhancement: HFM-DMGP comparison.

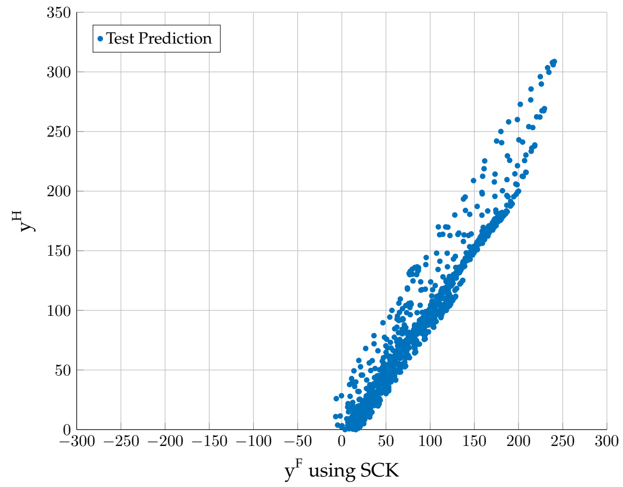


Figure 4.15 Results of accuracy enhancement: HFM-SCK comparison.

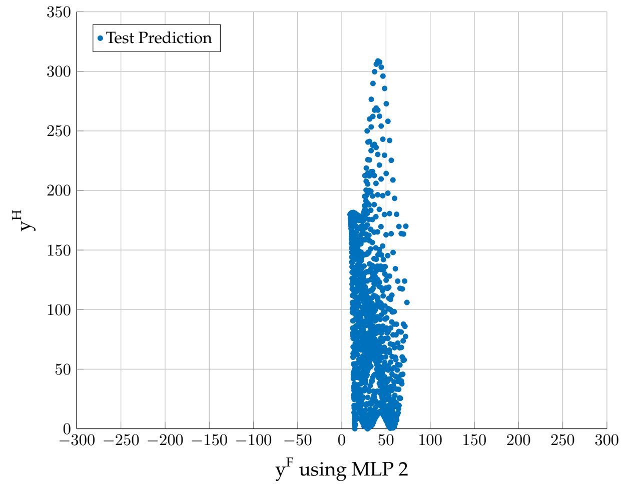


Figure 4.16 Results of accuracy enhancement: HFM-MLP2 comparison.

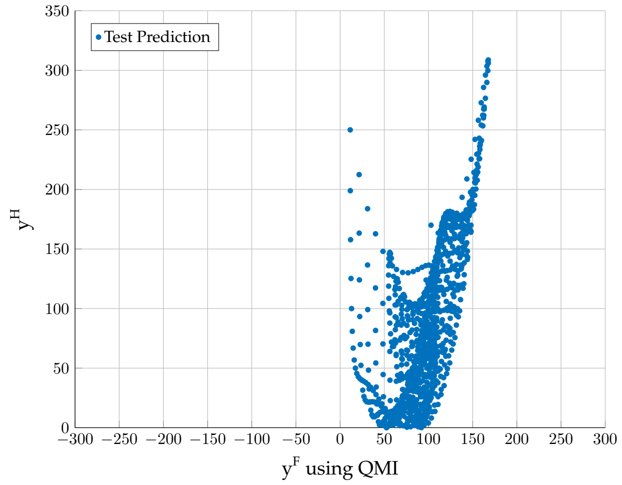


Figure 4.17 Results of accuracy enhancement: HFM-QMI comparison.

4.3 Model Design for Transferability

The purpose of the model design for transferability addresses the characteristic of transferring knowledge from known dynamic systems to unknown dynamic systems. The focus lies on the generalization capabilities of fusion methods based on static descriptive features as well as on operating conditions. This model design approach differs from the other model design approaches by focusing not on improving accuracy or increasing dimensionality but on designing a generic model. Hereby, the main purpose is not creating robust models in terms of control theory, it is more the capability to be able to emulate similar systems just based on static descriptive features and observations. The application of model design for transferability lies in the field of system variant handling. Often, many systems are used in different variants but are still based on the same physical laws, for example cars are sold in shorter or longer version with bigger or smaller tires as coupe or limousine. All chassis will face structural mechanical stress, but each car needs to be modeled individually to determine the resistance against fatigue in the specific configuration.

In terms of developing an MFM for operation-parallel simulation, existing HFMs, more specific order-reduced HFM, of known systems will be used as LFMs. Their system outputs will be fused to predict an unknown system output without actually modeling the unknown system. Although order-reduced HFMs are used, they describe the considered system inaccurately and therefore a prediction error will exist. For example, an HFM for a car with a length of $5\,\mathrm{m}$ still can be used to describe the structural mechanical stress for a car with a length of $5.5\,\mathrm{m}$, because the physical law lying behind the computation still applies. But there will be an intrinsic model error because some geometries may change and some forces will be different. Model design for transferability is useful in cases where a few order-reduced HFMs or LFMs of different dynamic system variants as well as observations from the unknown system are already available and many variants are not modeled but required to become a digital performance twin. Table 4.12 shows the prerequisites of the different MFM composition elements.

Table 4.12 Prerequisites for transferability using MFM.

Source	Dimension	Accuracy	Rel. Acquisition Time
Observation	- -	o/+	- -
LFM	- -	o/+	- -
HFM	o	+ +	+ +

- -: very bad/small | -: bad/small | o: sufficient | +: good/big | ++: very good/big

4.3.1 Work Flow

The first development step of the MFM modeling work flow is the reusage of design models as well as the interest region definition according to Figure 3.2. Applying this to the purpose of transferability, HFMs of similar systems will be reused and act as LFMs for unknown systems. The similar HFM needs to be given as an executable simulation model in the simulation environment. The more already designed models are available at this stage the better the data base and variety for the MFM composition will be. All HFMs have to be describable by the same static descriptive features, which is part of defining the interest region. These features, for example the car length or tire size, need to be determined in this stage because this will be a foundation for all following steps.

Subsequently, available observations from the unknown systems have to be determined as part of the second step of the MFM modeling work flow. Hereby, the selection of required observations should be done wisely ensuring all observation points to predict the unknown system behavior are selected on the one hand and minimizing the creation of complexity by including non information-adding observation points on the other hand. Furthermore, the LFMs which are going to be part of the MFM composition need to be specified.

This directly leads to the third step of the MFM modeling work flow, the MFM composition. Interpreting this for the purpose of transferability, the difference to the two other introduced model design purposes is the big data base which is required for the training of the fusion method. The selected different but similar systems need to be exploited as detailed as possible to provide a big training data base. This can lead to high computational effort but will pay off after the training phase because the resulting MFM will be much more accurate compared to an MFM which is trained on a smaller data base. In this case, different shapes of MFMs are possible to be used in an operation-parallel manner, either fully data-driven surrogates or hybrid approaches combining LFMs and observations. The general work flow to develop an MFM with the purpose of transferability can be defined as:

1. Set up a DoE for the HFM to exploit the required system output responses $\boldsymbol{y}$ based on different system inputs $\boldsymbol{u}$ of the similar HFMs.

2. Execute the DoE by applying it to the similar HFMs to provide a large training data base for the MFM composition.

3. Set up a DoE for measurement series to exploit the observations of at least one target system in a laboratory environment to make sure that these observations can be acquired during runtime in a sufficient quality.

4. Execute the DoE by applying it to one target system and acquire the observations.

5. Combine the system input $\boldsymbol{u}$, the HFM outputs $\boldsymbol{y}^{\mathrm{H}}$, the LFM output $\boldsymbol{y}^{\mathrm{L}}$ and the acquired observations $\boldsymbol{y}^{\mathrm{M}}$ into training, validation and test data.
6. Specify the selected fusion function set F_{f} (typically $a > b$) and execute the fusion operator training algorithm to determine the most suitable method m using the training and validation data set.
7. Assemble the MFM by determining the most suitable fusion operator for condition monitoring or according to the architecture shown in Figure 3.6. Use the test data set to exploit the newly designed MFM and check if the prediction accuracy fits the need.

The fourth step of the MFM work flow is the code generation for the target hardware. In the case of model design for transferability, the developed MFM will typically be used in a production environment with no hard real-time conditions. It is not designed for such cases, because this may go along with safety relevance and for this purpose specifically designed and really accurate models are required. Therefore, this type of model can be deployed as executable for a soft real time environment or as cloud service, for example in a containerized environment. Table 4.13 shows possible runtime environments.

Table 4.13 Runtime environments for transferability applications.

Real Time Req.[a]	Execution Env.	Runtime
Soft	IPC, DCS	RT Linux, Windows Embedded
No Req.	Cloud Service	IIoT Operating System

[a] Liu09.

4.3.2 Benchmark System

The model transfer modeling work flow will be explained using a specifically developed benchmark system. This benchmark system is designed to show the capabilities of the work flow in transferring knowledge from known dynamic systems with available models to similar but unknown dynamic systems with no available model. Like previously introduced, this type of purpose-driven modeling approach requires the largest data base. For simplification reasons, the systems for the benchmark test do not require the modeling effort which is typically required to use the model transfer work flow, but the concept can be shown anyway.

The target of this benchmark example is to predict the system response $y(t)$ without having a system model available. 16 different dynamic systems are used as training model stack and six different dynamic systems are used as test model stack. Hereby, the performance for interpolation and extrapolation will be evaluated separately. In this case, interpolation means, that unknown static descriptive features in the range of the smallest and biggest training value will be tested. In contrast, extrapolation means, that these unknown static descriptive features will be outside the known range. The training model stack consists of 16 2nd-order dynamic systems of the form which is shown in Figure 4.18, where d_b is the system damping and T_b is the system time constant.

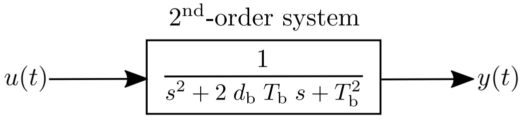

2nd-order system
$u(t)$
$\frac{1}{s^2 + 2\, d_\mathrm{b}\, T_\mathrm{b}\, s + T_\mathrm{b}^2}$
$y(t)$

Figure 4.18 Model transfer benchmark system example.

As first step of the modeling work flow for model transfer, the HFMs as well as the interest region have to be defined. In Table 4.14, the 16 different dynamic systems acting as HFMs are described by specifying the static descriptive features $C(x) = \{d_\mathrm{b}, T_\mathrm{b}\}$ which are used as training model stack. These features also define the interest region of this benchmark system which is $0.1 \leq d_\mathrm{b} \leq 0.9$ and $0.1 \leq T_\mathrm{b} \leq 0.9$. Table 4.15 shows the six test models which are considered not to be modeled, split into interpolation and extrapolation test set. All 2nd-order systems are stable since $d_\mathrm{b} < 1$ and therefore have conjugated-complex pole pairs.

After having completed the first step, the observations of the unknown systems have to be defined. In the benchmark example case, the only available observation is considered to be the system input $u(t)$. The models, which are going to be used for the MFM composition are all models of the training model stack. Hereafter, the different steps of the MFM composition for the purpose of model transfer can be done. First of all, the DoE for the exploitation of the training model stack has to be developed. In this case, the system input $u(t)$ is exploited in the range of $0.1 \leq u(t) \leq 0.9$. Overall, 100 different simulations are executed for each of the 16 systems with a simulation duration of 60 s. Hereby, the system input $u(t)$ is constant for 30 s and jumps to another amplitude for the remaining 30 s. An example for one of these exploitation simulations with the system input jumping from $u(t) = 0.3$ to $u(t) = 0.8$ is shown in Figure 4.20.

Table 4.14 Training model stack for model transfer benchmark.

Model	d_b	T_b	Model	d_b	T_b
1	0.2	0.2	9	0.6	0.2
2	0.2	0.4	10	0.6	0.4
3	0.2	0.6	11	0.6	0.6
4	0.2	0.8	12	0.6	0.6
5	0.4	0.2	13	0.8	0.2
6	0.4	0.4	14	0.8	0.4
7	0.4	0.6	15	0.8	0.6
8	0.4	0.8	16	0.8	0.8

Table 4.15 Test model stack for model transfer benchmark.

Interpolation			Extrapolation		
Model	d_b	T_b	Model	d_b	T_b
1	0.3	0.7	4	0.1	0.9
2	0.5	0.5	5	0.1	0.5
3	0.45	0.65	6	0.9	0.9

Finishing the exploitation phase, the composition of the MFM can be done by combining the system input u and the different HFM system outputs to one joint training set. In this case, four HFMs are selected to serve as LFM. These four HFMs have been randomly selected because just the concept of model transfer should be shown instead of an optimal model composition. This will be shown on the industrial case study example

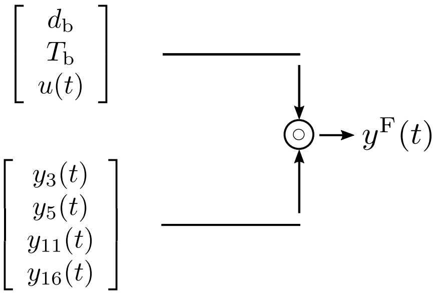


Figure 4.19 Model transfer benchmark fusion problem.

in section 5.3. The four selected HFMs are training model 3, 5, 11 and 16. Subtracting them from the training model set, the exploitation results of the 12 remaining HFMs are used as training data base, resulting in an overall size of 193,600 training samples. The selected fusion method set F_f includes the QMI, MLP and moSAIc methods for this modeling purpose and therefore can be written as $F_f = \{q[\alpha], h[\theta_h], m^s[\theta_h]\}$. All specified methods need to be part of the general fusion function set Γ_1^7. Co-Kriging is not really applicable for the modeling purpose of model transfer because it has its strength in problems which require a Γ_a^a mapping. Finally, Figure 4.19 shows the resulting fusion problem. Solving the fusion problem by computing the minimal ED on the training set by optimizing the different fusion methods delivers the results which are shown in Table 4.16. For the computation of the ED, a test set has been developed by feeding a random system input $0.0 \leq u(t) \leq 1.0$ which varies every 20 s into the interpolation and extrapolation test systems shown in Table 4.15. Figure 4.21 shows the system responses of interpolation system 2 and extrapolation system 5. Additionally, the test results of the different fusion methods are shown in Table 4.16. It can be seen, that moSAIc outperforms the other methods under consideration by 84.80% for the interpolation case and by 73.92% for the extrapolation case. Additionally, moSAIc is compared to the naive approach of similarity search. Hereby, the available system which has the most similar system properties in relation to the unknown one is compared. The results are shown in Table 4.17, where moSAIc is always compared against the most Similar System (SS). It can be concluded, that moSAIc performs really well on interpolation and sufficient on extrapolation cases.

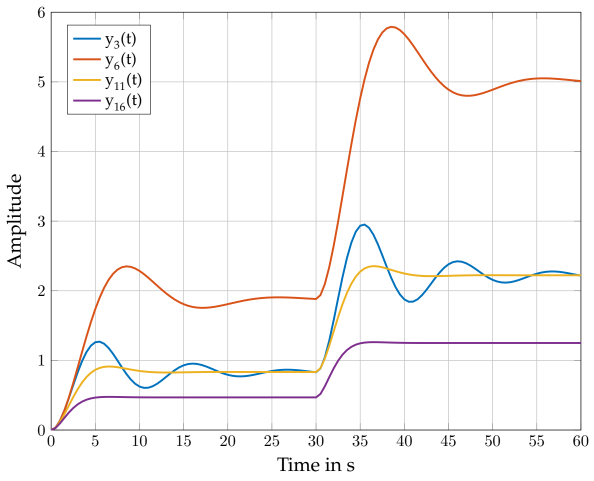


Figure 4.20 Exemplary system response of exploitation phase.

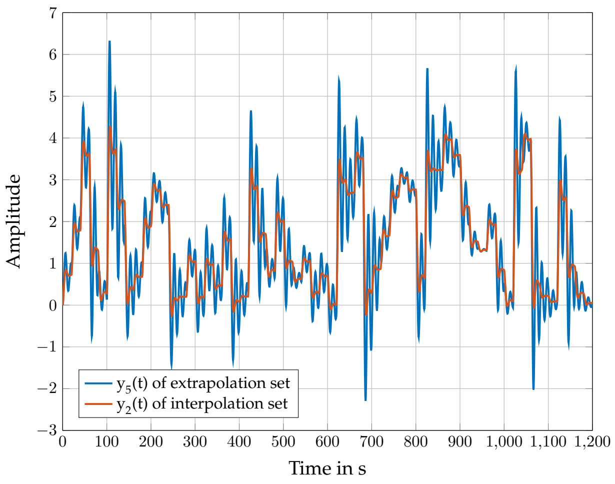


Figure 4.21 Exemplary system response of test phase.

Table 4.16 ED on inter- and extrapolation test set.

Model	ED Interpolation	ED Extrapolation
MLP 1	530.88	513.89
MLP 2	295.63	304.59
MLP 3	233.74	211.81
QMI	136.69	335.30
moSAIc	20.78	55.25

Table 4.17 Singular EDs - moSAIc compared to similarity search.

	Interpolation	
Model	Si. Search	moSAIc
1	SS3: 19.61	6.15
2	SS6: 56.07	19.26
3	SS7: 13.44	4.80
	Extrapolation	
4	SS4: 12.25	17.84
5	SS2: 67.40	49.89
6	SS16: 10.59	15.68

Summarizing the benchmark system evaluation for model transfer, the capabilities of the MFM modeling approach in terms of transferability to systems where no model but static descriptive features are available is shown. The biggest difference in comparison to the other modeling purposes is introduced as the large data base which is required for MFM for model transferability. The model transfer is done by learning relationships between available models and conclude to unknown models using an MFM which is composed by so-called base models as LFMs, system observations and static descriptive features. The performance of the previously introduced method moSAIc was shown in comparison to other suitable methods. Hence, moSAIc delivers consistently good results for inter- and extrapolation purposes. Furthermore, the limited generalization capabilities of multivariate regression approaches like QMI have been shown since QMI performs closely three times better on the interpolation set than on the extrapolation set.

5 Industrial Case Studies on Multi-fidelity Modeling

This chapter presents four different industrial case studies on dimensionality increase, accuracy enhancement and transferability. As example for dimensionality increase and soft-sensoring using multi-fidelity modeling, a model to estimate the temperature distribution of an electric motor is explained. The pressure drop control of an industrial flapper valve acts as case study for accuracy enhancement. Finally, a multi-fidelity model for the estimation of remaining useful life of different electric motor derivatives is introduced.

5.1 Temperature Estimation for Electric Motors

5.1.1 Case Study Description

Interactive predictive maintenance services including the computational model as well as an intuitive visualization become more and more relevant in the industry. In this context, an interactive thermal condition monitoring service for an electric asynchronous motor is introduced. The service requirements are a quickly executable thermal simulation model of an electric asynchronous motor which has the capability to estimate the temperature distribution on the rotor, stator and shaft. Additionally, a convenient side effect will be that the temperature distribution can be visualized interactively using a virtual reality glass. The purpose of online condition monitoring of electric motors is to provide insights on the actual motor system state and get interactive feedback based on actions which have applied to the real motor. The approach of multi-fidelity modeling for dimensionality increase is used to reuse thermal FEM models from the design phase of the motor. The applied multi-fidelity modeling work flow is shown in Figure 5.1.

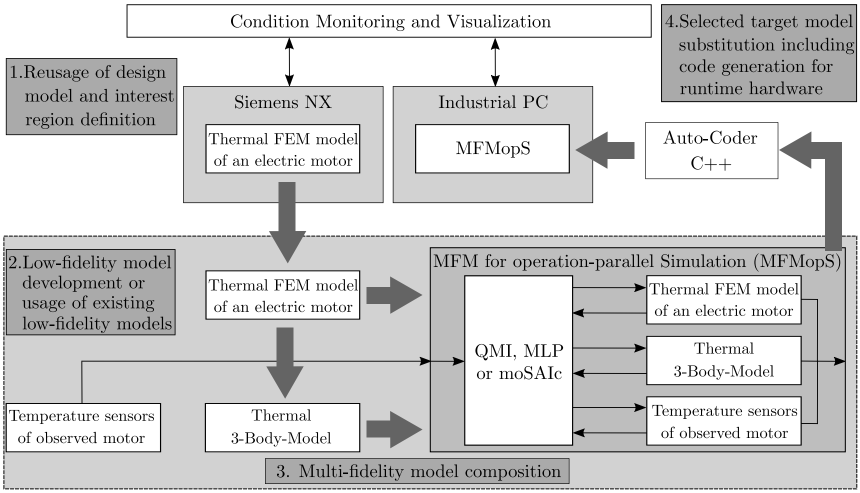
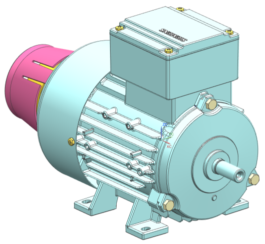

Condition Monitoring and Visualization
4.Selected target model substitution including code generation for runtime hardware
1.Reusage of design model and interest region definition
Siemens NX
Thermal FEM model of an electric motor
Industrial PC
MFMopS
Auto-Coder C++
2.Low-fidelity model development or usage of existing low-fidelity models
Thermal FEM model of an electric motor
MFM for operation-parallel Simulation (MFMopS)
QMI, MLP or moSAIc
Thermal FEM model of an electric motor
Thermal 3-Body-Model
Temperature sensors of observed motor
Temperature sensors of observed motor
Thermal 3-Body-Model
3. Multi-fidelity model composition

Figure 5.1 Development work flow for electric motor temperature estimation.

First of all, the thermal FEM model from the design phase of the motor is introduced in more detail. The base for the FEM model is a detailed CAD model. Starting with the CAD model, all non-simulation-relevant parts, for example small screws, need to be erased. As result of this step, the pure simulation-relevant geometry remains from the CAD model. In Figure 5.2, the remaining CAD model for the considered case study is shown. This model includes the motor housing, the stator, the rotor, the windings and the shaft, because these are the necessary parts.

After the relevant parts of the CAD model are isolated, the FEM model is created by specifying the physical phenomena which shall be simulated and specifying the meshing of the CAD model. In this case, 30619 nodes are generated on the geometry defined by a specific location on the grid. The physical phenomena to be simulated are heat conduction and heat convection. According to [Eva08], heat convection can be described as

$$\partial \frac{\vartheta(\vec{\mathbf{x}},t)}{\partial t} = -\Lambda_{\mathrm{th}}\, \Delta\, \vartheta(\vec{\mathbf{x}},t) \,, \tag{5.1}$$


(a) Motor Housing

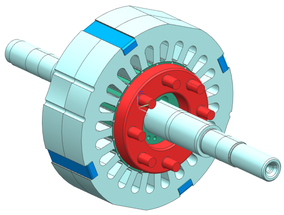

(b) Stator, Rotor, and Shaft Model

Figure 5.2 CAD models of the asynchronous motor.

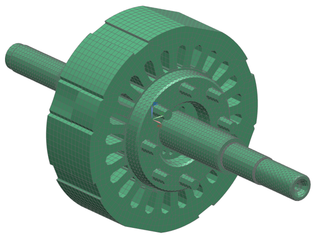
where $\vartheta(\vec{\mathbf{x}}, t)$ is the temperature depending on the location $\vec{\mathbf{x}}$ and the time t, $\Lambda_{\text{th}} \in \mathbb{R}_{\geq 0}$ is the conductivity constant depending on the medium and Δ is the Laplace operator. Additionally, heat conduction can be described by

$$\partial \frac{Q_{\text{H}}(t)}{\partial t} = C_{\text{th}} \cdot (\vartheta_1(t) - \vartheta_2(t)) \ , \tag{5.2}$$

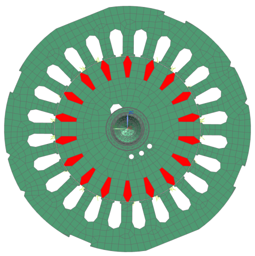
where $Q_{\text{H}} \in \mathbb{R}$ is the heat quantity and $C_{\text{th}} \in \mathbb{R}_{\geq 0}$ is the material-specific thermal capacity. It is important to specify the material properties of the different components, which are aluminum for the housing and rotor, steel for the stator and the shaft as well as copper for the windings in this case. Afterwards, boundary conditions are set defining an ambient temperature of $\vartheta_{\text{amb}} = 20^{\circ}C$, which serves also as initialization temperature for all components. The resulting FEM model is shown in Figure 5.3 (a). According to [PG15], the thermal losses are mainly induced by power heat losses of the ohmic resistance in the stator windings and aggregate to about 70-75% of the overall losses. Therefore, the thermal loads for the motor are placed in the windings. Hence, the placement of the thermal heat sources is shown in Figure 5.3 (b). Concluding this, the thermal loss P_{V} can be computed by

$$P_{\text{V}}(t) = 0,75 \cdot \frac{\pi \cdot n(t)}{30} \cdot T(t) \left(\frac{1}{\eta_{\text{M}}} - 1 \right) = \frac{\pi \cdot n(t) \cdot T(t)}{40} \left(\frac{1}{\eta_{\text{M}}} - 1 \right) , \tag{5.3}$$

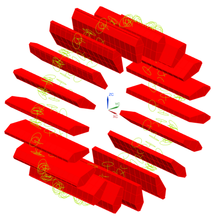
with n being the motor speed in rpm, T being the motor torque in Nm and $0 \leq \eta_{\text{M}} \leq 1$ being the motor efficiency. This thermal loss is assumed to be equally distributed over all heat sources in the FEM model, meaning



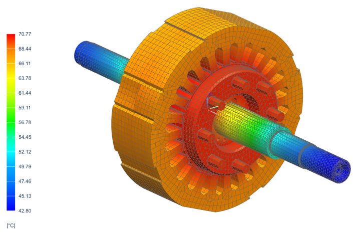
(a) FEM model of the motor



(b) Placement of thermal loads



(c) Close up of the thermal load



(d) Temperature distribution result plot

Figure 5.3 FEM models of the asynchronous motor.

all heat sources, which are shown in Figure 5.3 (b), radiate an equally distributed heat. With this, the FEM simulation model can be executed. For every time step k the temperature at each of the 30619 nodes is computed based on the introduced laws. The FEM model serves as HFM in terms of the multi-fidelity modeling work flow. The second big element of the case study work flow (see Figure 5.1) is the 3-Body-Model according to [Ném18], which serves as LFM. Hereby, the whole motor is subdivided into three different main bodies, the stator sheet pack (indexed with (fe)), the stator iron including windings (indexed with (cu)) and the rotor (indexed with (r)). The temperatures of the three bodies are computed by

$$\begin{aligned}\vartheta_{k+1}^{(\mathrm{fe})} &= \vartheta_k^{(\mathrm{fe})} + \left(-\frac{\Lambda_{\mathrm{th}}^{(\mathrm{fe})} + \Lambda_{\mathrm{th}}^{(\mathrm{cu})} + \Lambda_{\mathrm{th}}^{(\mathrm{r})}}{C_{\mathrm{th}}^{(\mathrm{fe})}}\right)\vartheta_k^{(\mathrm{fe})} \\ &\quad + \frac{\Lambda_{\mathrm{th}}^{(\mathrm{cu})}}{C_{\mathrm{th}}^{(\mathrm{fe})}}\vartheta_k^{(\mathrm{cu})} + \frac{\Lambda_{\mathrm{th}}^{(\mathrm{r})}}{C_{\mathrm{th}}^{(\mathrm{fe})}}\vartheta_k^{(\mathrm{r})} + \frac{P_{\mathrm{V}}^{(\mathrm{fe})}}{C_{\mathrm{th}}^{(\mathrm{fe})}}, \end{aligned} \tag{5.4}$$

$$\vartheta_{k+1}^{(\mathrm{cu})} = \vartheta_{k}^{(\mathrm{cu})} + \left(\frac{P_{\mathrm{V}}^{(\mathrm{cu},20°\mathrm{C})} \alpha^{(20°\mathrm{C})} - \Lambda_{\mathrm{th}}^{(\mathrm{cu})}}{C_{\mathrm{th}}^{(\mathrm{cu})}} \right) \vartheta_{k}^{(\mathrm{cu})} + \frac{\Lambda_{\mathrm{th}}^{(\mathrm{cu})}}{C_{\mathrm{th}}^{(\mathrm{cu})}} \vartheta_{k}^{(\mathrm{cu})} + \frac{P_{\mathrm{V}}^{(\mathrm{cu},20°\mathrm{C})}}{C_{\mathrm{th}}^{(\mathrm{cu})}}, \tag{5.5}$$

$$\vartheta_{k+1}^{(\mathrm{r})} = \vartheta_{k}^{(\mathrm{r})} + \frac{\Lambda_{\mathrm{th}}^{(\mathrm{r})}}{C_{\mathrm{th}}^{(\mathrm{r})}} \vartheta_{k}^{(\mathrm{fe})} - \frac{\Lambda_{\mathrm{th}}^{(\mathrm{r})}}{C_{\mathrm{th}}^{(\mathrm{r})}} \vartheta_{k}^{(\mathrm{r})} + \frac{P_{\mathrm{V}}^{(\mathrm{r})}}{C_{\mathrm{th}}^{(\mathrm{r})}}. \tag{5.6}$$

The non-linear 3-body-model by [Ném18] is a commonly used temperature estimation model in the field of safety applications to protect asynchronous motors from overheating. As last element of the multi-fidelity modeling work flow (see Figure 5.1), observations from the system are required. These observations are acquired to compute the thermal losses which serves as system input u for both the HFM and LFM. Therefore, the active electric power as well as the active torque and rotation speed need to be measured. Furthermore, two temperature sensors are considered to be mounted on the electric motor on the stator iron and inside the winding. This consideration is common, because many electric asynchronous motors include these two sensors for motor protection against overheating. For this case study, the two measurements are emulated by modulation of normal-distributed zero-mean noise on two HFM nodes.

The target of this industrial case study is the approximation of 94 selected nodes from the rotor, stator, shaft and windings based on information of two observations from the stator iron and windings as well as three computed temperatures from an LFM. Therefore, all suitable fusion methods need to be part of the general fusion function set Γ_{94}^{5}. The industrial case study takes a situation into account, where an asynchronous motor with a nominal power of 250 W and an efficiency $\eta_{\mathrm{M}} = 61.9\%$ has two operating modes: switched on and switched off. If the motor is switched on, a constant thermal load of

$$P_{\mathrm{V}}(t) = \frac{\pi \cdot 1000 \, \frac{1}{\mathrm{min}} \cdot 1 \, \mathrm{Nm}}{40} \left(\frac{1}{0.619} - 1 \right) = 48.32 \, \mathrm{W}, \tag{5.7}$$

is applied, otherwise a thermal load of 0 W is applied.

Having now specified step one and step two of the design work flow for dimensionality increase, the MFM composition starts. Referring to section

4.1.1, the DoE for the exploitation of the HFM is the first step which needs to be done. The training data set for this industrial case study is generated by running the motor for two hours: one hour switched on and afterwards one hour switched off starting from an initial temperature of $20\ ^{\circ}C$. A temperature sample of each HFM node as well as of each LFM dimension and observation is acquired every 30 seconds. The simulation execution time of the HFM is about one hour, the simulation execution time of the LFM is about a few seconds. Afterwards, 30619 nodes $\cdot$ 241 time steps $= 7,319,179$ HFM temperature samples, 3 bodies $\cdot$ 241 time steps $= 723$ LFM temperature samples and 2 sensors $\cdot$ 241 time steps $= 482$ observation samples are available. All three different information sources are synchronized in time. The HFM temperature samples which are relevant for the MFM composition sum up to 94 selected nodes $\cdot$ 241 time steps $= 22,654$ temperature samples which are selected from the $7,319,179$ HFM temperature samples. Thus, the training data set is assembled and the fusion operator training starts. Therefore, the fusion function set is specified as $F_\mathrm{f} = \{q[\alpha], h[\theta_h], m^\mathrm{s}[\theta_h]\}$ and the MFM can be assembled by solving the fusion operator optimization problem.

5.1.2 Case Study Results

The used methods of the fusion function set F_f are already introduced in chapter 3. For this case study, the design of moSAIc has to be introduced. Therefore, Figure 5.4 shows the applied design which competes against the other fusion methods. $\boldsymbol{u}(t)$ consists of the motor state, which means whether the motor is switched on ($\boldsymbol{u}(t) = 1$) or off ($\boldsymbol{u}(t) = 0$). For the static descriptive system information $\boldsymbol{C}$, the location of the respective temperature node is used. Hereby, the following scheme is used: If the temperature node is located on the

- stator, its corresponding index is $\boldsymbol{C} = 1$,
- circuit rings, its corresponding index is $\boldsymbol{C} = 2$,
- rotor, its corresponding index is $\boldsymbol{C} = 3$
- windings, its corresponding index is $\boldsymbol{C} = 4$.

Afterwards, the temperature estimations of the 3-body-model $y_1^{\mathrm{L}}, y_2^{\mathrm{L}}, y_3^{\mathrm{L}}$, and the observations $y_1^{\mathrm{M}}, y_1^{\mathrm{M}}$ are fused to estimate the 94 temperature nodes of the HFM.

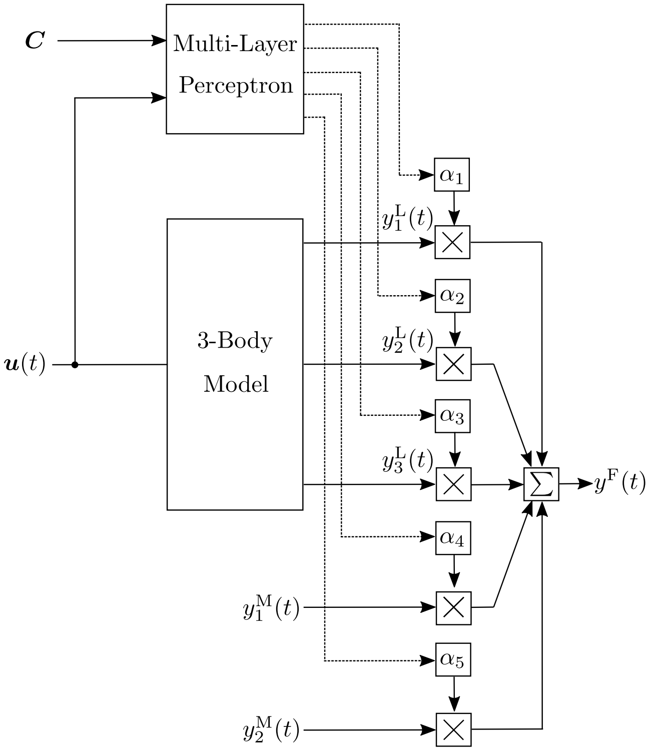


Figure 5.4 moSAIc architecture for case study.

The results of the individual methods which are part of the fusion function set F_{f} are shown in Table 5.1. From this table, it can be concluded, that the MLP 2 outperforms all other methods in both the training as well as the test set. This MLP method in general does perform consistently on the data sets, which leads to the conclusion that the ReLu activation function and the hidden layer size are most suitable for this specific application. In comparison to moSAIc, MLP 2 has an MRE of 4.1% on the training set and 8.6% on the test set, whereas moSAIc has an MRE of 3.7% on the training set and 7.9% on the test set. This leads to the conclusion, that moSAIc has bigger individual deviations on specific samples than MLP 2 which is a reason to choose the euclidean distance as measure for the optimization.

Table 5.1 ED on training and test set.

Model	ED Training	ED Test
MLP 1	5411.16	8868.00
MLP 2	232.83	1336.13
MLP 3	6432.27	10346.28
QMI	18367.37	16000401.21
moSAIc	539.28	1452.60

To illustrate the results more comprehensibly, Figure 5.5 shows a good example and Figure 5.6 shows a bad example on estimating the corresponding dimension´s temperature.

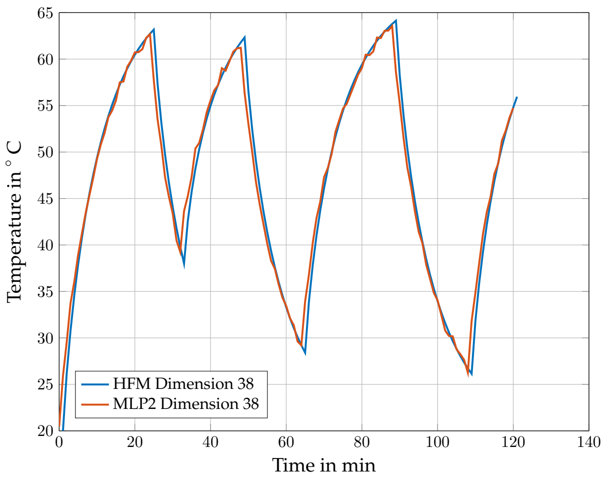


Figure 5.5 Example for a good temperature estimation.

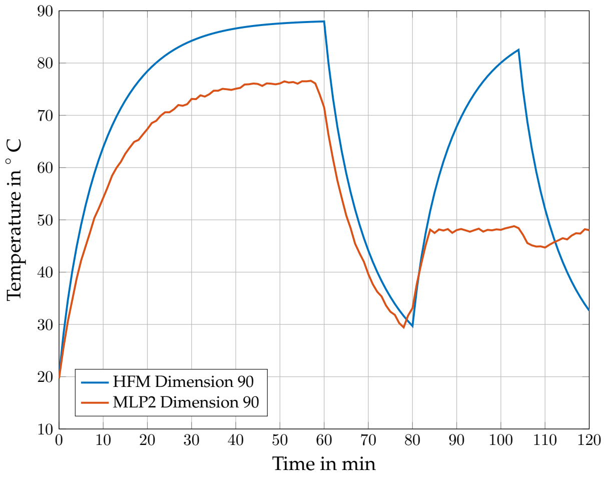


Figure 5.6 Example for a bad temperature estimation.

Concluding the results of the case study, the MLP 2 is chosen as method to estimate temperatures on different points of interest of an electric asynchronous motor during its operation. The constructed digital performance twin for condition monitoring purposes provides an acceptable accuracy with an MRE lower than 10% and transfers the knowledge from the design phase of the motor down to its operation. moSAIc could also be the right choice for this application because its performance is similar to the MLP 2. All other methods are not suitable according to Table 5.1. Pointing out to the real world application of this case study the computed temperatures are read from a visualization device like a virtual glass or tablet during the operation of the motor. This allows a plant's maintenance staff to interactively act with the motor using its digital performance twin.

5.2 Pressure Drop Control of a Flapper Valve

5.2.1 Case Study Description

The reduction of the time-to-market of digital assets is becoming an important field of industrial research. In this context, the digital performance twin generation of an industrial flapper valve is going to be introduced in the following. Flapper valves are commonly used in process industries to control volume flows and pressure drops within pipelines for example. The purpose of this case study is the accuracy enhancement of a low-fidelity model of the flapper valve which is going to be enhanced with high-fidelity samples to provide a multi-fidelity model. The resulting digital performance twin can be used for valve control for example. More in detail, the stationary pressure drop p of the flapper valve shall be controlled by measuring the volume flow Q_{V} and by controlling the valve lift D. Nowadays, this is done using a response surface model created by an expensive measurement set or computationally expensive CFD simulations. The accuracy enhancement approach is used to reduce the development time of the digital performance twin and replace the response surface approach with a multi-fidelity model. Figure 5.7 shows the applied multi-fidelity modeling work flow.

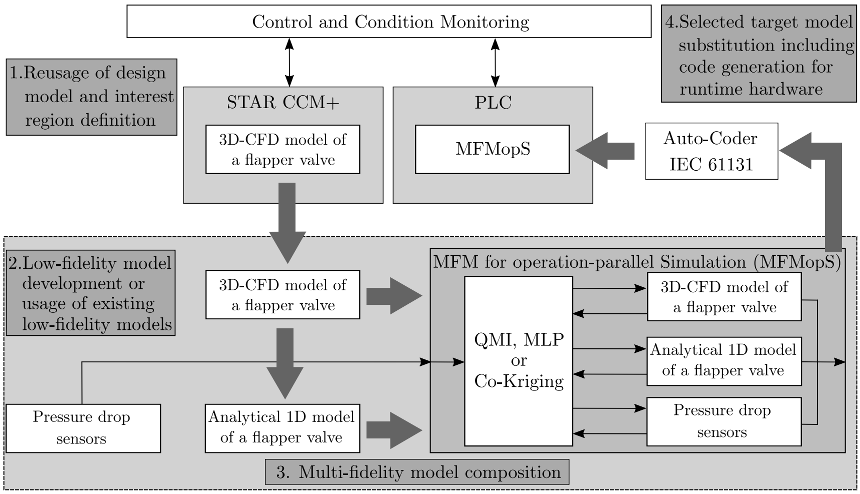
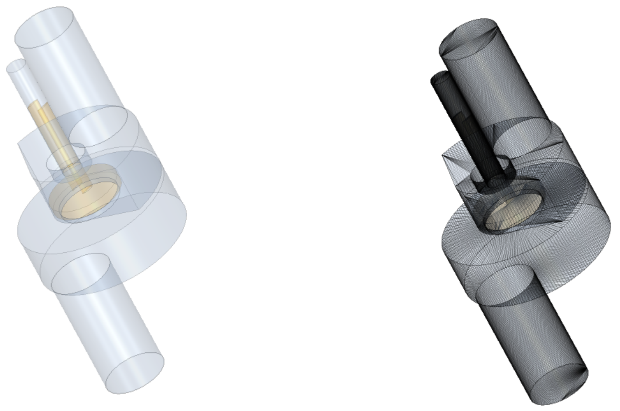

Control and Condition Monitoring
4.Selected target model substitution including code generation for runtime hardware
1.Reusage of design model and interest region definition
STAR CCM+
3D-CFD model of a flapper valve
PLC
MFMopS
Auto-Coder IEC 61131
2.Low-fidelity model development or usage of existing low-fidelity models
3D-CFD model of a flapper valve
MFM for operation-parallel Simulation (MFMopS)
QMI, MLP or Co-Kriging
3D-CFD model of a flapper valve
Analytical 1D model of a flapper valve
Pressure drop sensors
Pressure drop sensors
Analytical 1D model of a flapper valve
3. Multi-fidelity model composition

Figure 5.7 Development work flow for the flapper valve volume flow prediction model.

The high-fidelity model of the flapper valve is given as a turbulent CFD simulation model implemented in STAR CCM+. The geometry of the flapper valve is reused from its design phase and exploited with the CFD tool. More in detail, the Navier-Stokes-Equations are solved using the FVM such that the fluid velocity of each volume is computed individually. The applied geometry and the corresponding CFD mesh are visualized in Figure 5.8.



(a) Flapper valve geometry (b) Flapper valve mesh

Figure 5.8 Flapper valve representations in STAR CCM+.

According to [Bun+13], the Navier-Stokes-Equations for incompressible fluids are given as

$$\frac{\partial}{\partial t}\boldsymbol{v}_\mathrm{f} + (\boldsymbol{v}_\mathrm{f} \cdot \nabla)\, \boldsymbol{v}_\mathrm{f} = -\nabla \boldsymbol{p} + \frac{1}{\mathrm{Re}}\Delta \boldsymbol{v}_\mathrm{f} + \boldsymbol{g} \ , \tag{5.8}$$

$$\mathrm{div}\ \boldsymbol{v}_\mathrm{f} = 0 \ , \tag{5.9}$$

where $\boldsymbol{v}_\mathrm{f}$ is the fluid velocity field, ∇ is the Nabla operator, Δ is the Laplace operator, $\boldsymbol{p}$ is the pressure, Re is the Reynolds number and $\boldsymbol{g}$ is the external force field, for example gravitation. The first Navier-Stokes-Equation is also called impulse equation and the second one is also called continuity equation. The Navier-Stokes-Equations are solved using the FVM. Since these methods are only leveraged to compute the pressure drop p based on volume flow Q_V and valve lift D, they are not going to be introduced further. For more information, [Bun+13] introduces theses methods in detail. An exemplary simulation result for a fluid velocity field is shown in Figure 5.9. The overall simulation time of this high-fidelity model evaluation for only one combination of boundary conditions and water as liquid sums up to approximately 19 minutes.

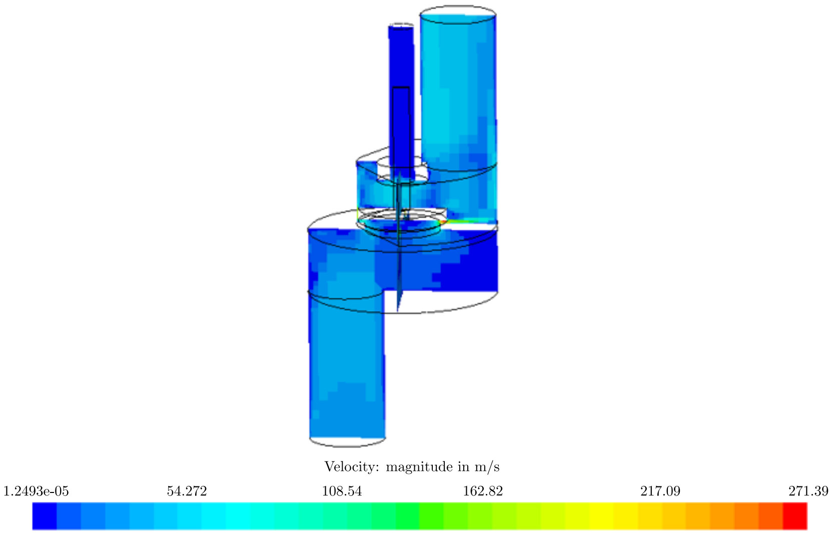


Figure 5.9 Fluid velocity plot after simulation.

Having introduced the case study background and HFM, the MFM design work flow for accuracy enhancement can be applied. The reusage of the design model is done by importing the geometry from the design phase of the flapper valve in the CFD simulation tool. For this case study, the interest region is defined as

$$R_\mathrm{Q} := [Q_\mathrm{V} = 50.0\,\mathrm{l/min},\ Q_\mathrm{V} = 250.0\,\mathrm{l/min}]\,, \tag{5.10}$$

$$R_\mathrm{D} := [D = 0.5\,\mathrm{mm},\ D = 5.0\,\mathrm{mm}]\,. \tag{5.11}$$

One requirement which needs to be fulfilled for the purpose of accuracy enhancement, is that the LFM and HFM cover the same input and output domain. Therefore, the LFM is going to be developed to fulfill this requirement as second step of the MFM design work flow.

As low-fidelity model, a simplified 1D model is used within the software environment of Simcenter Amesim to predict the pressure drop p of a flapper valve based on the volume flow Q_V and the valve lift D. Figure 5.10 shows the flapper valve representation in the 1D simulation environment.

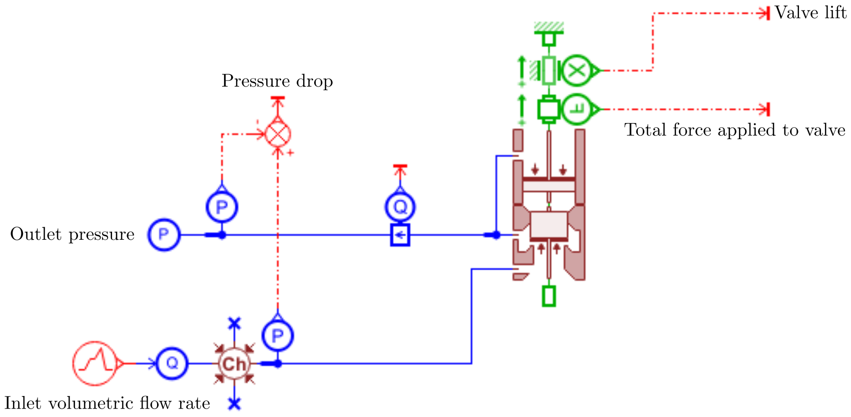
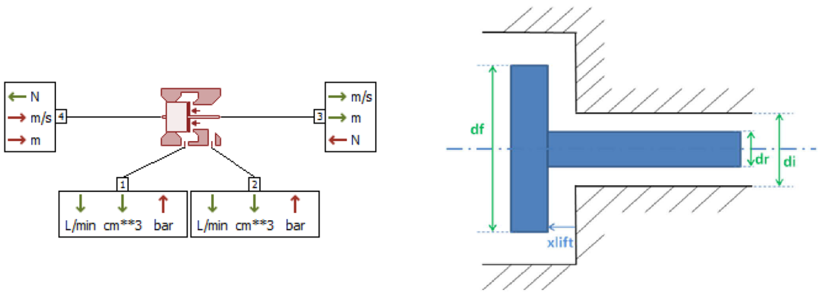


Figure 5.10 1D Amesim model of the flapper valve.

The flapper valve consists of a one-dimensional motion of a nozzle poppet acting on a flat flapper seat. Figure 5.11 shows the detailed component with its ports and the cross-section of the valve. The pressure p is computed at each hydraulic port of the valve. For the pressure computation at port 2 it is assumed that the pressure acts on an active area adjacent to the orifice and tends to open the orifice. Contrary, for the pressure computation at port 1 the pressure acts on the flapper seat area, which is a valid assumption for turbulent flows. The resulting outlet pressure is computed by applying

$$p = \frac{\rho}{2} \cdot \left(\frac{Q_V}{c_q \cdot A_F} \right)^2 , \tag{5.12}$$

where ρ is the density of the fluid, c_q is a parametric flow coefficient and A_F is the cross-sectional flapper valve area. Again, the modeling of the flapper valve is not focus of this case study and is therefore not explained further. The overall simulation time for one model evaluation sums up to approximately one second.



(a) Flapper valve component. (b) Flapper valve geometry.

Figure 5.11 Flapper valve representations in Amesim.

Having now both, the HFM and LFM, at hand, the MFM composition can be applied. For the DoE of the HFM, 15 randomly picked boundary conditions within the interest region are selected and simulated using the CFD simulation tool. Hereby, the simulation is executed until the solution reaches a stationary state fulfilling a stopping criteria. Afterwards, the DoE for the LFM exploitation is set up. For this case study, 50 different boundary conditions are simulated using the 1D simulation model. The selected fusion function set for this case study can be written as $F_\mathrm{f} = \{q[\alpha], h[\theta_h], g^c[\theta_G]\}$.

5.2.2 Case Study Results

For training purposes, many different HFM and LFM sample combinations are possible. For this case study, overall 19 HFM samples and 50 LFM samples are going to be combined as MFM by solving the fusion operator optimization problem. The HFM samples are selected wisely due to the very little data which is available and physical knowledge of the problem. The selected training samples of the HFM are visualized in Figure 5.12 and the selected training samples of the LFM are visualized in Figure 5.13.

Table 5.2 shows the results of the different fusion methods for both the training data and the test data. From these results can be concluded, that the Gaussian process based methods are most suitable to act as MFM to predict the pressure drop p based on volume flow Q_V and valve lift D.

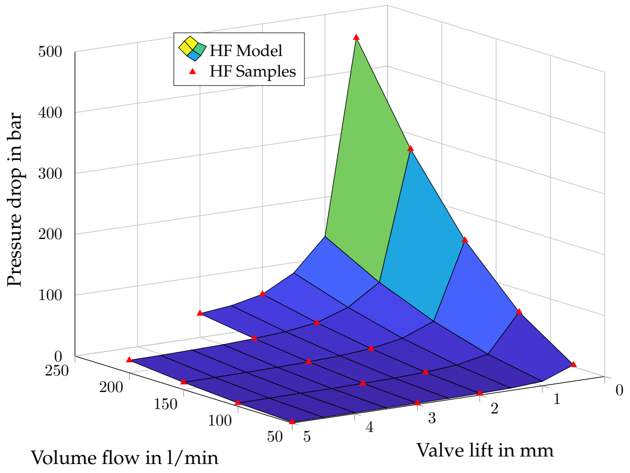


Figure 5.12 HFM flapper valve function including selected samples.

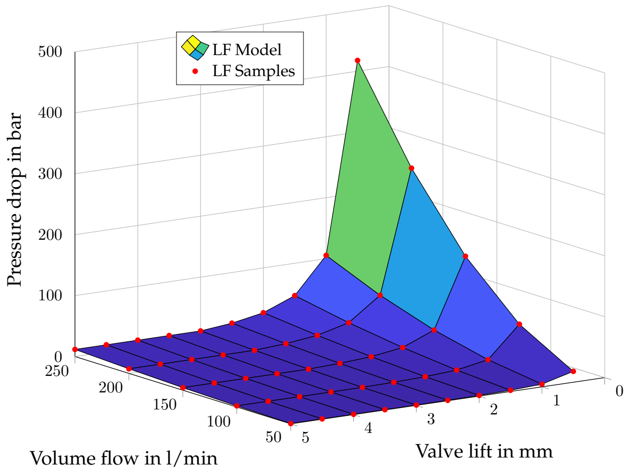


Figure 5.13 LFM flapper valve function including selected samples.

Unfortunately, the data basis is to small to train the DMGP model. The training fails due to issues while doing the Cholesky decomposition. SCK and ARCK outperform all other methods by far, because these methods do not require large data sets like the other methods do.

Table 5.2 ED on training and test set.

Model	ED Training	ED Test
MLP 1	1133.50	1834.50
MLP 2	473.24	665.77
MLP 3	589.17	628.50
QMI	204.57	426.84
ARCK	0.01	13.09
SCK	0.02	12.46

To underline these results, the Figures 5.14 and 5.15 show the performance of the LFM and the MFM using SCK in comparison to the HFM. The MRE in terms of accuracy of the LFM in comparison to the HFM is 54.36%. The MFM using SCK reaches an MRE of 9.38% which is much better than the standalone LFM.

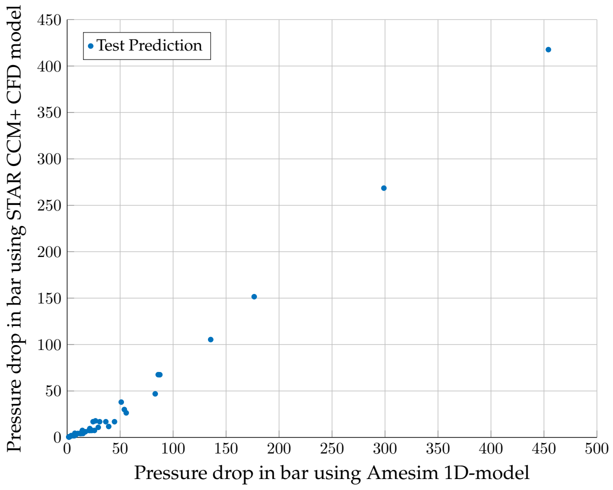


Figure 5.14 Results of accuracy enhancement: HFM-LFM comparison.

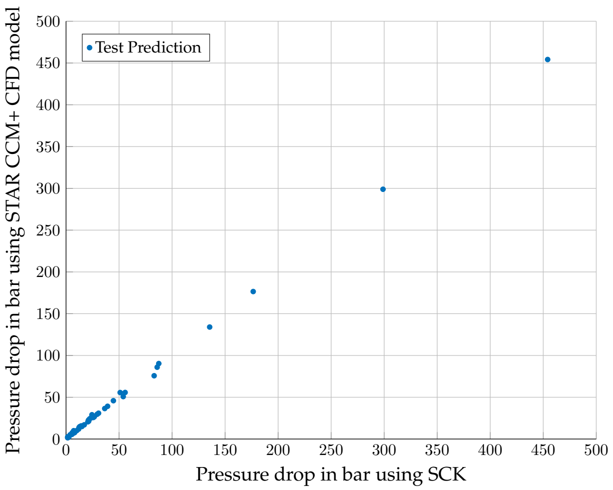


Figure 5.15 Results of accuracy enhancement: HFM-SCK comparison.

Concluding these results, the digital performance twin of a flapper valve which could be used for control purposes in process industries can be designed using multi-fidelity modeling with the purpose of accuracy enhancement. Nowadays, response surface models are created to act as surrogate models for these applications. The development time of the MFM in comparison to the response surface model can be reduced by 62.5% from approximately 16 hours for the response surface model down to approximately 6 hours for the MFM, accepting an MRE of 9.38% . This methodology can be scaled to many different designs of industrial flapper valves.

5.3 Motor Fleet Condition Monitoring

5.3.1 Case Study Description

The engineering process to develop digital performance twins is time-intensive in many application fields and requires expert knowledge. This can lead to really high development costs for digital performance twins and sometimes the costs overcome the need. This industrial case study focuses on the purpose of model design for transferability. Condition-based maintenance (CBM) for electric motors is a strategy to monitor the lifetime degradation process of industrial machinery due to different load-profiles. According to [Lei+18], the lifetime of industrial machinery components can be increased and unnecessary maintenance operations can be reduced with CBM. The main purpose of this case study is to estimate the mechanical Remaining useful life (RUL) of an electric motor. For one single electric motor, the modeling effort to create an individual digital performance twin can sum up to one month of engineering. It is not profitable to model whole motor fleets, meaning similar variants of an electric motor with individual properties. Therefore, the MFM modeling work flow with the purpose of transferability is applied to create a generic digital performance twin for condition monitoring. The modeling work flow for this industrial case study is visualized in Figure 5.16.

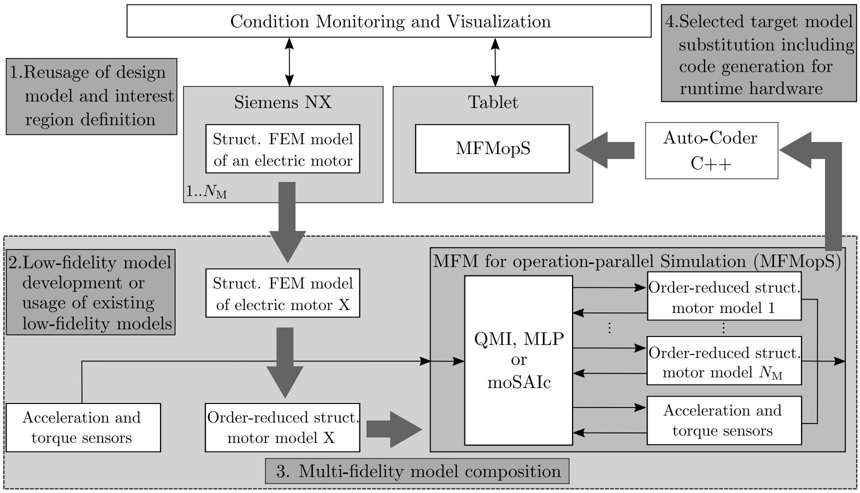

Condition Monitoring and Visualization
1.Reusage of design model and interest region definition
Siemens NX
Struct. FEM model of an electric motor
1..N_M
Tablet
MFMopS
4.Selected target model substitution including code generation for runtime hardware
Auto-Coder C++
2.Low-fidelity model development or usage of existing low-fidelity models
Struct. FEM model of electric motor X
Order-reduced struct. motor model X
Acceleration and torque sensors
MFM for operation-parallel Simulation (MFMopS)
QMI, MLP or moSAIc
Order-reduced struct. motor model 1
Order-reduced struct. motor model N_M
Acceleration and torque sensors
3. Multi-fidelity model composition

Figure 5.16 Development work flow for motor fleet RUL estimation.

Generally, RUL can be predicted using many different methodologies. Figure 5.17 gives an overview about these different methods. A commonly used approach for RUL assessment bases on physics-based simulation models. Hereby, the simulation models are used to evaluate the stress dis-

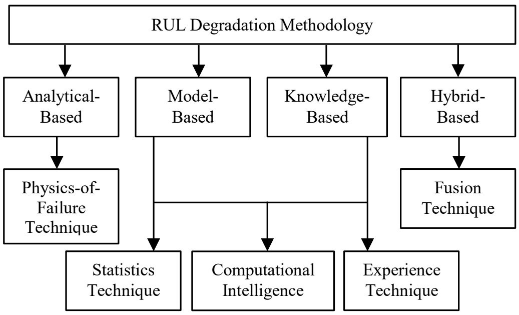


Figure 5.17 RUL estimation methodologies according to [Oko+14].

tribution on the motor under certain operating conditions. From the resulting stress distribution, critical regions are identified and the corresponding stress values are input to a material-specific Stress-cycle-curve (SNC) with the goal of computing the corresponding number of cycles to failure. High-cycle fatigue is one of the lifetime degradation modes, where loads below the yield strength of the material for $10^3 - 10^6$ cycles are applied to the component. According to [San+16], physics-based models have been developed and used to model high-cycle fatigue. In order to introduce this approach into an online monitoring framework, features are extracted from sensor signals which are used to identify the operating conditions and anomalies. These features are later on fed into the simulation model as boundary conditions.

The FEM is used to estimate the motor's RUL due to high-cycle fatigue by simulating its response under certain operating conditions. System displacements are the raw solution of the FE solver. These displacements need to be post-processed to compute the resulting stress distribution for given operating conditions by using a Basquin [Lee05] or Wöhler [Hai06] model. The second-order stress tensor is reduced to a single equivalent component by either selecting the most influential component of the damage process, for example shear stress, or by calculating an equivalent stress S_{eq} of the full tensor, such as the maximum-principle stress according to [FY98]. Hereby, the Basquin equation

$$S_{\text{eq}} = S_{\text{f}}\, N_{\text{L}}^{z}, \tag{5.13}$$

is used to estimate the lifetime at the most critical position. $N_{\mathrm{L}} \in \mathbb{N}$ denotes the number of load cycles until the end of the useful lifetime, $S_{\mathrm{f}} \in \mathbb{R}$ and z are the fatigue strength and fatigue coefficient. Figure 5.18 displays an exemplary SNC applying equivalent stress S_{eq} via N_{L} load cycles.

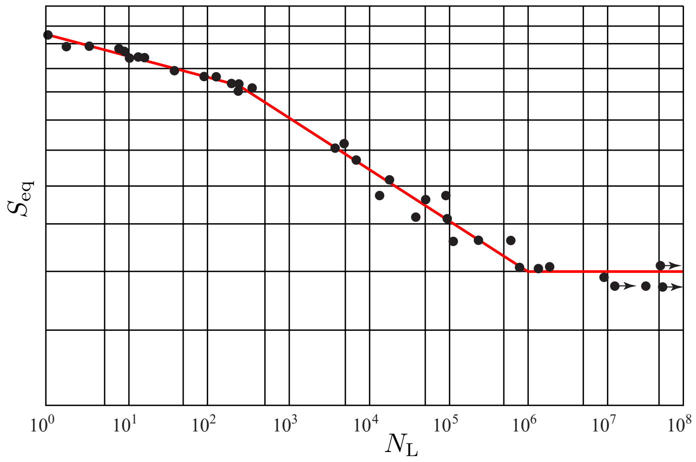


Figure 5.18 Exemplary SNC adapted from [BNS15].

Figure 5.19 visualizes the used simplified motor geometry to investigate the mechanical lifetime of the asynchronous motor. The motor is modeled using different sub-components, such as the housing, the rotor, the stator, the shaft and the bearings. Of course, other sub-components as windings or slip rings are also part of an asynchronous motor but they are left out intentionally due to complexity reduction. Other characteristic like housing fins are simplified due to the same reason. This results in a modeling error d^{m} which cannot be neglected, but results in a small computation time to demonstrate the concept of model design for transferability. Finally, the generic motor model can be described using three static descriptive features C which are the housing material M_{h}, the housing length l_{h} and the rotor length l_{r} .

Besides the static information, the lifetime of an asynchronous motor is influenced by the operating conditions. Electric motors are facing different types of operation anomalies like misalignment or rod fracture. For this case study, parallel misalignment $\delta \in \mathbb{R}$ and angular misalignment $\varphi \in \mathbb{R}$

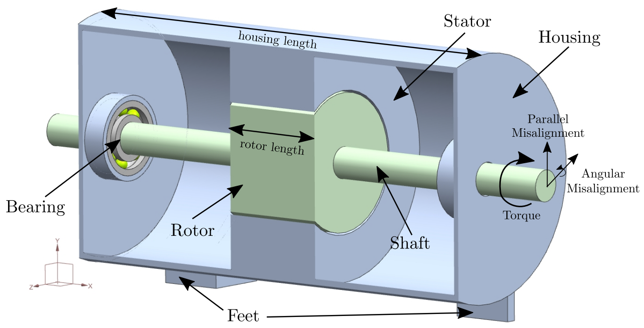


Figure 5.19 Section cut of the simplified motor geometry.

at the driving and non-driving end resulting in wear of the bearings are the considered anomalies. Additionally, the motor torque $T \in \mathbb{R}$ is considered as most influencing operating condition. The motor speed $n \in \mathbb{R}$ is considered by providing the information about the number of load cycles per minute and therefore allows to specify a time duration until the next maintenance is required.
Now, after having introduced the case study background, the MFM design work flow with the purpose of transferability can be applied. Therefore, the first step is the reusage of the design model and interest region definition. For this case study, the interest region is defined as

$$R_\Omega := [T = 1.92\,\mathrm{Nm},\ T = 8.33\,\mathrm{Nm}], \tag{5.14}$$

$$R_\Delta := [\delta = 0.00\,\mathrm{mm},\ \delta = 0.30\,\mathrm{mm}], \tag{5.15}$$

$$R_\Phi := \left[\varphi = 0.00^\circ,\ \varphi = 0.35^\circ\right]. \tag{5.16}$$

More detailed, the interest regions for misalignment are even sub-divided into two separate interest regions A and B because the approximation complexity is different for both interest regions. The interest regions A and B are defined as

$$R_{\Delta}^{\mathrm{A}} := [\delta = 0.20\,\mathrm{mm},\ \delta = 0.30\,\mathrm{mm}]\,, \tag{5.17}$$

$$R_{\Delta}^{\mathrm{B}} := [\delta = 0.00\,\mathrm{mm},\ \delta = 0.20\,\mathrm{mm}]\,, \tag{5.18}$$

$$R_{\Phi}^{\mathrm{A}} := \left[\varphi = 0.25^{\circ},\ \varphi = 0.35^{\circ}\right], \tag{5.19}$$

$$R_{\Phi}^{\mathrm{B}} := \left[\varphi = 0.00^{\circ},\ \varphi = 0.20^{\circ}\right]. \tag{5.20}$$

and their respective contour plots are shown in Figure 5.20 and 5.21.

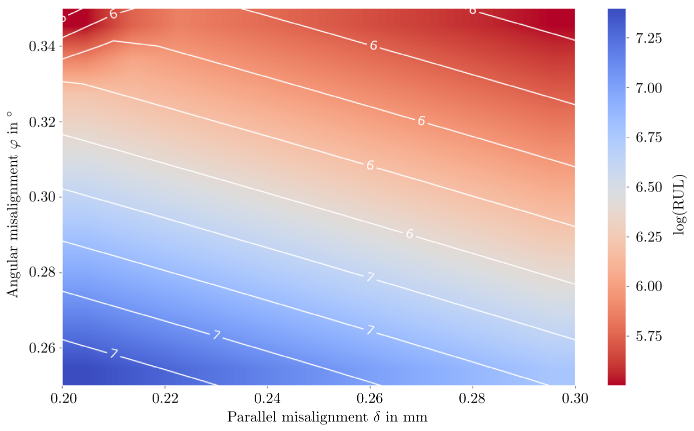


Figure 5.20 Misalignment range A.

Moving forward in the model design work flow for transferability, overall 18 different structural FEM models of parametric electric motors are part of the model stack for the exploitation phase. The 18 different motors are listed in Table 5.3 and the different static descriptive features are formulated as $\mathbf{C}(x) := [M_{\mathrm{h}}\ \ l_{\mathrm{h}}\ \ l_{\mathrm{r}}]^{\mathrm{T}}$. From these models, the models 1, 7, 12 and 18 are selected to be used as LFMs for the MFM composition. In the work of [Ber+19a] it is elicited why especially these models are the best fit to be used for the MFM composition. These four models are order-reduced to be executed during runtime by creating data-driven surrogate models of the FEM models. The model order-reduction approach is not explained more in detail because it is not focus of the industrial case study. The boundary conditions which act as system input $\boldsymbol{u}$ are

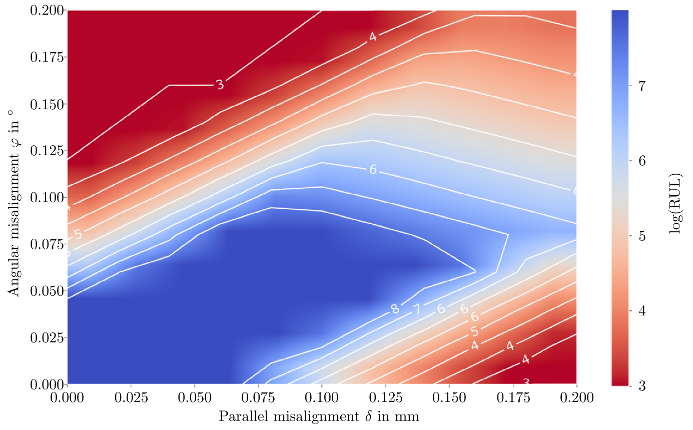


Figure 5.21 Misalignment range B.

defined as $\boldsymbol{u}(t) := [T(t) \;\; \delta(t) \;\; \varphi(t)]^{\mathrm{T}}$. The misalignment features are extracted from the acceleration sensor signals being measured by feeding moving-window observation snapshots into a trained neural network during system operation. Again, the feature extraction approach is not explained more in detail because it is not the focus of the industrial case study.

Since the second step of the MFM design work flow for transferability is completed, the MFM composition can start. As first part, the DoE for the exploitation of similar HFMs has to be specified. Each HFM is exploited for both misalignment range A and B using a full-factorial DoE which is shown in Table 5.4. Overall, eleven different load torques are applied for different anomaly constellations resulting in two times 1331 model evaluations for each of the 18 FEM models resulting in 47,916 simulations. The overall model exploitation represented by the simulation duration for the parametric simplified motor geometry takes about three days. Since the extracted features are already used as boundary conditions for the HFM evaluations and the feature extraction is not part of this case study, the next step of the model design work flow is the combination of the generation of training and test set for the MFM composition. For the case study, the evaluations of ten motors are assigned to the training set T and

Table 5.3 Motor model stack for exploitation phase.

Model	M_h	l_h in mm	l_r in mm
1		165.0	30.0
2		165.0	50.0
3	Aluminum	165.0	70.0
4	Al 5086	195.0	30.0
5		195.0	50.0
6	E_M = 72 GPa	195.0	70.0
7	ν = 0.33	225.0	30.0
8		225.0	50.0
9		225.0	70.0
10		165.0	30.0
11		165.0	50.0
12	Cast Iron	165.0	70.0
13	UNI 5007 Grade 25	195.0	30.0
14		195.0	50.0
15	E_M = 90 GPa	195.0	70.0
16	ν = 0.3	225.0	30.0
17		225.0	50.0
18		225.0	70.0

the evaluations of four motors are assigned to the test set V. Again, it is distinguished between interpolation and extrapolation purposes which are explained more in detail in section 5.3.2. Combined with the four motor models which are used as LFMs, all 18 motor models are used. Similar to the benchmark system for model transferability, the selected fusion function set can be written as $F_\mathrm{f} = \{q[\alpha], h[\theta_h], m^\mathrm{s}[\theta_h]\}$. Finally, the MFM can be assembled by solving the fusion operator optimization problem.

Table 5.4 DoE of model transferability case study.

Parameter	Values in Nm	
Motor Torque	1.92, 2.08, 2.27, 2.50, 2.77, 3.13, 3.57, 4.17, 5.00, 6.25, 8.33	
Parameter	**Parameter Range**	**Step Size**
Parallel Misalignment A	$0.20\,\text{mm} \leq \delta \leq 0.30\,\text{mm}$	0.1
Parallel Misalignment B	$0.00\,\text{mm} \leq \delta \leq 0.20\,\text{mm}$	0.2
Angular Misalignment A	$0.25^\circ \leq \varphi \leq 0.35^\circ$	0.1
Angular Misalignment B	$0.00^\circ \leq \varphi \leq 0.20^\circ$	0.2

5.3.2 Case Study Results

The industrial case study results for the model design for transferability are separated into four different analyses:

1. Misalignment Range A Interpolation,
2. Misalignment Range A Extrapolation,
3. Misalignment Range B Interpolation,
4. Misalignment Range B Extrapolation.

The misalignment ranges have already been introduced in the previous sub-chapter, the inter- and extrapolation differentiation is introduced now. It is relevant to evaluate the performance of the fusion operator for motors with housing and rotor length which are in the range of the training minima and maxima (interpolation) and which are not in the range of the training minima and maxima (extrapolation). Figure 5.22 shows the hypercube that is related to the inter- and extrapolation set. For the interpolation case, the maxima regarding housing and rotor length of the training model stock which are no base motors, so motors 3, 8, 9 and 16, are part of the training set. For the extrapolation case, these motors are exclusively part of the test. Table 5.5 shows the basis, training and test motors for the individual test case.

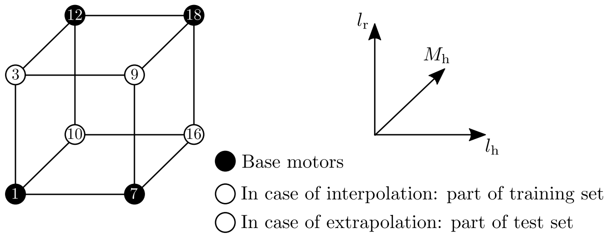

12
18
3
9
10
16
1
7
l_r
M_h
l_h
Base motors
In case of interpolation: part of training set
In case of extrapolation: part of test set

Figure 5.22 Inter- and extrapolation hyper-cube.

Table 5.5 Analysis sets for industrial model transferability case study.

Model Type	Interpolation	Extrapolation
Basis	1, 7, 12, 18	1, 7, 12, 18
Training	2, 3, 4, 6, 9 10, 11, 14, 16, 17	2, 4, 5, 6, 8 11, 13, 14, 15, 17
Test	5, 8, 13, 15	3, 9, 10, 16

The raw RUL data is pre-processed by applying the log-function to the raw RUL value due to its big spread varying between $10^2 - 10^7$ cycles. Consequently, the different methods which are trained to become the fusion operator are trained on the logarithmic RUL values. This does not change the purpose of the industrial case study because the condition monitoring service can still estimate the motor state and decide whether a maintenance shall be scheduled or not. Solving the fusion operator optimization problem delivers the results which are shown in Table 5.6 for misalignment range A and in Table 5.7 for misalignment range B. moSAIc outperforms the other methods in both application cases. All suitable methods perform better on the interpolation case than on the extrapolation case, just QMI does not perform well on the misalignment range A. For this industrial case study, the MLP topology and its training properties have been empirically optimized resulting in a 50-50-50 topology using a ReLu activation function. Providing another quality measure for moSAIc, the Mean absolute percentage deviation (MAPD), being defined as

$$\mathrm{MAPD} := \frac{100\%}{n} \sum_{i-1}^{n} \left| \frac{y_i^{\mathrm{H}} - y_i^{\mathrm{F}}}{y_i^{\mathrm{H}}} \right|, \tag{5.21}$$

for misalignment range B is 10.08% for the interpolation case and 17.72% for the extrapolation case. This provides a suitable uncertainty for the condition monitoring services.

Table 5.6 ED on misalignment range A.

Model	ED Interpolation	ED Extrapolation
Best MLP	136.25	394.01
QMI	$6.53 \cdot 10^5$	$2.87 \cdot 10^5$
moSAIc	6.33	9.56

Table 5.7 ED on misalignment range B.

Model	ED Interpolation	ED Extrapolation
Best MLP	129.44	154.88
QMI	15.76	18.61
moSAIc	10.68	18.41

The outer loop application for this condition monitoring service for electric motors is based on observations and static descriptive features which can be extracted from a data sheet. Now, after having found the most suitable fusion operator, the results are going to be visualized intuitively. Therefore, the following tablet application was developed leveraging the methodology of MFM modeling, the IIoT as well as signal processing algorithms to provide motor operators a simple visualization of their motor's health state. Figure 5.23 visualizes the data flow and processing steps of the condition monitoring service.
The acceleration and current signal cables are connected to an edge device, for example a Programmable logic controller (PLC) with fast input and outputs. The PLC discretizes the analog acceleration and current signals

and extracts features using a neural network based feature extraction algorithm. The required features, torque T and misalignment $\boldsymbol{m}_k = [\delta \; \varphi]^{\mathrm{T}}$, are then cyclically sent to an IIoT platform. The extracted features are read by a simulation server which executes the base model simulations with the observed misalignment and torque and uses the trained fusion operator to predict the RUL $\mathbf{r}_k$. Afterwards, the RUL is read together with the motor's operating conditions by the tablet application. This augmented reality application visualizes the motor health state estimation using a hologram-like representation with a colored shaft to indicate if the motor is healthy or if something is wrong. Figure 5.24 shows an example for a good health state applied to a motor in a laboratory environment. In case of a bad health state, the shaft color will change to yellow or even red. Concluding the industrial case study for model transferability, it was shown that the MFM design work flow is applicable to real industrial challenges with related prerequisites.

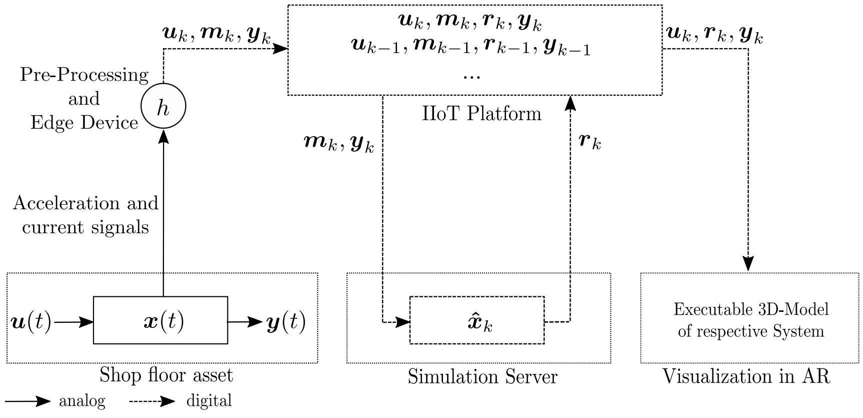

$\boldsymbol{u}_k, \boldsymbol{m}_k, \boldsymbol{y}_k$
$\boldsymbol{u}_k, \boldsymbol{m}_k, \boldsymbol{r}_k, \boldsymbol{y}_k$
$\boldsymbol{u}_{k-1}, \boldsymbol{m}_{k-1}, \boldsymbol{r}_{k-1}, \boldsymbol{y}_{k-1}$
...
$\boldsymbol{u}_k, \boldsymbol{r}_k, \boldsymbol{y}_k$
Pre-Processing and Edge Device
h
IIoT Platform
$\boldsymbol{m}_k, \boldsymbol{y}_k$
$\boldsymbol{r}_k$
Acceleration and current signals
$\boldsymbol{u}(t)$
$\boldsymbol{x}(t)$
$\boldsymbol{y}(t)$
$\hat{\boldsymbol{x}}_k$
Executable 3D-Model of respective System
Shop floor asset
Simulation Server
Visualization in AR
analog
digital

Figure 5.23 IIoT infrastructure including data flow.

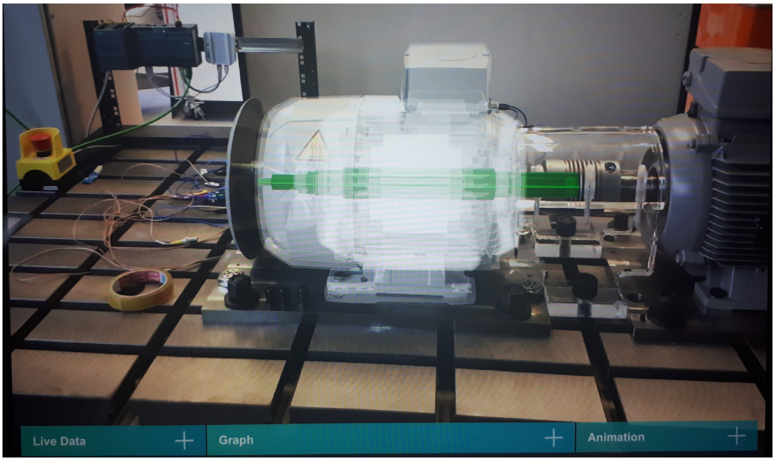


Figure 5.24 Motor health state holograms.

6 Conclusion

6.1 Summary

Concluding this thesis, the main achievements are the development work flow of a digital performance twin based on models from different PLM life cycle phases as well as a multi-fidelity optimization method to construct a digital performance twin. The novelty of this method is leveraging the concept of multi-fidelity modeling to fuse observations, low-fidelity models and high-fidelity models. For the resulting concept of multi-model data fusion, the different modeling approaches with the purpose of dimension increase, accuracy enhancement and transferability have been evaluated. Therefore, the fusion methods QMI, MLP, *moSAIc* and *Co-Kriging* have been selected to be used as fusion operator for the evaluation phase. Hereby, the development of *moSAIc* was a main contribution of this work. With this method it is possible to transfer available knowledge for modeled systems to predict system outputs of unknown similar systems without the necessity of modeling them.

The developed multi-fidelity modeling work flow has been applied to both benchmark systems and industrial case studies. The most interesting observation is its overall resilient applicability although the modeling purposes are significantly different. It shows that the combination of acquired real system observations and simulated low- and high-fidelity models result in a model which is better than its individual components, either in terms of accuracy, dimensionality or transfer.

For every modeling approach, the most suitable fusion method differs due to the individual characteristics of the methods. In the context of multi-model data fusion, the methods QMI and MLP perform well on use cases focusing on dimensionality increase which is soft-sensoring in the context of predictive maintenance. The method *Co-Kriging* performs best on use cases focusing on accuracy enhancement, whereas *moSAIc* outperforms all other methods in the context of model transferability.

Three major remarks have to be pointed out regarding the construction of multi-fidelity models using the multi-model data fusion approach and its applicability for the different use cases:

- **Model accuracy**: The aimed accuracy of the digital performance twin highly depends on the quality of the models being used to compose it. Therefore, the engineering knowledge and expertise of system experts will still be highly required and cannot be replaced by purely data-driven approaches.

- **Data quality**: The data which is used for the training of the digital performance twin needs to have a high data quality in terms of time-synchronization, location registration, and observation consistency. Missing or faulty selected data points lead to a wrong training data base which can lead to a bad model accuracy and therefore model unusability.

- **Purpose**: Digital performance twins should only be developed if they lead to a significant added value like increased predictive maintenance accuracy for example. Developing them just for the fact of having them available without an application is not the reason why they should exist.

Referring to the beginning of this thesis, the digitization is a progress which leads to a higher performance transparency and efficiency of industrial systems. In this work it is shown that digital performance twins, sustainably constructed out of already available simulation models and easily acquirable observations, pay off on this digitization. The use case of motor fleet condition monitoring already shows a first methodology transfer into an industrial service. In the context of industrial digitization, it still is and will be even more important to ensure the operationalizability of theoretical concepts to increase the trust into technology. In my opinion, the results of this work help to increase this trust by transferring theoretical concepts into measurable added value.

6.2 Outlook

In terms of further research in the field of multi-fidelity modeling of dynamic systems for operation-parallel simulation, many open topics remain. One of these possible research activities is related to the state of the dynamic system for the accuracy enhancement modeling purpose. The range of possible use cases can be drastically increased by considering transient phases of dynamic systems additionally to the already described stationary phases. Therefore, methods including a memory component have to be tested to be used as fusion operator. Especially the integration of feature-based methods like the Kalman filter or particle filter into the concept of multi-model data fusion offer a lot of research potential.

Another methodology which could be considered as fusion operator is fuzzy logic. Although the work of [Ram18] has shown, that fuzzy logic was applicable for the specific modeling purpose of dimensionality increase, the method could perform better on other modeling purposes, especially on transferability. Additionally, the evaluation of neuro-fuzzy methods being used as fusion operator is an interesting future research possibility.

The third possible future research activity lies in the field of adapting or extending the optimization algorithm to find the most suitable fusion operator regarding purpose-specific measure selection. The usage of the euclidean distance does not necessarily have to be the smartest measure to determine the fusion operator. Approaches of including additional stochastic moments as features may result in a more robust selection of the fusion operator.

As last possible future research activity definitely more industrial case studies have to be conducted to underline the applicability of the developed multi-fidelity modeling work flow. Just the measurable added value gained by using the multi-fidelity modeling work flow will verify its applicability. Possible industrial use cases are the consideration of both similar systems which are driven by an electric motor like compressors and totally different systems not related to electric motors.

Appendix

Bibliography

[Abd+15] **Abdallah, I. et al.** *Fusing Simulation Results from Multifidelity Aero-servo-elastic Simulators*. In: *12th International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP)*. Vancouver: The University of British Columbia, 2015.

[Aro16] **Arora, Jasbir S.** *Introduction To Optimum Design*. London, UK: Academic Press is an imprint of Elsevier, 2016.

[ALH97] **Ashraf, Muhammad, Loftis, Jim C., and Hubbard, K. G.** *Application of geostatistics to evaluate partial weather station networks*. In: *Agricultural and Forest Meteorology* 84.3-4 (1997), pp. 255–271.

[Bab+16] **Babaee, H. et al.** *Multi-fidelity Modelling of Mixed Convection Based On Experimental Correlations and Numerical simulations*. In: *Journal of Fluid Mechanics* 809 (2016), pp. 895–917.

[BGW15] **Benner, P., Gugercin, S., and Willcox, K.** *A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems*. In: *SIAM Review* 57.4 (2015), pp. 483–531.

[Böh15] **Böhnke, D.** *A Multi-Fidelity Workflow to Derive Physics-Based Conceptual Design Methods*. Dissertation. Hamburg: Technical University Hamburg-Harburg, 2015.

[BRH18] **Boschert, S., Rosen, R., and Heinrich, C.** *Next Generation Digital Twin*. In: *12th International Symposium on Tools and Methods of Competitive Engineering (TMCE)*. Delft: Delft University of Technology, 2018, pp. 209–218.

[BNS15] **Budynas, Richard Gordon, Nisbett, J. Keith, and Shigley, Joseph Edward**. *Shigley's mechanical engineering design*. Tenth edition. Mcgraw-Hill series in mechanical engineering. New York, NY: McGraw-Hill Education, 2015.

[Bun+13] **Bungartz, H.-J. et al.** *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. 2nd ed. Berlin and Heidelberg: Springer Spektrum, 2013.

[BM15] **Burger, Eric M. and Moura, Scott J.** *Gated ensemble learning method for demand-side electricity load forecasting*. In: *Energy and Buildings* 109 (2015), pp. 23–34.

[CLK14] **Choi, S. H., Lee, S. J., and Kim, T. G.** *Multi-fidelity modeling & simulation methodology for simulation speed up*. In: *2nd ACM SIGSIM/PADS Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS)*. New York: ACM Press, 2014, pp. 139–150.

[Eur18] **European Committee for Standardization**. *EN 13306: Maintenance - Maintenance terminology*. Berlin, 2018.

[Eva08] **Evans, L. C.** *Partial differential equations*. Providence, RI: American Math. Soc, 2008.

[FY98] **Fatemi, A. and Yang, L.** *Cumulative fatigue damage and life prediction theories: a survey of the state of the art for homogeneous materials*. In: *International journal of fatigue* 20.1 (1998), pp. 9–34.

[Fer+16] **Fernández-Godino, M. G. et al.** *Review of Multi-Fidelity Models*. In: *arXiv preprint arXiv:1609.07196* (2016).

[FB09] **Frey, T. and Bossert, M.** *Signal- und Systemtheorie: Mit 26 Tabellen, 64 Aufgaben mit Lösungen und 84 Beispielen*. 2nd ed. Wiesbaden: Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 2009.

[Gar19] **Gartner, Inc.** *Top 10 Strategic Technology Trends for 2019*. 2019.

[GS12] **Glaessgen, E. H. and Stargel, D. S.** *The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles*. In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 2012: Special Session on the Digital Twin*. Red Hook: Curran, 2012.

[GS18] **Glassey, J. and Stosch, M. von**. *Hybrid Modeling in Process Industries*. 1st ed. Milton: CRC Press, 2018.

[GMZ16] **Gouriveau, R., Medjaher, K., and Zerhouni, N.** *From Prognosis and Health Systems Management to Predictive Maintenancene: Monitoring and Prognostics*. 4th ed. Hoboken, NJ: John Wiley et Sons, Inc, 2016.

[Hai06] **Haibach, Erwin**. *Betriebsfestigkeit: Verfahren und Daten zur Bauteilberechnung*. Berlin: Springer, 2006.

[HLF17] **Hamilton, F., Lloyd, A. L., and Flores, K. B.** *Hybrid Modeling and Prediction of Dynamical Systems*. In: *PLoS computational biology* 13.7 (2017), pp. 1–20.

[HP16] **Heizmann, M. and Puente León, F.** *Modellbildung in der Mess- und Automatisierungstechnik*. In: *tm - Technisches Messen* 83.2 (2016), pp. 63–65.

[Him72] **Himmelblau, David Mautner**. *Applied non-linear programming*. New York: McGraw-Hill, 1972.

[Hof13] **Hofleitner, A.** *A hybrid approach of phyiscal laws and data-driven modeling for estimation: the example of queuing networks*. Dissertation. Berkeley: University of California, 2013.

[Hon+19] **Hong, Jihoon et al.** *Remaining useful life prediction using time-frequency feature and multiple recurrent neural networks*. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation*. Piscataway: IEEE, 2019.

[HYW10] **Hu, Chao, Youn, Byeng, and Wang, Pingfeng**. *Ensemble of datadriven prognostic algorithms with weight optimization and k-fold cross validation*. In: *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. New York: American Society of Mechanical Engineering, 2010.

[Hua+15] **Huang, E. et al.** *Multi-Fidelity Model Integration for Engineering Design*. In: *Procedia Computer Science* 44 (2015), pp. 336–344.

[Int14] **International Electrotechnical Commission**. *IEC 60050-351: International Electrotechnical Vocabulary - Part 351: Control Technology*. Berlin, 2014.

[Jam15] **Jamshidnezhad, Mohammad**. *Experimental Design In Petroleum Reservoir Studies*. Oxford, UK and s.l.: Elsevier, 2015.

[KGS09] **Kadlec, P., Gabrys, B., and Strandt, S.** *Data-driven Soft Sensors in the process industry*. In: *Computers & Chemical Engineering* 33.4 (2009), pp. 795–814.

[KGG11] **Kadlec, P., Grbić, R., and Gabrys, B.** *Review of adaptation mechanisms for data-driven soft sensors*. In: *Computers & Chemical Engineering* 35.1 (2011), pp. 1–24.

[KST10] **Kain, S., Schiller, F., and Trenner, T.** *Monitoring and Diagnostics Based On Synchronous Simulation Methods*. In: *Integrationsaspekte der Simulation: Technik, Organisation und Personal* (2010), pp. 445–452.

[Kap11] **Kapp, R.** *A factory simulation system for the integrated support of the continuous plant adaptation during operation*. Dissertation. Stuttgart: University of Stuttgart, 2011.

[Ken00] **Kennedy, M.** *Predicting the output from a complex computer code when fast approximations are available*. In: *Biometrika* 87.1 (2000), pp. 1–13.

[Kic17] **Kicsiny, R.** *Grey-Box Model for Pipe Temperature Based On Linear Regression*. In: *International Journal of Heat and Mass Transfer* 107 (2017), pp. 13–20.

[Kro16] **Kroll, A.** *Computational Intelligence: Methoden und technische Anwendungen*. 2nd ed. München: De Gruyter Oldenbourg, 2016.

[KRS11] **Kroschel, K., Rigoll, G., and Schuller, B.** *Statistische Informationstechnik: Signal- und Mustererkennung, Parameter- und Signalschätzung*. 5th ed. Berlin and Heidelberg: Springer, 2011.

[Kru+15] **Kruse, R. et al.** *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. 2nd ed. Wiesbaden: Springer Fachmedien Wiesbaden, 2015.

[Kub17] **Kubat, M.** *An Introduction to Machine Learning*. 2nd ed. Cham: Springer International Publishing, Imprint, and Springer, 2017.

[Küh13] **Kühnert, Christian**. *Data-driven methods for fault localization in process technology*. Dissertation. Karlsruhe: Karlsruhe Institute of Technology, 2013.

[Lee05] **Lee, Yung-Li**. *Fatigue testing and analysis: Theory and practice*. Amsterdam: Elsevier/Butterworth-Heinemann, 2005.

[Lei+18] **Lei, Yaguo et al.** *Machinery health prognostics: A systematic review from data acquisition to RUL prediction*. In: *Mechanical Systems and Signal Processing* 104 (2018), pp. 799–834.

[Lep+20] **Lepenioti, K. et al.** *Prescriptive Analytics: Literature Review and Research Challenges*. In: *International Journal of Information Management* 50 (2020), pp. 57–70.

[LW11] **Li, J. and Wang, H.** *Data Fusion of Multi-fidelity Model and its Application in low-speed reflexed Airfoil Shape Optimization*. In: *2011 IEEE 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*. Piscataway: IEEE, 2011, pp. 2910–2913.

[Liu09] **Liu, Jane W. S.** *Real-time systems*. Upper Saddle River, NJ: Prentice Hall, 2009.

[Mon91] **Montgomery, Douglas C.** *Design and analysis of experiments*. New York: Wiley, 1991.

[NW18] **Nazmus, Sakib and Wuest, Thorsten**. *Challenges and Opportunities of Condition-based Predictive Maintenance: A Review*. In: *Procedia CIRP* 78 (2018), pp. 267–272.

[Ném18] **Németh-Csóka, M.** *Thermisches Management elektrischer Maschinen: Messung, Modell und Energieoptimierung*. 1st ed. Wiesbaden: Springer Fachmedien Wiesbaden, 2018.

[Oko+14] **Okoh, C. et al.** *Overview of Remaining Useful Life Prediction Techniques in Through-life Engineering Services*. In: *Procedia CIRP* 16 (2014), pp. 158–163.

[PWG18] **Peherstorfer, B., Willcox, K., and Gunzburger, M.** *Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*. In: *SIAM Review* 60.3 (2018), pp. 550–591.

[PVK16] **Perdikaris, P., Venturi, D., and Karniadakis, G. E.** *Multifidelity Information Fusion Algorithms for High-Dimensional Systems and Massive Data Sets*. In: *SIAM Journal on Scientific Computing* 38.4 (2016), pp. 521–538.

[PG15] **Pohlandt, C. and Geimer, M.** *Thermische Modelle elektrischer Antriebsmaschinen unter dynamischen Lastanforderungen*. In: *Landtechnik* 70(4) (2015), pp. 97–112.

[Pue15] **Puente León, F.** *Messtechnik: Systemtheorie für Ingenieure und Informatiker*. 10th ed. Berlin and Heidelberg: Springer, 2015.

[Qi+18] **Qi, Q. et al.** *Digital Twin Service Towards Smart Manufacturing*. In: *Procedia CIRP* 72 (2018), pp. 237–242.

[RK16] **Raissi, M. and Karniadakis**. *Deep Multi-fidelity Gaussian Processes*. In: *arXiv preprint arXiv:1604.07484* (2016).

[RW06] **Rasmussen, C. E. and Williams, C. K. I.** *Gaussian processes for machine learning*. 1st ed. Cambridge and London: MIT Press, 2006.

[Rie+19] **Riesener, M. et al.** *The Digital Shadow as Enabler for Data Analytics in Product Life Cycle Management*. In: *Procedia CIRP* 80 (2019), pp. 729–734.

[Rod12] **Roddeck, W.** *Einführung in die Mechatronik*. 4th ed. Wiesbaden: Springer Vieweg, 2012.

[RP06] **Ruser, H. and Puente León, F.** *Methoden der Informationsfusion - Überblick und Taxonomie*. In: *Informationsfusion in der Mess- und Sensortechnik*. Ed. by **Beyerer, J., Puente León, F., and Sommer, K.-D.** Karlsruhe: Universitätsverlag, 2006, pp. 1–20.

[Sai91] **Saint Raymond, X.** *Elementary Introduction to the Theory of Pseudodifferential Operators*. 1st ed. Boca Raton and Bosa Roca: CRC Press, 1991.

[San+16] **Santecchia, E. et al.** *A Review on Fatigue Life Prediction Methods for Metals*. In: *Advances in Materials Science and Engineering* 2016.2 (2016), pp. 1–26.

[Sch+17] **Schleich, B. et al.** *Shaping the Digital Twin For Design and Production Rngineering*. In: *CIRP Annuals - Manufacturing Technology* 66.1 (2017), pp. 141–144.

[Sch+18] **Schluse, M. et al.** *Experimentable Digital Twins - Streamlining Simulation-Based Systems Engineering for Industry 4.0*. In: *IEEE Transactions on Industrial Informatics* 14.4 (2018), pp. 1722–1731.

[SHH17] **Schwab, S., Holzmüller, B., and Hohmann, S.** *Automated Verification of Switched Systems Using Hybrid Identification*. In: *Cyber Physical Systems. Design, Modeling, and Evaluation*. Ed. by **Berger, C., Mousavi, M. R., and Wisniewski, R.** Cham: Springer International Publishing, 2017, pp. 87–100.

[SAM16] **Souza, F.., Araújo, R., and Mendes, J.** *Review of soft sensor methods for regression applications*. In: *Chemometrics and Intelligent Laboratory Systems* 152 (2016), pp. 69–79.

[Sto+14] **Stosch, M. von et al.** *Hybrid Semi-Parametric Modeling in Process Systems Engineering: Past, Present and Future*. In: *Computers and Chemical Engineering* 60 (2014), pp. 86–101.

[UB16] **Unbehauen, H. and Bohn, C.** *Identifikation dynamischer Systeme: Methoden zur Modellbildung anhand von Messungen*. 1st ed. Wiesbaden: Springer Vieweg, 2016.

[ZZT13] **Zienkiewicz, O. C., Zhu, J. Z., and Taylor, R. L.** *The finite element method: Its basis and fundamentals*. Seventh edition. Oxford, UK: Butterworth-Heinemann, 2013.

List of Publications

[BH18] **Bergs, C. and Heizmann, M.** *Kombination unterschiedlicher Modellierungsansätze für die betriebsbegleitende Simulation industrieller Prozesse*. In: *15. Fachtagung EKA - Entwurf komplexer Automatisierungssysteme*. Magdeburg: Institut für Automation und Kommunikation e.V., 2018.

[BH19a] **Bergs, C. and Heizmann, M.** *Kombination unterschiedlicher Modellierungsansätze für die betriebsbegleitende Simulation industrieller Prozesse*. In: *at - Automatisierungstechnik* 67.3 (2019), pp. 183–192.

[BHH18] **Bergs, C., Heizmann, M., and Held, H.** *Hybrid modeling approaches with a view to model output prediction for industrial applications*. In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. Piscataway: IEEE, 2018, pp. 258–263.

[BH19b] **Bergs, C. and Held, H.** *Optimale Ansteuerung einer energieflexiblen Produktionsinfrastruktur*. In: *Zeitschrift für wirtschaftlichen Fabrikbetrieb (ZWF)* 114.1-2 (2019), pp. 16–19.

[Ber+19a] **Bergs, C. et al.** *Health Indication of Electric Motors Using a Hybrid Modeling Approach*. In: *tm - Technisches Messen* (2019).

[Ber+19b] **Bergs, C. et al.** *Novel method for online wear estimation of centrifugal pumps using multi-fidelity modeling*. In: *2019 IEEE 2nd International Conference on Industrial Cyber-Physical Systems (ICPS)*. Piscataway: IEEE, 2019, pp. 185–190.

[Hil+19] **Hildebrandt, M. et al.** *Remaining Useful Life Estimation for Unknown Motors Using a Hybrid Modeling Approach*. In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Piscataway: IEEE, 2019.

[Kha+19] **Khalil, Mohamed et al.** *IIoT-based Fatigue Life Indication Using Augmented Reality*. In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Piscataway: IEEE, 2019.

[Pre+18] **Prell, B. et al.** *Simulation von Steuerungsmaßnahmen zum energieflexiblen Fabrikbetrieb*. In: *12. Tagung VPP2018: Smarte Produktion und digitale Vernetzung*. Chemnitz: Fraunhofer IWU, 2018, pp. 79–88.

[Tur+19] **Tur, B. et al.** *Modelling of energy storage devices and converters for energy flow simulation in Plant Simulation*. In: *18. ASIM Fachtagung Simulation in Produktion und Logistik*. Chemnitz: Fraunhofer IWU, 2019.

List of Supervised Theses

[Ram18] **Ramirez Flores, D. A.** *Analysis of Diverse Methods for the Fusion of Theoretical and Data-driven Models in the Context of an Electric Induction Motor*. Bachelor Thesis. Karlsruhe: Karlsruhe Institute of Technology, 2018.

[Tur18] **Tur, B.** *Identifikation, Simulation und Prädikation energetischer Bedarfsprofile für die Produktion und Produktionsinfrastruktur*. Master Thesis. Nürnberg: Friedrich-Alexander-University Erlangen-Nürnberg, 2018.