Julian Belz

Fighting the Curse of Dimensionality with Local Model Networks

Dissertation

Schriftenreihe der Arbeitsgruppe
Mess- und Regelungstechnik – Mechatronik
Department Maschinenbau

Herausgeber Oliver Nelles

UNIVERSITÄT SIEGEN

Band 3

Impressum
Prof. Dr.-Ing. Oliver Nelles
Arbeitsgruppe Mess- und Regelungstechnik
Department Maschinenbau
Universität Siegen
57068 Siegen
ISSN 2193-0538
URN urn:nbn:de:hbz:467-14524
Zugl.: Siegen, Univ., Diss., 2018

Fighting the Curse of Dimensionality with Local Model Networks

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von
Dipl.-Ing. Julian Belz
aus Siegen

genehmigt durch die Naturwissenschaftlich-Technische Fakultät
der Universität Siegen
Siegen 2018

Betreuer und erster Gutachter
Prof. Dr.-Ing. Oliver Nelles
Universität Siegen

Zweiter Gutachter
Prof. Dr.-Ing. Thomas Carolus
Universität Siegen

Tag der mündlichen Prüfung
21. November 2018

Acknowledgments

This thesis was written during my time as a research assistant at the Institute of Mechanics and Control Engineering - Mechatronics of the University of Siegen. First and foremost I would like to thank my Ph.D. advisor Prof. Dr.-Ing. Oliver Nelles for his extremely valuable and generous support in carrying out this work, the willingness to discuss and the many motivating suggestions. The pleasant working atmosphere created by his leadership significantly contributed to the success of my work.

I would like to thank Prof. Dr.-Ing. Thomas Carolus for his interest in my work and the friendly acceptance of the role as additional reviewer for my Ph.D. thesis.

I am also very grateful to my former colleagues for the great time together, many fruitful discussions, their support, and their contribution to the very pleasant working atmosphere. In particular, I thank Diana Klein, Dorina Weichert, Geritt Kampmann, Tim Oliver Heinz, Tobias Münker, Benjamin Hartmann, Tobias Ebert, Oliver Bänfer, Hans-Werner Haupt and Torsten Fischer.

I highly appreciated the cooperation with colleagues from other Institutes of the University of Siegen, from the University of Klagenfurt, the Graz University of Technology and industrial partners from Bosch Engineering GmbH, Honda R&D Europe, and Daimler AG. Therefore, many thanks to Prof. Dr.-Ing. Thomas Carolus, Prof. Dr.-Ing. Horst Idelberger, Prof. Dr.-Ing. Martin Horn, Konrad Bamberger, Michael Horwath, Jakob Rehrl, Daniel Schwingshackl, Jana Schmidt, Mark Schillinger, Michael Fischer, Philipp Klein, and Frank Kirschbaum.

I would also like to thank all the students that supported me during my time at the University of Siegen, especially Laura Winkel, Tim Decker, Timm Julian Peter, and Thomas Levenig.

Last but not least, I would like to thank my whole family, my wife Sarah, and my friends for their love and unconditional support.

Betzdorf, November 2018
Julian Belz

Contents

Symbols and Abbreviations

Latin Symbols

b	Offset of an affine model
c	Coefficients used for the construction of random functions
D	Outer fan diameter
J	Objective function value (optimization)
k	Discrete time
$\mathcal{L}$	Likelihood function
m	Number of input filters
m_a	Slope of an affine model
M	Number of local models
M_{poly}	Number of terms used for the construction of random functions
n_s	Number of system states
n	System order
n_θ	Number of model parameters
n_{eff}	Number of effective parameters
n_p	Number of candidate inputs (cadinality of $\mathbb{P}$)
$n_{eff,i}$	Effective number of parameters of local model i
n_q	Number of requested query points
$n_{qLM,i}$	Number of requested query points per local model i
n_c	Number of optimization constraints
n_R	Rotational speed
N	Number of samples
p	Number of physical inputs
p_z	Number of physical inputs in the premise or $\underline{z}$-input space
p_x	Number of physical inputs in the consequent or $\underline{x}$-input space
$\mathbb{P}$	Set of all potential model inputs

P_G	Generalization performance
P_{shaft}	Shaft power
q^{-1}	Time shifting operator
$\underline{Q}_i$	Validity matrix of the i-th local model
$\underline{S}$	Smoothing matrix
$\underline{u}$	System or model inputs
v	Sigmoid splitting parameter
$\dot{V}$	Specific volume flow rate
$\underline{x}_s$	States of a dynamic system
$\underline{x}$	Local model inputs
y	Measured system output
$\hat{y}$	Model output
$\hat{y}_i$	Local model output
$\underline{z}$	Validity function inputs

Greek Symbols

α	Saturation parameter for sigmoidally saturated polynomials originating from the function generator
γ	Random function generator exponents
δ	Specific fan diameter
Δp	Pressure rise
ζ	Importance value for an input in the $\underline{z}$-input space
η	Efficiency
$\underline{\theta}$	Estimated model parameters
κ	Determines the steepness of the validity functions in the transition between two neighboring local models
λ	Regularization parameter ($\lambda \geq 0$)
μ	Expected value of the exponential distribution from which exponents of the function generator are drawn
ρ	Relevance factor for an input in the $\underline{z}$-input space
ρ_f	Fluid density
σ_n	Standard deviation; if squared σ^2 it is called variance

σ	Specific fan speed
$\underline{\varphi}$	Dynamic net inputs (filtered physical inputs and outputs)
Φ	Validity function
Ψ	Sigmoid function
$\widetilde{\Psi}$	Complementary sigmoid function
ψ	Specific pressure rise

Abbreviations

AIC_c	Corrected version of Akaike's information criterion
AMF	Air-mass flow
ANOVA	Analysis of variance
APRBS	Amplitude modulated random binary signal
BE	Backward elimination
BGS	Biggest gap sequence
CD	Combined design
CFD	Computational fluid dynamics
CV	Cross-validation
DoE	Design of experiments
DWC	Designs without corners
ECU	Engine control unit
EDLS	Extended deterministic local search
ES	Exhaustive search
FEM	Finite element method
FIR	Finite impulse response
FS	Forward selection
GA	Genetic algorithm
HILOMOT	HIerarchical LOcal MOdel Tree
HilomotDoE	HILOMOT for design of experiments
HVAC	Heating, ventilating, and air conditioning
IKMS	Intelligent k-means sequence
IQR	Interquartile range
LASSO	Least absolute shrinkage and selection operator
LH	Latin Hypercube
LMN	Local model network

LOLIMOT	LOcal LInear MOdel Tree
LOO	Leave-one-out
LS	Least squares
MAP	Intake manifold air pressure
MBDO	Metamodel-based design optimization
MDS	Median distance sequence
MIMO	Multiple-input multiple-output
MISO	Multiple input single output
MPG	Miles per gallon
MWEIS	Mixed wrapper-embedded input selection
NN	Nearest neighbor
NRBF	Normalized radial basis function
NRMSE	Normalized root mean squared error
PDI	Partial dependence input
RBIS	Regularization-based input selection
SISO	Single-input single-output
SNR	Signal-to-noise ratio
TP1	Test process one
TP2	Test process two
TP3	Test process three
TP4	Test process four
VIM	Variable intake manifold

Kurzfassung

Das Themengebiet der vorliegenden Arbeit ist die datenbasierte Modellbildung (Identifikation). Das Hauptaugenmerk liegt auf Verfahren der Eingangsselektion und der Versuchsplanung, die dazu dienen, Effekte des Fluchs der Dimensionlität abzuschwächen, indem sie spezielle Eigenschaften lokaler Modellnetze ausnutzen.

Der Modelltyp der lokalen Modellnetze ermöglicht die Aufteilung des Eingangsraums in die sogenannten Regelprämissen und -konklusionen. In dieser Arbeit entwickelte Verfahren im Bereich der Eingangsselektion nutzen diese einzigartige Eigenschaft lokaler Modellnetze aus. Potentielle Eingangsgrößen können individuell den Regelprämissen oder -konklusionen zugeteilt werden. Es wird gezeigt, dass sich ein besserer Bias/Varianz-Kompromiss durch die Ausnutzung dieser individuellen Zuordnung finden lässt. Des Weiteren eröffnen sich zusätzliche Interpretationsmöglichkeiten, da einer der beiden Eingangsräume direkt mit den Modellnichtlinearitäten verknüpft ist.

Der Beitrag der vorliegenden Arbeit im Bereich der Versuchsplanung betrifft sowohl aktive als auch passive Lernverfahren. Im Rahmen der passiven Lernverfahren werden Empfehlungen auf die folgenden drei Fragestellungen gegeben. Sollten (alle) Ecken im Versuchsraum vermessen werden? Welcher raumfüllende Versuchsplan führt voraussichtlich zur höchsten Modellgüte? Welche Reihenfolge sollte beim Vermessen eines Versuchsplans eingehalten werden? Der Beitrag im Bereich aktiver Lernverfahren umfasst eine Erweiterung des bereits bestehenden Algorithmus „Hierarchical Local Model Tree for Design of Experiments“ (HilomotDoE). Drei Schlüsselaspekte der experimentellen Modellbildung werden dabei adressiert: (I) Optimalität, (II) geringer Bias- und (III) Varianzfehler.

Die entwickelten Methoden zeigen anhand ausgewählter Anwendungsbeispiele ihre Stärken. Dazu zählen: Die Vorhersage des Verbrauchs von Kraftfahrzeugen, die Modellierung des Luftmassenstroms für Verbrennungsmotoren, die Metamodellierung

für Ventilatoren und die Erzeugung eines dynamischen Modells zur Regelung eines Heizungs-, Lüftungs- und Klimatisierungssystems.

Abstract

This thesis is settled in the field of data-based modeling (identification) and specifically focuses on the weakening of the effects of the *curse of dimensionality* with local model networks (LMNs). The methods for fighting the curse of dimensionality originate from the fields of input selection and design of experiments (DoE).

The model type of LMNs allows the distinction in two input spaces - the rule premises input space and the rule consequents input space. The developed input selection techniques exploit this unique property of LMNs and the possibility to assign potential inputs to each input space individually. It is shown that this additional freedom enables input selection methods using LMNs to find a better bias/variance trade-off. Furthermore, one of the two arising input spaces is directly connected to the nonlinear effects of the model, which allows for more detailed interpretations.

The DoE contributions of this thesis concern passive and active learning schemes. Recommendations for passive experimental designs are given based on extensive simulation studies with the help of a function generator. The following three questions are addressed. Should corners (vertices) be measured? Which space-filling experimental design is likely to yield the best model accuracy? What order of experimentation leads to the best model quality in early stages of the measurement process? Eventually, the contribution regarding active learning is an extension of the already existing hierarchical local model tree for design of experiments (Hilomot-DoE), addressing three important issues of modeling simultaneously: (I) optimality, (II) model bias, and (III) model variance.

All developed methods demonstrate their abilities on selected application examples, including the prediction of the fuel consumption of cars, the data-based modeling of the air-mass flow of combustion engines, the metamodeling of fans, and the generation of a dynamic model of a heating, ventilating, and air conditioning system for control design.

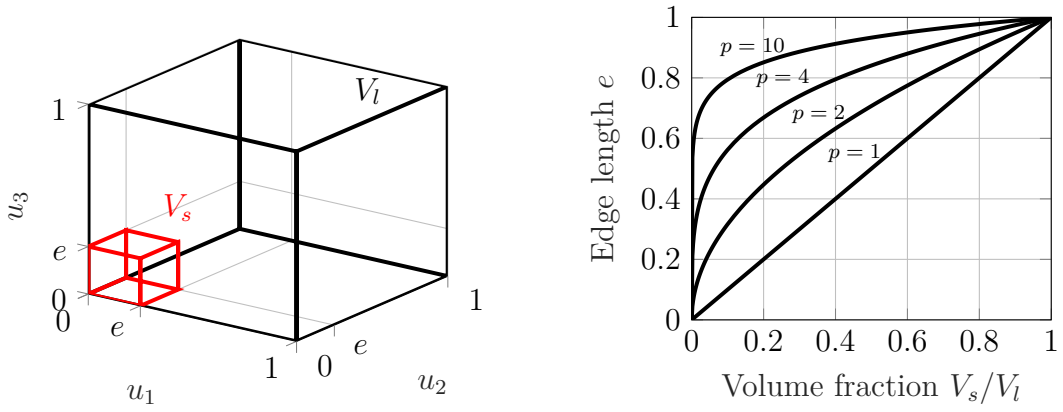
1 Introduction

An increasing amount of modern development methods are based on models. In the context of this thesis, a model can be seen as a mathematical description of a real-world process and can be utilized to predict the output of a technical system given the input values or sequences. Models can save time and resources since extensive investigations can be done on totally different time-scales compared to real-world measurements or applications. Besides the time-saving aspect, there are some advanced techniques in the field of optimization, design of controllers, supervision and many other fields that would hardly be possible without models. There are basically two ways to obtain models describing the technical system of interest:

- Using first principles to deduce a model. Insights from physical, chemical, biological, economical, etc. laws are needed in order to create so-called white-box models. Here, the term „white“ indicates the usage of available knowledge to obtain the models.

- Relying solely on measured data to derive a model. Since for the model derivation only experimental data is used, this approach is called experimental modeling. Other commonly used names are black-box modeling or system identification [96].

This thesis is settled in the field of experimental modeling and specifically concentrates on the weakening of the effects of the *curse of dimensionality* with the help of local model networks (LMNs). The phrase *curse of dimensionality* describes the exponential increase in effort with an increasing input space dimensionality. For example, the number of supporting points upon a grid grows exponentially with the input space dimensionality. As a result the required amount of data, the required storage, and eventually the required computation time increases exponentially. Figure 1.1 illustrates the curse of dimensionality exemplarily with the help of the ratio of volumes of a small (V_s) and a large (V_l) hypercube. The edge length e belongs to the smaller hypercube and is just a fraction of the edge length of the larger hypercube.



(a) (b)

Figure 1.1: Illustration of the curse of dimensionality. (a) The smaller hypercube with edge length e is shown inside the larger one. (b) The needed edge length to meet a specific volume fraction V_s/V_l is shown for several input dimensionalities p.

e has to increase more rapidly with an increasing input dimensionality p in order to meet a specific volume fraction V_s/V_l. For example, the necessary edge length e to reach a volume fraction of $V_s/V_l = 0.1$ is 0.1 for an input space dimensionality of $p = 1$, around 0.3 for $p = 2$, and approximately 0.8 for $p = 10$.

As a result, the data necessary to cover the whole volume equally dense grows exponentially with an increasing input space dimensionality. Luckily, in practice, the curse of dimensionality is typically much less severe due to the following properties that are often fulfilled for real-world problems:

- Some regions of the input space might be non-reachable. Due to physical restrictions or user-specified requirements not all combinations of input values are feasible or desired, respectively [15].
- Real-world problems often fulfill smoothness properties [15]. Small changes in the input variables usually cause only small changes in the target variables, leading to at least locally predictable behavior [96].
- Often it is sufficient to describe a technical system only in a small subregion of the input space with high accuracy while the model performance in other regions of the input space is not important [96].

However, methods used for experimental modeling have to exploit these real-world properties that weaken the curse of dimensionality. Crucial procedures in coping with the curse of dimensionality are input selection and design of experiments (DoE)

techniques [29] which are the main topics of this thesis. Alternative approaches which are not the main topic of this thesis are to exploit some of the above listed points on the level of the model structure. This can be done explicitly, e.g. with additive models, or implicitely, e.g. with sparsity induced via regularization. The motivation for this thesis is given in Section 1.1. Objectives and the structure of it are described in Section 1.2.

1.1 Motivation

An ongoing trend in the field of experimental modeling is an ever increasing amount of inputs used to describe technical systems by models [56]. This trend arises mainly due to the following two reasons:

- Ever increasing process complexities create the demand for higher degrees of freedom to control them via additional actuators, which leads to potentially more inputs for the models.
- The demand for higher model accuracies originating from stricter regulations and market demands, such that potentially more influences ($\hat{=}$ inputs) have to be incorporated into models to fulfill them.

The problem with an increasing input dimensionality is, as explained previously, the curse of dimensionality. The main motivation and topic of this thesis is the development and investigation of methods helping to cope with the curse of dimensionality. Most of the investigated methods exploit unique properties of LMNs. There are basically two main topics of this thesis, which are input selection and the DoE.

Input selection tackles the curse of dimensionality directly, since inputs are identified and removed that harm the model accuracy. There might be cases in which no input can be removed without harming the model accuracy, i.e. when the best bias/variance tradeoff is achieved with all inputs. DoE techniques specify locations in the input space at which measurements should be conducted in order to create a data set that can be used for the generation of a model. Efficient experimental designs keep the number of necessary measurements as low as possible while the process behavior can still be identified with the required level of accuracy. Therefore, DoE techniques help also to weaken the curse of dimensionality through the reduction of the data necessary to achieve a demanded model quality.

1.2 Objectives and Structure of this Thesis

One main objective of this thesis is the development of algorithms that exploit the unique property of LMNs to distinguish between linear and nonlinear effects for input selection tasks. Two different input spaces arise due to this unique property when using LMNs with local affine models which is the most popular choice. This leads to more degrees of freedom, because inputs can be assigned to each input space individually. Potentially, this additional freedom enables input selection methods using LMNs to better fine-tune the bias/variance tradeoff. Furthermore, one of the two arising input spaces is directly connected to the nonlinear effects of the model, which allows a more detailed interpretation. On the other hand, the problem of finding good input subsets gets harder due to an increased amount of potential input subsets because each physical input can be assigned to each of the two input spaces. In this thesis it should be investigated, if the exploitation of the input space separation brings more advantages or disadvantages. Therefore, one wrapper-like method is developed that exploits the separation of linear and nonlinear effects fully. In general, wrapper methods use a system identification algorithm and wrap the input selection around it, i.e. several input subsets are tried out and the best found subset is finally used.

Another objective of this thesis regards axis-oblique partitioning strategies for LMNs. Axis-oblique partitioning strategies have been proposed to more effectively deal with high-dimensional input spaces compared to much wider spread axis-orthogonal strategies, weakening the effects of the curse of dimensionality. Usually, the axis-oblique partitioning is obtained by nonlinear optimization techniques, introducing additional flexibility together with an increase in variance error. In this thesis a normalized L1 regularization for optimization-based axis-oblique partitioning strategies is proposed, which only penalizes the amount of obliqueness incorporated in the partitioning. It is exemplarily implemented for the axis-oblique partitioning strategy included in the HIerarchical LOcal MOdel Tree (HILOMOT) algorithm. It should be investigated if the proposed normalized L1 regularization increases the generalization performance of LMNs and reduces the variance error significantly.

Furthermore, it should be investigated if the final partitioning, i.e. after the training algorithm has finished, contains sufficient information about the nonlinear influences of all used LMN inputs. Therefore, a strategy to analyze the partitioning of LMNs has to be developed and investigated.

In addition to exploiting special properties of LMNs, it is tested if the visualization technique of partial dependence plots is able to reveal the most relevant inputs for a model. Partial dependence plots are scientifically not new, but in the experience of the author seldom used and rather unknown, at least for people with an engineering background.

In the field of DoE techniques two questions regarding the choice of the experimental design should be investigated:

Should corners (vertices) be measured? This question should be investigated with the help of the function generator described in Section 2.7 for several extrapolation scenarios and ratios of corner points in relation to the overall data set size.

Are maximin Latin Hypercube designs superior to Sobol sequences? A comparison of several space-filling designs in terms of the model quality and variations of it incorporates Sobol sequences, data coming from a uniform distribution, and several maximin optimized Latin Hypercube (LH) designs. The optimized LH designs differ in the way they are optimized. In particular maximin optimized LH designs generated with the extended deterministic local search (EDLS) algorithm [35] from phase one and two (see Section 4.2.2 for details) are used as well as one function implemented in the commercially available software Matlab.

Another objective of this thesis in the field of DoE deals with an often neglected aspect which regards the order in which the measurements should be conducted to yield the best possible model performance with fractions of the whole experimental design. Good models which require good data in early stages of the measurement process yield several advantages, e.g. time can be saved because demanded model qualities can be reached with less data. In addition, the model can be used earlier while the measurement process is still in progress. Methods to determine the order of experimentation are intended for passive learning strategies, where the DoE plan is known and fixed in advance to the measurement process. Therefore, it can be used in situations in which the degree of automation is not sufficient for active learning strategies. Additionally, these methods can also be used for the initial experimental design needed before an active learning strategy can start. Three different strategies to determine the order of experimentation are developed and compared to each other.

Eventually, the already existing HILOMOT for design of experiments (HilomotDoE) algorithm that follows an active learning strategy should be extended to incorporate a goal-orientation. This goal-oriented active learning strategy addresses three main goals: (I) The concentration on possibly near-optimum regions and (II) the focus on areas in the design space where the (meta-)model's performance is considered to be worst. Additionally, (III) new measurements should differ from already gathered data as much as possible. With these goals three important issues in modeling are addressed simultaneously: (I) optimality, (II) model bias, (III) model variance / uniformly space-filling property.

In order to fulfill all objectives this thesis is structured in the following chapters.

Chapter 2 provides important theoretical background information about experimental modeling in general, the bias/variance tradeoff, input selection, DoE techniques, metamodels, and a function generator that is extensively used throughout the whole thesis.

Chapter 3 presents the developed input selection methods that take advantage of the input space separation of LMNs. Investigations on artificially created test processes reveal advantages and shortcomings of the proposed methods. Additionally, partial dependence plots are reviewed as tool to find and visualize the most relevant inputs for an experimental modeling task.

Chapter 4 contains all developed and investigated methods related to DoE. Methods to determine the order of experimentation are described and compared with the help of a function generator. Advices about experimental designs are given based on investigations with the function generator. In particular, it is examined if measuring corners is advisable and what space-filling design (maximin LH or Sobol sequence) should be used. Eventually, the goal-oriented active learning strategy with LMNs is presented.

Chapter 5 provides a selection of examples in which the developed input selection and DoE techniques proved to be helpful in practice. Several input selection methods are applied to the auto miles per gallon data set, which is publicly available from the machine learning repository of the University of California Irvine (UCI) [80]. Other applications incorporate the prediction of the air-mass flow into a cylinder of a combustion engine, the metamodeling of fans, and a heating, ventilating, and air conditioning (HVAC) system. Due to space restriction not all applications in which the proposed input selection methods

were useful could be included in this thesis. An application in which the NO_x emissions are predicted based on 64 potential inputs is omitted as well as the prediction of cycle times for tugger train systems based on 47 inputs.

Chapter 6 contains the final conclusions of this thesis and an outlook of possible future research topics.

The main contributions and novelties of this thesis are listed in the following.

- Development of a mixed wrapper-embedded input selection method that fully exploits the input space separation offered by LMNs.
- Development of the normalized L1 split regularization for axis-oblique partitioning strategies which is implemented for the HILOMOT algorithm.
- Development of a partition analysis method for LMNs.
- Development of several methods that can be used to determine the order of experimentation for previously determined experimental designs.
- Development of a goal-oriented active learning strategy based on an already existing active learning strategy, namely HilomotDoE.
- Implementation and application of the mixed wrapper-embedded input selection to identify a good dynamic model structure for a HVAC system. The purpose of the identified dynamic model it to be used in a model-based control strategy. However, the model-based control strategy is not part of this thesis.
- Implementation and application of the goal-oriented active learning strategy for the generation of a data set used to train computational fluid dynamics (CFD) metamodels.

2 Nonlinear System Identification

In general, a system is a term used for the totality of elements that are related to each other and organized as delimited outwardly structure. By the definition of a boundary, the scope of a system as well as its in- and outputs are defined. For example the engine of a car can be seen as a system, where the motor rotation speed, the valve timing, and many more inputs influence the outputs, which could be the available torque, fuel consumption, exhaust fume emissions and so on. Another boundary might also include the motor control gear as well as parts of the drive train, leading to other in- and outputs. The scope of this thesis is settled in the field of nonlinear system identification, i.e. generating models that describe the input/output relationship of nonlinear systems solely based on measured data. Therefore, a system is the abstraction of an arbitrary process, that can be from all kinds of technical, economical, physical, chemical, etc. fields. The term process is synonymously used for system throughout this thesis.

An increasing amount of modern development methods are based on models. In the context of this thesis, a model can be seen as a mathematical description of a real-world process and can be utilized to predict the output of a technical system given the input values or sequences. Models can save time and resources since extensive investigations can be done on totally different time-scales compared to real-world measurements or applications. Besides the time-saving aspect, there are some advanced techniques in the field of optimization, design of controllers, supervision and many other fields that would hardly be possible without models. Figure 2.1 shows some scenarios in which models are utilized.

There are basically two ways to derive models, which are (I) using first principles and (II) using measured data. When using first principles, insights from physical, chemical, biological, economical, etc. laws are necessary to derive at least the structure of the model. The parameterization of so called white-box models might be supported by measurements. A key property of white-box models is the direct interpretability of all model parameters in terms of first principles [96]. The second way to obtain

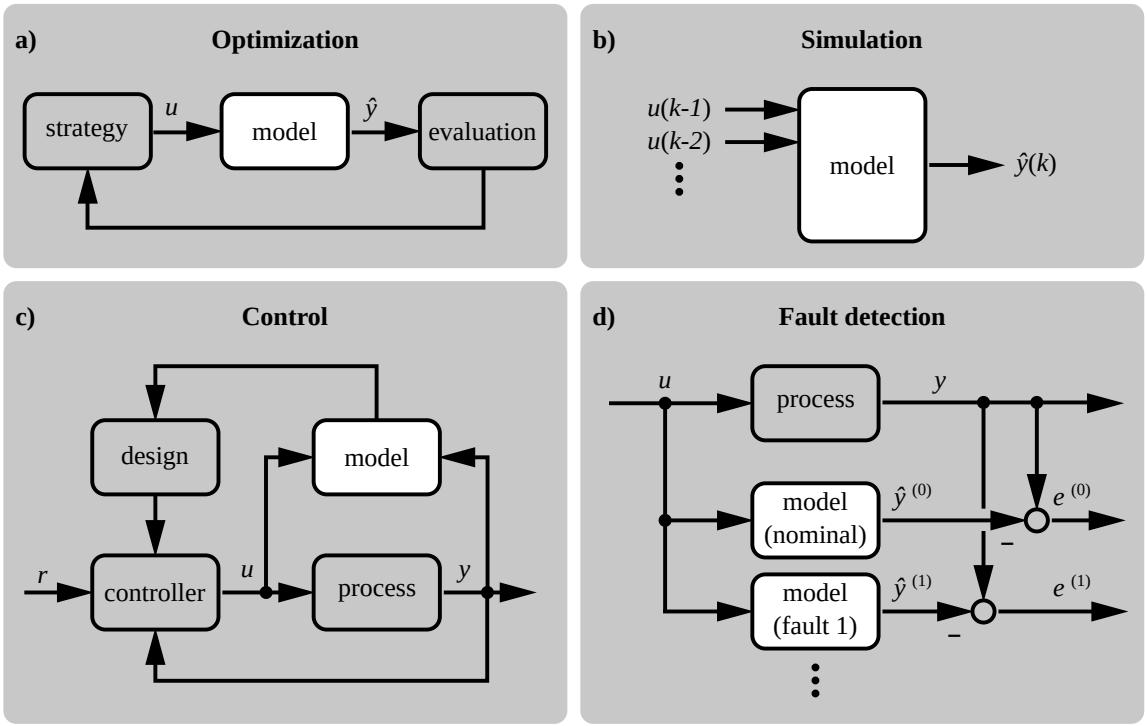


Figure 2.1: Models utilized for a) optimization, b) simulation, c) control and d) fault detection [96]

models is solely based on measured data. Since for the model derivation only experimental data is used, this approach is called experimental modeling. Other commonly used names are black-box modeling or system identification [96]. The term system identification often refers to the generation of *dynamic* models [126], [96]. Most real-world processes behave differently depending on the operating point, e.g. there are changes in the sensitivities of the system. Therefore the identified model structure should be able to yield nonlinear models to describe that behavior adequately. So-called gray-box models allow arbitrary nuances in combining white- and black-box modeling approaches. Additionally, other information sources such as qualitative knowledge formulated as rules may also contribute to the model generation [96]. Therefore gray-box models are any mixture of process insights and information extracted from measured data. This thesis focuses on black-box modeling approaches and a specific model type, namely local model networks.

The next sections provide information necessary to comprehend and acknowledge the concepts explained in Chapter 3 and 4. In most cases only elementary explanations are given and references to more detailed literature are provided. Section 2.1 describes the differences between static and dynamic models. The curse of dimensionality as well as the bias/variance tradeoff are topics of Section 2.2. The extensively

used model type of local model networks (LMNs) is described in Section 2.3 together with two training algorithms to obtain such models from data. Section 2.4 introduces commonly used methods to determine the most beneficial modeling inputs. Basics about the design of experiments (DoE) are described in Section 2.5. The last two sections deal with the task of metamodeling (Section 2.6) and a newly designed function generator to build examples for static regression problems (Section 2.7).

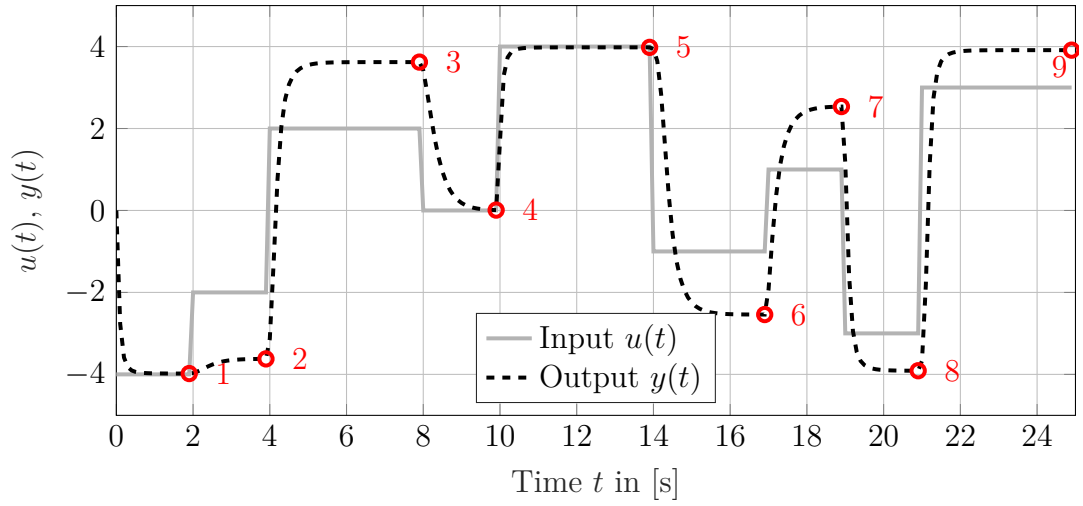
2.1 Static and Dynamic Models

The same real-world process or system might be described either by a *static* or *dynamic* model. For the sake of a simple notation all following explanations do only consider single-input single-output (SISO) and multiple input single output (MISO) systems. An extension to systems with multiple outputs is straightforward. Figure 2.2 visualizes the dynamic relationship as well as the static relationship of one exemplarily used process. The dynamic relationship, depicted in Fig. 2.2a, shows the time courses of the input of the system $u(t)$ and the corresponding output $y(t)$. In Fig. 2.2b the static relationship between the input u and the output y is shown. The circles and numbers in both figures indicate, where steady states of the process are reached. For the static relationship only steady states are of interest, no transient behavior is captured. Which of these two model types, i.e. static or dynamic, is appropriate depends on the intended use of the model and can not be answered in general. The error made through the static description of an actually dynamic process decreases the faster the system reaches its steady states.

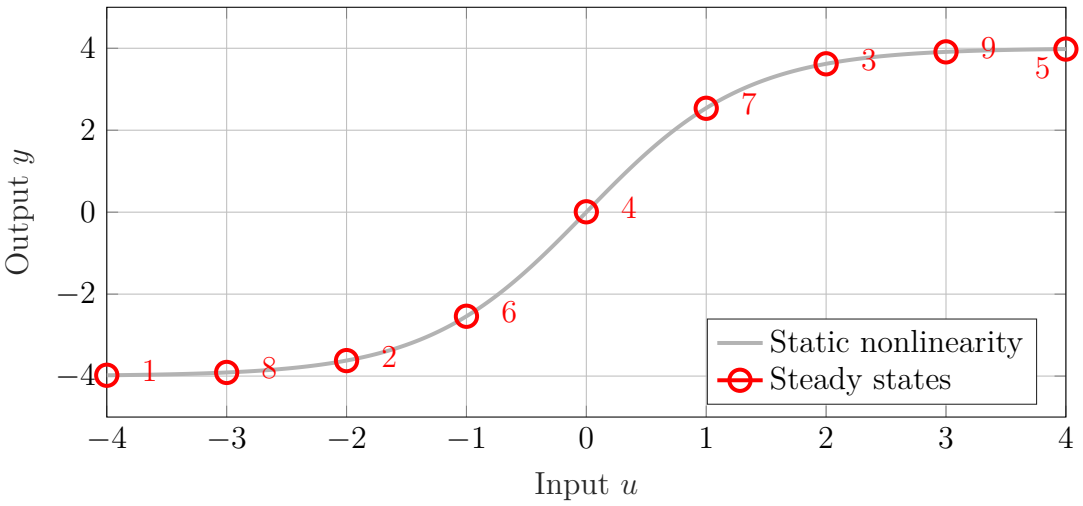
For the task of experimental modeling two general cases are distinguished. First, the generation of a static model and second, the identification of a dynamic model. If the transient behavior of the (real) system under investigation is not of interest or the steady states of the system are reached very quickly, a static model may be used. In this case the relationship between the inputs of the system and the output of the system can be described by an algebraic equation

$$y = f(\underline{u})\,, \tag{2.1}$$

where $f(\cdot)$ is the true function describing the system of interest. The goal of experimental modeling is to find a good approximation $\hat{f}$ of the true function f based on



(a) Dynamic relationship



(b) Static relationship

Figure 2.2: Distinction between the dynamic (a) and static (b) input/output relationship for the same process. Numbers indicate the steady states of the dynamic process.

a finite data set $D = \{\underline{u}(i), y(i)\}_{i=1}^{N}$ of N samples [13] with the system output y and p system inputs $\underline{u}(i) = \begin{bmatrix} u_1(i) & u_2(i) & \cdots & u_p(i) \end{bmatrix}$.

A time-invariant, deterministic, nonlinear dynamic system in discrete-time can generally be described by the state space representation [79]:

$$\underline{x}_s(k+1) = \underline{h}(\underline{x}_s(k), \underline{u}(k)) \tag{2.2}$$

$$y = g(\underline{x}_s)\,, \tag{2.3}$$

with the system states $\underline{x}_s$, discrete-time k as well as the nonlinear mappings $\underline{h}(\cdot)$ and $g(\cdot)$. One way to approximate the dynamic system would be to approximate the functions $\underline{h}(\cdot) = \begin{bmatrix} h_1(\cdot) & h_2(\cdot) & \cdots & h_{n_s}(\cdot) \end{bmatrix}$ and $g(\cdot)$, with the number of system states being n_s [96]. However, in practice complete state measurements are rarely realistic. One way to deal with missing states is to estimate them which leads to so-called *internal dynamics* approaches. However, no further details are given here, since this thesis focuses only on the so-called *external dynamics* approach. Details about internal dynamic approaches can be found in [2], [78], [116], [133] and [139]. A comparison between the external and the internal dynamics approach can be found in [96].

The external dynamics approach is the most frequently applied way to model nonlinear dynamic systems according to [96]. A visualization (for a series-parallel model) of the external dynamics approach is given in Fig. 2.3. Two parts can be distinguished, i.e. the external dynamic filter bank followed by a nonlinear static approximator [59]. In Fig. 2.3 general transfer functions for the inputs $G_i^{(u)}$ and outputs $G_i^{(y)}$ are shown. The works of Casdagli [23] and Poncet et al. [102] have shown that the input/output behavior of almost any nonlinear system can be approximated arbitrarily precisely with the external dynamics approach.

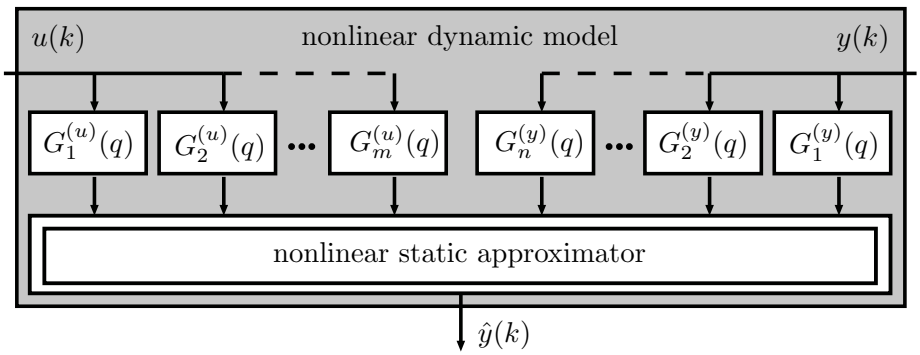


Figure 2.3: External dynamics approach for a SISO system with input order m and output order n

If simple time delays q^{-1} are used as filters, the model output $\hat{y}$ depends on delayed versions of the physical inputs u and outputs y:

$$\hat{y} = \hat{f}(\underline{\varphi}) \text{ with} \tag{2.4}$$

$$\underline{\varphi} = \begin{bmatrix} u(k) & u(k-1) & \dots & u(k-m) & y(k-1) & \dots & y(k-n) \end{bmatrix}, \tag{2.5}$$

with the maximum delay of the inputs m and outputs n.

Independent of the model type, i.e. static or dynamic, it is common practice to have one number representing the overall performance of it. Throughout this thesis usually the normalized root mean squared error (NRMSE)

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y(i)-\hat{y}(i))^2}{\sum_{i=1}^{N}(y(i)-\bar{y})^2}}, \tag{2.6}$$

with the model predictions $\hat{y}(i)$ and the measured outputs $y(i)$ is used. This error measure indicates the improvement (NRMSE < 1) by a model compared to the mean value of all measured outputs $\bar{y}$ (NRMSE $= 1$). Therefore it is independent of the range of the outputs and can simply be interpreted. Note that the NRMSE value can be calculated for different data sets, e.g. on training or test data.

2.2 Curse of Dimensionality and Bias/Variance Tradeoff

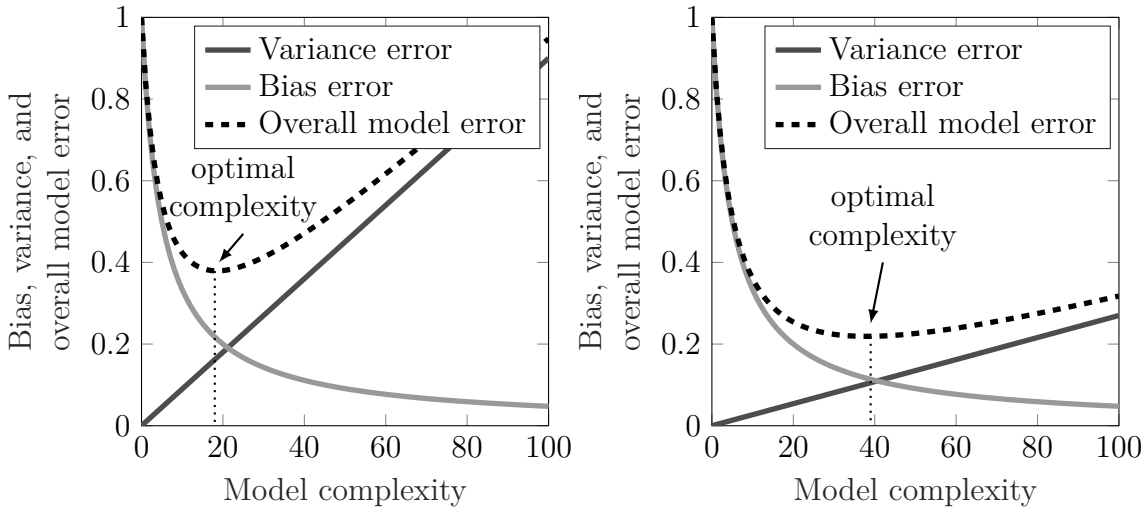
Even though the problem was known long before, Bellman coined the phrase *curse of dimensionality* in [7]. It describes the *exponential* increase in effort with an increasing input space dimensionality. For example, if the input space should be covered by data lying on a grid with four measurements per input dimension, the necessary amount of data for $p = 2$ inputs equals $4^p = 4^2 = 16$. For $p = 3$ the needed data amount is already at $4^3 = 64$ and for an input space dimensionality of $p = 10$ more than one million measurements would be necessary. Other illustrations of the curse of dimensionality can be found, e.g., in [57] and [29]. As a result the task of experimental modeling gets more difficult due to rapidly increasing memory requirements, computational complexity, model evaluation effort, etc. For real-world problems this implies that in high-dimensional input spaces all data sets are extremely sparse [29].

However, there are techniques that are able to cope with the curse of dimensionality mainly because of two reasons. The first reason is that data is often confined to specific regions of the input space. As a result directions in which important variations in the target variables occur are also often confined, resulting in a lower effective dimensionality [15]. The second reason originates from a smoothness property that is often fulfilled for real-world problems [15]. Small changes in the input variables usually cause only small changes in the target variables, leading to at least locally predictable behavior [96]. Crucial procedures in coping with the curse of dimensionality are input selection and DoE techniques [29] discussed in this thesis.

Input selection techniques try to find a subset of input variables leading to a model with the best possible generalization performance. This may even imply to omit inputs that are known to be relevant in a theoretical/first principles way. The reason for this can be explained by the bias/variance tradeoff. The error of a model can be decomposed into the bias error and the variance error [48]. The bias error arises due to a too simple model, that is not able to capture the complexity of the true functional relationship [53]. The origin of the variance error lies in the uncertainty of the estimated parameters [96]. Data sets used for training are always finite and noisy. As a result, there will be deviations of the estimated parameters if different data sets of one process are used to identify the model parameters. The model error due to these variations is the variance error and depends for least squares parameter estimates and large training data sets on the number of model parameters n_θ, the amount of used training data N, and the noise variance ${\sigma_n}^2$ as follows [57]:

$$\text{variance error} \propto {\sigma_n}^2 \frac{n_\theta}{N} \,. \tag{2.7}$$

Even though (2.7) is only exactly valid for least squares parameter estimates and large training data sets, it is remarkable that the same tendencies are qualitatively valid for small data sets and regardless of the special type of model (and estimation procedure) used [57]. As the variance error of a model increases with an increasing amount of model parameters according to (2.7), the bias error decreases. Additional parameters enable a model to describe more complex functional relationships and therefore increase the model flexibility. In order to minimize the overall model error, a good tradeoff has to be found given a specific amount of data [143]. Figure 2.4 shows typical curves for the bias, variance, and overall model error for different amounts of data. As can be seen from Fig. 2.4 and (2.7), the slope of the variance error is influenced by the data amount. Considering that each additional input variable typically increases the number of model parameters, there might be cases in which



(a) Few data samples

(b) Many data samples

Figure 2.4: Typical curves of the bias, variance, and overall model error for few (a) and many (b) data samples

the increase in variance error exceeds the improvement of the bias error. In such cases omitting theoretically important inputs yields benefits for the overall model error. Mathematical derivations of the model error decomposition into the bias and variance part can be found in [96], [53], and [15] amongst others.

In practice the variance part of the error cannot be detected on training data alone because it arises from the uncertainties in the model parameters caused by the particular noise realization and distribution of the training data. Because of this, special techniques are necessary to determine the possibly best model complexity and avoid an overly complex model that already describes the specific noise realization, which is called *overfitting* (low bias, high variance) [96]. Throughout this thesis two approaches for the model complexity determination are used. Either a separate validation data set or an information criterion is applied. Information criteria add a complexity penalty term to the training error to approximate the validation error. The herein used information criterion is a corrected version of Akaike's information criterion ($\mathrm{AIC_c}$):

$$\mathrm{AIC}_c = -2 \ln \mathcal{L}(\underline{\theta}|y) + 2n_{\mathrm{eff}} + \frac{2n_{\mathrm{eff}}(n_{\mathrm{eff}}+1)}{N - n_{\mathrm{eff}} - 1} \,, \tag{2.8}$$

with $\mathcal{L}$ being the likelihood function of the estimated model parameters $\underline{\theta}$ given the observations y, the number of effective parameters n_{eff}, and the number of data samples N. The last term of (2.8) corrects the value of the (uncorrected) AIC to

account for situations with relatively few data samples [20]. To assess the final model quality an additional test data set is required as described in more detail in [96]. Other approaches for the determination of the optimal model complexity are, among others, cross-validation, multi-objective optimization strategies or statistical tests, which are also described in [96].

2.3 Local Model Networks

The basic idea of LMNs is based on a divide-and-conquer strategy. One whole problem is subdivided into smaller problems that can be solved almost independently [94, 95]. LMNs employ local models within a basis function network [95] and can be seen as an extension of normalized radial basis function (NRBF) networks that fulfill the partition of unity [70]. See [103, 83, 45] for more details about NRBF networks. The weights of each basis function are simply replaced by local models. This idea has been suggested several times in the literature with changing names and different terminologies. Mixtures of local experts [63, 71], growing multi-experts networks [82] or receptive field weighted regression [127, 114, 115] are some of the names that mean basically the same model type. The methods of Takagi and Sugeno [129, 128, 44] can also be interpreted as LMNs with fuzzy interpolation between local affine models. A subcategory of LMNs are piecewise affine models [124, 41]. As the name indicates, the local models are of affine type and usually the validity functions are hard-switching, such that discontinuities between neighboring local models are possible [41]. The terminology in this thesis is geared mainly to the one used in [94, 95] and [100, 96, 97]. Despite the variety of publications dedicated to LMNs, one special property of this model type is rarely exploited. This property concerns the possibility to distinguish two types of input spaces for LMNs since the input variables on which the local models and the validity functions operate can be completely different. In [96] implications of the two arising input spaces are discussed in detail. One main goal of this thesis is to exploit this property in an input selection context, such that the best subsets of variables for the two input spaces are determined automatically.

Since LMNs can be expressed in a basis function context, the model output $\hat{y}$ is calculated as sum of M so called *local* models $\hat{y}_i$ weighted with their validity func-

tions Φ:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\underline{x})\Phi_i(\underline{z})\,. \tag{2.9}$$

As indicated in (2.9), the local models can depend on other variables than the validity functions. The vectors $\underline{x}$ and $\underline{z}$ consist of subsets of all physical inputs $\underline{u} = [u_1\ u_2\ \cdots\ u_p]$, with p being the number of physical inputs:

$$\underline{x} \subseteq \underline{u} \text{ and} \tag{2.10}$$

$$\underline{z} \subseteq \underline{u}\,. \tag{2.11}$$

The physical inputs contained in $\underline{x}$ and $\underline{z}$ are completely independent, i.e. they can be completely distinct ($\underline{x} \cap \underline{z} = \emptyset$), identical ($\underline{x} = \underline{z}$) or anything in between. From these definitions two independent input spaces arise, i.e. the $\underline{x}$-input space and the $\underline{z}$-input space. Figure 2.5 illustrates this separation into two classes of input variables for LMNs in contrast to a general nonlinear model. In a fuzzy interpretation the inputs contained in the vector $\underline{z}$ are in the rule premises (IF part) and the inputs contained in the vector $\underline{x}$ are in the rule consequents (THEN part). The previous explanations only address static models. However, an extension for the modeling of dynamic systems pursuing the external dynamics approach (see Section 2.1) is straightforward. Basically the physical inputs $\underline{u}$ in (2.10) and (2.11) have to be replaced by the dynamic model inputs $\underline{\varphi}$ from (2.5).

The local models can in principle be of any model type. Often linearly parameterized local models are chosen such that their parameters can efficiently be estimated

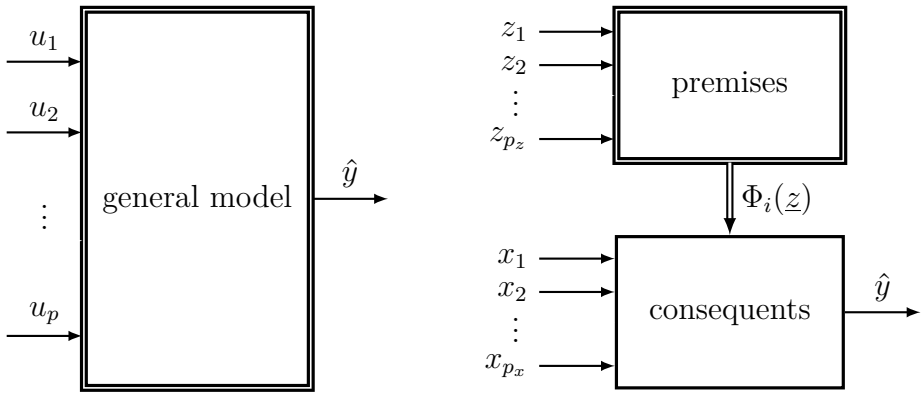


Figure 2.5: For LMNs the inputs can be assigned to the premise and/or consequent input space according to their nonlinear or linear influence on the model output [97]

by some sort of least squares (LS) optimization technique. The validity functions describe the regions where the local models are valid; they represent the contribution of each local model to the output [97]. For a reasonable interpretation of LMNs it is mandatory that the validity functions form a *partition of unity* [97]:

$$\sum_{i=1}^{M} \Phi_i(\underline{z}) = 1, \quad \forall \underline{z} \ . \tag{2.12}$$

If the partition of unity holds, the fraction contribution for local model i is Φ_i. To the knowledge of the author there exists no linearly parameterized mathematical description of the validity functions. As a result either heuristics or nonlinear optimization techniques have to be applied in order to determine the parameters of the validity functions. The determination of all parameters, i.e. the local model parameters and the validity function parameters, can be achieved via different algorithms. In this thesis two prominent algorithms are utilized, which are the LOcal LInear MOdel Tree (LOLIMOT) and the HIerarchical LOcal MOdel Tree (HILOMOT).

The LOLIMOT algorithm [99, 96] uses a heuristic to incrementally grow LMNs. In each iteration the number of local models is increased by one. Therefore one already existing local model is divided into two new local models, see Fig. 2.6. The worst performing local model (gray areas in Fig. 2.6) is determined and chosen for further subdivisions. All splits orthogonal to one of the input space axes going through the center of the worst performing local model are tested (dashed lines in Fig. 2.6). The split leading to the best performance of the global model is conducted and the next iteration starts again with the determination of the worst performing local model. Then the procedure continues until some termination criterion is fulfilled, e.g. a de-

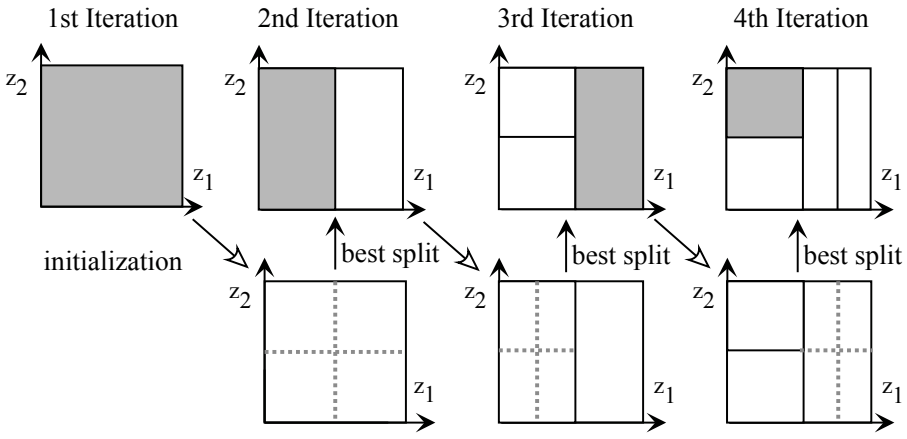
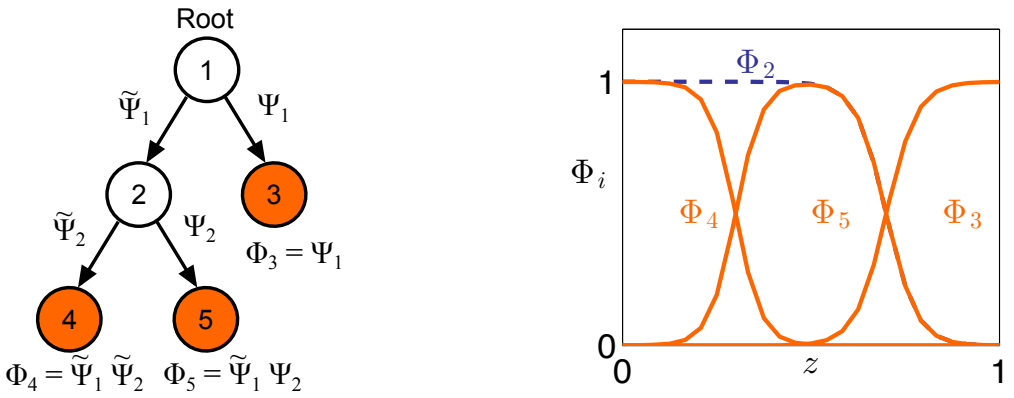


Figure 2.6: LMN structure optimization with LOLIMOT for a 2-dimensional $\underline{z}$-input space

manded model quality is reached or the generalization performance deteriorates. All subdivisions together are also called partitioning of the input space. Each subdivision creates one new validity function in (2.9). In case of the LOLIMOT algorithm, the validity functions are normalized Gaussians and the local models are polynomials of degree one (affine models).

The HILOMOT algorithm [97] is an extension of LOLIMOT in order to deal with high-dimensional input spaces more effectively. Important preliminary work for HILOMOT can be found in [18, 105, 37]. In the HILOMOT algorithm, the validities Φ are constructed by hierarchically linking sigmoid functions Ψ_i and/or their complementary functions $\widetilde{\Psi}_i$ in a multiplicative way as visualized in Fig. 2.7. The resulting structure can be represented by a binary tree, see Fig. 2.7a, where each circle corresponds to a node; nodes that are not further split are called leafs and represent the validity functions. Except for the root, each node has a so-called parent to which it is connected and which is located one hierarchy level closer to the root. Figure 2.7b visualizes the multiplicative construction of the validity functions. The former validity function Φ_2 (and parent node or split) is further split, leading to the new validity functions Φ_4 and Φ_5. Details about sigmoid splitting functions and their mathematical description follow in Chapter 3.

The HILOMOT procedure equals the one of LOLIMOT, but one more split is tested and subsequently a nonlinear optimization is performed additionally in each iteration. The additional split to be tested goes through the center of the worst performing local model and has the same orientation as the parent split. Through the nonlinear



(a) Binary tree

(b) Validity functions

Figure 2.7: Hierarchical binary tree (a) together with the corresponding validity functions (b)

optimization the location and orientation is adjusted in order to improve the model performance. Only the current split is optimized, all already existing splits are kept unchanged. Figure 2.8 visualizes the LMN structure optimization with HILOMOT.

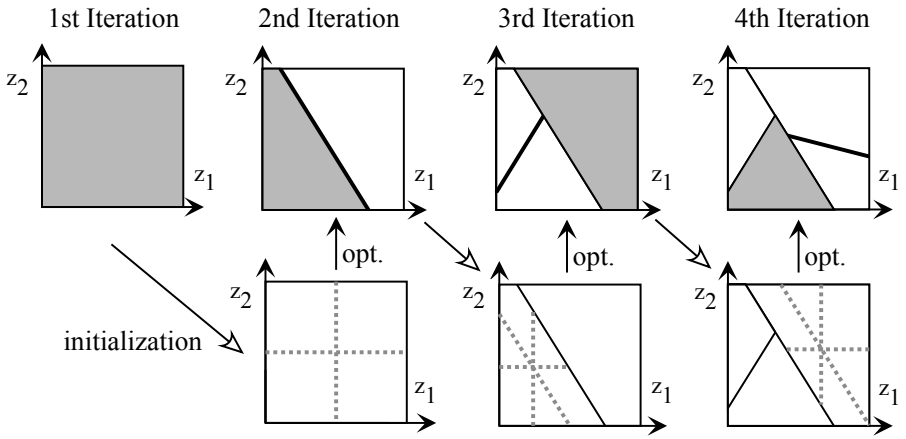


Figure 2.8: LMN structure optimization with HILOMOT for a 2-dimensional $\underline{z}$-input space

2.4 Input Selection

To face ever increasing accuracy demands in the real world, the number of variables that are considered to be important or can be manipulated has grown over the past decades. In general, a priori there is no information how useful which input variables are to model a certain kind of process. The term *useful* is chosen to make clear, that the best input variable subset regarding to model accuracy might not necessarily include all physically relevant input variables as stated in [76]. Additional input variables worsen the effects summarized under the principle „curse of dimensionality“, see Sec. 2.2. The variance error of the model is increased and the coverage of the input space with data samples becomes sparser. The question is, whether these negative effects are overcompensated by the information gain obtained through the corresponding input. However, there might even be input variables that have no relevance at all or that might be redundant [131]. Irrelevant input variables have no influence on the target variable at all, because there is no cause-and-effect connection. Redundant input variables influence the output variable but they are highly correlated to other input variables, such that the information carried by them is redundant. In this case incorporating just one of all highly correlated input variables

leads to a better bias/variance tradeoff. In summary, the reduction of the input space dimensionality of the model aims to

- improve the reliability and accuracy of the model by decreasing its variance error [27, 131, 93],
- reduce the time for model construction [64] and evaluation (once the input dimensionality reduction is finished - the determination of the best input variable subset corresponds to additional effort), and
- make the process under investigation more concise and transparent [29, 50].

There are three common input selection approaches which can be categorized into filter, wrapper, and embedded methods as stated in [51, 81, 138]. The main difference between filter and wrapper methods is the criterion used to evaluate merits of specific input subsets. Wrappers use a training algorithm as a black box and wrap the input selection around it [74]. The evaluation criterion is directly related to the performance of the resulting model. In contrast to that, filter methods do not rely on any model [111, 112, 69]. Often correlation or similarity estimates are used as mentioned in [52, 131]. Embedded methods utilize model-specific or training-algorithm-specific properties to find good input subsets as stated in [51, 131].

Formally the problem of input selection can be expressed as an optimization task, where the generalization performance P_G of a model should be maximized over all possible subsets from potential inputs $\mathbb{P}$:

$$\begin{aligned} &\underset{\underline{u}}{\text{maximize}} && P_G(\underline{u}) \\ &\text{subject to} && \underline{u} \subseteq \mathbb{P}\,. \end{aligned} \tag{2.13}$$

This is a combinatorial, nonlinear, and discrete optimization problem. Wrapper and embedded methods use criteria suited to measure the predictive power of a model directly, e.g. cross-validation or as suggested by Sindelar [122] the $\mathrm{AIC_c}$. Due to the absence of a model, filter methods need to use an intermediate criterion to approximate the generalization performance of potential models. In some cases even an input subset with less predictive power is preferred in favor of a more concise representation [29].

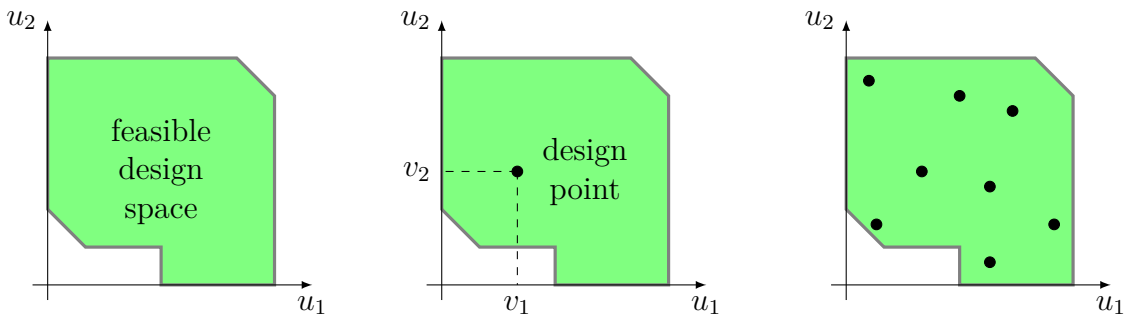
In order to find the best subset of inputs a feasible search strategy has usually to be applied. As stated by [131], trying out all possible input subsets would be the ideal approach but is by no means feasible even for a moderate number of potential

inputs p. Prominent search strategies include backward elimination (BE), forward selection (FS), and genetic algorithms.

A noteworthy approach to input selection that might be categorized as a wrapper technique is based on regularization methods and does not need a particular search strategy. The loss function to be minimized is extended by a regularization term, that penalizes nonzero model parameters associated with specific inputs. By increasing the regularization parameter, the number of selected inputs decreases automatically. Well known approaches include, among others, least absolute shrinkage and selection operator (LASSO) [132] and the elastic net [144].

2.5 Design of Experiments

Measured or simulated data plays the key role in experimental modeling. For the collection of data there is always effort necessary e.g. time, financial and/or material resources. The goal of DoE techniques is to use available resources in the most efficient way [39]. Some terms are defined in the following which are supported by the illustrations depicted in Fig. 2.9. The *design space* is spanned by all physical inputs. There might be infeasible regions in the design space due to restrictions that might arise from constructive, energy or any other limitations. Figure 2.9a exemplary shows the feasible region of a two-dimensional design space. A *design point* specifies values for each input as depicted in Fig. 2.9b. For this combination of input values, the target or output values have to be determined through measurements or simulations. The *experimental design* contains all design points, see Fig. 2.9c. The final data set



(a) Design space (b) Design point (c) Experimental design

Figure 2.9: Exemplary illustration of a design space (a), a design point (b) and a design of experiments (c)

consists of the experimental design together with all measured/simulated output values. DoE techniques are used to determine the locations of all design points of an experimental design. Through these techniques the amount of measurements should be kept at a necessary minimum.

In principle there are two ways for the determination of the DoE which are faced in Fig. 2.10. If a passive learning strategy is pursued, all points contained in the DoE are fixed and known before the first measurement is carried out. Typical designs for passive learning strategies comprise amongst other things (fractional) factorial and central composite designs, see [90] for more details. Often experimental designs following the passive learning strategy are optimized according to some optimality criterion, see [40, 1, 39] for common criteria, like e.g. D-optimality (minimization of the model's parameter variance), G-optimality (minimization of the model's output variance), etc.

In case of an active learning strategy the experimental design is not known completely a priori. Information gathered through already obtained measurements is used to select additional design points. In the context of active learning these adaptively determined design points are often called *queries*. As an example for the usefulness of active learning strategies, Clarke [29] uses logistic regression. In this field observations are needed where the sigmoidal functions are steep. Especially in high-dimensional input spaces hitting such regions is very unlikely if no information from already existing measurements is utilized. Active learning strategies have the

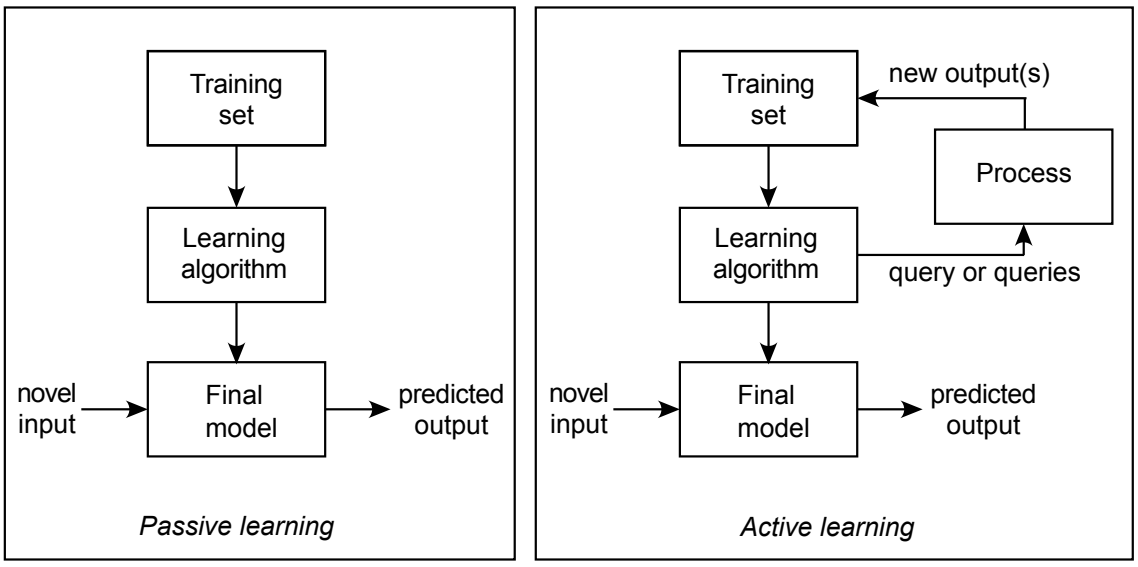


Figure 2.10: Comparison of passive and active learning strategies

potential to achieve the same model quality with significantly less data compared to passive learning strategies, as stated e.g. in [30, 31, 32, 84]. Active learners can be distinguished by their choice of the model and query strategies, i.e. how they choose the next query. A good survey on possible query strategies for active learners is given in [120].

2.6 Metamodeling

Because some of the applications presented in Chapter 5 incorporate metamodels, a short introduction to metamodeling is given here. As stated in [87], metamodels try to describe the true input/output relationship of deterministic computer simulations. Therefore, a metamodel is a computationally inexpensive "model of a model", that tries to approximate a computationally expensive simulation with high accuracy [121]. The decrease of computational complexity offers the possibility to intensively investigate a wide range of aspects. Typical application scenarios for metamodels are optimization tasks, where they substitute e.g. finite element method (FEM) or computational fluid dynamics (CFD) simulations, leading to the research field of metamodel-based design optimization (MBDO). Furthermore metamodels can be utilized to collect and visualize information about the process under investigation.

Figure 2.11 shows the process of generating a metamodel. The computationally expensive CFD simulation for the calculation of the efficiency of different impeller designs of centrifugal fans is chosen as an example for metamodeling. At first an experimental design is necessary, specifying which impeller designs should be CFD simulated in order to generate a training data set. This data set is then passed to a training algorithm following a black-box modeling approach. Arbitrary model types can be used for the metamodeling task. Prominent models include different types of artificial neural networks, Gaussian process models, splines, etc., see [6, 28]. Throughout this thesis only LMNs are used, see Section 2.3 for details.

2.7 Static Function Generator

During the development of methods and algorithms situated in the field of data-driven techniques it is always difficult to test new ideas. For the demonstration

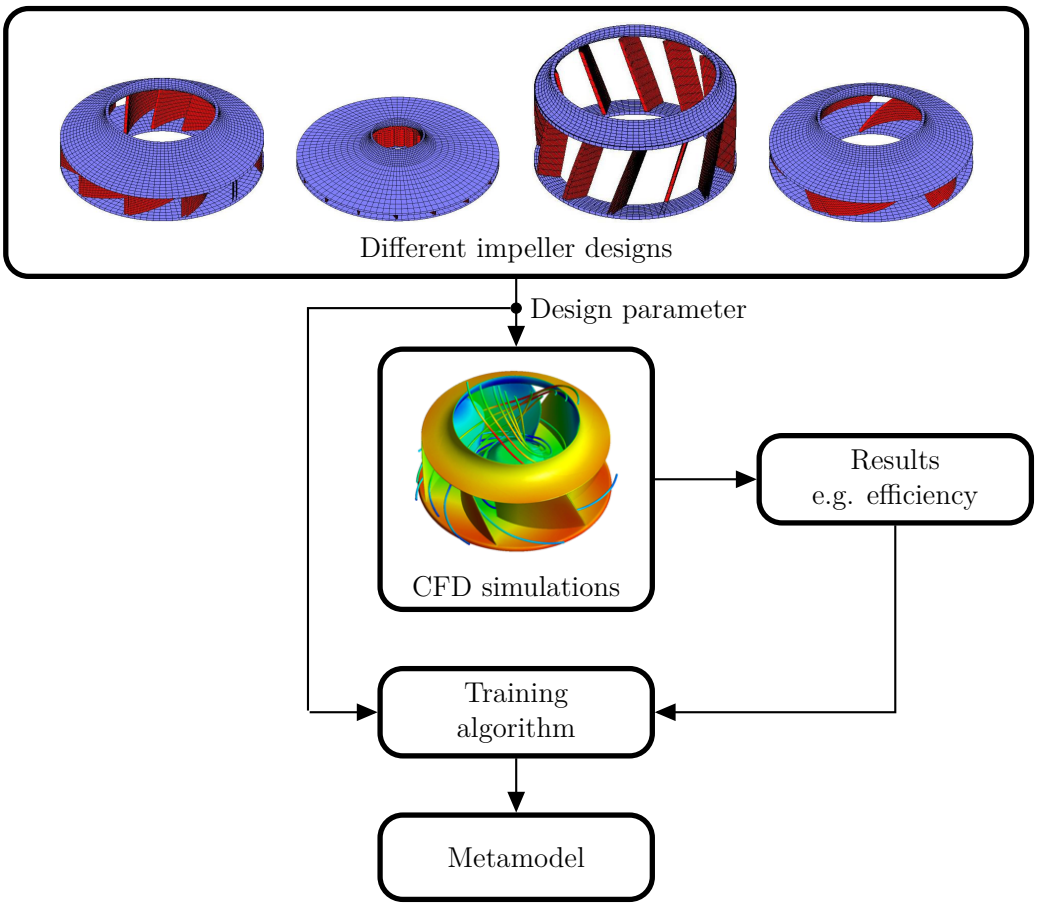


Figure 2.11: Metamodeling procedure for the substitution of computationally expensive CFD simulations

of strengths and weaknesses of new algorithms an arbitrary amount of synthetic examples would be of great help. In contrast to typical real-world or simulation data of specific applications, this offers the possibility to easily vary important factors such as:

- amount of data,
- dimensionality (number of inputs),
- strength of nonlinearity,
- data distribution,
- noise level, etc.

Furthermore the generation of multiple data sets of arbitrary size, e.g., for training, validation, testing is no problem. DoE and active learning strategies can be investigated nicely which is completely impossible for fixed data sets.

Infinite possibilities exist for building a function generator and there exists no overall "best" solution. Extremely few proposals for a function generator can be found in the literature. One very primitive approach has been made in the context of the massive online analysis (MOA) project [14] which is based on a random radial basis function network. A much more sophisticated approach proposed by Friedman [47] is based on additive Gaussians which gives an unfair advantage to all algorithms utilizing Gaussian kernels (typically used in support vector machines, radial basis function networks, Gaussian process models, etc.). The function generator used for investigations in this thesis is proposed in [11] and is meant to mimic static nonlinear regression problems. In contrast to other proposals, it is able to mimic saturation effects that often naturally arise in physical systems. Additionally a wide range of nonlinear characteristics can be generated and is controlled by very few parameters. However, the function generator is not intended to mimic a specific considered process. It is a generic way to generate test problems.

The function generator is based on randomly generated polynomials. For a p-dimensional input space, a polynomial arises from the sum of M_{poly} monomials according to the following equation:

$$g(u_1, u_2, \ldots, u_p) = \sum_{i=1}^{M_{poly}} c_i \cdot (u_1 - s_{i1})^{\gamma_{i1}} \cdot (u_2 - s_{i2})^{\gamma_{i2}} \cdot \ldots \cdot (u_p - s_{ip})^{\gamma_{ip}} \,. \tag{2.14}$$

Each monomial is a product of powers of variables with nonnegative integer exponents and a coefficient c_i. Here, these variables are the physical inputs u_j, $j = 1, 2, \ldots, p$ shifted by the randomly generated values s_{ij} with $i = 1, 2, \cdots, M_{poly}$. The shifts s_{ij} are drawn from a uniform distribution $\mathcal{U}[0, 1]$, whereas the coefficients c_i originate from a normal distribution $\mathcal{N}(0, 1)$. The physical input values should be normalized to the interval $[0, 1]$, such that all bases $u_j - s_{ij}$ lie in the interval $[-1, 1]$ after the shifts s_{ij} are subtracted. The powers γ_{ij} are non-negative integer values that are yielded by taking the floor of values coming from an exponential distribution with expected value μ. Therefore the strength of the nonlinearity is determined via two user-specified values M_{poly} and μ.

An optional extension is the transformation of the resulting polynomials with a sigmoid function:

$$h(u_1, u_2, \ldots, u_p) = \frac{1}{1 + \exp(-\alpha \cdot g(u_1, u_2, \ldots, u_p))} \,. \tag{2.15}$$

This yields functions that have the potential to possess saturation characteristics, which might be desired in order to mimic many typical real-world applications. The tuning parameter α determines the probability of saturation effects. Figure 2.12 shows exemplarily three two-dimensional polynomial functions originating from this function generator in the left column. The right column of Fig. 2.12 displays the sigmoidally saturated counterparts with $\alpha = 10$. As can be seen, the original polynomials are more or less deformed, depending on the output range of $g(\cdot)$. Note that for Fig. 2.12 all final output ranges have been scaled to the interval $[0, 1]$. A comparison to the function generator proposed by Friedman in 2001 [47] can be found in [10].

During investigations for this thesis one weakness of the polynomial-based function generator described in this section was observed. For relatively high-dimensional input spaces, the way the polynomial functions are created is very likely to lead to almost constant functions. With an increasing input dimensionality the number of bases in each of the M_{poly} monomials increases linearly and therefore more exponents have to be drawn. This increases the chance to obtain a high value for at least some of the exponents in each of the M_{poly} monomials, see (2.14). High values of the exponents are critical, because all bases $u_j - s_{ij}$ lie in the interval $[-1, 1]$ and the expression $(u_j - s_{ij})^{\gamma_{ij}}$ tends to zero if $|(u_j - s_{ij})| < 1$. This leads to counter-intuitive results if the expected value of the exponential distribution is chosen to be $\mu > 1$ in order to create more complex functions. To circumvent this weakness, the number of monomials M_{poly} has been increased to at least 20 and the standard deviation of the test function's output is tested on a large data set ($N = 10^5$) originating from a uniform distribution. The latter step is utilized to detect and discard almost constant test functions. Through the increased number of terms the probability that all M_{poly} terms become zero at the same time is decreased. In the case that the standard deviation of the output of a generated test function is below 0.01 an alternative test function is randomly generated.

An idea to improve the function generator in order to overcome the above described weakness systematically is to use the following modification of (2.14):

$$g_2(u_1, u_2, \ldots, u_p) = \sum_{i=1}^{p} \left[\sum_{j=1}^{M_{poly}} c_{ij} \prod_{k=1}^{i} b_{ijk}^{\gamma_{ijk}} \right] . \tag{2.16}$$

Already known symbols from (2.14) maintain their meaning. The only new symbol

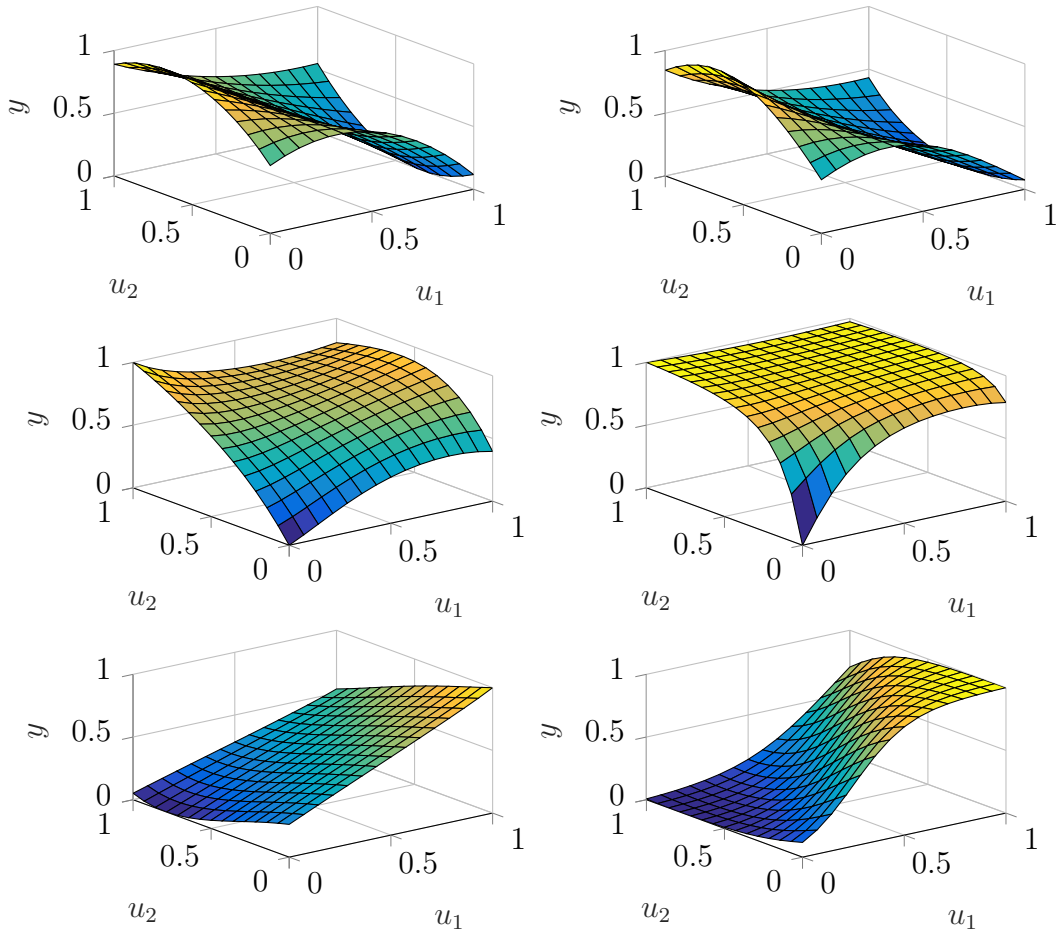


Figure 2.12: Two-dimensional polynomial example functions (left column) and their sigmoidally saturated counterparts (right column)

is an abbreviation for the bases:

$$b_{ijk} = u_l - s_{ijk}\,, \tag{2.17}$$

where index l determines which of the p inputs should be used with equal probability for the particular basis. For one of the M_{poly} monomials or a fixed value of j, respectively, i different inputs are selected by drawing values for l without replacement. The number of bases raised to the powers γ that are multiplied within one monomial is determined by the outer-sum-variable i. Thus, the originally proposed function generator from (2.14) is obtained if the outer-sum would start not at $i = 1$ but at $i = p$. Compared to the original function generator, this incorporates guaranteed low-order interactions, which are not prone to the weakness described in the last paragraph. The way the modified function generator works is related to the structure of functional analysis of variance (ANOVA) models, where the predic-

tion is composed of a sum of functions describing the main effects (functions of one variable) and interactions of two and more variables. More details about ANOVA models can be found in [46, 58]. Note that the modified function generator is just a proposal and has not been used for this thesis.

3 Input Selection Using Local Model Networks

The main goal of this thesis is to weaken the effects of the curse of dimensionality. In this chapter input selection methods in combination with local model networks (LMNs) are utilized to pursue this goal. As already described in Section 2.4, input selection methods try to find subsets of input variables that lead to the best possible bias/variance tradeoff. Advantages due to omitting input variables are:

- Fewer inputs (might) lead to a better bias/variance tradeoff.
- Fewer inputs lead to a better interpretability.
- The curse of dimensionality is weakened. Comparable model accuracies can be achieved with significantly fewer measurements.

When combined with LMNs additional advantages arise due to the possibility to automatically separate linear from nonlinear effects (as will be discussed in more detail in the next paragraph), such as:

- The number of variables contained in the $\underline{x}$- and $\underline{z}$-input space can be limited to the necessary minimum *individually*, leading to a possibly even better bias/variance tradeoff compared to a general nonlinear model.
- Possibility to separate linear and nonlinear effects when using local affine models.
- The design of experiments can be adjusted to exploit the knowledge about the linear and the nonlinear effects.

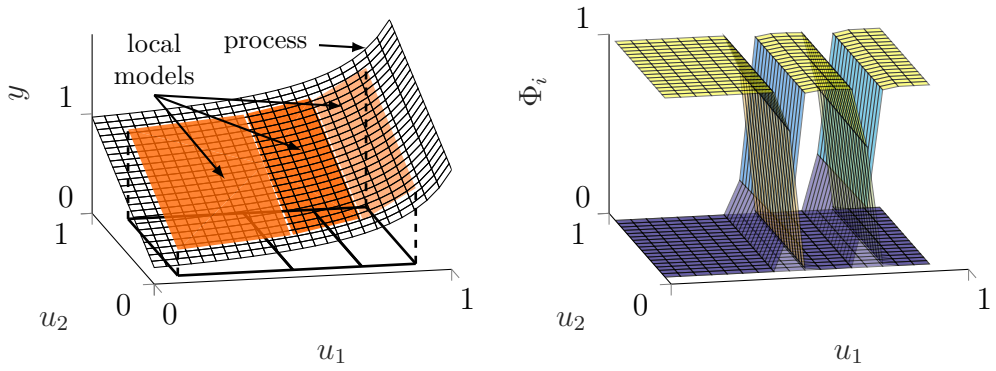
The main disadvantage caused by the input space separation regards the increased complexity due to the fact of having two input spaces. Input selection methods have to find good input subsets for both the $\underline{x}$- and $\underline{z}$-input space. In fact the number of

potential inputs to choose from is virtually doubled because each process input can be assigned to each input space individually.

The possibility to separate linear from nonlinear effects is a direct consequence of the ability of LMNs to separate the input space into a $\underline{x}$- and a $\underline{z}$-input space as visualized in Fig. 2.5. Variables contained in the $\underline{x}$-input space are used for the local models (rule consequents) whereas variables contained in the $\underline{z}$-input space belong to the validity functions (rule premises). If the local models are of affine type, the LMN is only able to follow a change in the slope of a process by switching to another local model. If some variables affect the process output only in an affine way, these variables are not needed in the $\underline{z}$-input space because the local models are able to capture their effects. To illustrate this, an artificial process following the equation

$$y(u_1, u_2) = \frac{0.2}{1.2 - u_1} + 0.8u_2\,, \tag{3.1}$$

is shown in Fig. 3.1a together with three affine local models. From (3.1) it is already clear, that input u_2 only has a linear effect on the process output whereas input u_1 contributes to the nonlinear process behavior. This can also be seen in Fig. 3.1. The affine local models are able to follow the process exactly in the u_2-direction. In order to follow the slope changes along the u_1-direction a partitioning along the u_1-axis is necessary. The validity functions Φ_i belonging to the local models from Fig. 3.1a are shown in Fig. 3.1b. It is easy to see that input u_2 has no influence on the validity functions and can therefore be omitted for the description of the partitioning. In this example necessary inputs for the $\underline{x}$- and $\underline{z}$-input space are $\underline{x} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}$ and $\underline{z} = u_1$, respectively.



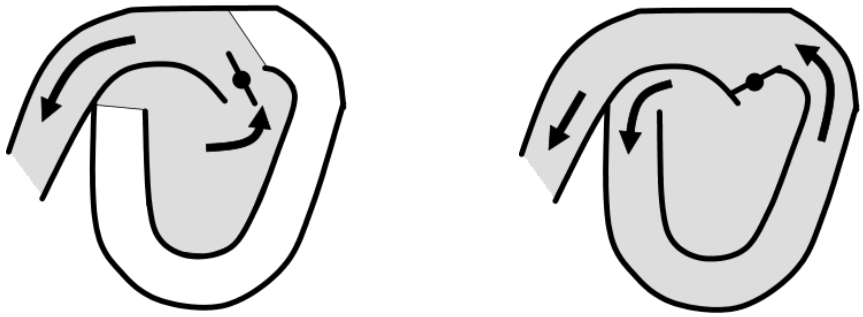
(a) Process with local models (b) Partitioning

Figure 3.1: Artificial process with one nonlinearly influencing input (u_1) and one linearly influencing input (u_2) together with three local models of an LMN (a) and its partitioning (b)

Besides the rather academic process from (3.1) a combustion engine with a variable-length intake manifold can also serve as an example for which the separation into $\underline{x}$- and $\underline{z}$-inputs is useful. The position of the swirl flap of the variable-length intake manifold changes properties of the dynamic system. As shown in Fig. 3.2 the position of the swirl flap changes the intake path. Through the intake elongation properties of the system obviously are subject to change, like time constants, gains, dead-times, etc. However, dynamic models describing the system for either an opened or closed swirl flap might not explicitly depend on the swirl flap position. As a result of this, an LMN describing the variable-length intake manifold system needs the swirl flap position only in the $\underline{z}$-input space, not in the $\underline{x}$-input space.

Another point of view is the following. The variables contained in the $\underline{z}$-input space define an operating point for which a specific and possibly affine local model is valid. The variables that define an operating point might not be needed to calculate the output of the corresponding local model. It is assumed that especially for dynamic models the number of $\underline{z}$-inputs typically can be kept small. In contrast to that, the number of $\underline{x}$-inputs might be relatively large in order to describe the dynamics of the local model adequately.

Mainly three different methods for input selection using LMNs are developed and investigated in more detail throughout the rest of this chapter. For all methods specifically designed test processes serve as benchmarks which are introduced in Section 3.1. Section 3.2 deals with a mixed wrapper-embedded input selection method. It utilizes existing training algorithms and wraps the input selection around it, but simultaneously exploits the LMN structure to separate the linear from the nonlinear effects. Section 3.3 elaborates a regularization-based input selection method through an extension of the HIerarchical LOcal MOdel Tree (HILOMOT) algorithm. Sec-



(a) Swirl flap open (b) Swirl flap closed

Figure 3.2: The position of a swirl flap in a variable-length intake manifold determines the intake path (gray-shaded regions)

tion 3.4 presents an embedded input selection method that extracts information about the nonlinearity of a process directly from the partitioning of the $\underline{z}$-input space. Eventually, Section 3.5 deals with a visualization technique that allows to inspect dependencies between input variables and the output even for high-dimensional input spaces. This visualization technique is applicable to any type of model and is therefore not restricted to the class of LMNs.

3.1 Test Processes

Test processes are used to demonstrate the abilities of presented input selection approaches. The focus lies on the possibility to distinguish between the $\underline{x}$- and $\underline{z}$-input space and the resulting input selection schemes that are depicted in Fig. 3.9. Throughout this chapter only local affine models are used, such that a distinction between linearly and nonlinearly influencing inputs is possible. As discussed in the introduction of Chapter 3, linearly influencing variables have to be included in the $\underline{x}$-input space and nonlinearly influencing variables have to be included in the $\underline{z}$-input space of an LMN in order to describe the corresponding process adequately. The test processes are designed such that all possible scenarios for the assignment of the physical inputs occur:

1. Physical inputs that are important only for the $\underline{x}$-input space,

2. physical inputs that are important only for the $\underline{z}$-input space,

3. physical inputs that are important for both the $\underline{x}$- and $\underline{z}$-input space, and finally

4. physical inputs that are not important for any of the two input spaces.

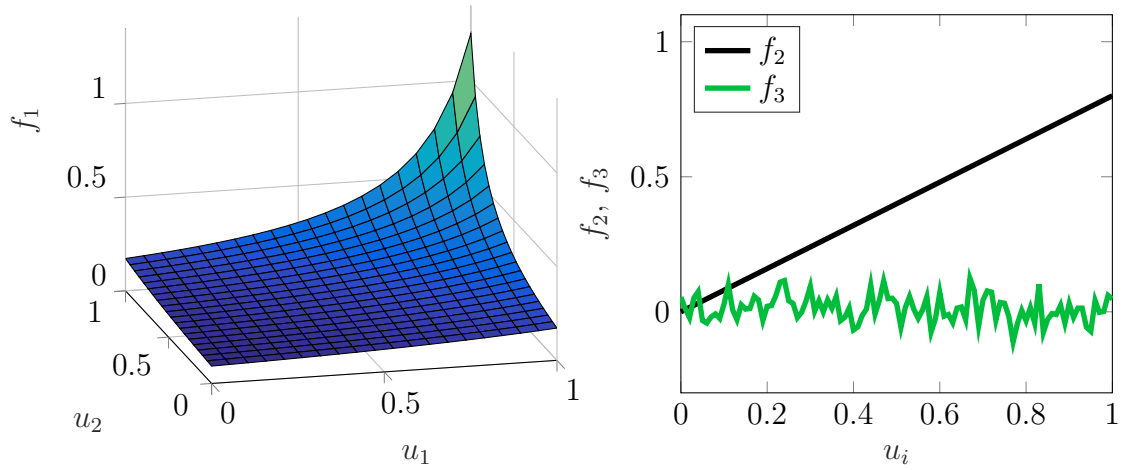
While being able to cover all of the above listed scenarios, the test processes are kept as simple as possible to enable for an easy understanding. The presented input selection approaches have to find the correct assignments of the physical inputs to the two existing input spaces. Specific values in test processes are chosen to illustrate e.g. variable importance or constraining the gain, etc.

Test Process One (TP1)

The first test process depends on four inputs, consists of three single functions that are summed up and follows the equation:

$$\begin{aligned} y(u_1, u_2, u_3, u_4) &= f_1(u_1, u_2) + f_2(u_3) + f_3(u_4) \\ &= \frac{0.1}{0.08 + 0.5(1 - u_1) + 0.5(1 - u_2)} + 0.8u_3 + u_4 \end{aligned} \tag{3.2}$$

Input u_4 corresponds to noise which is normally distributed $\mathcal{N}(0, 0.025)$ and therefore is important for neither the $\underline{x}$- nor the $\underline{z}$-input space. The values of all other inputs are assumed to lie in the interval $[0, 1]$. From (3.2) it is easy to see that inputs u_1 and u_2 are interacting and influence the output in a nonlinear way. Both inputs are therefore important for both the $\underline{x}$- and $\underline{z}$-input space. Because input u_3 has just a linear influence, it is only needed in the $\underline{x}$-input space. Functions f_1 and f_2 as well as one realization of the output of function f_3 are shown in Fig. 3.3. The only scenario that is missing in test process one (TP1) is scenario 2 - a variable important only for the $\underline{z}$-input space - but this will be covered in the second test process.

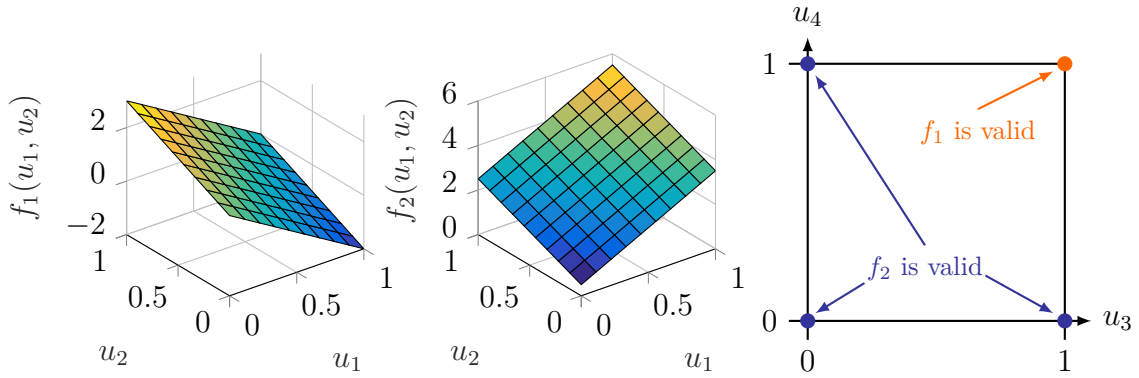


(a) Function $f_1(u_1, u_2)$ (b) Functions f_2 and f_3

Figure 3.3: Function f_1 (a) and the single functions f_2 as well as f_3 (b) of TP1

Test Process Two (TP2)

The second test process consists of two different planes which are shown in Fig. 3.4a and 3.4b. Which of the two planes is currently valid is defined by the values of



(a) Function f_1 (b) Function f_2 (c) Operating point variables

Figure 3.4: Operating points (a) that define when function f_1 (b) and function f_2 (c) are valid for TP2

two operating point variables as shown in Fig. 3.4c. Note that the operating point variables and the variables on which functions f_1 and f_2 depend are different ones. The operating point variables are discrete and are either equal to 0 or 1. Variables u_1 and u_2 are continuous and lie in the interval $[0, 1]$. Mathematically the output of test process two (TP2) can be calculated according to the following equation:

$$y = \begin{cases} f_1(u_1, u_2) & \text{for } u_3 = 1 \text{ and } u_4 = 1 \\ f_2(u_1, u_2) & \text{else} \end{cases} \tag{3.3}$$

with

$$f_1(u_1, u_2) = -3u_1 + 2u_2 + 1 \text{ and}$$
$$f_2(u_1, u_2) = 3u_1 + 2u_2 + 0.5\,.$$

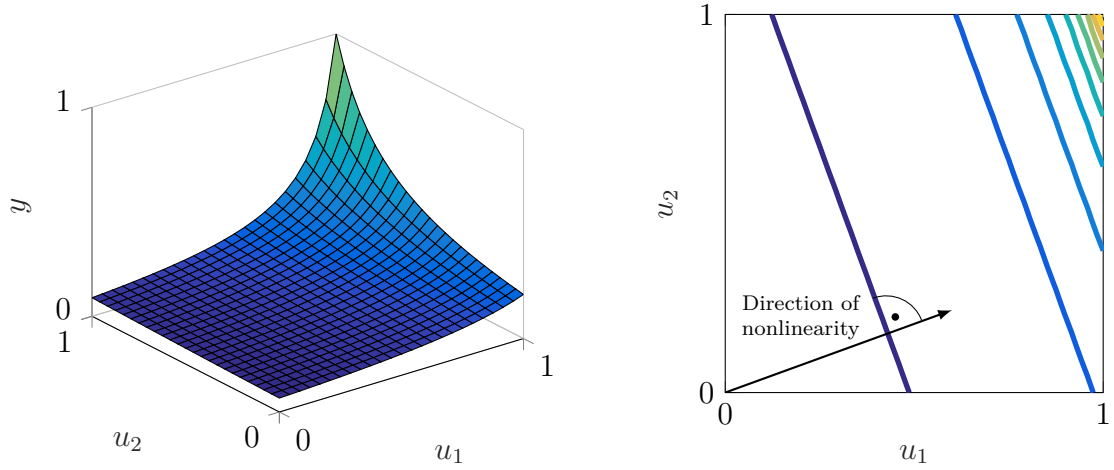
For TP2 the variables u_1 and u_2 are important to be included only in the $\underline{x}$-input space of an LMN whereas variables u_3 and u_4 are important to be included only in the $\underline{z}$-input space.

Test Process Three (TP3)

The third process is specifically designed to test the abilities of the embedded input selection approach presented in Section 3.4. Test process three (TP3) follows the equation

$$y(u_1, u_2) = \frac{0.1}{0.08 + 0.73(1 - u_1) + 0.27(1 - u_2)} \tag{3.4}$$

and is shown in Fig. 3.5. All inputs are important for both the $\underline{x}$- and $\underline{z}$-input space of an LMN but can be qualitatively rated regarding their nonlinear influence. The main direction of the nonlinearity is visualized in Fig. 3.5b and is orthogonal to the contour lines. From this direction it is obvious that input u_1 is more important to capture the nonlinearity than input u_2, because the angle between the u_1-axis and the nonlinearity is smaller than the angle between the u_2-axis and the nonlinearity. If the angle between the u_1-axis and the nonlinearity would be zero degrees, there would be no change in the process output in the u_2-direction, meaning that u_2 has no nonlinear influence on the output values at all.



(a) TP3 - surface plot

(b) TP3 - contour plot

Figure 3.5: Surface plot (a) and contour plot (b) of TP3

Test Process Four (TP4)

The fourth test process is a dynamic one and follows the equation:

$$y(k) = 0.1867 \arctan[u(k-1)] + 0.8187 y(k-1)\,. \tag{3.5}$$

The input values should be in the interval $[-3, 3]$ in order to create sufficiently strong saturation effects and therefore lead to fairly nonlinear behavior. Test process four (TP4) is a so-called Hammerstein process, which is a static nonlinearity followed by a linear dynamic system, as visualized in Fig. 3.6. Here, the nonlinearity is the arctangent function succeeded by a first order time-lag system. As can be seen from (3.5) the nonlinearity only affects the delayed input. Therefore, only this input should be needed in the $\underline{z}$-input space while the delayed output should only be needed

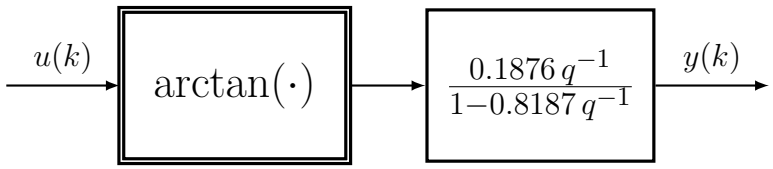
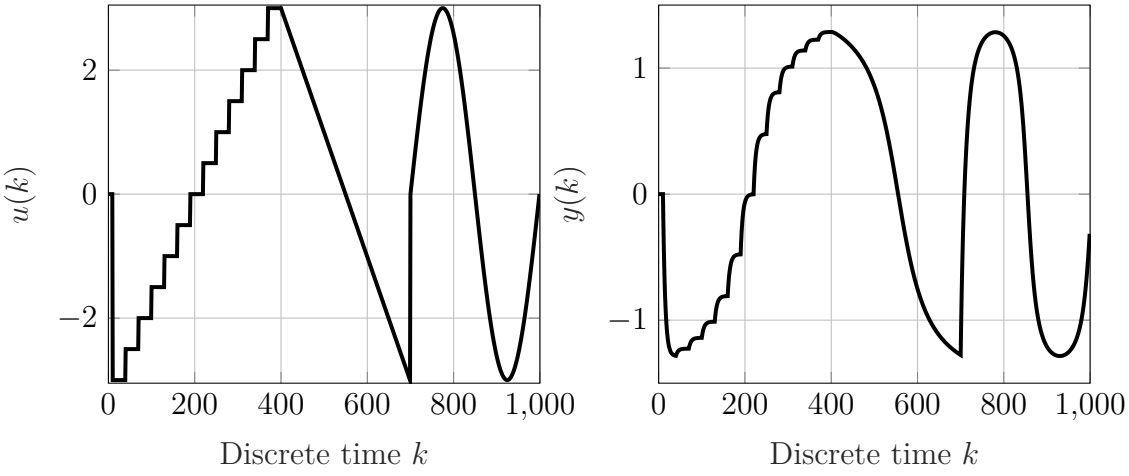


Figure 3.6: Block diagram of the Hammerstein system used as TP4

in the $\underline{x}$-input space. Figure 3.7a shows an excitation signal for TP4 and Fig. 3.7b the corresponding process output.

All test processes are used throughout the remaining sections of this chapter to show strengths and weaknesses of the presented input selection approaches. Whenever they are used, the number of data samples, possibly added noise, the distribution of the data, and other test conditions may vary and are mentioned explicitly. Note that not each test process is investigated for each input selection approach.



(a) Excitation signal $u(k)$ (b) Process output $y(k)$

Figure 3.7: Excitation signal (a) and the response of TP4 (b)

3.2 Mixed Wrapper-Embedded Input Selection Approach

In order to weaken the effects of the curse of dimensionality, a mixed wrapper-embedded input selection approach is presented that fully exploits a unique property of LMNs, namely the input space separation as explained in Section 2.3. As already stated in Section 2.4, the task of input selection can be formulated as a combinatorial, nonlinear, and discrete optimization problem. Due to the two arising input

spaces for LMNs, formulation (2.13) has to be changed to account for the increased complexity:

$$\begin{aligned} &\underset{\underline{x},\underline{z}}{\text{minimize}} && J(\underline{x},\underline{z}) \\ &\text{subject to} && \underline{x} \subseteq \mathbb{P} \\ & && \underline{z} \subseteq \mathbb{P}\,. \end{aligned} \tag{3.6}$$

J denotes an error measure of an LMN and therefore has to be minimized by choosing good $\underline{x}$- and $\underline{z}$-input subsets from all potential inputs $\mathbb{P}$. For the sake of simplicity it should be assumed in the following that all potential inputs $\mathbb{P}$ correspond to the physical inputs $\underline{u}$ unless otherwise mentioned. Note however, that in case of dynamic models the potential inputs $\mathbb{P}$ actually consist of filtered versions of the physical inputs $\underline{u}$ and outputs y combined in variable $\underline{\varphi}$, see (2.5) in Section 2.1. Theoretically (3.6) can be solved by just trying out all possible input subsets for the $\underline{x}$- and $\underline{z}$-input space and using the input combination that leads to the best model quality. Unfortunately this is not feasible even for a relatively moderate number of candidate inputs. Assuming that this number, equivalent to the cardinality of $\mathbb{P}$, is n_p, there are 2^{2n_p} possible input subsets, because each element in $\mathbb{P}$ can be assigned to the $\underline{x}$- and $\underline{z}$-input space individually. In order to find a solution in acceptable time, it is quite common to use heuristic search strategies that explore only parts of all possible input subsets, such as simple forward selection or backward elimination as described in more detail in Section 3.2.2.2.

The mixed wrapper-embedded approach tackles the optimization problem (3.6) by utilizing either LOcal LInear MOdel Tree (LOLIMOT) or HILOMOT as training algorithm and wraps the input selection around it. This is not just a wrapper approach because the possibility of both training algorithms to cope with the input space separation of LMNs is exploited. As illustrated in Fig. 3.8, a search strategy assigns physical inputs $\underline{u}$ to the $\underline{x}$- and $\underline{z}$-input space individually. With the chosen subsets $\underline{x} \subseteq \underline{u}$ and $\underline{z} \subseteq \underline{u}$ an LMN is trained. An evaluation criterion J_j is used to assess the model performance that is achieved with the subsets $\underline{x}_j$ and $\underline{z}_j$, belonging to model j. The evaluation criterion using measured (y) as well as predicted ($\hat{y}$) outputs, might incorporate information about the model complexity and can be used to guide the search for good $\underline{x}$- and $\underline{z}$-input subsets. Possible choices for the evaluation criterion that assess the model accuracy are, e.g., the k-fold cross-validation error or the validation error achieved on a separate data set not used for training.

As a result of the two input spaces, four different selection schemes are possible,

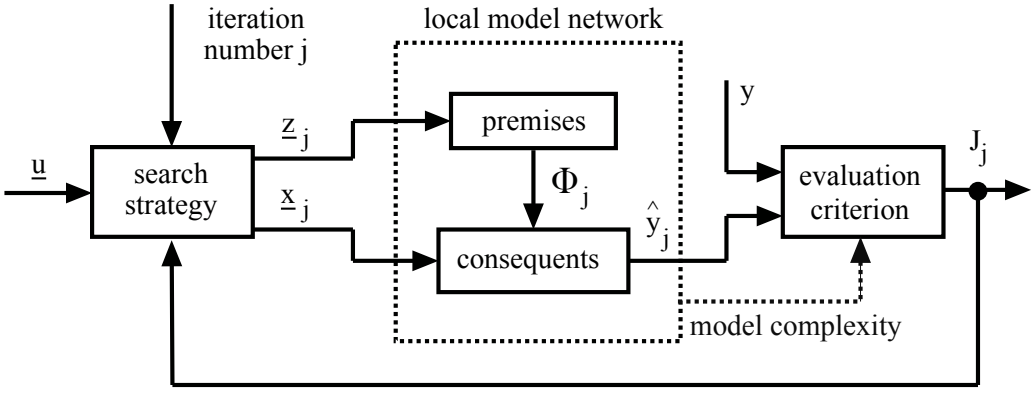


Figure 3.8: Block diagram of the mixed wrapper-embedded input selection approach. Because of the utilization of an LMN the search strategy has the possibility to assign the physical inputs $\underline{u}$ to the $\underline{x}$- and $\underline{z}$-input space individually.

which are visualized in Fig. 3.9 and explained in the following.

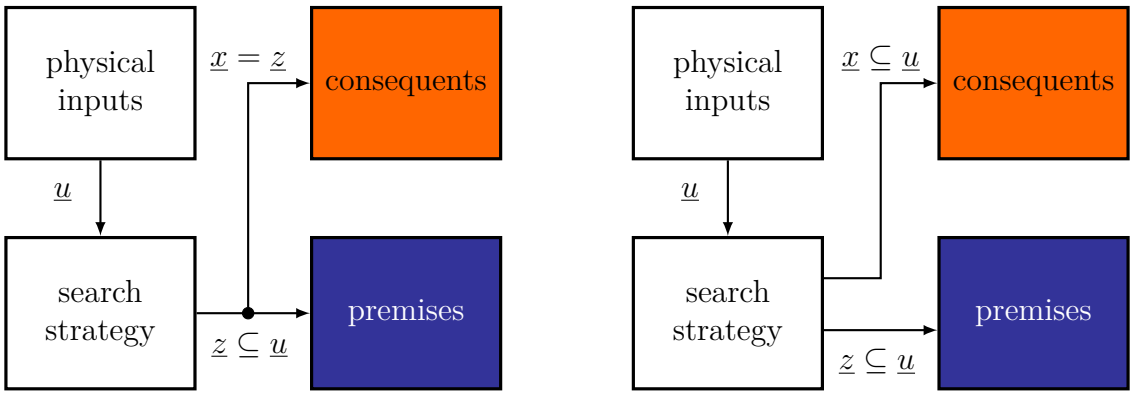
Linked $\underline{x}$-$\underline{z}$-input selection: The $\underline{x}$- and $\underline{z}$-input space contain the exact same physical inputs ($\underline{x} = \underline{z}$), which are a subset of all physical inputs as shown in Fig. 3.9a. This selection scheme does not exploit the input space separability and can therefore be done with any model type.

Separated $\underline{x}$-$\underline{z}$-input selection: The $\underline{x}$- and $\underline{z}$-input space are completely disjunct, meaning that both might contain arbitrary subsets of the physical inputs ($\underline{x} \subseteq \underline{u}$; $\underline{z} \subseteq \underline{u}$), see Fig. 3.9b. In this case the number of inputs to choose from is virtually doubled, because each physical input can be assigned to the $\underline{x}$- and $\underline{z}$-input space individually.

$\underline{x}$-input selection: Subsets of all physical inputs are only sought for the $\underline{x}$-input space, while the inputs contained in the $\underline{z}$-input space are kept fixed. Either the inputs for the $\underline{z}$-input space have to be chosen by expert knowledge, a former $\underline{z}$-input selection, or simply all physical inputs are chosen for the $\underline{z}$-input space as shown in Fig. 3.9c.

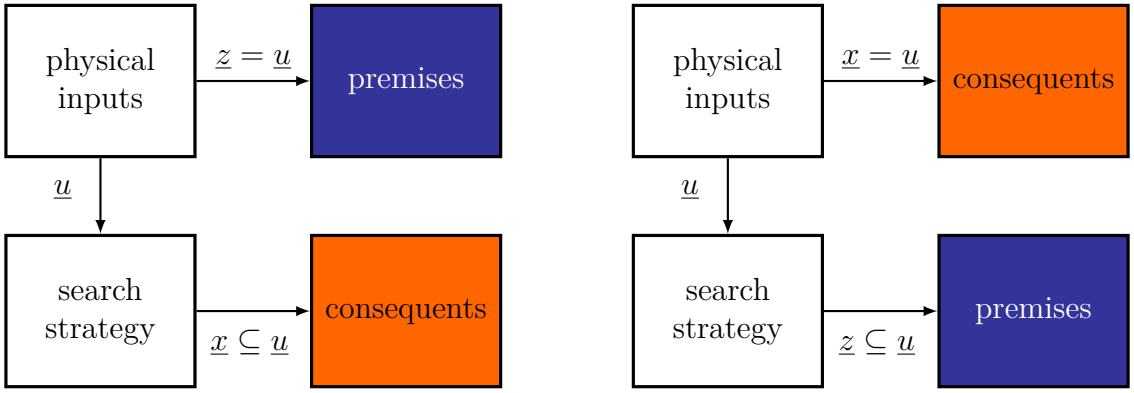
$\underline{z}$-input selection: Subsets of all physical inputs are only sought for the $\underline{z}$-input space, while the inputs contained in the $\underline{x}$-input space are kept fixed. Either the inputs for the $\underline{x}$-input space have to be chosen by expert knowledge, a former $\underline{x}$-input selection, or simply all physical inputs are chosen for the $\underline{x}$-input space as shown in Fig. 3.9d.

Which of the presented selection schemes should be used depends on the problem at hand. The most flexible one is the separated $\underline{x}$-$\underline{z}$-input selection for which the fewest



(a) Linked $\underline{x}$-$\underline{z}$-input selection

(b) Separated $\underline{x}$-$\underline{z}$-input selection



(c) $\underline{x}$-input selection

(d) $\underline{z}$-input selection

Figure 3.9: All possible selection schemes for the $\underline{x}$- and $\underline{z}$-input space

prior knowledge is necessary. For dynamic systems the $\underline{z}$-input selection might be of special interest if the structure of the dynamic (local) models is known from first principles. In this case the $\underline{z}$-input selection is able to reveal important inputs for the definition of the operating points. The $\underline{x}$-input selection as well as the linked $\underline{x}$-$\underline{z}$-input selection are of rather low interest for this thesis. In case of the $\underline{x}$-input selection the influence of harmful inputs - in terms of the bias/variance tradeoff - can be weakened by regularization techniques easily as a promising alternative to selection. The linked $\underline{x}$-$\underline{z}$-input selection does not bring anything new scientifically, because it is just a wrapper method that could be pursued with any model type.

Section 3.2.1 uses the test processes described in Section 3.1 and carries out first investigations with the separated $\underline{x}$-$\underline{z}$-input selection. Through the artificial demonstration examples the abilities of the mixed embedded-wrapper input selection approach should be emphasized. In Section 3.2.2 extensive simulation studies based on the function generator presented in Section 2.7 are performed to compare several

search strategies and evaluation criteria.

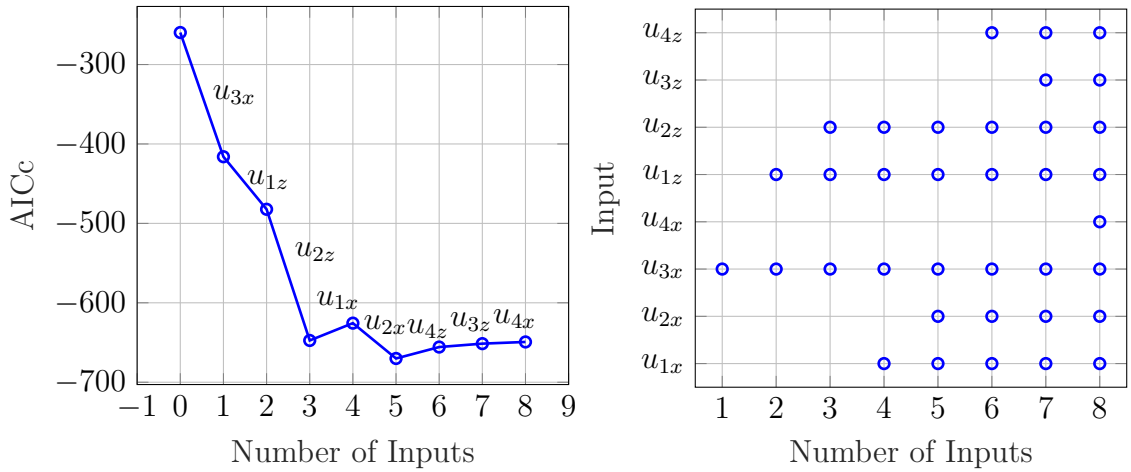
3.2.1 Investigation with Test Processes

The mixed wrapper-embedded input selection approach is tested with TP1 and TP2 in this section. A separated $\underline{x}$-$\underline{z}$-input selection is carried out for both test processes with backward elimination as the search strategy and Akaike's information criterion ($\mathrm{AIC_c}$) as evaluation criterion. More details about backward elimination and the $\mathrm{AIC_c}$ can be found in Section 3.2.2. HILOMOT serves as training algorithm for the LMNs, which is described in Section 2.3.

Test Process One

As already described in Section 3.1 it is known a-priori which inputs are important for the $\underline{x}$- and $\underline{z}$ input space for TP1. In particular, inputs u_1 and u_2 are important for both the $\underline{x}$- and $\underline{z}$-input space. Input u_3 is only important for the $\underline{x}$-input space, while input u_4 is only noise and is therefore not needed in any of the two input spaces. For this investigation a maximin optimized Latin Hypercube (LH) design with $N = 100$ samples is generated. Since input u_4 consists only of noise, no additional noise is added to the outputs. This data set is used to train the LMNs and to calculate the $\mathrm{AIC_c}$ values.

Figure 3.10 shows the result of the mixed wrapper-embedded input selection approach for TP1. The evaluation criterion is plotted against the number of inputs, which is simply the added number of variables contained in the $\underline{x}$- and $\underline{z}$-input space. Because the search strategy is a backward elimination, the graph can be read from right to left. In between two subsequent $\mathrm{AIC_c}$ values the input that is discarded in the corresponding backward elimination step is denoted. The index of the input u indicates the removal from the $\underline{x}$- or $\underline{z}$-input space, respectively. In Fig. 3.10b the subsets for a specific number of inputs are illustrated explicitly. For example, if the number of inputs is four, the inputs contained in the $\underline{x}$-input space are u_1 and u_3. The inputs contained in the $\underline{z}$-input space are u_1 and u_2. This illustration is not necessary for the backward elimination search strategy since the contained information is already included in Fig. 3.10a. However, for other search strategies, such as exhaustive search, this figure is necessary to represent the subsets belonging to a specific number of inputs. Since lower $\mathrm{AIC_c}$ values correspond to higher



(a) Evaluation criterion (b) Selected inputs

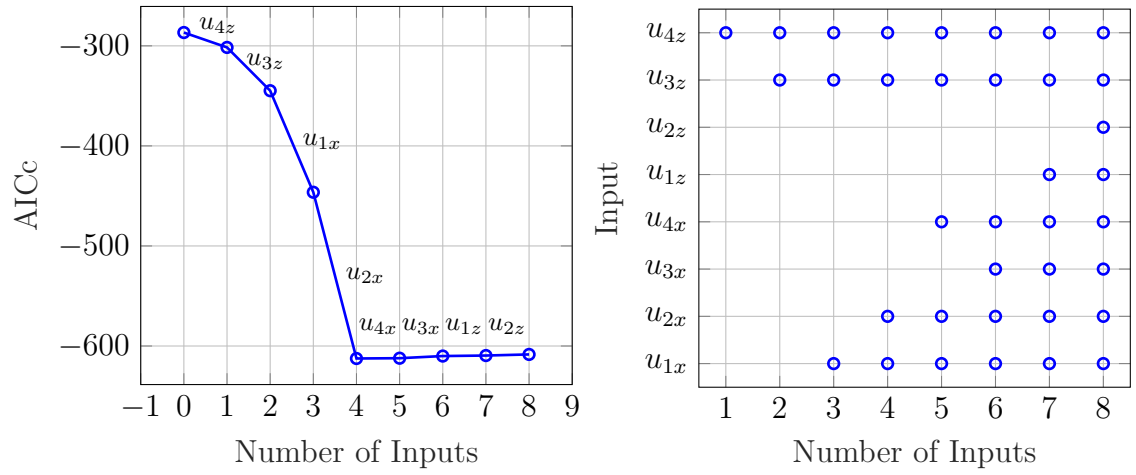
Figure 3.10: Evaluation criterion (a) and selected inputs (b) versus the number of inputs for test process one

model qualities, the best LMN is obtained if five inputs are chosen. Three of these inputs are contained in the $\underline{x}$-input space (u_1, u_2, u_3) and two are contained in the $\underline{z}$-input space (u_1, u_2), see Fig. 3.10b. Therefore, the optimal result according to prior-knowledge is obtained.

Test Process Two

TP2 has also four inputs which are virtually doubled due to the separated $\underline{x}$-$\underline{z}$-input selection. For this process u_1 and u_2 are important to be included only in the $\underline{x}$-input space while u_3 and u_4 are only necessary in the $\underline{z}$-input space. As already described in Section 3.1, u_3 and u_4 are discrete operating point variables that are either equal to 0 or 1. The design of experiments (DoE) consists of four two-dimensional maximin optimized LH designs, one for each possible combination of the two discrete inputs. Each LH design contains 25 samples such that the total number of data samples is $N = 100$. Zero mean white Gaussian noise is added to the process output such that the signal-to-noise ratio (SNR) equals 20 dB.

Figure 3.11 shows the results for TP2. Again the expected result is obtained. The best model quality is achieved with four inputs, yielding exactly the optimal solution. Remarkable in Fig. 3.11a is the continuous slight model improvement while discarding superfluous inputs and the sharp drop in performance as soon as important inputs are thrown out.



(a) Evaluation criterion (b) Selected inputs

Figure 3.11: Evaluation criterion (a) and selected inputs (b) versus the number of inputs for test process two

In summary, it is shown that the mixed wrapper-embedded input selection approach is able to assign variables to the $\underline{x}$- and $\underline{z}$-input space correctly for TP1 and TP2. All types of variables occur in these two test processes. Variables that are important in both the $\underline{x}$- and $\underline{z}$-input space, variables that are only important in one of the two input spaces and variables that are unimportant for both input spaces. The next section investigates the influence of the chosen search strategy and evaluation criterion on the input selection result.

3.2.2 Extensive Simulation Studies

The goal of this section is to provide recommendations for the choice of a search strategy in combination with an evaluation criterion. Investigated criteria are described in more detail in Section 3.2.2.1 and incorporate the error on a distinct validation data set, cross-validation, and Akaike's corrected information criterion (AIC_c). Investigated search strategies are forward selection (FS), backward elimination (BE), an exhaustive search (ES), and a genetic algorithm (GA). All search strategies are described in more detail in Section 3.2.2.2. Combinations of these search strategies with evaluation criteria are compared in terms of the achieved model performance and the computation time needed to find the presumably best subset of inputs. The function generator described in Section 2.7 is used to randomly generate several static test processes. With the help of the test processes training data sets and test data sets are created. Input selection is performed only with the training data sets.

As a result a presumably best subset of inputs is found for each search strategy in combination with a specific evaluation criterion. With the determined inputs a model is trained and for reference its model performance is assessed with the help of the test data set. The assessed model performances can eventually be compared.

The function generator is used to randomly generate 100 static test processes for input dimensionalities $p = 2, 3, \ldots, 10$ and training sample sizes of $N = 100$, $N = 500$, and $N = 1000$. The used training data sets are created by maximin LH designs generated with the extended deterministic local search (EDLS) algorithm proposed in [35] and briefly described in this thesis in Section 4.2.2. Zero mean white Gaussian noise is added to the outputs of the training data sets such that the SNR equals 30 dB. For each input dimensionality one Sobol sequence is used for creating the test data sets. The number of test data samples is $N_t = 10^5$, independent of the input dimensionality. No noise is added to the outputs in case of the test data sets.

For the investigations herein the most flexible selection scheme is used, which is the separated $\underline{x}$-$\underline{z}$-input selection. Since all inputs can be assigned to both the $\underline{x}$- and $\underline{z}$-input space, the number of inputs to choose from is virtually doubled as already discussed in Section 3.2. HILOMOT is used as training algorithm to build the LMNs with the subset of inputs selected by the particular search strategy, see Fig. 3.8 for a block diagram of the whole input selection procedure. Local affine models are used and the model complexities (number of local models) of the LMNs are determined by Akaike's corrected information criterion.

3.2.2.1 Evaluation Criteria

As already mentioned, the investigated evaluation criteria that are used to find good subsets of inputs incorporate the error on validation data, cross-validation (CV), and the $\mathrm{AIC_c}$. Each evaluation criterion is shortly explained in the following.

Validation Data

The usage of a separate validation data set to assess the model quality has already been mentioned in Section 2.2. Here, the error on the validation data set is only used as evaluation criterion to guide the input selection. The amount of validation data is chosen to be 20 % of the overall available training data. This means that in the CV

and AIC_c cases, 100% training data is available while for the validation-error-based criterion only 80% training data remains.

k-fold Cross-Validation

The number of folds for the k-fold CV is chosen to be $k = 10$ for this investigation following the recommendations in the publications [19] and [75]. Besides the number of folds, k represents also the number of runs that are necessary to assess the quality of a model. In each run one model is trained with $N - N/k$ data points that are randomly selected. Errors of the trained model are calculated on the left-out N/k data samples as visualized in Fig. 3.12. It is assured that each data sample is selected only once for the model quality assessment throughout all runs. When all runs are finished, all calculated errors on the left-out data points can be used to form a model quality measure, like the normalized root mean squared error (NRMSE), see (2.6). Since all data samples have been used for the model training as well as for the model quality assessment, k-fold CV uses the available data very efficiently. The price to be paid is the roughly $k + 1$ times higher computational cost. Note that after the best subset of inputs is determined according to the 10-fold CV error measure, a model is trained incorporating all available data.

Akaikes's Information Criterion

Akaike's corrected information criterion (AIC_c) is a combination of the error on training data and an added complexity penalty as already described in Section 2.2.

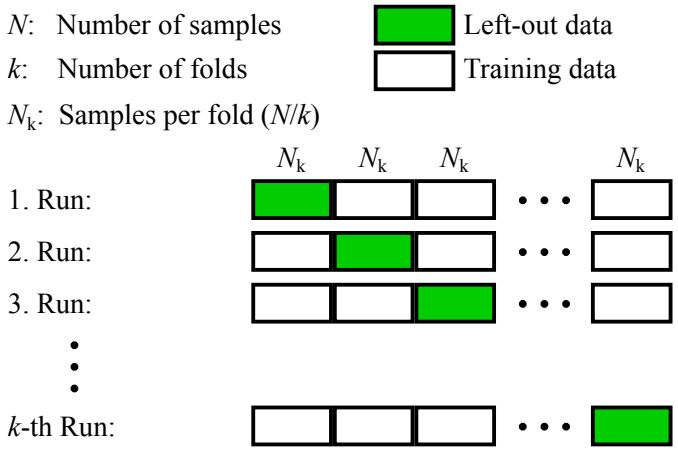


Figure 3.12: Schematic procedure of the k-fold CV

Note that the complexity penalty is proportional to both the number of local models and the number of inputs. Therefore, the AIC_c should be suited to compare models with different input subsets and should be able find a good bias/variance tradeoff.

The advantage of the AIC_c and the CV as evaluation criterion compared to the usage of a separate validation data set is that they use all available data for the training. In contrast to the 10-fold CV the AIC_c is computationally much less demanding.

3.2.2.2 Search Strategies

Four different search strategies are compared, which are forward selection (FS), backward elimination (BE), exhaustive search (ES), and a genetic algorithm (GA). FS starts with an empty subset of inputs and adds one input in each iteration of the search algorithm. Once one input is added it cannot be removed afterwards. In each iteration all not yet selected inputs are tentatively tested to be added, a model is trained, and the input leading to the best evaluation criterion is eventually selected. This is done until all inputs are selected for the presented investigations in this section.

In contrast to that, BE starts with all inputs and in each iteration of the search algorithm one of the inputs is discarded from the input subset. Once an input is removed from the subset, it cannot be added afterwards. To decide which input variable should be removed at the current iteration, each of the remaining input variables is tentatively discarded from the input subset and a model is trained. The evaluation criterion for each removed input is calculated and the input, that yields the best evaluation criterion is picked for removal at the current iteration. For the investigations in this section, the search algorithm stops after all inputs are removed.

The ES simply tries out all possible combinations of inputs. For each possible combination a model is trained and the evaluation criterion is calculated. Because the number of models that have to be trained grows exponentially with the number of potential inputs, this search strategy is only used up to $p = 5$ inputs. Because the separated $\underline{x}$-$\underline{z}$-input selection is used, the number of possible input combinations and therefore the number of necessary models to be trained reaches already $2^{2 \cdot 5} = 1024$.

The GA is a global optimization algorithm, meaning that it is theoretically capable of finding the globally best solution of a nonlinear optimization problem [96]. The

GA is inspired by natural evolution in which life forms evolve over generations in order to adapt to drastically changing environmental conditions. GAs work with populations of individuals, where each individual represents one solution to the optimization problem [88]. A so-called fitness function rates all individuals such that better solutions get a higher score. This score is important for the selection process, in which individuals have better chances to survive or to pass on parts of their genes to the next generation the higher their fitness score is. In that way individuals evolve over time to better solutions. Individuals selected to pass on their genes to the next generation of individuals change because of mutation or recombination - other genetic operators are not used for this investigation. More detailed information about GAs in general can be found in [88] or [77].

Within this section, each individual represents one subset of inputs. Therefore, each individual is a bit-string, containing just ones and zeros indicating if an input is part of the subset (1) or not (0). This GA-typical binary coding is particularly straightforward for combinatorial optimization tasks like input selection. For the calculation of the fitness scores the evaluation criteria from Section 3.2.2.1 are used. Note that for all evaluation criteria lower values are better and therefore get higher fitness scores. The mutation rate is set to 0.1, while the recombination rate is set to 0.8. The population size is chosen to be five times the number of potential inputs. The GA stops either after three generations without any improvement in the best fitness score of all individuals or after a maximum calculation time of 24 hours.

FS and BE both follow a rather simple heuristic and finding the best combination of inputs is by no means guaranteed. However, these search strategies are often used because of their tractability and the reasonably good results that are achieved with them. In case of ES the best combination of inputs is guaranteed to be included in all subsets that are tried out. Nonetheless, the optimal subset might not be recognized as the best combination of inputs if the used evaluation criterion fails to do so. As mentioned in [110], one possible reason is overfitting. This does not mean that each individual model overfits. It means that the selection itself induces overfitting. If the number of different input subsets that are compared based on a specific evaluation criterion becomes huge, chances increase to achieve the best score for an objectively not optimal subset just by chance. Being a nonlinear global optimizer, the GA has also the potential to find the best possible combination of inputs for the problem at hand and should therefore be capable to produce the same results as the ES. Anyhow, due to time limitations or other user-specified restrictions the results of the ES and GA do not need to be the same.

3.2.2.3 A-Priori Considerations

Before the comparison starts, some thoughts are provided about what can be expected. Deviations from these expectations are considered to be interesting results and worth to be discussed in some more detail. At first expectations about the evaluation criteria are summarized.

Using validation data as evaluation criterion is considered to perform worst in choosing good subsets of inputs. The number of samples used for the actual training is the smallest in this case. Additionally, the validation data set is used very often which increases the danger in the selection procedure of overfitting. The more models are tested on the validation data set, the more likely it becomes to assess one of these models as good just by chance and not because it has a really good generalization performance. This danger increases with an increasing number of tested models. Therefore the risk of observing this kind of overfitting is the highest for the ES and the GA search strategy.

The 10-fold cross-validation is considered to perform best in choosing good subsets of inputs. The available data is utilized in the most efficient way. Because not only one hold-out data set is used, the danger of overfitting should be far smaller compared to the validation data set. Additionally, the generalization performance is measured based on data not used for the training opposed to just adding some complexity penalty as it is done in case of the AIC_c.

For the AIC_c as evaluation criterion it is hard to make an educated guess about the expected performance in choosing subsets of inputs. On the one hand, the AIC_c appears to be rather simple by just augmenting the error on training data with a complexity penalty. On the other hand, this criterion shows very good results in choosing an appropriate model complexity for LMNs as shown in [56].

In case of the used search strategies it is expected that the BE finds on average better input subsets than the FS, because all interactions between inputs of the test functions are observable from the start. Especially in early stages of the FS these interactions might be hidden and more inputs are presumably needed to cover them. It is not straight away predictable if the rather simple greedy search strategies, which are FS and BE, perform better than the more extensive ones, which are ES and GA. On the one hand FS and BE are known to be less prone to overfitting [81], on the other hand, a lot of possibly good input subsets are never explored. Here, a strong

interdependency with the used evaluation criterion is likely to be present. Therefore, no a-priori guess exists about the ranking of the used search strategies.

3.2.2.4 Comparison Results

The results of the extensive simulation studies are used to compare all possible combinations of search strategies and evaluation criteria regarding the obtained model accuracies and required computation times. Additionally, the benefit of the mixed wrapper-embedded input selection (MWEIS) approaches compared to using all available inputs is pointed out. Another result of these studies is the *effective* dimensionality of the used test functions created with the function generator. Therefore the number of inputs that is actually chosen by the MWEIS approaches is reviewed.

The relative improvement RI defined as

$$\mathrm{RI} = \frac{\mathrm{NRMSE}_{\mathrm{all}} - \mathrm{NRMSE}_{\mathrm{subset}}}{\mathrm{NRMSE}_{\mathrm{all}}}\,, \tag{3.7}$$

is used to demonstrate the benefit of the MWEIS approach compared to using all available inputs. The NRMSE values are calculated on the test data, either with all available inputs (index *all*) or with the subset suggested by one of the MWEIS approaches (index *subset*). Because lower NRMSE values indicate better model performances, positive RI values indicate improvements gained through the MWEIS while deteriorations are indicated by negative RI values. The relative improvements gained through the input selection are also used to compare all possible combinations of search strategies and evaluation criteria with each other. The reference point is always the accuracy of a model using all inputs and is therefore the same for all MWEIS approaches. Thus, the biggest improvement corresponds to the best model accuracy in absolute terms.

Figure 3.13 shows the median relative improvements of all MWEIS approaches for all training data sizes and two input dimensionalities, which are $p = 6$ and $p = 10$. The median is determined over the 100 different test functions. The shown input dimensionalities are chosen because the advantages of the MWEIS approaches become clear from $p = 6$ on and increase with an increasing p until the highest improvements are obtained with $p = 10$. For input dimensionalities lower than 6 ($p < 6$) the median relative results look like Fig. 3.13c. No significant improvements are observable for the 10-fold cross-validation and the $\mathrm{AIC_c}$ while the model accuracies with subsets obtained with the validation error as evaluation criterion are deteriorated. In fact,

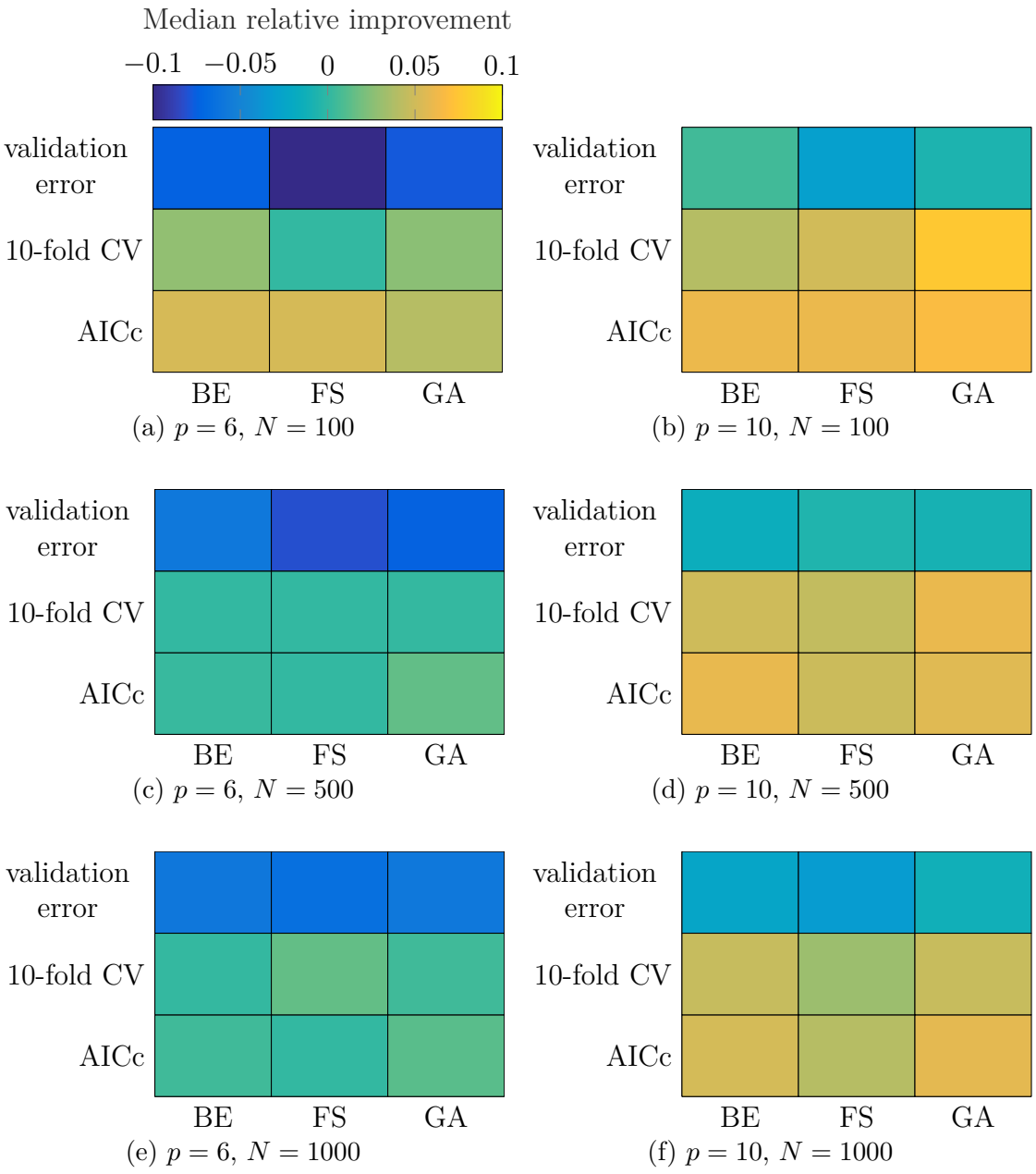


Figure 3.13: Median relative improvements for all combinations of search strategies and evaluation criteria for input dimensionalities $p = 6$ and $p = 10$ with training data set sizes $N = 100, 500$, and 1000

subsets chosen by the validation error as evaluation criterion have almost never a substantial advantage when compared to models using all inputs. The AIC_c for the MWEIS turns out to be the best choice as evaluation criterion among the tested ones. In most cases the model accuracies obtained with input subsets chosen by the AIC_c and the 10-fold cross-validation are on the same level. The advantage of the AIC_c increases with a decreasing input dimensionality and decreasing training data sizes.

The BE and the GA are the best search strategies, even if the advantage compared to the FS is quite small in most cases. No results of the ES are presented in Fig. 3.13, since ES is only used for smaller input dimensionalities (up to $p = 5$). The results obtained by the ES are comparable to the outcomes of the GA for input dimensionalities $p = 2, \ldots, 5$.

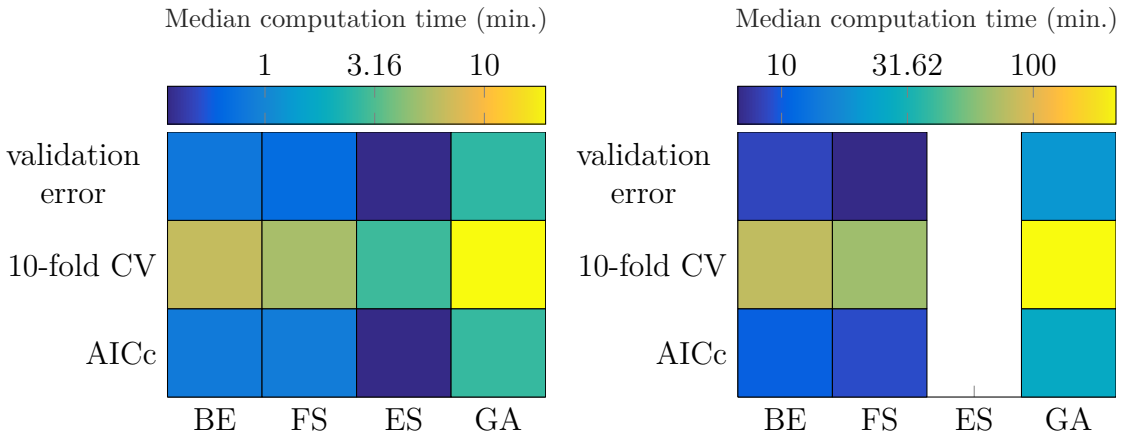
Unfortunately, it is not possible to make meaningful statements about the variations from the median test errors for different combinations of search strategies and evaluation criteria. There are two sources for the variations in the achieved test errors:

1. The 100 different test functions. Even if all MWEIS approaches would find the optimal subset of inputs for each of the 100 test functions, there would still be a certain amount of variation due to various complexities in the test functions. The exact amount of variation can not be quantified and thus no reference exists.

2. Possibly suboptimal subsets increase the range of achieved test errors and thus the degree of variation.

For the comparison of different search strategies with different evaluation criteria only the second source of variation is of interest. Since there is only one noise realization for each test process, it is not possible to determine the variation that comes from a specific combination of a search strategy with an evaluation criterion.

The required computation times of all MWEIS approaches are compared in Fig. 3.14 for input dimensionalities $p = 2$ and $p = 10$ with a training data size of $N = 1000$. Note that Fig. 3.14a and Fig. 3.14b are both scaled logarithmic but differently. Again, the median is taken over the number of different test functions. For the shown data amount the highest computation times occur. In general, the required computation time increases with an increasing input dimensionality as well as with an increasing amount of training samples. As expected, the 10-fold CV requires the highest computation times among all evaluation criteria. In general, using the



(a) $p = 2$, $N = 1000$ (b) $p = 10$, $N = 1000$

Figure 3.14: Required median computation times for all combinations of search strategies and evaluation criteria for input dimensionalities $p = 2$ (a) and $p = 10$ (b) with $N = 1000$

validation error as evaluation criterion requires slightly less computation time than the AIC_c. Note that even though the maximum calculation time is limited to 24 hours, the highest median computation time is only 210 minutes ($p = 10$, $N = 1000$, 10-fold CV, GA). Since the maximum required computation time at all is 9.5 hours, it can be concluded that the stopping criterion for all GA runs is always the number of generations without any improvement in the best fitness score (chosen to be 3).

Typically the following ordering occurs for the search strategies regarding the required computation time in ascending order: FS, BE, ES, and GA. A remarkable exception that can be seen in Fig. 3.14a occurs for $p < 4$ where the ES search strategy requires the least computation time among all search strategies. This is due to the fact that LMNs for all possible subsets of inputs can be trained simultaneously if enough CPU cores are available. In contrast to that the FS and BE search strategies add/eliminate one input in each iteration. Until it is known which input is added/discarded in the current iteration, no further results are calculated in parallel. Waiting on these intermediate results leads to higher computation times even though the computational effort is lower compared to the ES search strategy.

Figure 3.15 shows the median input subset size for all MWEIS approaches for selected input dimensionalities and a training data set size of $N = 100$. Input dimensionality $p = 4$ is selected to be shown because it is the lowest input dimensionality for which differences in the chosen input subset sizes become clear. Then, the number of

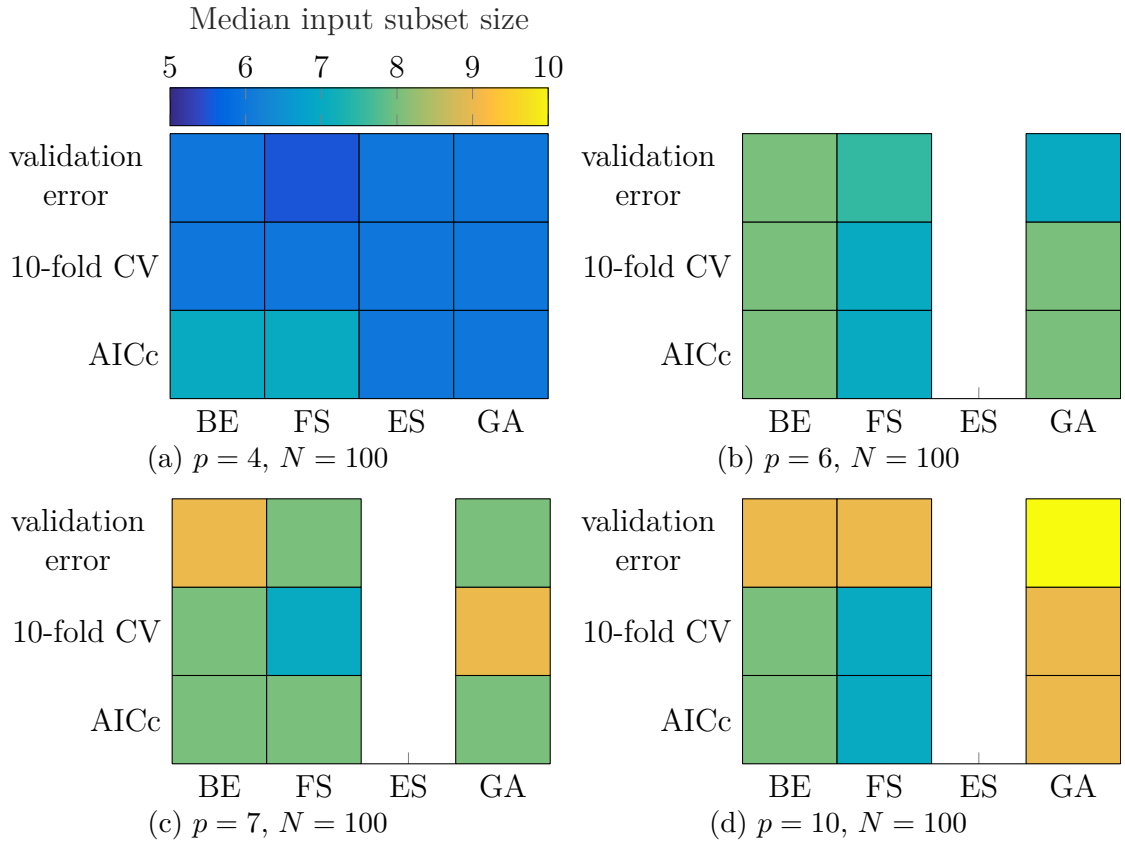


Figure 3.15: Median values of the suggested input subset sizes of all combinations of search strategies and evaluation criteria for input dimensionalities $p = 4, 6, 7$, and 10 with $N = 100$

inputs that are chosen increases with an increasing input dimensionality until input dimensionality $p = 7$. It is interesting to see that there is no significant increase in the number of selected inputs if the input dimensionality is further increased, compare Fig. 3.15c with Fig. 3.15d. It seems as if the *effective* dimensionality of the test functions generated with the function generator saturates. Note that due to the used separated $\underline{x}$-$\underline{z}$-input selection there are potentially 20 inputs that can be selected if the input dimensionality is $p = 10$.

Qualitatively similar results are obtained for the other two investigated training data set sizes $N = 500$ and $N = 1000$, but with generally bigger input subset sizes. Figure 3.16 shows how the median subset size increases with the number of samples for input dimensionalities $p = [2, 3, 5, 8, 9, 10]$. Median values are taken over the 100 different test functions with BE as search strategy and the $\mathrm{AIC_c}$ as evaluation criterion. For rather low-dimensional input spaces the subset size does not further increase with more available training data. For example, with a two-dimensional

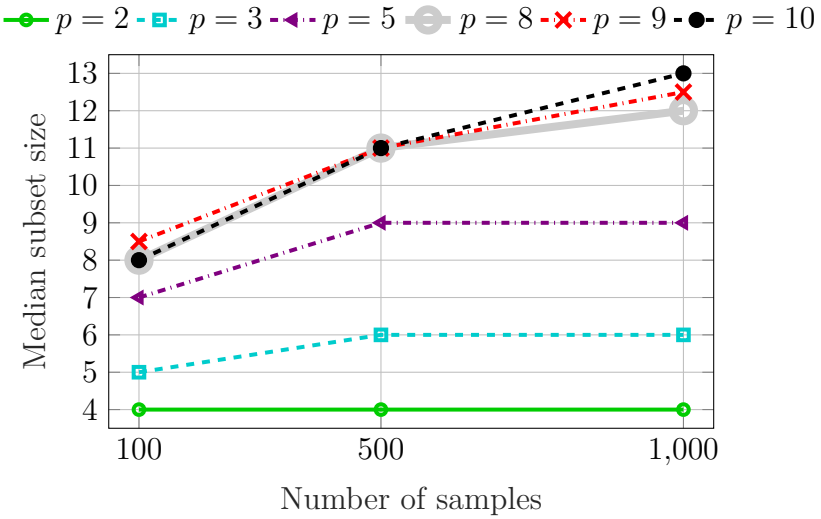


Figure 3.16: Median subset sizes for different input dimensionalities over all 100 test functions for BE as search strategy and AIC_c as evaluation criterion

input space ($p = 2$) a training data set size of $N = 100$ already covers the input space well and therefore carries sufficient information such that in terms of a good bias/variance tradeoff all beneficial LMN inputs could be selected. Therefore, no more increase in the subset size can be expected if the number of samples is increased. In contrast to that, the input space for $p = 10$ is almost empty in case of $N = 100$ samples. As a result the input subset size is kept rather small in order to achieve a good bias/variance tradeoff. As more information becomes available, i.e. N increases, the best bias/variance tradeoff is shifted to more complex models leading to bigger input subset sizes.

Typically the lowest number of inputs is chosen by the FS and BE search strategies, while GA chooses more inputs to be included in the suggested subset. The ES search strategy is typically on the same level as FS and BE. Comparing the investigated evaluation criteria, the ascending ordering regarding the number of typically chosen inputs is AIC_c, 10-fold cross-validation, and the validation error.

In summary, the presented results lead to the recommendation of using the BE search strategy combined with the AIC_c as evaluation criterion. The required computation time is only lower if FS is used, while the achieved model accuracies are typically among the best of all search strategies when BE is used. In some cases the GA leads to slightly better model accuracies compared to the BE but therefore typically chooses more inputs and requires more computation time. The AIC_c is recommended as evaluation criterion because it leads to model accuracies that are only seldom

outperformed by input subsets selected by 10-fold cross-validation. The number of chosen inputs is typically low compared to the other evaluation criteria and the required computation time is also among the lowest.

3.3 Regularization-Based Input Selection Approach

A normalized L1 regularization for axis-oblique partitioning strategies is described, which has been previously proposed by the author [12]. In contrast to commonly used L1 regularization techniques, only the amount of obliqueness incorporated in the input space partitioning is penalized. In principle, this approach can be applied to any axis-oblique partitioning strategy based on nonlinear optimization techniques. In this section the focus lies on an implementation for HILOMOT.

Axis-oblique partitioning strategies weaken the effects of the curse of dimensionality significantly compared to axis-orthogonal partitioning strategies. However, the additional flexibility comes along with an increased variance error. The normalized L1 regularization should account for the increased variance error and might be able to determine the most important input variables. Since only the partitioning is affected by the proposed regularization, only input variables in the $\underline{z}$-input space are selected. Additional effort is necessary to find the best subset of inputs for the $\underline{x}$-input space, e.g. a subsequent $\underline{x}$-input selection.

The main idea of regularization-based input selection is to introduce a penalization of high absolute values of the model parameters $\underline{\theta}$ such that these are pushed towards zero [132, 57]. Once a model parameter reaches zero, the input associated with it can be discarded. The whole loss function that has to be minimized is compounded of a part related to the model error $\underline{e} = \begin{bmatrix} e(1) & e(2) & \cdots & e(N) \end{bmatrix}^T$ and a part related to the sum of absolute values of the model parameters $\underline{\theta}$:

$$J_{L1} = \frac{1}{N} \underline{e}^T(\underline{\theta}) \underline{e}(\underline{\theta}) + \lambda ||\underline{\theta}||_1 \,. \tag{3.8}$$

The regularization strength is determined by λ, with $\lambda \geq 0$. By constantly increasing the regularization parameter λ more and more inputs are discarded and the model becomes less flexible. Typically, the regularization parameter λ is tuned such that the generalization performance of the model is maximized. This corresponds to the best bias/variance tradeoff.

Before details about the normalized L1 regularization are presented in Section 3.3.2, more details about HILOMOT are provided in Section 3.3.1. These information incorporate details about the validity construction and the corresponding splitting parameters. Eventually, Section 3.3.3 demonstrates the abilities of the normalized L1 regularization implemented for HILOMOT.

3.3.1 More Detailed HILOMOT Fundamentals

In the following some more details about the construction of the validity functions Φ in HILOMOT are given. These information are necessary to understand the concept of the regularization-based input selection approach. Note that this approach selects only inputs in the $\underline{z}$-input space. The validity of each local model generated with HILOMOT is constructed with the help of sigmoid splitting functions:

$$\Psi(\underline{z}) = \frac{1}{1 + e^{-\kappa(\underline{z}^T \cdot \underline{v} - v_0)}} \,. \tag{3.9}$$

The splitting parameters are the offset term v_0 and the vector $\underline{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_{n_z} \end{bmatrix}^T$ determining the distance from the origin and the direction of the split, respectively. The steepness of a sigmoid function in the transition area is usually determined by the norm of the vector that contains all splitting parameters $\underline{v}^* = \begin{bmatrix} v_0 & \underline{v}^T \end{bmatrix}^T$. This is visualized in Fig. 3.17. In case of HILOMOT the parameter κ is introduced to control the smoothness explicitly. Therefore the vector $\underline{v}^*$ is always normalized to a length of one. Because of the symmetric properties of $\Psi(\cdot)$, its complementary

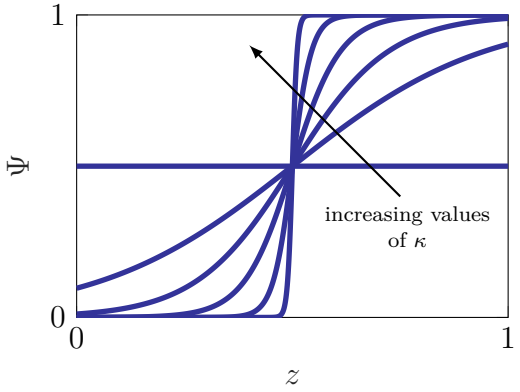


Figure 3.17: Explanation of the smoothness of a sigmoid function

function can be written as:

$$\widetilde{\Psi}(\underline{z}) = 1 - \Psi(\underline{z}) = \Psi(-\underline{z})\,. \tag{3.10}$$

The splitting functions and their complementary counterparts are linked in a multiplicative, hierarchical way in order to generate the validity functions of the LMNs constructed with HILOMOT:

$$\Phi_i(\underline{z}) = 1 \cdot \prod_{j \in P_i} \Psi_j(\underline{z}) \cdot \prod_{k \in \widetilde{P}_i} \widetilde{\Psi}_k(\underline{z})\,. \tag{3.11}$$

The sets P_i and $\widetilde{P}_i$ contain the indices of all splitting and complementary splitting functions that have to be multiplied in order to get the i-th validity function. Figure 3.18 illustrates this construction with the help of a binary tree. Each circle represents a node; nodes that are not further split are called leafs and correspond to the validity functions with which the local models are weighted. All splitting functions on the way from the root to a leaf have to be multiplied to obtain the final validity function. Because of the chosen splitting functions the partition of unity

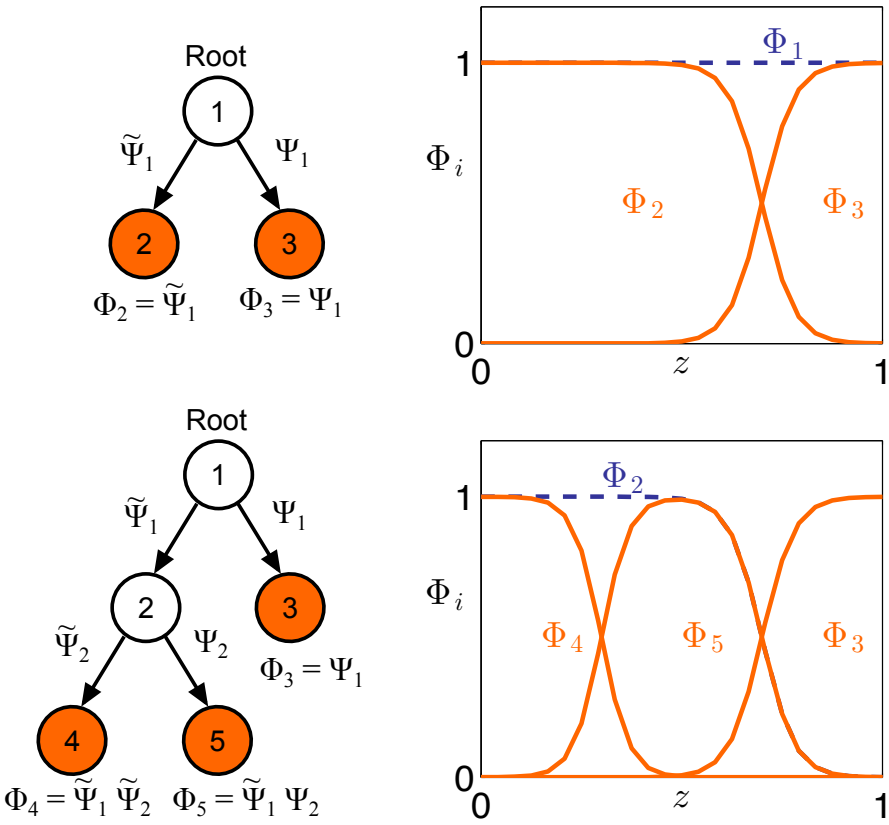
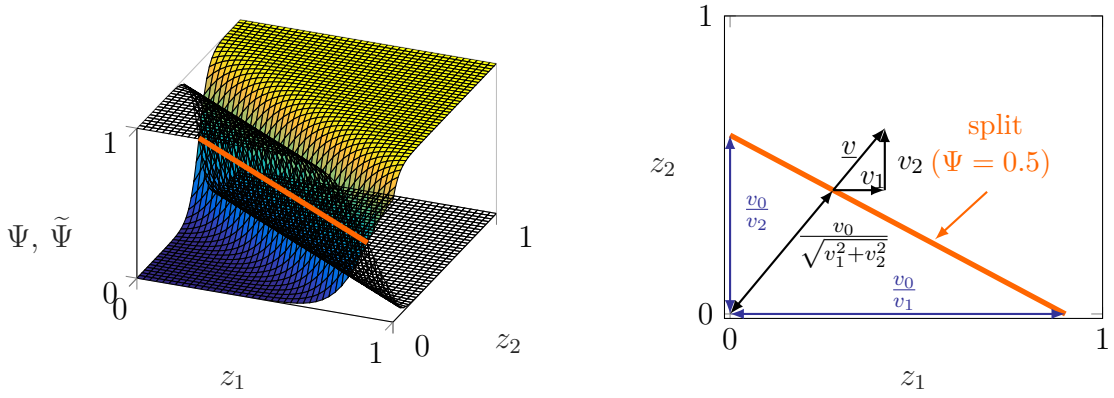


Figure 3.18: Construction of the validities of LMNs generated with HILOMOT

holds automatically [97]. The first and second split of the input space partitioning are shown in Fig. 3.18. Note that the validity function Φ_1 corresponds to one globally valid model and is therefore one everywhere in the $\underline{z}$-input space.

The meaning of the parameters contained in $\underline{v}$ is illustrated for a two-dimensional $\underline{z}$-input space in Fig. 3.19. A sigmoid and its complementary counterpart are visualized in Fig. 3.19a, while Fig. 3.19b shows the top-view on the z_1-z_2 plane together with the split and the interpretation of the splitting parameters. The orientation of the split is determined by the ratio of the splitting parameters $\underline{v}$ with respect to each other. The split orientation is important for the LMN since it determines the direction in which its slope is able to change if local affine models are used. In this case changes in the slope of the LMN can only be realized by switching to another local model and vector $\underline{v}$ points directly to the adjacent model. If one variable contained in the $\underline{z}$-input space has no nonlinear influence its corresponding splitting parameter is zero. Figure 3.20 shows three splits corresponding to special cases of the splitting parameters $\underline{v}$. Two orthogonal splits are shown in Fig. 3.20a and 3.20c as well as one split where all splitting parameter values are equal, here $v_1 = v_2$ shown in Fig. 3.20b.

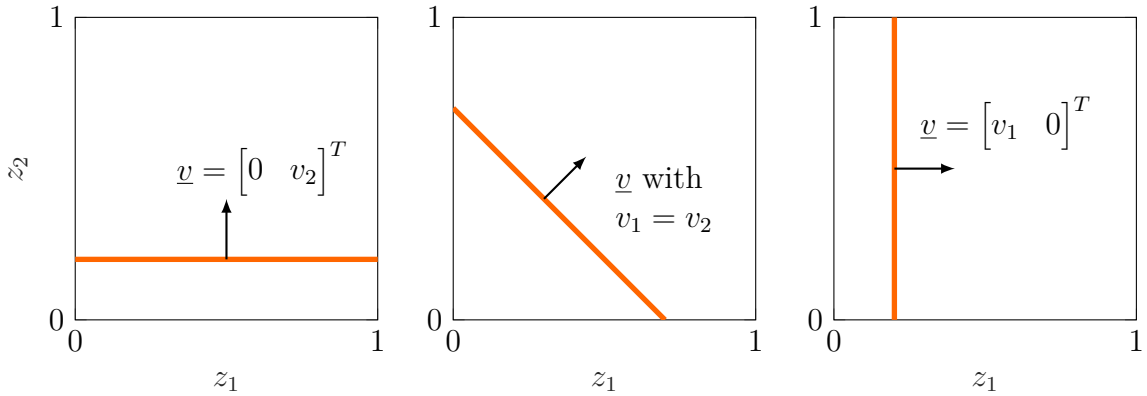
If the obliqueness of a split should be penalized - as it is the goal of the normalized L1 split regularization described in the following section - the sigmoid splitting parameters $\underline{v}$ should be pushed towards zero. The offset parameter v_0 is not regularized by this method, which means that the location of the split does not influence the penalty.



(a) Sigmoid and complementary function

(b) Topview on z_1-z_2 plane

Figure 3.19: Explanation of splitting parameters



(a) Split orientation: 0° (b) Split orientation: 45° (c) Split orientation: 90°

Figure 3.20: Three special cases of the sigmoid splitting parameters $\underline{v}$ without the offset parameter v_0

3.3.2 Normalized L1 Split Regularization

The main goal of the split regularization is to reduce the model variance introduced by the split optimization and to get a low-dimensional $\underline{z}$-input space. Therefore the regularization should favor axis-orthogonal splits, while an oblique partitioning should only be pursued if there is a significant performance gain. In order to achieve these goals, the following penalty term is proposed:

$$J_p(\underline{v}) = \frac{\lambda}{\sqrt{p_z} - 1} \left(\left\| \frac{\underline{v}}{\|\underline{v}\|_2} \right\|_1 - 1 \right) . \tag{3.12}$$

It is a 1-norm, where the splitting parameters $\underline{v}$ are normalized by their 2-norm. The regularization parameter is λ. The remaining adjustments, i.e. the subtraction of one and the division by $\sqrt{p_z} - 1$, lead to the scaling of the penalty term to the interval $[0, 1]$, if $\lambda = 1$. p_z is the number of variables contained in the $\underline{z}$-input space. The maximum of the penalty term is reached if all entries in $\underline{v}$ have the same absolute value. Its minimum is reached if all but one entries are zero, which corresponds to an axis-orthogonal split. The penalty term for $\lambda = 1$ is visualized for a two-dimensional $\underline{z}$-input space in Fig. 3.21. Note that the splitting parameter vector $\underline{v}$ does only include elements determining the orientation of the split, see Section 3.3.1, and therefore only the amount of obliqueness is penalized. The position in the $\underline{z}$-input space does not affect the penalty term (3.12).

In order to improve the performance of the nonlinear optimization the analytical gradient of the whole loss function is required. Assuming the NRMSE as unregularized

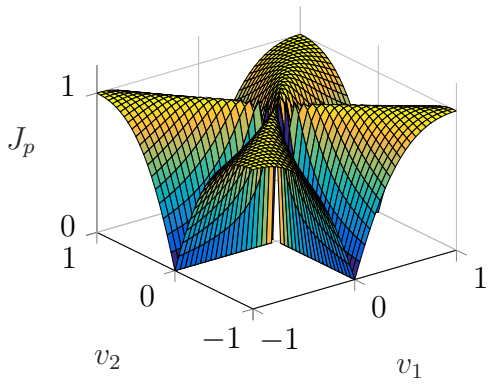


Figure 3.21: Penalty term for a two-dimensional $\underline{z}$-input space

loss function, the whole objective including the regularization term becomes

$$J(\underline{\theta}_{LM}, \underline{v}) = \sqrt{\frac{\sum_{i=1}^{N}(y(i) - \hat{y}(i, \underline{\theta}_{LM}, \underline{v}))^2}{\sum_{i=1}^{N}(y(i) - \bar{y})^2}} + J_p(\underline{v})\,, \tag{3.13}$$

with the measured outputs $y(i)$, the mean of all measured outputs $\bar{y}$, and the LMN output $\hat{y}(i)$ depending on the affine local model parameters $\underline{\theta}_{LM}$ and the splitting parameters $\underline{v}$. In [42] the analytical gradient for the unregularized case has already been derived. Therefore only the derivative of J_p with respect to the splitting parameters is derived here. A differentiable approximation for the absolute value of a given number has to be used for the calculation of the 1-norm. We use the smooth approximation proposed in [117]. The absolute value of a number x is approximately:

$$|x| \approx \frac{1}{\alpha}\left(\log\left(1 + \mathrm{e}^{-\alpha x}\right) + \log\left(1 + \mathrm{e}^{\alpha x}\right)\right)\,. \tag{3.14}$$

The approximation will be denoted as $|x|_\alpha$. Here, α is a parameter that determines how close the approximation comes to the true absolute value. As stated in [117] and demonstrated in Fig. 3.22, $|x|_\alpha$ converges to $|x|$ as α approaches infinity. With this approximation the penalty term becomes:

$$\begin{aligned} J_p(\underline{v}) =& \frac{\lambda}{\sqrt{p_z} - 1}\left(\left\|\frac{\underline{v}}{\|\underline{v}\|_2}\right\|_{1\alpha} - 1\right) \\ =& \frac{\lambda}{\sqrt{p_z} - 1}\left[\sum_{i=1}^{p_z}\frac{1}{\alpha}\left(\log\left(1 + \mathrm{e}^{-\alpha\frac{v_i}{\|\underline{v}\|_2}}\right) + \log\left(1 + \mathrm{e}^{\alpha\frac{v_i}{\|\underline{v}\|_2}}\right)\right) - 1\right]\,, \end{aligned} \tag{3.15}$$

where $\|\cdot\|_{1\alpha}$ is the sum of approximated absolute values of the corresponding vector elements. In [117] this is called *Smooth L1 Approximation Method*. Variable p_z denotes the number of inputs in the $\underline{z}$-input space. It follows the detailed derivation

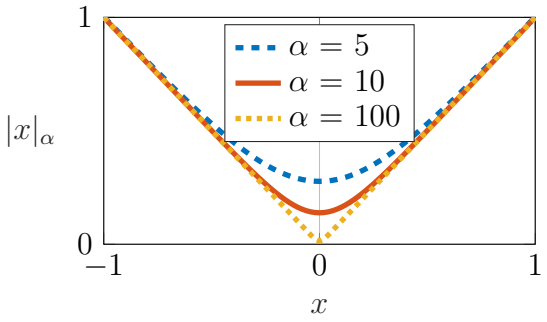


Figure 3.22: Approximation $|x|_\alpha$ of the absolute value $|x|$ for three different parameters of α

of the analytical gradient for splitting parameter v_j. At first the derivative of the SmoothL1 approximation with respect to an arbitrary variable x is derived:

$$\begin{aligned}\frac{\mathrm{d}|x|_\alpha}{\mathrm{d}x} &= \frac{1}{\cancel{\alpha}}\left(\frac{-\cancel{\alpha}\cdot \mathrm{e}^{-\alpha x}}{1+\mathrm{e}^{-\alpha x}}\cdot\frac{\mathrm{e}^{\alpha x}}{\mathrm{e}^{\alpha x}} + \frac{\cancel{\alpha}\cdot \mathrm{e}^{\alpha x}}{1+\mathrm{e}^{\alpha x}}\right)\\ &= \frac{\mathrm{e}^{\alpha x}-1}{\mathrm{e}^{\alpha x}+1} = \frac{\mathrm{e}^{\frac{2\alpha x}{2}}-1}{\mathrm{e}^{\frac{2\alpha x}{2}}+1} = \tanh\left(\frac{\alpha x}{2}\right).\end{aligned} \tag{3.16}$$

Another necessary inner derivative concerns the multiplicative inverse of the 2-norm with respect to one splitting parameter v_j:

$$\begin{aligned}\frac{\mathrm{d}\left(\sum_{i=1}^{p_z} v_i^2\right)^{-1/2}}{\mathrm{d}v_j} &= -\frac{1}{\cancel{2}}\left(\sum_{i=1}^{p_z} v_i^2\right)^{-3/2}\cdot\cancel{2}\cdot v_j\\ &= \frac{-v_j}{\left(\|\underline{v}\|_2\right)^3}.\end{aligned} \tag{3.17}$$

In the following, it is assumed that v_j is the last entry in the vector $\underline{v}$, i.e. $j = p_z$. This assumption simplifies the handling of summation indices without the loss of generality. With the help of (3.16), (3.17), and the chain rule, the partial derivative with respect to splitting parameter v_j equals:

$$\begin{aligned}\frac{\partial J_p}{\partial v_j} =& \frac{\lambda}{\sqrt{p_z}-1}\left[\left(\sum_{i=1}^{p_z-1}\tanh\left(\frac{\alpha v_i}{2\|\underline{v}\|_2}\right)\frac{v_i\cdot(-v_j)}{\|\underline{v}\|_2^3}\right)+\ldots\right.\\ &\left.\ldots+\tanh\left(\frac{\alpha v_j}{2\|\underline{v}\|_2}\right)\left(\frac{1}{\|\underline{v}\|_2}+\frac{v_j\cdot(-v_j)}{\|\underline{v}\|_2^3}\right)\right]\\ =& \frac{\lambda}{\sqrt{p_z}-1}\left[-v_j\left(\sum_{i=1}^{p_z}\tanh\left(\frac{\alpha v_i}{2\|\underline{v}\|_2}\right)\frac{v_i}{\|\underline{v}\|_2^3}\right)+\tanh\left(\frac{\alpha v_j}{2\|\underline{v}\|_2}\right)\frac{1}{\|\underline{v}\|_2}\right].\end{aligned} \tag{3.18}$$

The proposed penalization might get into trouble if all parameters get close to zero. In practice this is of minor concern, since it corresponds to a split which would have an infinite distance to the origin, see Fig. 3.19b. If such a split would result from the optimization it simply could be omitted.

As already mentioned, the main goal is to get a low-dimensional $\underline{z}$-input space and eliminate all unnecessary variables therein. A shrinkage of the remaining $\underline{z}$-inputs is not desired. Therefore the penalization is only used to determine the unnecessary $\underline{z}$-inputs. Afterwards an unregularized split optimization is done with all left-over $\underline{z}$-inputs. How many splitting variables are kept depends on the size of the regularization parameter λ. Therefore λ can be used to smoothly transform the axis-oblique ($\lambda \to 0$) to an axis-orthogonal ($\lambda \to \infty$) partitioning strategy.

An important task is to find a regularization parameter λ that leads to a good bias/variance tradeoff. Throughout the training of an LMN several values for λ are needed since in each iteration of HILOMOT a split optimization is carried out. For each of these optimizations a different value for λ might be optimal in terms of the bias/variance tradeoff. Two different approaches for the determination of λ values are investigated and compared in this thesis.

The first approach follows a simple heuristic. Typically, the model performance saturates with an increasing amount of local models because each additional split acts on a smaller subarea of the $\underline{z}$-input space. To account for this saturation an initial regularization parameter λ is weighted with the normalized *effective* number of data samples for the current split:

$$\lambda_{split} = \frac{\lambda}{N} \cdot \sum_{i=1}^{N} \Phi_{worstLM}(\underline{z}(i)) \,. \tag{3.19}$$

The effective number of data samples is the number of points contained in the local model that is currently subdivided by the split. This local model is denoted as worst local model (index *worstLM*), since its local error measure is worst compared to all other local models. The number of points in the worst local model can be calculated by the summation of the validity values $\Phi_{worstLM}$ of all N training data points. Through the normalization with N (3.19) measures the fraction of all data that is inside the local model to be split and it is guaranteed that $\lambda_{split} = \lambda$ in the first split. Then, for each following split the regularization of the current split is decreased according to the number of points that are affected by this split. For the initial regularization parameter λ several values have to be tested. For each of these

regularization parameters an LMN is trained with activated split regularization using the heuristic adaption of λ (3.19) for each split. The resulting LMNs can then be compared in terms of the achieved model quality revealing the best of the tested regularization parameters.

The second approach for the determination of the regularization parameter optimizes λ for each split. This is computationally more expensive than the first approach but has obviously the potential to outperform it. The objective for the optimization is an approximation of the leave-one-out (LOO) error that has to be minimized. For the approximative LOO error the partitioning is assumed to be known and fixed. This simplification helps to decrease the computational load tremendously since the problem is a linear estimation problem once the partitioning is known. In fact the LOO error can be calculated directly based on one least squares solution with the help of the smoothing matrix $\underline{S}$ as shown in [113]. This matrix describes the direct relationship between the measured process outputs $\underline{y}$ and the model outputs $\hat{\underline{y}}$:

$$\hat{\underline{y}} = \underbrace{\underline{X}\overbrace{\left(\underline{X}^T\underline{X}\right)^{-1}\underline{X}^T\underline{y}}^{\underline{\theta}_{LS}}}_{\underline{S}}, \tag{3.20}$$

with the regression matrix $\underline{X}$ and the model parameters $\underline{\theta}_{LS}$ obtained by least squares. Hartmann [56] derived an equation to directly calculate the LOO error for LMNs under the assumption of a fixed partitioning if the parameters of the local models are estimated by a locally weighted least squares estimation scheme:

$$\mathrm{LOOE} = \sum_{j=1}^{N} \frac{y(j) - \hat{y}(j)}{1 - \sum\limits_{k=1}^{M} S_{jj,k}}\,. \tag{3.21}$$

N denotes the number of training data samples, M is the number of local models. $S_{jj,k}$ is the entry in row and column j (diagonal entries) of the smoothing matrix $\underline{S}$ belonging to the parameter estimation of local model k. For more details about the derivation the reader is referred to [56]. The approximative LOO error is chosen as objective function for the optimization of the regularization parameter λ because it measures the generalization performance of the LMN while the computational costs are tolerable.

3.3.3 Investigation with Test Processes

Test processes one and four are used to demonstrate the abilities and shortcomings of the regularization-based input selection (RBIS) approach presented in the previous section. Both ways to determine the regularization parameter λ are tested, i.e. the heuristic from (3.19) and the optimization with respect to the approximation of the LOO error. Since the RBIS approach affects only the $\underline{z}$-input space, the results ideally reveal which inputs influence the output in a nonlinear way.

For the investigation of both test processes, the settings of the identification algorithm are identical. HILOMOT is used, but with the added penalty term for the split optimization, see (3.13). The smooth L1 approximation method with $\alpha = 650$ is used, such that an analytic gradient of the objective function can be provided, see (3.18). For the determination of the model complexity Akaike's corrected information criterion is used. Only affine local models are used. The usage of higher-order, locally valid polynomials would be counterproductive to the intended goal of the investigation. It should reveal nonlinearly influencing inputs. Having more complex local models could make splits along specific axes obsolete even if there is a nonlinear influence. For the heuristic determination of the split regularization parameter λ, four different initial values are tested which are 10^{-4}, 10^{-3}, 10^{-2}, and 10^{-1}. In addition to the RBIS approaches, LMNs are trained with the standard HILOMOT algorithm (no split regularization or $\lambda = 0$) and HILOMOT with deactivated split optimization (LOLIMOT-type of splitting). The deactivation of the split optimization results in an input space partitioning with only axis-orthogonal splits. Note that this is *not* equivalent to the RBIS approach with $\lambda = \infty$. In case the split regularization parameter is set to infinity, the split position can still be optimized. In contrast to this, HILOMOT with deactivated split optimization does not optimize the split position and simply splits the worst local model into two halves that both contain approximately the same amount of data samples.

Test Process One

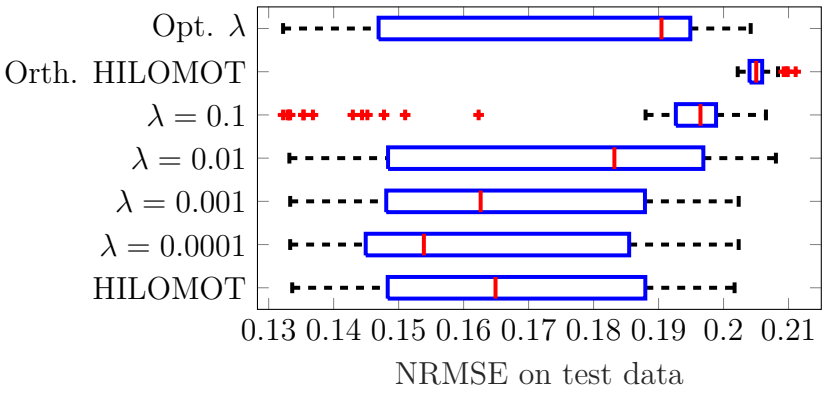
TP1 is static and therefore no dynamics have to be represented. The first and second input (u_1 and u_2) contribute nonlinearly to the process output, while input u_3 influences it only linearly. Input u_4 consists only of noise. Results obtained by the RBIS approach should indicate inputs u_1 and u_2 as nonlinear. Note that the

results of the RBIS approaches contain no information about the input's usefulness for the $\underline{x}$-input space.

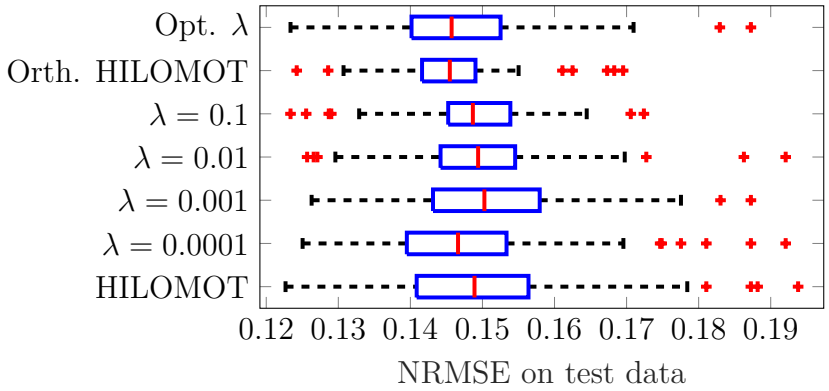
For the investigation of the RBIS approaches, 100 different training data sets are generated that only differ in the noise realization of the output values. The design of experiments is a maximin optimized LH design with $N = 100$ samples. Since input u_4 consists only of noise, no additional noise is added to the outputs. One Sobol sequence serves as test data set with $N_t = 10^5$ samples, which is used to assess the quality of the resulting models. Note that for the generation of the test data set input u_4 is neglected in order to create noise-free samples.

Because of the way the HILOMOT algorithm works, the best initial split does not necessarily lead to the best overall model performance when the training of the LMN is finished. In each iteration the worst local model is further split and only this new split is optimized, see Section 2.3 for a detailed description of HILOMOT. As a result the current split is optimal with respect to the existing partitioning, not incorporating any further changes in it. This is the reason why the value of already existing splits to the overall model performance can change during the training procedure. Having this in mind, it makes sense to look at the generalization performances with LMNs having just two local models in order to compare all RBIS approaches directly. In addition, the benefit of all distinct RBIS approaches for the generalization performance is of interest after the optimal model complexities are chosen (most likely leading to LMNs having more than just two local models).

Figure 3.23a shows boxplots of the achieved generalization performances of all investigated scenarios for LMNs with just two local models. Each boxplot contains the achieved generalization performances of the 100 different noise realizations. The red lines mark the median values for each method. The best of these median NRMSE values on test data is achieved with the heuristic and an initial value of the split regularization parameter of $\lambda = 10^{-4}$. The generalization performance achieved with the standard HILOMOT algorithm is only outperformed by the heuristic with initial split regularization parameters $\lambda = 10^{-4}$ and $\lambda = 10^{-3}$. From all RBIS approaches the optimization of the split regularization parameter turns out to be the second worst, performing only better than the heuristic with an initial split regularization parameter of $\lambda = 0.1$. The median of all optimized split regularization parameters is $\lambda_{split} = 0.09$ (not shown in the figure). The worst median NRMSE value on test data is achieved by HILOMOT with deactivated split optimization, abbreviated with *orth.* HILOMOT since all splits are axis-orthogonal. The lowest variations in the achieved



(a) Generalization performances with two local models



(b) Generalization performances with the optimal complexities

Figure 3.23: Histograms of the generalization performances of all investigated scenarios for two local models (a) and the optimal model complexity according to the AIC_c (b)

generalization performances occur for the orthogonal HILOMOT algorithm. The variations in the model performance are also very low for the heuristic with $\lambda = 0.1$. All other investigated scenarios are comparable in terms of their variations in the generalization performance.

Figure 3.23b shows boxplots of the achieved generalization performances where the model complexity of each LMN is set to its optimal value according to the AIC_c. All median NRMSE values on test data are very close to each other. The best values are achieved with the optimized split regularization parameter, the heuristic with $\lambda = 10^{-4}$, and the orthogonal HILOMOT algorithm. The variations in the generalization performances are the lowest for the orthogonal HILOMOT algorithm, followed by the heuristic with $\lambda = 0.1$ before the heuristic with $\lambda = 0.01$. Variations of the remaining scenarios are on a similar level.

Figure 3.24 shows the convergence of the median test errors over the number of

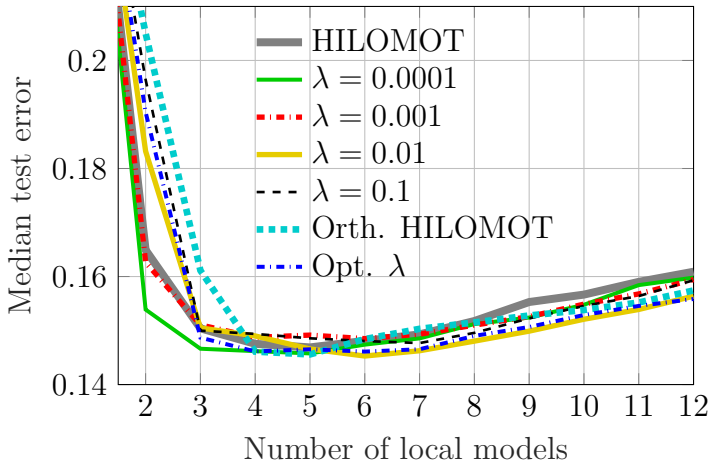
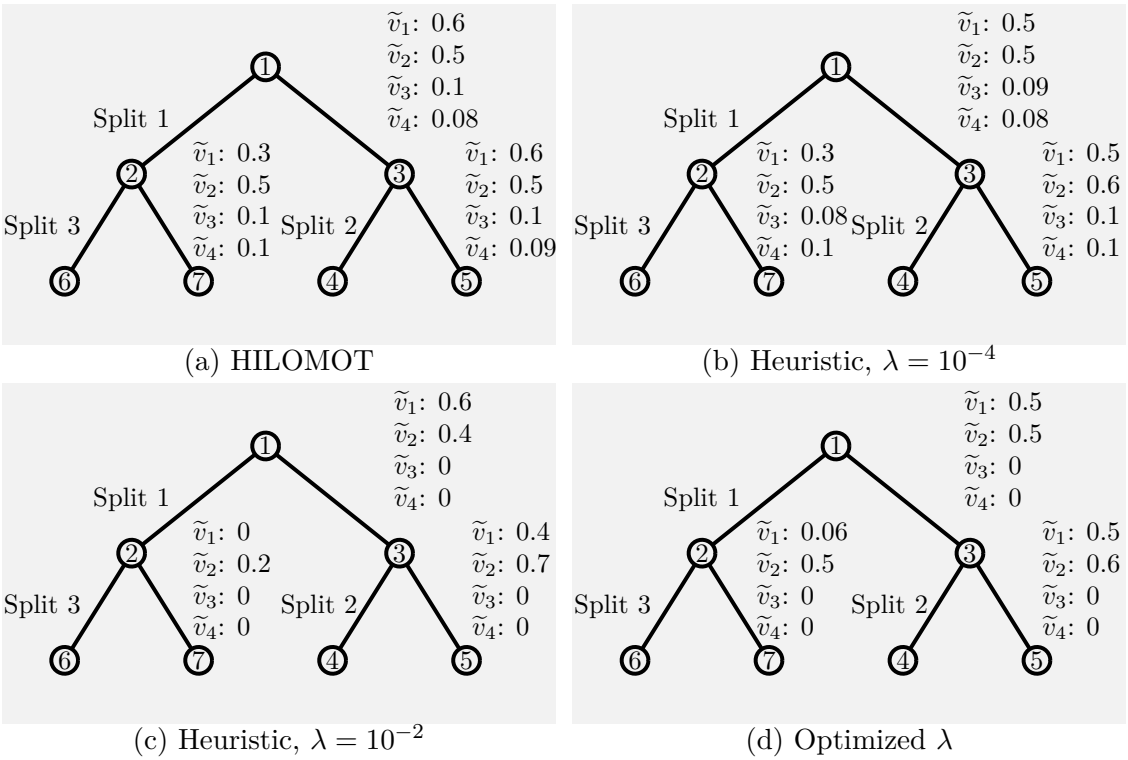


Figure 3.24: Median NRMSE values on test data over the number of local models

local models for all investigated scenarios. The heuristic with $\lambda = 10^{-4}$ converges the fastest, whereas the orthogonal HILOMOT algorithm has the slowest convergence rate.

Besides the generalization performance and the corresponding variations, it is interesting to see the effect of the RBIS approaches on the splitting parameters. Therefore the median splitting parameter values are shown for the first three splits in Fig. 3.25 for HILOMOT, the heuristic with $\lambda = 10^{-4}$ as well as $\lambda = 10^{-2}$, and the optimized λ values. Here, the median of the *absolute* splitting parameter values $\widetilde{v}_i$ is shown, such that a value of zero implies that for at least 50 of the 100 noise realizations the corresponding splitting parameter is zero. The information about how many splitting parameters are actually zero is important, since this corresponds to orthogonal splits. Without taking the absolute values, a zero median value would also result if the corresponding splitting parameter values would be normally distributed around zero. As a result of this, no conclusion about how many of the 100 noise realizations lead to orthogonal splits would be possible. The desired effect of all RBIS approaches is to push the splitting parameters corresponding to the third and fourth $\underline{z}$-input towards zero. These inputs have no nonlinear influence on the process output and splits in these directions are considered to be harmful for the generalization performance. Note that the binary trees shown in Fig. 3.25 are just for visualization purposes. Not all of the binary trees resulting from the 100 noise realizations have the exact same structure, e.g. split three might as well split node four or five further instead of node two.

Only small differences are observable when comparing the median absolute splitting parameter values of HILOMOT and the heuristic with $\lambda = 10^{-4}$ shown in Fig. 3.25a



(a) HILOMOT (b) Heuristic, $\lambda = 10^{-4}$

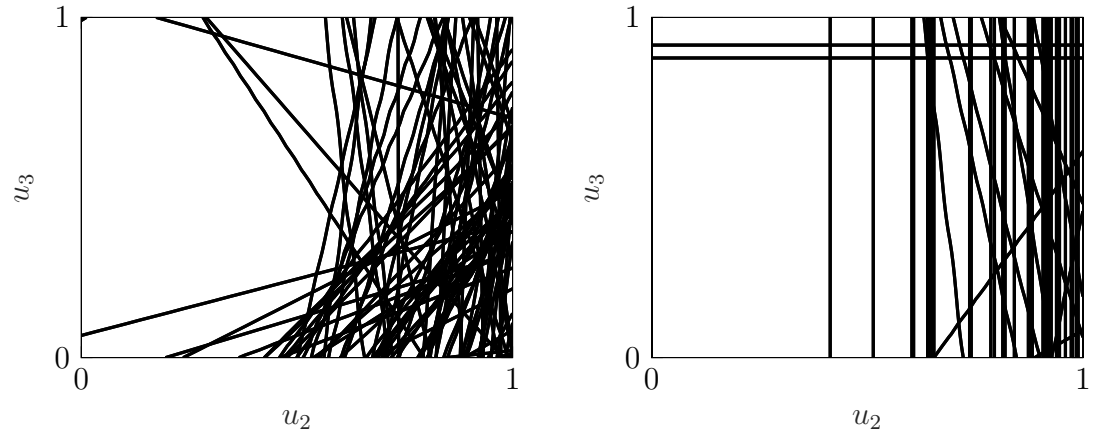
(c) Heuristic, $\lambda = 10^{-2}$ (d) Optimized λ

Figure 3.25: Binary trees together with the median of the absolute values of the splitting parameters $\widetilde{v}_i$ for HILOMOT (a), the heuristic with $\lambda = 10^{-4}$ (b) as well as $\lambda = 10^{-2}$ (c), and the optimized λ values (d) for splits one to three

and 3.25b, respectively. The desired effect of pushing v_3 and v_4 towards zero is only achieved if the initial split regularization parameter of the heuristic is raised to $\lambda = 10^{-2}$ and above. This desired effect is also achieved if the split regularization parameter λ is optimized, see Fig. 3.25d. The question arises why achieving the desired effect does not lead to higher model qualities and a significantly lower variance.

The variance in the model accuracies is likely not as significantly decreased as expected by the RBIS approaches due to the fact that there is still much variance incorporated in the split position. Figure 3.26a shows the first split for all 100 noise realizations in the u_2-u_3 subspace for the standard HILOMOT algorithm opposed to the ones obtained by the RBIS approach with optimized λ values depicted in Fig. 3.26b. Even though most splits are in fact orthogonal, there are a lot of different splits (varying positions) and also oblique splits.

A likely cause for the insignificantly increased generalization performances is visualized in Fig. 3.27. Histograms of the values of all four splitting parameters v_1, v_2, v_3,



(a) HILOMOT (b) Optimized λ

Figure 3.26: Input space partitionings of LMNs with two local models for all noise realizations in case of HILOMOT (a) and the RBIS approach with optimized λ (b) in the u_2-u_3 subspace

and v_4 are shown for the first split and all 100 noise realizations for HILOMOT and the RBIS approach with optimized λ. If a splitting parameter is zero, the split is parallel to the corresponding $\underline{z}$-input axis as explained in Section 3.3.1 and visualized in Fig. 3.20a. From an interpretation point of view this means the corresponding $\underline{z}$-input is irrelevant, since its value has no influence on the splitting function. As can be seen in Fig 3.26, the number of splits that are parallel to one of the $\underline{z}$-input axes is increased in general through the RBIS approach with optimized λ compared to the standard HILOMOT algorithm. In case of $\underline{z}$-inputs u_3 and u_4 this is desired and should improve the generalization performance of the models. However, the number of splits parallel to the $\underline{z}$-input axes u_1 and u_2 is also increased, which harms the generalization performance. Considering the interaction between u_1 and u_2 in TP1, the partitioning should ideally be oblique in the u_1-u_2 subspace.

Finally, the required training times of all investigated scenarios for TP1 are compared in Fig. 3.28. As expected, the smallest required training times are needed by the orthogonal HILOMOT algorithm and the largest ones are needed if the split regularization parameter is optimized. The higher the initial split regularization parameter is chosen for the heuristic determination method, the smaller the required training times are.

Despite the fact that the convergence rate of the heuristic with $\lambda = 10^{-4}$ is quite impressive, a significant advantage of using any of the RBIS approaches is not observable in case of the optimal model complexities. The median generalization performance as well as the deviations of it are only slightly improved and do not pay

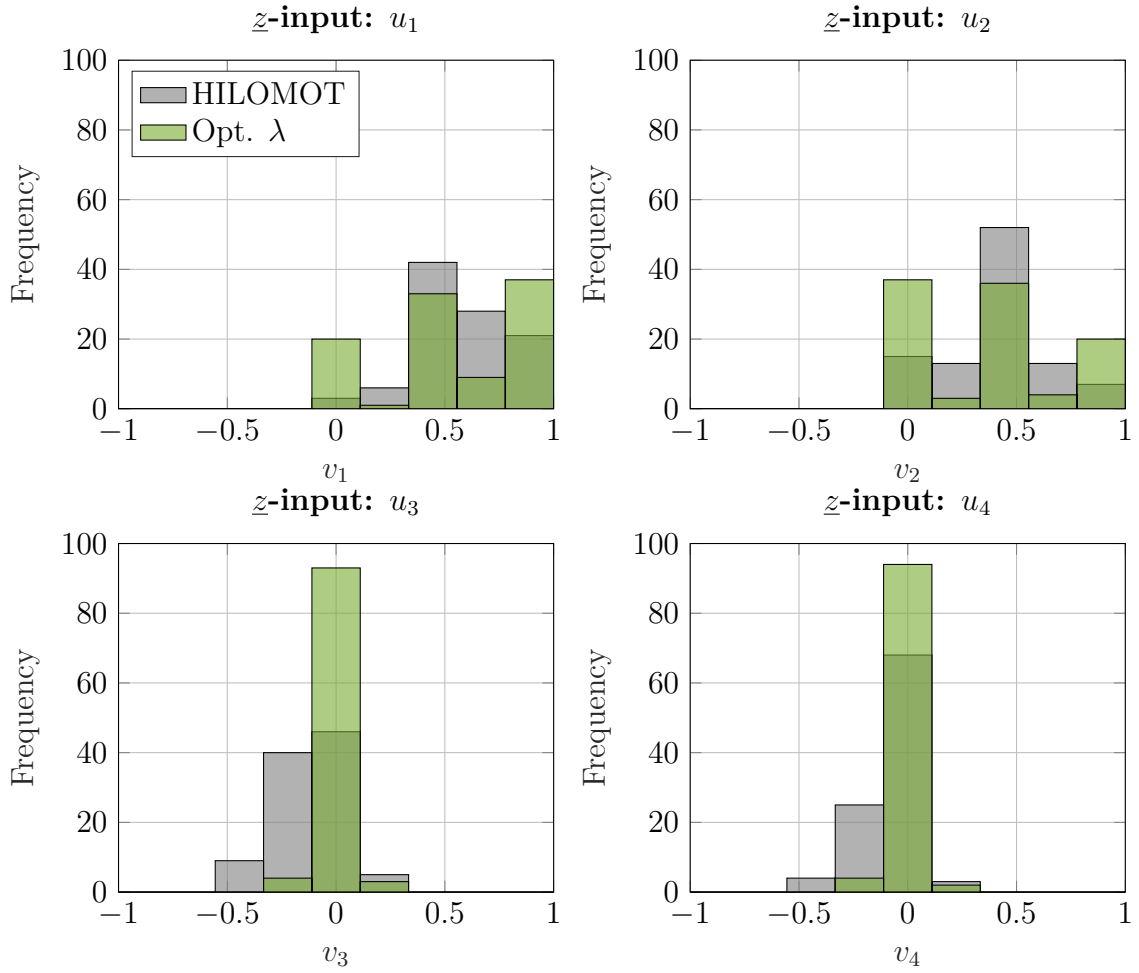


Figure 3.27: Histograms of the splitting parameter values obtained for the first split and all 100 noise realization of TP1 in case of HILOMOT and the RBIS approach with optimized λ

off the additional effort. It seems as if the introduced obliqueness penalty prevents an oblique input space partitioning in a lot of cases even if it is beneficial for the model's generalization performance.

Test Process Four

The dynamic TP4 is modeled with the external dynamics approach as explained in Section 2.1. Here, only simple delays q^{-1} are used as external filters and only one filter for the physical input as well as for the physical output is used, such that the inputs for the LMNs are $u(k-1)$ and $y(k-1)$. Using only simple delays for the external dynamics approach is known as NARX model. From the description of TP4 it is known that the nonlinearity only affects the delayed input $u(k-1)$ and not the

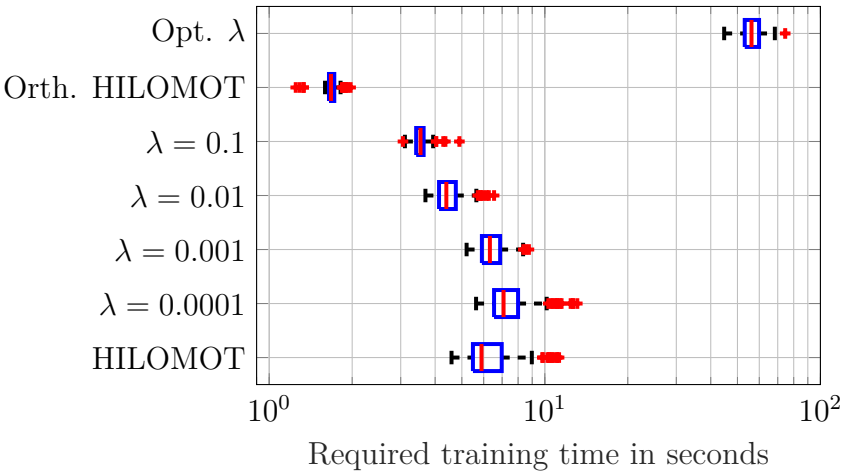
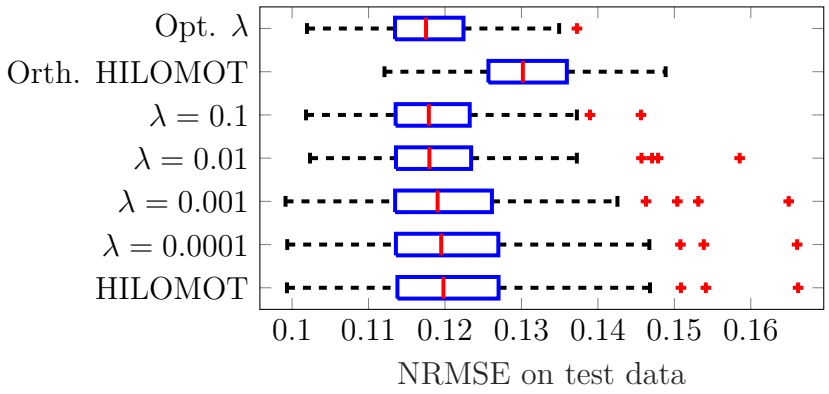


Figure 3.28: Boxplots of the required training times for all investigated scenarios of TP1

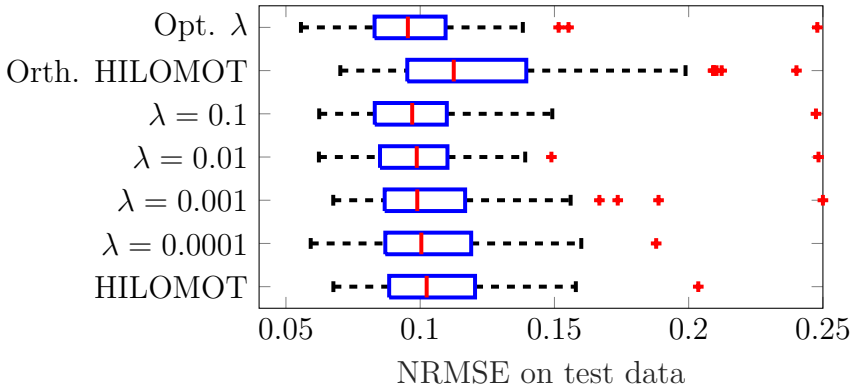
delayed output $y(k-1)$. Therefore, the RBIS approaches should be able to detect input $u(k-1)$ as important for the $\underline{z}$-input space while the splitting parameters corresponding to the delayed output $y(k-1)$ should be pushed to zero.

For the investigation of the RBIS approaches with TP4, 100 different training data sets are generated that only differ in the noise realization of the output values. The excitation signal used to obtain the training data sets is an amplitude modulated random binary signal (APRBS) consisting of $N = 250$ data samples. The maximum and minimum amplitude of the excitation signal is -3 and 3, respectively, in order to create sufficiently strong saturation effects. White Gaussian noise is added to the undisturbed process output to create the 100 noise realizations such that the SNR is 20 dB. As test data set serves another APRBS consisting of $N_t = 12.000$ as excitation signal with undisturbed process output values.

Again, the achieved generalization performances are for LMNs with just two local models and for LMNs with the optimal number of local models are shown in Fig. 3.29a and Fig. 3.29b, respectively. In case of LMNs with two local models the median NRMSE values are all pretty similar except for the orthogonal HILOMOT algorithm. The best values are achieved by the RBIS approaches with optimized λ as well as with the heuristic with $\lambda = 0.01$ and $\lambda = 0.1$. The worst median NRMSE value corresponds to the orthogonal HILOMOT. The variations from the median NRMSE values are very similar for all investigated scenarios with slightly less magnitude in case of the RBIS approaches with optimized λ, the heuristic with $\lambda = 0.01$, and $\lambda = 0.1$. For the LMNs with optimal model complexity the NRMSE values are again all pretty similar except for the orthogonal HILOMOT algorithm. The best



(a) Generalization performances with two local models

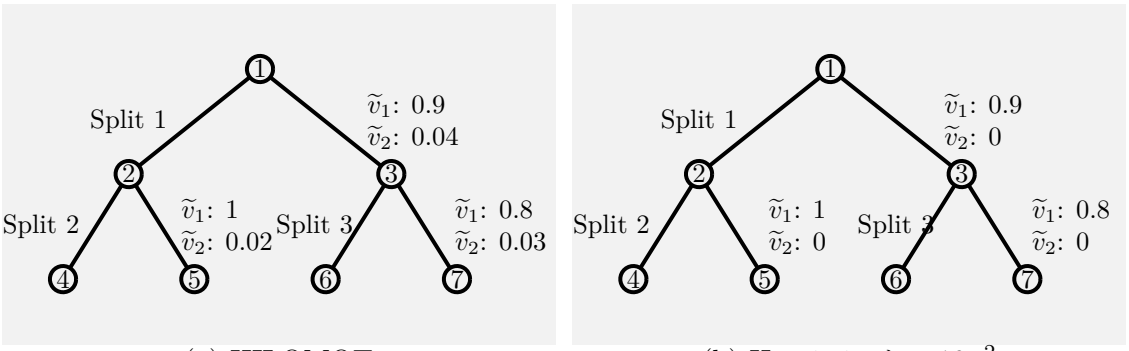


(b) Generalization performances with the optimal complexities

Figure 3.29: Histograms of the generalization performances of all investigated scenarios for two local models (a) and the optimal model complexity according to the AIC_c

NRMSE values as well as the smallest variations from it are achieved by the RBIS approaches with optimized λ and the heuristic with $\lambda = 0.1$. A bit surprising is the high variance of the LMNs obtained by the orthogonal HILOMOT algorithm. It results from a wider range of different optimal model complexities that are chosen by the AIC_c compared to all other training algorithms.

The inspection of the median absolute splitting parameter values reveals that the RBIS approaches with heuristic need at least an initial split regularization parameter of $\lambda = 0.01$ to push v_2 in the majority of all noise realizations to zero. The RBIS approach with optimized λ is also able to realize this desired outcome. Figure 3.30 shows exemplarily the binary trees of HILOMOT and the heuristic with $\lambda = 0.01$ together with the median absolute splitting parameter values over all noise realizations for the first three splits. Even without any regularization on the splitting parameters, the values of v_2 are very close to zero.



(a) HILOMOT (b) Heuristic, $\lambda = 10^{-2}$

Figure 3.30: Binary trees together with the median of the absolute values of the splitting parameters $\widetilde{v}_i$ for HILOMOT (a) and the heuristic with $\lambda = 10^{-4}$ (b) for splits one to three

Plots showing the model quality over the model complexity, the input space partitioning, and histograms of the splitting parameters over all noise realizations are omitted here, because they do not reveal any interesting information. Finally, the required training times are compared in Fig. 3.31. No surprises can be observed. The smallest training times are required by the orthogonal HILOMOT algorithm, while the most time-consuming variant is the RBIS approach with optimized λ. The heuristic RBIS approaches are all on a similar level independent of the used initial split regularization parameter. The required training times by HILOMOT lie in between the heuristic RBIS approaches and the orthogonal HILOMOT algorithm.

In summary, none of the RBIS approaches shows convincing results. Neither the

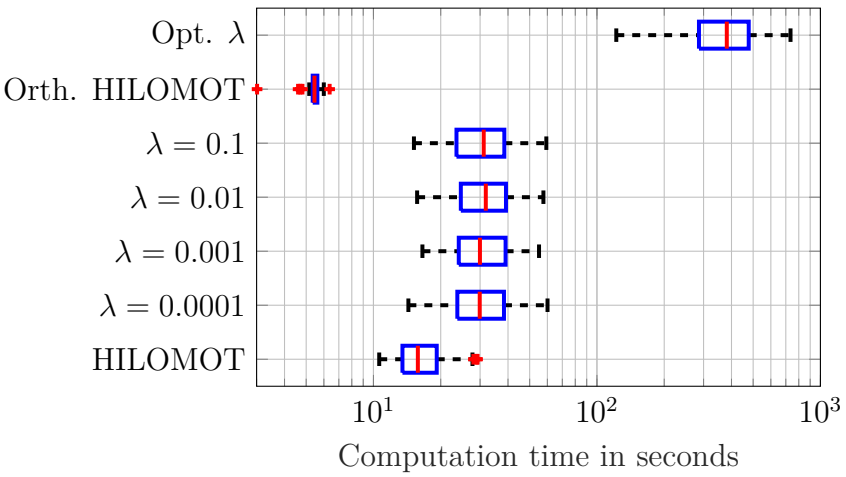


Figure 3.31: Boxplots of the required training times for all investigated scenarios of TP4

generalization performance of the LMNs nor the variance could be significantly improved independent of the test process under investigation. As shown for TP1, no good compromise could be found for the split regularization parameter that is able to keep splits parallel to unimportant $\underline{z}$-inputs while being able to reliably produce oblique splits in the subspace where they are needed. In case of TP4 no oblique splits are required at all, because only input $u(k-1)$ influences the output in a nonlinear way. Even in this scenario none of the RBIS approaches shows significant advantages compared to the standard HILOMOT algorithm. Based on all obtained results for TP1 and TP4, the RBIS approach can not be recommended to be used. However, the investigated scenarios are low-dimensional. For high-dimensional problems as they would arise e.g. in an nonlinear finite impulse response (FIR) modeling case this kind of regularization may pay off.

3.4 Embedded Approach

Embedded methods use model-specific or training-algorithm-specific properties to find good input subsets for the modeling task. In the following, special properties of LMNs with local affine models trained with HILOMOT are exploited. In this particular case, the directions of the splits represent the strength and main direction of the process nonlinearity. Since all local models are affine, changes in the slope of the process can only be described by the LMN by switching to another local affine model. If one input influences the process only linearly, all splits performed by HILOMOT will be approximately parallel to this dimension. The basic idea of the embedded approach is to exploit the aforementioned behavior of HILOMOT by analyzing the splitting parameters and evaluate the directions of all splits that are made. At the end the importance of each input compared to all other dimensions can be rated. Note that importance here only refers to an input's nonlinear influence. If the output of the process depends strongly on one input, but only in a linear way, this can be described by a large slope in the local model and it will be considered irrelevant by this embedded approach.

3.4.1 Partition Analysis

At first an LMN is trained and afterwards the partitioning is analyzed. As explained in detail in Section 3.3.1, the orientation of each split is determined by the relative,

absolute magnitude of the splitting parameters v_i, $i = 1, 2, \ldots, p_z$, see Figs. 3.19 and 3.20. The offset parameter v_0 does not influence the split orientation and therefore plays no role in the subsequent discussion. First only **one single split** is considered and the relevance factor ρ_i for the i-th $\underline{z}$-input equals:

$$\rho_i = \frac{|v_i|}{\sum_{j=1}^{p_z} |v_j|} \,. \tag{3.22}$$

For the special cases shown in Figs. 3.20a, 3.20b, and 3.20c the relevance factors turn out to be $\underline{\rho}_a = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$, $\underline{\rho}_b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T$, and $\underline{\rho}_c = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, respectively.

For the determination of the relevance factors for **the whole LMN**, so-called importance factors are calculated. After normalizing these importance factors, the relevance factors are obtained. For the determination of the importance factors ζ_i of each input in the $\underline{z}$-input space z_i, $i = 1, 2, \ldots, p_z$, the related splitting parameters of all splits are weighted and summed up:

$$\zeta_i = \sum_{j=1}^{M-1} |v_{ij}| \cdot PI_j \cdot w_{Nj} \,. \tag{3.23}$$

v_{ij} designates the splitting parameter of the j-th split belonging to z_i. Weighting factor PI_j is the performance improvement in the loss function achieved through split j. It is the difference between the loss function value before and after split j is carried out. Weighting factor w_{Nj} equals the number of points that are affected by split j. It is the number of points within the worst local model and can be measured by

$$w_{Nj} = \sum_{k=1}^{N} \Phi_j(\underline{z}(k)) \,, \tag{3.24}$$

due to the fuzzy nature of the splits. Note that an LMN with M local models possesses $M-1$ splits. Through the two weighting factors PI_j and w_{Nj} it is guaranteed that splitting parameter v_{ij} does only contribute to the importance factor if the corresponding split contributes to the improvement of the model performance and enough data points are affected. Finally the relevance factor for the whole LMN is obtained through the normalization of all importance factors:

$$\rho_i = \frac{\zeta_i}{\sum_{j=1}^{p_z} |\zeta_j|} \,. \tag{3.25}$$

A simple example that is shown in Fig. 3.32 should further illustrate the concept of the relevance factors. The partitioning consists of two axis-orthogonal splits. For simplicity the performance improvement factors PI_j, $j = 1, 2, \ldots, M-1$, are all assumed to be one in this example. Note that usually the performance improvement is smaller than one and approaches zero with an increasing number of local models (if the loss function values saturate). The first split possesses the splitting parameters $\underline{v}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and divides 16 data points in two halves such that the weighting factor becomes $w_{N1} = 16$. The second split possesses the splitting parameters $\underline{v}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ and divides eight data points in two halves leading to a weighting factor of $w_{N2} = 8$. Since the performance improvement factors are assumed to be one, the importance factors are

$$\zeta_1 = v_{11} \cdot w_{N1} + v_{12} \cdot w_{N2} = 16 \text{ and}$$
$$\zeta_2 = v_{21} \cdot w_{N1} + v_{22} \cdot w_{N2} = 8\,.$$

The final relevance factors for this example turn out to be:

$$\rho_1 = \frac{\zeta_1}{\zeta_1 + \zeta_2} = \frac{2}{3} \text{ and}$$
$$\rho_2 = \frac{\zeta_2}{\zeta_1 + \zeta_2} = \frac{1}{3}\,.$$

The result of the LMN partition analysis is a ranking of all inputs according to their relevance for the $\underline{z}$-input space. In addition to the ranking, the quantitative relevance is revealed. However, this information is not further used in this thesis. One idea exists that has not been investigated so far but should be mentioned here.

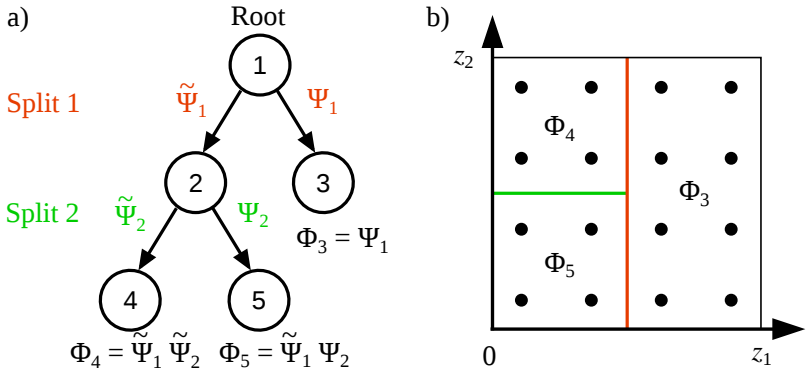


Figure 3.32: Illustration example for the relevance factor calculation: a) Shows the hierarchical binary tree with highlighted splits, b) shows the partitioning of the input space with 16 data points as dots

In tasks where distance measures are employed, e.g. to determine the largest gap in an already existing data set, the relevance factors might be used to scale the $\underline{z}$-input axes. Therefore, a weighting matrix

$$\underline{W} = \begin{bmatrix} \rho_1 & 0 & \cdots & 0 \\ 0 & \rho_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \rho_{p_z} \end{bmatrix} \tag{3.26}$$

is created with the relevance factors ρ_i, $i = 1, 2, \ldots, p_z$, as entries on its diagonal. This matrix can then be used to calculate the Mahalanobis distance between two points, $\underline{z}(i)$ and $\underline{z}(j)$:

$$d_M(\underline{z}(i), \underline{z}(j)) = \sqrt{(\underline{z}(i) - \underline{z}(j))^T \underline{W} (\underline{z}(i) - \underline{z}(j))}\,. \tag{3.27}$$

If the identity matrix $\underline{I}$ is used instead of the weighting matrix $\underline{W}$ in (3.27), the Mahalanobis distance reduces to the Euclidean distance. Using the weighting matrix $\underline{W}$ makes distances along more relevant $\underline{z}$-inputs appear larger. The larger a relevance factor of an $\underline{z}$-input is compared to the remaining $\underline{z}$-inputs, the more the focus lies on this particular $\underline{z}$-input. This might be beneficial for active learning strategies such as HILOMOT for design of experiments (HilomotDoE), which is described in more detail in Section 4.3.1. Through the focus on $\underline{z}$-inputs that are more relevant for the nonlinear behavior of a process, the design of experiments is potentially able to concentrate on the nonlinearity. As a result, the training data carries more information about the nonlinearity which can subsequently be exploited by experimental modeling approaches.

3.4.2 Investigation with Test Processes

TP2 and TP3 are used to demonstrate the abilities of the partition analysis explained in Section 3.4.1. For both test processes three different training data set sizes, namely $N = 50, 100$, and 200, and two different noise levels are part of the investigation. White Gaussian noise is added to the process outputs such that the SNR equals 20 dB and 25 dB. For each test process and each training data set size, 100 different noise realizations are generated. Here, neither validation nor test data is produced because the investigated LMN partition analysis does not affect the model accuracy. An LMN for each generated data set is trained and subsequently its partitioning

is analyzed. HILOMOT is used as training algorithm and the complexity of the LMNs is determined with the AIC_c. Only local affine models are used, such that the resulting relevance factors represent how much each $\underline{z}$-input contributes to the nonlinearity of the process. The evaluation of the results of the partition analysis is based exclusively on the knowledge about the two test processes.

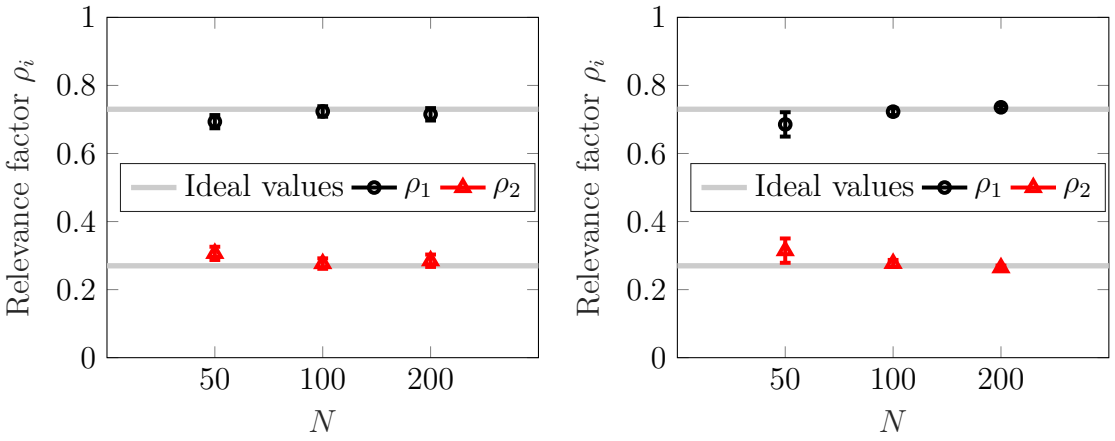
Test Process Three

First, the split analysis is used for TP3, which has only two inputs. A-priori it is known that both inputs contribute to the process output in a nonlinear way. However, input u_1 can be rated as more important for the nonlinear process behavior because the change in the slope in this direction is higher compared to the change in the slope in u_2-direction everywhere in the input space. In fact, the ideal values of the relevance factors can be determined exactly from prior knowledge. The derivative of (3.4) with respect to the inputs u_1 and u_2 is:

$$\nabla y(u_1, u_2) = \frac{0.1}{(0.08 + 0.73(1 - u_1) + 0.27(1 - u_2))^2} \begin{bmatrix} 0.73 & 0.27 \end{bmatrix}^T \tag{3.28}$$

As can be seen from (3.28) the gradient of TP3 points always in the same direction defined by the ratio of the vector entries 0.73 and 0.27. Since the relevance factors are scaled such that they all sum up to one, see (3.25), the values 0.73 and 0.27 already correspond to the ideal relevance factors. If the partitioning parameters of an LMN follow exactly this particular ratio, the local affine models should be able to adapt to the nonlinearity most efficiently. Deviations from this ratio should only occur due to noisy data and are likely to deteriorate the generalization performance of LMNs.

Figure 3.33 shows the median values of the obtained relevance factors together with the ideal ones for TP3. Additionally, the deviations around the median values are visualized as interquartile range (IQR), which is the distance between the 25-th and 75-th percentile. For both noise levels the obtained results are pretty similar. The median value of the relevance factors gets closer to the ideal values and the deviations shrink as the amount of training data increases. This is exactly the desired and expected result.



(a) SNR = 20 dB (b) SNR = 25 dB

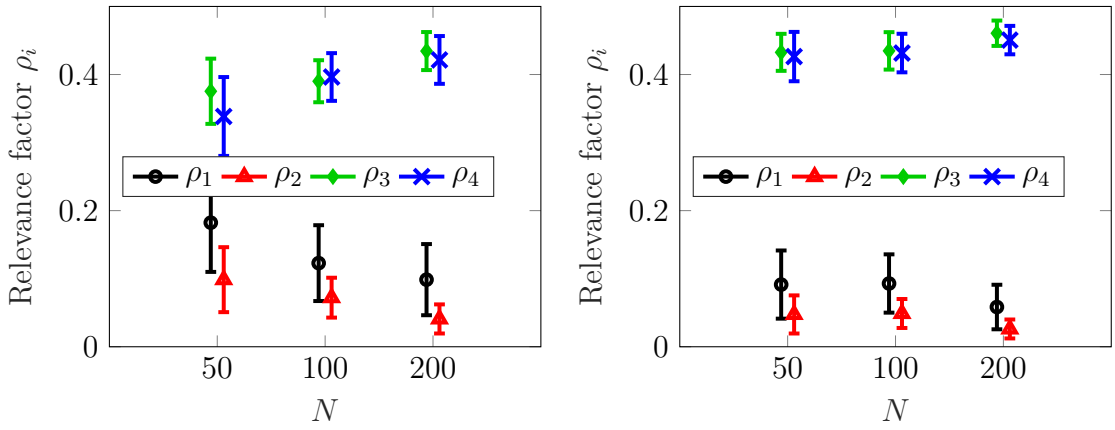
Figure 3.33: Relevance factors ρ_i for all $\underline{z}$-inputs of TP3 for two different noise levels and three different training data set sizes $N = 50$, 100, and 200. Marker indicate the median value; vertical lines represent the IQR.

Test Process Two

TP2 is scheduled by the binary input variables u_3 and u_4, that determine which of two planes is used to calculate the process output. Both planes depend only on the input variables u_1 and u_2. Therefore, u_1 and u_2 are only relevant for the $\underline{x}$-input space of an LMN while u_3 and u_4 are only relevant for the $\underline{z}$-input space. In case of TP2, no ideal values for the relevance factors are available. Only the qualitative relevance is known a-priori. Since the partition analysis does only rate the relevance for the $\underline{z}$-input space, relevance factors ρ_1 and ρ_2 should have lower values compared to ρ_3 and ρ_4.

Figure 3.34 shows the results for TP2. Again, the median relevance factor as well as a measure of the deviations from it are shown. As expected, the deviations from the median values can be decreased either by increasing the size of the training data set or by lowering the noise level. In general, the median relevance factors for the actually irrelevant $\underline{z}$-inputs u_1 and u_2 get quite high for $N < 200$. The relative rating of the relevance of each $\underline{z}$-input is in accordance with the knowledge about TP2. However, by reviewing the obtained results it is not clear that inputs u_1 and u_2 are irrelevant for the $\underline{z}$-input space. Furthermore, it is not clear why relevance factor ρ_1 has higher values compared to relevance factor ρ_2, since both corresponding inputs should be equally irrelevant.

In summary, it is shown that the LMN partition analysis is able to assign relevance factors that rank all $\underline{z}$-inputs according to their nonlinear influence correctly as



(a) SNR = 20 dB (b) SNR = 25 dB

Figure 3.34: Relevance factors ρ_i for all $\underline{z}$-inputs of TP2 for two different noise levels and three different training data set sizes $N = 50$, 100, and 200. Marker indicate the median value; vertical lines represent the IQR.

long as there is an influence at all. The results obtained for TP3 are excellent. Even for relatively low training data set sizes the a-priori known ideal relevance factors are obtained through the LMN partition analysis with good accuracy. For TP2 it is recognized that inputs u_3 and u_4 are most relevant for the description of the nonlinearity. On the downside, it is not clear from which relevance factor value on an $\underline{z}$-input should be considered as irrelevant. As shown in Fig. 3.34a, a median relevance factor of almost 0.2 is obtained for input u_1. This is a rather high value from which one might conclude wrongly a significant nonlinear influence of u_1. The problem of having no specific threshold value from which on an input can be considered relevant for the $\underline{z}$-input space is currently unsolved.

3.5 Visualization: Partial Dependence Plots

Another idea to find relevant inputs for data-based models is the visualization of dependencies between the inputs and the output. Apparently, visualizations of data or models resulting from the data are restricted to low-dimensional views. For input space dimensions of one and two, the dependencies can easily be visualized with the help of a model. As the number of input variables increases, it gets harder to obtain this information. One popular way to deal with this problem is by examining *slices* through the model, i.e., fixing all inputs except for one or two. However, such slices can be very misleading since the effect of the fixed variables is completely concealed. Even worse, not feasible input combinations can artificially be generated.

A much more powerful approach that can be applied to models of any dimension is the use of so called *partial dependence plots* [57]. The idea is to view a collection of plots, where each of these plots visualizes the partial dependence of the model on a selected small subset of input variables. According to [57], such a collection can seldom provide a comprehensive depiction of the whole model, but it can lead to helpful clues. Partial dependence plots are scientifically not new, but in the experience of the author seldom used and rather unknown, at least for people with an engineering background. Therefore, partial dependence plots are described in the next few paragraphs shortly following mainly the explanations from [57].

For simplicity, only the typical case is explained in the following, where the partial dependence of the model on just one single variable is considered. This input is called the partial dependence input (PDI). In order to generate a partial dependence plot, a function that estimates the partial dependence of the model on the PDI is needed and will be called the partial dependence function $\bar{f}_P$. This function represents the effect of the PDI on the model taking into account the average effects of all training data samples.

In order to determine the partial dependence function, some definitions are made in the following. If there are p different PDI variables u_j, there will be p partial dependence functions $\bar{f}_{P,j}$ with $j = 1, \cdots, p$. The vector $\underline{u}^{(-j)}(i)$ is the i-th training data sample without the PDI variable:

$$\underline{u}^{(-j)}(i) = \begin{bmatrix} u_1(i) & u_2(i) & \cdots & u_{j-1}(i) & u_{j+1}(i) & \cdots & u_p(i) \end{bmatrix}^T . \tag{3.29}$$

Therefore $\hat{y}(u_j, \underline{u}^{(-j)}(i))$ is the model output at the i-th training data sample, but the value of the PDI can be chosen arbitrarily. With the help of these definitions, the j-th partial dependence function can be estimated by

$$\bar{f}_{P,j}(u_j) = \frac{1}{N} \sum_{i=1}^{N} \hat{y}(u_j, \underline{u}^{(-j)}(i)) . \tag{3.30}$$

At any specified value for the PDI, the model output $\hat{y}(\cdot)$ varies due to changes in the training data samples $\underline{u}^{(-j)}(i)$, $i = 1, 2, \ldots, N$. The mean of this variation is estimated by the partial dependence function. The estimation of the corresponding variance is straight forward:

$$\sigma^2_{P,j}(u_j) = \frac{1}{N-1} \sum_{i=1}^{N} (\hat{y}(u_j, \underline{u}^{(-j)}(i)) - \bar{f}_{P,j}(u_j))^2 . \tag{3.31}$$

In general there might be more than just one PDI variable, see [57] for further details. However, for visualization purposes more than two PDI variables are not very valuable. Typically, the PDI is varied equidistantly from its minimum to its maximum value occurring in the available data set. This results in a mean curve and its corresponding variances when applying (3.30) and (3.31) to each of these PDI values.

A model with two inputs is utilized to illustrate features of the partial dependence plots. As demonstration example the following artificial process is approximated by an LMN:

$$y = u_1 + u_1 \cdot u_2^2 \,. \tag{3.32}$$

We define the first PDI variable as u_1 and the second one as u_2. So for each individual PDI variable, we obtain a partial dependence plot. The process as well as the partial dependence plots are shown in Fig. 3.35. Both partial dependence plots contain important information about the dependencies of the process on the input variables. In Fig. 3.35b the linear influence of u_1 can be seen and in Fig. 3.35c the quadratic

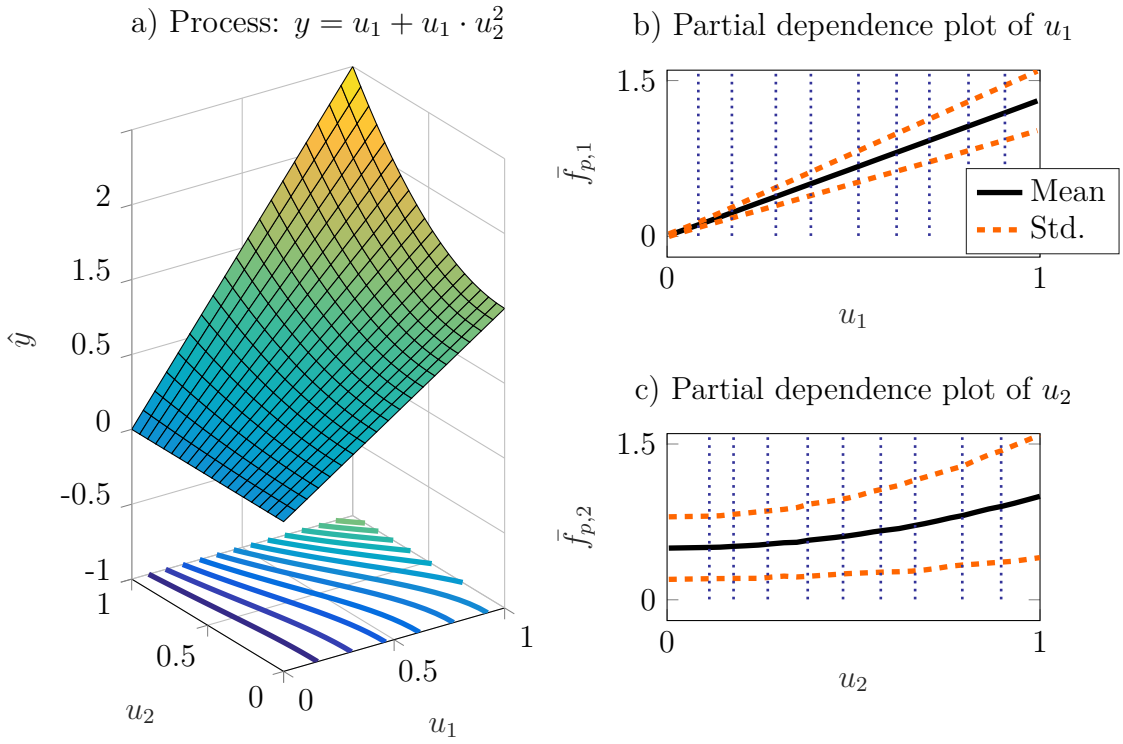


Figure 3.35: Model with two inputs (a) and its partial dependence plots of u_1 (b) and u_2 (c)

characteristics of u_2 are observable. Larger standard deviations indicate stronger influences of variables not chosen as PDI. Therefore the growing standard deviation in Fig. 3.35b with increasing values of u_1 implies a growing influence of u_2. The vertical, dotted lines in Fig. 3.35b and Fig. 3.35c give hints about the data distribution along the PDI variable. In between two vertical lines lie 10 % of the PDI values present in the training data set. The most left and the most right vertical line are omitted and would occur at the beginning and the ending of the mean curves. In the example shown in Fig. 3.35 the data is distributed uniformly along both axes u_1 and u_2.

One important thing to keep in mind is that the whole visualization approach highly depends on the model of the process. Because rather the model than the process itself is visualized, the obtained results are only trustworthy if the model accuracy is high.

3.5.1 Investigation with Test Processes

TP1 and TP2 serve as examples to further review the capabilities of the partial dependence plots for the detection of relevant inputs. In case of TP1 it is interesting to see if the irrelevance of input u_4 can be observed from the resulting partial dependence plots. For TP2 it is worth knowing what happens in case of the discrete, binary inputs u_3 and u_4.

In order to create the partial dependence plots, LMNs are trained with HILOMOT. Local affine models are used. The model complexity is automatically determined with the $\mathrm{AIC_c}$ criterion. Because both test processes have four inputs, the used training data sets share the same properties except for the added noise to the output. They consist of $N = 100$ training data samples and are obtained through a maximin LH optimization. In case of TP1 no additional noise is added to the output because input u_4 exclusively consists of noise. For TP2 white Gaussian noise is added such that the SNR equals 20 dB. For both test processes 100 different noise realizations are generated. The partial dependence plots shown in the following belong to just one of the 100 noise realizations. The shown plots are the typical outcome.

Test Process One

Figure 3.36 shows the partial dependence plots for all inputs of TP1. The nonlinear dependency of the output from inputs u_1 and u_2 are observable as well as the linear

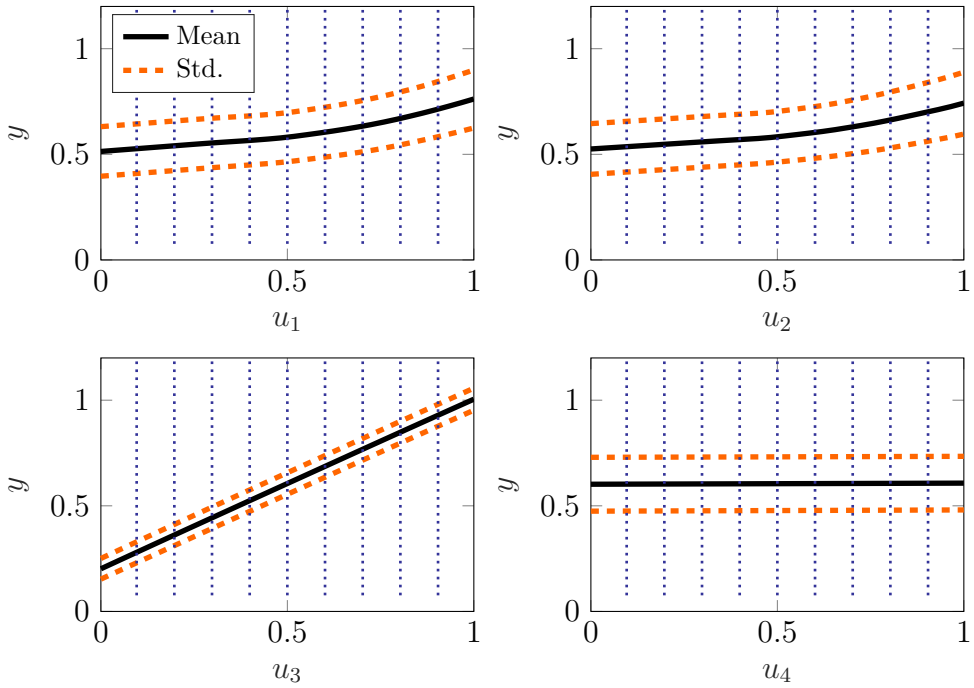


Figure 3.36: Partial dependence plots for all inputs of TP1

relationship between u_3 and the output. Furthermore it can be recognized that inputs u_1 and u_2 influence the output in a quite similar way. The partial dependence plot of input u_4 indicates that it influences the output not at all. The slope of the mean curve is zero independent of the value of u_4. Or in other words, the value of u_4 does not have a significant impact on the output value. These observations are in perfect accordance to the knowledge about TP1, see (3.2). Looking at the standard deviations it is noticeable that these are almost constant for each partial dependence plot independent of the value of the PDI. This indicates that the influence of an PDI does not depend on its value.

Test Process Two

Figure 3.37 shows the partial dependence plots for all inputs of TP2. The mean curve for input u_1 shows a nonlinear behavior, i.e. a decrease in the slope with increasing values of u_1. This is likely due to the fact that the model output gets pretty small for high u_1 values if both discrete inputs u_3 and u_4 equal one, see Fig. 3.4 and (3.3). However, this interpretation is not based on the partial dependence plot itself but on the knowledge available about TP2. Without this knowledge input u_1 appears to have a nonlinear influence. The origin of this nonlinear behavior, which is the interaction of u_1 with inputs u_3 and u_4, is unclear. In contrast to that, input u_2 seems

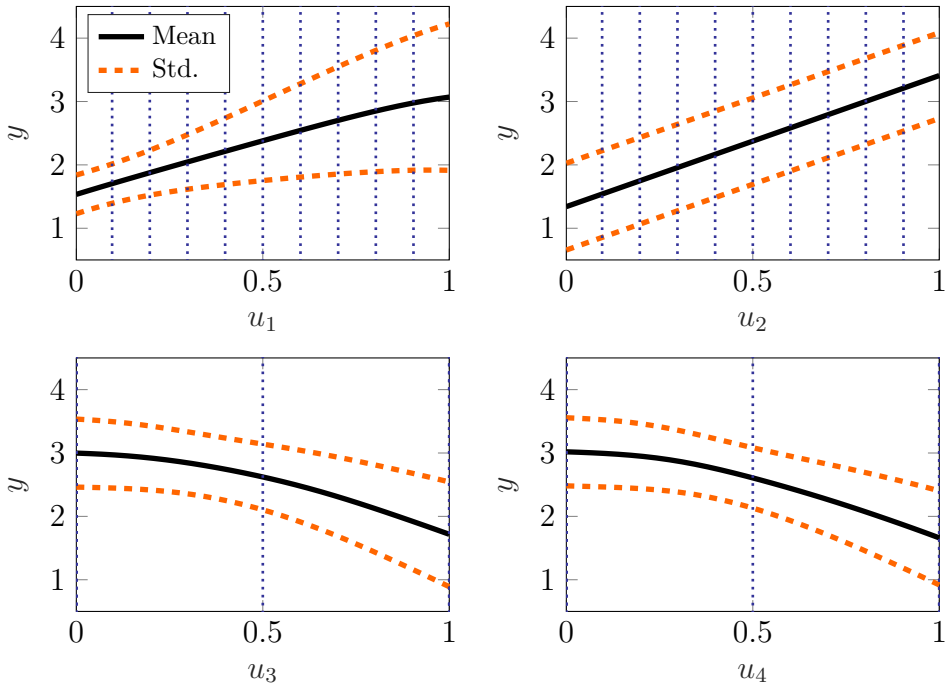


Figure 3.37: Partial dependence plots for all inputs of TP2

to influence the input in a linear way. In fact, for the influence of u_2 on the output it is irrelevant how the discrete inputs u_3 and u_4 are chosen because the coefficients for input u_2 are the same for both of the scheduled planes, see (3.3). From the partial dependence plots of u_3 and u_4 a nonlinear relationship is observable. The tendency to obtain lower output values for high values of u_3 and u_4 is clearly visible. The lack of vertical lines indicates that the distribution of values along the axes u_3 and u_4 is by far not uniform. In fact, it can be concluded that most points lie close to the minimum and maximum values of the two axes, which of course is a result of the binary nature of these two inputs.

The standard deviations shown in the partial dependence plots for inputs u_2, u_3, and u_4 are almost constant for all PDI values. For the partial dependence of u_1 the standard deviation increases with higher values of u_1. This indicates a growing importance of other variables. In fact, the deviations that can be expected due to switching between the two planes are higher for high values of u_1 (≈ 1) compared to low values of u_1 (≈ 0), see Fig. 3.4.

In summary, the partial dependence plots are a helpful tool to quickly review the average influence of single inputs on the model output. The distinction between linearly and nonlinearly influencing inputs seems to be possible from the observation of the mean curves. The standard deviations help to rate the influence of all non-

PDIs together compared to the PDI, since variations at each specific value of the PDI are only due the variations in the non-PDIs. Mean curves with a slope being zero for all possible values of the PDI indicate that there is no influence of that PDI variable at all. Additionally, helpful information about the data distribution along all single axes can be obtained.

Despite all the advantages, partial dependence plots have to be interpreted with care. For example, the fact that the slope of the mean curve of a partial dependence plot is always positive does not necessarily mean that this is true everywhere in the input space. This can be seen in the partial dependence plot for input u_1 of TP2 shown in Fig. 3.37. From the equation of this process it is clear, that there is a negative slope in u_1-direction if both operating point variables u_3 and u_4 equal one. This behavior is not observable from the partial dependence plot because on average the process output is still increasing with higher u_1 values. Since the training data covers the input space uniformly, only one fourth of it leads to the usage of the plane with negative slope in the u_1 direction, in particular when $u_3 = u_4 = 1$. Care is also necessary for the interpretation of partial dependence plots of discrete inputs. Instead of plotting whole mean curves it might be advisable to just plot values at discrete values that do exist in the training data set.

In addition, partial dependence plots are very sensitive w.r.t. the input data distribution. In the given examples, input data was evenly distributed due to the maximin LH design. Information extraction is much more difficult or even impossible if data is „strangly“ distributed. Especially correlations between inputs may lead to unreliable partial dependence plots as demonstrated for one application in Chapter 5.

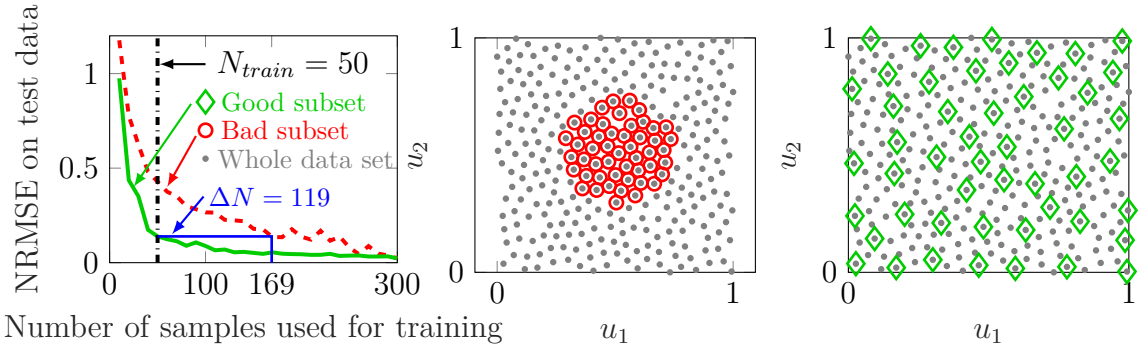
4 Design of Experiments Studies

As already outlined in Section 2.5, design of experiments (DoE) plays a key role in experimental modeling. Independent of the model type, the characteristics of a process can only be represented by a black-box model, if these characteristics are contained in the data used for the training. This thesis contributes to the further development of both passive and active learning strategies. Section 4.1 deals with the order in which experiments should be conducted, if the goal is to yield the best possible model performance with fractions of the whole experimental design. In black-box modeling tasks active learners need at least some data for the generation of an initial model, on which the strategy can rely on. Through a clever order of experimentation the amount of data needed for the initial model might be decreased. In Section 4.2 advice about the incorporation of corner measurements is given. The corners (vertices) of a hypercube are defined by the most extreme values of the input variables. This section focuses on the case where the amount of possible measurements is not much bigger than the number of corners. Finally, an extension to the active learning procedure HILOMOT for design of experiments (HilomotDoE) is presented in Section 4.3. This extension incorporates a new strategy to pick more than one query point which can easily be expanded for multiple-input multiple-output (MIMO) systems. Additionally, an alternative for the generation of so-called candidate points is presented.

4.1 Order Of Experimentation

An often neglected aspect and main topic of this section regards the order in which the measurements are conducted to yield the best possible model performance with fractions of the whole experimental design. Only little literature could be found addressing the order of experimentation aiming at this specific goal. For example, [34] and [62] consider the order of experimentation only in the context of neutralizing influences of undesirable factors on the experimentation or efforts needed to change

factor levels. Additionally, these two publications deal only with factorial designs, whereas methods proposed in this section can be applied to arbitrary experimental designs. The method presented in [130] aims at a reduction of the training set size in order to decrease computational demands and to improve the convergence speed of the model training. In differentiation to that, the proposed methods here aim to improve the convergence with respect to the *data amount* and does not consider computational demands at all. Good models in early stages of the measurement process yield several advantages, e.g. time can be saved because demanded model qualities can be reached with less data which is exemplarily demonstrated in Fig. 4.1. For this example the experimental design is a maximin Latin Hypercube (LH) from which subsets are chosen in a *bad* and a *good* way. In order to reach the same model quality $\Delta N = 119$ additional samples are needed following the *bad* ordering strategy, see Fig. 4.1a. The chosen subsets for $N_{train} = 50$ for both order determination strategies are visualized. Only points near the design space center are chosen for the subset designated as bad (red circles in Fig. 4.1b). The other subset, designated as good, originates from the intelligent k-means sequence (IKMS) explained in more detail in Section 4.1.3 (green diamonds in Fig. 4.1c). Of course Fig. 4.1 contrasts worst- and best-case scenarios which are extremely unlikely to happen in practice through the standard approach of random selection. The point of the approach proposed here is to guarantee the best worst-case scenario. As a result the model can be used earlier, while the measurement process is still in progress. For example, model-based optimization runs become more reliable with small data subsets. Active learning strategies enlarge the experimental design in an iterative and adaptive



(a) Test error progress

(b) *Bad* subset; $N_{train} = 50$

(c) *Good* subset; $N_{train} = 50$

Figure 4.1: Demonstration example for the importance of the order of experimentation. Test error progress (a) together with the chosen subsets in a *bad* (b) and a *good* way (c).

way, based on models trained with the currently available training data, such that the information gained by additional measurements is maximized. Typically, an initial experimental design is needed before the active learning phase can start [54]. Through a good order of experimentation the necessary amount of data serving as initial experimental design might be decreased. Note that throughout this thesis the term "measurement" is used synonymously for obtained data points, regardless of their origin. For example, the data might originate from real-world measurements as well as from computer simulations.

The proposed methods are based on the following considerations. Models relying only on small amounts of data are limited in their possible model accuracy. Through a clever placement of measurements, the possible model accuracy can be influenced. The aim of all order determination strategies is to find subsets of points from the overall experimental design, that lead to the possibly best model quality. Therefore a wide and nearly uniform coverage of the whole input space with small subsets is the goal if no prior knowledge about the process is available. The biggest gap sequence (BGS) tries to tackle the described considerations straight forward, filling the biggest gap of the input space with each additional point. One possible weakness of BGS might be the exclusive concentration on points near the boundary at the beginning of the algorithm, especially in high-dimensional input spaces. This potential issue motivates the median distance sequence (MDS) that tries to avoid the concentration of boundary-near points at the beginning. Adding the next point corresponding to the median distance instead of the maximum distance should lead to a better balance between points at the boundary and inside the input space for very small subsets. The IKMS method exploits in a first step possibly existing structures in the experimental design by the intelligent k-means algorithm [89]. All resulting clusters represent a specific region in the input space. Starting with all points next to these cluster centers should obtain information from all input space regions and in addition avoids the concentration of points at the boundary for very small subsets. It is a mechanism to balance the exploration and exploitation of the whole experimental design. At the beginning all areas of the input space covered by the experimental design are explored and then successively exploited in more detail.

The different methods to determine the order of experimentation for regression problems are explained in more detail in Sections 4.1.1 to 4.1.3. All methods are originally published in [8] by the author of this thesis and are based on the aforementioned considerations. Section 4.1.5 compares all methods for the determination of the order of experimentation on functions generated with the static function generator

that is explained in Section 2.7. Section 4.1.6 summarizes the order of experimentation findings and gives some remarks about the applicability for real-world test benches.

For the following explanations it is always necessary to distinguish between three sets of data points. Set $\mathbb{N}$ contains all points of a data set. $\mathbb{S}$, containing all already sorted points, and $\mathbb{F}$, containing all not yet sorted points, are non-overlapping subsets of $\mathbb{N}$. Here and throughout the rest of this section, "sorted" refers to the successional order of experimentation, i.e. the order in which the measurements should be conducted. The relationships between the previously defined sets can mathematically be expressed as follows:

$$\mathbb{N} = \mathbb{S} \cup \mathbb{F} \text{ and } \mathbb{S} \cap \mathbb{F} = \emptyset \ . \tag{4.1}$$

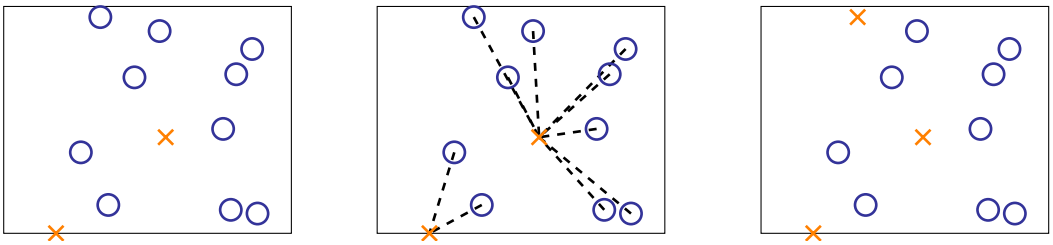
At the beginning of each method, $\mathbb{F}$ contains all data points ($\mathbb{F} = \mathbb{N}$). Then points are sequentially moved from $\mathbb{F}$ to $\mathbb{S}$ until $\mathbb{F}$ is empty ($\mathbb{F} = \emptyset$).

4.1.1 Biggest Gap Sequence

For the BGS, the first point to be added to $\mathbb{S}$ is the one closest to the center of gravity of all data points. In the following, one iteration of the BGS is explained and illustrated in Fig. 4.2. Here one iteration refers to all steps necessary to determine one data point, that should be added to the sorted list next. In Fig. 4.2a there are two already sorted points in $\mathbb{S}$ (orange crosses), whereas all other points still belong to set $\mathbb{F}$ (blue circles). Each point in $\mathbb{F}$ is now assigned to its nearest neighbor (NN) from subset $\mathbb{S}$. The dashed lines in Fig. 4.2b connect each point in $\mathbb{F}$ with its NN in $\mathbb{S}$. The corresponding distances (lengths of the dashed lines) from all points in $\mathbb{F}$ to their NN in $\mathbb{S}$ are calculated. The point with the maximum distance to its NN is selected and is moved from $\mathbb{F}$ to $\mathbb{S}$, see Fig. 4.2c. After adding a point to $\mathbb{S}$, the next iteration starts and the whole procedure continues until $\mathbb{F}$ is empty. Note that $\mathbb{S}$ is incremented in size only after the last step of each iteration.

4.1.2 Median Distance Sequence

The MDS starts similar to the BGS method, i.e. the first point added to $\mathbb{S}$ is the one closest to the center of all data points. After that, each point in $\mathbb{F}$ is assigned to



(a) Two already sorted points (x)

(b) Assigning each point in $\mathbb{F}$ (o) to its NN in $\mathbb{S}$ (x)

(c) Maximum NN is added to $\mathbb{S}$ (x)

Figure 4.2: Illustration of the procedure for the MDS and BGS methods

its NN from subset $\mathbb{S}$, like in the BGS strategy, see Fig. 4.2b. Again, the distances from all points in $\mathbb{F}$ to their NN in $\mathbb{S}$ are calculated. Now the point that corresponds to the median value of all calculated distances is moved from $\mathbb{F}$ to $\mathbb{S}$, instead of the one with the maximum distance. An update of the NN relationships between $\mathbb{F}$ and $\mathbb{S}$ has to be performed. After that the procedure continues until $\mathbb{F}$ is empty.

4.1.3 Intelligent k-means Sequence

For the ordinary k-means algorithm the number of clusters k as well as initial centroids (centers of the clusters) have to be specified by the user before the algorithm can proceed. In the *intelligent* k-means algorithm the number of clusters k and the corresponding initial centroids are automatically determined, before the ordinary k-means algorithm is used [89]. Therefore, the intelligent k-means clustering contains the ordinary k-means clustering algorithm but adds a clever initialization for it as a pre-processing step [26].

Intelligent k-means Initialization

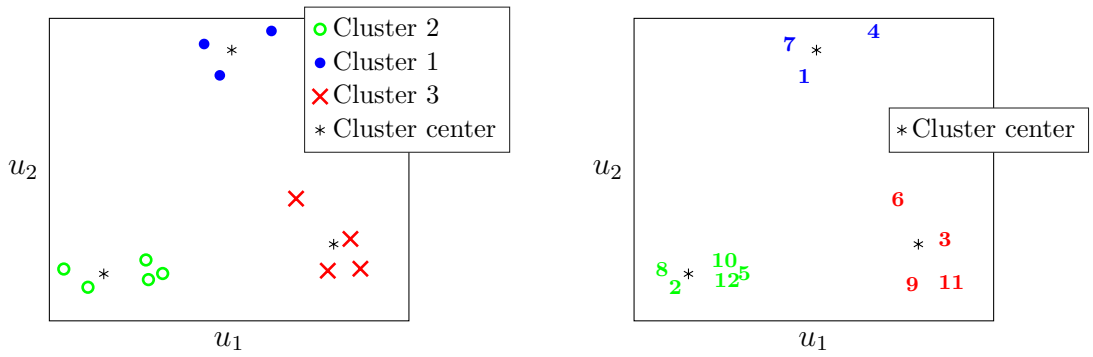
1. Determine the center of gravity of all data points $\underline{c}_1$.
2. Determine the data point $\underline{c}_2$ farthest away from $\underline{c}_1$. Treat both points as cluster centers.
3. All data points are assigned to their nearest cluster center.
4. In contrast to the ordinary k-means algorithm only $\underline{c}_2$ is updated to the center of gravity of all points assigned to $\underline{c}_2$. Cluster center $\underline{c}_1$ remains unchanged.

5. Repeat step 3 and 4 until convergence, i.e. no further changes in $\underline{c}_2$.

6. The cluster center $\underline{c}_2$ is saved and all data points belonging to it are removed from the data set for further calculations. If the number of points that have been assigned to cluster center $\underline{c}_2$ is greater than a user specified number (typically 2), it is used later for the initialization of the ordinary k-means clustering.

7. If there are data points left, go to step 1; otherwise stop.

At the end of this procedure there are k saved cluster centers for the initialization of the ordinary k-means algorithm leading to k clusters. Then all data points belonging to a cluster C_i, $i = 1 \ldots k$, are sorted according to the BGS strategy described in Section 4.1.1. As a result there are k sorted data point lists $\mathbb{L}_i$, $i = 1, 2, \ldots, k$, or cluster lists respectively. To get the final order of experimentation, the first element of each list is moved from $\mathbb{F}$ to $\mathbb{S}$ as described in more detail in Algorithm 1. In that way the first k points to be measured are the ones closest to the cluster centers, because these points are the first elements in the lists $\mathbb{L}_i$. After that, the second element of each cluster list is added until all data points are ordered according to the IKMS. Figure 4.3 demonstrates the IKMS procedure for twelve points. First, the intelligent k-means algorithm determines three clusters and assigns each point to one cluster, see Fig. 4.3a. The numbers shown in Fig. 4.3b represent the final ordering determined by the IKMS method.



(a) The intelligent k-means determines three clusters

(b) The numbers represent the sequence determined by the IKMS procedure

Figure 4.3: Explanation of the IKMS with the help of a demonstration example

Input: Set of sorted data point lists $\mathbb{L}_i$, $i = 1, 2, \ldots, k$
Output: One ordered list $\mathbb{S}$ which specifies the order of experimentation
Step 1: Define the set of not yet sorted points $\mathbb{F}$ as the union of all points contained in the lists $\mathbb{L}_i$, $i = 1, 2, \ldots, k$:
$\mathbb{F} := \mathbb{L}_1 \cup \mathbb{L}_2 \cup \ldots \cup \mathbb{L}_k$;
Step 2: Move the first element of each list $\mathbb{L}_i$, $i = 1, 2, \ldots, k$ to $\mathbb{S}$ until $\mathbb{F}$ is empty. The processing order of the lists is arbitrarily determined but fixed.
while $\mathbb{F} \neq \emptyset$ **do**
    `/* Go through all non-empty lists */`
    for *each non-empty list* **do**
        move first element of the current list to $\mathbb{S}$;
        if *current list is empty* **then**
            remove current list from the set of non-empty lists
        end
    end
    `/* Update the set of all not yet sorted points */`
    $\mathbb{F} := \mathbb{L}_1 \cup \mathbb{L}_2 \cup \ldots \cup \mathbb{L}_k$
end

Algorithm 1: Specification of the order of experimentation once the clustering of the data is finished according to the intelligent k-means algorithm

4.1.4 Other Determination Strategies

The aforementioned order determination strategies are compared to one active learning strategy and a simple randomization of the order of the measurements. As active learning strategy HilomotDoE [54] is utilized. For the selection of the point that should be moved next from $\mathbb{F}$ to $\mathbb{S}$ a local model network (LMN) is trained according to the HIerarchical LOcal MOdel Tree (HILOMOT) algorithm based on all points already present in $\mathbb{S}$. As a result the whole input space is partitioned into subregions and for each subregion a local error measure is calculated. The point in $\mathbb{F}$ is chosen that fills the biggest gap within the subregion with the worst local error measure. If there are no points from $\mathbb{F}$ inside the worst performing subregion, the second worst subregion is considered and so on. After one point is moved from $\mathbb{F}$ to $\mathbb{S}$, the whole procedure starts again (including a new training) and continues until all measurements are ordered. HilomotDoE requires some initial points in $\mathbb{S}$ such that the parameters of the local model network can be estimated. Typically $N_{ini} = 2(p+1)$ initial points are determined according to the BGS strategy. Further details about HilomotDoE can be found in Section 4.3 and in [55].

Optimality criteria are not considered here to determine the order of experimen-

tation. For these criteria typically a real-valued summary statistics of the Fisher information matrix is optimized to achieve some desired properties, like e.g. minimum variance of the estimated parameters (D-optimal design) [43]. The Fisher information matrix of the used model type (here it would be a local model network) depends on parameters which are responsible for the input space partitioning. If this partitioning of the local model network is determined and fixed, the Fisher information matrix could easily be derived. However, this information matrix would only be valid in case of the *global* (concurrent) estimation of all local model parameters. The used identification algorithm utilizes a *local* estimation scheme for these parameters in order to introduce a regularization effect, improve interpretability, and avoid overfitting, see [96] for further details. Therefore it is impossible to straightforwardly implement an optimality-criterion-based strategy.

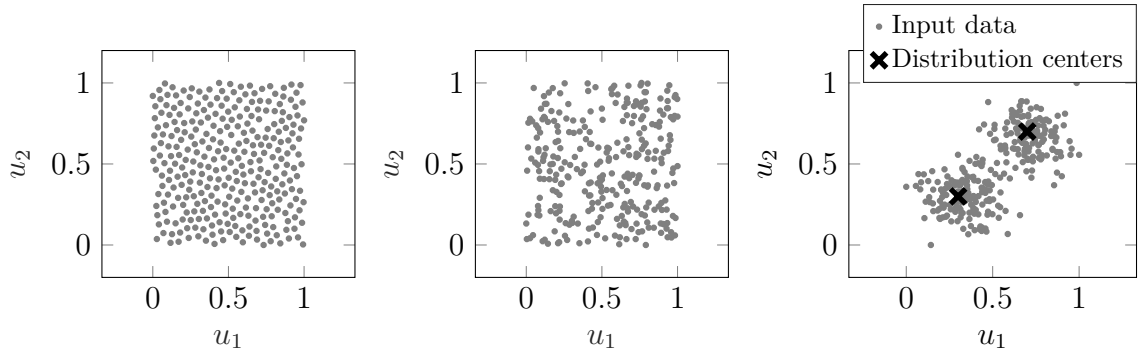
4.1.5 Comparison on Synthetic Functions

The experimental setup is described that is chosen to compare the different methods for the determination of the order of experimentation for synthetic test functions. These test functions are generated randomly with the help of the function generator described in Section 2.7. The dimensionality of the design space is varied from $p = 2$ to $p = 8$. Three training data sets with different input distributions are used for each random function and input dimension. Additionally, one huge test data set is generated for each random function and input dimension in order to assess the model quality. For each training data set the order of experimentation is determined according to the methods defined in Sections. 4.1.1, 4.1.2, 4.1.3, and 4.1.4. Then, for training data increments of 10 % (in the determined order) a model is trained and its performance is assessed using the test data. Strategies for the order determination are compared based on how quick the model performance increases (corresponding to an decrease of the test error). Therefore, results are averaged over all random functions created with the function generator. Here the advantages of a function generator are exploited, i.e. designing synthetic functions with desired properties such as the input dimensionality, the amount of data and the data distribution, to name only a few.

In order to conduct the comparison, three different input distributions are used. Maximin LH designs, optimized according to the algorithm proposed in [35], data samples drawn from a uniform distribution, and data samples drawn from two normal distributions with different mean values and equal standard deviations are employed.

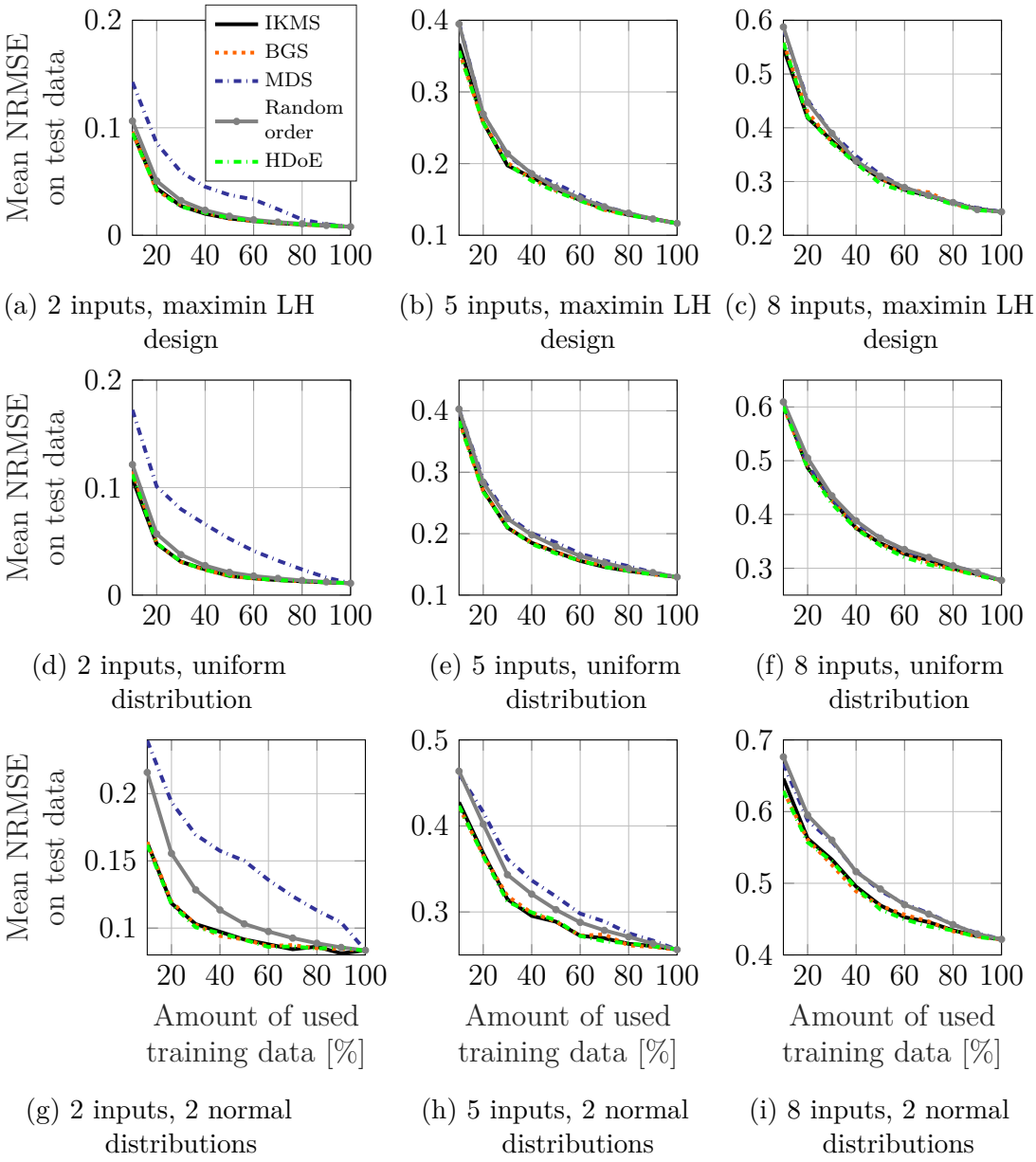
The two normal distributions have equal standard deviations, but the centers differ, $\mathcal{N}(0.3 \cdot \underline{1}, 0.12 \cdot \underline{I})$ and $\mathcal{N}(0.7 \cdot \underline{1}, 0.12 \cdot \underline{I})$, with $\underline{1}$ being a vector consisting of ones and $\underline{I}$ being the identity matrix. Figure 4.4 shows the different data distributions exemplarily in a two dimensional input space. It can be recognized that the input coverage decreases from the maximin LH design over data drawn from a uniform distribution down to the drawing from two normal distributions. Additionally the chance to repeat almost identical inputs twice or more often grows from Fig. 4.4a to 4.4c. The number of training samples for all data distributions is kept constant at $N = 300$, while the number of inputs varies from $p = 2, \ldots, 8$. Due to the constant number of training samples and the increasing input dimension, different local data densities arise.
In case of the uniform distribution and the two normal distributions 20 random test functions are used and 20 different realizations (per input dimensionality) are drawn from the probability density functions for the comparisons. Because there is only one deterministic LH design (per input dimensionality), the number of test functions for this input type is increased to 100. For each input dimension the number of test samples is kept constant at $N_t = 10^5$. The location of the test data samples is determined by a Sobol sequence [101] for each input dimension and is fixed for all synthetic functions. HILOMOT is used as training algorithm, see Section 2.3 for details.

The achieved test errors for an increasing amount of training data are presented in Fig. 4.5 for all three data distributions and input dimensionalities $p = 2$, $p = 5$, and $p = 8$. Each column corresponds to one input dimensionality, each row to a data distribution. Most diverse results are yielded in case of Fig. 4.5g, where the experimental



(a) Maximin optimized LH (b) Data drawn from uniform distribution (c) Data drawn from 2 normal distributions

Figure 4.4: Data used for the comparison of different order determination strategies



(a) 2 inputs, maximin LH design (b) 5 inputs, maximin LH design (c) 8 inputs, maximin LH design

(d) 2 inputs, uniform distribution (e) 5 inputs, uniform distribution (f) 8 inputs, uniform distribution

(g) 2 inputs, 2 normal distributions (h) 5 inputs, 2 normal distributions (i) 8 inputs, 2 normal distributions

Figure 4.5: Mean NRMSE on test data versus the amount of training data for different input dimensionalities and distributions

design is drawn from two normal distributions and the input dimensionality is $p = 2$. In this case, the IKMS, BGS, and the HilomotDoE based method perform equally well. MDS turns out to be the worst method on average, even worse than the randomly chosen order of experimentation. With an increasing input dimensionality, the benefit of IKMS, BGS, and HilomotDoE vanishes and the test errors of all methods get closer to each other. For input distributions yielding a good coverage of the input space, all procedures perform equally well except for MDS, which at least for two dimensional input spaces turns out to be the worst method. Altogether

MDS performs worst and is on average even beaten by random orderings. Because of this MDS results are omitted in subsequent discussions.

In Fig. 4.6 the achieved model performances are plotted against the amount of used training data. The mean normalized root mean squared error (NRMSE) of the randomly determined order of experimentation ($\overline{\text{NRMSE}}_{\text{rand}}$) is used to normalize each strategy's mean NRMSE value. All NRMSE values are calculated based on test data. The shown achieved model performances are averaged over all input distributions. Results are shown for input dimensionalities $p = 2$ and $p = 8$. The curves of all remaining input dimensions lay in between the shown ones. Compared to simple random ordering, the maximum benefit amounts to 19 % and is obtained at a usage of 20 % of the training data. The advantage of IKMS, BGS, and HilomotDoE is more pronounced for low-dimensional input spaces (or higher data densities). For almost all cases IKMS and BGS perform equally well and on a similar level as HilomotDoE (abbreviated as HDoE in the figure). Even though the active learning strategy is able to use more information, the performance is not significantly better. However, not the full potential of HilomotDoE is exploited since it can only pick data points present in the existing experimental design. Usually far more potential points would be provided to the active learning strategy in order to cover the design space better.

Figure 4.7 shows the comparison for all order of experimentation strategies, except for MDS, in case of 10 % and 50 % used training data for all input dimensions and input distributions individually. Based on the shown results, no general recommendation for BGS or IKMS can be given. All proposed methods are significantly superior to

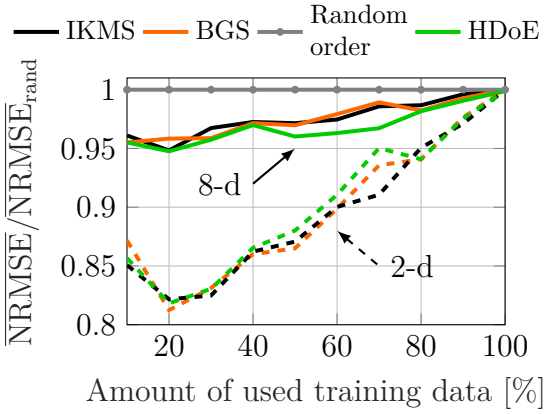
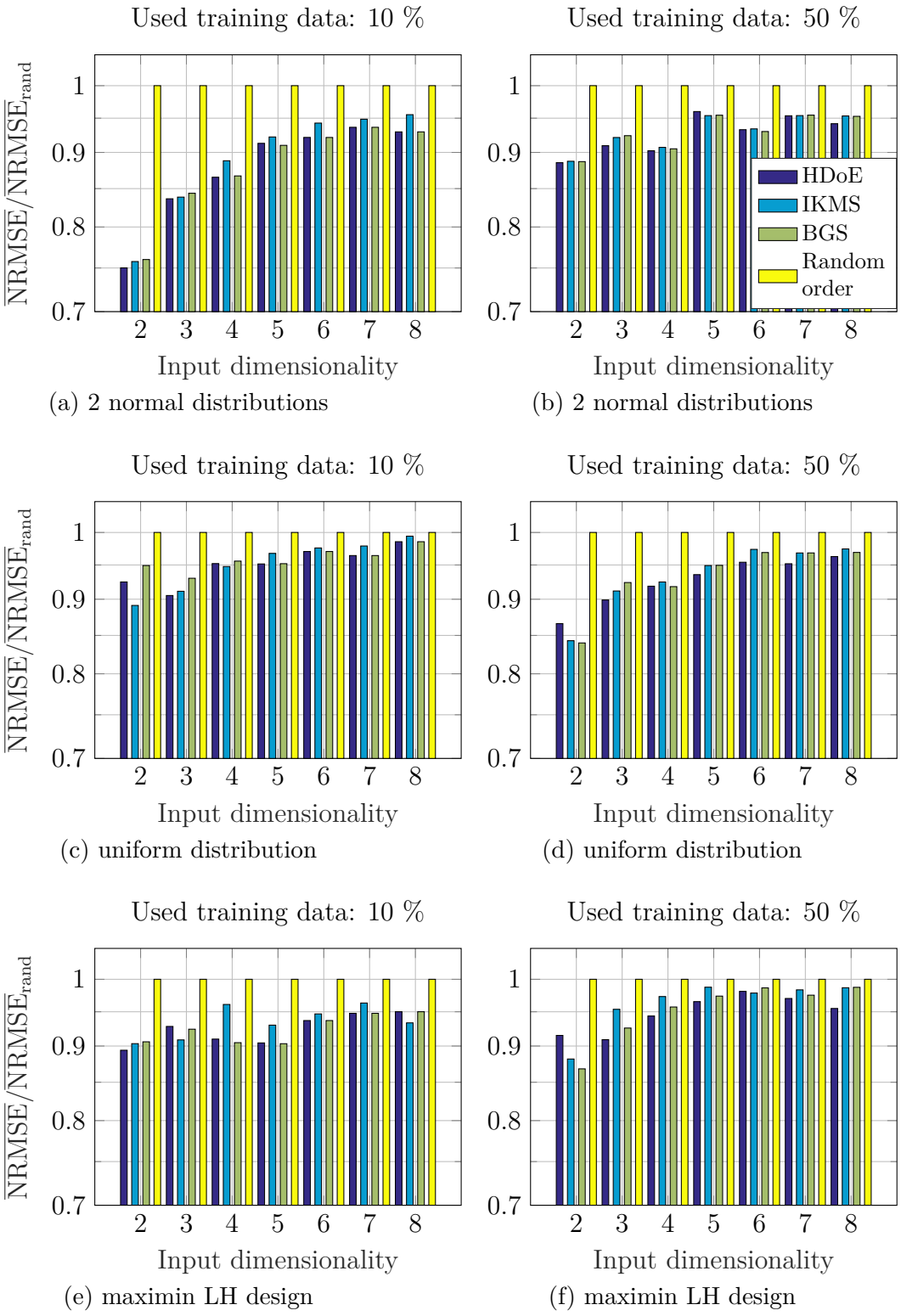


Figure 4.6: Model performances vs. the amount of used training data for $p = 2$ and $p = 8$, averaged over all input distributions



(a) 2 normal distributions (b) 2 normal distributions

(c) uniform distribution (d) uniform distribution

(e) maximin LH design (f) maximin LH design

Figure 4.7: Normalized mean NRMSE values vs. input dimensionality for 10 % (left column) and 50 % (right column) of used training data

a random ordering as it is typically carried out nowadays.

4.1.6 Summary

Models can already be used while the measurement process is still in progress. The accuracy of models in early stages of the measurement process highly depends on the order, in which the measurements are carried out. Several newly invented strategies for the order of experimentation are compared given an already existing experimental design for regression problems. The main purpose is to gain information from all sub-regions of the input space as early as possible, such that a certain reliability is ensured throughout the whole input space. With the help of a function generator, the data distribution and the data density is varied for several randomly generated synthetic functions. It is shown that especially dense and structured input data benefits from well ordered measurements. For sparsely and uniformly covered input spaces almost all presented ordering strategies perform equally well. In summary, IKMS and BGS perform in almost all scenarios equally well and are pretty close to the active learning strategy. The yielded improvements of both model-free ordering strategies (IKMS and BGS) lie between 5 % and 25 % compared to the random approach. In most cases HilomotDoE outperforms the model-free approaches, but only slightly. For real-world applications the proposed order determination strategies can be used in combination with active learning strategies. It might be a good idea to use one of the proposed order determination methods in very early stages of the measurement process, since the model lacks reliability if only very few data is available. At some point one may switch completely to the active learning strategy.

For the application on real-world test benches additional considerations are necessary. The proposed strategies are all based on distance measures. If there are hard-to- or expensive-to-vary factors on the test bench, a suitable input weighting can be used to influence the methods. Through such an input weighting, points appear closer or farther away, such that sequences can be generated, where hard-to-vary factors are only changed slightly going from point to point of the ordered list. Experimental design principles for controlling noise and bias are (I) replication, (II) blocking, and (III) randomization.

Replication cannot be influenced by the order of experimentation strategies, since it is assumed to determine the order of an already existing experimental design. If

this DoE should contain replication points or not has to be decided in advance, before the order of experimentation is determined.

Blocking is the division of the whole DoE into several, but similar groups. The motivation behind blocking is to determine the effect of variables that are not considered in the experimental design and that clearly influence the output variable. Examples for such so-called nuisance variables can be the room temperature or air humidity. Each generated group of the DoE should then be measured while the corresponding nuisance variables are at least approximately constant. Output variations between different groups can then be associated to the nuisance variables, whereas output variations within these groups can be attributed to the variables contained in the experimental design. If the a-priori determined experimental design contains several groups as a result of blocking, the strategies to determine the order of experimentation should be applied to each group individually.

Randomization is somehow already included in the proposed methods, i.e. IKMS and BGS, since biggest gaps are filled. As a result, points that are close to each other in the input space will be far away in the ordered lists generated by IKMS and BGS. Thus, the time between the measurements of two similar measuring points is increased and effects resulting from factors (inputs), that cannot be controlled, can average out.

The applicability to a real-world metamodeling task is demonstrated in Section 5.3. As the most promising methods, IKMS and BGS are used to determine the order of experimentation for the generation of a centrifugal fan metamodel, where data is yielded by computational fluid dynamics (CFD) simulations.

4.2 Advisability of Specific Experimental Designs

In the following, two questions concerning the DoE are addressed and answered. Section 4.2.1 deals with the advisability of incorporating corners (vertices) into the experimental design, i.e. the measurements of all inputs at their most extreme values. Intuitively this seems to be a good idea, but it is shown that adding these corner measurements turns out to be disadvantageous for the model quality in most cases. For the investigations several DoEs are used with and without explicitly incorporating corner measurements.

Section 4.2.2 compares commonly used space-filling experimental designs with respect to the resulting model quality. These DoEs are typically used when there is relatively little known about the process of interest and the whole design space should be explored uniformly. Especially in computer simulation experiments, leading to so-called metamodels, these experimental designs are typically utilized. It turns out that maximin LHs outperform all other space-filling experimental designs in nearly all tested cases.

4.2.1 Should Corners be Measured?

Typical human intuition favors the measurement of all inputs at their most extreme values, i.e., minima and maxima. If only box constraints exist, these combinations correspond to the corners of the input space. Measuring these corners in p dimensions is called a 2^p full factorial design in DoE language [90]. Other commonly used DoE strategies like D-optimality with polynomial model assumptions or partial fractional designs also incorporate many design space corners [90]. This section analyses in which cases (number of data points, dimensionality of the problem) measuring corners is advantageous and in which cases it is counter-productive. Obviously, in a regression context additional measurements *within* the design space are absolutely necessary to detect any nonlinear behavior. Intuitive benefits of measuring corners are:

- Extrapolation can be completely avoided if the physical minima and maxima of each input are measured. This should tighten the confidence intervals and improve reliability.
- Arbitrarily high order interactions can be discovered in principle.
- Optima with respect to the inputs frequently occur at the boundary of the input space. Thus, it might be a good idea to measure there in order to improve the accuracy of the model in these regions.

On the other hand, the number of corners of an p-dimensional input space is 2^p. The following three cases can be distinguished (N = number of data points that shall be measured):

- Low-dimensional problems ($N \gg 2^p$): The number of corners is negligible to the overall number of points to be measured. Therefore it seems reasonable to include the corners to cover the most extreme combinations.

- Medium-dimensional problems ($N > 2^p$): This is the critical case mainly discussed in this section and probably the most frequently occurring one (at least in an engineering context). It will be shown empirically that measuring corners offers increasing advantages when the amount of extrapolation and function complexity grow.

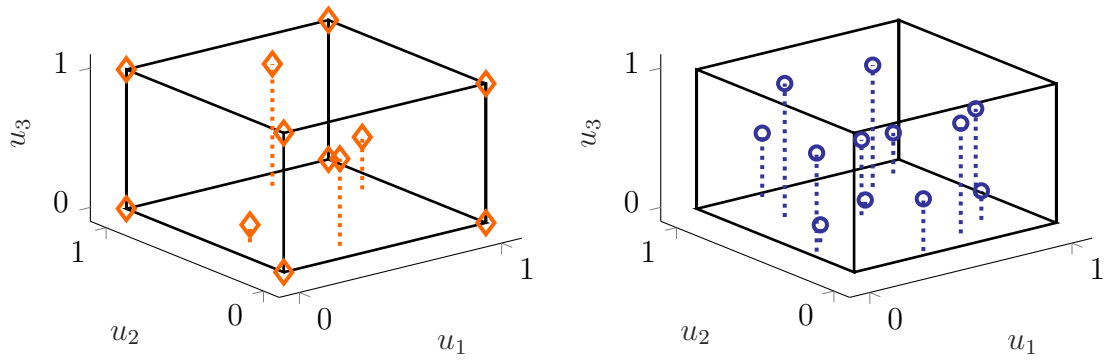
- High-dimensional problems ($N \leq 2^p$): The number of corners is similar or larger than the overall number of points to be measured. Therefore measuring corners is infeasible.

A-priori it is not clear for the second case, whether the inclusion of all corner points yields benefits for the final model performance.

In order to bring some light in the above addressed question, several design of experiments with and without explicitly added corner measurements are compared in terms of the achieved model accuracy. Therefore, the function generator described in Section 2.7 is used to generate a large amount of synthetic examples. The results are statistically evaluated.

For the comparisons made here, LH designs [86], Sobol sequences [125] and data drawn from a uniform distribution is supplemented with additional corner points. In order to achieve good space filling properties the LH designs are optimized with the extended deterministic local search (EDLS) algorithm described in [35]. Sobol sequences are inherently space-filling and the uniform distribution should also cover the design space in a more or less space-filling manner. In order to investigate the influence of corner points on the model quality, two data sets for each setting (input dimensionality p, number of points N, basic input design, i.e. LH, Sobol, and uniform) are created. One data set consists of $N_{wC} = N$ points, referred to as the designs without corners (DWC). The other data set consists of $N_C = N - 2^p$ points coming from one of the three basic input designs and is supplemented with 2^p corner points such that the overall number of samples is equal. The later one is referred to as combined design (CD). Both the CD and DWC design are compared in Fig. 4.8 exemplarily for a 3-dimensional input space and $N = 12$ samples. The comparison of the DWC and CD design corresponds to the question for which settings the explicit incorporation of corner points should be favored.

For the comparison the number of inputs is varied from $p = 2, \ldots, 8$, while the number of samples is held constant at $N = 300$. The DWC and CD designs are used for the training. In case of the data drawn from a uniform distribution and



(a) CD design with 8 corner points

(b) DWC design

Figure 4.8: Comparison of a CD and a DWC maximin LH design in a 3-dimensional input space. The number of points is $N = 12$.

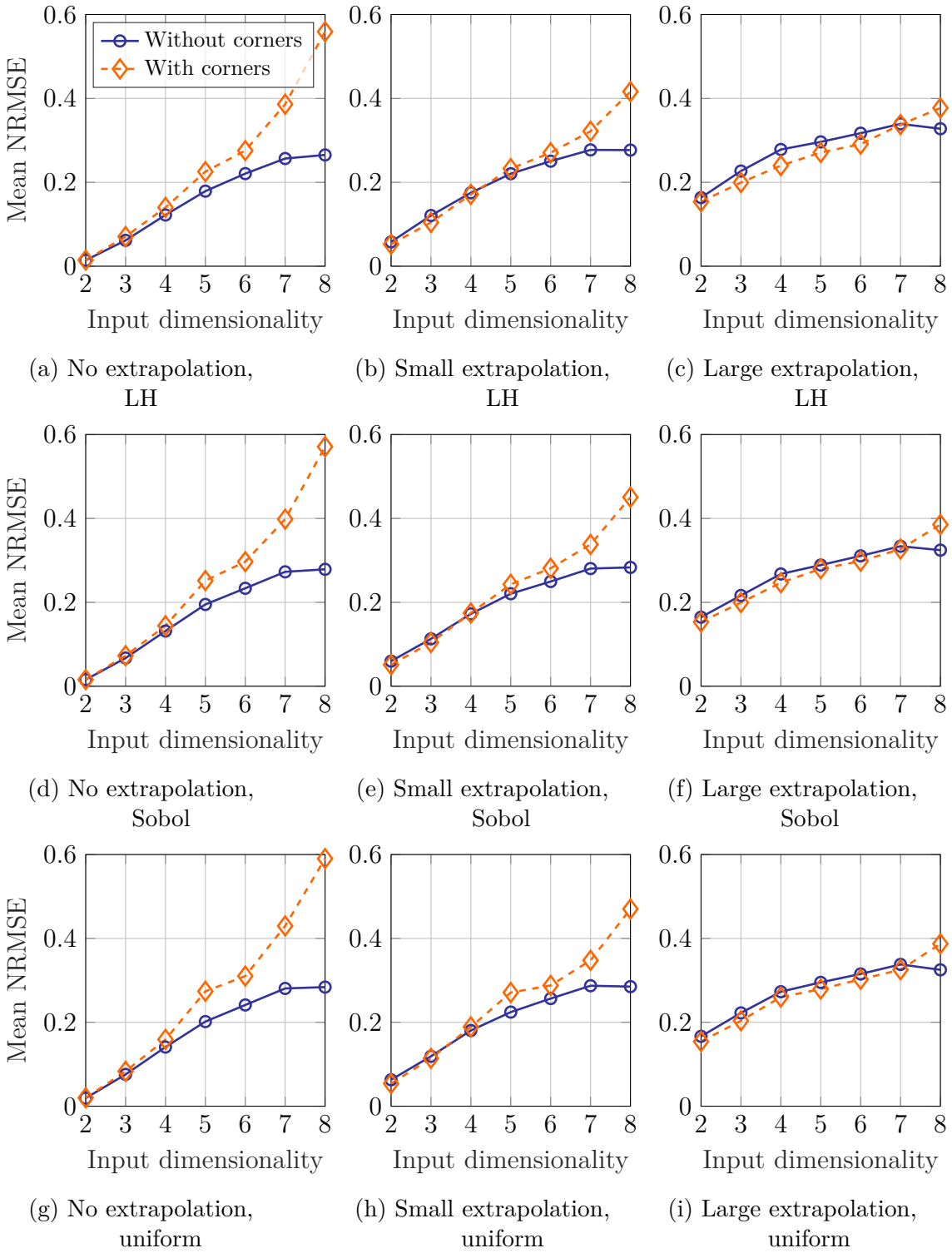
the Sobol sequences, 20 different realizations are generated and results are averaged over these 20 input designs. Only one LH design per setting is utilized. For each setting, 100 sigmoidally saturated random functions (see Section 2.7) are generated and statistically evaluated afterwards. For these functions $M = 10$ polynomial terms are used and the expected value of the exponential distribution is set to $\mu = 1$. Independently of the input dimensionality p and the case under consideration, the test data consists of $N_t = 10^5$ samples and it is generated by Sobol sequences. It is assured, that the Sobol sequences used for the test data and training data have no intersection. Therefore the models trained with data originating from the Sobol sequences should not have a significant unfair advantage compared to models trained with the other two input designs.

The training is done with the HILOMOT algorithm (see Section 2.3 and [97] for details) and three different extrapolation scenarios are investigated:

- No extrapolation: Design space for training is $[0, 1]^p$.
- Small extrapolation: Design space for training is $[0.1, 0.9]^p$.
- Large extrapolation: Design space for training is $[0.2, 0.8]^p$.

For all scenarios the test data points lie in the hypercube $[0, 1]^p$.

Results for the investigations with the help of the synthetic functions are shown in Fig. 4.9. The averaged NRMSE values are plotted against the input dimensionality for all extrapolation scenarios and all distributions of the training data. For the scenarios with no and small extrapolation, DWC designs yield smaller NRMSE



(a) No extrapolation, LH (b) Small extrapolation, LH (c) Large extrapolation, LH

(d) No extrapolation, Sobol (e) Small extrapolation, Sobol (f) Large extrapolation, Sobol

(g) No extrapolation, uniform (h) Small extrapolation, uniform (i) Large extrapolation, uniform

Figure 4.9: Comparison of designs with and without corners in terms of the achieved mean NRMSE on test data for all extrapolation scenarios and all training data distributions

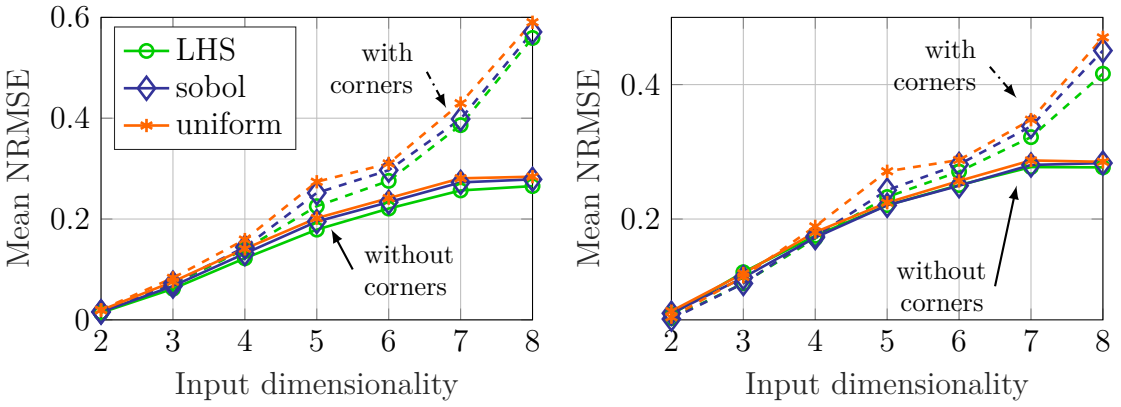
values and are therefore superior to the CD designs. In the scenario with large extrapolation the DWC designs outperform the CD designs only for the highest input dimensionality. The reason why the DWC designs are better suited for the highest input dimensionality is probably due to the high fraction of corner points in relation to the overall number of training samples. Table 4.1 shows the ratios defined by the corners of the design space (2^p) divided by the number of training samples N. Since the number of training samples is kept constant at $N = 300$, the shown ratio tends to one as the input dimensionality increases. This means that almost no samples are left to gather information from the „inside“, which is missing in order to approximate the given test function properly. The distribution of the training data seems to play no role, since the results are qualitatively very similar regardless if LH designs, Sobol sequences or data coming from a uniform distribution is used.

Interestingly the models obtained by LH designs perform always better than the one's from Sobol sequences in case of no and small extrapolation. In order to visualize this, results of all training data distributions are shown altogether in one graph in Fig. 4.10a in case of no extrapolation and in Fig. 4.10b in case of small extrapolation. This observation is investigated in more detail in Section 4.2.2.

Another noticeable result that can be seen in Fig. 4.9 is the decrease in the mean test NRMSE value for large extrapolation demands at least for higher input dimensionalities ($p > 6$). This phenomenon is counter-intuitive and occurs only in the CD designs but regardless of the used training data distribution. It also appears for different model types. The same phenomenon was observed with multilayer perceptron networks (see [96] for details about this model type) and Gaussian process models as described in detail in [107]. To clarify this phenomenon, designs only consisting of corner points are investigated, since the number of non-corner points is negligible in cases where this phenomenon is observed. The objective function J_t is the error on the test data which always lies in the hypercube $[0, 1]^p$ and is optimized with respect to the region $[b, 1-b]^p$ into which the training data is scaled down to, compare

Table 4.1: Ratios between the corners of the design space (2^p) and the number of training samples $N = 300$

Input dimensionality p	2	3	4	5	6	7	8
Ratio $2^p/N$	0.0133	0.0267	0.0533	0.1067	0.2133	0.4267	0.8533



(a) No extrapolation (b) Small extrapolation

Figure 4.10: Comparison of different distributions of the training data for no (a) and small (b) extrapolation

Fig. 4.11:

$$b_{opt} = \arg\min_{b} J_t \,,$$

The value b_{opt} leads to the minimal test error and corresponds to a specific region in which which the training data is located. This optimization problem is solved for all 100 eight-dimensional test functions generated with the function generator. On average b_{opt} turned out to be 0.2 with a standard deviation of 0.03. Assuming that each training data point supports a model in a local region around it with information about the process that should be approximated, this is a reasonable result. The region supported by a training data point is visualized in Fig. 4.11 for two cases corresponding to (i) $b = 0$ (upper right corner of the test data region) and (ii) $b > 0$ (lower left corner of the training data region). As can be seen for case (i), most of the supported region lies outside the test data hypercube and does therefore not contribute to a lower error on test data. Case (ii) covers more hypervolume of the test data hypercube leading to a better model performance on the test data. Note that the shape and size of the shown support regions in Fig. 4.11 are chosen arbitrarily to explain the mechanism responsible for the counter-intuitive observation that larger extrapolation demands lead to a better model performance. Of course the true shape and size of such regions is unknown and may vary with the location in the input space as well as with the specific process under consideration.

In summary, it can be recommended to measure corners only if “enough” data points remain inside the design space and large extrapolation is required. Another impor-

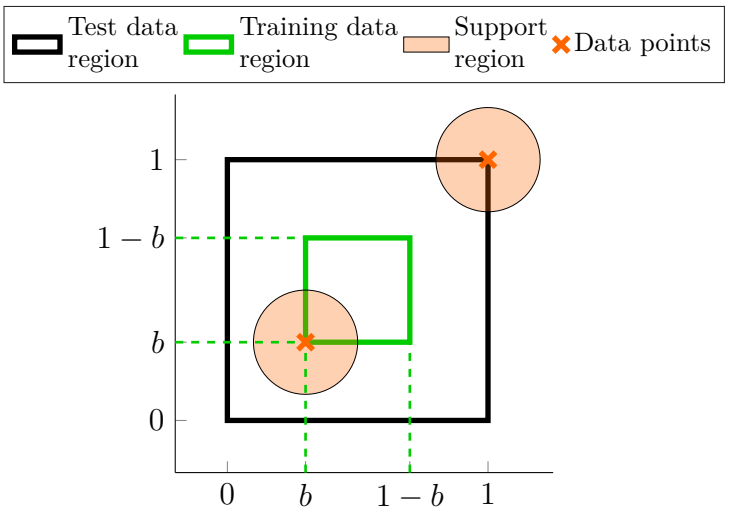


Figure 4.11: Visualization of the test data region, the training data region depending on b, and exemplary support regions for two data points

tant outcome is the superiority of maximin LH designs compared to Sobol sequences in most cases, leading directly to the investigations of Section 4.2.2.

4.2.2 Comparison of Space-Filling Experimental Designs

Commonly used space-filling experimental designs are compared by means of the model quality achieved with them. These DoEs are typically chosen for computer simulation experiments, leading to so-called metamodels, see Section 2.6 for details. In cases where there is relatively little prior knowledge about the process of interest, space-filling experimental designs have the advantage to explore the whole design space uniformly. This might be one reason for their popularity in the field of metamodel-based design optimization (metamodel-based design optimization (MBDO)), in which data is sequentially collected in regions of the design space, where an optimum is suspected. Note that the term *design* in MBDO refers to constructive design and not to the experimental design.

The experimental designs that are compared are several maximin LHs, originating from different optimization strategies, Sobol sequences and data coming from uniform distributions. Examples where LH designs are chosen as DoE are e.g. [136], [135], [137], [72], and [21]. Sobol sequences are increasingly used for computer experiments according to [104]. Examples for work in which they are used are [38], [134], and [28].

Most of the published related work, such as [67, 65, 66], [68], [49], and [24], relies on comparison criteria based on measures of the input space coverage or on the estimated prediction variance. In contrast to that, the comparison criterion used here is the achieved model quality. Again, the random function generator described in Section 2.7 is used to build several test functions. Through the randomness incorporated in the function generator a variety of different characteristics can be imitated and a favoring of any experimental design through a specific choice of functions is avoided. Once the test functions are generated, an arbitrary amount of data samples for training or testing purposes can be produced. Chen et. al. [25] conducted a comparison of some experimental designs based on the resulting model quality, but only for far less test functions compared to this work.

Three types of maximin LH designs are used for the comparison. Two of the maximin LH designs are obtained by the optimization with the EDLS algorithm [35]. The third maximin design is obtained by the function implemented in the commercially available software Matlab. As an additional reference, data drawn from a uniform distribution is also incorporated in the comparison.

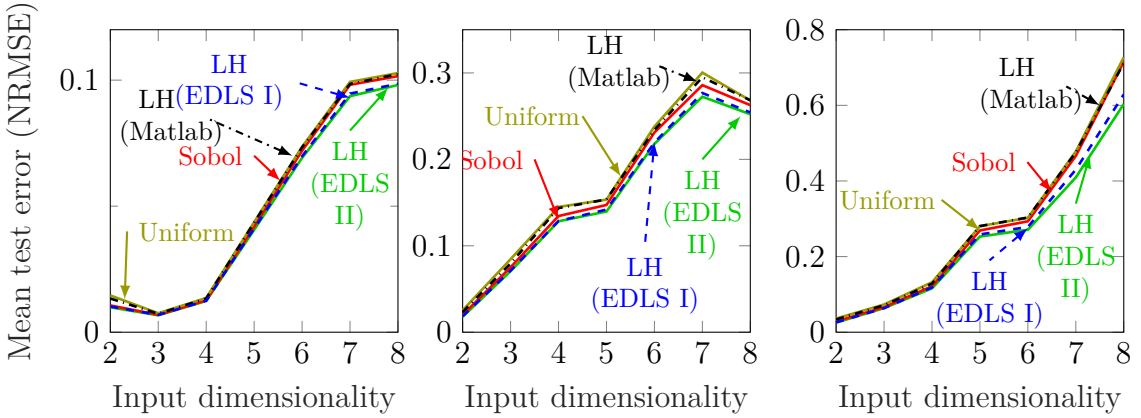
The EDLS algorithm tries to maximize the distance between the point pair having the smallest nearest-neighbor distance in a given data set (maximin criterion) by coordinate exchanges. Therefore the distances between all possible point pairs have to be determined and coordinates of the points having the smallest distance to each other are tentatively exchanged with other points. If such an tentative exchange leads to an improvement of the maximin criterion, it is actually carried out. In case no exchange partner can be found that leads to an improvement of the maximin criterion, phase one of the EDLS algorithm is terminated. In phase two of the EDLS algorithm additional points are considered for coordinate exchanges. These additional points are point pairs having the second (third, fourth, etc.) smallest distance to each other. For more details the reader is referred to [35].

The input space dimensionality is varied from $p = 2, \ldots, 8$ while the number of training samples is held fixed at $N = 300$. The test data to assess the model quality consists of $N_t = 10^5$ samples coming from a Sobol sequence independent of the input dimensionality. The high number of test samples compared to the training data size should guarantee a good measure of the generalization performance of the obtained models. It is assured that the Sobol sequence for the model quality assessment and the ones used for the training do not contain the same data points. For each investigated input dimensionality 30 different realizations of each experimental design

are generated and serve as training data for 30 different test functions created by the function generator. The realizations of the EDLS-optimized LH designs are obtained by varying the initial LH design. The design achieved after finishing phase one of the EDLS algorithm will be abbreviated as LH (EDLS I) in the following. Phase two takes the result of phase one and continues the optimization until convergence. The optimization result of phase two will be abbreviated as LH (EDLS II). For the Sobol sequences different parts of a very long sequence are taken.

The aforementioned settings for the comparison are done for three different complexities of the test functions. Through the variation of the μ parameter (see Section 2.7) functions are likely to arise that are of rather simple complexity ($\mu = 0.5$), of medium complexity ($\mu = 1$), and of high complexity ($\mu = 2$). All models are LMNs trained with HILOMOT as explained in Section 2.3. The model complexities are determined with the help of the Akaike's information criterion (AIC_c).

Figure 4.12 shows the mean test errors of all experimental designs versus the input dimensionality for all test function complexities. For each input dimensionality the results of all 30 test functions and 30 input data realizations are averaged to obtain the mean values for each experimental design. The NRMSE is calculated according to (2.6) and is employed on $N_t = 10^5$ test data samples. Lower NRMSE values indicate better generalization performance. The difference between the model qualities achieved with the investigated experimental designs increases with an increasing complexity of the test functions and an increasing input dimensionality. In all cases



(a) Simple complexity ($\mu = 0.5$)

(b) Medium complexity ($\mu = 1$)

(c) High complexity ($\mu = 2$)

Figure 4.12: Mean test errors of functions that are of simple complexity ($\mu = 0.5$), medium complexity ($\mu = 1$), and high complexity ($\mu = 2$)

the maximin LH designs optimized with the EDLS algorithm yield the best results. The continuation of the EDLS optimization (phase two) after phase one has completed brings advantages especially for complex functions ($\mu = 2$) and high input dimensionalities, see Fig. 4.12c. The next best experimental design according to the achieved model quality is the Sobol sequence. The maximin LH designs obtained by the Matlab function yield slightly better model performances compared to the data coming from the uniform distribution.

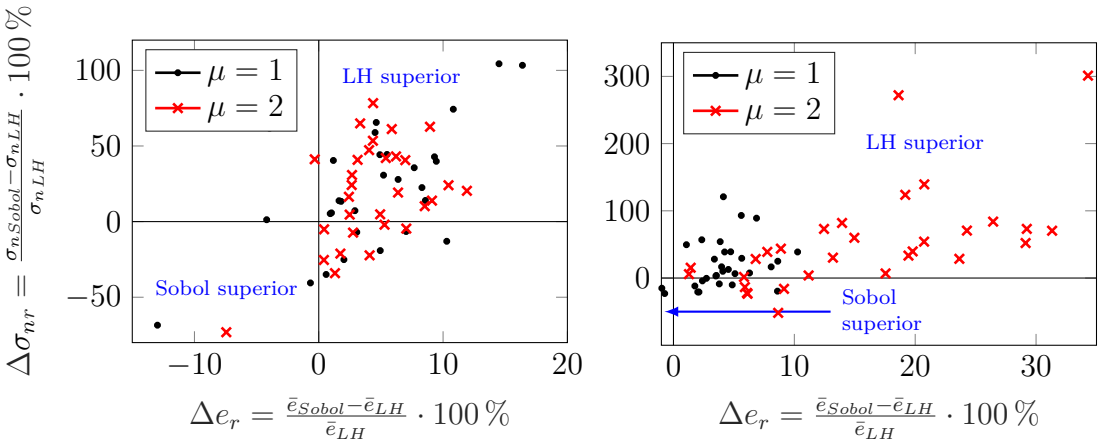
The variability due to varying test functions is far bigger than the variation due to different realizations of the experimental designs. Here, the standard deviations of the model errors resulting from the different realizations of the experimental designs are of interest. Therefore the standard deviations for each test function are compared. If an experimental design reveals a small standard deviation in the corresponding test errors, the danger of obtaining a bad DoE by picking just one realization for a specific task is small. This is an extremely appealing property in combination with a low mean value of the expected model error. In order to compare the best maximin LH design (EDLS II) with the Sobol sequences, the relative difference between the mean errors $\bar{e}_i$

$$\Delta e_r = \frac{\bar{e}_{Sobol} - \bar{e}_{LH}}{\bar{e}_{LH}} \cdot 100\,\%\,, \tag{4.2}$$

and between the standard deviations σ_{ni}

$$\Delta \sigma_{nr} = \frac{\sigma_{nSobol} - \sigma_{nLH}}{\sigma_{nLH}} \cdot 100\,\%\,, \tag{4.3}$$

is used. According to the definitions of (4.2) and (4.3), positive values indicate the improvement of the EDLS-optimized maximin LH design in percent compared to the Sobol sequence. Figure 4.13 shows the values of Δe_r and $\Delta \sigma_{nr}$ for all 30 medium and complex test functions together in one graph for input dimensionalities $p = 4$ and $p = 8$. Each dot and each cross corresponds to one test function. The upper right corner of the graph contains test functions for which the maximin LH design leads to better mean test errors ($\Delta e_r > 0$) and lower standard deviations ($\Delta \sigma_{nr} > 0$), which is the case for most of the 60 shown test functions in each graph. The selected graphs, i.e. Fig. 4.13a and 4.13b, show the results for the input dimensionalities where the EDLS-optimized maximin LH designs come off worst and best. The improvements in the relative mean error difference and the standard deviation difference go up to 35 % and 300 %, respectively. The maximum decline in Δe_r is about 15 % and 70 % in $\Delta \sigma_r$. However, in far more cases the EDLS-optimized LH designs are superior to Sobol sequences in both aspects.



(a) Input dimensionality $p = 4$ (b) Input dimensionality $p = 8$

Figure 4.13: Relative mean test errors and standard deviations for all 30 test functions of two complexities ($\mu = 1$, $\mu = 2$) for $p = 4$ and $p = 8$.

Figure 4.14 shows the required mean computation time for the generation of all investigated experimental designs. In this category the Sobol sequences are superior to all LH designs and are only outperformed by the uniform designs. On an absolute scale the optimization of the EDLS I LH designs takes up to one minute on average, while the generation of EDLS II designs consumes up to 660 minutes on average. The creation of all other designs takes far less than one second.

Unfortunately it is not possible to clarify with certainty what the reasons are that in almost all cases the maximin LH designs outperform the other investigated experimental designs regarding the mean model quality. The superiority regarding the variance in the model qualities originates probably due to the use of an optimization

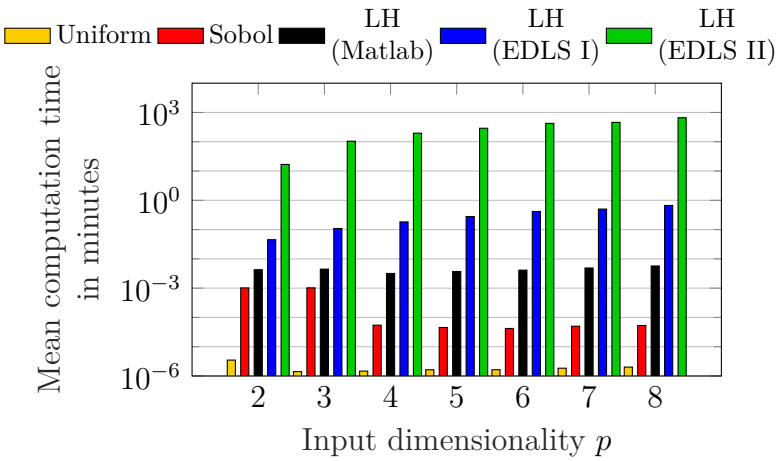
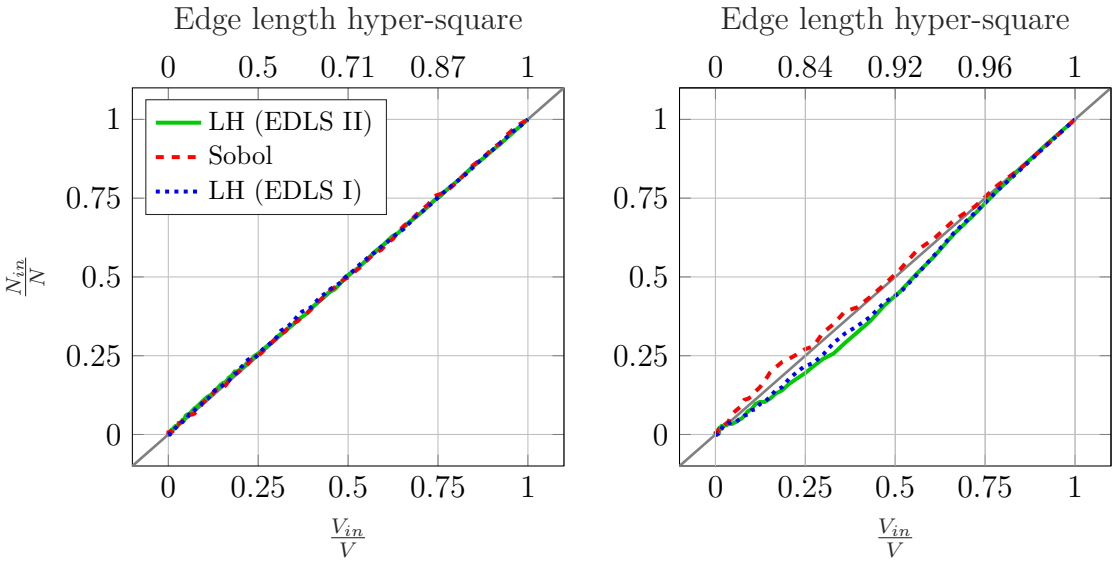


Figure 4.14: Mean computation time for the generation of the experimental designs

algorithm in order to generate the training data. Through the optimization the variation between different training data sets should be very low which finally results in a lower model quality variance. However, one property of the designs is investigated in some more detail in the following in order to find a hint that might explain the superiority regarding the better mean model quality. For the comparison of the space-filling designs carried out in this section it is assumed that designs covering the input space uniformly are advantageous for the expected model quality. In the opinion of the author, one crucial point is to fulfill the uniformity requirement for reasonable few data samples. For example data drawn from a uniform distribution should cover the input space uniformly, but quite a lot of samples are needed until clumps and empty spaces vanish. Figure 4.4b shows $N = 300$ samples drawn from a uniform distribution in a two-dimensional input space and clumps as well as empty spaces are apparently present. A commonly used measure to quantify the geometric non-uniformity of points is the L_∞-discrepancy [123]. For any data set containing N points it is defined as:

$$D_N^* = \sup_{K \subseteq \mathbb{R}^p} \left| \frac{N_{in}(K)}{N} - \frac{V_{in}(K)}{V} \right| . \tag{4.4}$$

In this equation K defines a hypercube contained within the p-dimensional input space $\mathbb{R}^p$ with one corner of it located at the origin $\underline{0}$. The number of points inside K is N_{in} and the hypervolume of K is V_{in}. V corresponds to the total hypervolume in which all data points N are contained. In other words, a hypercube is sought in which the discrepancy between the proportion of points inside the hypercube differs the most from the proportion of points that *should* be present according to the volume proportion of that hypercube. In Fig. 4.15 the proportion of points is plotted against the proportion of hypervolume of a hyper-square for the two- and eight-dimensional case. The shown curves correspond to the worst L_∞-discrepancy of all 30 input data realizations of the maximin LH designs, i.e. EDLS I and II, and the Sobol sequences. On top of each shown figure there is an alternative x-axis showing the edge length of the hyper-square that corresponds to the hypervolume V_{in}. The closer the curves are to the diagonal, the lower is the discrepancy measure according to (4.4). Curves lying below the diagonal represent a too low data density and vice versa. For the two-dimensional case shown in Fig. 4.15a all experimental designs lie upon the diagonal. With an increasing input dimensionality the curves deviate more and more from the diagonal. On average the Sobol sequences tend to be a little bit above the diagonal whereas the maximin LH designs are typically below it. The interpretation of this observation is that the LH designs possess more points very close to the boundary



(a) $p = 2$ (b) $p = 8$

Figure 4.15: Proportion of points inside a defined hypervolume to all points (N_{in}/N) versus proportion of that hypervolume to the total hypervolume (V_{in}/V) for input dimensionalities $p = 2$ and $p = 8$

of the outer hypercube. In an eight-dimensional input space the ratio $V_{in}/V = 0.5$ corresponds to an edge length of ≈ 0.92, see Fig. 4.15b. At this point the curves of the two LH designs lie clearly below the diagonal, meaning that in relation to the proportion of the hypervolume too few data samples lie inside the hyper-square. Viewed from a different angle the number of points in the remaining hypervolume $(V - V_{in})$ has to be relatively high, because the point $(V_{in}/V = 1, N_{in}/N = 1)$ has to be met exactly. Since the advantage of the maximin LH designs with respect to model quality increases with the input dimensionality, see Fig. 4.12, this property of the training data set seems to be favorable w.r.t. to the model quality.

In summary, maximin LH designs created by the EDLS algorithm (and probably other optimization schemes as well) are superior to all other investigated experimental designs regarding both the achieved model qualities and the variation of these model qualities. These advantages have to be paid off by higher computational effort for the DoE. However, the LH designs can be optimized and stored in advance to a specific task, such that this drawback can be weakened at least for some applications. Another drawback of the maximin LH designs regards their extensibility. If just a few points should be added to an already optimized LH design it is not as easy as to append additional points to an existing Sobol sequence, because further optimization runs are necessary. The results are important for both the initial DoE

Table 4.2: Advantages and drawbacks of Sobol sequences and maximin LH designs

Maximin LH designs	Sobol sequences
+ Better model qualities on average	– Lower model qualities on average
+ Lower deviations in the model qualities (even worst case delivers satisfactory results)	– Higher deviations in the model qualities (worst case might be very inferior)
– Higher computational effort and therefore more time-demanding to generate the experimental designs	+ Lower computational effort and therefore less time-demanding to generate the experimental designs
– Difficult to add further design points incrementally	+ Easy to add further design points incrementally
– Not suited for very high amounts of data samples ($N > 10^4$) because optimization takes too long	+ Very high amounts of data samples not a problem
• Higher L_∞-discrepancy on average (theoretically a drawback, but has a positive effect in the investigations of this section)	• Lower L_∞-discrepancy on average (theoretically an advantage, but has no positive effect in the investigations of this section)

and subsequent refinements of it in presumably near-optimum regions of the design space. Table 4.2 compares the advantages and drawbacks of the Sobol sequences and the maximin LH designs.

4.3 Goal-Oriented Active Learning with Local Model Networks

A methodology for goal-oriented active learning with LMNs is presented in the following. The basic concept arose by reason of a real-world application where training data for a CFD metamodel should be generated. However, this section focuses not on the application to the problem at hand but describes the developed methodology in a more theoretical context. The application is presented in Section 5.3. The developed methodology is invented for (meta-)models that try to describe a technical system universally - not only one occurrence of it. In other words, the model should be able to describe the technical system well in general and does not only focus on specific areas of the design space, like optima-near regions as metamodel-based design optimization (MBDO) techniques do.

The goal-oriented nature originates from three main targets that are addressed simultaneously during the active learning procedure. (I) The concentration on possibly near-optimum regions and (II) the focus on areas in the design space where the (meta-)model's performance is considered to be worst. Additionally, (III) new measurements should differ from already gathered data as much as possible. With these goals three important issues in modeling are addressed simultaneously: (I) optimality, (II) model bias, (III) model variance/uniformly space-filling property. In order to fulfill all goals, special properties of LMNs are utilized (embedded approach). Through the structure of LMNs it is possible to assign *local* model errors to specific regions in the input space. New measurements are preferably placed in such high-error zones, while concentrating on presumably near-optimum regions that differ most from the already existing training data. As will be shown in the application section, the invented methodology is able to extend the area of achievable design points in the input space and leads to a better exploitation of optimum-near regions.

As already described in Section 2.5, active learners can be distinguished by their query strategies, i.e., how they choose the next query. The herein presented goal-oriented active learning strategy is an extension of the HilomotDoE algorithm proposed in [55]. The main focus of the query strategy of HilomotDoE is to reduce the error of an LMN, as described in some more detail in Section 4.3.1. Since the error of the model can be decomposed into a bias and variance part [96], HilomotDoE

addresses and reduces both error parts as described in [56]. There are two main innovations compared to the already existing HilomotDoE algorithm. One innovation regards the generation of so-called *candidate points*. Candidate points are *potential* queries from which only one or a small subset is chosen to be measured. In the original HilomotDoE algorithm, candidate points are generated randomly by drawing values from a uniform distribution or Sobol set. The goal-oriented active learning strategy generates the candidate points in a completely deterministic way as described in more detail in Section 4.3.2. The other innovation regards the generation of more than one query. There are already existing strategies to pick more than one query with the HilomotDoE algorithm described in [56]. The strategy outlined in Section 4.3.1 offers advantages especially when dealing with MIMO systems.

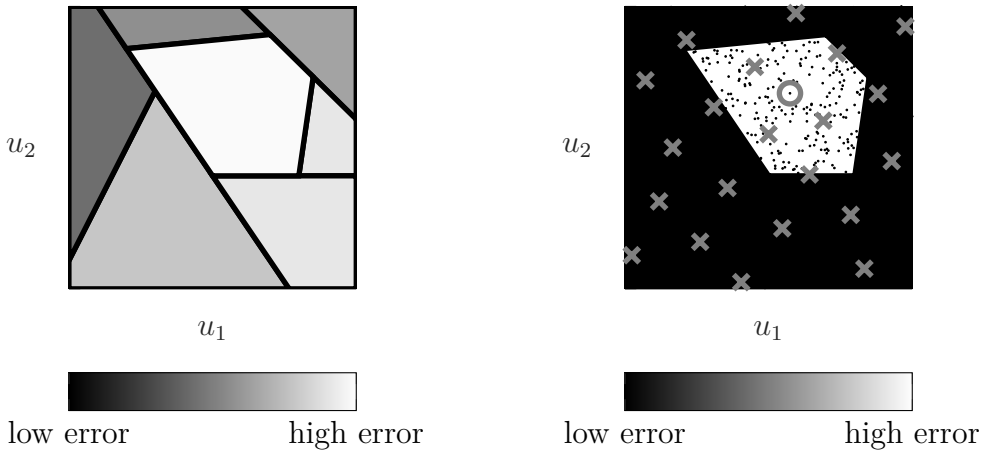
4.3.1 Active Learning with Local Model Networks

As already mentioned, the active learning strategy with LMNs aims at reducing the error of the final model. It is assumed that the highest error reduction can be obtained if the queries are placed in areas of the input space with the highest local error measure $e_{LM,i}$, $i \in \{1, \ldots, M\}$ with M being the number of local models. For the calculation of the local error measures $e_{LM,i}$ the measured outputs $\underline{y}$, the local model outputs $\underline{\hat{y}}_i$, the effective number of local parameters $n_{eff,i}$ and the local validity matrix $\underline{Q}_i = \mathrm{diag}(\Phi_i(\underline{u}))$ are required:

$$e_{LM,i} = \sqrt{\frac{(\underline{y} - \underline{\hat{y}}_i)^{\mathrm{T}} \underline{Q}_i (\underline{y} - \underline{\hat{y}}_i)}{\mathrm{trace}(\underline{Q}_i) - n_{eff,i}}}\,,\ i \in \{1, \ldots, M\}\ . \tag{4.5}$$

Each squared error between a local model output and the measured output is weighted with the corresponding validity value and is divided by the leftover degrees of freedom. For the sake of simplicity (4.5) handles only the case with one output (multiple input single output (MISO) case). For the MIMO case a local error measure arises for each output individually. For more details on how to compute the effective number of parameters $n_{eff,i}$ for each local model please refer to [56].

Figure 4.16 visualizes the situation during the active learning phase for a two-dimensional input space if only one query is sought. An LMN is trained with HILOMOT based on all currently available data. As one result the partitioning of the input space is obtained and for each local model a local error measure according to (4.5) can be calculated, see Fig. 4.16a. In the original, not yet extended version



(a) Qualitative local errors

(b) Training data (x), candidates (·) and chosen query (o)

Figure 4.16: Partitioning of an LMN with local errors (a) and with focus on the worst local model together with training data (x), candidates (·) and the chosen query (o)

of HilomotDoE, candidate points are generated through random sampling from a uniform distribution (dots in Fig. 4.16b). From all randomly generated candidate points only the ones lying in the local model with the worst local error measure are considered in the following. The candidate point inside the worst local model that fills the greatest hole of the already measured training data is chosen as query. In order to find the greatest hole, the nearest neighboring training data point to each considered candidate is determined. The candidate point with the greatest distance to its nearest neighboring training data point is chosen as query. During the distance calculations, all training data samples are considered, but only candidates within the worst performing local model are taken into account. After the measurement at the current query has finished, the training data set is updated, a new LMN is trained with HILOMOT, and the next query can be determined.

For algorithmic efficiency reasons it might be reasonable to request more than one query at a time. If $n_q > 1$ queries are demanded, a new strategy for HilomotDoE is proposed. Still HILOMOT is used to train an LMN and yield a partitioning of the input space. Then, all local error measures are calculated and normalized, such that their sum equals one:

$$\hat{e}_{LM,i} = \frac{e_{LM,i}}{\sum_{j=1}^{M} e_{LM,j}} \,. \tag{4.6}$$

The number of demanded queries from each local model $n_{qLM,i}$ is obtained by rounding the overall number of demanded queries n_q multiplied with the corresponding normalized local error measure $\hat{e}_{LM,i}$:

$$n_{qLM,i} = \lfloor n_q \cdot \hat{e}_{LM,i} \rceil \ . \tag{4.7}$$

With this approach more queries are demanded from a local model, the bigger its local error measure is in relation to the local error measures of all other local models. Again, for the sake of simplicity (4.6) and (4.7) consider only the MISO case, but an extension for MIMO systems is straightforward. The normalized error measures $\hat{e}_{LM,i}$ have to be extended by an index specifying the output number. Additionally an extra (possibly weighted) summation over all outputs has to be included in the denominator of (4.6). Then, (4.7) has to be evaluated for each output and the number of requested queries is obtained by summing up these queries for each local model and each output.

4.3.2 Generation of Candidate Points

As already mentioned, the other novelty is the deterministic generation of the candidate points which are subsequently used by HilomotDoE. A block diagram of the whole goal-oriented active learning procedure is shown in Fig. 4.17. The candidate points are generated through an optimization based on the currently available model, trained with all yet accessible measurements. With the help of the model the objective function $J(\underline{u})$ can be determined for each design vector $\underline{u}^T = \begin{bmatrix} u_1 & u_2 & \cdots & u_p \end{bmatrix}$. In order to distribute the initial values for the optimization runs in a space-filling manner throughout the whole input space, the EDLS algorithm proposed in [35] is used to generate a maximin LH design. To prevent the optimization from generating too similar candidate points, n_c constraints for each initialization point coming from the LH design are applied. If no constraints would be demanded all optimizations lead to the same optimum assuming the optimization finds the global optimum. More details about the used constraints follow in the next paragraph. All resulting candidate points are provided to HilomotDoE, that determines queries as described in previous paragraphs (biggest hole criterion). After the queries are measured, they are added to the available measurements. Through the altered training data the model and therefore the outcome of optimizations with the new model might change, even if neither the constraints nor the LH design is changed.

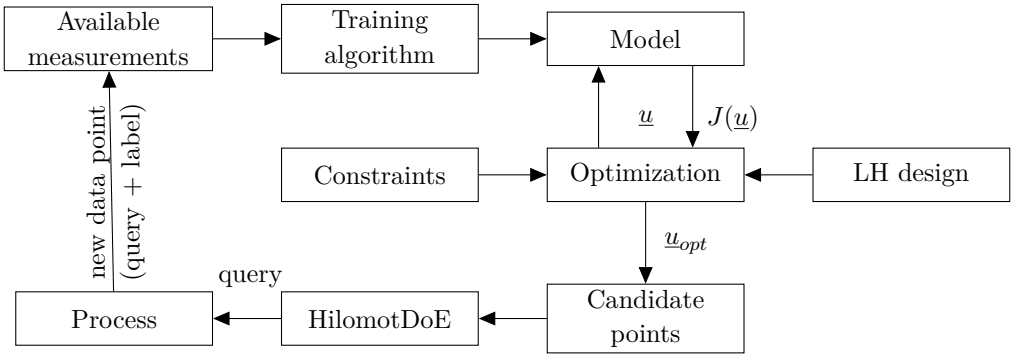


Figure 4.17: Block diagram of the goal-oriented active learning procedure

Different constraints induce a desired variability in the optimization solutions by forcing the optimization to meet the constraints. These constraints might reflect later usage scenarios, if some design parameters are bounded, e.g. due to limited available construction space. Including optimal solutions meeting such constraints in the training data set should improve the resulting model quality for these application scenarios. In contrast to that, variations in the optimization solutions due to local optima or convergence problems are not desired. The corresponding geometries are not of special interest for later usage scenarios since they are obviously suboptimal. If the number of optimization variables is p, there are $n_c = 2^p$ possible constraint combinations specifying which variables are held fixed. For each point contained in the LH design all possible constraint combinations are applied. Constrained optimization variables are held fixed at the value of the corresponding LH design point. As an example Fig. 4.18 together with Table 4.3 illustrate this for $p = 2$ optimization variables for the i-th point of the LH design. Four optimization scenarios arise with different constraints, all listed in Fig. 4.18 and Table 4.3. The variable $u_{j,LH}$ denotes the value coming from a point in the LH design that is fixed during the corresponding optimization run. In case of „opt.“ in a field of Table 4.3, the value of the corresponding variable is yielded by the optimization. With this approach $N_c = N_{LH} \cdot n_c = N_{LH} \cdot 2^p$ optimization runs are necessary to obtain all candidate points.

The presented goal-oriented active learning strategy is very process oriented, since candidate points are concentrated around near-optimum regions given some deterministically specified constraints. Because of the high specificity of the HilomotDoE extension no adequate artificial example is created. However, the usefulness of the presented strategy is demonstrated in Section 5.3 for the active generation of training data for a CFD metamodel.

Table 4.3: Each column contains one optimization scenario. An optimization variable u_j might either be constrained to a specific value ($u_{j,LH}(i)$) or can be free, i.e., the value is determined through the optimization (opt.).

Optimization scenario:	1	2	3	4
Opt. variable u_1	opt.	opt.	$u_{1,LH}(i)$	$u_{1,LH}(i)$
Opt. variable u_2	opt.	$u_{2,LH}(i)$	opt.	$u_{2,LH}(i)$

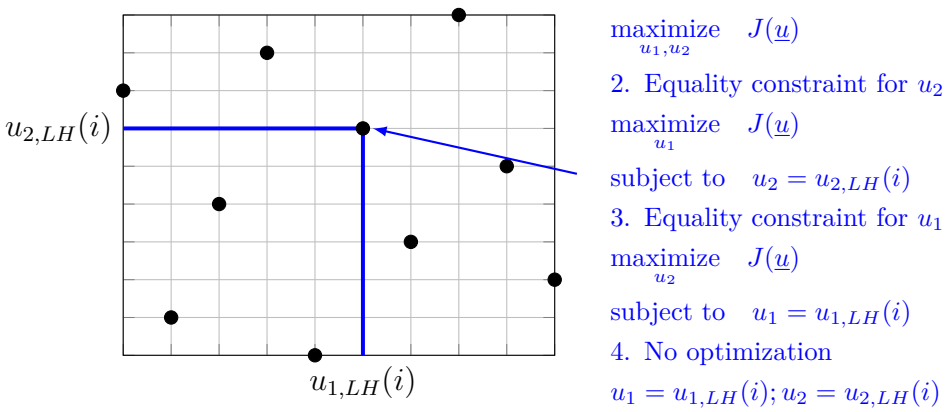


Figure 4.18: Illustration of all optimization scenarios for point i of an LH design with two inputs

5 Applications

In this chapter some applications of the proposed input selection and design of experiments (DoE) techniques are presented. The shown applications prove the usefulness and applicability of the concepts and methods described in Chapters 3 and 4. The first application example presented in Section 5.1 considers the auto miles per gallon (MPG) data set that is freely available from the machine learning repository of the University of California Irvine (UCI) [80]. The second application presented in Section 5.2 deals with the intake manifold of an combustion engine. Computational fluid dynamics (CFD) metamodels predicting the efficiency of centrifugal and axial fans are topic of Section 5.3. Some of the developed DoE techniques are applied for the generation of a data set for the centrifugal fan CFD metamodels. The mixed wrapper-embedded input selection is applied to both the centrifugal and the axial fan CFD metamodels in Section 5.3.3. The last application concerns a heating, ventilating, and air conditioning (HVAC) system and is shown in Section 5.4. The first three applications deal with static modeling tasks, while the aim in the HVAC application is to build a dynamic model for control design of the process.

5.1 Miles Per Gallon Data Set

The MPG data set concerns city-cycle fuel consumption to be predicted in terms of three multi-valued discrete and four continuous input variables [106]. The input variables are:

- Number of cylinders u_1 (multi-valued discrete),
- displacement u_2 (continuous),
- horsepower u_3 (continuous),
- weight u_4 (continuous),

- acceleration u_5 (continuous),
- model year u_6 (multi-valued discrete), and
- origin u_7 (multi-valued discrete).

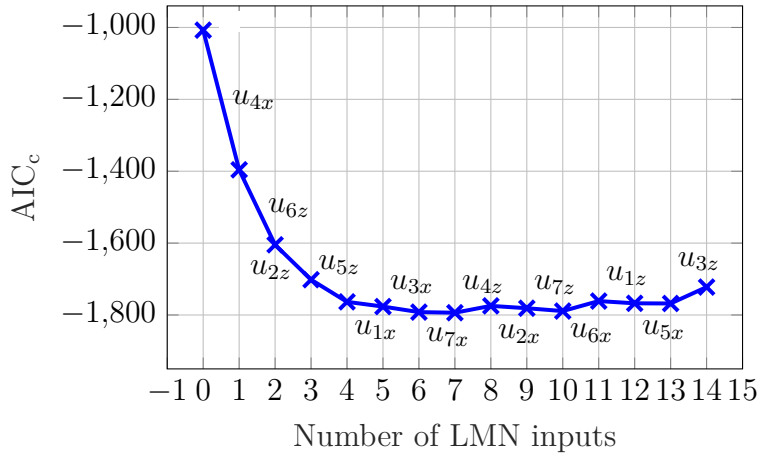
The fuel consumption is measured in miles per gallon (MPG); higher MPG values correspond to lower fuel consumption. The used data set is obtained from the machine learning repository of the University of California Irvine (UCI) [80] and contains $N = 392$ samples. This data set is split into a training and test data set containing $N_{train} = 333$ and $N_{test} = 59$ samples, respectively. For the splitting of the data set a deterministic and distance-based algorithm is used with the goal to avoid extrapolation and is explained in detail in Appendix A.

The auto MPG data set is used to demonstrate the abilities of the separated $\underline{x}$-$\underline{z}$-input selection in Section 5.1.1. In addition to that, the regularization-based input selection (RBIS) approach is tested on the real-world example in Section 5.1.2. Finally, partial dependence plots are shown and discussed in Section 5.1.3.

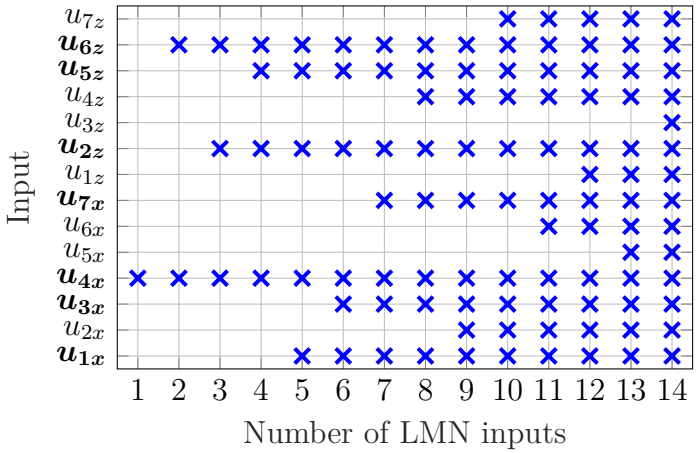
5.1.1 Mixed Wrapper-Embedded Input Selection

For the investigation with a separated $\underline{x}$-$\underline{z}$-input selection the search strategy as well as the evaluation criterion are chosen according to the recommendations in Section 3.2.2, i.e. backward elimination (BE) and Akaike's information criterion (AIC_c), respectively. Local affine models are used for the local model networks (LMNs) trained with the HIerarchical LOcal MOdel Tree (HILOMOT). The model complexity of the LMNs is determined with the help of the AIC_c. Due to the input selection scheme, there are 14 possible LMN inputs.

Figure 5.1a shows the AIC_c values versus the number of LMN inputs. The best AIC_c value is obtained with a subset size of seven LMN inputs, since lower AIC_c values indicate better model performances. In between two subsequent AIC_c values the LMN input that is removed in the corresponding BE step is denoted. To simplify the determination of the inputs contained in the best subset, Fig. 5.1b is provided and shows which input subset corresponds to a specific number of LMN inputs. As can be seen, the best input subset contains four and three inputs in the $\underline{x}$- and $\underline{z}$-input space, respectively. The chosen LMN inputs belonging to the best subset are marked in bold type in Fig. 5.1b. Table 5.1 summarizes the assignment of each physical input variable to both the $\underline{x}$- and $\underline{z}$-input space. Interestingly, all



(a) Evaluation criterion curve



(b) Selected inputs

Figure 5.1: AIC$_c$ values versus number of LMN inputs (a) and selected inputs for the auto MPG data set

Table 5.1: Assignment of all physical inputs of the auto MPG example to the $\underline{x}$- and $\underline{z}$-input space for the best subset containing seven LMN inputs

$\underline{x}$-input space	$\underline{z}$-input space
u_1, u_3, u_4, u_7	u_2, u_5, u_6

physical inputs are incorporated in the best subset, but either exclusively in the $\underline{x}$-input space or the $\underline{z}$-input space. This implies that a classic wrapper input selection approach would have failed to reduce the number of inputs. Therefore, the auto MPG example benefits from the exploited possibility to distinguish between the $\underline{x}$- and $\underline{z}$-input space. A comparison of the model quality achieved with the best input subset opposed to a model with all inputs is omitted here, because it is included in the next section, see Fig. 5.2.

5.1.2 Regularization-Based Input Selection

The RBIS approaches with the heuristic as well as with the optimization-based determination of the split regularization parameters are tested on the auto MPG application. Findings from the separated $\underline{x}$-$\underline{z}$-input selection from the previous section are not used here. Therefore, all physical inputs u_1 up to u_7 are contained in both the $\underline{x}$- and $\underline{z}$-input space during the RBIS calculations. For the heuristic determination of the split regularization parameters the same four initial values are used as previously, which are 10^{-4}, 10^{-3}, 10^{-2}, and 10^{-1}, see Section 3.3.3. The optimization-based determination of the split regularization parameter has the leave-one-out (LOO) error as objective.

In Fig. 5.2 the model qualities of all RBIS approaches and additionally the model quality with the best subset according to the separated $\underline{x}$-$\underline{z}$-input selection from the previous section are compared. Additionally the achieved model qualities of the LOcal LInear MOdel Tree (LOLIMOT) and HILOMOT are visualized without the use of any input selection technique, i.e. all inputs are used for both the $\underline{x}$- and $\underline{z}$-input space. There is no decrease in the model quality through any of the input selection techniques. The best model quality is achieved by the model with only seven LMN inputs resulting from the separated $\underline{x}$-$\underline{z}$-input selection. The best RBIS result is obtained if the heuristic is used for an initial split regularization parameter of $\lambda = 10^{-3}$.

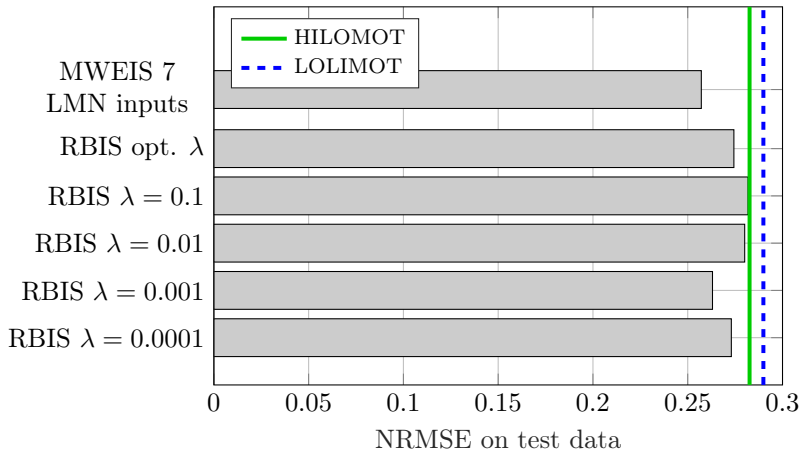


Figure 5.2: Comparison of the achieved model qualities with all RBIS approaches, the best input subset found with an separated $\underline{x}$-$\underline{z}$-input selection, as well as LOLIMOT and HILOMOT

In contrast to the investigated test processes in Section 3.3.3, the RBIS approach is able to increase the model accuracy more clearly. A likely cause is the higher dimensionality of the auto MPG application compared to the used test processes, where the maximum number of inputs is four instead of seven for the auto MPG application. It is assumed that through the increased input dimensionality the benefits of regularizing the splitting parameters become more pronounced. Additional investigations of the RBIS approach with processes having higher-dimensional input spaces are suggested for future research.

5.1.3 Visualization: Partial Dependence Plot

The partial dependence plots for all inputs of the model with the best quality are shown in Fig. 5.3. The model used for the visualization is obtained through the separated $\underline{x}$-$\underline{z}$-input selection and has four $\underline{x}$-inputs and three $\underline{z}$-inputs. The nonlinear dependency on the inputs contained in the $\underline{z}$-input space, which are the displacement, the acceleration, and the model year, is clearly visible from the mean curves. The mean curves of the remaining inputs indicate an affine behavior as could be expected from the fact that these inputs are only contained in the $\underline{x}$-input space.

All mean curves show a monotonic behavior. The fuel consumption generally increases

- with fewer cylinders,
- for higher values of the displacement,
- for higher values of the horsepower,
- for higher values of the weight,
- for higher acceleration values, and
- the earlier the car is built.

Additionally, the fuel consumption increases very slightly when the origin changes from Europe over Japan to the USA. Obviously an increasing fuel consumption for fewer cylinders is physically implausible. This strange effect if further elaborated in Section 5.1.4.

Besides the global influence of all inputs, sensitivities are observable through the slopes of the shown mean curves. The MPG values are most sensitive with respect

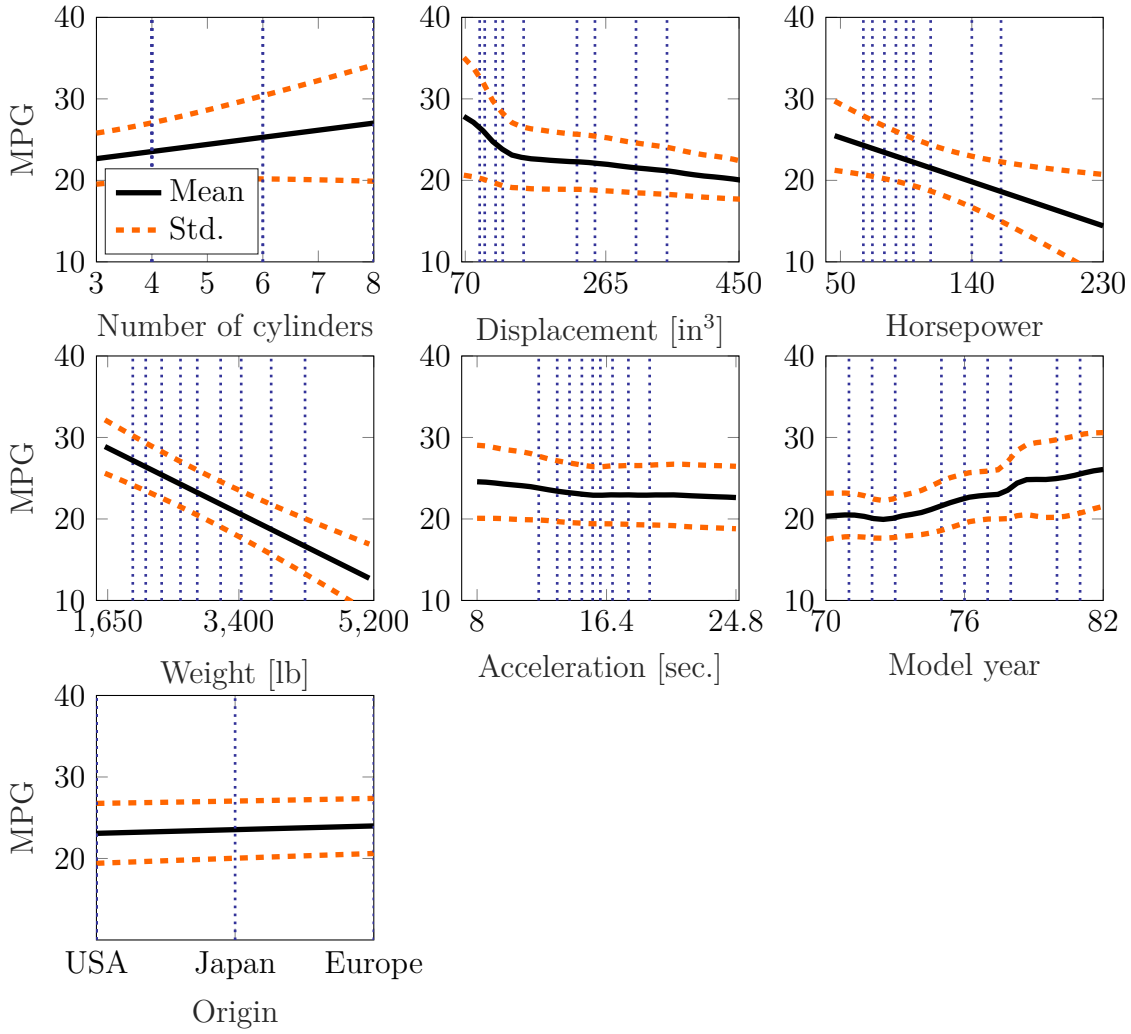


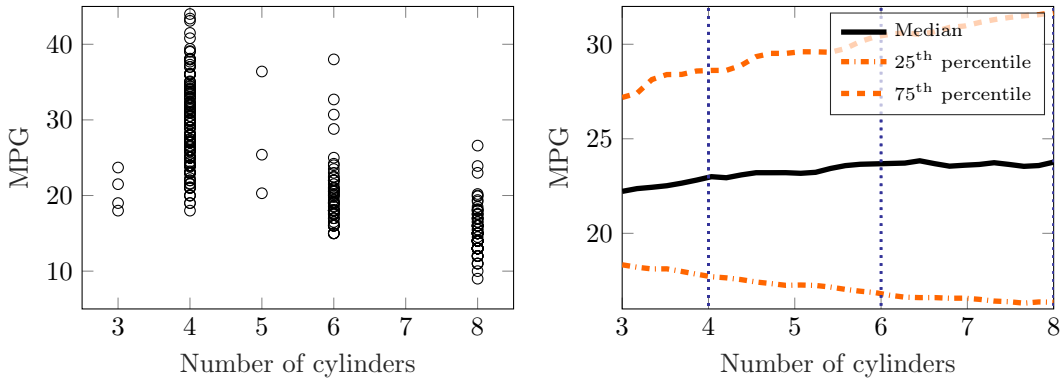
Figure 5.3: Partial dependence plots of the best auto MPG model for each physical input

to the weight and the horsepower. Additionally, the MPG values are sensitive to changes in the displacement but only for low displacement values.

5.1.4 Critical Assessment of Partial Dependence Plots

The partial dependence plot of the number of cylinders shown in Fig. 5.3 indicates lower MPG values meaning an increasing fuel consumption for fewer cylinders. This is obviously physically implausible and the reason for this strange effect is further investigated in this section.

At first the training data without the incorporation of any model is reviewed. In Fig. 5.4a, the MPG values are plotted against the number of cylinders. Although



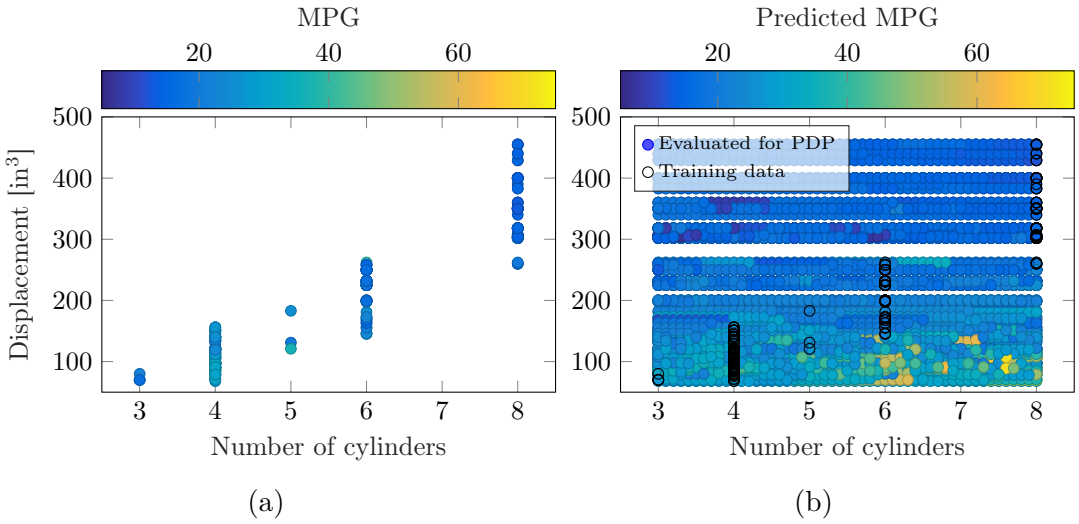
(a) Training data

(b) Partial dependence plot with median curve

Figure 5.4: (a) MPG versus number of cylinders. (b) Partial dependence plot for the number of cylinders with the median instead of the mean curve.

the MPG values for each number of cylinders vary widely, there is a trend towards increasing fuel consumption with more cylinders. Since the data does not provide a reason for the partial dependence plot of the number of cylinders, it can be assumed that the underlying model delivers implausible predictions. At this point it is not clear whether the model provides only a few, but huge, implausible predictions that can be regarded as a kind of outlier. In order to verify this, the way the partial dependence plot is created is changed. As explained in Section 3.5, the *average* effects of all training data samples are taken into account in order to represent the effect of one input variable on the model. Instead of taking the average effects of all training data into account, the *median* effects are shown in Fig. 5.4b, since the median is known to be statistically more robust against outliers. Because the same implausible trend of the fuel consumption still maintains, there have to be more unreliable model predictions than just a few outliers.

Finally, the reason for the implausible model predictions could be found and is illustrated in Fig. 5.5. Naturally, the number of cylinders and the displacement are highly correlated as can be seen in Fig. 5.5a. This leads to regions in the input space in which no information is available for the model. In particular, no data and therefore no information is available for high displacements and few cylinders and vice versa. If the model is forced to make predictions in these regions of the input space, it can be considered as extrapolation with a high danger of unreliable predictions. This is exactly what happens. For the creation of the partial dependence plot the model is indeed forced to extrapolate as can be seen in Fig. 5.5b. It can be observed that in the region of low displacements and many cylinders the model



(a) (b)

Figure 5.5: (a) Displacement versus number of cylinders together with the training data and the MPG values therein. (b) Displacement versus number of cylinders together with points that have to be evaluated for the partial dependence plot (PDP) creation together with predicted MPG values.

predicts a very low fuel consumption. The training data set does not contain such small consumption values, see Fig. 5.5a. In conclusion, extreme care is advised in the interpretation of partial dependence plots, in particular if highly relevant inputs are strongly correlated which provokes extreme extrapolation in the partial dependence plots.

5.2 Air-Mass Flow Prediction

In modern combustion engines the engine control unit (ECU) is responsible for managing the engine performance. It is used to control e.g. the air-fuel ratio, the idle speed, and the variable valve timing [61]. Therefore, the ECU has to model the air-mass flow (AMF) depending on the engine's operating point. Parameters needed to calculate the AMF are traditionally stored in look-up tables with supporting points upon a grid. This runs into difficulties due to the required storage, which grows exponentially with higher input dimensions. In order to overcome the exponential increase of required storage, Bänfer et. al. proposed, investigated, and realized the substitution of the look-up tables with LMNs, see [98, 5] for details. The data set used in this section originates from the cooperation of the University of Siegen with Continental Automotive GmbH, in which the substitution of look-up tables by LMNs has been developed.

The AMF into the combustion chamber of a gasoline engine should be predicted in order to determine the optimal amount of fuel that has to be injected for a complete combustion. The AMF y depends on the following six inputs:

- Engine speed u_1,
- valve timing for the intake camshaft u_2,
- valve timing for the exhaust camshaft u_3,
- position of the swirl flap u_4,
- position of the variable intake manifold (VIM) flap u_5, and
- intake manifold air pressure (MAP) u_6.

A schematic sketch of the technical system is shown in Fig. 5.6.

It is known that the system can be adequately described with scheduled affine models [61]. These affine models depend explicitly on the MAP u_6 while the dependency on the remaining inputs is implicitly contained in operating-point-dependent slopes m_a and offsets b. In particular, the slope m_a and offset b depend only on the inputs $u_1, u_2, \ldots, u_5$:

$$\hat{y}(u_1, u_2, u_3, u_4, u_5, u_6) = m_a(u_1, u_2, u_3, u_4, u_5) \cdot u_6 + b(u_1, u_2, u_3, u_4, u_5)\,. \tag{5.1}$$

The AMF prediction has traditionally been modeled by grid-based look-up tables. Recently it has been described by LMNs which is shown in Fig. 5.7. As can be seen, the position of the swirl flap u_4 and the position of the VIM flap u_5 are treated in a special way. Both of these inputs are binary ones. As a result there are four

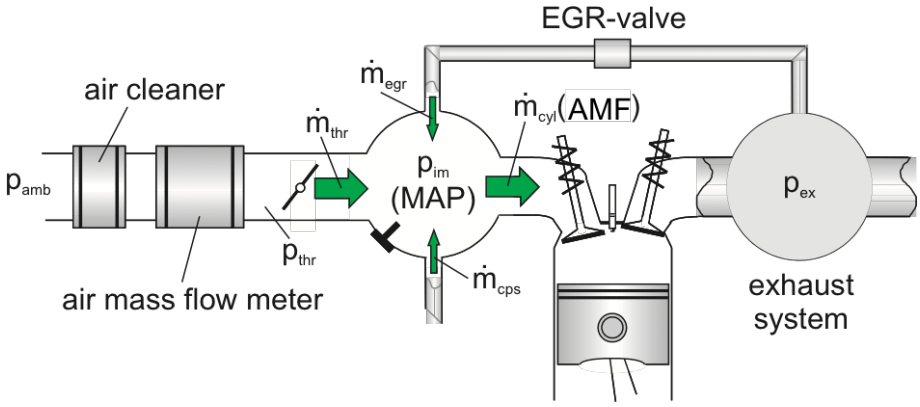


Figure 5.6: Schematic sketch of the intake manifold of a combustion engine

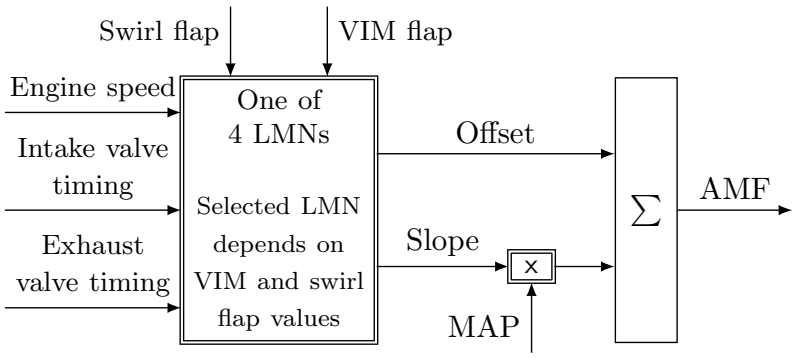


Figure 5.7: Schematic sketch of the „traditional“ model structure for the AMF prediction. Depending on the positions of the VIM and swirl flap, one of four LMNs is used to determine the offset and the slope for an affine model depending on the intake MAP.

possible combinations of u_4 and u_5 values. For each combination an individual LMN is trained with three inputs, namely the engine speed u_1 as well as the valve timings for the intake and exhaust camshaft u_2 and u_3, respectively. The outputs of each LMN are the slope m_a and the offset b, which are used to finally calculate the AMF under consideration of the MAP.

All of the previously described prior knowledge about the modeling of the AMF is not exploited for the investigations in this section. Instead, it should be analyzed if a separated $\underline{x}$-$\underline{z}$-input selection is able to reveal the model structure known from expert knowledge solely based on data. For the separated $\underline{x}$-$\underline{z}$-input selection all physical inputs u_1 to u_6 are treated equally and are used as inputs for one LMN. Additionally, the partitioning of the resulting LMN is analyzed following the embedded input selection approach described in Section 3.4.1.

For the investigations, one data set is available with a total number of $N = 3045$ samples. This data set is split into one training data set consisting of $N_{train} = 2589$ samples and one test data set consisting of $N_{test} = 456$ samples. For the splitting of the data set the deterministic and distance-based algorithm described in Appendix A is used with the goal to avoid extrapolation.

5.2.1 Mixed Wrapper-Embedded Input Selection

The separated $\underline{x}$-$\underline{z}$-input selection is used with BE as search strategy and the $\mathrm{AIC_c}$ as evaluation criterion. Note that due to the used input selection scheme the number of LMN inputs to choose from is twelve. Only local affine models are used for

LMNs that are trained with HILOMOT. The model complexity of the LMNs is also determined with the AIC_c.

Figure 5.8 shows errors on training and test data versus the number of LMN inputs. In between two subsequent error values the input is denoted that is discarded in the corresponding BE step. Besides the information about the input number, the input space from which the input is removed is denoted, i.e. x or z. The minimum normalized root mean squared error (NRMSE) value is obtained with eleven LMN inputs. If the number of LMN inputs is decreased to six, the NRMSE value does not change significantly. Therefore, having six LMN inputs can be seen as a good compromise between the model quality and the simplicity of the model.

The model structure identified solely based on the available data for six LMN inputs is shown in Fig. 5.9. It matches the „traditional“ model structure which is based on expert knowledge, see Fig. 5.7. In other words, the six most important inputs found with the separated $\underline{x}$-$\underline{z}$-input selection are in accordance with expert knowledge and are assigned to the correct input space. Note that the likelihood of obtaining this result just by chance is 1/924, because the number of possible combinations of six inputs out of twelve is $\binom{12}{6} = 924$.

Finally, the model quality of the LMN with six inputs resulting from the separated $\underline{x}$-$\underline{z}$-input selection is compared to the existing model structure. The main difference of both models is the handling of the binary inputs u_4 (swirl flap) and u_5 (VIM flap). In the existing model structure, these inputs are used to decide which of four distinct LMNs is used. In contrast to that, the model structure identified with the

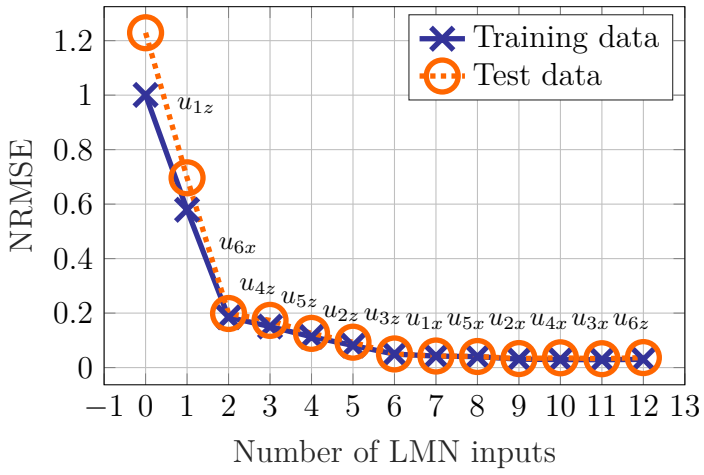


Figure 5.8: NRMSE values on training and test data versus the number of LMN inputs for the AMF prediction

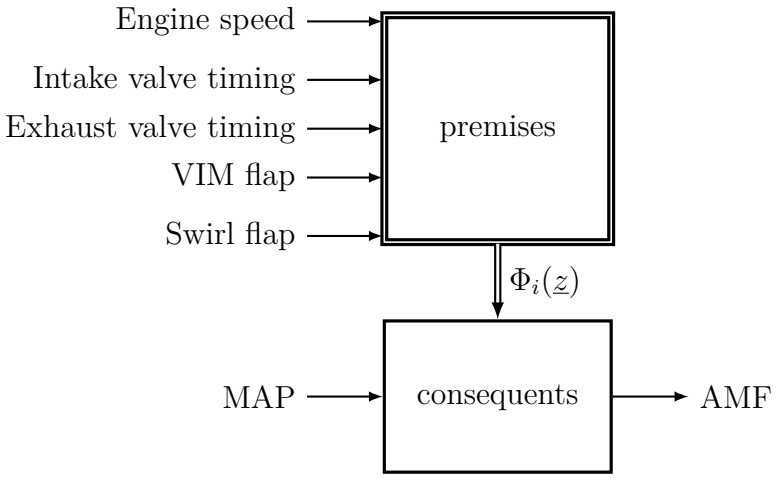


Figure 5.9: Visualization of the model structure identified with the separated $\underline{x}$-$\underline{z}$-input selection

separated $\underline{x}$-$\underline{z}$-input selection uses only one LMN with u_4 and u_5 only incorporated in the $\underline{z}$-input space. It turns out that the model quality of the existing model structure is slightly better in terms of the achieved NRMSE value on test data, namely 0.03 opposed to 0.05.

In summary, the confidence in the separated $\underline{x}$-$\underline{z}$-input selection is increased through the AMF application example. Even though the model structure identified solely based on the available data is worse compared to the traditional structure in terms of the achieved prediction quality, it is very close to it. Additionally, the six most important inputs are revealed and are assigned correctly to the respective input space.

5.2.2 Partition Analysis

In order to rank all inputs according to their nonlinear influence, an LMN with local affine models is trained and subsequently the resulting input space partitioning is analyzed as described in Section 3.4.1. HILOMOT is used to train the LMN. The model complexity is determined with the $\mathrm{AIC_c}$. All physical inputs are used as inputs for the $\underline{x}$- and $\underline{z}$-input space.

The resulting relevance factors are shown in Fig. 5.10. The engine speed has the highest relevance factor while the swirl flap has the lowest relevance factor. Surprisingly, the relevance factor of the intake MAP is not the lowest even though it is the first $\underline{z}$-input that is removed in the separated $\underline{x}$-$\underline{z}$-input selection, see Fig. 5.8.

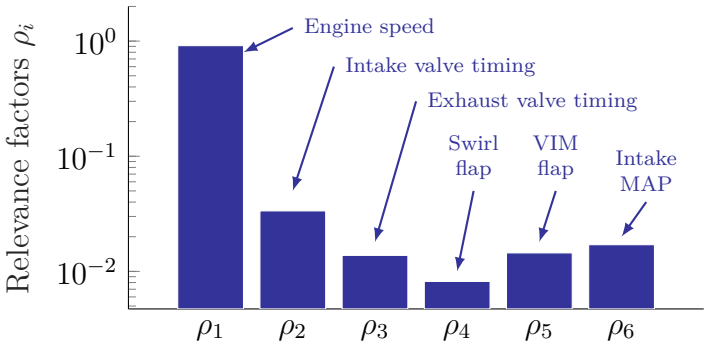


Figure 5.10: Relevance factors ρ_i for the LMN that predicts the AMF

Compared to the relevance factor of the engine speed, all other relevance factors are almost zero and differences are only visible due to the logarithmic scaling of the y-axis. It seems that due to the small differences in the relevance factors the ranking of inputs u_2 to u_6 is not trustworthy. Based on the obtained relevance factors, only the engine speed seems to be relevant for the $\underline{z}$-input space. From prior knowledge and from the separated $\underline{x}$-$\underline{z}$-input selection this can not be confirmed. However, the results of the separated $\underline{x}$-$\underline{z}$-input selection indicate that having the engine speed (u_1) in the $\underline{z}$-input space and the intake MAP (u_6) in the $\underline{x}$-input space already leads to quite a good model, see Fig. 5.8.

In summary, the partition analysis fails to fully convince. Even though the partition analysis is able to reveal the most relevant $\underline{z}$-input, which is the engine speed, the ranking of the remaining inputs seems to be completely random. In defense for the partition analysis, it can be said that the differences between the relevance factors $\rho_2, \rho_3, \rho_4, \rho_5$, and ρ_6 are too small to draw conclusions about their relative ranking. However, it can not be recommended to rely on the partition analysis results in general. Performing an additional mixed wrapper-embedded input selection seems to be mandatory in order to find good input subsets for the $\underline{z}$-input space.

5.3 Fan Metamodeling

In this section the generation of CFD metamodels serves as example to prove the usefulness and applicability of some contributions of this thesis. In particular, advantages obtained through the chosen order of experimentation, the goal-oriented active learning strategy with LMNs, and the mixed wrapper-embedded input selection are pointed out. Two different types of fans are considered, which are centrifugal and

axial fans. Most of the information contained in this section about the axial fans is obtained from [3]. Information regarding the centrifugal fans originates from [4], [8], and [9].

The metamodels should capture the behavior of either centrifugal or axial fans. The general purpose of fans is to generate a gaseous fluid flow under build-up of pressure. Typically, the design of a new fan comprises two main targets. Firstly, the design point (i.e. the desired flow rate $\dot{V}$ and pressure rise Δp) must be fulfilled. Secondly, the shaft power P_{shaft} shall be as low as possible. The achievability of the first design target mainly depends on the choice of the outer fan diameter D and the rotational speed n_R. Cordier [33] found that the specific fan diameter

$$\delta = \frac{D}{\left(\frac{8}{\pi^2}\right)^{1/4}\left(\frac{\Delta p}{\rho_f}\right)^{-1/4}\dot{V}^{1/2}} \tag{5.2}$$

and the specific fan speed

$$\sigma = \frac{n_R}{(2\pi^2)^{-1/4}\left(\frac{\Delta p}{\rho_f}\right)^{3/4}\dot{V}^{-1/2}} \tag{5.3}$$

of all built fans and pumps lie in a narrow band around the curve depicted in Fig. 5.11 which were later known as the Cordier curve and the Cordier diagram, respectively. The original Cordier diagram is based on fan performance data stemming from the 1950s, but its validity was confirmed in numerous more recent studies, see e.g. the work by Willinger et al. [140, 141, 142]. Figure 5.11 furthermore indicates the typical realm of centrifugal and axial fans. The rest of the Cordier-band is associated with other fan types such as propellers or mixed-flow. The second design target (the minimization of P_{shaft}) is equivalent to the maximization of the aerodynamic efficiency defined as

$$\eta = \frac{\dot{V} \cdot \Delta p}{P_{shaft}}\,. \tag{5.4}$$

Independent of the fan type, i.e. centrifugal or axial, only the impeller as the key component with respect to aerodynamic efficiency is investigated.

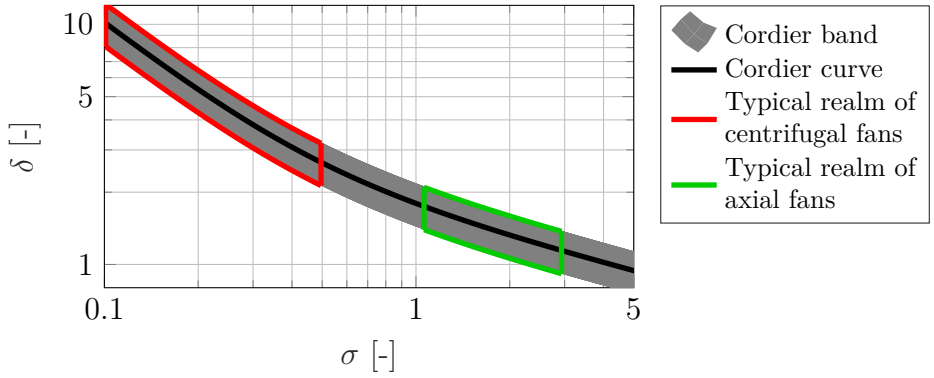


Figure 5.11: Cordier diagram with indication of the typical realm of centrifugal and axial fans

Centrifugal Impeller Geometry

The centrifugal impeller geometry is described by nine geometrical parameters including, among others, the number of blades, the inner diameter, the inlet width, the outlet width, the inlet blade angle, and the outlet blade angle. Those parameters are supposed to be most relevant in order to adapt the impeller geometry to a large variety of potential design points [17]. Figure 5.12 shows a technical drawing in which almost all geometrical design parameters are explained. Note that the centrifugal fan metamodels have ten inputs, including the geometrical parameters as well as the flow rate.

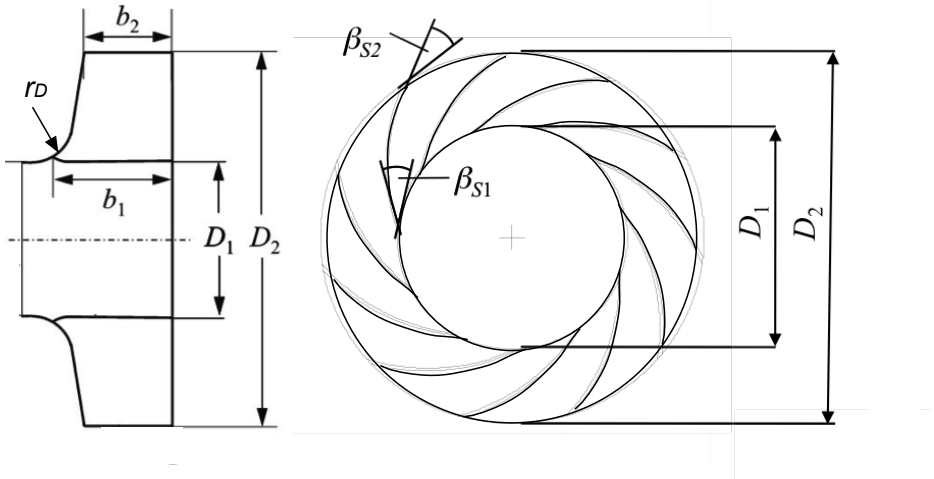


Figure 5.12: Impeller design parameters of the centrifugal impeller

Axial Impeller Geometry

The axial impeller geometry is described by 26 geometrical parameters including, among others, the number of blades, the hub-to-tip ratio, the chord length, the angle of incidence, and the sweep angle. Figure 5.13 shows two technical drawings in which some of the geometrical parameters are depicted. Note that the axial fan metamodels have 28 inputs, including the geometrical parameters as well as the flow rate and the target flow rate.

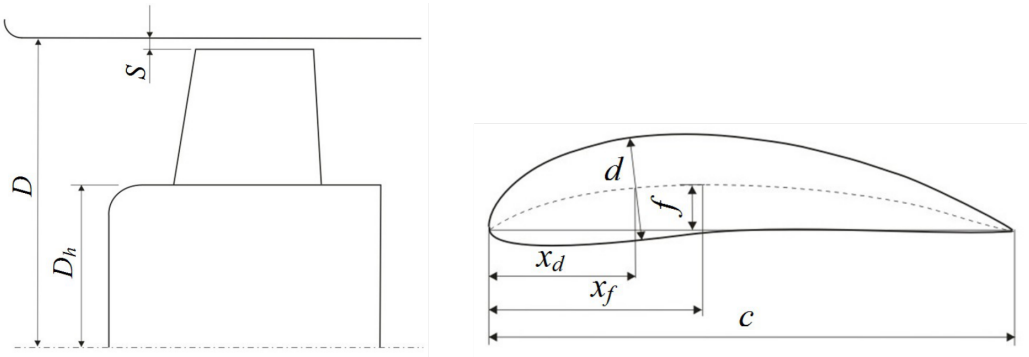


Figure 5.13: Some impeller design parameters of the axial impeller

Why Metamodels?

The generated CFD metamodels approximate, among other flow field quantities, the efficiency and the pressure rise of fans depending on geometric parameters of the impeller. The CFD metamodels are used to aerodynamically optimize the design parameters of the impeller with the evolutionary algorithm described in [3]. In principle, CFD simulations could directly be coupled with optimization algorithms to optimize impeller designs. However, this would lead to a very high computational expense required for the optimization of each impeller design. Both ways to optimize the impeller geometry are depicted in Fig. 5.14. By using a metamodel, the computational demand that is necessary for the numerical CFD simulations reduces to the generation of data, which is needed for the metamodel training as shown in Fig. 2.11. Once the metamodel is trained no further CFD simulations are necessary. The time needed to compute the efficiency and the pressure rise for one impeller geometry at one specific volume flow could be reduced from approximately 30 minutes to 0.02 seconds by substituting the CFD simulation with the metamodel. More information about the centrifugal impeller parameterization, the CFD model, and

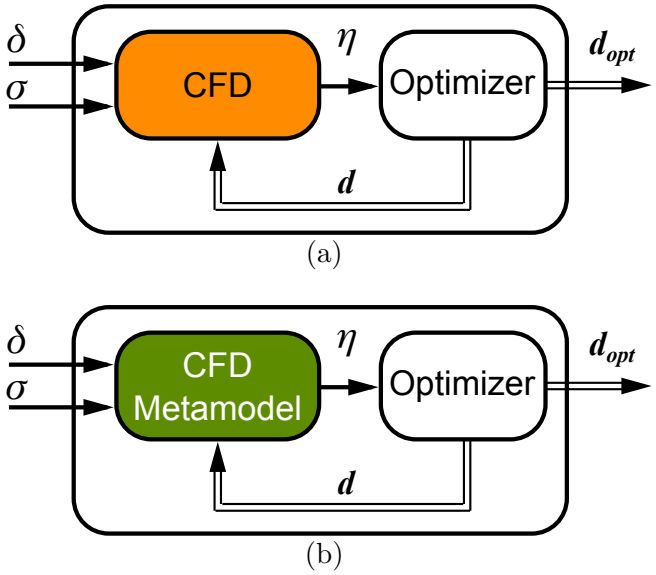


Figure 5.14: Impeller geometry optimization with the help of CFD simulations (a) and a CFD metamodel (b)

the optimization of the CFD grid resolution can be found in [4]. For the same information regarding the axial impeller the reader is referred to [3]. A more detailed discussion about fans in general can be found in [16]. The fundamentals of fluid dynamics are covered in [92].

Design of Experiments – Centrifugal Fan Metamodel

The training data acquisition for the metamodel generation can be divided into two phases. The first one is a passive learning phase in which CFD simulations are carried out according to a predefined maximin optimized Latin Hypercube (LH) design. In this phase the intelligent k-means sequence (IKMS) method is used to determine in which order the CFD simulations should be carried out that are contained in the design of experiments. Section 5.3.1 describes the effect of all order determination methods on the metamodel quality. The second phase follows an active learning scheme in which information about already simulated impeller designs is exploited to determine further CFD simulation queries. The goal-oriented active learning strategy with LMNs is used for the second phase. Section 5.3.2 reviews the influence on the model quality and additional improvements of the resulting metamodels. Roughly, 900 and 3000 different impeller geometries are CFD simulated in phase one and two, respectively.

By using this two phase data acquisition approach, it is assured that the whole design space is explored in phase one such that metamodel predictions are reliable for a variety of different impeller geometries. Once the metamodel quality is saturated or considered to be sufficiently reliable, information from the metamodel is used to plan additional measurements. By exploiting information of the metamodel these adaptively planned measurements aim to improve the metamodel in areas of the design space where the metamodel's performance is considered to be worst while simultaneously concentrating on near-optimum regions and geometrical designs that differ as much as possible from already measured geometrical designs. Since the quality of the metamodel was saturated after the CFD simulation of approximately 900 different impeller geometries in phase one, phase two was started and used for the rest of the project duration.

Design of Experiments – Axial Fan Metamodel

The training data acquisition for the metamodel generation consists only of a passive learning phase. It dates back to times where the DoE know-how was less developed. At first, 2000 different geometrical designs are generated with the help of a Sobol sequence [125] and CFD simulated. Subsequently, 1000 random candidate points are generated from which the 10 % are chosen that are farthest away from the already existing points. The second step is repeated several times until the DoE contains approximately 12,000 different geometrical designs.

5.3.1 Order of Experimentation

Although the CFD simulations have been carried out according to the IKMS method during the first data acquisition phase, this section tries to compare all order of experimentation methods except for the median distance sequence (MDS) method. Because the MDS method's results on the artificial test processes are the worst ones, it is completely neglected here. In addition, HILOMOT for design of experiments (HilomotDoE) and a simple randomization are used to determine an ordering of all training data samples as references. For an explanation of how HilomotDoE is used to determine the order of experimentation, see Section 4.1.4. However, all calculations are done in retrospect, after all data for the CFD metamodel generation has been completely collected. Metamodels for both relevant outputs, i.e. the specific pressure rise ψ and the efficiency η, are considered.

In order to compare all order determination strategies on a fixed data set, all available data is randomly split into 85 % for training and 15 % for testing purposes. This is done 50 times, such that there are 50 different training and test data sets. For each of this data splittings, the order of experimentation for the training data is determined with each order determination strategy, i.e. biggest gap sequence (BGS), IKMS, HilomotDoE, and the simple randomization. Once each order determination strategy has determined the ordering for each of the 50 training data sets, LMNs are trained with fractions of these sorted training data sets. At first an LMN with the first 10 % of the training data is trained, then with the first 20 % of the training data, and so on until all training data is used, as visualized in Fig. 5.15. HILOMOT is used as training algorithm to generate the LMNs and with the help of the test data the model quality is assessed. As a result, one curve of the model quality versus the amount of used training data arises for the BGS, IKMS, and HilomotDoE strategy and each split of the whole data set. In case of the randomized ordering, ten different random orderings are determined for each of the 50 training data sets, such that there are 500 of these curves. Figure 5.16a and 5.16b show the mean values of these curves for the prediction of ψ (specific pressure rise) and η (efficiency) in case of the BGS, IKMS, and HilomotDoE strategies. In case of the randomized orderings only the best and worst curve (out of all 500) are shown. An order determination strategy is considered to be good if the resulting model quality quickly improves. This means a good model quality is reached with a rather low training data set size. Comparing the two model-free algorithms, IKMS and BGS perform equally well and on a similar level as the active learning strategy, which is abbreviated with HDoE. Showing the best and worst case of the random orderings should provide a feeling how good or bad the strategies perform on an absolute scale. Note that the error of the worst

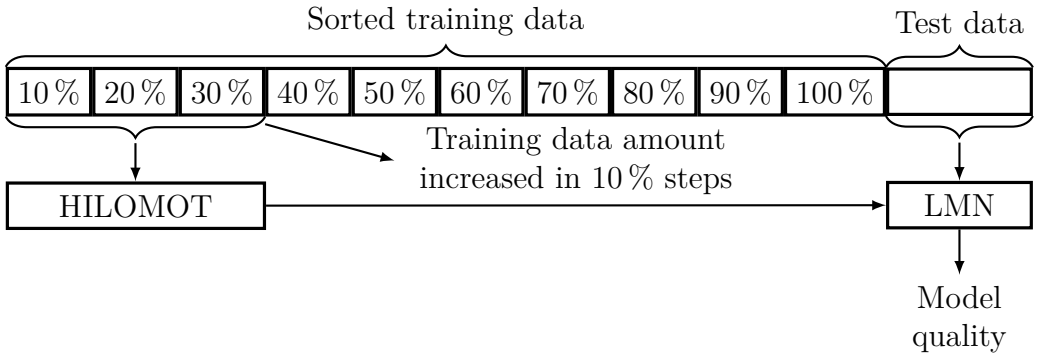
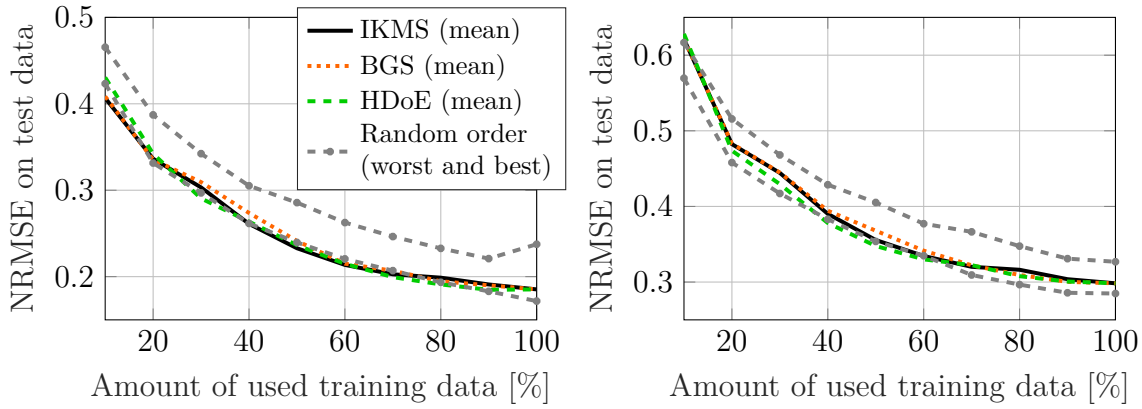


Figure 5.15: Procedure to assess the model quality for 10 % increments of the sorted training data. Here, only the model quality assessment for 30 % of the training data is visualized.



(a) ψ prediction (b) η prediction

Figure 5.16: Comparison of different order determination strategies for the prediction of ψ and η

and best random order do not coincide with the error of the other determination strategies at 100 % of the used training data. This is due to the fact, that each of the two shown random sequence curves is generated with only one of the 50 resulting training data sets while the other curves represent mean values. The IKMS, BGS, and HDoE curves are very close to the best case of all random sequences for all amounts of used training data.

The advantage of using a specific order of experimentation is pointed out for the real-world example of generating a data base used to build a CFD metamodel. The achieved model qualities with the two model-free approaches, namely the BGS and the IKMS, are even comparable to an active learning strategy (HilomotDoE) as can be seen in Fig. 5.16. The reason for the comparable model qualities might be the rather high-dimensional input space ($p = 9$) while the overall amount of training data even at the end of the first data acquisition phase is rather low ($N \approx 900$). In other words, HilomotDoE might still be exploring the whole input space instead of concentrating on highly nonlinear regions in it. Another disadvantage for HilomotDoE here is, that the candidate points are restricted to the data points contained in the existing data set and can not be chosen arbitrarily in the input space.

5.3.2 Goal-Oriented Active Learning

In case of the active learning for the CFD metamodel the goal-oriented procedure illustrated in Fig. 4.17 is applied with minor adjustments to meet problem specific needs. A maximin LH design for an 11-dimensional input space is generated with

$N_{LH} = 586$ samples. Two of the eleven inputs specify a design point defined by the desired flow rate and pressure rise. These two inputs always belong to the set of constrained optimization variables, since the efficiency is maximized for these design points. The remaining $p = 9$ inputs correspond to the geometric parameters that should be optimized. It follows that there are $n_c = 2^p = 512$ possible optimization constraints for each point contained in the LH design. As explained in Section 4.3.2, all possible combinations of constrained optimization variables are applied to each point contained in the LH design, resulting in $N_c = 512 \cdot 586 = 300\,032$ candidate points after all optimization runs are finished.

Here, in each loop of the active learning strategy a list of 500 query points is determined. Each query point is intended to be CFD-simulated. As soon as new CFD simulations are finished and the data set is updated with new information, the process of updating the list of query points starts again. The number of demanded queries is chosen quite high to prevent the current query list from becoming empty, i.e. all queries have been CFD-simulated, before the new query list is readily determined. Until the update of the query point list is finished, the „old“ list is further used. Once the optimization of all candidate points is accomplished and all new 500 queries are calculated, the new list is used.

The influence of the proposed goal-oriented active learning strategy is evaluated based on three aspects. First, the model quality is assessed for increasing portions of training data. Second, the extension of achievable design points in the Cordier diagram is shown. And third, the improvements of the achievable total-to-static efficiencies are visualized for all achievable points in the Cordier diagram.

For the model quality assessment 545 data points are chosen from the CFD data basis according to the data splitting procedure explained in Appendix A. All 545 chosen data points cover the area of possible design points spanned by the desired flow rate and pressure rise uniformly. These designs together with the achieved total-to-static efficiencies serve as test data for the models generated with different amounts of training data. The target value of the model is the total-to-static efficiency η_{ts}. Figure 5.17 shows the curve for the model quality versus the training data amount. The dashed line marks the point, where the passive learning phase has ended and the active learning strategy has started. It is observable that the model quality at the end of the passive learning phase started to saturate. A likely cause for the saturation is that the global characteristics of the centrifugal fans are already identified with the 900 different impeller designs. No new information seems to be

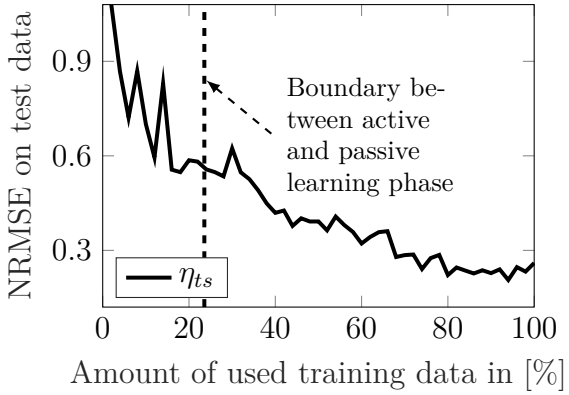


Figure 5.17: NRMSE on test data originating from CFD-based optimization runs vs. amount of training data

added by further exploring the design space uniformly since the model quality does not improve. Hence, a more goal-orientated DoE strategy is needed that exploits already available information in form of the metamodel to find new queries (impeller designs) that contain new information. Note that the test data used to create the shown curve here differs from the test data used in the previous section. Therefore the errors from Fig. 5.17 are not comparable to the ones shown in Fig. 5.16.

Figure 5.18 shows the extension of achievable regions in the Cordier diagram through the proposed goal-oriented active learning strategy. For almost each viable specific fan speed σ higher specific fan diameters δ are possible. Additionally, lower and higher specific fan speeds could be achieved through the active learning phase compared to the passive one. The impact of the extension of achievable points cannot

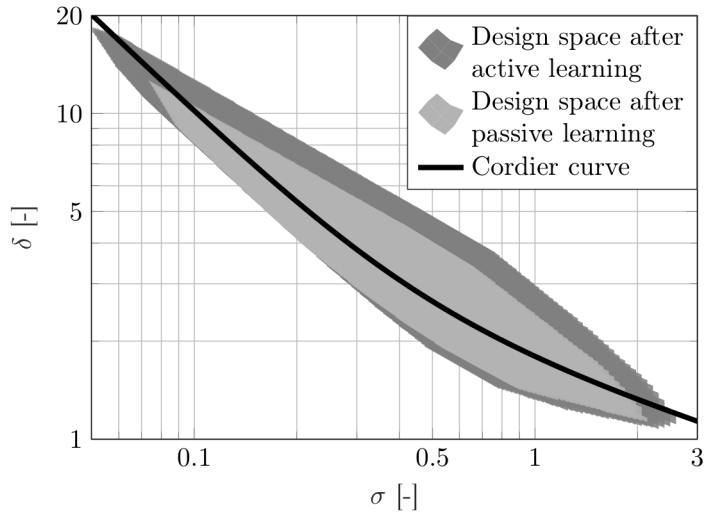


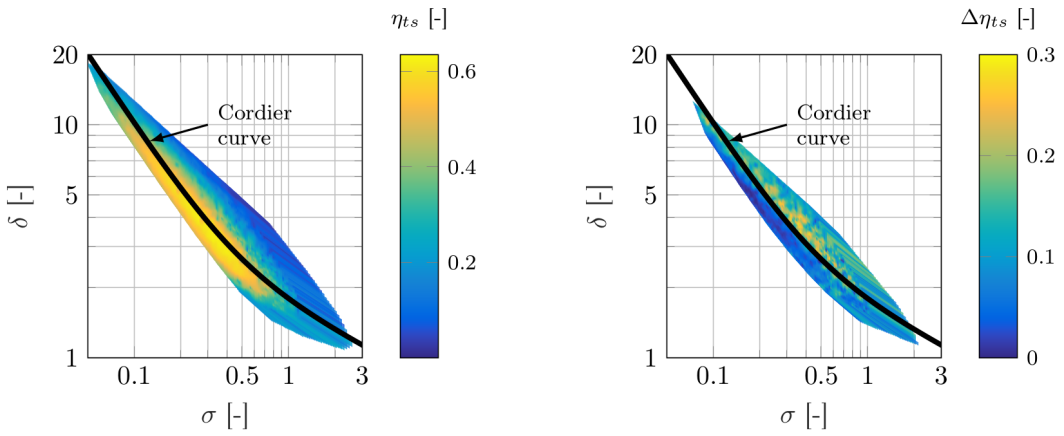
Figure 5.18: Extension of achievable design point area in the Cordier diagram

be evaluated without taking into account the corresponding efficiencies. Therefore, the discussion of the impact is postponed to the end of this section.

The absolute total-to-static efficiencies in the area of possible impeller designs after the active learning strategy and the corresponding improvements are shown in Fig. 5.19a and 5.19b, respectively. The efficiency could be improved at some points from $\eta_{ts} \approx 0.3$ (before the active learning took place) to $\eta_{ts} \approx 0.6$. The improvements are mostly achieved above the Cordier curve, where the specific fan diameter δ is relatively high. The efficiency improvements through the active learning can only be shown in areas, where data from the passive learning phase is available because a reference value exists only there, compare Fig. 5.18. That is the reason, why the area covered by possible designs is smaller in Fig. 5.19b compared to Fig. 5.19a.

The extension of achievable regions in the Cordier diagram together with the absolute efficiencies shown in Fig. 5.19a indicate that designs in the extended regions are not attractive to be realized in practice. The achieved efficiencies in the extended regions are rather low and therefore not desirable. However, the efficiency improvements close to the Cordier curve are quite high and demonstrate the usefulness of the active learning strategy for one real-world application.

Through an extension of the already proposed HilomotDoE [54] algorithm, a goal-orientation is introduced, see Section 4.3. There are three goals: (I) The concentration on possibly optimal geometries, (II) the focus on areas in the input space with inferior generalization performance and (III) a high diversity of training data samples. In the application example shown here, all three goals are met. (I) The



(a) Absolute efficiencies

(b) Efficiency improvements

Figure 5.19: Absolute total-to-static efficiencies after the active learning strategy (a) and improvements through the active learning strategy (b)

maximum efficiencies could be improved as shown in Fig. 5.19, (II) the overall model performance could be increased as can be seen in Fig. 5.17, and (III) the diversity increase of the training data samples leads to an extension of achievable design points, see Fig. 5.18.

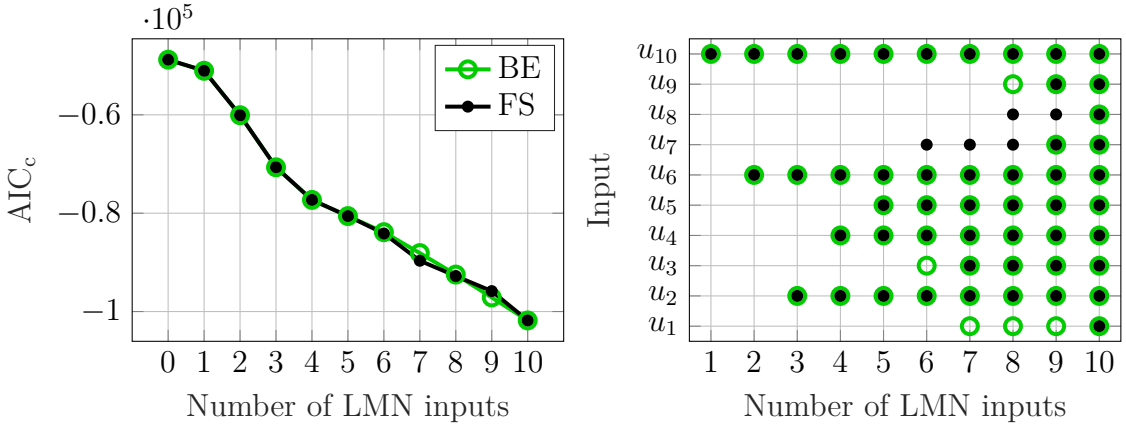
5.3.3 Mixed Wrapper-Embedded Input Selection

The mixed wrapper-embedded input selection is applied to both the centrifugal and the axial fan metamodel. In this section the linked $\underline{x}$-$\underline{z}$-input selection is used with the $\mathrm{AIC_c}$ as evaluation criterion. Two search strategies are employed and compared, which are BE and forward selection (FS). For both input selection approaches 80 % of all available data is used while 20 % of it is saved to subsequently assess the model quality for specific input subsets on fresh data. For data splitting the deterministic and distance-based algorithm explained in detail in Appendix A is used.

Centrifugal Fan Metamodel

Figure 5.20 shows the results of the linked $\underline{x}$-$\underline{z}$-input selection for the centrifugal fan metamodel. Regardless of the used search strategy, it is observable that no better bias/variance tradeoff could be found by removing inputs, see Fig. 5.20a. This means that each of the inputs carries enough information to overcompensate the additionally introduced variance. It can be concluded that (I) only physically meaningful inputs have been chosen to describe the behavior of the centrifugal fans and (II) the experimental design ensures a meaningful variation of these inputs. This outcome is reasonable and was expected since experts in the field of fans, including several experts from established companies, discussed and finally agreed on which inputs to use for the metamodel.

For this application, there is no noticeable difference between the two used search strategies. From one up to five LMN inputs the subsets chosen by BE and FS are in fact identical, see Fig. 5.20b. But even with different subsets of inputs for the higher-dimensional cases, the model accuracies are comparable. Table 5.2 presents errors on the test data set for several input subsets, including six, eight, and ten LMN inputs. Of course, the error for the model incorporating all available inputs is identical. With six LMN inputs the model based on the subset selected by BE



(a) AIC_c vs. number of LMN inputs (b) Selected inputs

Figure 5.20: Obtained results of the linked $\underline{x}$-$\underline{z}$-input selection for the metamodel of centrifugal fans

performs insignificantly better. In case of eight LMN inputs the model based on the subset selected by FS is slightly better.

According to commonly used impeller design methodologies, as described e.g. in [22], the most important geometrical parameters are those that directly affect the guidance of the gaseous fluid flow and how it enters and exits each flow channel. Therefore, the most important geometrical parameters for the CFD metamodel should be:

- The inner diameter u_2,
- the inlet blade angle u_3,
- the inlet width u_5,
- the outlet width u_6, and
- the outlet blade angle u_4.

Table 5.2: NRMSE values for models based on different LMN input subsets evaluated on the test data set of the centrifugal fan metamodel

Number of LMN inputs	6	8	10
Selected BE subset	**0.4598**	0.3773	0.3041
Selected FS subset	0.4622	**0.3691**	0.3041

Note that the order of the list above does not reflect the importance of the individual parameters, but rather the path from the gaseous fluid flow from the impeller inlet to the outlet. Since the rotational speed for all CFD simulations was fixed, the inner diameter together with the inlet blade angle determine the flow angle with which the gaseous fluid enters each flow channel. The inlet and outlet width have a big impact on how the cross-section area, through which the gaseous fluid flows, changes from the inlet to the outlet of the impeller. The change in the cross-section area has a significant influence on the deceleration of the gaseous fluid [36] which in turn affects the losses of the impeller and therefore the efficiency [22]. The outlet blade angle directly influences the deflection of the gaseous fluid flow at the impeller outlet.

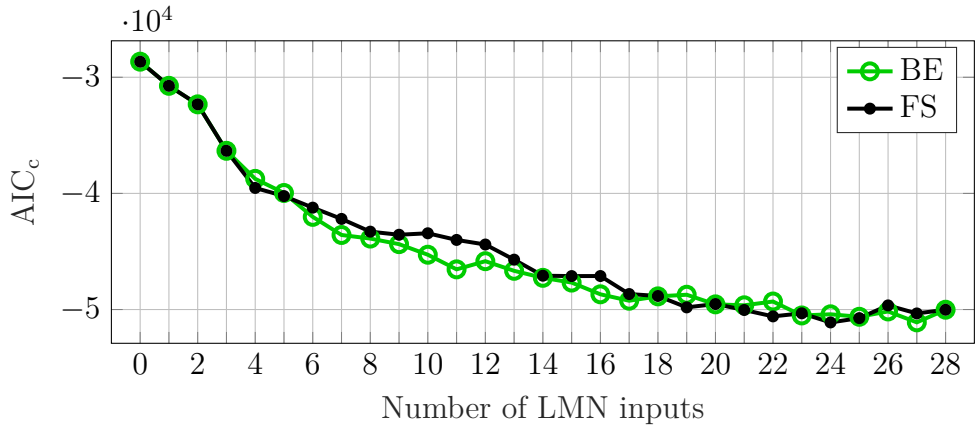
The five most important geometrical parameters according to the linked $\underline{x}$-$\underline{z}$-input selection with BE as search strategy are in perfect accordance with expert knowledge, i.e. all five parameters from the above list are selected, see Fig. 5.20b. The only metamodel input that is considered even more important is the volume flow rate u_{10}, which is also quite reasonable because it specifies the operating point of the centrifugal fan. When using the FS search strategy, the shroud radius u_7 is considered to be more important than the inlet blade angle u_3 which seems not reasonable according to expert knowledge. However, no significant influence on the model quality can be observed in Fig. 5.20a.

Axial Fan Metamodel

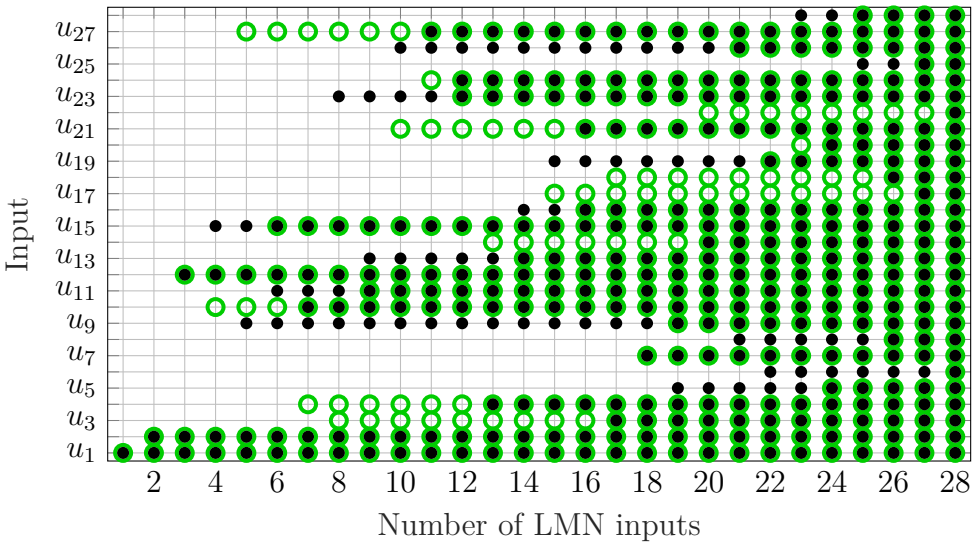
Figure 5.21 shows the results of the linked $\underline{x}$-$\underline{z}$-input selection for the axial fan metamodel. Both search strategies deliver for most input subset sizes comparable $\mathrm{AIC_c}$ values. From 20 up to 28 LMN inputs no more significant improvements are observable. The best achieved $\mathrm{AIC_c}$ values are obtained with 27 and 24 LMN inputs for BE and FS, respectively. The biggest difference in the achieved $\mathrm{AIC_c}$ values appears at eleven LMN inputs.

Both used search strategies lead to identical input subsets from a subset size of one up to three, see Fig. 5.21b. According to the achieved $\mathrm{AIC_c}$ values, both search strategies deliver comparable results for almost all subset sizes. Selected input subsets are compared regarding the error on the test data set in Table 5.3. The compared subsets include:

- 11 LMN inputs because the biggest difference in the achieved $\mathrm{AIC_c}$ value of both search strategies appears at this point, see Fig. 5.21a.



(a) AIC_c vs. number of LMN inputs



(b) Selected inputs

Figure 5.21: Obtained results of the linked $\underline{x}$-$\underline{z}$-input selection for the metamodel of axial fans

- 24 LMN inputs because the lowest AIC_c value is achieved in case of FS.
- 27 LMN inputs because the lowest AIC_c value is achieved in case of BE.
- 28 LMN inputs, corresponding to a model incorporating all available inputs.

Table 5.3: NRMSE values for models based on different LMN input subsets evaluated on the test data set of the axial fan metamodel

Number of LMN inputs	11	24	27	28
Selected BE subset	**0.5934**	0.5168	**0.4814**	0.4339
Selected FS subset	0.6450	**0.4948**	0.5305	0.4339

Again, it is hard to decide whether one of the two search strategies delivers significantly better subsets. A noteworthy fact is that, according to the test errors, no model quality improvement can be achieved by removing any of the 28 LMN inputs. This could not be concluded from the obtained AIC_c values, see Fig. 5.21a. However, this indicates again that (I) only physically meaningful inputs have been chosen to describe the behavior of the axial fans and (II) the experimental design ensures a meaningful variation of all inputs.

Prior to the input selection of the axial fan metamodel, an expert in the field of axial fans made an assumption about the most important inputs according to his expertise. Seven inputs are stated as presumably most important for the efficiency of the axial fans, including

- the volume flow u_1,
- the target volume flow u_2,
- the number of blades u_3,
- the hub-to-tip ratio u_4,
- the tip clearance u_5,
- the chord length at midspan u_{12}, and
- the maximum camber at midspan u_{15}.

Reasons for this assumption can be found, e.g., in [22]. It is remarkable that in case of the BE search strategy the subset with seven LMN inputs contains five of these inputs, see Fig. 5.21b. The subset containing seven inputs found with FS includes four of the assumed seven most important inputs.

Summary

For the two fan metamodels the mixed wrapper-embedded input selection approach is not able to improve the model accuracy by removing inputs through the linked $\underline{x}$-$\underline{z}$-input selection. Both metamodels are a good example for reasonably selected inputs and a well chosen DoE. However, with the trend of an ever increasing input space dimensionality it will get harder, even for domain experts, to select the right amount of inputs for the best bias/variance tradeoff. Even though the mixed wrapper-embedded input selection approach is not able to improve the accuracy of

the metamodels, information about the most important inputs is gathered. In case of the axial fan metamodel expert knowledge confirms the results regarding the input importance obtained from the input selection. Input selection allows to gain insights and confidence in the data-driven models which is a very important factor for industrial acceptance of these abstract approaches.

5.4 Heating, Ventilating, and Air Conditioning System

A heating, ventilating, and air conditioning (HVAC) system, that has already been investigated in [119] and [108] is the real-world application dealt with in this section. The goal is to build a dynamic model that can be used for model predictive control [118]. The focus in this section lies on the generation of a good dynamic model and not on the model predictive control. The mixed wrapper-embedded input selection approach with the separated $\underline{x}$-$\underline{z}$-input selection is applied to the HVAC system in order to find the best subset of inputs. Three data sets are available, that serve as training, validation, and test data, which are described in more detail at the end of this section. The training data is only used for the estimation of all LMN parameters during the training with LOLIMOT. Here, LOLIMOT is used instead of HILOMOT due to the increased required computation time that results from the very high number of potential LMN inputs, which is 140, see Section 5.4.1 for details. With the help of the simulation error on validation data the model complexity, i.e. the number of local models, is chosen. Additionally, the simulation error on validation data is used as evaluation criterion during the separated $\underline{x}$-$\underline{z}$-input selection. For this investigation the results obtained with the validation error as evaluation criterion for the separated $\underline{x}$-$\underline{z}$-input selection turned out to be superior compared to using the $\mathrm{AIC_c}$. The test data set is only used after the separated $\underline{x}$-$\underline{z}$-input selection is finished. The reason for the inferiority of the $\mathrm{AIC_c}$ as evaluation criterion in this particular application is unknown. It is assumed that the $\mathrm{AIC_c}$ is not that well suited for the model complexity determination of dynamic models for short sampling times and therefore also struggles as evaluation criterion during the input selection.

Problem Configuration

The setup presented in Fig. 5.22 shows a typical application of a series connection of cooling and heating coils. The air is dehumidified by the cooling coil in order to adjust the air humidity. Since the air temperature is decreased by the cooling coil too, the air has to be reheated by the heating coil in order to meet the desired temperature. The power of the coils can be adjusted by valves. For the cooling coil the water mass flow is varied via the valve position u_1, whereas for the heating coil the mixing ratio of the returned cold water and the hot supply water is varied via the valve position u_2. For this configuration, a constant air mass flow of $\dot{m}_a = 1$ kg per second is assumed. The output signals of the system are the temperature y_1 and the relative humidity y_2 of the outlet air. The input signals of the system are the valve positions u_1 and u_2, the temperature d_1 and the relative humidity d_2 of the inlet air, the inlet water temperature d_3 of the cooling coil and the water supply temperature d_4 of the heating circuit. The valve position u_1 and u_2 are actuating variables which can be varied arbitrarily to excite the dynamic system as desired. In contrast to that, the LMN inputs d_1, d_2, d_3, and d_4 can be measured but not be manipulated freely. The units of the measured temperatures and humidities are °C and %, respectively. The data sets used in this work are generated on a real world system shown in Fig. 5.23. This pilot plant is provided by the company Fischer&Co Luft- und Klimatechnik in Graz/Austria. Figures 5.22 and 5.23 are kindly provided by Daniel Schwingshackl (University of Klagenfurth), Jakob Rehrl (Graz University

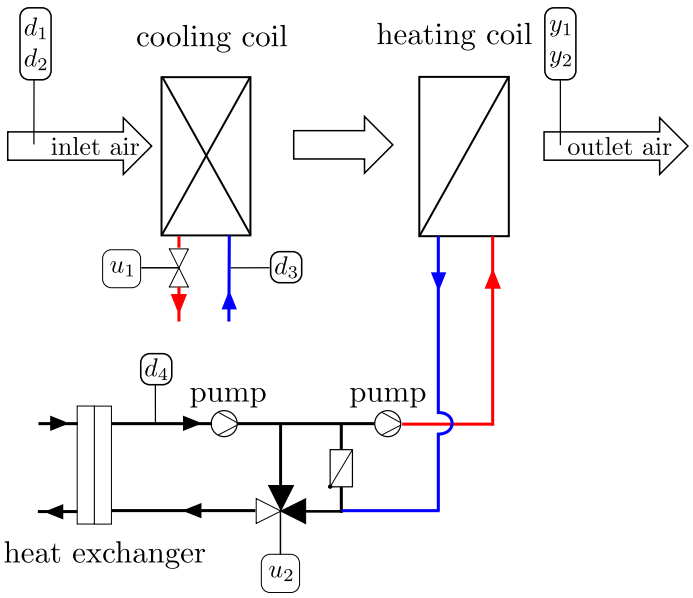


Figure 5.22: Schematic sketch of the HVAC system

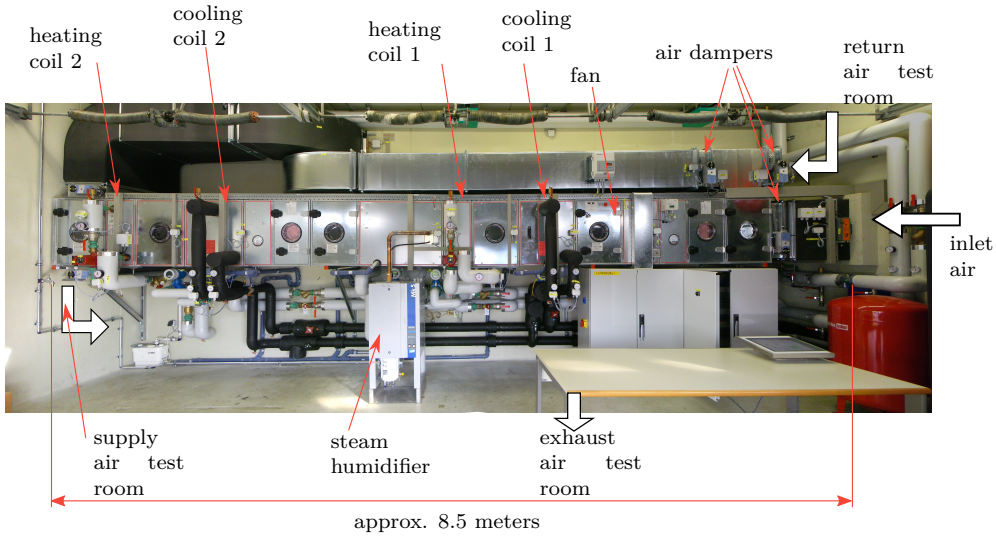


Figure 5.23: Picture of the real-world HVAC system [119]

of Technology), and Martin Horn (Graz University of Technology).

Available Data Sets

The training data set consists of $N_{\text{train}} = 5490$ samples, the validation data set has $N_{\text{val}} = 224$ samples and there are $N_{\text{test}} = 262$ test samples. The sampling frequency for all data sets is $f_0 = 1/16$ Hz. As already mentioned, only two valve positions u_1 and u_2 can be manipulated. As can be seen in Fig. 5.24, the excitation signal for the valve positions consists of several steps around different operating points. The variation in the measurable disturbances is by far weaker and lower frequent compared to the manipulated variables. Figure 5.24 also shows the signals of the control variables. Especially the dependency of both control variables on the second valve position u_2 is clearly observable.

5.4.1 Mixed Wrapper-Embedded Input Selection

The search strategies used for the separated $\underline{x}$-$\underline{z}$-input selection are backward elimination (BE) and forward selection (FS). In order to test the mixed wrapper-embedded input selection, we assume to have very limited insights on the physical background of the HVAC system. Nevertheless the maximum and minimum delay n_{max} and n_{min}, respectively, have to be provided for the BE search strategy because it starts

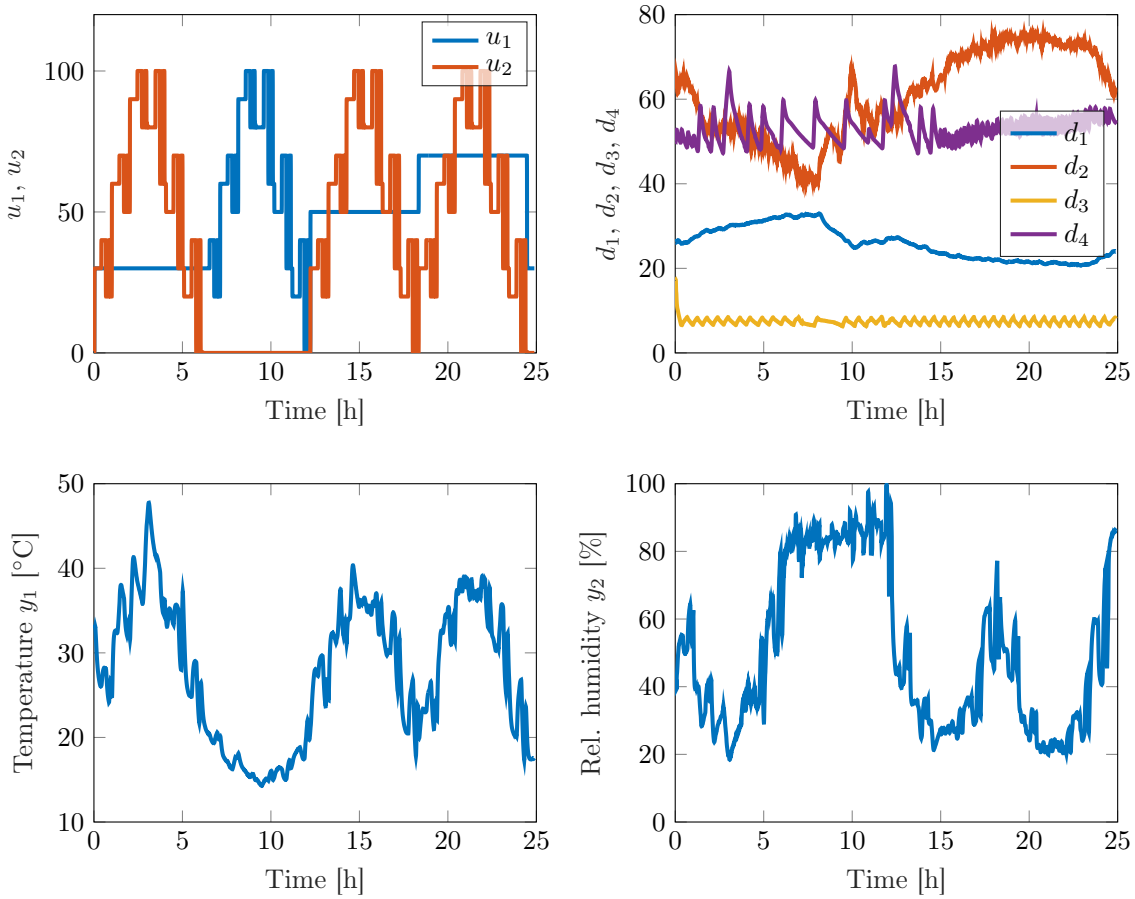
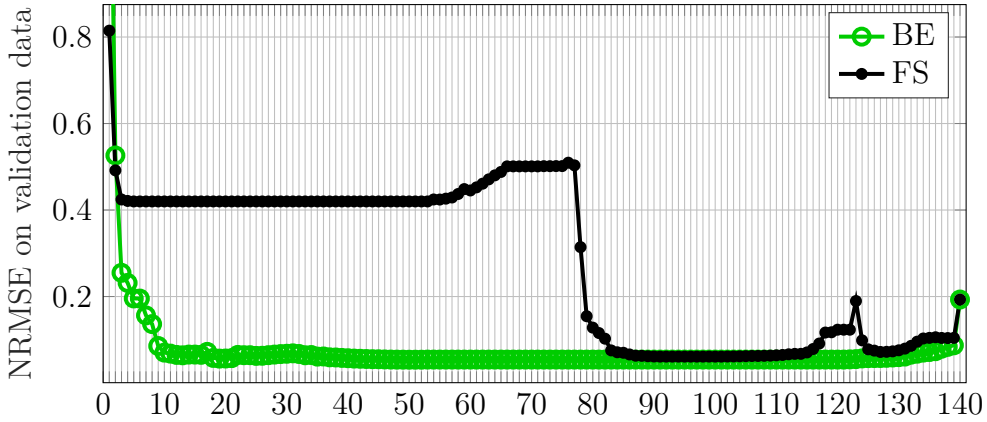


Figure 5.24: HVAC training data set consisting of the manipulated variables u_1 and u_2 (valve positions), the measurable disturbances d_1 (inlet air temperature), d_2 (rel. humidity inlet air), d_3 (inlet water temperature cooling coil), d_4 (water supply temperature), and the control variables y_1 (outlet air temperature) and y_2 (rel. humidity outlet air)

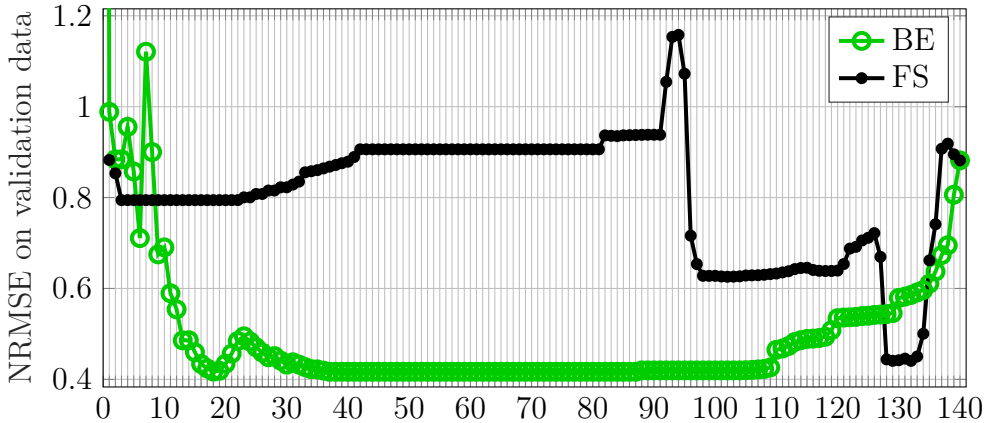
with all potential inputs. n_{min} and n_{max} are chosen to be 1 and 10, respectively, for all physical inputs as well as for all physical outputs. For each target value, i.e. the temperature y_1 and the relative humidity y_2, distinct models are built and therefore distinct input selections are performed. Because of the chosen minimum and maximum delays, there are 70 possible LMN inputs for each of the two input spaces, i.e. the $\underline{x}$- and $\underline{z}$-input space. As a result the total amount of potential LMN inputs is virtually doubled to 140. Following the separated $\underline{x}$-$\underline{z}$-input selection with the two search strategies BE and FS, the methods choose LMN inputs to be removed or added without paying attention from which input space it is eliminated from or added to.

Results

For the performed separated $\underline{x}$-$\underline{z}$-input selection, Fig. 5.25a shows results for the temperature y_1 and Fig. 5.25b the results for the relative humidity y_2. The validation error is plotted against the number of LMN inputs for both search strategies. It is evident that for both outputs y_1 and y_2 the subsets found by BE are far superior to the ones found by FS. Especially for input subsets containing less than 80 inputs, the difference in the model qualities achieved with the found subsets is huge. The main reason for this turns out to be a lack of delayed outputs in the $\underline{x}$-input space of the LMNs in the FS case. For the temperature y_1, the FS selects the first delayed output, in particular $y_1(k-3)$, as the 78-th LMN input. Therefore, subsets with less



(a) Temperature y_1



(b) Relative humidity y_2

Figure 5.25: Obtained results of the separated $\underline{x}$-$\underline{z}$-input selection for the HVAC system

than 78 LMN inputs lead to local finite impulse response (FIR) models. In case of the relative humidity y_2 the same effect takes place, but the first delayed output, in particular $y_2(k-6)$, is selected as 94-th LMN input. In this application one huge disadvantage of FS becomes clearly visible. The interaction between several inputs are not discoverable by simply adding one input at a time. In contrast to that, when following a BE strategy, all interactions are present from the start. The deterioration of the model performance by removing one input that is important in combination with any other input is directly recognizable. In the following only results of the BE are further discussed because of the far worse model accuracies obtained with subsets found by FS.

The performance improvement through the mixed wrapper-embedded input selection with BE as search strategy is significant for both outputs. In general, the achieved model performance for the temperature is by far better than for the relative humidity. One noticeable aspect are the long plateaus of the simulated validation error for both output variables. For the temperature the same validation error is achieved between 54 and 121 LMN inputs. In case of the relative humidity there are two of these plateaus, one ranging from 37 up to 87, and the other from 88 up to 104. We discovered that during these plateaus only inputs from the $\underline{z}$-input space are eliminated that are never chosen as split directions. The $\underline{x}$-input space is kept unchanged. Therefore discarding those LMN inputs does neither affect the $\underline{z}$-input space partitioning nor the parameter estimation of the local models. As a result all LMNs corresponding to one plateau are in fact identical, which explains the constant validation errors. The calculations took 90 hours (17 for FS) for the temperature and 41 hours (14 for FS) for the relative humidity on a cluster node with 12 CPUs and 48 GB of RAM. The difference in the calculation times originates from the fact, that generally a higher complexity (more local models) was chosen for the temperature models throughout the whole input selection procedure. This might also explain the much lower error rates for the temperature model.

For the temperature, the optimal subset size of LMN inputs according to the simulation error on validation data is 50, for the relative humidity it is 37. Now several LMN input subsets are compared to each other in terms of their NRMSE values (J) on all three available data sets, see Table 5.4. Besides the best LMN input subsets, models including all LMN inputs are listed. In addition, local minima with far less LMN inputs are compared as well, in particular the temperature model relying on 19 LMN inputs and the relative humidity model relying on 18 LMN inputs. Highlighted loss function values indicate the best value for each data set and each output. Rows

two to four of Table 5.4 show the loss function values for the model of the temperature. Especially the performance improvement on validation and test data through the separated $\underline{x}$-$\underline{z}$-input selection is impressive. Even with only 19 LMN inputs almost the same model quality is achieved compared to a model with 50 LMN inputs. Since the curves of the simulated model outputs for 19 and 50 LMN inputs are barely distinguishable, Fig. 5.26 shows only the case with 19 LMN inputs. In addition the process output (here the temperature y_1) from the test data and the test error is shown.

Rows five to seven of Table 5.4 show the model performances for the relative humidity. Here the improvement on validation data is significant, but the improvement on test data is only moderate. Eventually, the generalization performance does not reach the same accuracy level as the temperature model, but the simulated model output follows the measured relative humidity quite well. Figure 5.27 demonstrates

Table 5.4: NRMSE values (J) on different LMN input subsets for the training, validation and test data sets

number of LMN inputs	J_{train}	J_{vali}	J_{test}
19 LMN inputs (y_1)	0.013	0.052	**0.230**
50 LMN inputs (y_1)	**0.013**	**0.049**	0.248
140 LMN inputs (y_1)	0.025	0.193	0.550
18 LMN inputs (y_2)	**0.046**	0.417	0.723
37 LMN inputs (y_2)	0.070	**0.417**	**0.408**
140 LMN inputs (y_2)	0.073	0.882	0.459

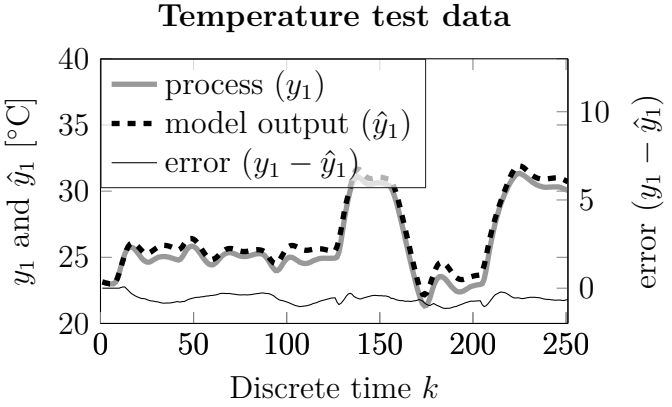


Figure 5.26: Measured vs. simulated temperature for the test data set with 19 LMN inputs

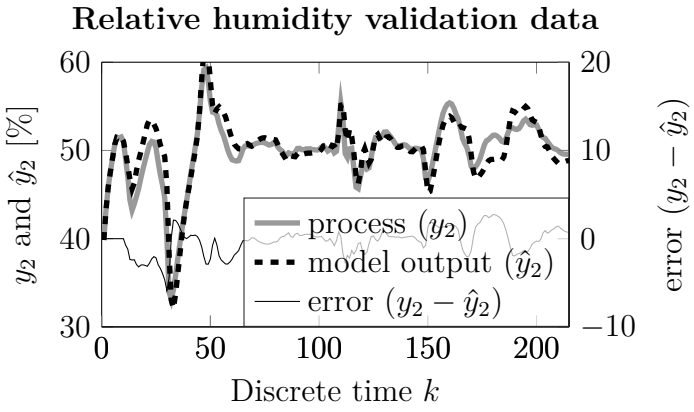


Figure 5.27: Measured vs. simulated relative humidity for the validation data set with 37 LMN inputs

this for 37 LMN inputs on the validation data.

In addition to the gain in model performance, the separated $\underline{x}$-$\underline{z}$-input selection makes the final model more concise and simplifies its interpretation. After a closer look at the best LMN input subsets it is remarkable that for both outputs there are only two LMN inputs left in the $\underline{z}$-input space (serving as operating point variables). In both cases, these are the same physical inputs, namely the valve positions u_1 and u_2, but with different delays. Actually, nonlinearities originating from the valve characteristics are identified correctly to be important for the definition of an operating point. This is extremely appealing since these are the manipulated variables in control of the HVAC system. Therefore their influence is most important and the accuracy of the model w.r.t. u_1 and u_2 is crucial. Since the excitation of u_1 and u_2 in the training data is most active it is a natural outcome of the input and order selection procedure to keep both in the LMN inputs.

For the model of the temperature the LMN inputs that are left in the $\underline{z}$-input space are $u_1(k-3)$ and $u_2(k-1)$. The LMN inputs $u_1(k-2)$ and $u_2(k-2)$ are in the relative humidity model's $\underline{z}$-input space. Since the dimensionality of both $\underline{z}$-input spaces is two-dimensional, we can visualize the partitioning of them. Figure 5.28 shows the partitioning for the temperature model exemplarily. More splits along the u_2-axis indicate a stronger nonlinearity in that direction. Each validity area contains a number that corresponds to the relevant local affine model.

The interpretation of the selected LMN inputs for the affine local models ($\underline{x}$-input space) is difficult, since there are so many variables left. Easy interpretation, however, regards the relative humidity model, for which the physical input d_1 (inlet air

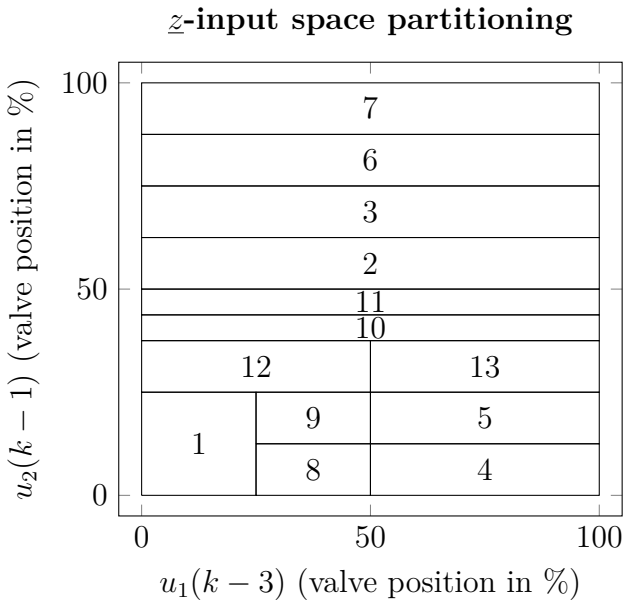


Figure 5.28: Partitioning of the $\underline{z}$-input space for the model of the temperature

temperature) is neglected completely, i.e. none of its delayed versions is used. In summary, it is shown that the mixed wrapper-embedded input selection approach, in particular the separated $\underline{x}$-$\underline{z}$-input selection, for the modeling of the HVAC system leads to a significant improvement and simplification of the model as shown in Table 5.4. Additionally, the interpretability is increased at least for the $\underline{z}$-input space that can be described for both outputs with only two operating point variables.

6 Conclusions and Outlook

This thesis is settled in the field of experimental modeling and specifically focuses on the weakening of the effects of the *curse of dimensionality*. This phrase describes the exponential increase in effort with an increasing input space dimensionality, see Section 2.2 for more details. Therefore there is the need to keep the input space dimensionality of the models as low as possible. On the other hand, there is a trend to increase the input space dimensionality in modern model-based applications mainly due to the following two reasons:

- Ever increasing process complexities create the demand for higher degrees of freedom to control them via additional actuators, which leads to potentially more inputs for the models.

- The demand for higher model accuracies originating from stricter regulations and market demands, such that potentially more influences ($\hat{=}$ inputs) have to be considered to fulfill them.

Because of the increased process complexities it gets harder even for domain experts to determine the right amount of inputs used for an experimental modeling task in order to find a good compromise for the aforementioned conflict.

This thesis investigates the possibility to automatically select the right amount of inputs solely based on data. All presented input selection methods heavily rely on special properties of local model networks (LMNs). In particular, the possibility to separate linear from nonlinear influences when using local affine models is exploited exhaustively, see Section 2.3 and Chapter 3 for more details. In addition to input selection methods, the curse of dimensionality is also tackled by the development and investigation of design of experiments (DoE) techniques.

Conclusions

The findings for all input selection methods are summarized in the following.

The Mixed wrapper-embedded input selection (MWEIS) approach exploits the input space separation of LMNs into a $\underline{x}$- and $\underline{z}$-input space fully, enabling it to distinguish between linear and nonlinear effects as explained in detail in Section 2.3 and 3.2. It is shown that the combination of a backward elimination (BE) as search strategy and Akaike's information criterion ($\mathrm{AIC_c}$) together outperform other combinations of search strategies and evaluation criteria. Other investigated search strategies are: Forward selection (FS), an exhaustive search, and a genetic algorithm. Other investigated evaluation criteria include: 10-fold cross-validation and the model error on a distinct validation data set. With the help of test processes the ability to detect linear and nonlinear effects is demonstrated. For the auto miles per gallon (MPG) data set it is shown that exploiting the separability of linear and nonlinear effects leads to advantages compared to classical wrapper approaches. In this example, all physical inputs are included in the best subset, but exclusively only either in the $\underline{x}$- or $\underline{z}$-input space. In case of the prediction of the air-mass flow (AMF) into the combustion chamber of a gasoline engine, a model structure is identified that meets exactly the available expert knowledge about the process. Eventually, the mixed wrapper-embedded input selection approach is able to increase the accuracy of a heating, ventilating, and air conditioning (HVAC) model by finding the appropriate dynamic model order for both input spaces. For the HVAC system possible shortcomings of FS are demonstrated impressively. Models based on subsets found by BE are magnitudes better than models based on subsets found by FS.

The regularization-based input selection (RBIS) approach penalizes the obliqueness of splits that are made in order to partition the $\underline{z}$-input space. In principle, it can be implemented for any axis-oblique partitioning strategy based on split optimizations. In this thesis, it is implemented for the already existing HIerarchical LOcal MOdel Tree (HILOMOT) algorithm as explained in detail in Section 3.3. Unfortunately this approach is not able to show the desired behavior for the test processes, which includes an improvement of the LMN's generalization performance as well as a reduction of the variance error. It turns out that no good compromise could be found for the split regularization parameter that is able to keep splits parallel to unimportant $\underline{z}$-inputs

while being able to reliably produce oblique splits in subspaces where they are needed. However, in the auto MPG application the RBIS approach improved the generalization performance clearly. A likely cause is the increased input dimensionality for the auto MPG application compared to the used test processes. It might well be, that the benefit of the RBIS approach comes only into play for higher-dimensional problems.

The embedded approach that analyzes the partitioning of an already trained LMN rates all available inputs in the $\underline{z}$-input space quantitatively as explained in detail in Section 3.4. Currently this quantitative rating serves just as information for the user and is not further utilized. The embedded approach is able to rank the inputs of the $\underline{z}$-input space correctly for the test processes. However, it is not clear from which relevance factor value on an $\underline{z}$-input should be considered as irrelevant. Reliably determining a threshold seems difficult. In case of the AMF application the partition analysis fails to indicate all relevant inputs for the $\underline{z}$-input space. From the mixed wrapper-embedded input selection it is known which physical inputs should be assigned to the $\underline{z}$-input space. Only one of them, namely the engine speed, is indicated to be relevant by the partition analysis for the $\underline{z}$-input space.

Partial dependence plots are not scientifically new, but are reviewed in this thesis as a tool to visualize the average influence of single inputs on the model output. The distinction between linearly and nonlinearly influencing inputs seems to be possible from the observation of the mean curves in case of the test processes. Despite all advantages, partial dependence plots have to be interpreted with care. It has to be considered that the on-average-effect of an input might be different from the effect at specific points in the input space. Additionally, the auto MPG application shows problems of partial dependence plots with correlated inputs, here the number of cylinders and the displacement. As a result the model is forced to extrapolate heavily for the generation of the partial dependence plot since there is no data from which the model could learn a proper prediction of the MPG in case of small displacements and a high number of cylinders. This leads to implausible and wrong trends of the corresponding partial dependence plot.

The findings for all design of experiments topics dealt with in this thesis are summarized in the following.

The order of experimentation is important if the model of interest should be used as early as possible. In particular, the model may already be used while measurements are still in progress. The focus lies on the best possible model accuracy at any time. Several distance-based methods have been developed to determine the order of experimentation in order fo fulfill the aforementioned goals. The methods are the biggest gap sequence (BGS), the median distance sequence (MDS), and the intelligent k-means sequence (IKMS). All order determination methods are compared to each other and to multiple random orderings of the measurements. In addition these methods are also compared to an active learning strategy that serves as reference. The results obtained on test processes generated with the function generator described in Section 2.7 and for the generation of a computational fluid dynamics (CFD) metamodel point out that the IKMS and BGS methods perform equally well and by far better than the MDS method. In fact IKMS and BGS perform on average as good as the best case of all randomly generated orderings. Even though the active learning strategy can use more information about the test process under investigation, it is only on par with the IKMS and BGS method.

Should corners be measured? This question is investigated with the help of the function generator described in Section 2.7 for several extrapolation scenarios and ratios of corner points in relation to the overall data set size. The goal is to give recommendations for cases in which the number of corners is a substantial amount of all data points that can be measured. It turns out that measuring corners can only be recommended if „enough“ data points remain inside the design space and large extrapolation is required.

The comparison of space-filling designs with the help of the function generator described in Section 2.7 incorporates Sobol sequences, data coming from a uniform distribution, and several maximin optimized Latin Hypercube (LH) designs. The optimized LH designs differ in the way they are optimized. In particular, maximin optimized LH designs generated with the extended deterministic local search (EDLS) algorithm from phase one and two (see Section 4.2.2 for details) are used as well as one function implemented in the commercially available software Matlab. Maximin LH designs created by the EDLS algorithm (and probably other optimization schemes as well) are superior to all other investigated experimental designs regarding both the achieved model qualities and the variation of these model qualities. These advantages have to be paid off by higher computational effort for the creation of the DoE.

However, the maximin LH designs can be optimized and stored in advance to a specific task, such that this drawback can be weakened at least for some applications. Another drawback of the maximin LH designs regards their extensibility. If just a few points should be added to an already optimized LH design it is not as easy as to append additional points to an existing Sobol sequence, because further optimization runs are necessary. The results are important for both the initial DoE and subsequent refinements of it in presumably near-optimum regions of the design space.

The goal-oriented active learning with LMNs is an extension of HILOMOT for design of experiments (HilomotDoE) and addresses simultaneously three main goals: (I) The concentration on possibly near-optimum regions and (II) the focus on areas in the design space where the (meta-)model's performance is considered to be worst. Additionally, (III) new measurements should differ from already gathered data as much as possible. With these goals three important issues in modeling are addressed simultaneously: (I) optimality, (II) model bias, (III) model variance/uniformly space-filling property. The proposed goal-oriented active learning with LMNs proved to fulfill all goals during the generation of a data set used to train CFD metamodels of centrifugal fans. The maximum reachable efficiencies could be improved, the overall model performance could be increased, and the diversity increase of the training data samples can be seen in the extension of achievable design points.

Outlook

For future research the following topics are of interest in the opinion of the author.

- Since the mixed wrapper-embedded input selection is computational demanding and takes quite a lot of time, the following idea should lead to a speed-up of the necessary calculation times. The author suggests to increase the complexity penalty of Akaike's information criterion for the determination of the model complexity during the input selection or to utilize the Bayesian information criterion instead. As a result, the number of local models should decrease on average and each individual training should take less time to be completed. One difficulty might be to find a reasonable value for the complexity penalty in order to still guarantee good bias/variance tradeoffs.

- The mixed wrapper-embedded input selection should be tested in combination with LMNs employing local models that are regularized finite impulse response (FIR) models as described in [91]. In particular, a $\underline{z}$-input selection seems to be reasonable in case of LMNs with local, regularized FIR models because the regularization already takes care of a good bias/variance tradeoff for the $\underline{x}$-input space (or the rule consequents). However, to the knowledge of the author, the automatic selection of inputs for the $\underline{z}$-input space (or the rule premises) has not been addressed so far.

- Because the regularization-based input selection revealed advantages only for a rather high input space dimensionality, in particular for the auto MPG application, it is suggested to further investigate this approach with higher-dimensional test processes, i.e. $p \gg 4$.

- For the regularization-based input selection a different procedure is suggested for further research. In the procedure explained in Section 3.3, shrinkage of the splitting parameters belonging to important inputs for the $\underline{z}$-input space is avoided by a subsequent unregularized split optimization incorporating only the important $\underline{z}$-inputs ($v_i \neq 0$) *for each split*. It is suggested to omit this subsequent unregularized split optimization for each split and to analyze the resulting partitioning of the final LMN with the embedded input selection approach presented in Section 3.4.1. Afterwards a training with the standard HILOMOT algorithm (without any split regularization) is performed where all inputs that have a relevance factor of zero are removed from the $\underline{z}$-input space. Advantages of the newly suggested procedure could be:

  - The decision if an input is relevant for the $\underline{z}$-input space is based on several splits and should therefore be more reliable.

  - The unregularized split optimization within the standard HILOMOT algorithm circumvents the need of finding a good compromise between axis-oblique and axis-parallel splits. Relevant inputs for the $\underline{z}$-input space are identified prior to the standard HILOMOT algorithm. In the subsequent standard HILOMOT algorithm splits in the remaining $\underline{z}$-input space can be arbitrarily oblique without being penalized.

- Once information is obtained about which inputs act in a (mostly) linear or a nonlinear way by the mixed wrapper-embedded input selection, this information should be utilized for future experimental designs. Inputs only contained

in the $\underline{x}$-input space can be treated differently in the DoE opposed to inputs that are contained in the $\underline{z}$-input space. Depending on the used local model type, optimal experimental designs can be used, e.g. D-optimal, A-optimal, or G-optimal ones, see [40, 1, 39]. Space-filling designs are suggested for the inputs contained in the $\underline{z}$-input space. Because the properties of the nonlinearity are typically not known in advance, i.e. before measurements are taken, the $\underline{z}$-input space should be uniformly covered. First investigations have already been carried out in [60].

- As shown for the auto MPG application, partial dependence plots are prone to yield implausible results if inputs are highly correlated. Therefore it is suggested to omit all input combinations that have to be evaluated by the model in order to create the partial dependence plots that force the model to extrapolate. For the decision whether an input combination forces the model to extrapolate, a one-class-classification algorithm, like e.g. [73] can be used.

A Data Splitting

In experimental modeling the main task is to create models solely based on measurements with a high ability to generalize well on unseen data [15]. As described in Section 2.2, using all available data in order to tune the parameters of an artificial neural network is inappropriate because overfitting can not be detected. Therefore, all available data is typically split into several data sets which are then used for training, validation, and testing, also described in Section 2.2. At best, the training, validation, and test data set contain all typical characteristics of the overall available data. Depending on the data set in which typical characteristics are missing, problems are likely to occur. For the following considerations it is assumed that a typical characteristic is missing only in one of the three data sets.

Training data set: The model is not able to learn some aspects of the process under consideration and therefore is not able to generalize well. The lack of generalization performance can be detected by the validation or test data set because these data sets contain the missing characteristics.

Validation data set: The model is able to learn all aspects of the process under consideration. However, the model might overfit the specific characteristic missing in the validation data set. Because this characteristic is not contained in the validation data set, poor generalization behavior related to it can not be detected. As a result, the model complexity might be determined suboptimally. With the help of the test data the generalization performance can be assessed properly.

Test data set: The model is able to learn all aspects of the process under consideration and the model complexity is determined properly. However, the assessment of the model quality with the test data set might be overly optimistic or pessimistic.

The danger of lacking characteristics in one of the three data sets grows with a decreasing data set size, if the data is split randomly the worst-case scenario may be

arbitrarily bad. Unfortunately, simple random splitting is used in most applications according to [109]. Especially in engineering applications data is often scarce because measurements are time-consuming and expensive. Therefore, additional effort should be put in the data splitting strategy. Commonly used data splitting strategies can be found in [85] and [109]. For this thesis, a rather simple deterministic and distance-based strategy to split the available data into a training and a test data set is carried out. The explicit generation of a validation data set is omitted here, since Akaike's information criterion ($\mathrm{AIC_c}$) (calculated with the training data) proved to be a reliable substitute, at least for local model networks (LMNs) as shown in [56]. The used data splitting method is explained in the following.

The user-demanded amount of test data should be selected from the whole data set such that all regions of the input space are represented. At the same time, big data gaps in the remaining training data set should avoided. In order to fulfill this requirements, all points contained in the whole data set are sorted according to the biggest gap sequence (BGS) which is explained in detail in Section 4.1.1 and illustrated in Fig. A.1. As a result, one number is assigned to each point, representing its place on the sorted list, see Fig. A.1d. If the user-demanded number of test data points is N_{test}, the first N_{test} points from the second half of the sorted list are chosen as test data as visualized in Fig. A.2. The reasoning behind this heuristic is the following. Points at the beginning of the sorted list would probably lead to extrapolation in the test data. As a result, the test error would be overly pessimistic. Points at the end of the sorted list might be very close to points in the training data set, leading to an overly optimistic test error. According to the author's experience, the selected heuristic leads to a good compromise and thus to a realistic evaluation of the generalization performance of the model.

In the example shown in Fig. A.1, the whole data set contains $N = 12$ samples and three points should be chosen as test data. Therefore, point 7, 8, and 9 are selected, which are highlighted in Fig. A.1d. If the first three points from the sorted list would have been chosen as test data, points 2 and 3 might already force the resulting model to extrapolate. If the last three points would have been chosen as test data, points 11 and 12 are very close to points 4 and 5, respectively.

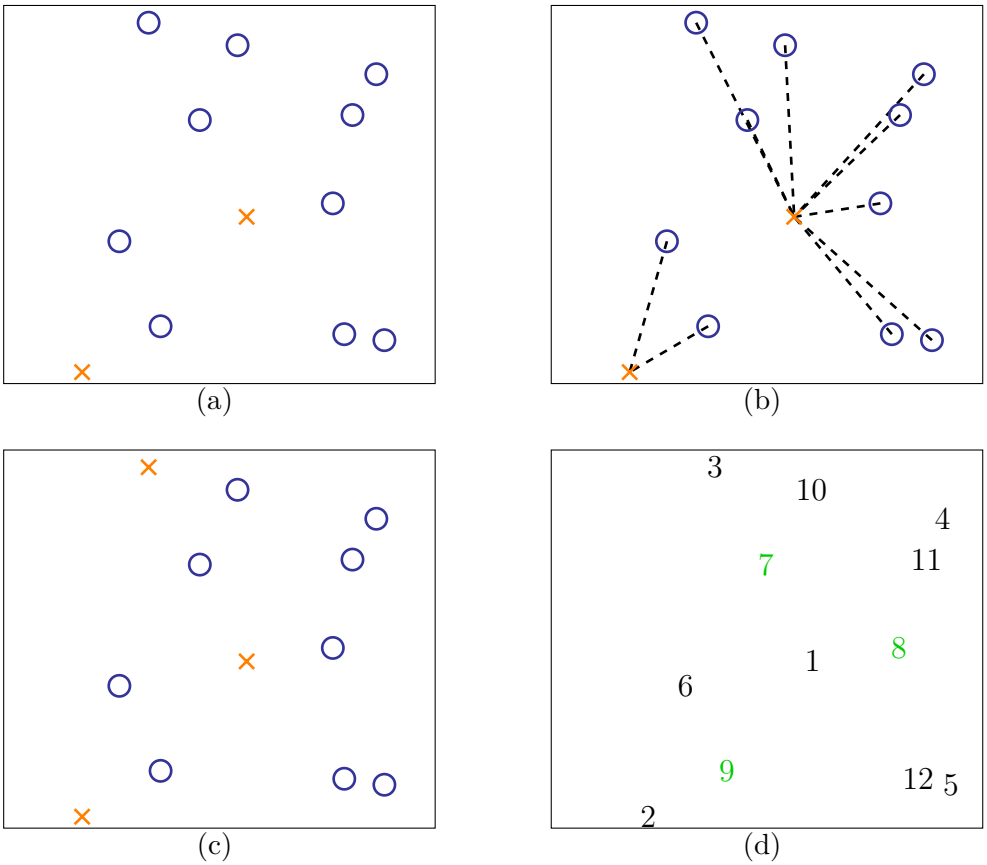


Figure A.1: Illustration of the data sorting according to the BGS procedure. (a) The first two sorted points (x) are the one closest to the center of gravity of all points and the one that is farthest away from it. (b)-(c) The next point (x) is iteratively added to the sorted list by choosing the one of the remaining points (o) with the maximum distance to the next already sorted point. (d) Numbers denote the place in the sorted list of all points.

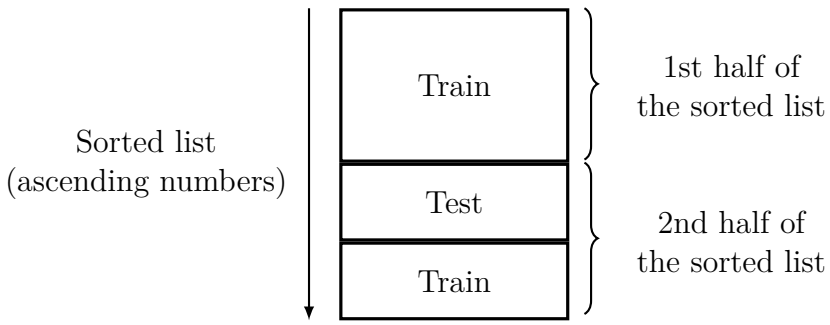


Figure A.2: Visualization of the heuristic used to pick test data from the whole data set sorted according to the BGS method

References

[1] AC Atkinson. The Usefulness of Optimum Experimental Designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 59–76, 1996.

[2] Mihiar Ayoubi. *Nonlinear System Identification Based on Neural Networks with Locally Distributed Dynamics and Application to Technical Processes.* PhD thesis, TU Darmstadt, 1996.

[3] Konrad Bamberger. *Aerodynamic Optimization of Low-Pressure Axial Fans.* PhD thesis, University of Siegen, November 2015.

[4] Konrad Bamberger, Julian Belz, Thomas Carolus, and Oliver Nelles. Aerodynamic Optimization of Centrifugal Fans Using CFD-Trained Meta-Models. In *16th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC)*, Hawaii, USA, April 2016.

[5] Oliver Bänfer, Oliver Nelles, Josef Kainz, and Johannes Beer. Local Model Networks with Modified Parabolic Membership Functions. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 1, pages 179–183. IEEE, 2009.

[6] Russell R Barton. Metamodeling: A State of the Art Review. In *Simulation Conference Proceedings, 1994. Winter*, pages 237–244. IEEE, 1994.

[7] Richard E. Bellman. *Adaptive Control Processes.* Princeton University Press, 1961.

[8] Julian Belz, Konrad Bamberger, and Oliver Nelles. Order of Experimentation for Metamodeling Tasks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 4843–4849, Vancouver, Canada, July 2016.

[9] Julian Belz, Konrad Bamberger, Oliver Nelles, and Thomas Carolus. Goal-Oriented Active Learning with Local Model Networks. *International Journal of Computational Methods and Experimental Measurements*, 6(4):785–796, 2018.

[10] Julian Belz and Oliver Nelles. Function Generator Application: Shall Corners Be Measured? In *Proceedings of the 25th Workshop on Computational Intelligence*, pages 271–287, Dortmund, Germany, November 2015.

[11] Julian Belz and Oliver Nelles. Proposal for a Function Generator and Extrapolation Analysis. In *IEEE International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, pages 282–287, Madrid, Spain, September 2015.

[12] Julian Belz and Oliver Nelles. Normalized L1 Regularization for Axis-Oblique Tree Construction Algorithms. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, Hawaii, USA, 2017. IEEE.

[13] Richard A Berk. *Statistical Learning From a Regression Perspective*. Springer Science & Business Media, 2008.

[14] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science & Business Media, 2006.

[16] Frank P Bleier. *Fan Handbook: Selection, Application, and Design*. McGraw-Hill New York, 1998.

[17] L Bommes, J Fricke, and R Grundmann. *Ventilatoren*. Vulkan Verlag, Essen, 2003.

[18] L. Breiman. Hinging Hyperplanes for Regression, Classification, and Function Approximation. *Information Theory, IEEE Transactions on*, 39(3):999–1013, 1993.

[19] L. Breiman and P. Spector. Submodel Selection and Evaluation in Regression. The X-Random Case. *International Statistical Review/Revue Internationale de Statistique*, pages 291–319, 1992.

[20] K.P. Burnham and D.R. Anderson. Multimodel Inference Understanding AIC and BIC in Model Selection. *Sociological methods & research*, 33(2):261–304, 2004.

[21] Xiwen Cai, Haobo Qiu, Liang Gao, and Xinyu Shao. Metamodeling for High Dimensional Design Problems by Multi-Fidelity Simulations. *Structural and Multidisciplinary Optimization*, pages 1–16, 2017.

[22] Thomas Carolus. *Ventilatoren*. Vieweg+Teubner Verlag, 2013.

[23] Martin Casdagli. A Dynamical Systems Approach to Modeling Input-Output Systems. In *A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity*, volume 12, page 265 ff. Addison-Wesley Publishing Co, 1992.

[24] Ray-Bing Chen, Dai-Ni Hsieh, Ying Hung, and Weichung Wang. Optimizing Latin Hypercube Designs by Particle Swarm. *Statistics and Computing*, 23(5):663–676, 2013.

[25] Victoria CP Chen, Kwok-Leung Tsui, Russell R Barton, and Martin Meckesheimer. A Review on Design, Modeling and Applications of Computer Experiments. *IIE transactions*, 38(4):273–291, 2006.

[26] Mark Ming-Tso Chiang and Boris Mirkin. Intelligent Choice of the Number of Clusters in k-Means Clustering: An Experimental Study with Different Cluster Spreads. *Journal of classification*, 27(1):3–40, 2010.

[27] S.L. Chiu. Selecting input variables for fuzzy models. *Journal of Intelligent and Fuzzy Systems-Applications in Engineering and Technology*, 4(4):243–256, 1996.

[28] Biagio Ciuffo, Jordi Casas, Marcello Montanino, Josep Perarnau, and Vincenzo Punzo. Gaussian Process Metamodels for Sensitivity Analysis of Traffic Simulation Models: Case Study of AIMSUN Mesoscopic Model. *Transportation Research Record: Journal of the Transportation Research Board*, (2390):87–98, 2013.

[29] Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang. *Principles and Theory for Data Mining and Machine Learning*. Springer Science & Business Media, 2009.

[30] David A Cohn. Minimizing Statistical Bias with Queries. Technical report, DTIC Document, 1995.

[31] David A. Cohn. Neural Network Exploration Using Optimal Experiment Design. *Neural Networks*, 9(6):1071 – 1083, 1996.

[32] David A. Cohn, Zoubin Ghahramani, and Michael I Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 1996.

[33] O Cordier. Ähnlichkeitsbedingungen für Strömungsmaschinen. *Brennstoff-Wärme-Kraft (BWK)*, 5(10):337–340, 1953.

[34] Alexander A Correa, Pere Grima, and Xavier Tort-Martorell. Experimentation Order in Factorial Designs: New Findings. *Journal of Applied Statistics*, 39(7):1577–1591, 2012.

[35] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Heinz, Geritt Kampmann, and Oliver Nelles. Extended Deterministic Local Search Algorithm for Maximin Latin Hypercube Designs. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 375–382, Cape Town, South Africa, December 2015.

[36] Christoph Engel. *Untersuchung der Laufradströmung in einem Radialventilator mittels Particle Image Velocimetry (PIV).* PhD thesis, Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften» Maschinenbau und Verfahrenstechnik» Institut für Energie-und Umweltverfahrenstechnik, 2007.

[37] S. Ernst. Hinging Hyperplane Trees for Approximation and Identification. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 2, pages 1266–1271. IEEE, 1998.

[38] Kai-Tai Fang and Runze Li. Uniform Design for Computer Experiments and its Optimal Properties. *International Journal of Materials and Product Technology*, 25(1-3):198–210, 2005.

[39] Valerii V Fedorov and Peter Hackl. *Model-Oriented Design of Experiments*, volume 125 of *Lecture Notes in Statistics*. Springer Science & Business Media, 2012.

[40] Valerii Vadimovich Fedorov. *Theory of Optimal Experiments*. Elsevier, 1972.

[41] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A Clustering Technique for the Identification of Piecewise Affine Systems. *Automatica*, 39(2):205–217, 2003.

[42] Torsten Fischer, Benjamin Hartmann, and Oliver Nelles. Increasing the Performance of a Training Algorithm for Local Model Networks. In *World Congress of Engineering and Computer Science (WCECS)*, pages 1104–1109, San Francisco, USA, October 2012.

[43] Ian Ford, DM Titterington, and Christos P Kitsos. Recent Advances in Nonlinear Experimental Design. *Technometrics*, 31(1):49–60x, 1989.

[44] Bjarne A Foss and Tor A Johansen. On Local and Fuzzy Modelling. In *Third International Conference on Industrial Fuzzy Control and Intelligent Systems (IFIS)*, pages 80–87. IEEE, 1993.

[45] Jason AS Freeman and David Saad. Learning and Generalization in Radial Basis Function Networks. *Neural Computation*, 7(5):1000–1020, 1995.

[46] Jerome H Friedman. Multivariate Adaptive Regression Splines. *The annals of statistics*, pages 1–67, 1991.

[47] Jerome H Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, pages 1189–1232, 2001.

[48] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural computation*, 4(1):1–58, 1992.

[49] A. Grosso, A. Jamali, and M. Locatelli. Finding Maximin Latin Hypercube Designs by Iterated Local Search Heuristics. *European Journal of Operational Research*, 197(2):541–547, 2009.

[50] Jie Gui, Zhenan Sun, Shuiwang Ji, Dacheng Tao, and Tieniu Tan. Feature Selection Based on Structured Sparsity: A Comprehensive Study. *IEEE transactions on neural networks and learning systems*, 2017.

[51] I. Guyon. *Feature Extraction: Foundations and Applications*, volume 207. Springer Verlag, 2006.

[52] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

[53] Chris Harris, Xia Hong, and Qiang Gan. *Adaptive Modelling, Estimation and Fusion From Data: A Neurofuzzy Approach*. Springer-Verlag Berlin Heidelberg, 2002.

[54] B. Hartmann, T. Ebert, and O. Nelles. Model-Based Design of Experiments Based on Local Model Networks for Nonlinear Processes with Low Noise Levels. In *American Control Conference (ACC), 2011*, pages 5306–5311. IEEE, 2011.

[55] B. Hartmann and O. Nelles. Adaptive Test Planning for the Calibration of Combustion Engines – Methodology. *Design of Experiments (DoE) in Engine Development*, pages 1–16, 2013.

[56] Benjamin Hartmann. *Lokale Modellnetze zur Identifikation und Versuchsplanung nichtlinearer Systeme*. PhD thesis, Universität Siegen, April 2014.

[57] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*, volume 1. Springer New York, 2001.

[58] Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Wiley Online Library, 1990.

[59] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Inc., second edition, 1999.

[60] Tim Oliver Heinz, Julian Belz, and Oliver Nelles. Design of Experiments – Combining Linear and Nonlinear Inputs. In *Proceedings of the 27th Workshop on Computational Intelligence*, pages 211–226, Dortmund, Germany, 2017.

[61] John B. Heywood. *Internal Combustion Engine Fundamentals*. McGraw-Hill, Inc., 1988.

[62] Hisham Hilow. Comparison Among Run Order Algorithms for Sequential Factorial Experiments. *Computational Statistics & Data Analysis*, 58:397–406, 2013.

[63] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive Mixtures of Local Experts. *Neural computation*, 3(1):79–87, 1991.

[64] J.S.R. Jang. Input selection for anfis learning. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 1493–1499. IEEE, 1996.

[65] Ruichen Jin, Wei Chen, and Agus Sudjianto. On Sequential Sampling for Global Metamodeling in Engineering Design. In *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 539–548. American Society of Mechanical Engineers, 2002.

[66] Ruichen Jin, Wei Chen, and Agus Sudjianto. An Efficient Algorithm for Constructing Optimal Design of Computer Experiments. *Journal of Statistical Planning and Inference*, 134(1):268–287, 2005.

[67] Mark E Johnson, Leslie M Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990.

[68] Rachel T Johnson, Douglas C Montgomery, Bradley Jones, and John W Fowler. Comparing Designs for Computer Simulation Experiments. In *Proceedings of the 40th Conference on Winter Simulation*, pages 463–470. Winter Simulation Conference, 2008.

[69] Antonia J Jones. New Tools in Non-Linear Modelling and Prediction. *Computational Management Science*, 1(2):109–149, 2004.

[70] Roger D Jones, YC Lee, CW Barnes, GW Flake, K Lee, PS Lewis, and S Qian. Function Approximation and Time Series Prediction with Neural Networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 649–665. IEEE, 1990.

[71] Michael I Jordan and Robert A Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural computation*, 6(2):181–214, 1994.

[72] Olumayowa T Kajero, Tao Chen, Yuan Yao, Yao-Chen Chuang, and David Shan Hill Wong. Meta-Modelling in Chemical Process System Engineering. *Journal of the Taiwan Institute of Chemical Engineers*, 2016.

[73] Geritt Kampmann and Oliver Nelles. One-class LS-SVM with Zero Leave-One-Out Error. In *2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, pages 1–6, Dec 2014.

[74] M. Karagiannopoulos, D. Anyfantis, SB Kotsiantis, and PE Pintelas. Feature selection for regression problems. *Proceedings of HERCMA07*, 2007.

[75] Ron Kohavi et al. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

[76] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273 – 324, 1997.

[77] F. Kursawe and H.-P. Schwefel. Optimierung mit Evolutionären Algorithmen. *Automatisierungstechnische Praxis*, 39(9):10–17, 1997.

[78] IJ Leontaritis and Stephen A Billings. Input-Output Parametric Models for Non-Linear Systems Part I: Deterministic Non-Linear Systems. *International Journal of Control*, 41(2):303–328, 1985.

[79] Asriel U Levin and Kumpati S Narendra. Identification Using Feedforward Networks. *Neural Computation*, 7(2):349–369, 1995.

[80] M. Lichman. UCI Machine Learning Repository, 2013.

[81] H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Chapman & Hall, 2007.

[82] Chu Kiong Loo and Mandava Rajeswari. Growing Multi-Experts Network. In *TENCON 2000. Proceedings*, volume 3, pages 472–477. IEEE, 2000.

[83] David Lowe. Adaptive Radial Basis Function Nonlinearities, and the Problem of Generalisation. In *First IEE International Conference on Artificial Neural Networks*, pages 171–175. IET, 1989.

[84] David JC MacKay. Information-Based Objective Functions for Active Data Selection. *Neural computation*, 4(4):590–604, 1992.

[85] Robert J May, Holger R Maier, and Graeme C Dandy. Data Splitting for Artificial Neural Networks Using SOM-Based Stratified Sampling. *Neural Networks*, 23(2):283–294, 2010.

[86] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245, 1979.

[87] M. Meckesheimer, A. J. Booker, R. Barton, and T. Simpson. Computationally Inexpensive Metamodel Assessment Strategies. *AIAA journal*, 40(10):2053–2060, 2002.

[88] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, 1992.

[89] Boris Mirkin. *Clustering for Data Mining: A Data Recovery Approach*. Chapman & Hall/CRC, London, 2005.

[90] Douglas C Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2008.

[91] Tobias Münker and Oliver Nelles. Local Model Network with Regularized MISO Finite Impulse Response Models. In *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*, pages 1–8. IEEE, 2016.

[92] Bruce Roy Munson, Donald F Young, and Theodore Hisao Okiishi. *Fundamentals of Fluid Mechanics*. New York, 1990.

[93] M. Munson and R. Caruana. On Feature Selection, Bias-Variance, and Bagging. *Machine Learning and Knowledge Discovery in Databases*, pages 144–159, 2009.

[94] Roderick Murray-Smith. Local Model Networks and Local Learning. *Fuzzy Duisburg*, 94:404–409, 1994.

[95] Roderick Murray-Smith and T.A. Johansen. Local Learning in Local Model Networks. In *Artificial Neural Networks, 1995., Fourth International Conference on*, pages 40–46. IET, 1995.

[96] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001.

[97] Oliver Nelles. Axes-Oblique Partitioning Strategies for Local Model Networks. In *IEEE International Symposium on Intelligent Control*, pages 2378–2383, Munich, Germany, October 2006.

[98] Oliver Nelles, Oliver Bänfer, Josef Kainz, and Johannes Beer. Local Model Networks - The Prospective Method for Modeling in Electronic Control Units? *ATZelektronik worldwide*, 3(6):36–39, 2008.

[99] Oliver Nelles and Rolf Isermann. Basis Function Networks for Interpolation of Local Linear Models. In *Proceedings of the 35th IEEE Conference on Decision and Control (CDC)*, volume 1, pages 470–475, 1996.

[100] Oliver Nelles, S Sinsel, and R Isermann. Local Basis Function Networks for Identification of a Turbocharger. In *UKACC International Conference on Control (Conf. Publ. No. 427)*, volume 1, pages 7–12. IET, 1996.

[101] Harald Niederreiter. Low-Discrepancy and Low-Dispersion Sequences. *Journal of number theory*, 30(1):51–70, 1988.

[102] Andreas Poncet, Jean L. Poncet, and George S. Moschytz. On the Input-Output Approximation of Nonlinear Systems. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 1500–1503. IEEE, 1995.

[103] Michael JD Powell. Radial Basis Functions for Multivariable Interpolation: A Review. In *Algorithms for Approximation*, pages 143–167. Clarendon Press, 1987.

[104] Luc Pronzato and Werner G Müller. Design of Computer Experiments: Space Filling and Beyond. *Statistics and Computing*, 22(3):681–701, 2012.

[105] Predrag Pucar and Mille Millnert. Smooth Hinging Hyperplanes - An Alternative to Neural Nets. In *Proceedings of 3rd European Control Conference*, volume 2, pages 1173–1178, 1995.

[106] J Ross Quinlan. Combining Instance-Based and Model-Based Learning. In *ICML*, page 236, 1993.

[107] C.E. Rasmussen and C. Williams. Gaussian Processes for Machine Learning. 2006.

[108] Jakob Rehrl, Daniel Schwingshackl, and Martin Horn. A Modeling Approach for HVAC Systems Based on the LoLiMoT Algorithm. In *19th IFAC World Congress*, pages 10862–10868, 2014.

[109] Z. Reitermanová. Data splitting. *WDS 2010 Proceedings of Contributed Papers*, 1:31–36, 2010.

[110] J. Reunanen. Overfitting in Making Comparisons Between Variable Selection Methods. *The Journal of Machine Learning Research*, 3:1371–1382, 2003.

[111] Carl Rhodes and Manfred Morari. Determining the Model Order of Nonlinear Input/Output Systems Directly from Data. In *IEEE Proceedings of the American Control Conference*, volume 3, pages 2190–2194, 1995.

[112] Carl Rhodes and Manfred Morari. Determining the Model Order of Nonlinear Input/Output Systems. *AIChE Journal*, 44(1):151–163, 1998.

[113] Ryan M Rifkin and Ross A Lippert. Notes on Regularized Least Squares. 2007.

[114] Stefan Schaal and Christopher G Atkeson. From Isolation to Cooperation: An Alternative View of a System of Experts. *Advances in Neural Information Processing Systems*, pages 605–611, 1996.

[115] Stefan Schaal and Christopher G Atkeson. Receptive Field Weighted Regression. *ATR Human Information Processing Laboratories, Tech. Rep. TR-H-209*, 1997.

[116] Benedikt Gregor Eric Schenker. *Prediction and Control Using Feedback Neural Networks and Partial Models*. PhD thesis, Swiss Federal Institute of Technology Zürich, Switzerland, 1996.

[117] Mark Schmidt, Glenn Fung, and Romer Rosales. Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches. In *Machine Learning: ECML 2007*, pages 286–297. Springer, 2007.

[118] Daniel Schwingshackl, Jakob Rehrl, and Martin Horn. Model Predictive Control of a HVAC System Based on the LoLiMoT Algorithm. In *European Control Conference (ECC)*, pages 4328–4333, 2013.

[119] Daniel Schwingshackl, Jakob Rehrl, and Martin Horn. LoLiMoT Based MPC for Air Handling Units in HVAC Systems. *Building and Environment*, 96:250 – 259, 2016.

[120] Burr Settles. Active Learning Literature Survey. *University of Wisconsin, Madison*, 52:55–66, 2010.

[121] S. Shan and G. G. Wang. Metamodeling for High Dimensional Simulation-Based Design Problems. *Journal of Mechanical Design*, 132:051009, 2010.

[122] R. Sindelar and R. Babuska. Input selection for nonlinear regression models. *Fuzzy Systems, IEEE Transactions on*, 12(5):688–696, 2004.

[123] Amith Singhee and Rob A Rutenbar. Why Quasi-Monte Carlo is Better than Monte Carlo or Latin Hypercube Sampling for Statistical Circuit Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(11):1763–1776, 2010.

[124] Anders Skeppstedt, Lennart Ljung, and Mille Millnert. Construction of Composite Models from Observed Data. *International Journal of Control*, 55(1):141–152, 1992.

[125] Ilya Meerovich Sobol'. On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86 – 112, 1967.

[126] Torsten Söderström and Petre Stoica. *System Identification*. Prentice Hall International (UK) Ltd., 1989.

[127] K Stokbro, DK Umberger, and JA Hertz. Exploiting Neurons with Localized Receptive Fields to Learn Chaos. *Complex Systems*, 4(3):603–622, 1990.

[128] Michio Sugeno and GT Kang. Structure Identification of Fuzzy Model. *Fuzzy Sets and Systems*, 28(1):15–33, 1988.

[129] Tomohiro Takagi and Michio Sugeno. Fuzzy Identification of Systems and its Applications to Modeling and Control. *IEEE transactions on Systems, Man, and Cybernetics*, (1):116–132, 1985.

[130] Tatiana Tambouratzis. Counter-clustering for training pattern selection. *The Computer Journal*, 43(3):177–190, 2000.

[131] P.N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to Data Mining*. Pearson Addison Wesley Boston, 2006.

[132] Robert Tibshirani. Regression Shrinkage and Selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[133] Ah Chung Tsoi and Andrew D Back. Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures. *IEEE Transactions on Neural Networks*, 5(2):229–239, 1994.

[134] Hua-Ping Wan and Wei-Xin Ren. Parameter Selection in Finite-Element-Model Updating by Global Sensitivity Analysis Using Gaussian Process Metamodel. *Journal of Structural Engineering*, 141(6):04014164, 2014.

[135] G Gary Wang. Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points. *Journal of Mechanical Design*, 125(2):210–220, 2003.

[136] G Gary Wang, Zuomin Dong, and Peter Aitchison. Adaptive Response Surface Method–A Global Optimization Scheme for Approximation-Based Design Problems. *Engineering Optimization*, 33(6):707–734, 2001.

[137] G Gary Wang and S Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

[138] Lipo Wang, Yaoli Wang, and Qing Chang. Feature Selection Methods for Big Data Bioinformatics: A Survey from the Search Perspective. *Methods*, 111:21–31, 2016.

[139] Ronald J Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989.

[140] Reinhard Willinger. Das CORDIER-Diagramm für Strömungsarbeitsmaschinen: Eine theoretische Begründung mittels Stufenkennlinien. In *VDI-Berichte*, number 2112, pages 17–28, 2010.

[141] Reinhard Willinger. Theoretical Interpretation of the CORDIER-Lines for Squirrel-Cage and Cross-Flow Fans. In *Proc. ASME TurboExpo*, pages 675–684, Copenhagen, Denmark, 2012.

[142] Reinhard Willinger and Michael Köhler. Influence of Blade Loading Criteria and Design Limits on the Cordier-Line for Axial Flow Fans. In *Proc. ASME TurboExpo*, Düsseldorf, Germany, 2014.

[143] Achilleas Zapranis and Apostolos-Paul Refenes. *Principles of Neural Model Identification, Selection and Adequacy: With Applications to Financial Econometrics*. Springer-Verlag London Limited, 1999.

[144] Hui Zou and Trevor Hastie. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.