TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Regelungstechnik

Modeling and Optimization for Stationary Base Engine Calibration

Benjamin Berger

Vollständiger Abdruck der von der Fakultät für Maschinenwesen
der Technischen Universität München zur Erlangung
des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Georg Wachtmeister
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. habil. Boris Lohmann
	2. Univ.-Prof. Dr.-Ing. Clemens Gühmann, (Technische Universität Berlin)

Die Dissertation wurde am 28.06.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 29.11.2012 angenommen.

To my family and Carina

ABSTRACT

This thesis presents new approaches and results for modeling and optimization for stationary base engine calibration.

At first, the requirements on the modeling are discussed, in order to determine the most suitable modeling technique for this topic in an extensive comparison. The Gaussian process modeling can be identified as the most promising approach. With the Gaussian process modeling, the highest precision with a low amount of measurements can be achieved, the modeling can be fully automated with the maximum marginal likelihood method, a dependable performance on complex problems can be obtained and an accurate prediction of the uncertainty of the model can be estimated.

However, recent approaches in engine calibration do not consider outliers in the measurements and an automatic adaption to bad distributed data. Therefore, based on the results of the model comparison, the most promising modeling technique is enhanced in order that a new robust modeling framework for stationary base engine calibration can be obtained. An automatic transformation of the measurement data ensures that the modeling assumptions on the data distributions are met. Even if outliers are contained in the data set, a robust Gaussian process formulation guarantees that the modeling is asymptotically unbiased, meaning that the model is tending to the real engine behavior as the number of measurements tends towards infinity.

Since state of the art model-based online optimizations for engine calibration do not use a fully probabilistic approach and can only handle a single objective function, a new, improved online optimization approach is introduced. As a Gaussian process modeling is used, additional information, such as an accurate prediction of the variance and the marginal likelihood probability density function of the model parameters, can be exploited for the online modeling, in order to obtain an increased performance at a lower amount of measurements compared to other approaches. With the new multi-objective online optimization, more objectives can be regarded and the Pareto optimal areas can be determined.

All these new contributions enhance the performance for modeling and optimization, and therefore they are able to reduce time and costs on the test bench, improve the reliability of modeling and optimization results, assist the calibration engineers and increase the user acceptance of model-based techniques in engine calibration. Various theoretical examples and practical applications demonstrate the performance of these new approaches.

ACKNOWLEDGMENTS

This thesis summarizes the results of my appointment at the Institute of Automatic Control at the Technische Universität München in cooperation with KRATZER AUTOMATION AG.

First, I would like to deeply thank my thesis advisor Prof. Boris Lohmann for taking the risk and offering me the opportunity to join the Institute of Automatic Control and to achieve a doctorate degree.

I would like to thank Prof. Clemens Gühmann for his interest in my work and for being the second examiner of this thesis, and I would also like to thank Prof. Georg Wachtmeister for chairing my thesis committee.

Furthermore, I am grateful to my former and current colleagues at the institute for their support and for providing a warm and friendly atmosphere. You made me feel comfortable right from the start.
Moreover, I would like to deeply thank all my students for their great contributions.

I am very grateful to KRATZER AUTOMATION AG for financing this project. Especially, I would like to deeply thank Florian Rauscher. I benefited greatly from the many fruitful discussions with him, and he was always an immense support to me, especially during the hard times.
In addition, I would like to thank Dino Reichelt from MAN Truck & Bus AG Nürnberg, who gave me the opportunity to test the algorithms under real conditions.

A very special appreciation goes to my friends and my family for their understanding and support. Extraordinary thanks go to my parents, Bruno and Edith Berger, and to my sister Stefanie Berger for their love, care and support throughout all my years of education.

My deepest thanks go to Carina Freutsmiedl. I know that I spent very much time on working in the last months. You helped me at most during these days. I will never forget that.
It is time to give you something back!

Munich, 03.July 2012

TABLE OF CONTENTS

SYMBOLS AND NOTATION

Matrices are capitalized and vectors are in bold type. A consistent notation is used throughout the thesis. In the following, the most important symbols and abbreviations are given.

Symbols:

$\|\mathbf{A}\|$	determinant of $\mathbf{A}$ matrix
$p(a, b)$	joint probability (density), probability of a and b
$p(a\|b)$	conditional probability (density), probability of a given b
$\mathcal{D}$	set of measurements, $\mathcal{D} := \{(\mathbf{x}_n, t_n)\|n \in \{1...N\}\}$
D	dimension of input space $\mathcal{X}$
ϵ	noise
$\mathbb{E}$	expectation
k	kernel function, $k(\mathbf{x}, \mathbf{x}') := \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}')$
$\mathbf{K}$	Gram (covariance) matrix, $\mathbf{K} := \boldsymbol{\Phi}\boldsymbol{\Phi}^T$
λ, α, β	regularization parameters
$\mathcal{L}$	likelihood function
M	number of model parameters
$\mathcal{N}$	normal distribution, Gaussian distribution
N	number of measurements
ϕ_j	basis function, $\mathbb{R}^D \to \mathbb{R}$
$\boldsymbol{\phi}$	set of basis functions, $\boldsymbol{\phi} := (\phi_1, ..., \phi_M)^T$
$\boldsymbol{\Phi}$	design matrix, defined by $(\boldsymbol{\Phi})_{n,j} = \phi_j(\mathbf{x}_n)$
t_n	target / measured value of the n-th measurement
$\mathbf{t}$	targets / measured values (of the set of measurements), $\mathbf{t} := (t_1, ..., t_N)^T$
$\boldsymbol{\Theta}$	model parameters
$\mathbb{V}$	variance
$\mathbf{x}$	input vector (state vector), $\mathbf{x} \in \mathcal{X}$
$\mathbf{x}_*$	input vector of the prediction (test input), $\mathbf{x}_* \in \mathcal{X}$
$\mathbf{x}_n$	input vector of the n-th measurement, $\mathbf{x}_n \in \mathcal{X}$
$\mathbf{X}$	inputs (of the set of measurements), $\mathbf{X} := (\mathbf{x}_1, ..., \mathbf{x}_N)^T$
$\mathcal{X}$	input space, $\mathcal{X} \subset \mathbb{R}^D$
y	model
y_*	model prediction

Abbreviations:

DoE	design of experiments
ECU	electronic control unit
GP	Gaussian process
HHT	hinging hyperplane tree
KASIO	KRATZER system identification and optimization toolbox
LLR	linear model with local RBF terms
LOLIMOT	local linear model tree
MBOO	model-based online optimization
MCMC	Markov chain Monte Carlo (methods)
MLP	multilayer perceptron (neural network)
MOO	multi-objective optimization
NRMSE	normalized root mean square error
NSGA-II	non-dominated sorting genetic algorithm-II (MOO algorithm)
RBF	radial basis function (neural network)
RMSE	root mean square error
RSSE	regularized sum of squares error function, defined in (2.15)
RVM	relevance vector machine
SE	squared exponential (kernel function)
SSE	sum of squares error function (least squares), defined in (2.6)
SVM	support vector machine

Chapter 1

INTRODUCTION

The internal combustion engine is a widely used system for mobile propulsion in vehicles. In recent years the manufacturers of the engines have to meet two new demands:

- In order to reduce air pollution, governments around the world introduced emission standards. An example are the European emission standards (EURO standards), which were introduced in the EU member states. Two important quantities of the EURO standards, the particulate matter and the sum of HC and NOx emissions of a diesel engine for passenger cars, are shown in figure 1.1.

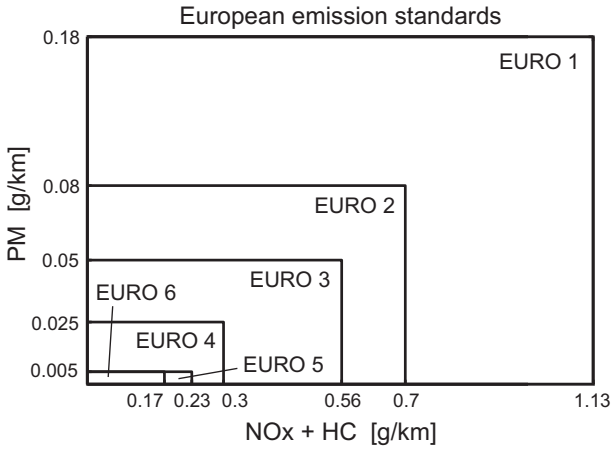


Figure 1.1: NOx+HC and particulate matter of the EURO standards

- Due to increasing oil prices, the demand for low consumption engines rises. At the same time, the customer does not accept limitations in dynamics and comfort, which is putting increasing pressure on the automobile industry.

In order to meet these demands, the manufacturers introduced new technologies, e.g. variable valve train or exhaust gas recirculation. With these technologies, new degrees of freedom enable to control the combustion process more precise. Figure 1.2 shows some of them and the goals which are pursued.
The task of engine calibration deals with the question how to calibrate these adjustment parameters.

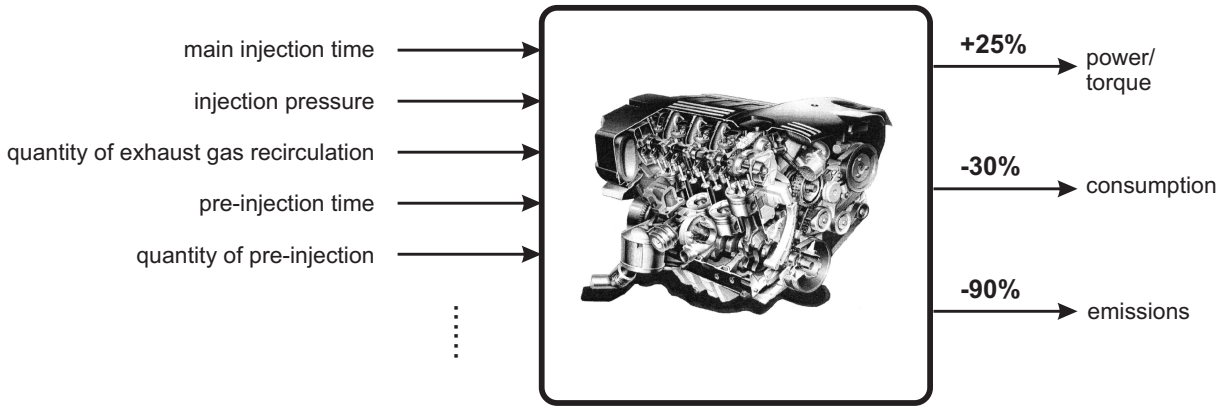


Figure 1.2: The number of parameters increased rapidly in the last years. Power and torque could be increased, consumption and emissions could simultaneously be reduced, see [53].

1.1 Introduction into Engine Calibration

In order to control the additional parameters of the engine, the Electronic Control Unit (ECU) was introduced in the 1970s [12]. In engine calibration, measurements, e.g. from the test bench, are taken to parameterize the ECU with optimal settings for these parameters. In former days these tasks could be performed manually by engineers and test bench operators. Since the number of labels on the ECU and the complexity of the problem is increasing rapidly in the last years, new methods have been developed and used [76].

Figure 1.3 gives an overview of several calibration tasks, see [53]. The vehicle calibration covers the calibration of different comfort and dynamic functions for the transient states of the vehicle and the calibration of the transmission. The stationary calibration covers numerous, very different tasks. The torque structure is the central coordinating function of the ECU, which uses the required torque as a reference. Other functions provide a safe and clean operation of the engine. For more information see [12, 53, 76].

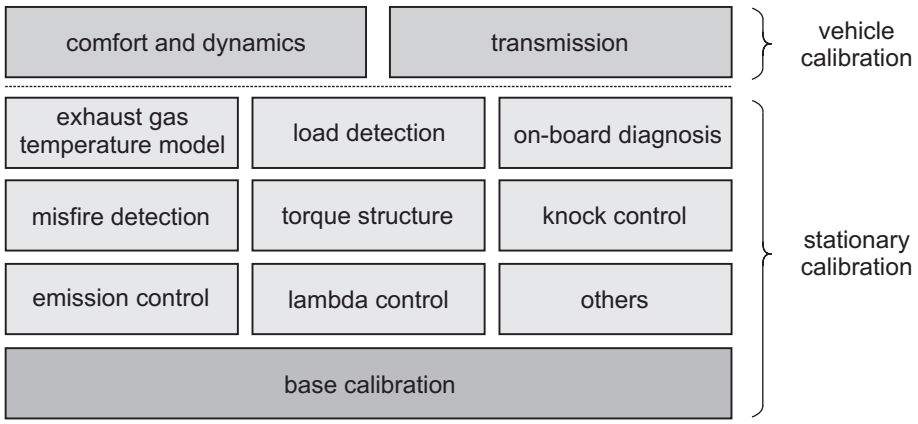


Figure 1.3: Examples of different tasks in engine calibration, see [53].

The base calibration constitutes the prerequisite for all other calibration tasks. In this task the settings of the basic adjustment parameters, like main injection time or injection pressure, are determined and stored in maps on the ECU. These engine operating maps are functions of a discretization of the whole operating range X_{OP}, which is spanned by the torque and speed of the engine. The operating range X_{OP}, its discretization into single operating points x_{OP} and the d-dimensional parameter space X_P, which is spanned by the d adjustment parameters, are shown in figure 1.4. As we will see soon, due to physical restrictions, e.g. engine limits like knocking or too high exhaust gas temperature, not every point in the parameter space X_P can be set on the engine. Therefore, on every operating point x_{OP} the parameter space X_P reduces to the feasible parameter space $X_{FP}(x_{OP}) := \{x \in X_P | (x_{OP}, x)$ can be set on the engine$\}$.
The aim of base calibration is now, to find the optimal values x_{opt} for the parameters for every operating point x_{OP} given a certain objective function $\Phi : X_{OP} \times X_P \to \mathbb{R}$, which can be written as

$$x_{opt}(x_{OP}) = \underset{x \in X_{FP}(x_{OP})}{\arg\min} \; \Phi\left((x_{OP}, x)\right). \tag{1.1}$$

At the end, the optimal settings $(x_{opt}(x_{OP}))_i$ for each parameter $(X_P)_i$, $i \in \{1, .., d\}$ are stored in maps on the ECU, as shown at the bottom right in figure 1.4.

In most cases many different and conflicting objectives have to be considered in engine calibration. Hence, the objective function Φ is a compromise of a multi-objective function $\Psi : X_{OP} \times X_{FP} \to \mathbb{R}^{d_{Obj}}$, which covers the d_{Obj} different objectives. The way how this compromise is chosen, depends on the distinct areas in the operating range X_{OP}, see figure 1.4. The part load (pl) area is the biggest area in the operating range X_{OP} and occurs most commonly in the driving cycle, where the EURO standards are measured. The objectives in this area are mainly focused on consumption and emissions. At full load the maximum of power and torque should be achieved and in the idle area a good engine smoothness should be considered, in order to realize a maximum in comfort.
Often, parameters are optimized which have only a low dynamic, like the valve which controls the exhaust gas recirculation. Therefore, the smoothness of the engine maps of these parameters influences the dynamic behavior of the vehicle, since a big change in the settings of the parameters needs a lot of time, see also [35] and [53]. Hence, another objective in engine calibration is to get smooth engine maps for parameters with low dynamics. This goal is often achieved by a subsequent smoothing of the maps, see [86] and [35].

At the end two different approaches for optimization, which are quite common in engine calibration, should be mentioned: local and global optimization. In local optimization only a single operation point x_{OP} at a time is considered for calculation of the optimal parameters x_{opt}. If engine load and speed are also taken into account, then a global optimization is performed. Clearly, it is obvious that e.g. the optimization of the smoothness of the maps is a global problem. Therefore, it is clear that not all problems can be solved by local optimization and often global optimization has to be performed.

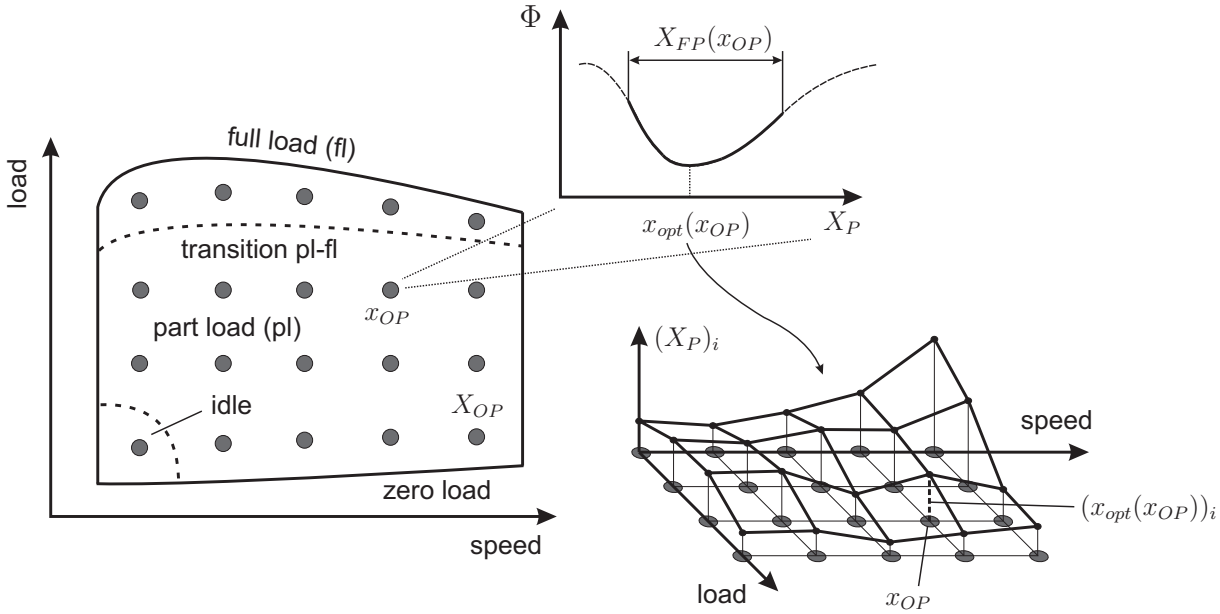


Figure 1.4: On the left side the operating range X_{OP} with the distinct areas are shown, see also [21]. In every area the objective Φ for calibration is different. On top, the objective function Φ over the parameter space X_P is plotted. The aim of base calibration is, to find the minimum x_{opt} of the objective function and to store this minimum in engine operating maps on the ECU. On the bottom right a map of a single parameter i is shown. A more detailed explanation is given in the text.

1.2 State of the Art in Engine Calibration

In this section an overview of state of the art techniques for engine calibration is given. Two main approaches, a measurement-based and a model-based optimization, can be distinguished.

1.2.1 Measurement-Based Optimization

If only few parameters should be optimized (2-3 parameters), then the parameter space is low-dimensional and measurement-based optimization can be used [21]. This approach has many advantages. The techniques are simple to implement and easy to understand. Therefore, the engineer needs no special knowledge and can interpret the results very fast. However, as we will see soon, due to the curse of dimensionality these approaches cannot be used for an optimization with many parameters. Since the number of parameters increased rapidly in the last years, as said above, nowadays measurement-based optimization is only used for special problems. Therefore, this thesis focuses rather on model-based optimization. However, for a deeper understanding of the problem and for the sake of completeness, two different approaches of measurement-based optimization are given in the following. For a further discussion see [53] and [21].

1.2.1.1 Full Factorial Optimization

In this approach the parameter space is discretized into a dense grid, which is often called a full factorial design, and measurements are taken at each point. As said above, for d parameters the parameter space is d-dimensional and the number of measurements N scales with

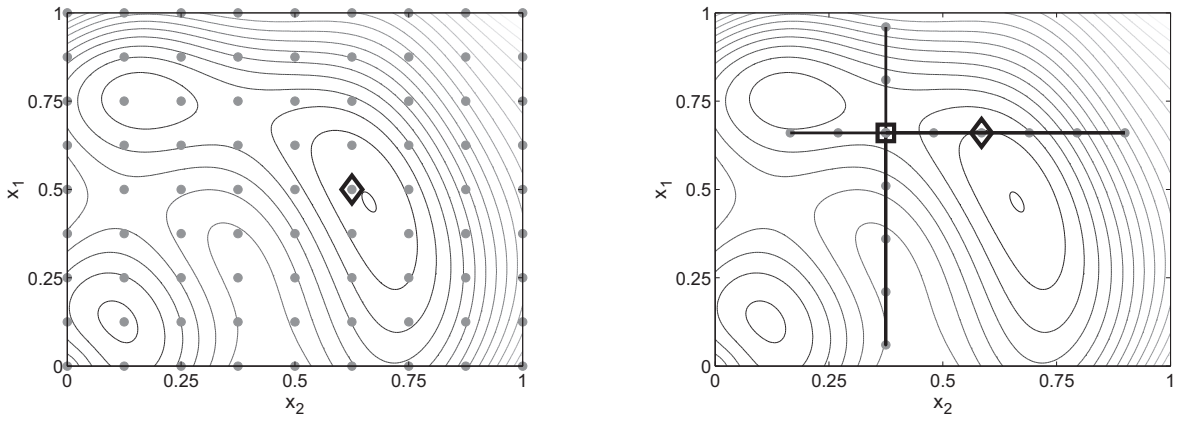
$$N = ND^d, \tag{1.2}$$

where ND is the number of measurements in each dimension and therefore determines the density of the grid. In figure 1.5 (a) a simple example of a full factorial optimization is given. In this plot a dense grid over a two-dimensional space is shown and the optimal point in this grid is marked. By increasing the density of the grid, it clearly can be seen that the optimal parameters can be determined with arbitrary accuracy. However, from (1.2) it follows that the number of measurements increases exponentially with the number of parameters. As the number of measurements is directly linked with time and costs on the test bed, this approach cannot be used for an optimization with many parameters.

1.2.1.2 One Factor at a Time

Another optimization method which was used for engine calibration is "One Factor at a Time", see [53] and [21]. At this approach only one adjustment parameter is manipulated at once, while all other parameters are fixed at a constant value. Figure 1.5 (b) shows a simple example of this method, where two parameters should be optimized. First, the parameter x_1 is manipulated while the parameter x_2 is fixed. Therefore, in this step the problem reduces to a one-dimensional optimization. After finding the optimum of this step, which is marked by a rectangle, the parameter x_1 is fixed and the parameter x_2 is manipulated.

Since this method cannot identify interactions of the different parameters, it clearly can be



(a) Full Factorial Optimization

(b) One Factor at a Time

Figure 1.5: Measurement-Based Optimization, see [21].

seen that this procedure has to be repeated numerous times in order to find the next local optimum. As the number of interactions increases, if the number of parameters increases, this method cannot be used for more complex optimization problems.

1.2.2 Model-Based Optimization

The engine test bench is an expensive system. In addition, measurements on the test stand are often time-intensive. Hence, in recent years a lot of effort has been made in order to reduce time and costs on the test bench. An important improvement constitutes the automation of the test bench, which allows an automated operation overnight and therefore increases the capacity utilization. Nevertheless, as the complexity in engine calibration is increasing rapidly in recent years, as said above, new methods have been developed and used.
Another major step in engine calibration was the introduction of model-based optimization, see [33], [76] and [101]. In this approach measurements from the test bench are taken in order to build black-box or gray-box models. This method has numerous advantages. Some of them are:

- The optimization of the adjustment parameters can be performed with these surrogate models instead of the real engine. Therefore, by using modern optimization routines, a lot of objectives and constraints can be considered. Further, if the engine is used in different vehicle types, then often different objectives have to be taken into account. Hence, different optimizations can be performed for each type of vehicle, without a multiple using of the test bench.
- As we will see soon, numerous different methods have been developed, which allow to reduce the number of measurements, without limitations in accuracy of the models. Hence, time and costs on the test bench can be reduced.
- By using computer aided visualization techniques for the models, engineers can gain a better understanding of the basic relationships of the problem. This allows the engineer to gain a well-founded knowledge of the engine in a short time.

Two different approaches for model-based optimization, offline and online optimization, can be distinguished.

1.2.2.1 Model-Based Offline Optimization

The model-based offline optimization is characterized by strict separation between the measurements on the test bench and the modeling on the PC. Figure 1.6 shows a schematic diagram of this method. In an initial step, an experimental design is planned. Various different techniques had been developed and used for this task, like optimal design of experiments or space filling designs, see [6, 54, 92]. After taking measurements on the test bench, a modeling of the desired parameters is performed. With these models, the optimal settings of the parameters can be found by numerical optimization. Afterwards, these optimal values are verified on

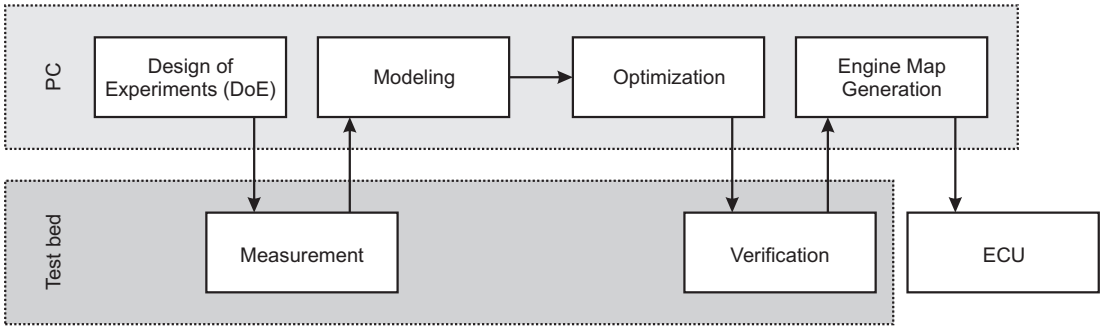


Figure 1.6: Model-Based Offline Optimization

the test bench. If the verification was successful, then the engine operation maps are generated and stored on the ECU.
A serious drawback of this approach is that the quality of the models is not checked during the measurements on the test bench. If the model quality is bad, e.g. if too few measurements are performed or too many outliers occur in the data, then the prediction of the optimal values will be wrong. Thus, the verification will fail and the optimization has to be started over again. This drawback can be overcome with the model-based online optimization.

1.2.2.2 Model-Based Online Optimization

In the last years, there is a trend to model-based online optimization, where measurement, modeling and optimization are not strictly separated (see [53] and the references therein). Hence, the modeling and optimization algorithms are in a permanent interaction with the test bench, which allows the models to give a feedback of their quality.
This has various advantages. First, the modeling can give a feedback if already enough measurements are taken and the measurement on the test bed can be stopped. Hence, the test bench time can be reduced to an optimal amount. Second, the models can provide information in which areas the measurements should be taken in order to achieve the maximum of information. Thus, the test bench time can be used more efficiently.
Hence, time and costs on the test bed can be considerably reduced by the usage of model-based online optimization [53].

As shown in the schematic diagram in figure 1.7, this approach is divided into four stages. In the initial stage, a start design is planned and measured on the test bench. This stage is equivalent to the first steps in model-based offline optimization and therefore often called offline DoE. With these first measurements, the initial models are calculated. Based on these models an online modeling is performed in the second stage. In this process, the goal is to improve the models, in order to assure that these models are able to represent the real engine behavior. Therefore, measurements are taken on the test bench at areas which reduce the model error at most, and after every measurement the models are updated. If the prediction of the models is accurate enough, then an optimization routine is performed in the third stage.

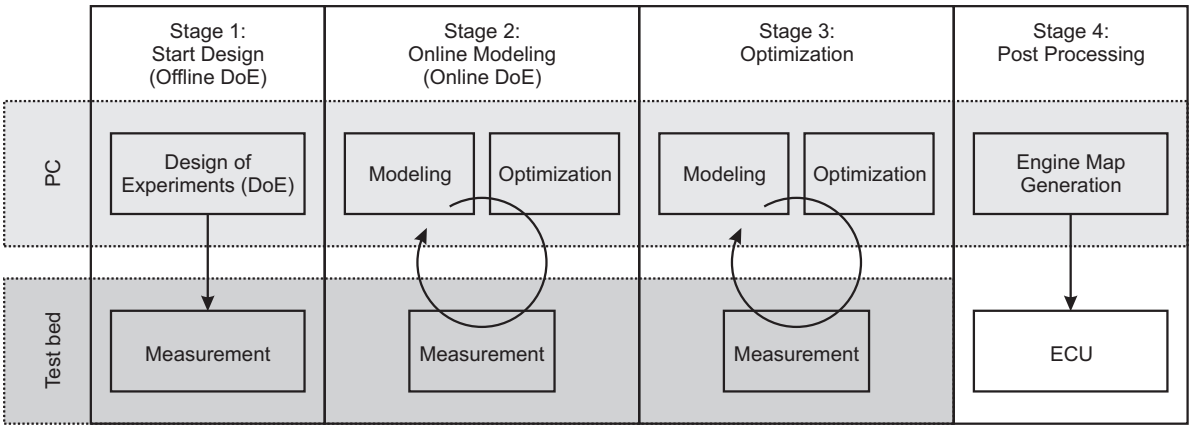


Figure 1.7: Model-Based Online Optimization

The aim of this task is to find the optimal settings for the adjustment parameters for every operation point and to take measurements near these optima. This has the advantage that the models are very precise in the optimal areas since the density of measurements is higher in these regions. At the end, the engine operation maps are generated and stored on the ECU.

1.2.2.3 Stationary and Dynamic Engine Calibration

For measurement data acquisition and modeling, stationary and dynamic approaches have been developed and used for offline and online optimization in engine calibration.

Stationary and Dynamic Measurement Data Acquisition
As said above, many calibration tasks are performed stationary, and therefore also the measurement is often performed stationary. Figure 1.8 illustrates the characteristics of such a stationary measurement. At first, the adjustment parameters are set on the desired values via automatic control in the control time (CT). These adjustments influence the measurement variables. Since some measurement variables have a low dynamic, e.g. temperatures of the engine, a stabilization time (ST) is waited, in order to reach a stationary state of the engine. During the averaging time (AT) the mean values of the measurement variables are calculated. Hence, the whole measurement time results from the sum of the stabilization time (ST) and the averaging time (AT). Depending on the dynamic of the considered measurement variables and on the noise on the measurements, the time for a single stationary measurement can add up to a few minutes. The measurements in this thesis were taken on state of the art test benches and took between 2 and 5 minutes for a single stationary measurement.

In order to avoid this time-intensive procedure, in recent years several non-stationary measurement techniques have been developed. Some of them are intended for a stationary modeling, like Sweeping [108] and Slow Dynamic Slope [52]. In these approaches, the measurements are performed in a way such that the dynamic behavior of the system is suppressed and the stationary values can be calculated [21].

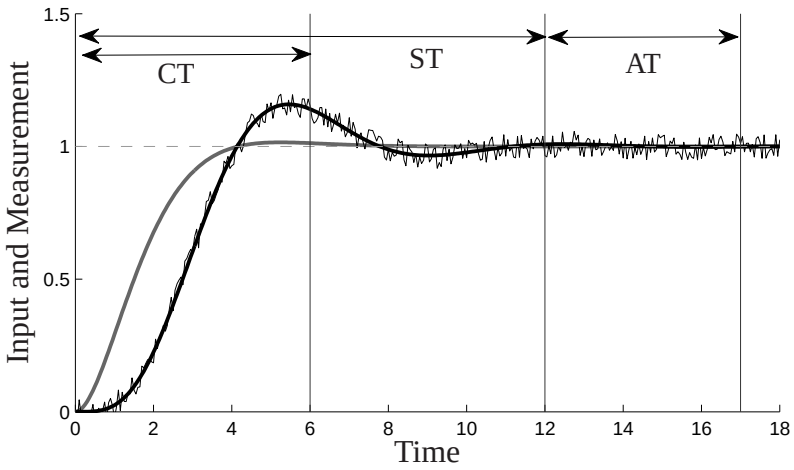


Figure 1.8: Stationary Measurement: Control (CT), Stabilization (ST) and Averaging Time (AT), [21].

Other approaches are intended for a dynamic modeling. Therefore, the dynamic behavior of the engine is measured by using sinus sweeps or APRBS signals (amplitude modulated pseudo-random binary signal) as inputs for the adjustment parameters, which should be regarded as a dynamic measurement in the following.

Stationary and Dynamic Modeling
As said above, nowadays in base calibration the adjustment parameters are typically optimized on the stationary state of the engine. In addition, until now many other functions on the ECU are only realized as stationary functions and in many cases an optimization of the transient state is not possible. Therefore, a stationary modeling is most commonly used in engine calibration.

A dynamic modeling has two advantages. First, a dynamic measurement can be used, which allows to save a considerable amount of time compared to a stationary measurement. Second, the dynamic behavior of the engine can be represented. Since e.g. a lot of emissions are generated in the transient states of the engine, a dynamic optimization of these states could be a great improvement, if the results of this optimization could be considered in future versions of the ECU. Hence, the dynamic modeling gained a lot of interest in recent years and a lot of research has been made, e.g. see [21, 34, 35, 70].

Discussion
Therefore, a dynamic modeling and optimization shows a great potential for the future. Nevertheless, this thesis focuses on stationary engine calibration because of two reasons.

First, the aim of this work was the development of a complete framework, which is able to cope with real demands resulting from practical applications. A typical and well known problem for a dynamical modeling arises, when it comes to the approximation of quantities, which are hard to measure. While a good dynamic modeling has already been performed for

NOx and consumption, a dynamical modeling of soot is still a tough task, e.g. see [102] and [35]. Nevertheless, the optimization of soot is an important aspect in engine calibration and cannot be neglected in practical applications. Other problems for a practical realization of dynamical approaches arise from time delays of emission measurements, which are varying over the operating range, and safety functions of the ECU [21]. Further, since a lot of effort from the user is needed for a dynamical modeling, the acceptance of these methods is still not as high as for stationary approaches.
Second, there are still many open questions in the area of stationary engine calibration, which are discussed in the following chapters. Further, a lot of results for stationary calibration are also applicable to dynamic calibration, like a comparison of different types for modeling. Therefore, the techniques in this thesis are regarded under a general viewpoint, wherever possible. Nevertheless, the development and application of methods for stationary engine calibration should be the focus in this thesis.

1.3 Scope of this Thesis

The aim of this thesis is the development of a complete framework for modeling and optimization for stationary base engine calibration. As mentioned above, there already exist a lot of approaches and algorithms for this topic. However, in the literature the different methods are often examined separately and a comprehensive overview of all different methods does not exist until now. Hence, so far it is hard to determine which approaches are most suitable for stationary engine calibration. Therefore, one aspect of this work is the evaluation and comparison of the different methods, the other aspect is the improvement of these methods, wherever possible.

The contributions of this thesis can be summarized as follows:

- **Extensive overview and comparison of different types of modeling in theory and practice (Chapter 3):** Various types of modeling have been developed and used for stationary engine calibration. Nevertheless, there exists no comparison which considers a comprehensive number of different techniques and simultaneously examines the theory extensively. Often, only a few approaches are considered, or just the practical performance on different data sets are compared. Hence, the determination which type of modeling is most suitable for stationary engine calibration is still an important open question. In this thesis an extensive theoretical model comparison is given, which can further be confirmed with practical examples. From this comprehensive study a recommendation for a most suitable modeling can be given.
- **A new outlier-robust modeling (Chapter 4):** A problem for state of the art algorithms for engine calibration arises, if outliers occur in the measurement data. It is shown in the overview of the different types of modeling, that outliers are not considered in recent approaches, and that they have to be removed before model training, in order to get a good model quality and an accurate prediction. This has serious drawbacks be-

cause usually a manual interaction is needed to identify the outliers, since an automatic detection of the outliers is not very robust or computationally very expensive, if there are many outliers in the data. Based on the results of the model comparison, the most suitable type of modeling is extended, in order to achieve a formulation, which is robust to outliers. In addition, a nonlinear transformation of the measurement data is integrated in the approach, in order to improve the reliability of this new modeling framework for engine calibration.

- **A new, improved model-based online optimization (Chapter 6):** Compared to model-based offline optimization, with a model-based online optimization time and costs on the test bench can be remarkably reduced. In addition, this approach assists the calibration engineers by providing models, which have a high accuracy in the optimal areas. Therefore, the calibration engineers do not need to verify the optimum and no additional process loops have to be performed, as this can be the case, if a model-based offline optimization is used and the verification of the optimum fails. However, only a few online techniques exist for stationary engine calibration, and all of them suffer from various drawbacks. Hence, a new approach is presented, and the performance of this method is illustrated.

In addition, at every time in this thesis the needs of the calibration engineers are considered in the approaches, which is rarely found in the literature. Hence, all developed concepts can be used in an automatic and robust way, in order that the users of these techniques are not challenged by complex mathematical issues, if the assumptions of the approaches are not precisely met. In this way, the calibration engineers are assisted by the developed tools, in order to increase the user acceptance of model-based techniques in engine calibration, which is still an important issue.
Therefore, many approaches in this thesis may include advanced mathematical concepts, however, this is only a challenge for the developers of the calibration tools and not for the users, since at any time these complex issues can be used in a robust way, without any required manual interaction.

All techniques in this thesis are presented from a general viewpoint, in order to draw additional conclusions for other fields of research. Nevertheless, it is assured that we never loose sight of the aims of stationary base engine calibration.

1.4 Structure of the Thesis

An overview of the basics of modeling for engine calibration is given in chapter 2. At first, some general theoretical preliminaries are discussed, which are used throughout the thesis. Afterwards, an overview of an extensive number of different modeling approaches is given.

The aim of chapter 3 is to identify the most suitable approach for stationary base engine calibration out of this overview. For this reason, the requirements on the modeling are examined, which result from the application of engine calibration. These requirements allow to compare

the different types of modeling with each other and enable to identify the most promising technique with theoretical and practical examinations.

However, all state of the art modeling approaches suffer from some shortcomings, which are discussed in chapter 4. Hence, based on the results of the model comparison, in chapter 4 the most suitable approach is further enhanced in order to overcome these drawbacks. An automatic nonlinear transformation of the measurements assures that the modeling assumptions on the data distributions are met, and a Student's-t noise assumption provides an outlier-robust model behavior.

In chapter 5 the state of the art techniques for optimization in engine calibration are discussed. Different algorithms for single- and multi-objective optimization are analyzed, and suitable methods are chosen for each application. In addition, different approaches for design of experiments for engine calibration and for model-based online optimization are examined.

However, since the state of the art online optimization procedures suffer from various drawbacks, which are discussed in chapter 6, a new approach is presented. Due to the use of a full probabilistic modeling and a multi-objective technique, an increased performance and usability of the online optimization can be obtained. This is demonstrated on various theoretical examples and practical applications.

In section 7 some short remarks are given on the implementation of the algorithms, the user interfaces and the connection to the test bench. The thesis is concluded with a summary of the results and possible future works.

Chapter 2

BASICS OF MODELING FOR ENGINE CALIBRATION

In engine calibration the aim of modeling is to approximate several different unknown engine functions $\Psi_i : X_{OP} \times X_{FP} \rightarrow \mathbb{R}$, $i \in \{1...d_{Obj}\}$, by known functions $y_i : X_{OP} \times X_{FP} \rightarrow \mathbb{R}$, which are referred to as models of the unknown functions Ψ_i.

It generally can be distinguished between a physical modeling, which is often called a white box modeling, an empirical modeling out of data, which is called black box modeling, and a mixture of both types, where some prior knowledge can be used and integrated in the modeling and the remaining modeling is performed on measurement data, which is called gray box modeling.
Since the internal combustion is a complex process, which is influenced by thermodynamics, fluid dynamics and chemistry, a precise modeling and simulation of this process is not practicable today. Even if the computing power will still increase exponentially in the future, it will take decades until an accurate combustion simulation is possible in a reasonable amount of time [45, 78, 130]. Hence, nowadays only very simplified physical models of the combustion are used for simulation in engine calibration. Further, in engine calibration physical models are used in domains, where the combustion does not need to be considered.
In base engine calibration many effects of the combustion can't be neglected. A typical example is the modeling of different emissions, e.g. soot, with respect to many different adjustment parameters, e.g. input pressure. Obviously, in this example the fluid flow and the chemical reactions in the cylinder have a major influence on the formation of soot and can't be neglected. However, as said above, a detailed physical modeling and numerical simulation is not practicable today. Therefore, in base engine calibration measurements are taken from the test bench and with this data a black box or a gray box modeling is performed.

A set of measurements $\mathcal{D} := \{(\mathbf{x}_n, t_n)|n \in \{1...N\}\}$, where N is the number of measurements, contains the values of the adjustment parameters $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$, which are called inputs, and the measured values $\mathbf{t} = (t_1, ..., t_N)^T$, which are often called targets. The calculation of a black box model is called training, whereas the evaluation of the model is referred to as prediction. Since the goal is to model the behavior of an unknown system from measured data, the term modeling corresponds to the term system identification and will be used interchangeably in the following.

When an empirical black box modeling is used, it can be distinguished between an interpolation model and a regression model.
Since measurements, which are taken from the test bench, always contain a certain amount of noise, in engine calibration an important goal of the modeling is to suppress the dependency of the noise on the model. Therefore, only regression models are considered for engine calibration [92].

Further, it can be distinguished between a parametric and a non-parametric modeling. For the parametric models the model structure is specified in advance, and after the training of the model parameters, the measurement data $\mathcal{D}$ can be discarded. This is contrary to non-parametric types of modeling, where the model structure is not specified a priori and the prediction of the models relies on the training data $\mathcal{D}$, or a subset of it.

In this chapter the theoretical preliminaries and an overview of different types for modeling are given in an abbreviated version. For a more detailed overview see [11], [84] or [41].

2.1 Preliminaries

From the discussion above, it follows that our measurements t_n are given by

$$t_n = y_n + \epsilon_n \tag{2.1}$$

where $y_n = y(\mathbf{x}_n)$ is a model of a function of the engine and ϵ_n is a random noise variable whose value is chosen independently and identically distributed (i.i.d.) for each observation n.

In this section some techniques and general theoretical properties for modeling are derived, which are used in the further thesis.
The first section introduces the common method of maximum likelihood and shows how a normal noise assumption leads to the method of least squares. The second section discusses how accurate a modeling can become in average, with a finite amount of training data, through the bias variance dilemma. Further, two important sources of error are shown: overfitting and underfitting. In addition, some techniques for choosing a good compromise for the bias variance trade-off are discussed.

2.1.1 Maximum Likelihood, Normal Noise and Least Squares

Typically, the model y contains parameters $\mathbf{\Theta}$, which can be tuned on the training data. Two common approaches exist for the determination of suitable parameters: the maximum likelihood method and the maximum a posteriori approach. In this section the maximum likelihood approach is considered.

If ϵ in (2.1) is an i.i.d. noise, then from (2.1) it follows that the measurements t_n are given by a certain probability density function (pdf) p_f. The joint probability $p(\mathbf{t}|\mathbf{y}, \mathbf{X}, \mathbf{\Theta})$ of the n

measurements $t_1, ..., t_n$ for a given value for $\mathbf{\Theta}$ is then given by

$$p(\mathbf{t}|\mathbf{y}, \mathbf{X}, \mathbf{\Theta}) = p_f(t_1|y_1, \mathbf{x}_1, \mathbf{\Theta}) \cdot p_f(t_2|y_2, \mathbf{x}_2, \mathbf{\Theta}) \cdot ... \cdot p_f(t_N|y_N, \mathbf{x}_N, \mathbf{\Theta}). \tag{2.2}$$

In the framework of the maximum likelihood approach the measurements $\mathbf{t}$ are considered to be fixed and the likelihood function

$$\mathcal{L}(\mathbf{\Theta}|\mathcal{D}) := p(\mathbf{t}|\mathbf{y}, \mathbf{X}, \mathbf{\Theta}) = \prod_{i=1}^{N} p_f(t_i|y_i, \mathbf{x}_i, \mathbf{\Theta}) \tag{2.3}$$

is then maximized with respect to the model parameters $\mathbf{\Theta}$, where $\mathcal{D}$ refers to the set of measurements (see above).

Example: normally distributed noise
Now an independent zero mean Gaussian distribution with variance σ^2 is assumed for the noise, so that $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$. Hence, from (2.1) it follows that the probability distribution of a single measurement t_n is given by the normal distribution

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(t_n - y_n)^2}{2\sigma^2}\right), \tag{2.4}$$

where $y_n = y(\mathbf{x}_n, \mathbf{\Theta})$. In the same way as above, we can derive the joint distribution of our measurements

$$\begin{aligned} p(\mathbf{t}|\mathbf{y}, \mathbf{\Theta}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \sigma^2\mathbf{I}) &= \prod_{n=1}^{N} p(t_n|y_n) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(t_n - y_n)^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{\sum_{n=1}^{N}(t_n - y(\mathbf{x}_n, \mathbf{\Theta}))^2}{2\sigma^2}\right), \end{aligned} \tag{2.5}$$

which is simply the product of the probability distributions of the single measurements. For an easier understanding partially the dependence on $\mathbf{\Theta}$ and $\mathbf{X}$ was neglected.

As said above, in the framework of maximum likelihood the model parameters $\mathbf{\Theta}$ are chosen in a way which maximizes the joint distribution (2.5). Obviously, the joint distribution is maximized as the negative exponent in (2.5) is minimized. Hence the maximum likelihood solution for an independent Gaussian noise assumption is given by minimizing

$$\text{SSE}(\mathbf{\Theta}) := \sum_{n=1}^{N}(t_n - y(\mathbf{x}_n, \mathbf{\Theta}))^2 \tag{2.6}$$

which is often called the sum of squares error function (SSE) or the method of least squares.

Hence, it can be seen that the minimization of the sum of squares error function is equivalent to the maximization of the likelihood function under an independent Gaussian noise distribution. In other words, the calculation of the model parameters $\mathbf{\Theta}$ via the least squares approach leads to the same results as via the maximum likelihood approach with a Gaussian noise assumption.

2.1.2 The Bias-Variance Dilemma and Overfitting

In this section some basic problems of modeling are discussed, which appear when data sets of *limited* size are used, which is always the case in practical applications. Clearly, if we had an unlimited amount of data, then we can approximate an unknown function with arbitrary accuracy. However, in practice we have a data set $\mathcal{D}$ containing only a finite number N of data points, and therefore there will always be some error left.

The discussion is started by the examination of the bias variance dilemma. For this topic we consider a system behavior y_S, which we want to approximate with a model $y_M = y_M(\mathbf{x}, \mathcal{D})$. As said above, we cannot observe the system directly, but we can observe measurements t_n, which are shifted by random noise ϵ given by (2.1). Now we want to decompose the expectation of the squared loss function

$$\begin{aligned}\mathbb{E}\left[e^2\right] &= \mathbb{E}\left[(t_n - y_M)^2\right] = \mathbb{E}\left[(y_S + \epsilon - y_M)^2\right] \\ &= \mathbb{E}\left[(y_S - y_M)^2\right] + \mathbb{E}\left[\epsilon^2\right] + 2\mathbb{E}\left[\epsilon(y_S - y_M)\right] \\ &= \mathbb{E}\left[(y_S - y_M)^2\right] + \mathbb{E}\left[\epsilon^2\right], \end{aligned} \tag{2.7}$$

since ϵ is uncorrelated with y_S and y_M. We can further decompose the model error

$$\begin{aligned}\underbrace{\mathbb{E}\left[(y_S - y_M)^2\right]}_{(\text{model error})^2} &= \mathbb{E}\left[(y_S - \mathbb{E}[y_M] - (y_M - \mathbb{E}[y_M]))^2\right] \\ &= \mathbb{E}\left[(y_S - \mathbb{E}[y_M])^2\right] + \mathbb{E}\left[(y_M - \mathbb{E}[y_M])^2\right] \\ &\quad - 2\mathbb{E}\left[(y_S - \mathbb{E}[y_M])(y_M - \mathbb{E}[y_M])\right] \\ &= \mathbb{E}\left[(y_S - \mathbb{E}[y_M])^2\right] + \mathbb{E}\left[(y_M - \mathbb{E}[y_M])^2\right] \\ &= \underbrace{(y_S - \mathbb{E}[y_M])^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}\left[(y_M - \mathbb{E}[y_M])^2\right]}_{\text{variance}}. \end{aligned} \tag{2.8}$$

We see that the expected squared difference between the real system behavior y_S and our model y_M can be expressed as the sum of two terms.
The first term, which is called the squared bias, describes how the real behavior differs from the prediction of our model averaged over all data sets. This systematic error can be influenced by the flexibility of our model. If the flexibility of our model is increased (e.g. by increasing the degree of our polynomial model, or by increasing the number of neurons of our neural network), the bias error will approach to zero, if we use an universal approximator for modeling [84]. Therefore, one ambition is that the model should have many parameters in order to be flexible enough to approximate every nonlinear engine mapping.
The second term, which is called the variance, describes how the model of the individual data sets vary around their average, and hence indicates how the model y_M is sensitive to the particular choice of data set. This random error can be influenced by the flexibility of our model as well. If the flexibility of our model is decreased, the model will not vary around as much and therefore the variance error will decrease [64]. Therefore, another ambition is that the model should have few parameters in order to be not too flexible.

Hence, we can summarize this examination.
A very simple model with few parameters will have a small variance error, but simultaneously it will not be able to adapt the real behavior very well and therefore it will have a large bias, which is called *underfitting*. A very complex model with many parameters will have a small bias error, but simultaneously the solution will strongly adapt random noise instead of the real behavior and therefore it will have a large variance error, which is called *overfitting*.

Therefore, a major task in system identification is the determination of an *optimal* flexibility, which is a good compromise between the bias and variance error.
Various techniques have been developed for this problem. In the next sections an overview is given.

2.1.2.1 Maximum A Posteriori and Regularization

A common approach for parametric models is to estimate the parameters $\boldsymbol{\Theta}$ not only via the likelihood function, but also to incorporate a penalty term, which reduces the flexibility of the model. This is called regularization. Typical approaches choose the model for which the quantity

$$f(\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})) - g(\boldsymbol{\Theta}, \mathcal{D}) \tag{2.9}$$

is largest. Here f and g are functions which depend on the different approaches and $\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$ is the likelihood function (2.3). Hence, these approaches consist of two terms. The likelihood term $f(\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}))$ typically gets larger if the flexibility of the model is increased, whereas the penalty term $-g(\boldsymbol{\Theta}, \mathcal{D})$ typically tries to reduce the flexibility. The goal is now to find suitable functions f and g, which allow to determine an optimal degree for the flexibility of the model. Various different approaches have been developed, like the Akaike information criterion (AIC) [3] or the Bayesian information criterion (BIC) [109]. In this thesis, we will examine a method called ridge regression, which is very common in machine learning. But instead of fixing the functions f and g in advance, we will see that the complexity penalty term arises naturally, if we use a full Bayesian approach.

In the Bayesian viewpoint we assume that an a-priori distribution $p(\boldsymbol{\Theta})$ for our model parameters $\boldsymbol{\Theta}$ is given. Then we use the likelihood function $\mathcal{L}(\mathcal{D}|\boldsymbol{\Theta})$ and the Bayes theorem to calculate the a posteriori distribution

$$p(\boldsymbol{\Theta}|\mathcal{D}) \propto p(\boldsymbol{\Theta})\mathcal{L}(\mathcal{D}|\boldsymbol{\Theta}). \tag{2.10}$$

Now we can choose the model parameters at the mode of this posterior distribution, which is called a maximum a posteriori (MAP) approach. Hence, we choose

$$\hat{\boldsymbol{\Theta}}_{MAP} = \arg\max_{\boldsymbol{\Theta}} p(\boldsymbol{\Theta}|\mathcal{D}) = \arg\max_{\boldsymbol{\Theta}} p(\boldsymbol{\Theta})\mathcal{L}(\mathcal{D}|\boldsymbol{\Theta}). \tag{2.11}$$

Clearly, if we use a non-informative prior for $p(\boldsymbol{\Theta})$, like a constant function (we should be aware that this would be no probability distribution anymore), then the maximum a posteriori estimation is identically with the maximum likelihood estimation.

Example: normally distributed noise and parameters
As in the example above, an i.i.d. Gaussian distribution with variance σ^2 is assumed for the noise. Further, we assume that our model parameters underlie a prior distribution. For simplicity, in this example a Gaussian distribution of the form

$$p(\boldsymbol{\Theta}) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}\right) \tag{2.12}$$

should be considered, where α is the precision of the distribution and M is the number of model parameters. With the likelihood function (2.5) for the normal noise distribution, the posterior distribution can be calculated from (2.10) as

$$p(\boldsymbol{\Theta}|\mathcal{D}) \propto \exp\left(-\frac{\sum_{n=1}^{N}(t_n - y(\mathbf{x}_n, \boldsymbol{\Theta}))^2}{2\sigma^2} - \frac{\alpha}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}\right). \tag{2.13}$$

Obviously, seeking the mode of this probability function is equivalent to the minimization of

$$\frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(\mathbf{x}_n, \boldsymbol{\Theta}))^2 + \frac{\alpha}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}, \tag{2.14}$$

where $\beta = 1/\sigma^2$, which is again equivalent to the minimization of the regularized sum of squares error function (RSSE)

$$\text{RSSE}(\boldsymbol{\Theta}) := \sum_{n=1}^{N}(t_n - y(\mathbf{x}_n, \boldsymbol{\Theta}))^2 + \lambda\|\boldsymbol{\Theta}\|^2, \tag{2.15}$$

where $\lambda = \alpha/\beta$. This result should be compared with (2.9). We will see shortly, that the flexibility of the model can be reduced as the parameter λ is increased and vice versa. Hence, this parameter controls the effective complexity of the model.
In statistics the approach (2.15) is called ridge regression, in the area of neural networks it is known as weight decay, and it is very common in machine learning as well as in modeling for engine calibration.

Example: simple polynomial regression
Now the most important key concepts of this section should be demonstrated by a simple polynomial regression. Consider a simple 1-D polynomial model

$$y(x) = \sum_{j=0}^{d_o} a_j x^j \tag{2.16}$$

is trained by some data, as shown in figure 2.1. In this figure training data (circles) is sampled from a function and shifted by normal noise. In practice this function would be an unknown nonlinear engine mapping. With this data a polynomial regression is performed.
In the left plot polynomials of low $(d_o = 1)$, medium $(d_o = 3)$ and high $(d_o = 10)$ orders

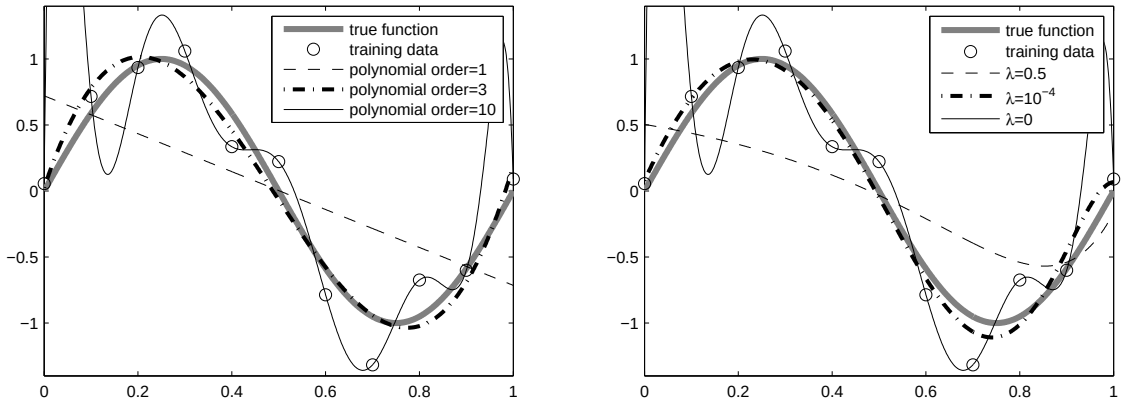


Figure 2.1: Example of a simple 1-D polynomial regression. Left: the order of the polynomial is varied. Right: A polynomial of order 10 is used and a penalty term is introduced and varied.

are used, and the training is performed by using the maximum likelihood method via the SSE function (2.6). It clearly can be seen that the low-order polynomial is not able to adapt the nonlinearity of the function which should be approximated. This model has a very low flexibility, a large bias and is underfitted. In comparison to that, the higher order polynomial is highly oscillating around the function which should be approximated. This model has a very high flexibility, a large variance error and is overfitted. The third order polynomial is a good compromise for the bias variance trade-off.
In the right plot a polynomial of order 10 is used and the training is performed by using the maximum a posteriori method via the RSSE function (2.15) with different values for λ. It clearly can be seen that an increasing value for λ reduces the flexibility of the model, and that the performance of this approach is similar to the reduction of the order of the polynomial in the left plot.
Further, this example should demonstrate the importance of choosing an optimal flexibility for the modeling.

2.1.2.2 Stepwise Regression

Another technique for finding the optimal flexibility of the model is stepwise regression. Compared to regularization, where the number of model parameters $\boldsymbol{\Theta}$ are fixed in advance, in the stepwise regression approach the model parameters are selected from a set of admissible regressors during the training. It can be distinguished between forward selection, backward elimination and stepwise selection techniques.

In forward selection approaches, which are often called growing in the area of neural networks, the modeling is typically started with a simple structure and only a few model parameters. Then, the performance of the other model parameters from the admissible set of regressors is evaluated, e.g. through a statistical test or something similar, and the most suit-

able parameter is added to the model. This is repeated until an abort criterion determines that the flexibility of the model is sufficient.
The backward elimination approaches, which are often called pruning in the area of neural networks, pursue the contrary procedure. First, it is started with a very flexible model, and then the flexibility of the model is reduced, by removing parameters from the model, until an abort criterion is reached.
The stepwise selection approach is a combination of forward selection and backward elimination. At each iteration, before forward selection is performed and a new parameter is added to the model, all already selected parameters undergo some statistical significance test, and those regarded as insignificant are removed from the model. Clearly, stepwise selection is computationally more expensive, but can also give better results than forward selection or backward elimination alone [84].

2.1.2.3 Cross-Validation and Early Stopping

Another technique for finding the optimal flexibility of the model is early stopping, which is uniquely used for neural networks. In order to understand this approach, at first some principles, which lead to method of cross-validation, have to be discussed.

As we have seen above, overfitting can occur if the model is trained via the SSE function (2.6). Hence, the model performs well on the data on which it is trained, since the SSE function is minimized, but it will perform poorly on the other areas in the input space. Hence, a simple approach to estimate the quality of a model is to train it on a training data set and evaluate its performance on a different data set. The performance on this different data set is called generalization.
For this method the measurement data has to be split up into separate parts. This can cause problems if the number of measurements is small, since not all areas of the input space will be covered from all separate data sets. As said above, the engine test bench is an expensive system, and therefore as few measurements as possible should be taken. Hence, in this thesis a method called cross-validation is considered, which tries to minimize the impact of sparse data. This approach is illustrated in figure 2.2.
For cross-validation the data is partitioned into S disjunct parts. Then $S-1$ parts are used to train the model, and the remaining part is used for validation. This procedure is then repeated for all S possible combinations. After that, the validation errors from all runs are averaged to obtain a reliable estimate for the model performance. If the data is very scarce, then often $S = N$ is chosen, where N is the number of measurements, which is called the leave-one-out technique.
The advantage of cross-validation is the possibility to use a large data set for the training in each run, while at the same time every data point is used for validation during all runs. A major drawback of this technique is that the model training is performed S times, and this can be problematic for models in which the training is itself computationally expensive.

Now the method of early stopping can be discussed.

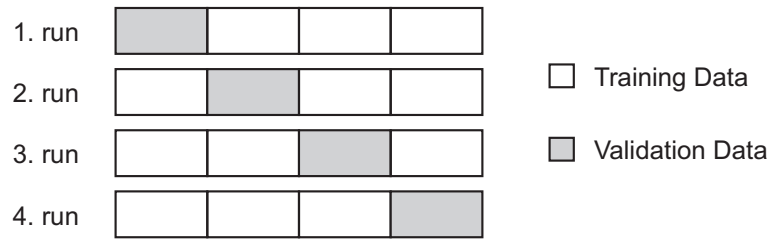


Figure 2.2: S-fold cross-validation, illustrated for $S = 4$, [11].

The training of a neural network is an iterative optimization, which we will see later, where typically the SSE function (2.6) of the training data is minimized. If the error of a validation set is measured during the training, then this error often shows a decrease at first, followed by an increase in the later optimization steps. As we saw above, if the model is overfitted, then the model performs well on the training data, but poor on the validation data. In contrast, if the model is underfitted, then the model performs poor on both, the training data and the validation data. Hence, the method of early stopping ends the optimization routine when the validation error reaches its minimum.
A major drawback of early stopping is that a validation set has to be used, and therefore not all data can be used for training, which is problematic since the measurement data is scarce. This drawback can slightly be reduced if the cross-validation approach is used. Nevertheless, since the training for the neural network is computationally expensive, cross-validation is hardly applicable. In addition, [92] has shown that the regularization technique (2.15) performs better in the area of engine calibration than early stopping, and therefore early stopping won't be considered in the further thesis.

2.2 Linear Regression Models

In this chapter we consider models, which are linear combinations of fixed functions $\phi_j : \mathbb{R}^D \to \mathbb{R}$ of the input variables $\mathbf{x} = (x_1, ..., x_D)$, and therefore can be written as

$$y(\mathbf{x}, \boldsymbol{\Theta}) = \sum_{j=1}^{M} \theta_j \phi_j(\mathbf{x}) = \boldsymbol{\Theta}^T \boldsymbol{\phi}(\mathbf{x}) \tag{2.17}$$

where $\boldsymbol{\Theta} = (\theta_1, ..., \theta_M)^T$ and $\boldsymbol{\phi} = (\phi_1, ..., \phi_M)^T$. It should be mentioned that the notation 'linear modeling' does not mean that this class of models is linear in the inputs $\mathbf{x}$, but rather these models are linear in its model parameters $\boldsymbol{\Theta}$. Therefore, this class of models shares simple analytical properties and yet can be nonlinear with respect to the input variables. The nonlinear functions $\phi_j(\mathbf{x})$ are often called basis functions.

One advantage of linear modeling (2.17) is, that we can directly obtain a closed-form solution for the minimization of the SSE function (2.6) and the RSSE function (2.15). For this solution we first have to introduce the matrix $\boldsymbol{\Phi}$ of size $N \times M$, called the design matrix, whose

elements are given by $(\mathbf{\Phi})_{n,j} = \phi_j(\mathbf{x}_n)$, so that

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_M(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_M(\mathbf{x}_N) \end{pmatrix}. \tag{2.18}$$

If the number of measurements N is greater than the number of model parameters M, then for linear modeling the model parameters $\mathbf{\Theta}$ which minimize (2.6) are given by

$$\mathbf{\Theta}_{ML} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t} = \mathbf{\Phi}^\dagger\mathbf{t}\,, \tag{2.19}$$

where $\mathbf{\Phi}^\dagger := \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}$ is the Moore-Penrose pseudo-inverse of the matrix $\mathbf{\Phi}$ and the subscript ML refers to the maximum likelihood solution. The derivation of this solution is straightforward and can be found in e.g. [11]. It can be obtained by calculating the gradient of (2.6) and setting it to zero.
The model parameters $\mathbf{\Theta}$ which minimize (2.15) can be derived in an analogous way, and the result is given by

$$\mathbf{\Theta}_{MAP} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t}\,, \tag{2.20}$$

where the subscript MAP refers to the maximum a posteriori solution. It clearly can be seen that for the limit $\lambda \to 0$ the ML solution (2.19) can be obtained.

2.2.1 Polynomial Regression

If the basis functions ϕ_j are chosen to be

$$\phi_j(\mathbf{x}) = \mathbf{x}^j, \tag{2.21}$$

where $j = (j_1, ..., j_D)$ is a D-dimensional multi-index and $\mathbf{x}^j = x_1^{j_1}\cdot\ldots\cdot x_D^{j_D}$, then a polynomial regression is performed. With these basis functions and from (2.17), the polynomial model is given by

$$y(\mathbf{x}, \mathbf{\Theta}) = \sum_{|j|\leq\gamma} \theta_j \mathbf{x}^j \tag{2.22}$$

where $|j| = j_1 + ... + j_D$. Here, γ is called the degree or order of the polynomial. If e.g. $\gamma = 1$ is chosen, then a linear regression is performed.

Due to the simplicity of the polynomial model, it was the first modeling technique which was used in engine calibration [33, 77]. Today, this approach is still the most commonly used type of modeling and offered in every common commercial product for stationary base engine calibration, like in the PAoptimizer [31] from KRATZER AUTOMATION AG, in ASCMO [58, 59, 100] from ETAS, in the AVL CAMEO Tool [33], in the Easy-DoE Toolsuite [44] from IAV, in the Model-Based Calibration Toolbox [104, 105, 118] from MathWorks and in the Intelligent Calibration Tool [89] from Kristl, Seibt & Co GmbH and Magna Powertrain.

2.2.1.1 Polynomial Stepwise Regression

A common method to obtain an optimal flexibility of the polynomial model is to perform a polynomial stepwise regression, which was previously described in section 2.1.2.2 under a general viewpoint. This technique is most commonly used in engine calibration, in order to obtain a good compromise for the bias variance trade-off for polynomial models.
In this thesis we will concentrate on the t-test for stepwise regression, which can be found in [92] and [26]. As we will see later, due to the general drawbacks of polynomial modeling, other types of modeling will be recommended. Therefore, the specific type of hypothesis test will not be of great interest in this thesis, and we will only need it as a comparison to other types of modeling. Hence, it will only be discussed in a very short form.

For the hypothesis test we assume that the measurement noise is i.i.d. normally distributed, more formally: $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_N)$. If we denote $\boldsymbol{\Theta}$ as the true model parameters and $\hat{\boldsymbol{\Theta}}$ as the estimated model parameters, then from this assumption it follows that our estimated model parameters are given by the following normal distribution [26]

$$\hat{\boldsymbol{\Theta}} \sim \mathcal{N}\left(\boldsymbol{\Theta}, \sigma^2 (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}\right). \tag{2.23}$$

With this result we can now perform a hypothesis test in which it is tested, if a certain model parameter $\hat{\theta}_j$ should be removed from the whole set of model parameters $\hat{\boldsymbol{\Theta}}$. For this test we have to evaluate [92]

$$t_j = \frac{\hat{\theta}_j}{\sqrt{\frac{1}{N-M} ||\mathbf{t} - \boldsymbol{\Phi}\hat{\boldsymbol{\Theta}}||^2 \left[(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1}\right]_{jj}}}. \tag{2.24}$$

The hypothesis, that the coefficient $\hat{\theta}_j$ can be removed from the model, can be discarded if [26]

$$|t_j| > t_{1-\tilde{\alpha}, N-M}\,, \tag{2.25}$$

where $t_{1-\tilde{\alpha}, N-M}$ is the $(1-\tilde{\alpha})$ quantile of the Student's t-distribution with $(N-M)$ degrees of freedom, which is discussed at length in chapter 4.

With this hypothesis test forward selection can be performed by testing the fitness of a single parameter θ_j, which is not yet integrated in the set of model parameters $\hat{\boldsymbol{\Theta}}$, through adding this single parameter temporarily to the model. If the significance $|t_j|$ is greater than a predefined value $t_{1-\tilde{\alpha}_1, N-M}$, then this parameter can be integrated in the model. Further, also backward elimination can be performed by removing the model parameters which have a lower significance than a predefined value $t_{1-\tilde{\alpha}_2, N-M}$ in every training step.
As in [92], the stepwise selection algorithm was enhanced by a QR decomposition, in order to make the algorithm numerically more stable and to reduce the computational effort.

2.2.2 RBF Networks

Various different kinds of radial basis function (RBF) networks exist. A very common choice for the basis functions is

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}||\mathbf{x}-\boldsymbol{\mu}_j||^2_{\boldsymbol{\Sigma}_j}\right) \tag{2.26}$$

with

$$||\mathbf{x}-\boldsymbol{\mu}_j||_{\boldsymbol{\Sigma}_j} = \sqrt{(\mathbf{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}_j(\mathbf{x}-\boldsymbol{\mu}_j)} \tag{2.27}$$

where $\boldsymbol{\Sigma}_j$ is a positive definite norm matrix [84]. With this definition of the RBF network, a schematic structure of (2.26) and (2.17) can be drawn, which is illustrated in figure 2.3. This structure shows the membership of the RBF networks to the class of artificial neural networks (ANN). The inputs are connected to M radial basis functions (2.26), which are regarded as a hidden layer of M neurons in the nomenclature of neural networks. Typically, RBF networks consider only one output at a time. Therefore, the output layer consists only of one linear summation (2.17), which is regarded as a linear output layer.
It can be seen that the model (2.17) is linear in the parameters $\boldsymbol{\Theta}$, but nonlinear in the parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$. A generalization of this approach can be derived from the Gaussian process viewpoint, which will be discussed extensively in section 2.4.2, where the training is performed by simultaneous optimization of all the parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, $j \in \{1, ..., M\}$.

In engine calibration it is common not to optimize all the parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ at the same time, but rather in an iterative procedure [34, 92, 118]. In base engine calibration $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are often determined in a first step, and in a second training step the parameters $\boldsymbol{\Theta}$ are determined by the maximum likelihood solution (2.19) [34]. Often, these two steps are repeated alternately, in order to achieve a better solution [118].
It should be noted that the classification of the RBF networks to the class of linear models refers to the determination of the parameters $\boldsymbol{\Theta}$, which is common in engine calibration [92].

Numerous different approaches exist for the determination of the parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$. Two techniques should be considered here, which are conventional in engine calibration [92]. In the first technique the training data in the input space is clustered, and a basis function is placed in the areas where the density of training data is high, which is a method of unsupervised learning [84]. The other technique places the basis function in areas, where a systematic difference of the data and the model can be detected.

Other types of RBF networks can be developed by choosing normalized basis functions, which lead to the general regression neural networks (GRNN) [84] and the Nadaraya-Watson model [11]. At last, it should be mentioned that the RBF network is an universal approximator on a compact subset of $\mathbb{R}^D$. This means that a RBF network with an increasing number of hidden neurons can approximate any continuous function with arbitrary precision.

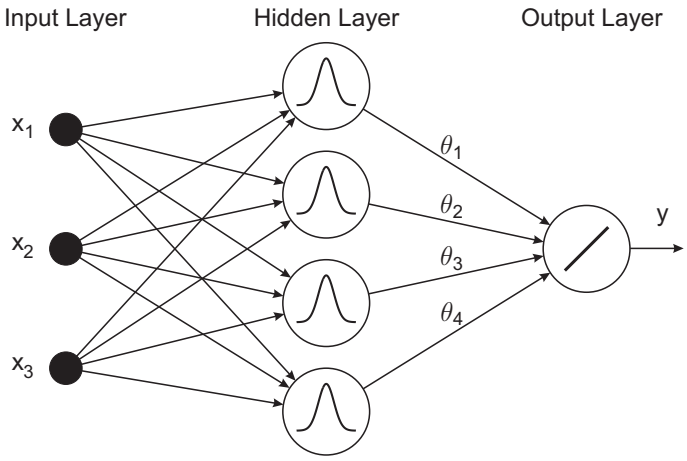


Figure 2.3: Schematic structure of the RBF network.

2.2.3 The LLR Model

The LLR model (Linear model with Local RBF terms) was introduced by [92].
In [92] it is argued that a polynomial model is able to approximate the global behavior of a function, however, difficulties arise when it comes to adaptation of local behavior. At the same time it is stated that RBF models are good at approximation of local properties, but they cannot adapt the global behavior well.
Hence, [92] introduced the LLR model as a combination of a polynomial model and a RBF model.

The training algorithm of the LLR model is given as follows:
At first, the training data in the input space is clustered, and RBF terms are placed in areas, where the density of training data is high. Then an iterative procedure is performed. Initially, a polynomial stepwise regression is performed. The RBF terms, which are already included in the model, are considered and integrated in the regression. If the stepwise regression has converged, then RBF terms are placed in areas, where a systematic difference of the data and the model can be detected. After the inclusion of the RBF terms, the polynomial stepwise regression is performed again, since the significances of the polynomial terms are changed during the RBF training. Afterwards, more RBF terms can be added to the model. This iterative procedure is terminated, if the modeling has converged, or if a predefined maximum number of training cycles has been reached.

2.3 Local Linear Models

Polynomial regression suffers from some drawbacks, which are discussed at length in section 3.2. However, as these linear models also have many advantages, more sophisticated approaches were developed and used, which try to minimize the disadvantages of the polynomials.

One approach is local linear modeling, which is very common in engine calibration. This method divides the whole input space in many subspaces, and then a linear modeling is performed in every single subspace. Usually a polynomial modeling of low order is chosen for the linear models in engine calibration, but in principle also other types of linear models would be possible.

If the intersections between the local models are fuzzy, then the modeling is also denoted as a neuro fuzzy modeling. The Takagi-Sugeno fuzzy models [117] are the most common types of neuro fuzzy models. In these approaches, rules or submodels are used for approximation. The rules R_i of a Takagi-Sugeno model are dependent on the model inputs $\mathbf{x}$ and have the following form

$$\begin{aligned} R_i : &\text{ if } (x_1 \text{ is } A_{i,1}) \text{ and ... and } (x_D \text{ is } A_{i,D}) \\ &\text{then } y_i = \sum_{j=1}^{M_i} \theta_{i,j}\phi_{i,j}(\mathbf{x}) = \boldsymbol{\Theta}_i^T \boldsymbol{\phi}_i(\mathbf{x}) \end{aligned} \tag{2.28}$$

where $\phi_{i,j}$ refers to the $j-$th basis function (see section 2.2) of the $i-$th rule or submodel, and $A_{i,j}$ refers to the $j-$th dimension of the subspace A_i in which the $i-$th rule is valid. Every subspace A_i is defined by its center and neighborhood. In engine calibration the influence $\tilde{\Lambda}_i$ of a local model is often specified through a Gaussian function with center $\boldsymbol{\mu}_i$ and standard deviation $\boldsymbol{\sigma}_i$, which can be different in each input dimension, so that

$$\tilde{\Lambda}_i(\mathbf{x}) = \exp\left(-\frac{1}{2}\sum_{j=1}^{D}\frac{(x_j - \mu_{i,j})}{\sigma_{i,j}^2}\right). \tag{2.29}$$

In order that the Gaussian functions sum to 1 for any $\mathbf{x}$, it is required to normalize them. The so-called membership functions Λ_i are determined by

$$\Lambda_i(\mathbf{x}) = \frac{\tilde{\Lambda}_i(\mathbf{x})}{\sum_{i=1}^{M_S}\tilde{\Lambda}_i(\mathbf{x})} \quad , \quad \sum_{i=1}^{M_S}\Lambda_i(\mathbf{x}) = 1 \tag{2.30}$$

where M_S refers to the number of rules or submodels. In order to evaluate the model output y, the outputs y_i of all submodels are weighted with the membership functions to give

$$y(\mathbf{x}) = \sum_{i=1}^{M_S}\Lambda_i(\mathbf{x})y_i(\mathbf{x}). \tag{2.31}$$

Instead of going into more detail about neuro fuzzy modeling, it is referred to [84], and the following subsections describe the approaches which are most commonly used in engine calibration.

2.3.1 LOLIMOT

The abbreviation LOLIMOT stands for LOcal LInear MOdel Tree, and this modeling technique was introduced in [83]. In this approach, the input space is partitioned by a tree-

construction algorithm and the local models are interpolated by overlapping local basis functions.
The partitioning is based on the well known CART (classification and regression trees) method. In each partitioning step of the LOLIMOT algorithm, all submodels are divided by axis orthogonal cuts in all possible dimensions, and the partitioning in which the best improvement could be achieved is maintained. At the end of this iterative procedure, the whole input space is divided into hypercuboids and the resulting structure is equivalent to a Takagi-Sugeno fuzzy system. In practice often linear polynomials are used for the local submodels, but sometimes also polynomials of higher degrees are applied.
LOLIMOT is widely used in engine calibration and can be found in, e.g., [36, 91, 107].

2.3.2 HHT

In comparison to LOLIMOT, the HHT (hinging hyperplane tree) algorithm allows also intersections, which are not axis orthogonal. This is achieved by using a nonlinear optimization, in order to obtain the optimal directions of the straight intersections. Further, it can be distinguished between flat HHT structures [14] and hierarchical HHT structures [25, 121]. For the local submodels only linear polynomials are used in this approach.

2.3.3 Local Neuro Fuzzy Models

In [47] and [46] another local linear modeling is presented, which is used for identification in engine calibration. Instead of using straight intersections like in the LOLIMOT or HHT algorithms, the input space is divided into subspaces with ellipsoidal contour lines. The positions of the subspaces are determined with an EM (expectation maximization) algorithm. In this approach, a polynomial modeling is applied for the local submodels. Compared with the two other methods, this algorithm has the greatest flexibility, but it is also the one with the highest computational costs.

2.4 Nonlinear Regression Models

As linear and local modeling techniques suffer from several limitations, which is extensively discussed in chapter 3, other types of modeling have been developed and used in engine calibration. All these approaches have in common that they are nonlinear in their model parameters Θ, and therefore these techniques are referred to as nonlinear regression models.

At first, the multilayer perceptron (MLP) neural network is considered, since this is the most commonly used nonlinear modeling technique in engine calibration. Then, so-called kernel techniques, like Gaussian processes (GP) and support vector machines (SVM), are examined.

2.4.1 Multilayer Perceptron Neural Networks

The MLP network is used in various different fields of application. It is widely used in machine learning [11] where it is applied, e.g., to image recognition, speech recognition or machine translation. In engine calibration it was introduced by [77], and it has been successfully integrated into the online optimization concept mbminimize at BMW [116].

A MLP with a single hidden layer is defined by

$$\mathbf{y}(\mathbf{x}) = \boldsymbol{\Theta}^o f_{\tanh}(\boldsymbol{\Theta}^h \mathbf{x}) \tag{2.32}$$

where the vector $\mathbf{y} \in \mathbb{R}^{N_o}$ contains the N_o outputs of the MLP network, the matrices $\boldsymbol{\Theta}^h \in \mathbb{R}^{N_h \times D}$ and $\boldsymbol{\Theta}^o \in \mathbb{R}^{N_o \times N_h+1}$ refer to the weights of the network, and the function $f_{\tanh} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h+1}, \mathbf{x} \mapsto (\tanh(x_1), ..., \tanh(x_{N_h}), 1)$ contains the activation functions of the N_h hidden neurons of the MLP. A schematic structure of the MLP network is drawn in figure 2.4, where $D = 3$ inputs, $N_h = 4$ hidden units and $N_o = 2$ outputs are chosen.
It would be possible to choose more hidden layers or other activation functions than the $\tanh-$function. But this architecture is important, because it has been shown by [42] that networks with one hidden layer and the $\tanh$ activation function are universal approximators as the number of hidden units tends to infinity. Therefore, this structure is most commonly used in engine calibration, and it will also be examined in this thesis. Further, in engine calibration it is common to calculate an own MLP network for each engine value. Hence, only one output for each MLP will be considered in this thesis.

During the training of the MLP, the parameters $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}^h, \boldsymbol{\Theta}^o\}$ are adjusted in order to adapt the nonlinear behavior of a function. In the context of neural networks, these parameters are often referred to as weights of the network.

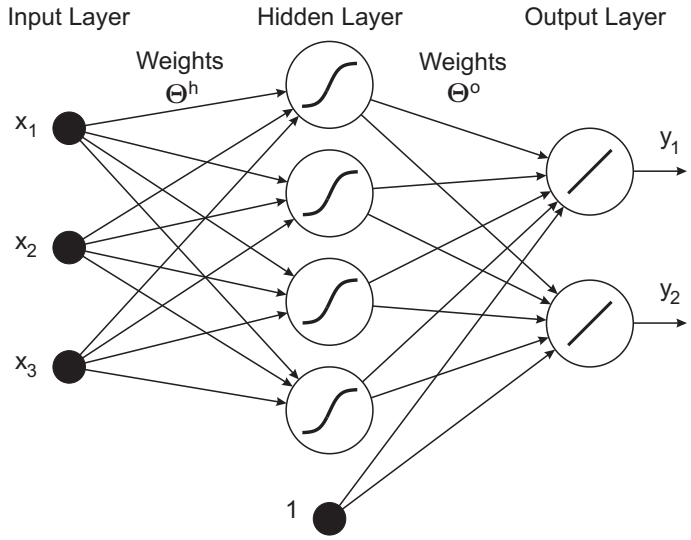


Figure 2.4: Schematic structure of the MLP network. The connections represent the weights and the nodes represent the outputs of the layers.

Various training procedures exist for MLP networks, see [10] for an overview. The appropriateness of a training algorithm depends on the specific problem. For the training of a few hundred weights, the Levenberg-Marquardt algorithm is a good choice [38], and therefore this algorithm is most commonly used in engine calibration [21].
The Levenberg-Marquardt technique is a numerical optimization method for nonlinear least squares problems. Like most other training approaches, this algorithm requires the gradient of an error function, e.g. (2.6) or (2.15), with respect to the parameters $\boldsymbol{\Theta}$. This gradient can be calculated efficiently with the well known error backpropagation procedure. With this gradient information, the Levenberg-Marquardt technique interpolates between the Gauss-Newton algorithm and the steepest descent method. In this way, the Levenberg-Marquardt approach is more robust than the Gauss-Newton algorithm alone, and it is highly probable that this algorithm is converging, even with a bad initialization [79].

As discussed in the bias-variance section 2.1.2, the flexibility of the model has to be adjusted to an optimal value in order to avoid overfitting and underfitting. In this section above, various different techniques have been examined, which all can be applied to MLP networks. However, as said in subsection 2.1.2.3, the regularization technique (2.15) has been found to work very well in practice. In order to apply this approach to the MLP network, it is convenient to consider the equation (2.14) for the regularized sum of squares error function.
The aim of the regularization is to find optimal values for the parameters α and β in (2.14). For this task, the Bayesian regularization has been found to be a very effective approach, which does not require any further data than the training set, and therefore this method is commonly used in engine calibration [92], [21]. This technique was introduced in [65] and integrated in the Levenberg-Marquardt training in [30]. Instead of reproducing the derivation of the Bayesian regularization, it should be referred to the literature given above. The regularization parameters at every Levenberg-Marquardt step can be calculated as follows [92]:

$$\alpha = \frac{M}{2\|\boldsymbol{\Theta}\|^2 + \operatorname{trace}\left(H^{-1}_{RSSE,\boldsymbol{\Theta}}\right)} \tag{2.33}$$

$$\beta = \frac{N - \gamma}{2 \cdot \mathrm{SSE}} \tag{2.34}$$

$$\gamma = M - \alpha \operatorname{trace}\left(H^{-1}_{RSSE,\boldsymbol{\Theta}}\right) \tag{2.35}$$

where SSE is the sum of squares error function (2.6) and $H_{RSSE,\boldsymbol{\Theta}}$ is the Hessian of the regularized sum of squares error function (2.15) with respect to the model parameters $\boldsymbol{\Theta}$. The variable γ represents the number of parameters which are effectively used in the model [11]. By comparing this variable to the number of all parameters M, it can be checked if the MLP network contains enough weights. [92] suggested, that if the number of parameters which is used in the model γ is bigger than 80% of all model parameters M, hence if $\gamma > 0.8M$, then the MLP should be enlarged by adding additional hidden units to the network. With this procedure an appropriate size of the network can be determined automatically.

2.4.2 Gaussian Processes

As it will soon be discussed in chapter 3, the Gaussian process regression has various advantages compared to other techniques for stationary base engine calibration. Therefore the discussion of this modeling will be more thorough than the ones of the other approaches. However, an even more detailed examination can be found in [98].

The motivation for the use of Gaussian processes (GP) in engine calibration is straightforward. As we will see in section 3.2, polynomial regression has some significant drawbacks, since the basis functions ϕ_j in (2.17) have to be chosen in advance, before the model training. However, typically we do not know which basis functions are suitable before the training data is observed. Therefore, [26] suggests to work with an infinity number of basis functions, which can be achieved with GP regression, which we will see shortly. In addition, [68] remarks that Gaussian processes are useful tools for automated tasks. Further, as said above, RBF and MLP networks with Bayesian regularization are widely used in engine calibration. As there is a relation between GP and neural networks, which is discussed in the sections 3.3 and 3.5, it was assumed that GP also work in practice.

However, the analysis on GP models and their use for regression and prediction is far from new [68]. Already in 1880, T.N. Thiele was analyzing time-series using Gaussian processes, see [63]. Within the geostatistics field, regression using GP is called kriging, see [17] and [26]. Moreover, ARMA (autoregressive moving average) models and Kalman filters can be viewed as forms of Gaussian process models [11]. Further, GP are used in the task of global optimization (e.g. see [50]).

2.4.2.1 Dual Representation

The Gaussian process viewpoint is a non-parametric approach, and this type of modeling is somewhat different than the other modeling techniques which have been discussed so far. In order to gain a better understanding of GP regression, the derivation of the formulas is started with the dual representation, in which initially the well known parametric viewpoint is considered (for more detailed information see [11]).

We start the discussion by replicating equation (2.17), where a model is considered which is linear in the model parameters $\boldsymbol{\Theta}_{\text{lin}}$

$$y(\mathbf{x}, \boldsymbol{\Theta}_{\text{lin}}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\Theta}_{\text{lin}} \tag{2.36}$$

and where $\boldsymbol{\phi}(\mathbf{x})$ is a vector of nonlinear basis functions of $\mathbf{x}$. As said above, typically the parameters $\boldsymbol{\Theta}_{\text{lin}}$ of this model are determined by minimizing the regularized sum-of-squares error function (2.15), in which we can incorporate (2.36) to give

$$\text{RSSE}(\boldsymbol{\Theta}_{\text{lin}}) = \sum_{n=1}^{N} (t_n - \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\Theta}_{\text{lin}})^2 + \lambda \|\boldsymbol{\Theta}_{\text{lin}}\|^2. \tag{2.37}$$

As we have seen, the closed-form solution for this problem is given by (2.20), and with this result the model can be calculated by

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T \left(\lambda \mathbf{I}_M + \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \mathbf{t} \tag{2.38}$$

which can be shown through a detailed transformation [11] to be equivalent to

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\Phi}^T \left(\lambda \mathbf{I}_N + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1} \mathbf{t}. \tag{2.39}$$

Now we introduce the kernel function $k(\mathbf{x}, \mathbf{x}')$ and the Gram matrix $\mathbf{K}$

$$k(\mathbf{x}, \mathbf{x}') := \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}') \tag{2.40}$$

$$\mathbf{K} := \boldsymbol{\Phi}\boldsymbol{\Phi}^T \tag{2.41}$$

which are linked through

$$k(\mathbf{x}_m, \mathbf{x}_n) = k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}_m) = (\mathbf{K})_{n,m}. \tag{2.42}$$

If we substitute (2.40) and (2.41) into (2.39), then we obtain the following formulation for our model

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t} \tag{2.43}$$

where we have defined the vector $\mathbf{k}(\mathbf{x})$ with elements $k_n = k(\mathbf{x}, \mathbf{x}_n)$.

From (2.43) we see that the solution of the regularized least squares problem (2.37) can be expressed completely by the kernel function $k(\mathbf{x}, \mathbf{x}')$. Hence, we can now work directly with the kernel function without explicit calculation of the basis functions $\boldsymbol{\phi}(\mathbf{x})$. This allows us to choose kernel functions where the vector $\boldsymbol{\phi}(\mathbf{x})$ contains implicitly many (even infinite) basis functions, which we will see shortly. This is regarded as the kernel trick or the kernel substitution in the literature, and this technique was first published in [2].

2.4.2.2 The Squared Exponential Kernel

A common choice for the kernel function is the squared exponential kernel (or sometimes called squared exponential covariance function)

$$k_{SE}(\mathbf{x}, \mathbf{x}') := \theta_\sigma^2 \exp\left(-\sum_{j=1}^{D} \frac{(x_j - x_j')^2}{2\theta_{l,j}^2}\right) \tag{2.44}$$

with the signal variance θ_σ^2 and the length-scale parameters in each input dimension $\theta_{l,j}$. Since the Gaussian process regression is a form of non-parametric modeling, the parameters $\{\theta_\sigma^2, \theta_{l,1}, ..., , \theta_{l,D}\}$ are called hyperparameters in the area of machine learning.
The length-scale hyperparameters have an interesting property. As we will see shortly, we can

estimate the values of all hyperparameters out of the training data. In doing so, it is possible that different inputs obtain different values for the length-scale parameters. As it can be seen from (2.44), if a particular parameter $\theta_{l,j}$ becomes high, the function becomes relatively insensitive to the corresponding input variable x_j. Hence, with the squared exponential kernel it becomes possible to detect input variables that have little or much effect on the model. Therefore, we are able to interpret the model also from a physical viewpoint. Inputs which have a high or low value for $\theta_{l,j}$, have a low or high nonlinear behavior. This determination of the importance of a certain input is called automatic relevance determination, and it is well known in machine learning. Instead of going into more detail on this technique, it is referred to the literature [98] and [11].

Further, the squared exponential kernel has another interesting property. It can be shown, that the vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$ that corresponds to this kernel has infinite dimensionality. This can be seen by expanding the kernel through a power series [11]. Hence, in the GP viewpoint we are able to perform regression, where implicitly an infinite number of basis functions is used.
Because of these and other advantages which are also discussed in chapter 3, at every GP regression in this thesis the squared exponential kernel (2.44) is used unless otherwise stated.

2.4.2.3 Training and Prediction

In order to apply Gaussian processes for regression, we need to consider the noise ϵ_n on our measurements t_n of the engine, which are given in (2.1). In this section an i.i.d. normal noise is considered[1], which is common in engine calibration, so that

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \sigma^2\mathbf{I}) \tag{2.45}$$

where σ^2 is the variance of the normal distribution. We follow the definition of Gaussian processes from [98]:

Definition 2.1. *A* Gaussian process *is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

From this definition it follows that the distribution $p(\mathbf{y}|\mathbf{X})$ at the observed input locations $\mathbf{X}$ is a multivariate Gaussian distribution

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \mathbf{K}, \mathbf{X}) \tag{2.46}$$

[1] In engine calibration usually the SSE function (2.6) or the RSSE function (2.15) is used for the training of the parametric models. As shown in the sections 2.1.1 and 2.1.2.1, this is equivalent of choosing a normally distributed noise (2.45). It should be noted, however, due to outliers in the measurements, which are quite common in engine calibration, this assumption is sometimes *not* a good one. In chapter 4 a possible solution will be presented.

with mean $\boldsymbol{\mu}$ and whose covariance is defined by the Gram matrix $\mathbf{K}$. For notational convenience, we will suppress the dependence on $\mathbf{X}$ in the following, and we will consider a zero-mean Gaussian Process.

Now we want to perform training. In the Gaussian process viewpoint, this means that we want to infer the hyperparameters out of the training data.
The Gram matrix contains the hyperparameters of the squared exponential kernel, so that $\mathbf{K} = \mathbf{K}(\theta_\sigma^2, \theta_{l,1}, ..., , \theta_{l,D})$. In addition, we do not know the variance σ^2 of the measurement noise (2.45). For notational convenience we collect all the hyperparameters into a single vector of hyperparameters $\boldsymbol{\Theta} := \{\theta_\sigma^2, \theta_{l,1}, ..., \theta_{l,D}, \sigma^2\}$.
There are two common training techniques for Gaussian processes: the marginal likelihood technique and the leave-one-out cross-validation technique. In this thesis the marginal likelihood technique is preferred, because it performed well on practical problems, see section 3.7, and we can achieve a robust formulation of GP with this training method, see section 4.
The marginal likelihood is the integral of the likelihood times the prior

$$p(\mathbf{t}|\boldsymbol{\Theta}) = \int p(\mathbf{t}|\mathbf{y}, \boldsymbol{\Theta}) p(\mathbf{y}|\boldsymbol{\Theta}) \mathrm{d}\mathbf{y}. \tag{2.47}$$

$$= \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}) \tag{2.48}$$

where we used (2.45), (2.46) and standard formulas given in [98]. The term marginal likelihood refers to the marginalization over the function values $\mathbf{y}$. Because of numerical reasons, it is convenient to optimize the log likelihood

$$\ln p(\mathbf{t}|\boldsymbol{\Theta}) = -\frac{1}{2}\ln|\mathbf{K} + \sigma^2\mathbf{I}| - \frac{1}{2}\mathbf{t}^T\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi). \tag{2.49}$$

As said above, with (2.49) the hyperparameters $\boldsymbol{\Theta}$ can be optimized on the training data. Hence, in practical implementations the derivatives of $\ln p(\mathbf{t}|\boldsymbol{\Theta})$ with respect to the elements of $\boldsymbol{\Theta}$ are calculated, and a quasi-Newton method can be used for optimization.

After the training, we want to predict the value y_* of our Gaussian process model at a new input location $\mathbf{x}_*$.
Since our GP model is a stochastic process, the value y_* will be distributed. Using the definition, again it follows that the observed measurements $\mathbf{t}$ and the prediction y_* are jointly Gaussian distributed [98], which can be written as

$$\begin{bmatrix} \mathbf{t} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{k}(\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*)^T & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right). \tag{2.50}$$

From this joint distribution we can derive the conditional probability distribution $p(y_*|\mathbf{t})$, which is a Gaussian distribution with mean and variance given by

$$\mathbb{E}[y_*|\mathbf{x}_*, \mathbf{t}] = \mathbf{k}(\mathbf{x}_*)^T\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{t} \tag{2.51}$$

$$\mathbb{V}[y_*|\mathbf{x}_*, \mathbf{t}] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{x}_*). \tag{2.52}$$

2.4.2.4 Illustrating some Properties of GP

A simple theoretical example is given in figure 2.5 where some basic properties of Gaussian process regression are illustrated.

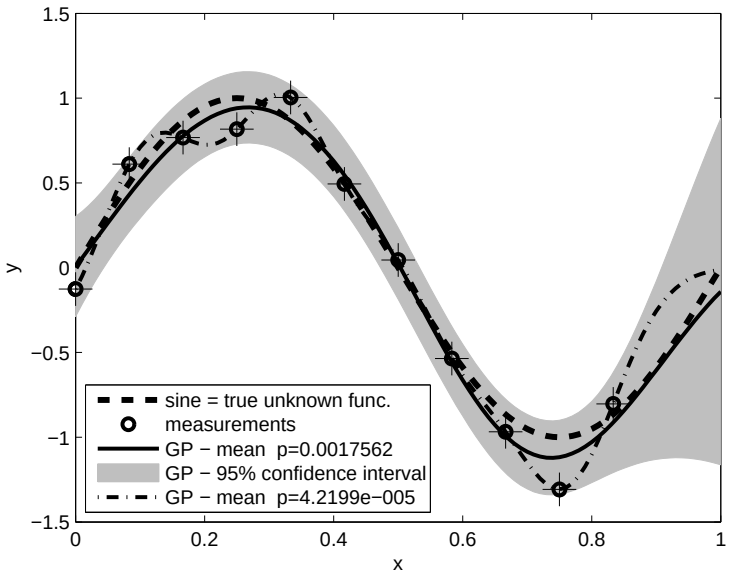


Figure 2.5: Illustration of some properties of GP in a simple example.

The dashed line represents the unknown function (sine), which in practice could be any non-linear engine mapping. One only knows some measured data (circles), which is shifted by random noise.
With this measured data a Gaussian process model can be trained by optimizing the log marginal likelihood (2.49). After the training the prediction can be performed. The predicted mean (solid line - calculated from (2.51)) represents the estimated function value and with the predicted variance (2.52), a 95% confidence interval can be drawn, which represents the degree of certainty where the estimated function is expected. Note the widening confidence interval on the right edge, which represents an increasing uncertainty of the real function behavior due to difficult extrapolation and the lack of measurements in this area.
Further, another GP is considered whose hyperparameters are not optimized on the log marginal likelihood (2.49). From this GP, the predicted mean (dot-dashed line) is plotted, and one could clearly interpret this GP model as overfitted on the training data. It can be seen, that the probability $p(\mathbf{t}|\boldsymbol{\Theta})$ (calculated from (2.49)) of this overfitted GP is much less than the probability of the optimized GP. In this way, through optimizing the hyperparameters, overfitting can be avoided.

2.4.2.5 The Relevance Vector Machine

As we will see in section 3.8, the computational cost of Gaussian processes is relatively high, compared to the other types of modeling which are presented in this chapter. This can be critical in some applications. Therefore, other kernel techniques have been developed and used, which try to minimize the computational effort. The two most common techniques are the relevance vector machine (RVM) and the support vector machine, which is discussed in the next section. These approaches are often called sparse kernel machines, since the main idea of these techniques is to use only a subset of training data points for predictions.

The relevance vector machine (introduced by [119]) is a linear model (2.17) of the form studied in section 2.2. In the above section, we chose the prior distribution (2.12) for the linear model parameters $\boldsymbol{\Theta}_{lin}$, which results into the RSSE function (2.15) where we were able to calculate the closed-form solution (2.20). In contrast to this, in the RVM framework a separate hyperparameter α_j for each linear model parameter $\theta_{lin,j}$ is introduced, so that the prior takes the form

$$p(\boldsymbol{\Theta}_{lin}|\boldsymbol{\alpha}) = \prod_{j=1}^{M} \mathcal{N}\left(\theta_{lin,j}|0, \alpha_j^{-1}\right) \tag{2.53}$$

where α_j represents the precision of the corresponding parameter $\theta_{lin,j}$, $\boldsymbol{\alpha}$ denotes $(\alpha_1, ..., \alpha_M)^T$ and $\boldsymbol{\Theta}_{lin}$ denotes $(\theta_{lin,1}, ..., \theta_{lin,M})^T$. It can be shown by maximizing the marginal likelihood (2.47) with respect to the new hyperparameters $\boldsymbol{\alpha}$, that a significant proportion of $\boldsymbol{\alpha}$ goes to infinity. Therefore, the associated basis functions $\phi_i(\mathbf{x})$ play no role in the predictions made by the model and so are effectively pruned out, resulting in a sparse model [11]. The basis functions that survive are called relevance vectors.

Although usually not presented as such, the relevance vector machine is actually a special case of a Gaussian process. It is pointed out in [97] and [119], that the RVM is equivalent do a Gaussian process with the kernel function

$$k_{\text{RVM}}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{M} \frac{1}{\alpha_j} \phi_j(\mathbf{x})^T \phi_j(\mathbf{x}'). \tag{2.54}$$

2.4.3 Support Vector Machines

Since the 1990's there has been an explosion of interest in kernel machines, and in particular in the support vector machine (SVM) [98]. Also in the area of engine calibration the SVM is becoming more and more popular.
We start the discussion of SVM by considering a model which is linear in its parameters $\boldsymbol{\Theta}_{lin}$ as in (2.17) in section 2.2. Here, we repeat (2.17) and we add explicitly an offset parameter b to the model, so that

$$y(\mathbf{x}) = y(\mathbf{x}, \boldsymbol{\Theta}_{lin}, b) = \boldsymbol{\Theta}_{lin}^T \boldsymbol{\phi}(\mathbf{x}) + b. \tag{2.55}$$

In section 2.2 we minimized the RSSE function (2.15), and we were able to calculate the closed-form solution (2.20). In comparison to that, the SSE part in (2.15) is replaced by an $\tilde{\epsilon}$-insensitive error function [127]

$$E_{\tilde{\epsilon}}(y(\mathbf{x}_n) - t_n) := \begin{cases} 0, & \text{if } |y(\mathbf{x}_n) - t_n| < \tilde{\epsilon} \\ |y(\mathbf{x}_n) - t_n| - \tilde{\epsilon}, & \text{otherwise} \end{cases} \tag{2.56}$$

in order to obtain a sparse solution of the support vector regression, which we will see soon (An exception are the least squares support vector machines (LS-SVM). Nevertheless, the LS-SVM can be directly viewed as a special case of a Gaussian process.). With this replacement, we therefore seek to minimize the regularized error function given by

$$C \sum_{n=1}^{N} E_{\tilde{\epsilon}}(y(\mathbf{x}_n) - t_n) + \frac{1}{2}\|\boldsymbol{\Theta}_{lin}\|^2 \tag{2.57}$$

where C is (by convention) the (inverse) regularization parameter. It can be shown by a detailed derivation [114], that the model (2.55) which minimizes (2.57) can be found by solving the following dual optimization problem

$$\begin{aligned} &\text{maximize} \quad \begin{cases} -\frac{1}{2} \sum\limits_{n=1}^{N} \sum\limits_{m=1}^{N} (\alpha_n - \alpha_n^*)(\alpha_m - \alpha_m^*) k(\mathbf{x}_n, \mathbf{x}_m) \\ -\tilde{\epsilon} \sum\limits_{n=1}^{N} (\alpha_n + \alpha_n^*) + \sum\limits_{n=1}^{N} (\alpha_n - \alpha_n^*) t_n \end{cases} \\ &\text{subject to} \quad \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) = 0 \;\text{ and }\; \alpha_n, \alpha_n^* \in [0, C] \end{aligned} \tag{2.58}$$

with the model (2.55) to give

$$y(\mathbf{x}) = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}) + b = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) k(\mathbf{x}, \mathbf{x}_n) + b \tag{2.59}$$

where we introduced the kernel function k (The variables α_n and α_n^* result from a dual optimization to (2.58), which is not discussed in more detail, in order to abbreviate this section). The offset b can be found by considering a data point $(\mathbf{x}_n, t_n)_{n \in \{1,\ldots,N\}}$ for which $0 < \alpha_n < C$ is satisfied, and then b can be obtained by

$$b = t_n - \tilde{\epsilon} - \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) k(\mathbf{x}, \mathbf{x}_n). \tag{2.60}$$

From (2.59) it can be seen that only the points $\mathbf{x}_n$ contribute to the model, where the values α_n and α_n^* are nonzero. These points are called support vectors. For all other points the contribution vanishes, and therefore the SVM results in a sparse model.

Chapter 3

MODEL COMPARISON IN THE CONTEXT OF ENGINE CALIBRATION

In the previous chapter, various different types of modeling have been introduced and discussed. All these types have been or can be used in stationary base engine calibration. However, we are typically interested in the most suitable type of modeling, which can also be used for model-based online optimization. Therefore, in this chapter the different modeling techniques are compared against each other.

In most previous publications on modeling in engine calibration, only a single or a few different types of modeling are considered. Hence, with these publications it is not possible to give a reliable recommendation of a most suitable type of modeling. Only a model comparison, which considers as much different techniques as possible, can result in a meaningful recommendation.
In fact, there exist a few model comparisons which analyze a greater number of modeling approaches, but none of them examines the theory and practical applications as comprehensive as the following, and none of them can give as clear recommendations as this one.
In [92] a model comparison is given, which considers a relatively low number of modeling techniques and which neglects the most promising approaches. Additionally, in [92] it is mentioned that a further examination of more types of modeling could help to improve the quality of regression.
In [34] and [70] model comparisons for dynamic modeling are given, where a comprehensive number of different approaches is considered. However, in these publications only the modeling performance on different practical data sets are compared and no extensive examination of theoretical properties is regarded. Hence, no clear recommendation of the most suitable type of modeling can be given, but rather a tendency which techniques are appropriate.

In contrast to the application of engine calibration, there exists a lot of literature in which the theoretical properties of the modeling techniques and the relationships between the different approaches are examined comprehensively under a general viewpoint, without considering a specific application or system. However, each of the techniques has its advantages and drawbacks, since it is not possible to specify a specific algorithm which works best on *all* possible applications, which directly follows from the no free lunch theorems for supervised learning [132, 133]. These theorems show that all algorithms have an equivalent average performance over all possible problems.

Hence, only by considering a specific problem or application, one can give a meaningful recommendation for a specific algorithm. Therefore, we will discuss the requirements on the modeling, which follow from the application of engine calibration, and then we examine the appropriateness of the different modeling algorithms.

In addition, there exists no publication which highlights Gaussian processes for engine calibration, compared to the other techniques. However, as we will see soon, the Gaussian process regression is the most suitable type of modeling for stationary base engine calibration. Therefore, another motivation for this chapter is to recommend Gaussian processes and to show the improvements which can be obtained, compared to the other state of the art algorithms.

The following sections are partially extensions of the results published in [7] and [9]. First, the requirements on the modeling are discussed, which result from the application of engine calibration. As said above, we will see that the Gaussian process regression has various advantages compared to other state of the art algorithms. Hence, it will be sufficient to compare the GP regression with each of the other types of modeling, in order to identify this approach as the most promising one, and this is performed in the later sections. Further, an investigation on the practical performance of different types of modeling is examined, where the theoretical assumptions are verified, and the results of the theoretical comparison are illustrated.

3.1 Requirements on the Modeling in the Context of Engine Calibration

In this section some important requirements on the modeling in model-based engine calibration are examined, which follow from Chapter 1, where the application of engine calibration is discussed. These requirements allow to draw conclusions in the further sections. A focus is made on the model-based online optimization (see section 1.2.2.2), but most of the requirements are also important for the model-based offline optimization (see section 1.2.2.1).

(REQ1) The modeling must be suitable for *high-dimensional* problems (5-10 input dimensions). The term 'high-dimensional' refers to the application of engine calibration. In the machine learning area, a high-dimensional problem would regard a few hundred inputs. A practical example in engine calibration is the optimization of a diesel engine with the 6 parameters: quantity and time of the pre-injection, quantity and time of the post-injection, main injection time and injection pressure. This leads to a 6 dimensional input space.

(REQ2) As mentioned in section 1.2.2, the engine test bench is an expensive system. Hence, the number of measurements should be minimized, in order to reduce time and costs of the calibration. Therefore, the modeling should be able to achieve a good performance with *as few measurements as possible*. That is why every measurement should contribute a maximum of information to the model.

(REQ3) Referring to section 2.1.2, the modeling should be *flexible* enough, so that every nonlinear engine mapping can be approximated. A good adaptation of the model to the measurement data should always be possible.

(REQ4) Also referring to section 2.1.2, the algorithm should be able to determine the optimal flexibility of the model and the problem of *overfitting has to be solved*. The model-flexibility must never be too big, and an overfitting on the measurement data has to be avoided. Thus, the modeling has to be robust to noise on the measurements.

(REQ5) The requirements (REQ3) and (REQ4) have to be met every time the modeling is performed. In addition, these tasks have to be performed *automatically and dependably*, so that an automated online optimization with no manual interaction is possible. This requirement is crucial. If, at any time, the modeling is not able to be flexible enough or overfitting occurs, in an automated online optimization bad models will lead to wrong predictions and useless measurements will be taken at undesired regions. In the worst case, without manual interaction, a large part of measurements would be meaningless and the optimization would cause high costs.

(REQ6) As mentioned in section 1.2.2.2, in the model-based online optimization we want to take measurements in areas where the model quality is bad, in order to improve the prediction of the model. Hence, the modeling has not only to be able to predict an expectation about the true engine behavior, but also a quantity about the *certainty and probability* of the model is important for an automated online optimization. Only with this quantity, measurement points cannot only be placed at the assumed optimum, but also where a big uncertainty about the model-expectation occurs.

Clearly, one can formulate additional requirements for a good modeling for engine calibration (e.g. like physical interpretation of the model). Some of them are discussed in the further sections. However, a full list of all possible requirements is beyond the topic of this thesis, but we will see that, if we assume that these requirements are the most important ones, it will be possible to identify the most suitable modeling with this choice.

3.2 Gaussian Processes compared to Polynomial Regression

At first, it may seem that a comparison between Gaussian processes and polynomial regression will not be very meaningful, since the drawbacks of polynomial regression are well known, and therefore the GP regression seems to be a more appropriate modeling.
However, as we will see shortly, this comparison will allow us to draw conclusions, which, on the one hand, illustrate the advantages of GP regression compared to linear modeling (2.17) in general, and which, on the other hand, can be adopted to other types of modeling as well (like the local linear modeling in section 2.3). Therefore, we want to compare the most common type of modeling for engine calibration, the polynomial regression, to the Gaussian processes, which are rarely used in engine calibration.

Polynomial regression has several advantages compared to Gaussian processes. Polynomials have a simple form, are well known and easy to understand. Further, as polynomial regression is a special form of linear modeling (2.17), we can obtain a closed-form solution ((2.19) and (2.20)) for the model parameters, and therefore, this modeling is computationally cheap and easy to implement.

In order to avoid overfitting (requirement (REQ4)), statistical tests can be used, see section 2.2.1.1. These tests remove parameters which are not significant and therefore not needed in the model. In this way, a big set of admissible basis functions can be chosen for regression, which increases the potential flexibility of the modeling (requirement (REQ3)), without the fear of obtaining overfitting.

However, as already mentioned above, there are some drawbacks of polynomial regression in theory and practice.
One disadvantage of polynomial regression is a bad extrapolation of the data. Polynomials, which are not constant over the whole input space, tend very fast to high (positive or negative) values outside the region of the measurement data. In comparison to that, using the SE kernel (2.44), Gaussian processes tend to the mean of the data, if every measurement is far away from the prediction.
Further, Gaussian processes indicate a growing uncertainty of the prediction very fast in an increasing of the variance (2.52) (e.g. see the right edge at figure 2.5). This property makes it easy for the user to distinguish which predictions one can trust. A confidence interval can also be calculated for polynomials [26], [57]. But this estimation of the prediction error relies on the accuracy of the polynomial model, and as we will see soon and in section 3.7, sometimes the performance of the polynomial model will be bad. Hence, in these cases one cannot trust the estimation of the confidence interval, too. This is contrary to the good performance of GP regression in engine calibration, which will be shown in section 3.7.

Another drawback is, that polynomials of high order tend to waviness and 'end-effects'. This can be illustrated by Runge's phenomenon, which describes the problem of oscillation at the edges of an interval. Although this phenomenon is a problem of interpolation, these oscillations also can be observed at regression, if the order of the polynomial increases, see figure 3.1.
The thick dashed line in the different plots in figure 3.1 indicates Runge's function, which is given by $\frac{1}{1+x^2}$. From this function, training data (circles) is sampled for regression. In the different plots the number of training data is varied, and for clarity, in the last plot only the half training data is drawn. With this training data, a Gaussian process model is calculated and a polynomial stepwise regression is performed.
It can clearly be seen that polynomial regression has problems on approximating this function. With few measurements, the polynomial stepwise regression will only take a polynomial of lower order as a significant one, which is not flexible enough to approximate Runge's function. By increasing the number of measurements, a polynomial of higher order is chosen, which is highly oscillating. This performance of the polynomial should be compared to the Gaussian process regression (gray). With the same amount of training data, the GP performs clearly much better.

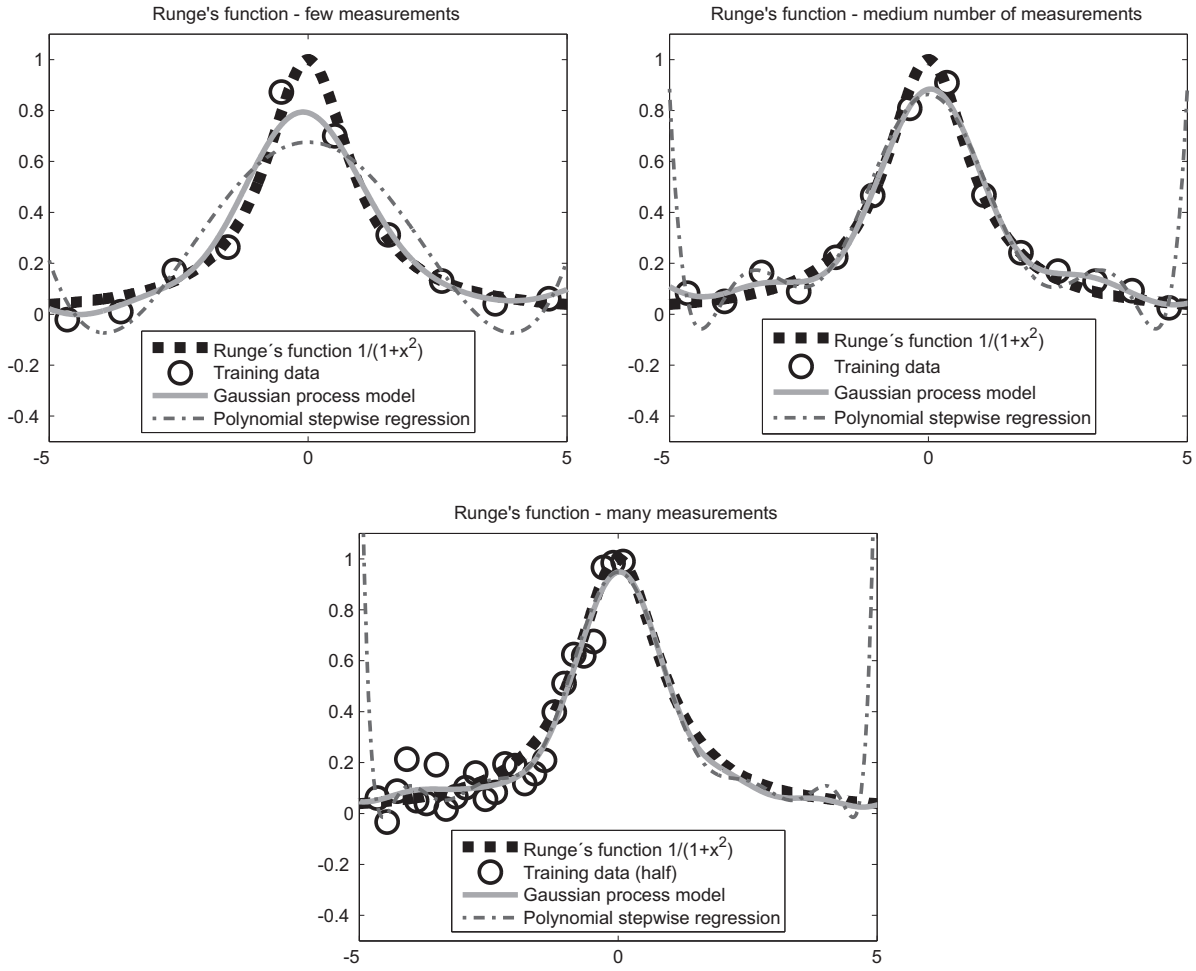


Figure 3.1: Illustration of some drawbacks of polynomial regression.

These oscillation effects are strongly related to the undesired effect of nonlocal behavior in polynomial regression. If polynomials are used for regression, [69] showed that measurements can have a large and undesired influence on the predicted function at a location, which is very different from the location where the measurements have been made. E.g. one can show that an increase in the measurements at one location can cause a decrease in a very other location.

A deeper understanding of this phenomenon and this comparison can be gained through considering the polynomial regression under a Gaussian process viewpoint.
As we saw in the dual representation in section 2.4.2.1, we can reformulate a linear modeling into a modeling with a kernel function $k(\mathbf{x}, \mathbf{x}')$. Hence, since the polynomial regression is a special case of linear modeling, we can calculate the kernel function k of the polynomial basis functions $\boldsymbol{\phi}(\mathbf{x})$ through (2.40). By doing so, one can discover the nonlocal behavior of the polynomial kernel function [11], which explains Runge's phenomenon.

In addition, this analysis shows that the Gaussian process viewpoint is a generalization of linear modeling (and especially polynomial regression). As said in section 2.4.2.1, the advantage of GP regression is that we can use kernel functions k, which can only be expressed by an infinite dimensional vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$, which would be the case if one uses the common squared exponential kernel (2.44).
Hence, now we can see another reason why polynomials (and linear models in general) do bad at approximating some functions (like in figure 3.1). It is the limited number of basis functions. If one would add the basis function $\phi_j(x) := \frac{1}{1+x^2}$ to the linear modeling in figure 3.1, then a perfect fit would be obtained. But how can one know which basis functions to use? As a consequence, in many practical applications the number of basis functions needs to grow rapidly, often exponentially, with the number of inputs [11]. Therefore, [26] suggests to go another way. The solution is to work with an infinity number of basis functions, which is given at the Gaussian process viewpoint, if one chooses the SE kernel (2.44).

As a conclusion, polynomial regression should only be performed for low complex problems, which can be approximated by polynomial terms of lower order. In the context of engine calibration, this means that only the behavior of a few adjustment parameters can be modeled through a polynomial, which clearly contradicts requirement (REQ1), and only measurement variables with smooth characteristics should be approximated by this simple approach.

3.3 Gaussian Processes compared to RBF Networks and the LLR Model

As the RBF Networks and the LLR model have much in common, they will be compared against Gaussian processes in a single section.

RBF Networks

As discussed in section 2.2.2, the RBF Network, given by (2.17) and (2.26), is linear in the parameters $\boldsymbol{\Theta}$ and nonlinear in the parameters $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, $j \in \{1, ..., M\}$. The performance of the RBF regression depends on the training of the parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$.

If all the parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ are simultaneously optimized under a probabilistic perspective, then we can determine the RBF performance under a Gaussian process viewpoint.
As said above, the Gaussian process viewpoint can be seen as a generalization of the linear modeling. Instead of performing the regression with the linear basis functions (2.26), we can calculate the kernel function (2.40) of these basis functions, in order to perform the regression with a Gaussian process with the corresponding kernel function, which will lead to the same result. Hence, a comparison between RBF modeling and Gaussian process modeling can be reduced to a comparison of the performance of the different kernel functions in practical applications. As we will see soon in section 3.7, a suitable kernel function is the squared exponential covariance function (2.44), which shows a good performance on practical problems of engine calibration. Further, no improvement could be achieved by replacing this kernel

function with others [40].
In addition, [67] showed an interesting relationship between a dense radial basis function network and the squared exponential kernel (2.44). In this examination a RBF modeling with the basis functions (2.26) is considered, where the matrix $\boldsymbol{\Sigma}_j$ consists only of diagonal elements whose values are equal for all basis functions, hence $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$. It is shown by [67] and [98], that *this RBF modeling tends to a Gaussian process with the squared exponential kernel, as the RBF terms get dense in the input space and the number of RBF basis functions tends towards infinity*.

In the Gaussian process formulation, all the parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ of the RBF model would be optimized simultaneously. In contrast to this technique, in engine calibration normally not all parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ are optimized at the same time by the maximum likelihood or the maximum a posteriori principle. Rather, the parameters $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ are often determined in an other procedure, e.g. by clustering the training data in the input space or by placing the basis functions in suitable areas, without considering a probabilistic perspective for the parameters $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$.
This has many advantages. Often, these procedures have a simple structure, and typically they are easy to implement. In addition, the methods can usually exploit numerical advantages, like the usage of the closed-form maximum likelihood solution (2.19). Hence, these procedures are often computationally cheap.
But as it will be discussed at length in chapter 6, this probabilistic perspective will be very useful to evaluate the probability of the parameters and therefore the model quality. Through the additional information of the model quality, the online optimization routine can receive a feedback, if already enough measurements are taken, and the measurement on the test bed can be stopped. Hence, the test bench time can be reduced to an optimal amount. Without this probabilistic viewpoint, other methods (e.g. cross validation) have to be used, which always need additional measurements for testing the model quality, and therefore these methods need more measurements for the same performance, which clearly violates requirement (REQ2).
Further, the RBF networks, which are used in engine calibration, do not provide a quantity about the certainty and probability of the model [34, 92, 118] which is also a direct consequence of the lack of the probabilistic interpretation of the parameters, and this clearly violates requirement (REQ6).

All these drawbacks of these *particular types* of RBF networks, which are used in engine calibration [92], can be overcome by a reformulation of the RBF modeling in a Gaussian process formulation, as said above. With this reformulation, all parameters $(\boldsymbol{\Theta}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ can be determined from a probabilistic perspective and also the modeling provides an estimation of the variance of the prediction. Consequently, this raises the question why a RBF modeling should be used at all.
Therefore, in this thesis a modeling with Gaussian processes using the squared exponential kernel (2.44) is preferred.

LLR Model

Since the LLR model is a combination of two linear models, the training of this model is usually much faster than the training of a nonlinear model. Further, if some additional data is added to the measurement, also the retraining of the model is very fast, which is useful for online optimization where the models are updated very often.

However, a drawback of the LLR model is that the basis functions

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \boldsymbol{\mu}_j||^2}{2r_j^2}\right) \tag{3.1}$$

are used for the RBF part, where r_j is the radius of the RBF terms. As only a single parameter r_j is used for the fitting of the RBF neurons, no optimization can be performed for the length scaling of the different inputs, compared to the automatic relevance determination approach of Gaussian processes in section 2.4.2. Hence, the global performance of the RBF terms will not be very well and therefore the polynomial regression has been integrated in the LLR model. But clearly, this drawback could be overcome if one would use the basis functions (2.26) for the modeling.

Since the LLR model is a combination of a polynomial model and a RBF model, the LLR model can also be regarded under the Gaussian process viewpoint. The combination of these two models can be obtained by a Gaussian process, where the kernel function is a summation of two kernel functions

$$k(\mathbf{x}, \mathbf{x}') = k_{poly}(\mathbf{x}, \mathbf{x}') + k_{RBF}(\mathbf{x}, \mathbf{x}') \tag{3.2}$$

where $k_{poly}(\mathbf{x}, \mathbf{x}')$ is the kernel function of the polynomial and $k_{RBF}(\mathbf{x}, \mathbf{x}')$ is the kernel function of the RBF model. This representation through a Gaussian process has the advantage that all parameters can be optimized under a probabilistic perspective, which already was discussed extensively.
However, as said above, the squared exponential covariance function (2.44) has been identified as a suitable kernel function, which shows a good performance on practical data, and no improvement could be achieved by replacing this kernel function with others.

3.4 Gaussian Processes compared to Local Linear Models

As discussed in section 3.2, polynomial regression should only be used for low complex and low dimensional problems in engine calibration. However, as these linear models also have many advantages, more sophisticated approaches were developed and used, which try to minimize the disadvantages of the polynomials.
One possible solution can be obtained if the whole input space is divided into many smaller subspaces. With this partitioning, the complexity of regression of every single subspace is smaller than the complexity of the regression of the whole input space. Hence, a polynomial

(or other linear) modeling can again be used for regression for every single (local) subspace. The whole (global) model is then obtained through the composition of all local subspaces. This type of modeling can be regarded as local linear modeling, see section 2.3.

An example of local linear modeling is given in figure 3.2.

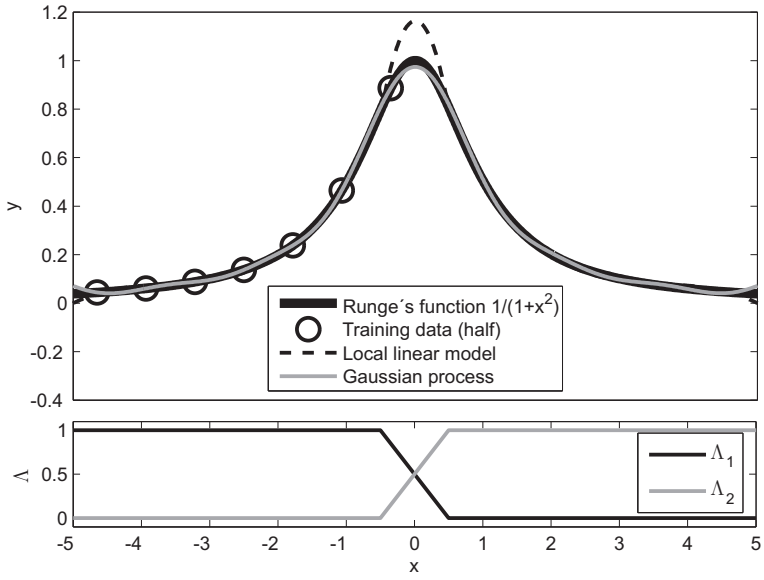


Figure 3.2: Illustration of local linear modeling.

As in figure 3.1, the aim of this example is the approximation of Runge's function $\frac{1}{1+x^2}$, and as above, training data is sampled (circles) from this function. In order to perform regression with the local linear modeling, the whole (one-dimensional) input space is split up into two parts, and a polynomial of order 3 is applied in each part. In the lower plot the membership functions Λ_1 and Λ_2 are shown, which (for simplicity) are taken to be linear, and in the upper plot the local linear model (dashed line) is drawn, which is given by (2.31). Also the Gaussian process regression (gray) is plotted. Clearly, this local linear model gives a better fit than the highly oscillating polynomial models in figure 3.1.

Local linear modeling has various advantages compared to Gaussian processes. A relatively simple structure leads to a fast training speed of the model. Further, since all the local models are typically polynomials, local linear modeling shares all the advantages from the polynomial modeling, like an easy implementation.

A human interpretability of a local linear model is often seen as another major strength. With LOLIMOT, HHT and local neuro fuzzy models, which were discussed in the sections 2.3.1, 2.3.2 and 2.3.3, it is stated that a human interpretation can be given by the particular (tree) structure, which is learned from the data. However, in practice it is found that this structure is very sensitive to the details of the data, so that small changes to the training data can result in very different sets of splits [11, 41].
As described in section 2.4.2.2, a human interpretation of a Gaussian process model can be

obtained through an automatic relevance determination (ARD), where the degree of nonlinearity of each input can easily be determined, see also [66, 81]. It is also possible to incorporate prior knowledge with ARD. If one knows the degree of nonlinearity from an input a priori (e.g. from a similar engine), then this knowledge can easily be incorporated into the measurement design. A similar incorporation of prior knowledge is also possible with LOLIMOT, HHT and local neuro fuzzy models, see e.g. [71].

However, local linear modeling suffers also from some drawbacks.
In a local linear model, the intersections between the submodels are critical areas.
Since measurements will only belong to a single submodel, they will only provide information for this specific submodel, even if these measurements lie very closely to another submodel. This clearly contradicts to requirement (REQ2), since every measurement should contribute a maximum of information to the model. As a consequence, this lack of information will result in a bad prediction at the intersection.
This characteristic can also be observed in figure 3.2. Because each submodel has no information about measurements which lie outside its subspace, the submodels do not know that the function decreases outside their subspace, and therefore, the local linear model predicts a high estimation at the intersection between the submodels. A better approximation of the intersection can only be given if more training data is sampled. It should be noted, that this problem does not occur if one uses a GP for approximation, since the GP uses all information, see figure 3.2. Further, it should be mentioned that this problem gets worse, if the number of inputs increases [9]. In addition, the interpolation behavior of local linear models at V-type situations, see [84] page 409, can cause strange results at the intersections.

Another problem of local linear modeling arises, if the type of intersections does not suit the nature of the function, which should be approximated.
Based on the example in figure 3.2, this problem is illustrated by considering a multidimensional Runge function, like

$$f_{MR} = \prod_{i=1}^{d} \frac{1}{1 + x_i^2}, \tag{3.3}$$

where x_i are the different inputs. Obviously, like in the one-dimensional case in figure 3.1, polynomial regression will not provide a good fit on this function. Hence, a local linear modeling will divide the whole input space into smaller subspaces. If only straight intersections are possible for modeling (e.g. with LOLIMOT and HHT), then a good fit can be obtained if every axis is split up into two parts, like in figure 3.2. But in a d-dimensional input space, this will lead to 2^d independent submodels, and since the number of measurements increases with the number of submodels, also the number of measurements increases exponentially with the number of inputs.
In [41] and [11] other examples of simple functions are given, where a similar poor performance of local linear modeling can be determined.

Generally, if the measurement values cannot be approximated by polynomials and if the type of intersections does not comply with the nature of the problem, then the number of submod-

els, and therefore the required number of measurements, will grow rapidly with local linear modeling in a high dimensional input space, which clearly contradicts to requirement (REQ1) and (REQ2).
It should be noted, that this is not the case with GP regression, since only one global model and no submodels exist. Hence, like the MLP network, Gaussian processes can get along better with the curse of dimensionality than local linear models, see also [84].
These theoretical considerations will be further illustrated with practical data sets in sections 3.7.3 and 3.7.4.

3.5 Gaussian Processes compared to MLP Networks

The MLP network is widely used in the area of machine learning and also in engine calibration, see e.g. the mbminimize concept of BMW [116]. In model-based offline optimization, see [76], model-based online optimization, see [53], and even in dynamic model-based online optimization, see [21], the MLP is used for modeling.
Surprisingly, the MLP modeling motivated the use of GP regression in this work. It was shown by [92], that the mbminimize concept, which is an online optimization concept, partially based on a committee of MLP networks, does not perform as well as the EGO algorithm, which is an online optimization concept, based on the DACE model, on noise free experimental data. However, because the DACE model cannot cope with noise on the measurements, this approach could not be used for noisy engine calibration tasks [92]. Nevertheless, since the DACE model is only a noise free Gaussian process, which can easily be extended with a noise term, see section 2.4.2.3, and because the committee of MLP networks works well in practice, it was assumed that GP regression could perform even better on engine calibration tasks.

Like Gaussian processes, the MLP networks comply with many requirements which are listed in section 3.1.
As mentioned above, the MLP can cope better with the curse of dimensionality (REQ1) than local linear models, see also [84], and therefore it needs fewer measurements for the same problem (REQ2). By adding additional hidden units, the flexibility of the MLP can be enlarged (REQ3), and with Bayesian regularization, overfitting can be avoided (REQ4). Further, it was found that this automatic model training is very dependable [92] (REQ5).
Due to random network initializations and nonlinear optimization of multimodal problems, a MLP modeling will generate different functions, which is sometimes seen as a drawback of a modeling with MLP's. However, exactly these different functions can be used to identify a quantity about the certainty and probability of the model (REQ6). If different MLP models predict similar values, then the certainty of the prediction is expected to be high and vice versa. Hence, a committee of MLP's can be used for model-based online optimization, which has been implemented in the query-by-committee approach by [92].

Therefore, in order to evaluate the performance of MLP networks compared to GP regression, we have to examine the differences and the relationships between these types of modeling.

At first, we will evaluate the computational costs of these approaches.
The training of a MLP network is performed through a nonlinear optimization of the M network weights. The computationally most expensive task of this optimization is the inversion of the Hessian $H_{RSSE,\boldsymbol{\Theta}}$ in (2.33) and (2.35). Since this matrix is of size $M \times M$, the computational complexity of the MLP training scales with $O(M^3)$. As will be discussed extensively in section 3.8, the GP training is performed through a nonlinear optimization of the hyperparameters, and the computational complexity of this training scales with $O(N^3)$, where N is the number of training data points. Hence, if $N \gg M$, which is typically the case in applications where we want to estimate a simple behavior out of a huge data set, then the MLP training will be much faster than the GP training. However, according to requirement (REQ2), in stationary base engine calibration we usually want to estimate the behavior out of a data set, which is as small as possible. Hence, nearly every measurement will result in an improvement of the accuracy of the model, which can only be included in the MLP network, if the number of model parameters M is increased. Thus, in engine calibration often $N \approx M$, and therefore the computational complexity of the MLP training will scale similar to the GP training for small data sets.
The same analysis can be performed with the prediction of the models. If $N \gg M$, then the prediction of the MLP network will be much faster, and if $N \approx M$, then the computational effort of both approaches will be similar.

There exists a simple relationship between MLP networks and Gaussian processes. [81] has shown that, using a Gaussian prior for the parameters which results into the common error function (2.14), *the distribution of functions generated by a MLP network will tend to a Gaussian process in the limit of an infinite number of neurons*.
Therefore, instead of a MLP modeling, one can perform a GP modeling with an equivalent result, if the neural network kernel function $k_{NN}(\mathbf{x}, \mathbf{x}')$ is used, which can be found in [98].

The update formulas (2.33)-(2.35) for the Bayesian regularization are results of a Gaussian approximation of the posterior distribution [81]. Hence, in the convergence proof of [81], Markov chain Monte Carlo (MCMC) methods are used in order to take the full advantage of the available data.
However, since the MCMC methods are computationally very expensive, in engine calibration only the the approximation scheme (2.33)-(2.35) is used. Therefore, in engine calibration the prediction of a MLP modeling will not be as accurate as a Gaussian process with the neural network kernel function. Especially the error between the MLP model and the real function behavior, according to the value (2.8), will not converge to zero, if the number of measurements tends to infinity, since only a MLP network of limited size can be used with the approximation scheme [81], and therefore there will always be a bias left in the model, because the MLP is only a universal approximator, if the number of neurons tends to infinity.
These theoretical considerations could be confirmed on practical data sets, and also other works, independent from this one, came to the same conclusion. E.g. in [58] it is observed, that the RMSE of the MLP model does not converge to the noise level, if the number of measurements increases, whereas the RMSE of a GP model converges to this minimum value.

The fact, that the MLP converges to the GP, raised a broad discussion in the area of machine learning, if Gaussian processes could possibly replace neural networks, see [67]. Further, why should one use query-by-committee with MLP networks, if a Gaussian process can be used, which produces the same results as a committee of an infinite number of MLP's, each with an infinite number of neurons?

However, a modeling with MLP's may be preferred for applications where the number of measurements is large, because a modeling with Gaussian processes is computationally expensive, see section 3.8.

3.6 Gaussian Processes compared to Support and Relevance Vector Machines

As we will discuss at length in section 3.8, the computational cost of Gaussian processes is high, since the kernel function $k(\mathbf{x}_n, \mathbf{x}_m)$ has to be evaluated for all possible pairs $\mathbf{x}_n$ and $\mathbf{x}_m$ of training points, which can be seen from (2.42). In some applications, this can be computationally infeasible during the training, and also the prediction might require excessive computation times.
Therefore, the support and relevance vector machines have been developed and used. These types of modeling use only a subset of the training data for prediction, and therefore these techniques are called sparse kernel machines.
Hence, the support and relevance vector machines do not stand in direct competition with the Gaussian process regression. The sparse kernel machines are computationally cheaper, but the predictions will not be as accurate as Gaussian processes. Therefore, a modeling with GP should always be preferred for applications where the computational effort is acceptable, and the SVM or RVM have to be used where the computational complexity for a GP modeling is too high.

The support vector machines have the advantage that the optimization (2.58) is a quadratic programming problem, whereas the marginal likelihood maximization for the relevance vector machine is non-convex and may be multimodal, which may lead to several local minima.
But support vector machines also suffer from some drawbacks.
As it can be seen from the equations in section 2.4.3, there is no probabilistic viewpoint in this modeling. Hence, the evaluation of the model quality typically requires cross-validation procedures, which are computationally expensive. Further, the predictions are not probabilistic either. However, as stated in requirement (REQ6), we need a quantity of how much we can trust the prediction. Since to the unique global minimum of (2.58), it is not possible either to build a committee of SVMs, like the MLP committee in the mbminimize concept [92]. Hence, the support vector regression does not comply with requirement (REQ6).
In addition, at support vector regression it is necessary to estimate the insensitivity parameter $\tilde{\epsilon}$ of (2.56) and the regularization parameter C of (2.57). This also generally requires a cross validation procedure, which is wasteful both of data and computation.

Furthermore, for a wide range of regression tasks the support vector machine does not achieve as sparse solutions as the relevance vector machine [119], while at the same time the generalization error, and therefore the quality of the prediction, is similar.
A problem of relevance vector machines arises, when it comes to the prediction at areas, which are far away from training cases. As argued in section 2.4.2.5, the use of the kernel function (2.54) allows to obtain a sparse solution for the RVM. But as shown in [97], it has also the undesirable effect that the predictive uncertainties get smaller the further one moves away from the training cases. Although the work [97] tries to fix this problem, it is shown that the kernel function (2.54) of the relevance vector machine is good for computational reasons, but bad for modeling reasons [98].

3.7 Practical Model Comparison on a Diesel Engine

In the sections above we discussed various theoretical properties. We discussed that the RBF network converges to a GP, that the MLP network converges to a GP, that the RVM is a special case of a GP and that the SVM has various disadvantages compared to GP. The relationships between these approaches can be well determined from the theory, and under consideration of the requirements of engine calibration, we were able to draw conclusions, which of these types of modeling is most suitable for stationary base calibration.
Further, we examined some drawbacks of polynomial regression and local linear modeling on theoretical examples. However, we only *claimed* that these drawbacks would also occur in practice. Hence, in this section these drawbacks are illustrated at a practical application, and we will demonstrate that the GP approach has various advantages compared to local linear modeling and polynomials, when it comes to practical data.
In addition, we already mentioned that the squared exponential kernel is suitable for approximation with GP, and now we want to review this statement.
Last but not least, a further motivation for this section is to demonstrate the performance of the Gaussian process approach. For every GP model in this section, the hyperparameters had been determined by a fully automatic nonlinear optimization without any manual interaction. Hence, a user of this approach does not need to have knowledge about GP regression, but nevertheless, with this automatic technique he is able to calculate a meaningful model out of the data, which performs better than the other state of the art approaches.

Therefore, in this section Gaussian process regression is applied on NOx, consumption and soot measurements of a diesel engine and compared to polynomial stepwise regression and local linear modeling.
The measurements result from a cooperation between KRATZER AUTOMATION AG and MAN Truck & Bus AG Nürnberg.

780 measurements have been taken from 11 operating points, which are shown in figure 3.3. The adjustment parameters (and therefore the inputs of the models) are the main injection time, injection pressure, quantity and time of the post injection and the desired air-fuel mixture ratio, which is controlled by the quantity of exhaust gas recirculation. As a global model is

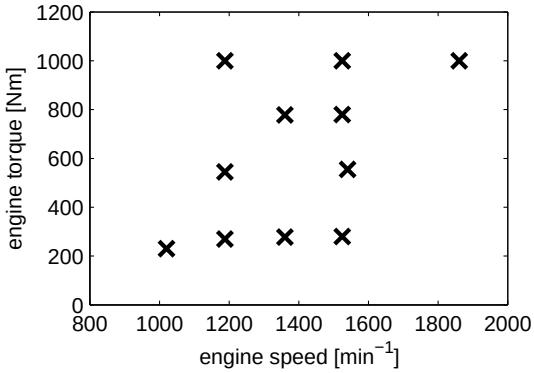


Figure 3.3: Operating points of the measured data.

calculated, also the engine speed and engine torque are taken as inputs. This leads to a 7-dimensional input space. For reasons of confidentiality, all measurements are scaled to an interval of [0 1].
At first, the modeling of the NOx and consumption measurements is considered. After that, the modeling of soot will be regarded.

3.7.1 Global Modeling of Consumption and NOx Emissions

For the NOx emissions and the consumption measurements, Gaussian processes, which are described in section 2.4.2, and polynomial stepwise regression, which is described in section 2.2.1.1, are applied on the training data and compared against each other.
In order to determine which admissible set of regressors should be used for stepwise regression, full polynomial models with all coefficients up to 4th order in all of the 7 input dimensions are applied to the whole measurement data, so that 330 polynomial coefficients have to be estimated out of 780 measurements. Figure 3.4 shows the measured-predicted plots of this modeling.
It can be seen that in both cases the models can adapt the nonlinearity of NOx and consumption. Therefore, it is assumed that a polynomial model of order 4 in all input dimensions is satisfactory for the complexity of the characteristics of NOx and consumption (in the later sections it will be shown, that this is not the case for the soot emissions). Hence, for the global polynomial stepwise regression, it will be sufficient to choose the already mentioned 330 coefficients for the admissible set of regressors. In order that the polynomial modeling performs well, the measurements have been taken from a d-optimal design.

The figures 3.5 and 3.6 show a comparison of global Gaussian process models and polynomial stepwise regression models.
In these figures the training data is varied. From the total set of 780 measurements, a subset is randomly selected and used for training, and the remaining measurements are used for validation. After that, a Gaussian process model and a polynomial stepwise regression model were calculated with the same training data. Table 3.1 shows the normalized RMSE (NRMSE)

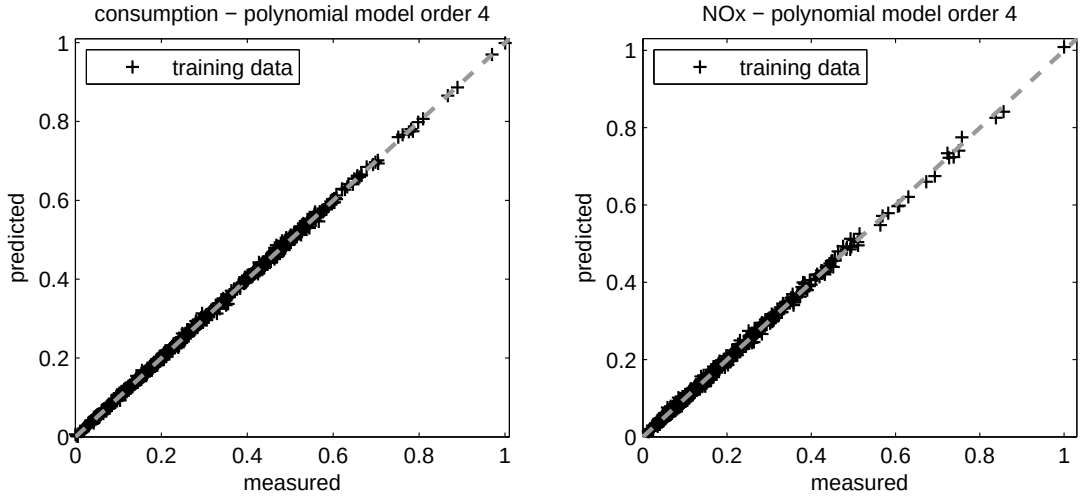


Figure 3.4: Measured-predicted plots of global polynomial models for consumption (left) and NOx emissions (right) of a diesel engine. A polynomial of order 4 has (in both cases) been used for modeling the measured data.

for the training and the validation data of the figures 3.5 and 3.6.

It is obvious that both types of modeling are able to adapt the characteristics of the measurements and, like in the simple example in section 3.2 in figure 3.1, the models improve if the number of training data is increased.
Further, it can be seen that the Gaussian process models always perform slightly better than the polynomial models. Therefore, for this NOx and consumption measurements, if one is using a Gaussian process modeling, either the model will have a better performance than the polynomial modeling, or one can reduce measurements in order to get the same performance than the polynomial modeling.
However, this positive effect of Gaussian process modeling is not very big for NOx and consumption. This is due to the low complexity of the characteristics of NOx and consumption. Since the behavior of both quantities is very smooth, they can easily be approximated by a

Consumption - NRMSE	Gaussian processes		polynomial stepwise regression	
	training data	validation data	training data	validation data
90% training data	0.006268	0.012265	0.004709	0.019829
70% training data	0.006252	0.012561	0.004220	0.027917
20% training data	0.002445	0.022655	0.009099	0.042445
NOx - NRMSE	Gaussian processes		polynomial stepwise regression	
	training data	validation data	training data	validation data
90% training data	0.002294	0.019084	0.004995	0.033580
70% training data	0.002599	0.014920	0.003808	0.034643
20% training data	0.001976	0.021771	0.015270	0.039939

Table 3.1: NRMSE of consumption (figure 3.5) and NOx (figure 3.6).

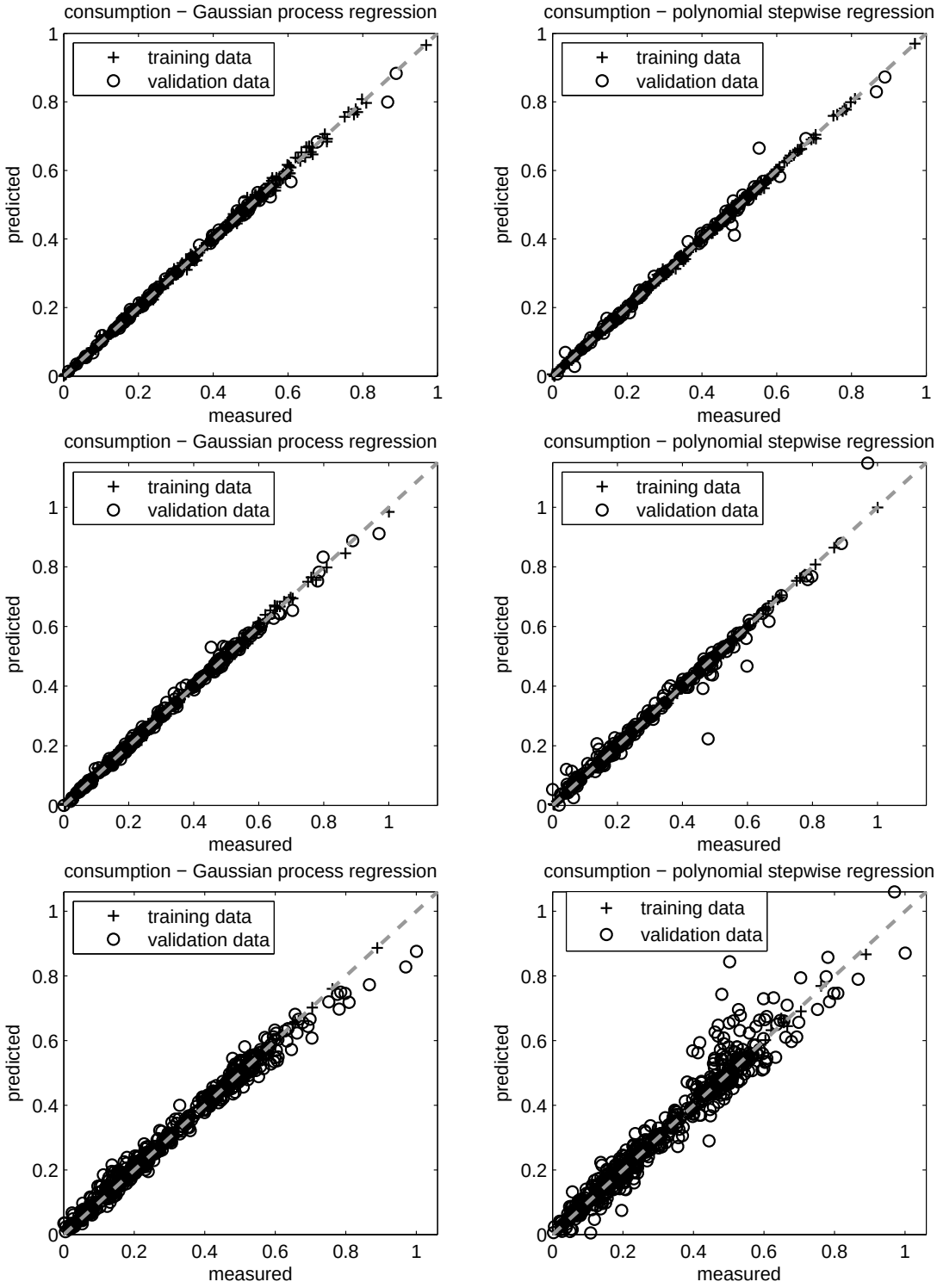


Figure 3.5: Comparison of modeling consumption. Top: Regression with 702 measurements for training (90%) and 78 measurements for validation (10%); Middle: Regression with 546 measurements for training (70%) and 234 measurements for validation (30%); Bottom: Regression with 156 measurements for training (20%) and 624 measurements for validation (80%); Left: Gaussian process modeling; Right: Polynomial stepwise regression.

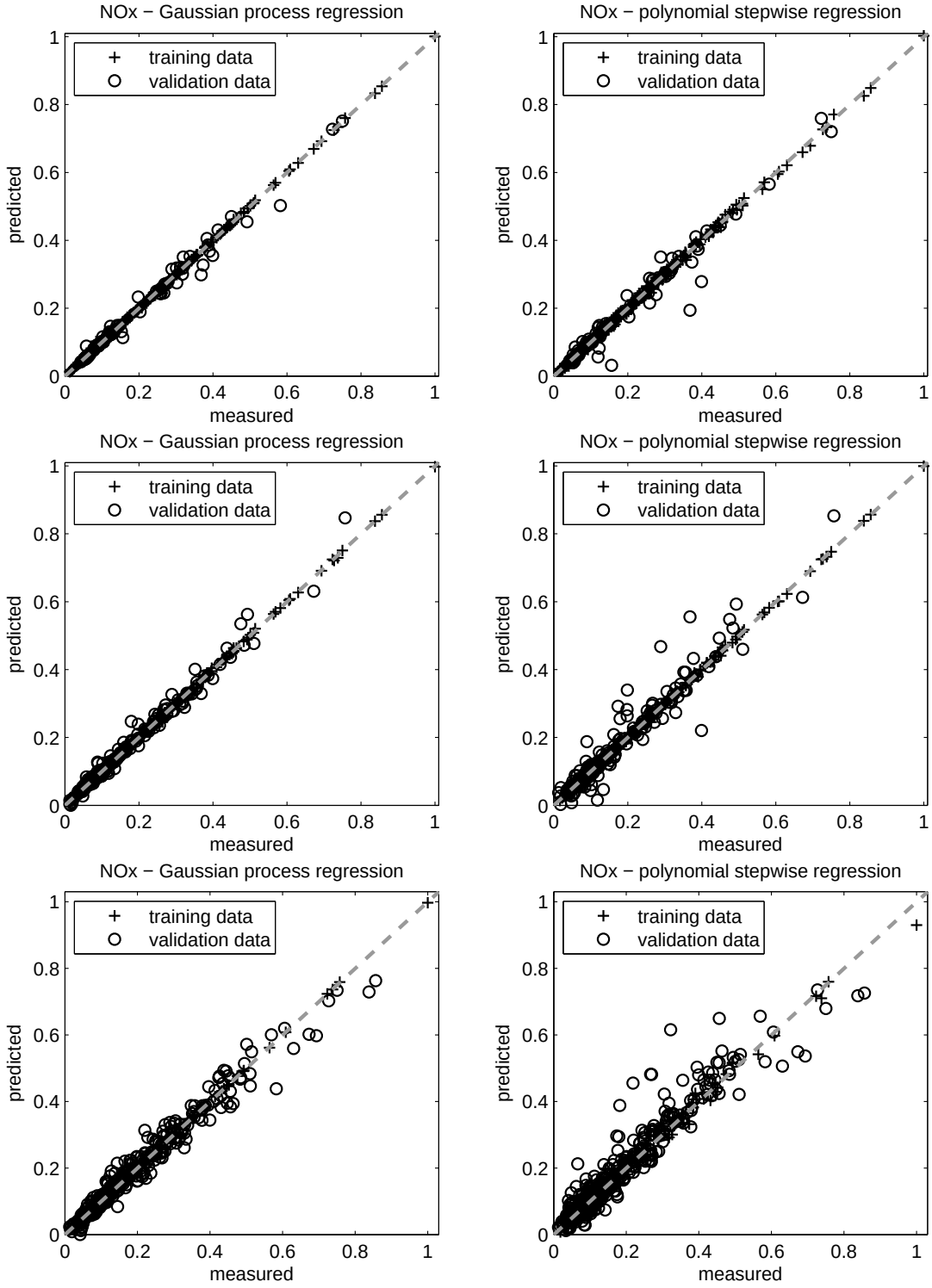


Figure 3.6: Comparison of modeling NOx emissions. Top: Regression with 702 measurements for training (90%) and 78 measurements for validation (10%); Middle: Regression with 546 measurements for training (70%) and 234 measurements for validation (30%); Bottom: Regression with 273 measurements for training (35%) and 507 measurements for validation (65%); Left: Gaussian process modeling; Right: Polynomial stepwise regression.

polynomial of a low (4th) order. In the following, the soot emissions of a diesel engine are considered, which are more difficult to model, and the main advantages of the Gaussian process regression will become obvious.
But it has been shown, that even if the complexity of the function which should be approximated is low and a polynomial modeling can be performed, the Gaussian process regression is useful as well and will also have a good performance.

3.7.2 Global Modeling of Soot Emissions

Now the modeling of the soot emissions is considered. 25 outliers had to be removed, so that only 755 measurements are available for the modeling (In the next section a modeling will be presented, which is robust to outliers. However, for the moment we want to move on with the standard approach).
In order to illustrate the increasing complexity when it comes to the modeling of soot, compared to NOx and consumption, again a polynomial regression of order 4 in all input dimensions is performed with the measurement data, and the result is shown in figure 3.7.

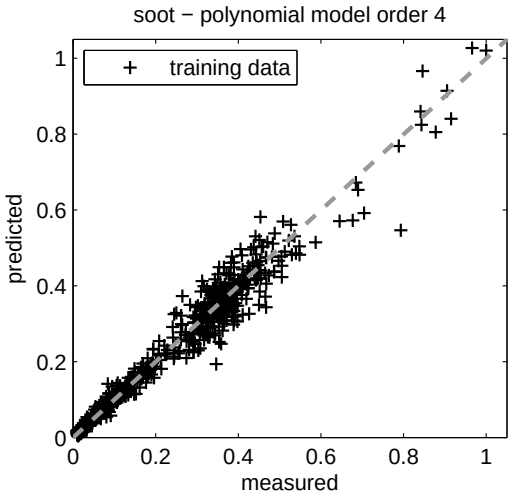


Figure 3.7: Measured-predicted plot of a global polynomial model for soot emissions. As in figure 3.4, a polynomial of order 4 has been used for modeling the measured data.

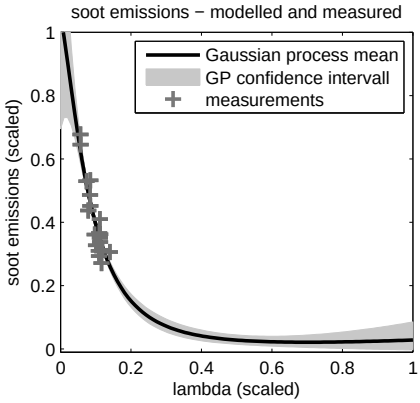


Figure 3.8: Intersection plot of soot emissions. From the model and the measurements, the highly nonlinear behavior can be seen.

By comparison of figure 3.4 and figure 3.7, it can clearly be seen that a polynomial of order 4 is not able to adapt the nonlinear behavior of the soot emissions. Although all data has been used for training and none for validation, the polynomial is not able to reproduce the soot emissions in the middle and top region.
The reason that soot is more complicated to model than NOx or consumption lies in the complexity of this quantity. There is more noise on the measurements and the behavior of soot is much more nonlinear than the behavior of NOx or consumption. In figure 3.8 an intersection plot of a soot model is given. That one can trust this model will soon be shown. Also mea-

surement data is plotted in figure 3.8. The fast increasing soot emissions at a small lambda value can be seen. It is clear that a polynomial of a low order is not able to cover this highly nonlinear behavior. Hence, from figure 3.7 and 3.8 it follows that the admissible set of regressors for the polynomial modeling requires also terms, which are higher than the 4th order.
It should be mentioned, that a full polynomial model of order 5 in all 7 input dimensions already contains 792 coefficients. If the polynomial degree is increased to 6 or 7, then the number of coefficients increases to 1716 and 3432 respectively. Therefore, one cannot calculate a full polynomial of an order of 5 or higher without being overfitted, since only 755 measurements are available. But like mentioned above, with polynomial stepwise regression overfitting can be avoided, since only the significant regressors are chosen for the model.

The figure 3.9 shows a comparison of a global Gaussian process model and global polynomial stepwise regression models.
From the total set of 755 measurements, a subset of 85% is randomly selected and used for training, and the remaining measurements are used for validation. After that, a Gaussian process model and polynomial models with stepwise regression were calculated with the same training data. As it is not clear which polynomial order for the admissible set of regressors should be chosen, this order is varied. Table 3.2 shows the NRMSE for the training and the validation data of figure 3.9.

	Gaussian process model	polynomial stepwise regression; admissible set of regressors has order 4	order 5	order 6	order 7
training data	0.006031	0.044208	0.015171	0.015062	0.011441
validation data	0.023704	0.065992	0.055462	0.076661	0.163660

Table 3.2: NRMSE of soot (figure 3.9).

From the plots in figure 3.9 it can clearly be seen, that a polynomial stepwise regression model, where the admissible set of regressor is of a low (4th) order, is not able to adapt the nonlinearity of the soot emissions. This was expected, since we already showed this fact for a full polynomial of order 4 in figure 3.7. Further, it can be seen that polynomials, where the admissible set of regressors is of a higher order, can adapt the nonlinearity better. But obviously, these polynomial models have some outliers in the validation data. These outliers get worse, if the polynomial degree is increasing. Note that the polynomial model, where the admissible set of regressors is of order 5, has an outlier on the top right, whose error is 0.3. If the order of the admissible set of regressors is increased to 6 or 7, then the error of the highest outlier increases to a value of 0.5 and 1.6 respectively (!). It should be reminded that all measurements are scaled to an interval of [0 1], and an outlier of an error of 1.6 means, that the error of the prediction is 160% of the spread between the highest and the lowest measurement.

In order to illustrate this effect of the outliers, figure 3.10 shows two intersection plots of the models.
These intersection plots had been made by varying only one input parameter, like the time (left) or the quantity (right) of the post injection, and keeping all other parameters at a con-

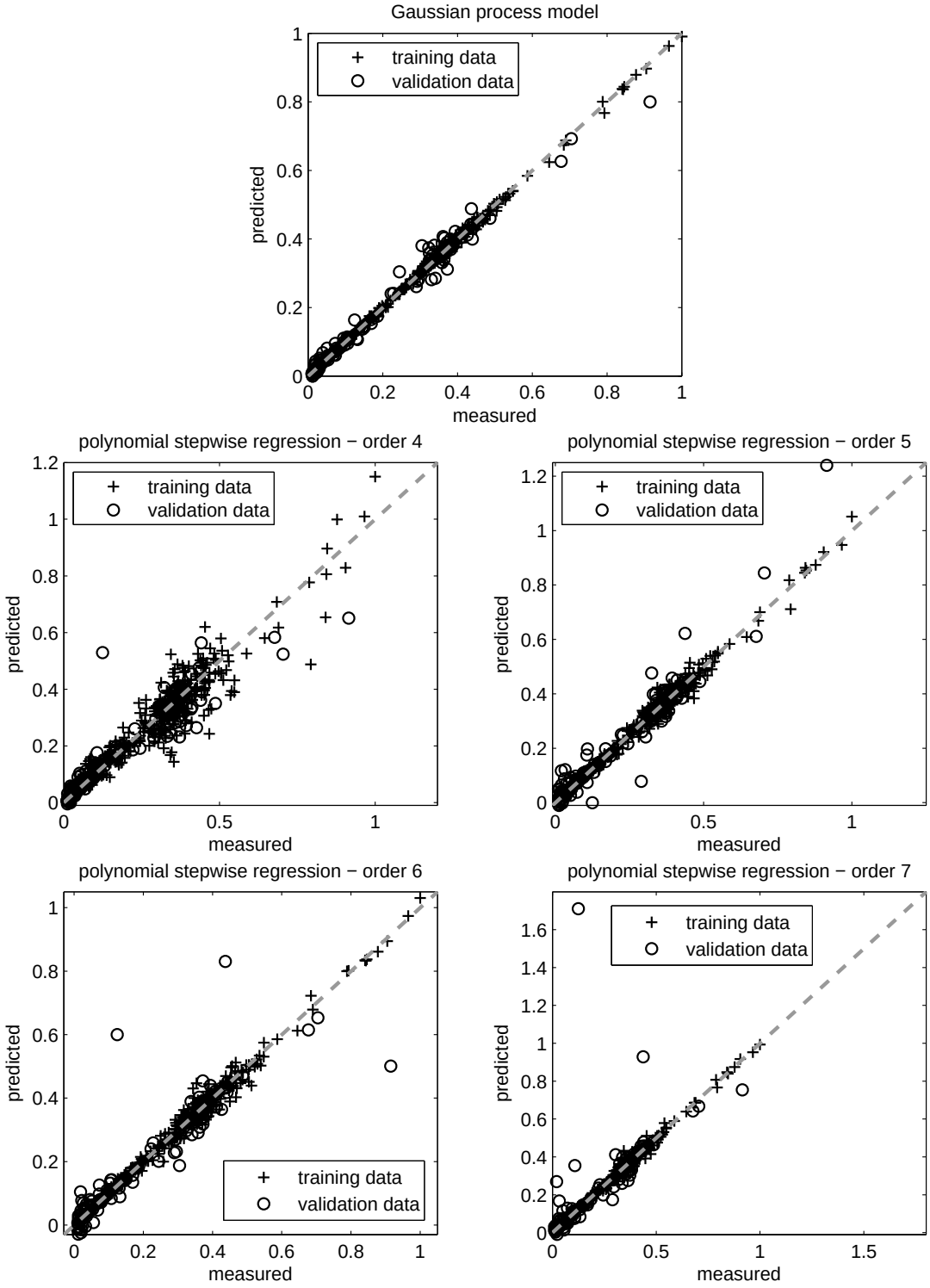


Figure 3.9: Comparison of modeling soot emissions with 642 measurements for training (85%) and 113 measurements for validation (15%); Top: Gaussian process modeling; Bottom: Polynomial stepwise regression, where the admissible set of regressors is given by a full polynomial of order 3, 4, 5 and 6.

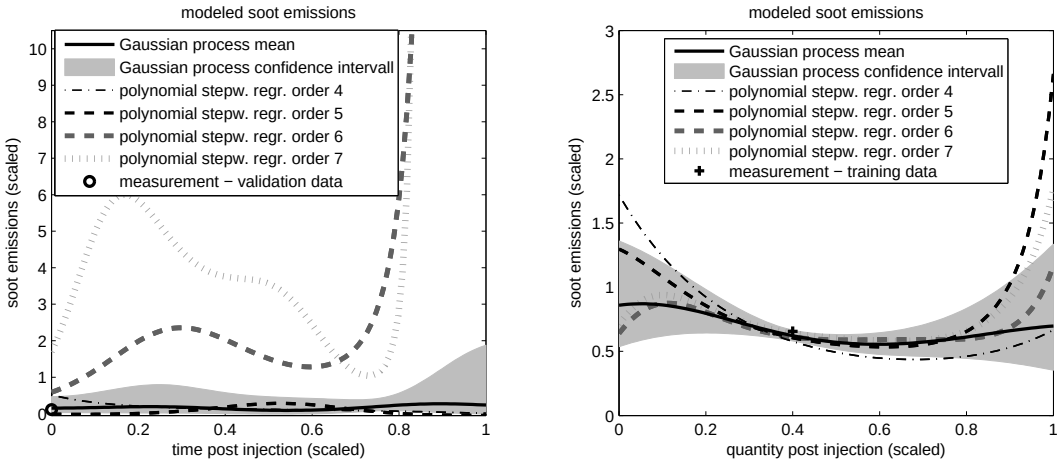


Figure 3.10: Intersection plots of modeled soot emissions with Gaussian processes and polynomial stepwise regression. In this figure the modeling is drawn over the time (left) and quantity (right) of the post injection, while all other inputs are held on constant values. The increasing values of the polynomial modeling at the edges of the interval can be seen.

stant value. As it can be seen, the SE kernel function (2.44) produces a smooth behavior of the GP model, which constitutes our model assumption in stationary base calibration. Further, again it should be mentioned that all measured data was scaled to an interval of [0 1] and the scaling of the plots should be noted. At some areas, some polynomial models predict a value, which is over 1000% higher than the highest value in the measurements. Is is clearly very unlikely that this prediction is a good one.

With figure 3.10 the appearance of the outliers in figure 3.9 can now be understood. Polynomial models, which contain coefficients with a high polynomial order, tend to oscillate in regions where measurements are scarce. This effect is equivalent to the effect in the theoretical example in section 3.2 in figure 3.1. As in the theoretical example, the polynomial models tend to oscillate where no data is available. But unlike in the theoretical example, the application on the diesel engine has not only one input parameter, but the input space is now 7-dimensional. Due to the curse of dimensionality, the volume is increasing rapidly when the input dimensionality is higher, and therefore there are many subspaces in the input space where measurements are scarce. In these subspaces, polynomial models, which contain coefficients of a high order, will tend to predict unlikely high values. Hence, the polynomials cannot be used for a global modeling of the soot emissions with this amount of data.

As in the theoretical example in figure 3.1 or in the practical example in section 3.7.1, clearly, the performance of the polynomial models could be improved, if the number of measurements would be increased. However, increasing the number of measurements is not necessary, if one is using a Gaussian process model, since from figure 3.9 and table 3.2 one can see that this approach gives a good global approximation of the soot emissions.

3.7.3 Comparison of Local and Global Modeling of Soot Emissions

In this section a comparison of local and global modeling of the soot emissions is examined. As mentioned above, the reason why polynomials cannot give a good global approximation to the soot emissions is the highly nonlinear behavior. In order to reduce the nonlinearity for the polynomial modeling, a local modeling for the polynomial stepwise regression is considered. In order to abbreviate this section, not every single type of local linear modeling (e.g. LOLIMOT, HHT, ect.) will be discussed here. Therefore, we do not want to consider any specific type of intersection. Thus, we will examine only a single operating point at a time for the polynomial model. This local modeling has the advantage that the input dimensionality decreases by a factor of 2, since engine speed and engine torque do not need to be regarded as inputs. Therefore, due to the curse of dimensionality, the space of regression is decreasing, and compared to a global model in section 3.7.2, polynomial models again can be used.
This specific type of modeling is also often called a local modeling in the literature, but this should not be mixed up with a general local linear modeling, where the intersections can also divide the space of the adjustment parameters. However, there exist different approaches, which combine the local linear operating point models to a global model [113]. In addition, we will see that we can draw conclusions from this specific type of local modeling, which verify our theoretical thoughts in section 3.4, independent from any specific partitioning algorithm.

The figures 3.11 and 3.12 show a comparison of global Gaussian process models and local polynomial stepwise regression models.
From the total set of the 755 measurements, two operating points were chosen: one at an engine speed of 1000 1/min and engine torque at 230 Nm, the other at engine speed of 1525 1/min and engine torque at 780 Nm. At these operating points 67 and 76, respectively, measurements were performed. These measurements were divided in 57 and 63, respectively, measurements for training (85% and 83%) and 10 and 13, respectively, measurements for validation (15% and 17%). With this training data, local polynomial stepwise regression models have been calculated. Since it was not clear which polynomial order should be chosen for the admissible set of regressors, this order had been varied.
Finally, in the figures 3.11 and 3.12 these polynomial models can be compared to the global Gaussian process model. For this GP model, only the measurement data at the certain operating point is plotted. It should be noted that, at the considered operating points, the global GP model has the same training and the same validation data as the local polynomial models. Table 3.3 shows the NRMSE for the training and the validation data of the figures 3.11 and 3.12.

As observed above, polynomials with a high order tend to oscillate (this can be seen by the outliers in the measured-predicted plots) and polynomials of a low order cannot approximate the nonlinearity of the soot emissions. But compared to the global polynomial modeling in section 3.7.2, there is always a local model which is not oscillating, which can be seen by the fact that there are no outliers in the validation data, and where the polynomial order is able to approximate the nonlinearity. Therefore, the local polynomial approach works better than the global polynomial modeling. Nevertheless, it can be seen that the local data of the global GP

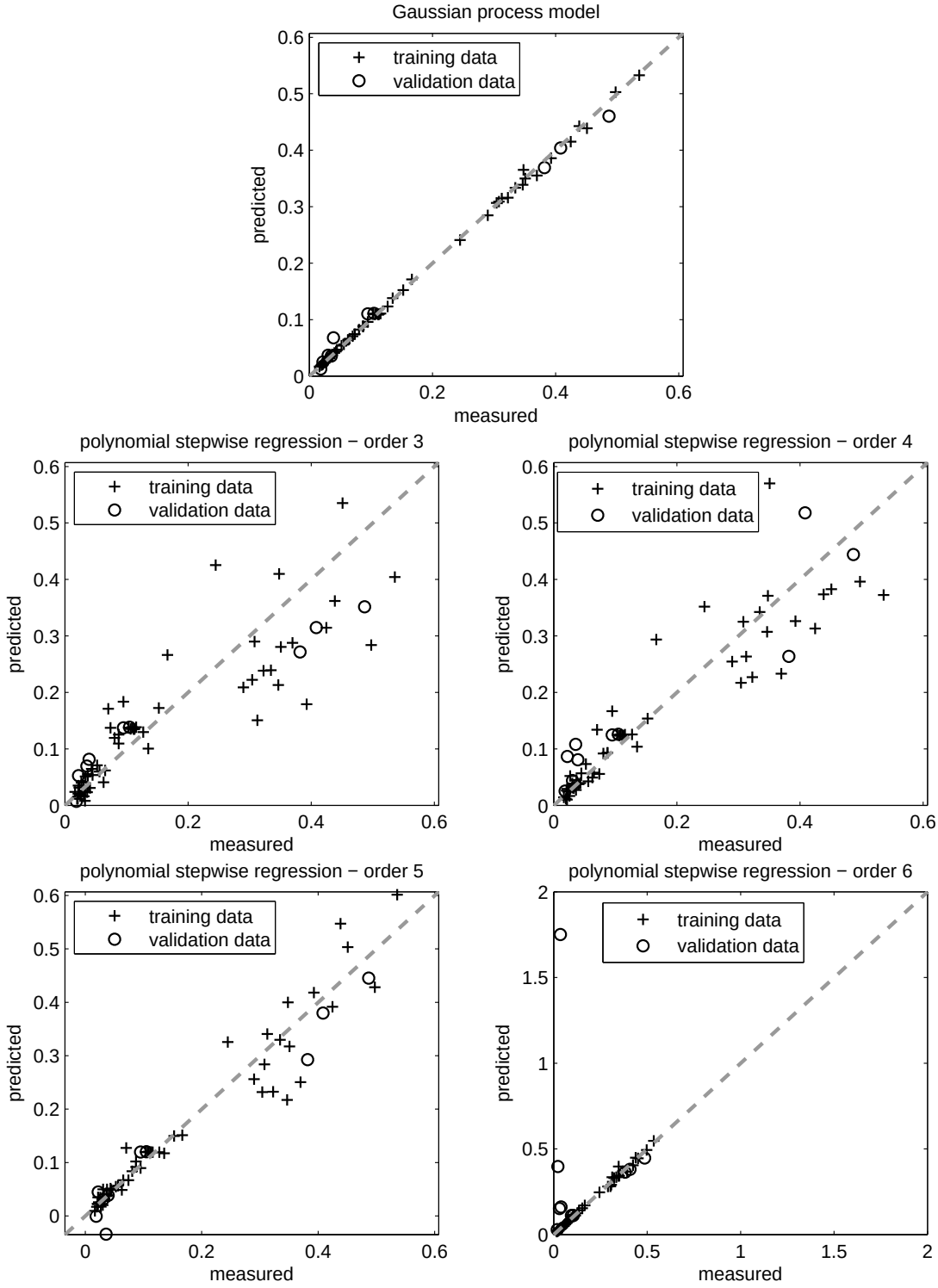


Figure 3.11: Comparison of global Gaussian process models and local polynomial stepwise regression models. From the total set of the 755 measurements, only the operating point at engine speed of 1000 1/min and engine torque of 230 Nm is considered. At this operating point there are 57 measurements for training (85%) and 10 measurements for validation (15%). Top: Measurements of the global Gaussian process modeling at this operating point; Bottom: With the local measurements, a local polynomial stepwise regression was performed, where the admissible set of regressors was given by a full polynomial of order 3, 4, 5 and 6.

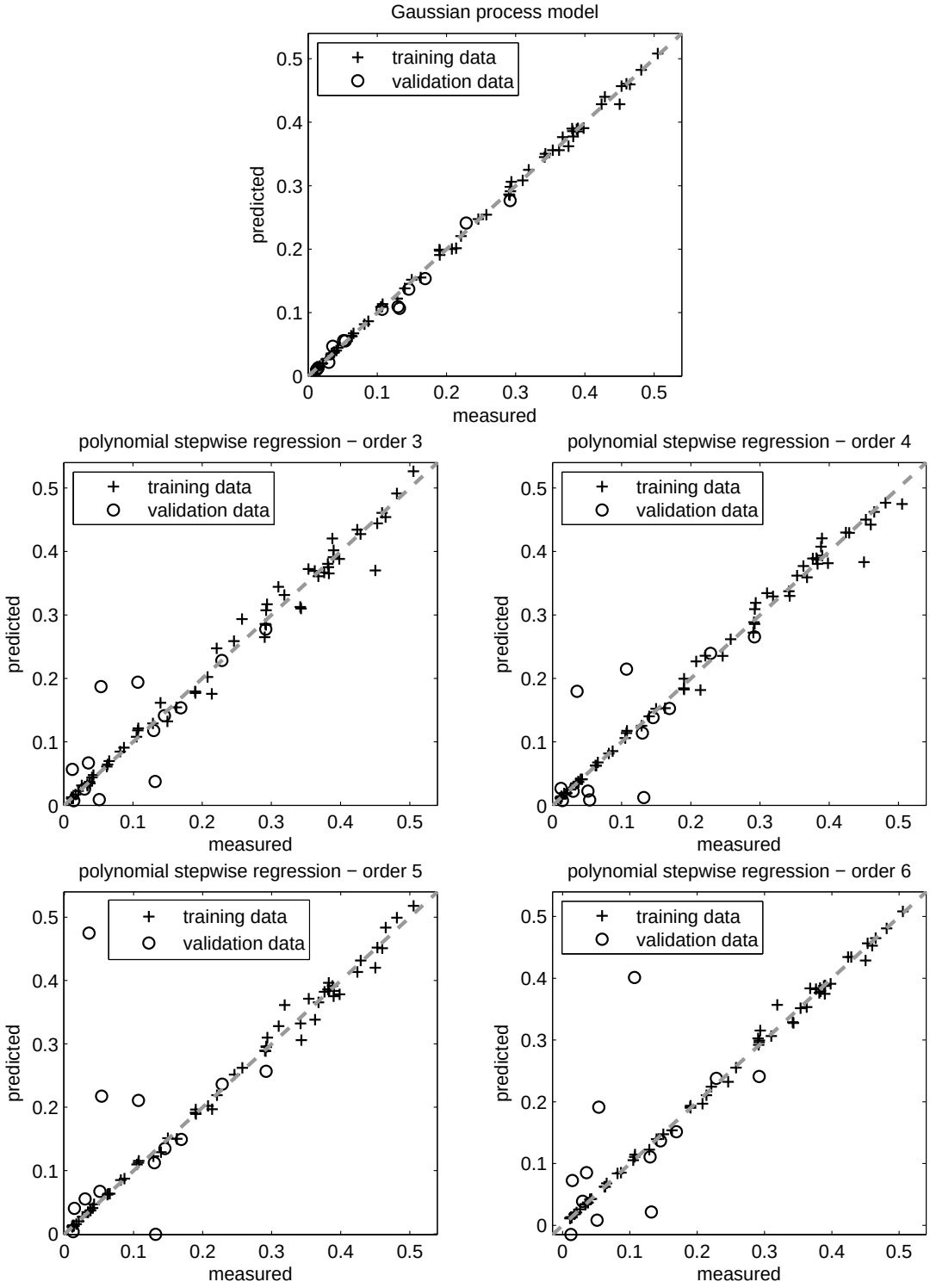


Figure 3.12: Comparison of global Gaussian process models and local polynomial stepwise regression models. From the total set of the 755 measurements, only the operating point at engine speed of 1525 1/min and engine torque of 780 Nm is considered. At this operating point there are 63 measurements for training (83%) and 13 measurements for validation (17%). Top: Measurements of the global Gaussian process modeling at this operating point; Bottom: With the local measurements, a local polynomial stepwise regression was performed, where the admissible set of regressors was given by a full polynomial of order 3, 4, 5 and 6.

Soot - NRMSE - operating point: engine speed = 1000 1/min; engine torque = 230 Nm					
	Gaussian process	polynomial stepwise regression; admissible set of regressors has			
	model	order 3	order 4	order 5	order 6
training data	0.004579	0.072715	0.058365	0.039499	0.009832
validation data	0.014404	0.067889	0.063581	0.041354	0.558070
Soot - NRMSE - operating point: engine speed = 1525 1/min; engine torque = 780 Nm					
	Gaussian process	polynomial stepwise regression; admissible set of regressors has			
	model	order 3	order 4	order 5	order 6
training data	0.005770	0.017621	0.014039	0.011615	0.008433
validation data	0.012361	0.055226	0.062795	0.139190	0.099872

Table 3.3: NRMSE for figures 3.11 and 3.12.

model has always a better performance than the local polynomial models at all polynomial orders.
In order to illustrate the difference of global and local modeling, with the local measurements also a local GP model is calculated and the results are shown in figure 3.13. Clearly, the local

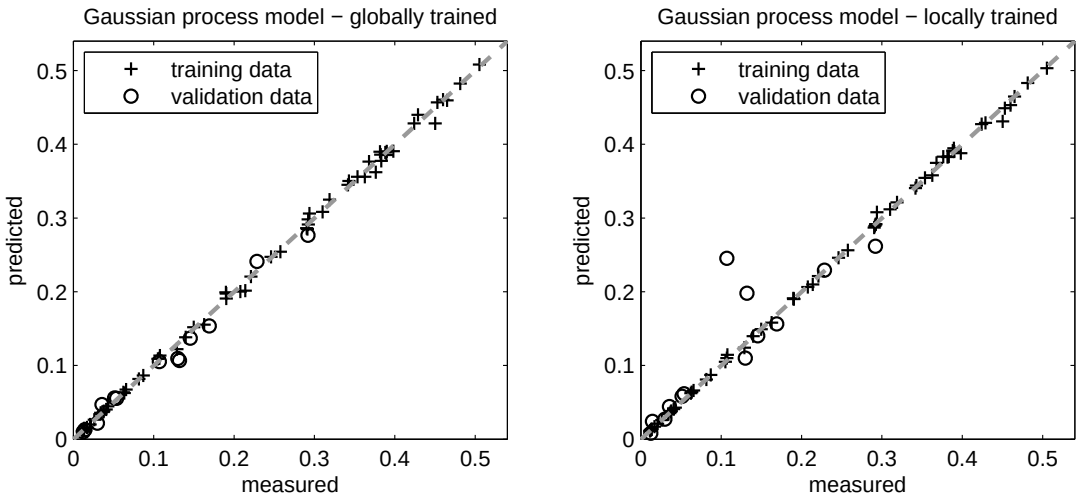


Figure 3.13: Comparison of a global Gaussian process model and a local Gaussian process model at the operating point at engine speed of 1525 1/min and engine torque of 780 Nm.

GP model has a better performance than the local polynomial models. But it can also be seen that the global model has a better performance on the validation data than the local one.
The reason for this lies in the fact that a global modeling can use all information of all measurements at once. Hence, the information from operating points, which are near to the considered operating point, can be incorporated by the global GP model. Therefore, all measurements contribute a maximum of information to the global model (REQ2), see also section 3.4.

3.7.4 Conclusion of the Practical Model Comparison

In this section, various theoretical thoughts of the sections 3.2 and 3.4 were illustrated on practical data sets.
We have seen that, even for low complex problems, the *GP approach performs better than the polynomial stepwise regression*, and that the advantages of GP modeling become more significant, if the complexity of the problem increases. Further, in the practical application above, we saw that a *global (GP) modeling performs better than a local modeling*, which can also be justified by the theoretical examinations in section 3.4. Hence, *we conclude that GP regression is more suitable than a local linear modeling for engine calibration tasks.*
Further, it could be seen that the squared exponential covariance function (2.44) is a suitable kernel function, which shows a good performance on practical problems of engine calibration, and therefore this kernel function will be used for all other practical examples in the further course of the thesis.
In addition, we demonstrated that the fully automatic Gaussian process approach has a very good performance on all data sets, and hence, this type of modeling is suitable for model-based online optimization in engine calibration, see section 6.

3.8 Drawbacks of Gaussian Processes

It seems to be a law of nature that the advantage of getting the most information out of the measurement data comes with the drawback of a high computational effort.

For the training of the Gaussian process, a nonlinear optimization of the hyperparameters is performed, where the likelihood function (2.49) is optimized. Hence, in every optimization step, the matrix inversion in (2.49) has to be performed. Since this matrix is of size $N \times N$, the computational effort of the training scales with $O(N^3)$. Although this matrix inversion is not performed explicitly, but rather a Cholesky factorization is calculated, obviously, this modeling will be computationally infeasible if the number of measurements is high, which is a significant limitation of Gaussian processes.
This drawback of a high training time can be slightly mitigated in model-based online optimization, where the hyperparameters do not have to be updated for every single new measurement. However, also the prediction requires a high computational effort. From (2.51) and (2.52) we see that, for the prediction of a single point $\mathbf{x}_*$, the kernel function $k(\mathbf{x}_*, \mathbf{x}_n)$ has to be evaluated N times (once for every single measurement $\mathbf{x}_n, n \in \{1, ..., N\}$).
Hence, on a standard PC this method could just be tested for $N = 10.000$. Clearly, Gaussian processes are not suitable for dynamic engine calibration, where the number of measurements is very high. But GP are appropriate for stationary base calibration, because in this application the number of measurements is rarely higher than 10.000.
For the application of Gaussian processes to quantities of a diesel engine in section 3.7, 780 measurements were considered, and the required time for training was about 20 seconds on a standard computer.

3.9 Conclusion and Discussion

In this chapter various different modeling techniques have been discussed. By considering the requirements of stationary base engine calibration, we were able to draw conclusions, which type of modeling is most suitable for this application.

The Gaussian process approach complies with all requirements which are listed in section 3.1, and therefore we recommend this technique for stationary base calibration.
Among other advantages of GP regression, this approach can be fully automated, and therefore it does not need any manual interaction from the user. Further, this technique shows the best performance on practical data sets, and even with a low amount of data, it is able to give an accurate prediction for the mean and variance.

However, there are other applications where it may be useful to differ from the requirements given above.
If the number of measurements is too high to calculate a full Gaussian process model, then other methods of nonlinear modeling can be suitable, like MLP networks, the RVM or the SVM.
If more measurements are available and a strong human interaction in the process is needed, then a local linear modeling may be helpful.
A polynomial stepwise regression should only be used for low complex problems.

Chapter 4

ROBUST GAUSSIAN PROCESS MODELING FOR ENGINE CALIBRATION

In the previous two chapters various different types of modeling have been discussed, and it was shown that the Gaussian process modeling is the most suitable one for stationary base engine calibration. Hence, we will concentrate on this approach in the further course of the thesis. However, there are still two important issues which often occur in practical applications and which are not yet considered by the modeling:

(ROB1) In practical data sets *the assumptions for the distributions of measurements and noise are often not met*. For our Gaussian process model, as well as for many other types of modeling, we assumed that the noise on the measurements is i.i.d. Gaussian distributed, see (2.45). In addition, from the definition of Gaussian processes it follows that the model is Gaussian distributed, see (2.46). However, as we will see soon, in many practical applications these assumptions are not met, especially when one is trying to model emissions. In order to improve the ability of the modeling to express the real behavior of the data, a *transformation of the measurements* will be presented.

(ROB2) In practical data sets *outliers often occur in the measurements*, especially in quantities which are hard to measure, as the soot emissions, which was already mentioned in section 3.7.2. These outliers are not considered by the modeling, since we assumed an i.i.d. Gaussian noise. Hence, as in section 3.7.2, these outliers have to be removed before model training, in order to get a good model quality and a good prediction. This has serious drawbacks because usually a manual interaction is needed to identify the outliers, since an automatic detection of the outliers is not very robust or computationally very expensive, if there are many outliers in the data, which we will see soon. In contrast to state of the art algorithms for engine calibration, a modeling based on Gaussian processes will be presented, which is *robust to outliers*.

In order to improve the model quality if these two problems, (ROB1) and (ROB2), occur, in this chapter the Gaussian process technique will be extended, and this type of modeling will be referred to as robust Gaussian process modeling in this thesis. Hence, the term 'robust modeling' refers to a modeling which has a good performance, even if the measurements and the noise are not i.i.d. normally distributed and outliers occur in the training data.

The idea of robust regression is far from new.
There exist several different types of transformation for non-normally distributed data, which have already been developed and used in engine calibration. Nevertheless, since these transformations suffer from some drawbacks, which will be extensively discussed in section 4.1, a new technique will be presented.
In addition, although there exists no modeling approach which is robust to outliers in engine calibration, in other fields of research, especially in statistics and machine learning, outlier-robust techniques have already been developed. However, as we will see soon, some additional work is required for a meaningful implementation for engine calibration.

The goal of this chapter is the development of a new robust modeling framework for stationary base engine calibration, which could be achieved by modifying state of the art approaches from other fields of research and by introducing new techniques.

In the next section a new nonlinear transformation will be presented. In the other following sections the formulas for a Gaussian process modeling, which is robust to outliers, will be derived. The discussion of this modeling is an extension of the results published in [8].

4.1 Nonlinear Transformation of the Data

As already discussed above, we assumed that the noise ϵ_n on the observed measurements $\mathbf{t}$ is i.i.d. normally distributed, see (2.45), and that our model $\mathbf{y}$ is Gaussian distributed, see (2.46). However, for some applications in engine calibration these assumptions might be inappropriate.
A common example is the modeling of emissions, e.g. NOx. In figure 4.1 (a) a typical distribution of NOx measurements is given. Due to reasons of confidentiality, the labeling of the x-axis is not shown. Nevertheless, it can be seen that a lot of measurements are taken at low NOx rates (typically near zero) and only a few measurements are taken at high NOx values. Since a space filling design was used for taking these measurements, it is, however, clearly very unlikely that these observations come from a system which is normally distributed. Hence, a Gaussian distributed model will not be appropriate for the approximation of this quantity.
In addition, the measurement noise often scales with the mean of the data, which is known as a special form of heteroscedasticity. Again, this is typically the case when one is measuring emissions. Usually, the measurement error at low emission values is much smaller than the measurement error at high emission values, and therefore it is common to estimate the measurement noise as a percentage of the mean value. However, as we assumed an identically distributed noise (homoscedasticity), the prediction of the variance will clearly not be a very good one [26].

If these problems occur, a transformation of the data might be suitable [26]. As the modeling should be fully automated in order to apply it to model-based online optimization, we seek for a method which automatically determines if it is appropriate to transform the data or if not.
In this thesis we suggest the following two-step approach for the transformation of the data,

which has been found to work very well in practice:

1. Nonlinear transformation of the data to a normal distribution (if necessary)

2. Linear (affine) transformation (normalization) of the data to zero mean and unit variance (if, after the first step, the data is normally distributed, then, after this linear transformation, the data will be standard normally distributed, so that $\mathbf{t} \sim \mathcal{N}(0, \mathbf{I})$)

The second (linear) transformation is performed due to our consideration of a zero mean, unit variance Gaussian process. On the one hand this simplification has been made for notational convenience, but on the other hand this procedure is useful, since we are able to choose meaningful starting points for the nonlinear optimization of the hyperparameters, which are invariant of the scaling of the measurements. As the linear transformation is a very simple (but very useful) and unproblematic technique, in this section the focus lies on the nonlinear transformation.

By far, the most commonly used nonlinear transformation is the Box-Cox transformation [13]. This transformation is parameterized through a coefficient $\tilde{\lambda}$ and is for $x > 0$ given by

$$f_{\text{BoxCox}}(x) := \begin{cases} \dfrac{x^{\tilde{\lambda}} - 1}{\tilde{\lambda}}, & \tilde{\lambda} \neq 0 \\ \ln(x), & \tilde{\lambda} = 0. \end{cases} \tag{4.1}$$

Since we do not only want to model in the transformed space, but we also want to give predictions in the origin space, we have to apply the inverse of the transformation to our model output. The inverse of the Box-Cox transformation is given by

$$f^{-1}_{\text{BoxCox}}(y) := \begin{cases} (\tilde{\lambda} \cdot y + 1)^{1/\tilde{\lambda}}, & \tilde{\lambda} \neq 0 \\ \exp(y), & \tilde{\lambda} = 0. \end{cases} \tag{4.2}$$

However, in order to calculate the inverse, y has to be in the image of f_{BoxCox}, which requires that $y > -\frac{1}{\tilde{\lambda}}$ for $\tilde{\lambda} > 0$ and $y < -\frac{1}{\tilde{\lambda}}$ for $\tilde{\lambda} < 0$. Exceptions are $\tilde{\lambda} = 1$ and $\tilde{\lambda} = 0$, which can always be inverted.
In many cases the constraint, that y has to be in the image of f_{BoxCox}, is unproblematic, since the model is often only interpolating the measurements and not extrapolating. However, in model-based online optimization we are usually seeking for an extremum, whose value is higher or lower than all other measurements.
Hence, the Box-Cox transformation is inappropriate for the use in engine calibration. Therefore, in [92] a different version of the Box-Cox transformation is developed, which can be inverted for all y values. This transformation requires two additional parameters $v_1, v_2 > 0$,

and is given by

$$f_{\text{Pol}}(x) := \begin{cases} (x^{\tilde{\lambda}} - 1)/\tilde{\lambda}, & \tilde{\lambda} > 0, x \geq v_1 \\ v_1^{\tilde{\lambda}-1} \cdot (x - v_1) + (v_1^{\tilde{\lambda}} - 1)/\tilde{\lambda}, & \tilde{\lambda} > 0, x < v_1 \\ \ln(x), & \tilde{\lambda} = 0 \\ (x^{\tilde{\lambda}} - 1)/\tilde{\lambda}, & \tilde{\lambda} < 0, x \leq v_2 \\ v_2^{\tilde{\lambda}-1} \cdot (x - v_2) + (v_2^{\tilde{\lambda}} - 1)/\tilde{\lambda}, & \tilde{\lambda} < 0, x > v_2. \end{cases} \tag{4.3}$$

The problem of this technique is the determination of suitable values for v_1 and v_2. In [92] the parameters are partially fixed to $v_1 = 0.1$ and $v_2 = 2.5$, but this arbitrary choice can be inappropriate in many applications. In addition, we seek for an approach which is fully automatic, since we want to use the modeling in model-based online optimization.
In contrast, it has been found to be very useful to transform the data by a simple logarithm. This is due to the fact that in engine calibration the distribution of the measurements is either symmetric, e.g. consumption, temperature, etc., and therefore no transformation is needed, or the distribution of the measurements is right-tailed, e.g. emissions, as the already discussed NOx measurements in figure 4.1 (a), and therefore a transformation with the logarithm can give good results, see below. However, as we will see soon, a left-tailed distribution cannot be meaningfully transformed with a logarithm, but this problem never occurred in practical tasks in engine calibration. Further, as the measurement noise often scales with the mean of the data, except for an additive constant, an identically distributed noise can be obtained with a logarithmic transformation [26].
Hence, we suggest the following transformation for engine calibration

$$f_{\text{nTrans}}(x) := \ln(x + \tilde{\lambda}) \tag{4.4}$$

which is also parameterized through a coefficient $\tilde{\lambda}$. In order to perform this transformation, $x > -\tilde{\lambda}$ has to be satisfied. The inverse is given by

$$f_{\text{nTrans}}^{-1}(y) := \exp(y) - \tilde{\lambda} \tag{4.5}$$

which can be calculated for all $y \in \mathbb{R}$, and therefore the problem of the inversion, as in the Box-Cox transformation (4.1), does not appear in this approach. The parameter $\tilde{\lambda}$ is important, since it controls the degree of nonlinearity of the transformation. It can be determined through a maximum likelihood consideration, which is similar to the estimation of the Box-Cox parameter of (4.1) published in [13] and [43]. If we want that the transformed measurements are normally distributed around a constant mean value, then the probability density function for the untransformed observations, and hence the likelihood in relation to these original observations, is given by

$$\frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{(\tilde{\mathbf{x}} - \text{mean}(\tilde{\mathbf{x}}))^T(\tilde{\mathbf{x}} - \text{mean}(\tilde{\mathbf{x}}))}{2\sigma^2}\right) \cdot \left(\prod_{n=1}^{N} \left|\frac{d\tilde{x}_n}{dx_n}\right|\right) \tag{4.6}$$

where we have defined the vector $\tilde{\mathbf{x}}$ with elements $\tilde{x}_n = f_{\text{nTrans}}(x_n)$ and the variable $\sigma := \frac{1}{N-1}(\tilde{\mathbf{x}} - \text{mean}(\tilde{\mathbf{x}}))^T(\tilde{\mathbf{x}} - \text{mean}(\tilde{\mathbf{x}}))$. The last term in (4.6) is the Jacobian of (4.4). Now we seek for the parameter $\tilde{\lambda}_*$, which maximizes the likelihood. This value can be found by maximizing the log likelihood L_{nTrans} of (4.6), which is given by

$$L_{\text{nTrans}}(\tilde{\lambda}) = -\frac{N}{2}\ln(\sigma^2) - \sum_{n=1}^{N}\ln(x_n + \tilde{\lambda}) \tag{4.7}$$

with $\sigma^2 = \sigma^2(\tilde{\lambda})$ as above, and where we have neglected constant terms and insert the Jacobian of (4.4).

In addition, we suggest to evaluate the skewness of the distribution of the measured data, which can be determined by

$$\frac{N}{(N-1)(N-2)}\sum_{n=1}^{N}\left(\frac{x_n - \text{mean}(\mathbf{x})}{\sigma}\right)^3. \tag{4.8}$$

If the skewness is greater than 1 and if $\frac{\max(\mathbf{x})}{\min(\mathbf{x})} > 20$, as suggested in [123], then the nonlinear transformation is performed.

An example of an application of the nonlinear transformation is given in figure 4.1. In figure 4.1 (a) the distribution of NOx measurements is drawn. Due to reasons of confidentiality, the

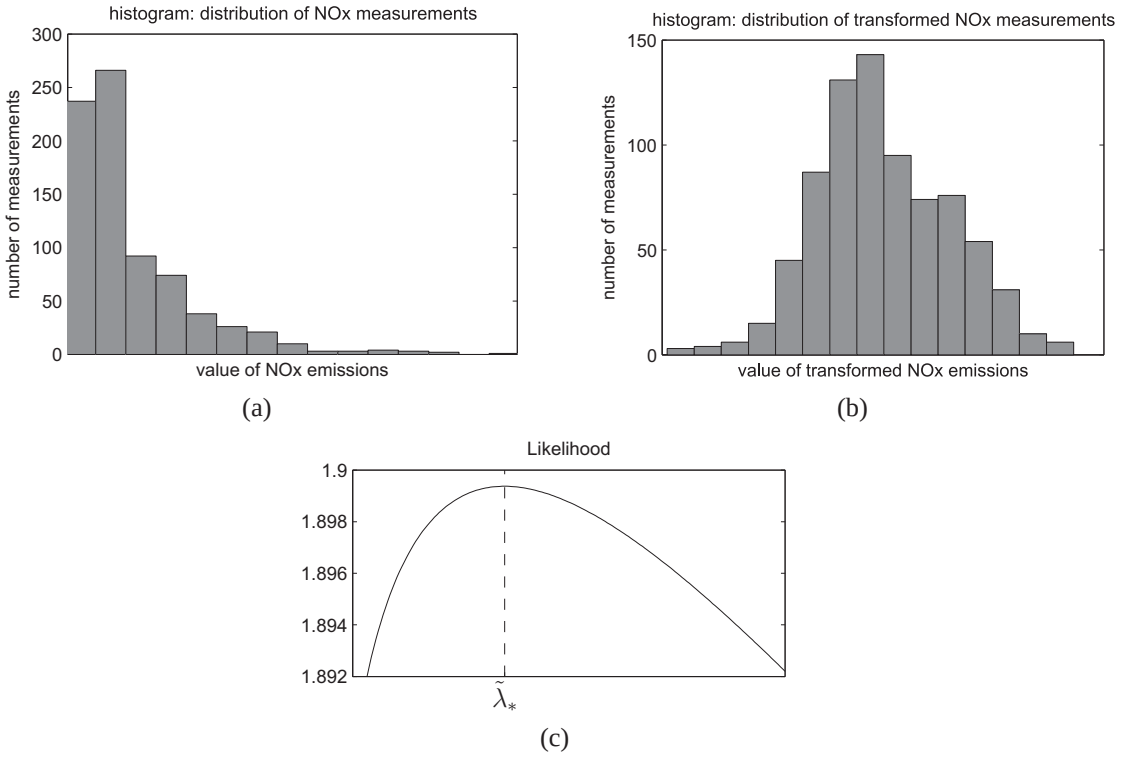


Figure 4.1: Example of a nonlinear transformation of NOx measurements.

labeling of the x-axis is not shown. It clearly can be seen that the distribution is right-tailed. Hence, the nonlinear transformation is applied to these measurements and the results are drawn in figure 4.1 (b). For this transformation the log likelihood (4.7) was maximized, which is shown in figure 4.1 (c), where the optimal $\tilde{\lambda}$ value, $\tilde{\lambda}_*$, is marked.
Clearly, the transformed distribution in figure 4.1 (b) is not perfectly Gaussian shaped, but it is much better than the original distribution. Hence, our model assumptions are much more appropriate in the transformed space than in the origin one.
An interesting property of this approach becomes obvious, if it is applied to measurements, which are already Gaussian distributed. For this data, the $\tilde{\lambda}_*$ value which maximizes the log likelihood (4.7) will tend towards very high values (infinity). This will lead the transformation (4.4) to tend towards a linear (affine) transformation, and hence the shape of the distribution is not changed. Therefore, we reject the nonlinear transformation, if the parameter $\tilde{\lambda}$ tends towards high values, e.g. $10 \cdot \max(\mathbf{x})$.

As said above, only distributions which are right-tailed can be meaningfully transformed with (4.4). If the data would be left-tailed distributed, then a meaningful transformation with (4.4) could be achieved by negating x, so that $\tilde{f}_{nTrans}(x) := f_{nTrans}(-x)$, but this procedure was not necessary, since only right-tailed data occurred in practical problems in engine calibration.

Although the transformation (4.4) is very simple, in this work it has been found that this approach achieves very good results for engine calibration tasks, as in the example of the NOx measurements above. Further, due to the maximum likelihood estimation of the parameter $\tilde{\lambda}$, this technique can be fully automated, and therefore, it can be used for model-based online optimization.

4.2 A Student's-t likelihood

As discussed in chapter 2, the training for polynomials, RBF networks, LLR models, local linear models and MLP networks is performed by minimizing the SSE function (2.6) or the RSSE function (2.15). This minimization is equivalent to the maximization of the likelihood function or the a posteriori distribution under a Gaussian noise assumption, see sections 2.1.1 and 2.1.2.1. In comparison to that, the relevance vector machines and the (conventional) Gaussian processes directly use a normal noise assumption (2.45) for modeling in engine calibration. Hence, all these state of the art approaches assume a normal noise[1].

As we will see shortly, this Gaussian likelihood is not robust to outliers. Hence, we have to

[1] An exception is the support vector machine, which minimizes an $\tilde{\epsilon}$-insensitive error function (2.56). The equivalent likelihood of this error function is given by $\frac{1}{2(1+\tilde{\epsilon})} \exp(-E_{\tilde{\epsilon}}(y_n - t_n))$, see [114], which converges to the Laplacian distribution in the limit of $\tilde{\epsilon} \to 0$. This distribution is more robust to outliers than a normal noise assumption [96], but it has no additional parameter ν, as the Student's-t likelihood. Therefore, due to this lack of degree of freedom, the results of this modeling will not be as good as the following robust formulation, which will be discussed shortly. Hence, in order to abbreviate this section and due to the general drawbacks of SVM's, which were discussed extensively in section 3.6, we will not go into detail into this examination.

replace this noise assumption with another distribution. Typical likelihoods, which are robust to outliers, are the Laplacian, the cumulative logistic and the Student's-t distributions [96]. During this work all these likelihood functions had been examined and tested.
The Student's-t likelihood is given by

$$\mathrm{St}(t_n, y_n, \sigma, \nu) := \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi}\sigma} \left(1 + \frac{(t_n - y_n)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}, \tag{4.9}$$

where ν is the number of degrees of freedom, σ is the scale parameter, influencing the variance of the distribution, and Γ is the gamma function. If the parameter ν tends towards infinity, then the Student's-t distribution tends towards the Gaussian distribution. Hence, the Student's-t distribution is a generalization of the Gaussian distribution, and this is also a very advantageous property of this likelihood function. As we will see shortly, by controlling the parameter ν, we are able to control the outlier-robustness of our model.
This is in comparison to the Laplacian and the cumulative logistic likelihood, which have only a single parameter for controlling the variance of the distributions, but no additional parameter which controls the outlier-robustness. This behavior is somewhat equivalent to a Student's-t distribution with a fixed value of ν. Hence, with the Laplacian and the cumulative logistic likelihood we restrict the flexibility of our model, and we cannot converge to a solution with a Gaussian distributed noise, as it is the case for the Student's-t likelihood.
Therefore, in this thesis, we chose the Student's-t distribution for the noise assumption in our robust modeling.

In figure 4.2 the Student's-t distribution is compared to the Gaussian distribution and the advantages of a Student's-t likelihood, when it comes to outliers, is shown.

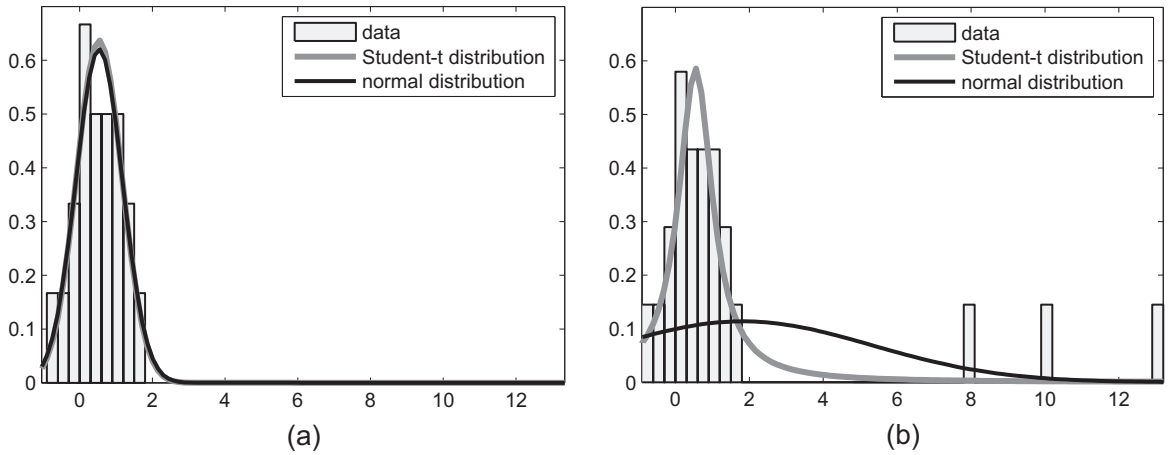


Figure 4.2: Comparison of a normal noise assumption and a Student's-t noise assumption, without outliers (a) and with outliers (b), based on [11].

In the plots of figure 4.2, data is sampled and a normal distribution and a Student's-t distribution is fitted by the method of maximum likelihood. In the left plot (a) no outliers occur. As the Student's-t distribution converges to a normal distribution, as the parameter ν increases

towards infinity, both distributions have the same shape under maximum likelihood, if the data is normally distributed. Hence, if there are no outliers in the data, our new modeling will converge to the conventional state of the art Gaussian process modeling as discussed in section 2.4.2. As discussed above, this will be an advantageous property of our robust modeling, since in many applications the normal noise assumption will be appropriate, as for engine quantities which are relatively easy to measure, see e.g. the NOx and consumption measurements in section 3.7.1.
The drawbacks of a Gaussian distribution regarding outliers can be seen in plot 4.2 (b). In this plot three outliers have been added at the right edge. It clearly can be seen that the outliers strongly distorted the shape of the Gaussian distribution, while the shape of the Student's-t distribution is widely unaffected, since the parameter ν was adapted through the method of maximum likelihood.
For a deeper discussion on the robustness of the Student's-t distribution regarding outliers see [11, 90, 126].

Hence, the assumption of a normally distributed noise can be a serious drawback of recent types of modeling in engine calibration. If a Gaussian likelihood is used for modeling and outliers occur in the data, then the prediction of the model will be distorted by the outliers. Therefore, in order to obtain a good model quality, all outliers have to be removed before model training in state of the art algorithms.
For these approaches with a normal noise assumption, a common strategy to detect and remove outliers in an automatic way is to perform the modeling with the whole data set, calculate the residuals between the model and the targets, remove the measurements with the highest residuals and then perform the modeling with the remaining data again [107, 124]. With this approach, it is hoped that after some iterations all outliers have been removed from the data set and that the model will finally provide a good prediction. Obviously, this method will be suitable if the data set is small and if only a few outliers are contained in the training set. But clearly, this approach will be inappropriate for bigger data sets with more outliers, since the modeling has then to be performed numerous times, which can be computationally infeasible for models in which a single training is itself computationally expensive. We will discuss this topic again, when applications with practical data sets are considered, as in the sections 4.5 and 4.6.
Therefore, in state of the art algorithms for model-based offline optimization, the outliers are often removed manually, see [24, 37, 62, 104, 107]. Clearly, outliers in model-based online optimization would cause a serious problem, because no manual interaction is possible, and hence wrong predictions will cause the optimization routine to perform useless measurements in undesired regions, see also requirement (REQ5) in section 3.1. Thus, state of the art online optimization is only performed for quantities of an engine which are relatively easy to measure, as consumption, see [53], and not for quantities where the risk of outliers is much higher, like soot, see section 3.7.2.

A possible solution to these problems will be presented in the next sections, where the normal noise assumption will be replaced by a Student's-t noise assumption, in order to achieve a modeling which is robust to outliers.

More mathematically spoken, for a limited fraction of outliers, we will develop a modeling which is asymptotically unbiased, meaning that the model is tending to the real engine behavior as the number of measurements tends towards infinity. As the number of outliers usually scales with the number of measurements, it should be noted that a modeling with a normal noise assumption does not possess this property, since every outlier will result in a bias in the model [126], which we will demonstrate in the later sections.

But before that, some remarks on the importance of the combination of the nonlinear transformation (4.4) and outlier-robust modeling are given.
As discussed above, by considering figure 4.1 (a), it can be seen that most of the data is located at low NOx rates and only very few measurements had been made at high NOx values. If we compare this behavior of the untransformed data to the outlier-robust behavior of the Student's-t distribution in figure 4.2 (b), it should become clear that it is very likely that an outlier-robust modeling will reject the high NOx values of the plot 4.1 (a), since the model assumptions are not appropriate in the untransformed space. Therefore, only through performing the transformation (4.4), an outlier-robust modeling can meaningfully approximate the high NOx values.
Hence, only through the combination of the nonlinear transformation and the outlier-robust model, which is discussed in the next sections, a new robust modeling framework for stationary base engine calibration can be obtained, and only the whole framework can be used reliably for an automatic online optimization.

4.3 An Outlier-Robust Gaussian Process Modeling for Engine Calibration

As already discussed above, although there exists no modeling approach which is robust to outliers in engine calibration, in other fields of research, especially in statistics and machine learning, the idea of robust regression is far from new. Outlier rejection from a Bayesian perspective was already analyzed in 1961 by [18]. A Student's-t noise assumption for linear regression was already studied by [32, 131], and [82] introduced a Student's-t noise assumption for Gaussian process regression. Other robust formulations could be achieved, for example, with mixtures of Gaussians, the Laplacian or the cumulative logistic distribution, e.g. see [61, 96], which are not considered here, as discussed above.
In this section different approaches of GP regression with a Student's-t likelihood will be discussed, an approach which is suitable for engine calibration will be chosen, the remaining problems of this technique will be examined, and a solution which is appropriate for online optimization will be given.
In order to keep the derivation of the formulas of the robust modeling clearly arranged, we will have to repeat some equations of chapter 2.

As in section 2.4.2.3, in order to apply Gaussian processes for regression, we need to consider the noise ϵ_n on our measurements t_n of the engine, which are given by (2.1). As said above,

an i.i.d. Student's-t noise assumption is made for ϵ_n, in order to achieve a robust formulation, so that

$$p(\mathbf{t}|\mathbf{y}, \sigma, \nu) := \prod_{n=1}^{N} \text{St}(t_n, y_n, \sigma, \nu), \tag{4.10}$$

where St is the Student's-t distribution (4.9). We follow the definition 2.1 of Gaussian processes, and as in section 2.4.2.3, we will consider a zero-mean Gaussian process, so that

$$p(\mathbf{y}|\mathbf{\Theta_K}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}), \tag{4.11}$$

where we use $\mathbf{K} = \mathbf{K}(\mathbf{\Theta_K})$ the Gram (covariance) matrix (2.41) of the squared exponential kernel (2.44) with the hyperparameters $\mathbf{\Theta_K} := \{\theta_\sigma^2, \theta_{l,1}, ..., , \theta_{l,D}\}$ for modeling. For notational simplicity, we collect all the hyperparameters into a single vector of hyperparameters $\mathbf{\Theta} := \{\mathbf{\Theta_K}, \mathbf{\Theta}_{\text{St}}\}$, where $\mathbf{\Theta}_{\text{St}} := \{\sigma, \nu\}$ contains the hyperparameters of the Student's-t distribution.

In the next section we will examine a training algorithm for this modeling. In order to find an appropriate one, we have to consider some requirements of the application of engine calibration. In section 4.3.2, a suitable approximation for the predictions of the model is given.

4.3.1 Training

As discussed in section 2.4.2.3, in the Gaussian process viewpoint, training is performed by inferring the hyperparameters $\mathbf{\Theta}$ out of the training data. As for the conventional GP models, we can find suitable hyperparameters for our robust model through maximizing the marginal likelihood

$$p(\mathbf{t}|\mathbf{\Theta}) = \int p(\mathbf{t}|\mathbf{y}, \mathbf{\Theta}_{\text{St}}) p(\mathbf{y}|\mathbf{\Theta_K}) \text{d}\mathbf{y}. \tag{4.12}$$

If we would use a normal noise assumption, then the integral can be calculated by using standard formulas, see section 2.4.2.3. But since we use a Student's-t distribution (4.10) for $p(\mathbf{t}|\mathbf{y}, \mathbf{\Theta}_{\text{St}})$, this integral becomes analytically intractable.
Hence, we need to approximate (4.12). In literature this is referred to as approximate inference.

4.3.1.1 Methods for Approximate Inference

State of the art algorithms for approximation of (4.12) are Markov Chain Monte Carlo (MCMC) methods (see [80] for a review), the expectation propagation (EP) algorithm (see [73]), the factorized variational approximation (VB) (see [61] and [120]) and the Laplace approximation (see [126] and [68]).
With MCMC methods we are able to approximate (4.12) to arbitrary accuracy, and therefore

these techniques achieve the best results, but they are also computationally very expensive, see [85]. As the focus of this work is an algorithm which can be used for model-based online optimization, as said above, the computational cost is an important factor, and since the MCMC approximation requires about 500 times more computational effort as e.g. the Laplace approximation [85], the MCMC methods cannot be used.
The EP algorithm has been proven to be a good method for approximate inference in many applications, but here the use of EP is problematic, since the Student's-t likelihood is not log-concave and accurate approximate inference with EP is very hard due to posterior multimodality, see [112] for details. [126] shows, that the performance of the Laplace approximation is slightly better than the performance of the factorial VB. In addition, the Laplace approximation is clearly faster than the factorial VB [85, 126].
Hence, in this thesis we concentrate on the Laplace approximation.

4.3.1.2 The Laplace Approximation of the Marginal Likelihood $p(\mathbf{t})$

The Laplace approximation for Gaussian processes has extensively been used for classification problems [98]. In [126] this approach is applied to GP regression with a Student's-t likelihood, and during this work it has been found that this technique also works well for engine calibration problems.
For more general information on the Laplace approximation see [68].

In the framework of the Laplace approximation, we seek for a Gaussian approximation for $p(\mathbf{t}|\mathbf{y})p(\mathbf{y})$ in order to approximate (4.12), where, again for notational convenience, we suppressed the dependence on $\mathbf{\Theta}_{\text{St}}$ and $\mathbf{\Theta}_{\mathbf{K}}$. This can be achieved by introducing

$$\psi(\mathbf{y}) := \ln p(\mathbf{t}|\mathbf{y}) + \ln p(\mathbf{y}), \tag{4.13}$$

seeking the mode $\tilde{\mathbf{y}}$ of $\psi(\mathbf{y})$

$$\tilde{\mathbf{y}} := \arg\max_{\mathbf{y}} \psi(\mathbf{y}) \tag{4.14}$$

and the Laplace approximation results in

$$\begin{aligned} p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) &\approx \mathcal{N}(\mathbf{y}|\tilde{\mathbf{y}}, \mathbf{A}^{-1}) \\ &= \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}(\mathbf{y}-\tilde{\mathbf{y}})^T\mathbf{A}(\mathbf{y}-\tilde{\mathbf{y}})\right) \end{aligned} \tag{4.15}$$

where $\mathbf{A}$ is the negative Hessian of $\psi(\mathbf{y})$ at the mode $\tilde{\mathbf{y}}$

$$\mathbf{A} := -\nabla\nabla\psi(\tilde{\mathbf{y}}) = -\nabla\nabla \ln p(\mathbf{t}|\tilde{\mathbf{y}}) + \mathbf{K}^{-1}. \tag{4.16}$$

Using (4.9) and (4.10) we can calculate $-\nabla\nabla \ln p(\mathbf{t}|\mathbf{y}) =: \mathbf{W}$ as

$$(\mathbf{W})_{n,n} = -(\nu+1)\frac{r_n^2 - \nu\sigma^2}{(r_n^2+\nu\sigma^2)^2}, \tag{4.17}$$

where $r_n := (t_n - y_n)$ and $(\mathbf{W})_{n,m} = 0$ if $n \neq m$.
With the Laplace approximation (4.15), we can now determine (4.12) as

$$\ln p(\mathbf{t}|\mathbf{\Theta}) \approx \ln q(\mathbf{t}|\mathbf{\Theta}) = \ln p(\mathbf{t}|\tilde{\mathbf{y}}) - \frac{1}{2}\tilde{\mathbf{y}}^T\mathbf{K}^{-1}\tilde{\mathbf{y}} - \frac{1}{2}\ln|\mathbf{B}| \tag{4.18}$$

with

$$\mathbf{B} := \mathbf{K}\mathbf{A} = \mathbf{I} + \mathbf{W}^{1/2}\mathbf{K}\mathbf{W}^{1/2} \tag{4.19}$$

by using standard formulas given in [98]. Here, $q(\mathbf{t}|\mathbf{\Theta})$ stands for an approximation of $p(\mathbf{t}|\mathbf{\Theta})$.

4.3.1.3 Optimization of the Hyperparameters

As said above, with the approximation (4.18) of the marginal likelihood (4.12), the hyperparameters $\mathbf{\Theta}$ can be optimized on the training data. For this task a L-BFGS-B optimization has been used, which is a limited-memory quasi-Newton code for bound-constrained optimization, see section 5.1 and [15] and [135]. For this method the derivatives of (4.18) w.r.t. the hyperparameters $\mathbf{\Theta}$ are needed. Since the calculation of these derivatives is somewhat lengthy, only the results are presented here. If θ_j is a single element of $\mathbf{\Theta}$, then the derivatives are given by

$$\begin{aligned}\frac{\partial \ln q(\mathbf{t}|\mathbf{\Theta})}{\partial \theta_j} =& \frac{\partial \ln p(\mathbf{t}|\tilde{\mathbf{y}})}{\partial \theta_j} + \frac{1}{2}\tilde{\mathbf{y}}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{K}^{-1}\tilde{\mathbf{y}} - \frac{1}{2}\operatorname{trace}\left((\mathbf{W}^{-1}+\mathbf{K})^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \\ & - \frac{1}{2}\operatorname{trace}\left((\mathbf{K}^{-1}+\mathbf{W})^{-1}\frac{\partial \mathbf{W}}{\partial \theta_j}\right) \\ & - \frac{1}{2}\sum_{n=1}^{N}\left[(\mathbf{K}^{-1}+\mathbf{W})^{-1}\right]_{nn}\frac{\partial^3 \ln p(\mathbf{t}|\tilde{\mathbf{y}})}{\partial \tilde{y}_n^3}\left[\mathbf{B}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\nabla \ln p(\mathbf{t}|\tilde{\mathbf{y}})\right]_n .\end{aligned} \tag{4.20}$$

Various aspects of a suitable GP implementation for classification, which are given in [98], can also be applied to our problem. Additional numerical aspects of the Laplace approximation for GP regression can be found in [126] and [96].

4.3.1.4 Implementation for Engine Calibration

With the optimization of the hyperparameters on the marginal likelihood, the GP regression results in a fully automatic approach. For this robust modeling some implementations, as [126] and [96], already exist. However, there are still some open problems which result from the application in engine calibration.

By applying real data from a combustion engine to the implementation, often only a very poor result could be achieved. The reason for this is, that due to the additional parameter ν of the

Student's-t distribution, the marginal likelihood (4.18) often has numerous different local optima for real data sets. Hence, in order to get good results, a good choice of the initial values $\mathbf{\Theta}_{\text{Init}}$ for the hyperparameters is crucial, since otherwise the optimization routine will often get stuck in a bad local optimum.
Therefore, one possible solution would be a multistart optimization, where many initial values are used and the most reasonable model is chosen in the end. But as it is said above, computational speed is an important factor and therefore this procedure cannot be used.

In order to avoid these computationally intensive methods, we can use a strategy which has been found to work very well in practical problems in engine calibration. Let us first consider an application where only a few outliers are contained in the data. If we use a normal noise assumption for modeling, then the outliers will distort the prediction of the model, but we will get a good guess for the parameters $\mathbf{\Theta}_{\mathbf{K}}$, since only a few outliers are in the data set. If the number of outliers is increased, then the estimated parameters $\mathbf{\Theta}_{\mathbf{K}}$ might get shifted from the real values, due to the wrong noise assumption. But clearly, if the number of outliers is not too big, then this displacement will be a small one. Hence, it may be a good initial guess for a later optimization with a Student's-t noise assumption.
Therefore, the following solution was implemented: First, a training of a Gaussian process with a normal noise assumption is performed. This model assumes that there are no outliers in the training data. Then the hyperparameters of this model, $\mathbf{\Theta}_{\text{NN}}$, are taken as the initial values for the hyperparameters, $\mathbf{\Theta}_{\text{Init}}$, for the robust Gaussian process optimization and the initial value for ν is chosen to be relatively high.
In practical applications, like in the sections 4.5 and 4.6, very good results have been achieved with this method at low computational costs. With this approach, the training for a robust Gaussian process model requires approximately twice as much computing time as the conventional GP model.

4.3.2 Prediction

In this section we want to predict the value y_* of our Gaussian process model at a new input location $\mathbf{x}_*$.
If we use the Laplace approximation a second time, then the approximate distribution of y_* will also be Gaussian, and therefore defined by its mean and variance, which are given by [98]

$$\mathbb{E}_q[y_*|\mathbf{t}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \tilde{\mathbf{y}} \tag{4.21}$$

$$\mathbb{V}_q[y_*|\mathbf{t}, \mathbf{x}_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}(\mathbf{x}_*) \tag{4.22}$$

These results are being achieved by finding an approximation $q(\mathbf{y}|\mathbf{t}, \mathbf{\Theta})$ for $p(\mathbf{y}|\mathbf{t}, \mathbf{\Theta})$. For more details see [98].
Instead of going into more detail in the derivation of the formulas, a focus in this thesis lies on the discussion of the most important properties of this new robust approach and on the illustration of the advantages for engine calibration. These tasks are performed in the next sections.

4.4 A Simple Theoretical Example

In this section a simple theoretical example of a Gaussian process regression is examined, where a normal noise assumption and a Student's-t noise assumption are compared, which is illustrated in figure 4.3.

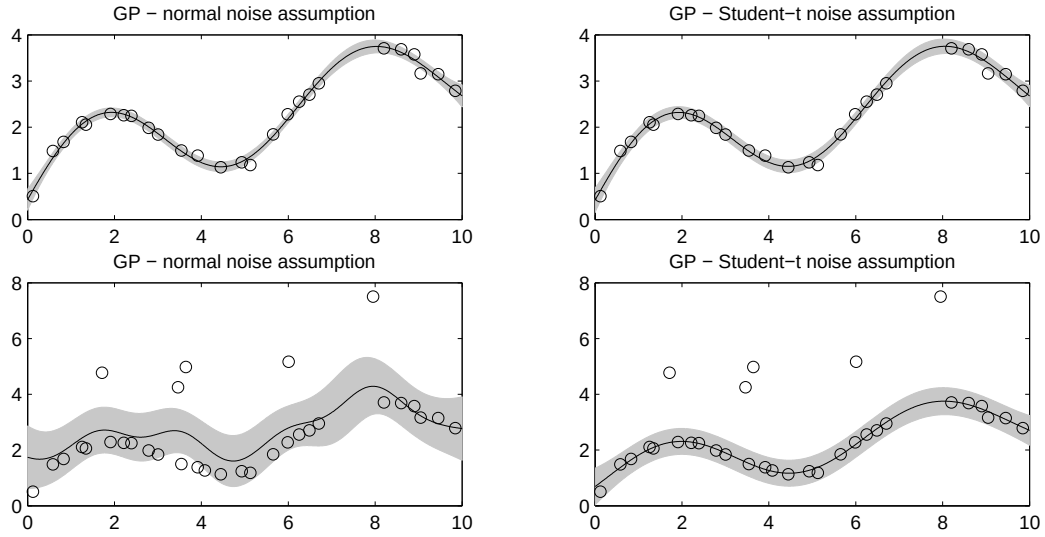


Figure 4.3: Comparison of Gaussian process regression with normal noise assumption (left) and Student's-t noise assumption (right), without outliers (top) and with outliers (bottom). The training data (circles), the predicted mean (solid line) and the predicted variance (confidence interval) are plotted.

From the function

$$\sqrt{x} + \sin(x), \tag{4.23}$$

which in practice could be any nonlinear engine mapping, training data (circles) is sampled and shifted by random noise. With this data Gaussian process models were calculated. The predicted mean (solid line) represents the estimated function value, and with the predicted variance a 95% confidence interval can be calculated, which represents the degree of certainty where the estimated function is expected.

If the noise on the measurements is normally distributed, as in the top row, both models give the same result, since, during the optimization of the hyperparameters $\boldsymbol{\Theta}$, ν increases towards infinity and the Student's-t distribution converges to a normal distribution. This should be compared to figure 4.2 (a). If outliers occur in the training data, as in the bottom row, then the prediction of a modeling with a normal noise assumption will get biased in the neighborhood of the outliers, while a modeling with a Student's-t noise assumption will be widely unaffected by the outliers, which should also be compared to figure 4.2 (b).

4.5 A Practical Example on a Diesel Engine

In this section a Gaussian process model with a normal noise assumption is applied on NOx and soot measurements of a diesel engine and compared to a Student's-t noise assumption.

In this application only a single operation point is considered and only local models are trained. The inputs of the models are the main injection time, injection pressure, quantity and time of the pre-injection and quantity of exhaust gas recirculation. This leads to a 5 dimensional input space. For reasons of confidentiality all measurements are scaled to an interval of $[0\ 1]$.

From a total set of 279 measurements, 35 measurements are randomly removed for model validation and the remaining 244 are used for training. With this data a Gaussian process model had been trained with a normal noise assumption and a Student's-t noise assumption. The performance of these models is shown in the measured-predicted plots in figure 4.4 and the NRMSE of these plots are given in table 4.1.

Since the NOx emissions (top row) can be measured relatively well, no outliers occur in the measurement data and the model with the Student's-t noise assumption (right) gives pretty

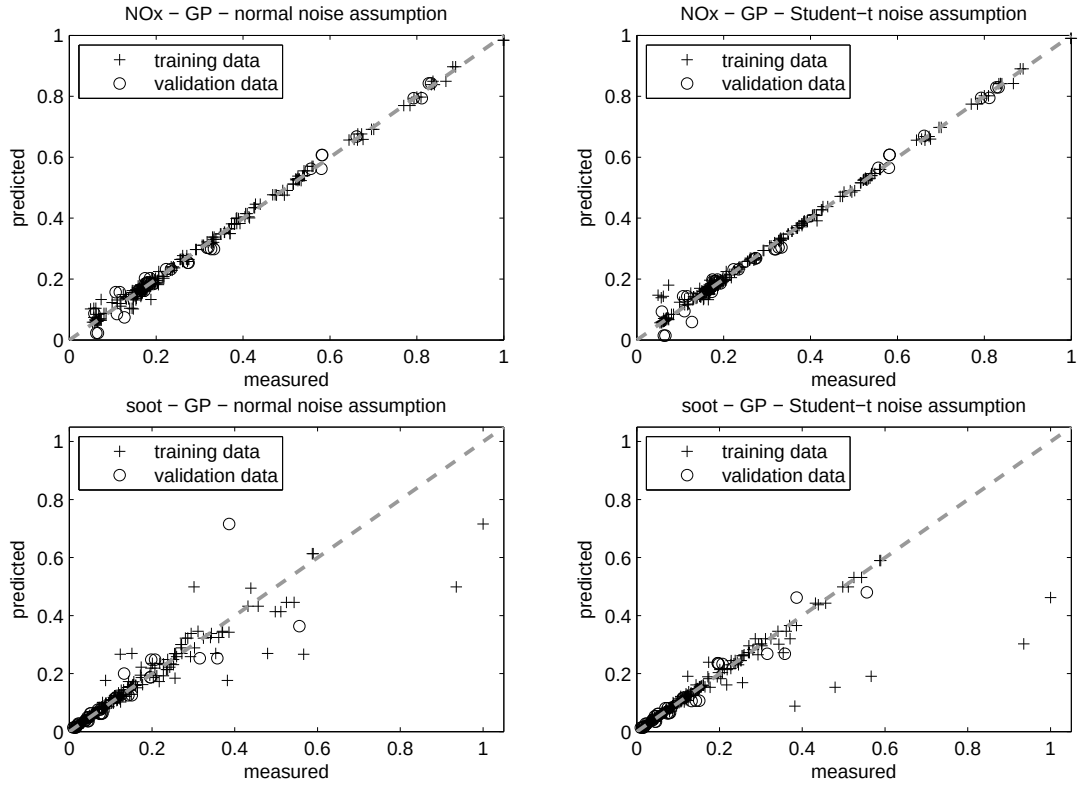


Figure 4.4: Measured-predicted plots for training and validation data of NOx (top) and soot (bottom) emissions with normal noise assumption (left) and Student's-t noise assumption (right).

		training data	validation data
NOx	normal noise	1.330 %	2.865 %
	Student's-t noise	1.471 %	2.854 %
soot	normal noise	5.069 %	9.926 %
	Student's-t noise	6.687 %	5.565 %

Table 4.1: NRMSE of NOx and soot for training and validation data of figure 4.4.

much the same result as the model with the normal noise assumption (left). The performance of both models for NOx is quite good, but it should be noted that this is not the case for the soot emissions (bottom row). Since the soot emissions are much harder to measure, see section 3.7.2, outliers occur in the measurement data. These outliers will distort the model with the normal noise assumption (left). This is a serious problem with state of the art models for engine calibration. It is very hard to determine which of these measurements is an outlier and which prediction is only biased by outliers. If one would calculate a model with more inputs, an even higher nonlinearity and more measurements, this problem would become even more severe, since the number of outliers usually increases with the number of measurements.

It should be noted that this is not a problem if one uses a Gaussian process modeling with a Student's-t noise assumption (bottom right). With this modeling a better fit on the training data is achieved and the prediction of the validation data is very accurate. Further, it is easy to determine the five outliers in the data, which can be measured again if required. It should be noted that, due to the fact that all the outliers of the soot emissions are in the training data, the NRMSE of the training data is higher than the NRMSE of the validation data with the Student's-t noise assumption in table 4.1.

4.6 Examination of the Potential of Robust GP Regression

After these short theoretical and practical examples, in this section we want to gain a deeper insight into the robustness of this new modeling approach, and therefore the limits of this technique will be examined. Even if there are many outliers in the data, it will be illustrated that one can be confident in the robust GP regression, from which it follows that this new technique will provide a substantial advantage for engine calibration tasks.

First, the theoretical example in figure 4.5 is considered. In this example the function (4.23) from section 4.4 is regarded. For the training data (circles), 30 data points were sampled from this function and shifted by random noise. In addition, outliers have been added to the training data. As in the theoretical example above, with this data Gaussian process models with a normal noise assumption (left) and a Student's-t noise assumption (right) were calculated and the predicted mean (solid line) and the predicted variance (confidence interval) are plotted.
In the different rows of figure 4.5, the number of outliers have been varied. In the top row 6 outliers are contained in the training data, the second row contains 9 outliers, the third row contains 10 outliers, the fourth row contains 11 outliers and the last row contains 34 outliers, which are more than the 30 correct data points.

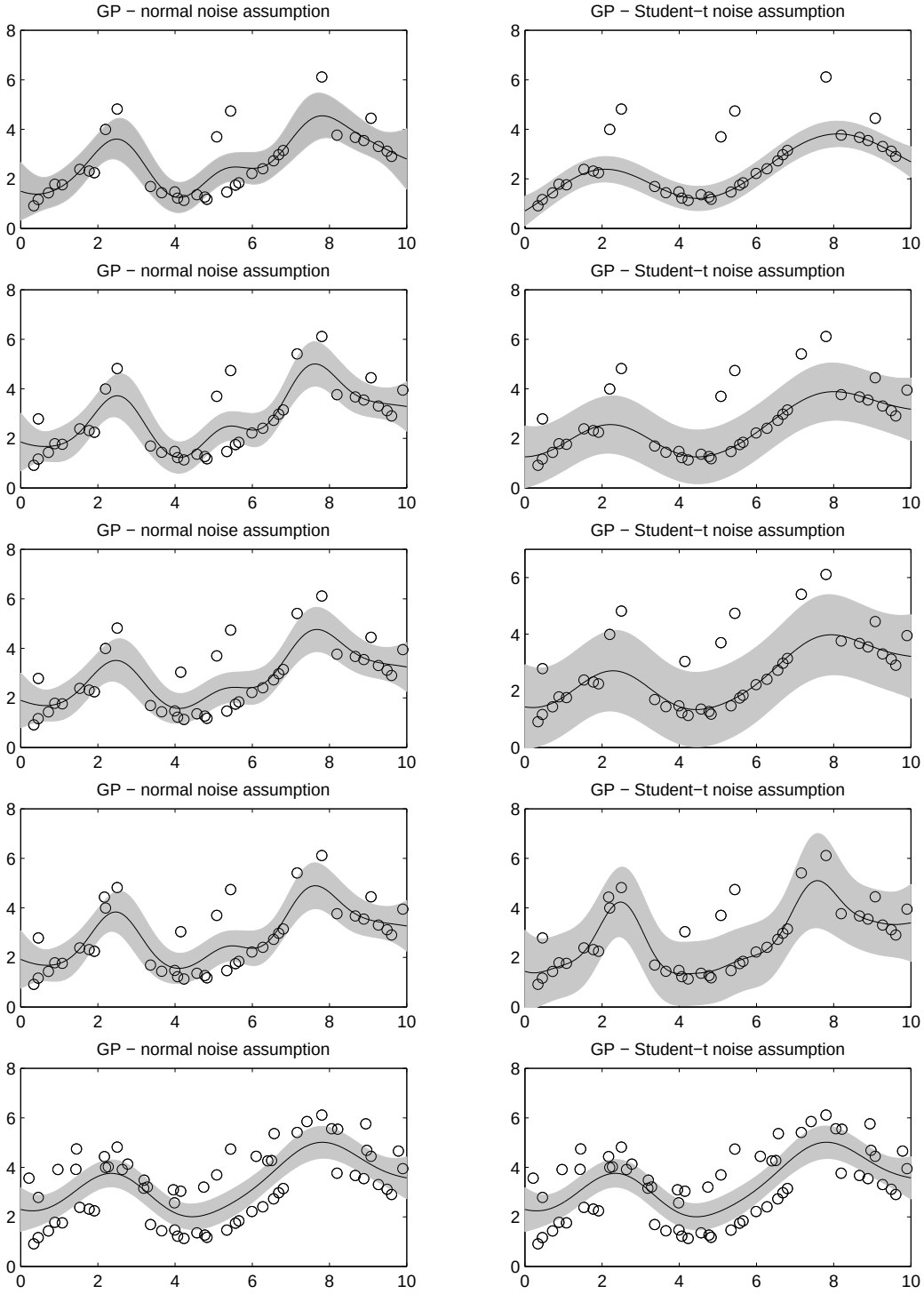


Figure 4.5: Comparison of Gaussian process regression with normal noise assumption (left) and Student's-t noise assumption (right). The number of outliers is varied.

If only 6 outliers (17%) are contained in the training data set, as in the first row, then the model with the Student's-t noise assumption is able to reject all outliers, and therefore it is able to predict the function (4.23). If the number of outliers is increased, then the robust GP model has increasing difficulties in rejecting them. If the number of outliers is increased to eleven, as in the fourth row, the model with the Student's-t noise assumption cannot distinguish clearly between outliers and correct measurements anymore. In the extreme case, if there are more outliers than correct data points, as in the last row, the model will fail on predicting the function (4.23), since it is obviously not possible to distinguish clearly between outliers and correct measurements for any type of modeling without additional information. Here, in the case of many outliers, the Student's-t distribution converges to a Gaussian distribution. Further, it can be seen that the GP model with the normal noise assumption is distorted by any outlier in all plots in figure 4.5.

We can see from the example in figure 4.5, that the new modeling is robust to outliers until a critical maximum quantity is reached. Now we want to examine this maximum quantity of outliers in a practical data set from engine calibration. This examination will illustrate that one can trust the robust modeling over a wide range of number of outliers.

In the figures 4.6 and 4.7, the 780 consumption measurements of a diesel engine from section 3.7.1 are considered, which have been transformed by (4.4). A detailed description of these measurements was given in section 3.7. From these 780 measurements, 273 measurements (35%) have been randomly selected and used for training, and the remaining 507 measurements (65%) have been used for model validation. With the training data, Gaussian process models with normal noise assumptions (left columns of figures 4.6 and 4.7) and Student's-t noise assumptions (right columns of figures 4.6 and 4.7) have been calculated.
In the top row of figure 4.6, the measured-predicted plot of the modeled consumption measurements is shown. As in section 3.7.1, it can be seen that the performance of both types of modeling is very good on this engine quantity, since consumption can easily be measured, and hence there is only little noise on the measurements. Therefore, especially the performance on the huge validation data set is good.
Now, we want to examine the performance of the new robust modeling technique when it comes to outliers. Therefore, a subset of the training data points is randomly selected and shifted by a big random noise. These points are the outliers. In the different rows of the figures 4.6 and 4.7 the number of outliers is varied. Table 4.2 gives an overview of how many correct data points are used for training and how many outliers are contained in the training set, in each row.
For clarity, the bottom row of figure 4.6 is considered as an example. As in every row, 273 measurements have been used for training and 507 measurements have been used for validation. From the 273 training data points, 78 had been randomly selected and distorted by a big random noise. The other 195 data points in the training set had not been changed.

From the measured-predicted plots in the figures 4.6 and 4.7, it can be seen that the GP model with the normal noise assumption has a bad performance in this experiment. This was expected, since every outlier will distort a model with a Gaussian noise assumption, as in the

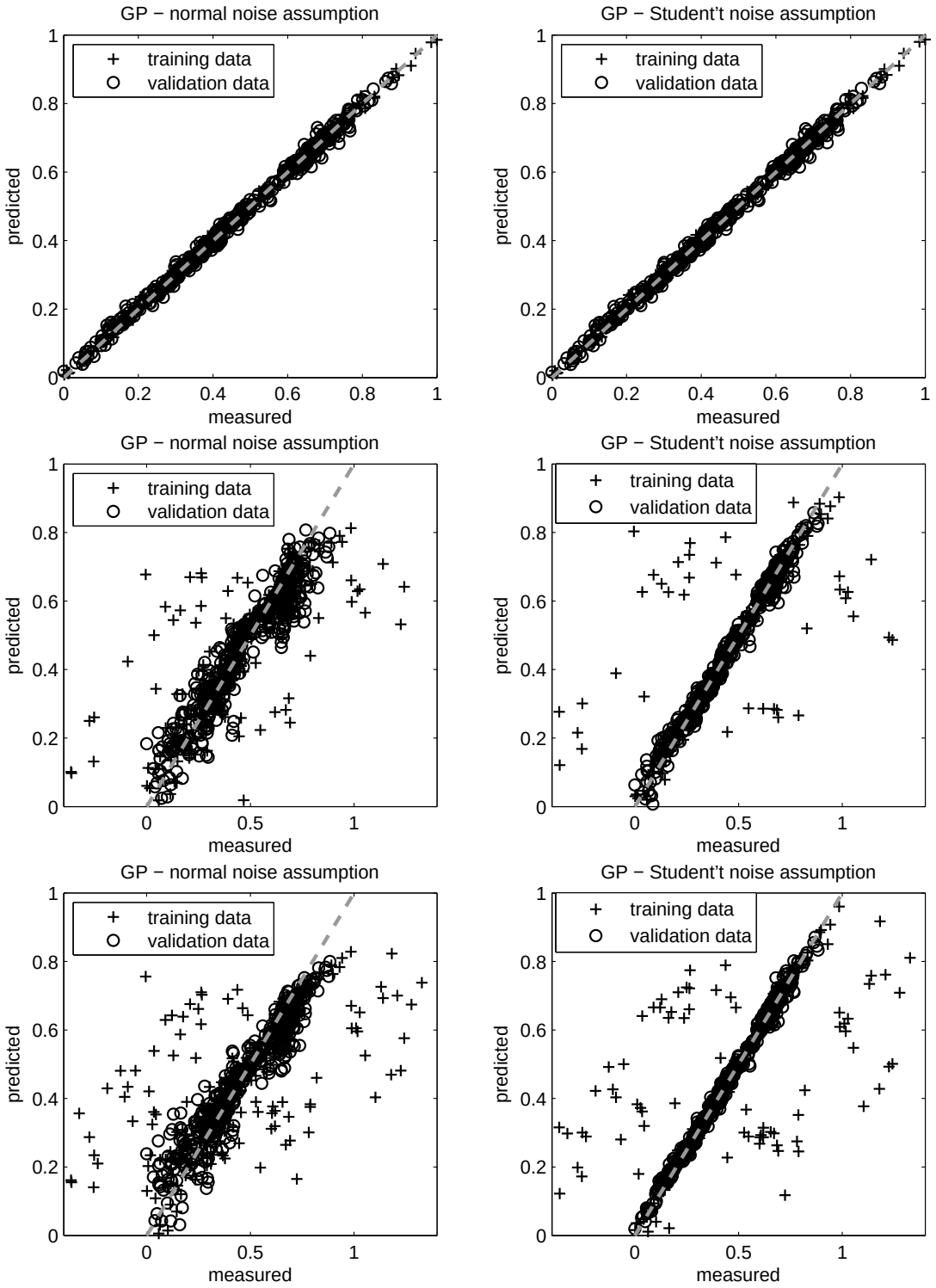


Figure 4.6: Measured-predicted plots for training and validation data of consumption measurements. The modeling was performed with a normal noise assumption (left) and a Student's-t noise assumption (right). In the different rows the number of outliers is varied.

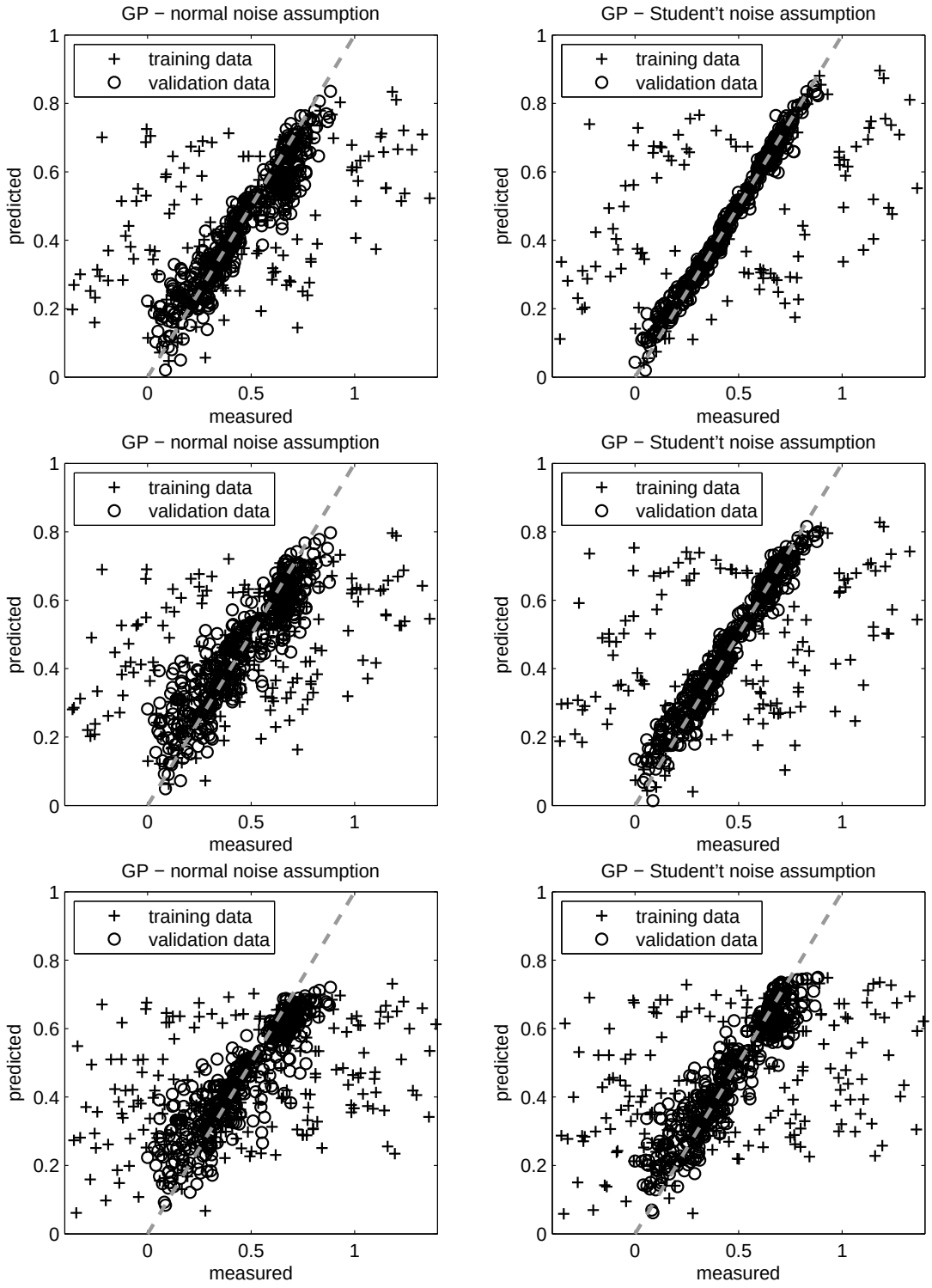


Figure 4.7: Measured-predicted plots for training and validation data of consumption measurements. The modeling was performed with a normal noise assumption (left) and a Student’s-t noise assumption (right). In the different rows the number of outliers is varied.

	number of training data points		number of validation data points
	correct data	outliers (distorted data)	
top row figure 4.6	273 (100%)	0 (0%)	507
middle row figure 4.6	234 (86%)	39 (14%)	507
bottom row figure 4.6	195 (71%)	78 (29%)	507
top row figure 4.7	156 (57%)	117 (43%)	507
middle row figure 4.7	117 (43%)	156 (57%)	507
bottom row figure 4.7	78 (29%)	195 (71%)	507

Table 4.2: Number of correct training data points, outliers and validation data points of figures 4.6 and 4.7.

theoretical experiments above.
In contrast, it can be seen that the robust GP model has a very good performance, even if the number of outliers gets large. Even if there are 30% outliers in the data, the new robust model can reject all of them and it is able to perform very well on the validation data set. This should be sufficient for engine calibration tasks, since the number of outliers will rarely be higher than 30% of the measurements. Hence, it is illustrated that one can trust this new technique over a wide range of number of outliers.
Only when the number of outliers was very high, the performance of the robust modeling got worse, as in the lower two rows in figure 4.7.

From the theory, the theoretical maximum number of outliers can be estimated, for which our robust modeling will give a good performance. It can be shown that a modeling with a Student's-t noise assumption (4.9) can (theoretically) reject up to m outliers, if the data set contains at least $2m$ measurements [90]. Obviously, the critical maximum number of outliers may vary in practical applications and it may depend on the quality of the rest of the measurements.
But clearly, the examinations in this section emphasize the strengths of the robust GP regression, and they demonstrate that this new approach can provide a substantial advantage for engine calibration tasks.

4.7 Conclusion and Discussion

In this section a new modeling framework for engine calibration was presented, which could be developed by modifying state of the art approaches from other fields of research and by introducing new techniques.
This new framework meets two important requirements from engine calibration. Due to the use of a transformation, it is robust to differences between the assumed and the real distributions of the data. Due to the use of a Student's-t likelihood, it is robust to outliers.

The main drawback of this new approach is an increasing computational cost. The new modeling framework requires approximately twice as much computing time as the conventional GP regression, which is already in itself a computationally expensive method.

This is a severe limitation of this new approach, which restricts the use to particular applications, where the number of measurements is not too large. Hence, as conventional Gaussian processes, this new method is not suitable for dynamic engine calibration, but it is appropriate for stationary base engine calibration.
Therefore, a possible future work would be to integrate the new robust techniques into sparse kernel machines, in order to reduce the computational effort for dynamic engine calibration tasks. However, this was not a focus in this work, since the quality of the prediction of sparse kernel machines is not as high as with a full GP model, and because the computational cost of the presented approach is acceptable for stationary base calibration.

The main properties and advantages of this new modeling framework can be summarized as follows:

- *Dependable performance.* In many practical applications in engine calibration, outliers occur in the data set and the distributions of the measurements are not Gaussian shaped. Compared to state of the art algorithms for base calibration, in this chapter a new framework was presented, which achieved a dependable performance under these complex conditions. Even if there are many outliers in the data set, the prediction of the model can be trusted.
- *No manual interaction is required.* With state of the art types of modeling, outliers have to be removed before the model training is performed, in order to achieve an accurate prediction. Since an automatic detection of outliers is not very robust or computationally very expensive with these state of the art techniques, the outliers usually have to be removed manually. In comparison, with this framework the outliers do not have to be removed before the training is performed, and due to the use of a numerical optimization of the marginal likelihood (4.18), the robust GP results in a fully automatic approach. Therefore, this new technique enables an increased automation of the modeling process, and obviously, this saves time and resources for engine calibration tasks.
- *Online optimization for complex quantities is possible.* If the model is distorted by outliers, then in an automated online optimization bad models will lead to wrong predictions and useless measurements will be taken at undesired regions. Hence, a large part of measurements would be meaningless and the optimization would cause high costs. Thus, state of the art online optimization is only performed for quantities of an engine which are relatively easy to measure, as consumption, and not for quantities where the risk of outliers is much higher, like soot. Compared to these state of the art techniques, with this new modeling framework it is possible to perform an online optimization even for complex quantities of the engine.
- *Increased user acceptance.* In comparison to state of the art approaches, which require a manual interaction in order to identify the outliers, with this new framework the users of calibration tools can rely on a fully automatic and dependable modeling. Hence, this approach assists the calibration engineers. Instead of searching for outliers and evaluating the model quality, the users of calibration tools can concentrate on their main tasks with this new robust modeling. This clearly increases the user acceptance for model-based calibration techniques.

Chapter 5

BASICS OF OPTIMIZATION FOR ENGINE CALIBRATION

Mathematical optimization algorithms are required for various different tasks in engine calibration. E.g., when a GP modeling is performed, the likelihood function (4.18) or (2.49), respectively, is optimized w.r.t. the hyperparameters. Further, with these models the calibration engineer can optimize the consumption and the emissions of the engines. In addition, the process of performing measurements can be improved through using a model-based online optimization, as we will see soon.
In this chapter the state of the art optimization techniques for engine calibration will be discussed in an abbreviated version.

In this thesis only continuous optimization and no combinatorial optimization is considered. Let $\mathcal{X}$ be a set and $f : \mathcal{X} \to \mathbb{R}$ be a function. The aim of an optimization algorithm is to find a $\mathbf{x}^* \in \mathcal{X}$, so that $f(\mathbf{x}^*) \leq f(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{X}$. Then, $\mathbf{x}^*$ is called a global minimum, f is called cost function or objective function and we define $\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) := f(\mathbf{x}^*)$. It is no restriction to consider only minimization problems, since a maximization of f can be reached through a minimization of $-f$. Often, it is not possible to find the global optimum. Therefore, many algorithms seek for a $\mathbf{x}^* \in \mathcal{X}$, so that $f(\mathbf{x}^*) \leq f(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{U}(\mathbf{x}^*)$, where $\mathcal{U}(\mathbf{x}^*) \subset \mathcal{X}$ is a neighborhood of $\mathbf{x}^*$. Then, $\mathbf{x}^*$ is called a local minimum. If the objective function has more than a single local minimum, then the function is called multi-modal, otherwise unimodal.
In typical optimizations exist several constraints on the set $\mathcal{X}$, which can be expressed by the $N_{c,eq}$ equality constraints $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ and by the $N_{c,ieq}$ inequality constraints $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$, with the functions $\mathbf{g} : \mathcal{X} \to \mathbb{R}^{N_{c,eq}}$ and $\mathbf{h} : \mathcal{X} \to \mathbb{R}^{N_{c,ieq}}$. If these constraints are taken into account during the optimization, then a constraint optimization is performed. Further, the feasible set $\mathcal{X}_F$ is defined as the set of points $\mathbf{x} \in \mathcal{X}$, which satisfies the constraints, so that $\mathcal{X}_F := \{\mathbf{x} \in \mathcal{X} | \mathbf{g}(\mathbf{x}) = \mathbf{0}, \mathbf{h}(\mathbf{x}) \leq \mathbf{0}\}$.
For a more detailed discussion it is referred to [88].

Similar to the 'no free lunch' theorems for supervised learning, there exist the 'no free lunch' theorems for optimization [134]. These theorems show that all optimization algorithms have an equivalent average performance over all possible problems. Hence, there exists no optimizer which is superior for all possible tasks. Therefore, in order to get a good performance on a specific application, one has to choose an optimization algorithm which is appropriate for the specific problem. This property will be crucial in the further sections, where we will

choose an optimizer for each task, after considering the specific requirements of these applications.

In the next sections the state of the art optimization techniques, which are used in stationary base calibration, are discussed. In the sections 5.1 and 5.2 two different types of optimization for a single objective function f are examined, and in section 5.3 optimization techniques for more than one objective function are discussed. The calculation of optimal test plans with design of experiments is discussed in section 5.4, and in section 5.5 the state of the art approaches for online optimization for engine calibration are examined.

5.1 Classical Nonlinear Optimization

In this thesis, algorithms which use the gradient information for optimization are referred to as classical nonlinear optimization algorithms. Obviously, these methods require that the objective function is continuous differentiable (or at least that the objective function can be approximated by a piecewise continuous differentiable function) and that the gradient can be calculated (approximated) efficiently.
If these requirements are met, then a method of the broad class of numerical nonlinear optimization algorithms can be used. Since a nonlinear function cannot be minimized in a single step, the basic idea of these algorithms is, starting from a point $\mathbf{x}_0$, to generate a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ...$, which converge to a local optimum of the objective function. In this process, the next point of the iteration $\mathbf{x}_{n+1}$ is determined through the value, the gradient and the Hessian of the objective function at the point $\mathbf{x}_n$.

Generally, it can be distinguished between first order approaches and second order methods. First order methods use only the information of the first order derivatives, expressed by the gradient. The simplest technique is the method of the steepest descent, in which the iteration of points is given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \hat{\alpha} \frac{\nabla f(\mathbf{x}_n)}{\|\nabla f(\mathbf{x}_n)\|}, \tag{5.1}$$

where $\hat{\alpha} > 0$ is the step size. For a small enough $\hat{\alpha}$ value, the objective function can always be reduced. However, these first order approaches have only a low convergence rate, which means that they need many iterations in order to proceed to the optimum.

Second order approaches make also use of the second order derivatives, expressed by the Hessian of the objective function. These algorithms usually have a higher convergence rate, and therefore the objective function has to be evaluated fewer times, in order to achieve the same accuracy of the optimum. This is especially important for the optimization of problems, where a single evaluation of the objective function is computationally expensive, like the evaluation of the log likelihood functions (4.18) and (2.49). Hence, for the optimization of the hyperparameters of a Gaussian process, a second order approach is used.
These approaches consider a second order approximation (Taylor expansion) of the objective

function around a point $\mathbf{x}_n$, which is given by

$$f(\mathbf{x}) \approx f(\mathbf{x}_n) + (\mathbf{x} - \mathbf{x}_n)^T \nabla f(\mathbf{x}_n) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_n)^T \mathbf{H}(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n). \tag{5.2}$$

If this approximation is exact and if $\mathbf{H}(\mathbf{x}_n)$ is positive definite, then by setting $\nabla f(x) = 0$,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}(\mathbf{x}_n)^{-1} \nabla f(\mathbf{x}_n) \tag{5.3}$$

would be the optimum of the objective function. However, the approximation (5.2) will often only be valid in a limited region around $\mathbf{x}_n$ and $\mathbf{H}(\mathbf{x}_n)$ is often modified, in order to obtain a positive definite matrix. Hence, often the iteration

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \hat{\alpha}\, \hat{\mathbf{H}}_n^{-1} \nabla f(\mathbf{x}_n) \tag{5.4}$$

is performed, where $\hat{\mathbf{H}}_n$ is the modified Hessian and $\hat{\alpha}$ is the step size, which is usually obtained by a line search or a trust region method [88].

However, the exact evaluation of the Hessian of the log likelihood functions (4.18) and (2.49) is computationally expensive. Hence, a method was chosen where the Hessian is approximated during the iterations of the optimization, which are called quasi Newton methods. A common update formula, which showed a good performance in this application, is the BFGS procedure, which is given by

$$\hat{\mathbf{H}}_{n+1} = \hat{\mathbf{H}}_n - \frac{\hat{\mathbf{H}}_n \hat{\mathbf{s}}_n \hat{\mathbf{s}}_n^T \hat{\mathbf{H}}_n}{\hat{\mathbf{s}}_n^T \hat{\mathbf{H}}_n \hat{\mathbf{s}}_n} + \frac{\hat{\mathbf{v}}_n \hat{\mathbf{v}}_n^T}{\hat{\mathbf{v}}_n^T \hat{\mathbf{s}}_n} \tag{5.5}$$

where $\hat{\mathbf{s}}_n := \mathbf{x}_{n+1} - \mathbf{x}_n$ and $\hat{\mathbf{v}}_n := \nabla f(\mathbf{x}_{n+1}) - \nabla f(\mathbf{x}_n)$ [88]. The procedure is initialized with the identity matrix, which corresponds to the steepest descent method in the first iteration. Further, this approach assures that the matrix $\hat{\mathbf{H}}$ is positive definite, so that $-\hat{\mathbf{H}}_n^{-1} \nabla f(\mathbf{x}_n)$ is guaranteed to be a descent direction. Constraints on the objective function can be handled with the BFGS-B optimization, which is a bound-constrained algorithm.

With these classical methods, the optimal value $\mathbf{x}^*$ of a local optimum can be determined in a low amount of computing time and precisely, except for errors which occur due to the finite machine accuracy. If the objective function is multi-modal, then a multistart strategy can be useful for finding the global optimum. This approach performs the optimization several times from different starting points, and in the end the minimal local optimum will be chosen. With this technique the ambition is that at least a single optimization run will converge to the global optimum, if the optimization is performed many times.
As said above, the BFGS procedure, combined with a multistart strategy, was suitable for optimizing the likelihood function of the GP model w.r.t. the hyperparameters, since the likelihood function usually has only a few local minimums, see section 6.2.
But clearly, these classical methods have serious drawbacks, if the objective function has numerous different local minimums, since the optimization has to be started many times.

5.2 Evolution Strategies

An alternative optimization approach for continuous functions is the evolution strategy. Evolution strategies (ES) belong to the class of evolutionary algorithms, which are optimization routines, which imitate the principles and ideas of evolutionary processes of nature. The general functionality of most of the evolutionary algorithms is given in algorithm 1.

Algorithm 1 Evolutionary Algorithm (general)

input: objective function f, search space $\mathcal{X}$ and further parameters
output: last population $P_{i_{end}} \subset \mathcal{X}$ and optimum $\mathbf{x}^*$

1: initialize the starting population P_0, set $i := 0$
2: **repeat**
3: generate P_t' out of P_i through *recombination*
4: generate P_i'' out of P_i' through *mutation*
5: generate P_{i+1} out of P_i'' through *selection*
6: set $i := i + 1$
7: **until** abort criterion = true

At the beginning of the evolutionary algorithm the initial set $P_0 \subset \mathcal{X}$ contains the initial points $P_0 := \{\mathbf{x}_{0,1}, \mathbf{x}_{0,2}, ...\}$. According to the evolutionary processes in nature, the points $\{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ...\}$ are called individuals and the set P_i is called the i-th population. During the optimization process the next population P_{i+1} is generated out of P_i by recombination (sometimes called crossover), mutation and selection.
It should be noted that the notation for evolutionary algorithms is not consistent in the literature and depends on the specific class of algorithms, such as genetic algorithms, evolution strategies, genetic programming, etc., see [5, 87]. As said above, in this thesis we focus on evolution strategies, which are discussed extensively in [99, 110, 111].
Due to the fact that the recombination is of lower importance for ES [92, 99], this operation was neglected in this thesis, and an ES with mutational self-adaptation of the step size was implemented, since this algorithm showed a good performance on practical problems. In this algorithm the mutation is carried out by adding a normally distributed random variable to the individuals

$$\mathbf{x}_{i,j}' = \mathbf{x}_{i,j} + \mathbf{z} \quad \text{with} \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\xi}_{i,j}),\ \mathbf{x}_{i,j} \in P_i' \text{ and } \mathbf{x}_{i,j}' \in P_i'' \tag{5.6}$$

where $\boldsymbol{\xi}_{i,j}$ is called step size, which is passed on from population to population.
The selection of the ES is carried out by evaluating the objective function and by selecting the best individuals for the next population.

Instead of going deeper into detail of this state of the art approach for single-objective optimization, it should be referred to the literature given above. Rather, the main properties of this optimization approach should be discussed in comparison to the classical nonlinear optimization methods, which were examined above.
Evolution strategies usually converge much slower to a local optimum than classical nonlinear approaches, since they often can only obtain a linear convergence rate. Hence, as mentioned

above, the classical approaches are useful for optimizations where the optimum has to be determined precisely and where the objective function contains only a few local minimums. Therefore, the BFGS method is used for optimizing the log likelihood functions w.r.t. the hyperparameters, since a small change in the hyperparameters can cause a very different model behavior and because the log likelihood usually has only a few local minimums.
However, for various other tasks, e.g. for optimization of the models and for single-objective online optimization, the precise values of the optimums are not of major interest, but rather one is interested in the first decimal places of the optimum, since it is usually not possible to set an adjustment parameter on the engine to an exact value. In addition, in these tasks often numerous different local optimums occur. E.g. the objective function of a complex single-objective online optimization can easily have dozens or even hundreds of local optimums. Hence, a classical nonlinear optimization with a multistart procedure has to be performed numerous times, in order to assure that an appropriate value of the objective function can be found with an adequate probability. This is computationally infeasible for many practical problems in engine calibration.
In comparison to that, through working with an evolution strategy it was always possible in practical applications to find an adequate candidate for the optimum in a reasonable amount of computing time. Obviously, a convergence to the global optimum of such a highly multi-modal problem cannot be guaranteed, but one can easily show that the probability for finding a suitable candidate of the objective function increases, if the number of individuals increases, and since the computational efficiency of the ES can be enhanced by parallelization of the evaluation of the objective function, an ES with a high number of individuals (typically a few thousands) can be used in practical applications.
Therefore, in this thesis an evolutionary strategy with a high number of individuals was used for single-objective optimizations, where the value of the optimum does not have to be determined to high accuracy, but rather where a robustness to highly multi-modal behavior is important.

5.3 Multi-Objective Optimization

In the last two sections optimization algorithms for a single objective function f have been discussed. In engine calibration tasks, however, often more different objectives have to be considered, as discussed in chapter 1. Typical examples are the optimization of a diesel engine, where mainly consumption, NOx and soot emissions have to be minimized, and the optimization of a direct injection gasoline engine, where mainly consumption and soot emissions have to be minimized.
In these optimization tasks often a trade-off occurs. A common example is the NOx-soot trade-off. Typically, it is not possible to minimize NOx and soot at the same time, but rather one has to decide, e.g., if a higher NOx rate has to be tolerated, in order to decrease the soot emissions.

Hence, for the multi-objective problem we seek for the solution of

$$\min_{\mathbf{x}\in\mathcal{X}} \; [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{d_{Obj}}(\mathbf{x})] \quad \text{s.t.} \;\; \mathbf{g}(\mathbf{x}) = \mathbf{0},\; \mathbf{h}(\mathbf{x}) \leq \mathbf{0} \tag{5.7}$$

where f_i are the different objective functions and $\mathbf{g}$ and $\mathbf{h}$ are the constraints, which were defined above. As for the single-objective optimization, here it is also sufficient to consider only minimization problems. Further, the feasible objective space $\mathcal{Y}_F$ is defined as the set of points which can be reached through the objective functions f_i and the feasible set $\mathcal{X}_F$, so that $\mathcal{Y}_F := \bigcup_{\mathbf{x}\in\mathcal{X}_F} \{(f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{d_{Obj}}(\mathbf{x}))\}$.

This multi-objective problem is illustrated in figure 5.1 (a). In this example two objectives f_1 and f_2 are considered, and the feasible objective space $\mathcal{Y}_F$ is a two dimensional area.
The ambition is to minimize both objectives f_1 and f_2. However, as one can see from the plot, there is no unique solution to this problem. In order to explain the basic ideas of the multi-objective problem (5.7), the three points A, B and C in figure 5.1 (a) are considered. The point A is no solution of the multi-objective problem (5.7), since there exist points which have a lower value of f_1 and at the same time a lower value of f_2, as the points B and C. Both points B and C are solutions of the multi-objective problem (5.7), since there exist no points in the feasible objective space $\mathcal{Y}_F$, which have a lower value in f_1 at the same f_2 value, or which have a lower value in f_2 at the same f_1 value. The points B and C are called Pareto optimal points (sometimes also named Pareto efficient points). The set of all Pareto optimal points is called Pareto frontier and this set is indicated by the dashed-dotted line in figure 5.1 (a).
The Pareto frontier of d_{Obj} different objectives can consist of several connected sets, which are in turn submanifolds of $\mathbb{R}^{d_{Obj}}$ with the maximum dimension of $d_{Obj} - 1$. In this thesis we constrain the number of objectives to two or three, and therefore in practical problems the Pareto frontier consists of connected sets, which are either one-dimensional curves or two-dimensional surfaces.
Another important property of multi-objective problems, which should be mentioned here, is domination. One point $\mathbf{y}$ dominates a point $\mathbf{y}^*$, if each parameter of $\mathbf{y}$ is not greater than the

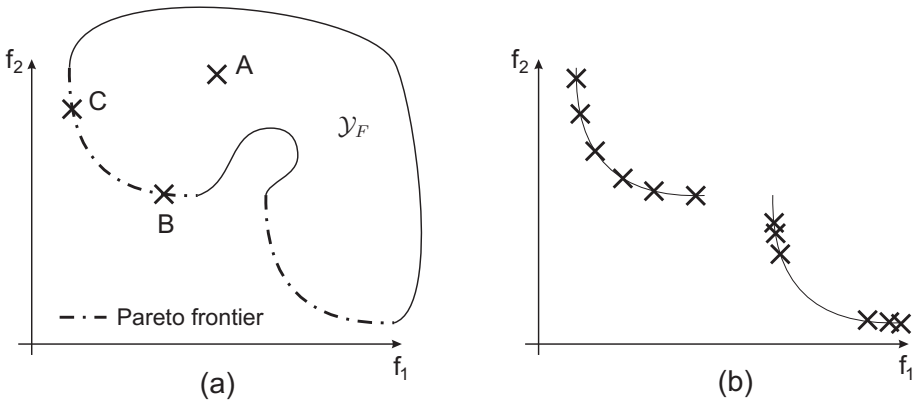


Figure 5.1: Basic principles of multi-objective optimization.

corresponding parameter of $\mathbf{y}^*$ and at least one parameter is strictly less: that is, $y_i \leq y_i^*$ for each i and $y_i < y_i^*$ for some i. This is written as $\mathbf{y} \prec \mathbf{y}^*$. Hence, from the example in figure 5.1 it follows that $B \prec A$ and $C \prec A$.

As in single-objective optimizations, the multi-objective problem (5.7) can contain several local minimums.
In addition, since it is only possible to observe the Pareto frontier at discrete points, as we will see soon, the diversity of these points is of interest, as illustrated in figure 5.1 (b).
In this plot the same Pareto frontier as in figure 5.1 (a) is indicated by the solid lines. However, practical implementations for multi-objective optimization can only provide a limited amount of points, which indicate an approximation of the Pareto frontier, as the points in figure 5.1 (b). By the comparison of both parts of the Pareto frontier, it can be seen that the points of the approximation of the upper left part are better distributed than the points of the approximation of the lower right part of the Pareto frontier. Therefore, one can estimate the upper left part of the Pareto frontier better than the lower right part, although the number of Pareto optimal points is the same in both parts. Hence, in order to obtain a useful approximation of the Pareto frontier, it is important that the Pareto optimal points are equally distributed in the objective space, which is called a good diversity.
See also [19] for a good introduction into multi-objective optimization.

Several different approaches for multi-objective optimization (MOO) exist in the literature. As in single-objective optimizations, it can generally be distinguished between classical (gradient based) nonlinear optimization algorithms and evolutionary algorithms, and the properties of these approaches for multi-objective optimization are similar to the properties of the classical and evolutionary approaches for single-objective optimization.
In this thesis the evolutionary algorithms are considered for MOO, since the precision of the values of the points, which indicate the Pareto frontier, does not need to be very high, but rather a rough global approximation of the Pareto frontier in a short amount of computing time and a robustness to highly multi-modal behavior is important.

During this work several different evolutionary techniques were examined, as, e.g., the NSGA-II [19, 20] and the SPEA2 [136, 137]. The NSGA-II approach showed the best performance on practical problems in engine calibration, and therefore this algorithm was used in this thesis. For a more detailed information on this state of the art technique it should be referred to the literature given above.

5.4 Design of Experiments for Model-Based Offline Optimization in Engine Calibration

Design of experiments (DoE) is a widely used term in engine calibration. Typically, in engine calibration the whole process, from planning the experiments, over modeling, up to optimization, is denoted by DoE. However, in statistics (and also in this thesis) DoE is only referred

to the first step of the whole process: the design of the test plan, before the measurement is performed on the test bench. This initial task is important in model-based offline optimization, where the whole design of the measurements is fixed, before any observations are made.
As discussed in section 1.2.2, model-based online optimization has various advantages compared to model-based offline optimization, and therefore this thesis focuses on online optimization. However, since offline optimization is very common in engine calibration and online optimizations also need a design of the initial (starting) measurements, DoE is discussed in this section in an abbreviated version.

In general, two main types of DoE can be distinguished in engine calibration: optimal design of experiments for linear models and space filling designs.
First, optimal design of experiments will be discussed.

If a linear modeling (2.17) is used (although this is not the best choice, as discussed in chapter 3), then from (2.23) it can be seen that the covariance of the estimated parameters is given by

$$\sigma^2(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}. \tag{5.8}$$

The parameters $\hat{\mathbf{\Theta}}$ can be determined very precisely, if the covariance is as small as possible. Hence, the precision of the model depends only on the noise of the measurements σ^2 and on the design matrix $\mathbf{\Phi}$. Therefore, if we use a linear model, then we are able to calculate the optimal design of the measurements, before any observations are made on the test bench. This means that, for linear models, the information of the measurements cannot improve the design of the experiments. It should be noted that this is not the case if we use a nonlinear modeling. For a nonlinear modeling, the information of the previous measurements is definitely useful for planning the next measurements, and therefore online optimizations are suitable for nonlinear models, as it will be discussed in the next section.
In order to optimize the precision of the linear model, we have to minimize the covariance matrix (5.8). Since the minimum of a matrix is not uniquely defined, different criteria have been developed. Without the claim of completeness, some of them are [6]:

$$\begin{aligned}
&\text{A-optimal design:} && \min \operatorname{trace}(\mathbf{\Phi}^T\mathbf{\Phi})^{-1} \\
&\text{D-optimal design:} && \min \det(\mathbf{\Phi}^T\mathbf{\Phi})^{-1} \\
&\text{E-optimal design:} && \min \lambda_{\max}(\mathbf{\Phi}^T\mathbf{\Phi})^{-1},
\end{aligned}$$

where the D-optimal design is most commonly used in engine calibration.
Algorithms, which are able to compute such a design, are, e.g., the Fedorov algorithm [27], the modified Fedorov algorithm [16], the DETMAX algorithm [74, 75] and the k-exchange algorithm [49]. Since the k-exchange algorithm has a fast convergence for a large number of experiments [122], in this thesis this approach, combined with a D-optimal design, was implemented. However, this technique was only implemented in order to increase the customer acceptance, since the users of calibration tools are very used to this technique.

Nevertheless, this approach has some serious shortcomings, and therefore it is not recommended in this thesis.

The basic problem of this technique is that the design is only optimal for the specific linear model, which has to be defined before observations are made on the test bench. In some applications this may not be problematic, since a lot of prior knowledge may be available in these tasks. However, in many applications in engine calibration, the behavior of the engine is not known in such detail that the structure of a linear model can be fixed a priori, before any measurements are made on the test bench (compare also the discussion in section 3.2).
This has severe consequences. Typically, a change of the structure of the linear model, e.g., a change of the degree of the polynomial model, is critical, and, in the worst case, not possible with such a design. Hence, if the a priori assumptions differ from the real engine behavior, then the areas, in which the measurements are placed, are not optimal, and, in the worst case, new measurements have to be made, which is clearly time- and cost-intensive.

In addition, as discussed in chapter 3, a nonlinear modeling (e.g. a Gaussian process model) is more suitable in engine calibration, since it can adapt the degree of nonlinearity itself, and because it does not make as strict assumptions as a linear model with a fixed structure. Hence, it is meaningful to use a measurement design which does not make strict assumptions on the engine behavior, too [51].
Such experimental designs are space filling designs, such as Latin hypercubes [72].
Further, low-discrepancy sequences [60], such as the van der Corput sequence [125], the Halton sequence [39], and the Sobol sequence [115], guarantee also a good distribution of the measurements in the input space. In addition, these sequences assure that a combination of different designs does not lead to the loss of an evenly distribution.
In practical applications in engine calibration these sequences showed a very good performance, and therefore these approaches are used for the initial measurements of a model-based online optimization.

5.5 State of the Art Model-Based Online Optimization in Engine Calibration

As discussed in the last section, for a nonlinear modeling the information of the previous measurements is helpful to improve the decision where the next measurement should be placed. In order to achieve this improvement, the modeling and optimization algorithms are in a permanent interaction with the test bench, which allows the models to give a feedback of their quality, and which enables to make optimal decisions based on the previous observations. This has various advantages, as discussed in section 1.2.2.2, and therefore time and costs on the test bed can be considerably reduced by the usage of model-based online optimization [53].

However, nearly all commercially available calibration tools are purely designed for model-based offline optimization and do not provide online optimization features.

The Model-Based Calibration Toolbox [104, 105, 118] from MathWorks is uniquely designed for office use and does not have a connection to the test bench.

Also, the Easy-DoE Toolsuite [44] from IAV GmbH is designed for office use. The connection to the test bench is performed with ORION from A&D Company and IAV GmbH, which is not able to perform an online optimization [1].
The AVL CAMEO Tool [33] consists of a test bed and an office version [4]. For the test bed version, a toolbox named iPROCEDURE ADAPTIVE DOE exists, which is able to adjust the initial test plan by automatically adding further points. However, the performance of this approach is limited through the use of very simple regression models [53]. In addition, this approach does not perform an optimization during the measurement, in order to identify the most suitable places of the next points, and therefore this technique is not regarded as an online optimization in this thesis.

In comparison to these commercial products, BMW developed an own solution called mbminimize [55, 93, 116], which is able to perform online optimizations.
As already discussed in section 3.5, this approach uses a committee of MLP networks (see section 2.4.1) and LLR models (see section 2.2.3). The expectation and the variance of this committee of N_{COM} models can be calculated by [92]

$$\mathbb{E}_{\text{COM}}(\mathbf{x}) = \frac{1}{N_{\text{COM}}} \sum_{i=1}^{N_{\text{COM}}} y_i(\mathbf{x}) \tag{5.9}$$

$$\mathbb{V}_{\text{COM}}(\mathbf{x}) = \frac{1}{N_{\text{COM}} - 1} \sum_{i=1}^{N_{\text{COM}}} \left(y_i(\mathbf{x}) - \mathbb{E}_{\text{COM}}(\mathbf{x})\right)^2, \tag{5.10}$$

where y_i, $i \in \{1, ..., N_{\text{COM}}\}$ are the different model outputs of the committee. With these values, which represent the expectation and the uncertainty of the real engine behavior, an online optimization is performed. This online optimization is divided into distinct stages (phases). In the first stage measurements are placed in areas, where the uncertainty of the true engine behavior is maximal (maximum value $\mathbb{V}_{\text{COM}}(\mathbf{x})$). These measurements in the first stage improve the global model quality.
In the further stages not only the maximum uncertainty is considered, but also the areas of the input space where the behavior of the engine is optimal (e.g. minimal consumption). Therefore, the variance $\mathbb{V}_{\text{COM}}(\mathbf{x})$ and the expectation $\mathbb{E}_{\text{COM}}(\mathbf{x})$ of the committee are combined to a single value, and the importance of the uncertainty and the optimality are weighted by an additional parameter, which is varied in each stage. With this procedure a smooth progress is achieved, where at the beginning mainly the uncertain areas are measured (which is called exploration), and at the end mostly the optimal areas are examined (which is called exploitation), see [53, 92]. Hence, in the later stages of the online optimization the local model quality around optimal areas is improved.

Therefore, with this approach the quality of the models could be increased, while at the same time the required numbers of measurements could be reduced, which leads to a more efficient use of the test bench time. Thus, at BMW time and costs could be remarkably reduced with online optimization [53, 103].

Nevertheless, the mbminimize approach has severe shortcomings. Since it was designed for an

online optimization for consumption of a gasoline engine, it can only handle a single objective function. However, as discussed in section 5.3, in modern engines often several different objectives have to be considered, and therefore the mbminimize concept cannot be applied to these challenging problems. In addition, due to the choice of the modeling, this approach cannot make use of the potentials of a fully probabilistic model, like Gaussian processes, as we will see soon.
In order to overcome these drawbacks, a new and improved model-based online optimization concept will be presented in the next chapter.

5.6 Conclusion and Discussion

In this chapter a summarization of the most important optimization concepts for engine calibration was given in an abbreviated version.
At the beginning, a selection of state of the art algorithms for single- and multi-objective optimization was presented, the properties of the different approaches were discussed, and a suitable technique was chosen for each optimization problem in this thesis.
After that, different methods for design of experiments for offline optimization were discussed, and a recommendation for an appropriate approach was given.
At the end, the state of the art for model-based online optimization in engine calibration was examined, and the advantages of such an online optimization were discussed. However, as already mentioned above, these state of the art online approaches suffer from various disadvantages. These drawbacks, and how they can be overcome, are discussed in the next chapter, where a new approach for model-based online optimization is presented.

Chapter 6

IMPROVED MODEL-BASED ONLINE OPTIMIZATION FOR ENGINE CALIBRATION

In the previous chapter the state of the art in model-based online optimization has been discussed, and it was outlined that time and costs can be remarkably reduced with this approach. Hence, also in this work these techniques for model-based online optimization (MBOO) were examined. However, it was found out that the state of the art approaches have several severe shortcomings:

(MBOO1) As already mentioned in the last chapter, the state of the art online optimizations were developed for the optimization of consumption of a gasoline engine, and hence *they can only handle a single objective function*. However, as discussed in section 5.3, due to the increasing complexity in modern engine calibration tasks, often several different objectives have to be regarded, and therefore also the online optimization should be able to deal with more objective functions.
Thus, in this work a new multi-objective online optimization for engine calibration was developed. This new approach is able to place measurements in Pareto optimal areas, in order to improve the quality of different objectives in the most important domains.

(MBOO2) State of the art techniques for online optimization *do not use a fully probabilistic approach* for the modeling part. Therefore, *they are not able to predict the uncertainty of the model accurately*. However, as we will see soon, this is a crucial property for online optimization, since we want to place measurements in areas where the estimated model error (and therefore the predicted uncertainty) is high, in order to reduce this error. In addition, due to the lack of probabilistic features, state of the art techniques *are not able to estimate the quality of the model* and the reliability of the prediction with a low amount of data. Nevertheless, as we will see soon, this is an important property, since it enables the online optimization to evaluate if already enough measurements have been taken at the test bench and the optimization can be stopped. Hence, in state of the art online optimizations the number of measurements has to be fixed manually, in advance, and therefore it is possible that too few or too many measurements are made.
In contrast, in this thesis the fully probabilistic Gaussian process regression is used for the model-based online optimization. This approach allows to perform a fully

automatic optimization with an increased performance, because of two reasons. First, the reliability of the prediction can be evaluated by the marginal likelihood probability distribution of the model parameters (see section 6.2.2.1). Second, the prediction of the uncertainty can be estimated by the variance of the model, from which we will make extensive use in the following, when we are, e.g., calculating lower confidence bounds, searching for highest variances or performing cross-validation with confidence errors, as it will be examined below.

While there exists no GP assisted online optimization in engine calibration, in other fields of research (e.g. in the domain of global optimization) the idea of optimization by using Gaussian stochastic processes is far from new. However, as we will see soon, these state of the art approaches from other fields of research can be further enhanced, in order that the performance for engine calibration tasks is improved.

The goal of this chapter is the development of a model-based online optimization for engine calibration, which has an increased performance compared to traditional state of the art approaches for model-based optimization. This is achieved by the design of new techniques under consideration of the requirements of the calibration process and of the needs of the calibration engineers.

Although these approaches are developed for an online optimization on the test bench, it should be noted that these techniques can also be used for a model assisted optimization of an expensive-to-evaluate computer simulation, which is a further field of application in engine calibration [53]. In order to apply these approaches to a computer simulation, the only required modification is that the measurement noise σ has to be set to zero.

In the next section the basic problems and challenges of such an online optimization are discussed, and a two-stage approach is presented in order to meet these demands. In section 6.2 the first stage, called online modeling, and in section 6.3 the second stage, named online optimization, is presented.

6.1 A Two-Stage Approach

The ambition of an online optimization is to identify the optimal areas of the objective functions with as few measurements as possible. This is achieved by placing the measurements in areas, where the estimated gain of information for the online optimization is expected to be high. Since the objective functions in engine calibration are typically multi-modal, there exists a trade-off for the online optimization.
First, starting from no information of the calibration problem, the optimization has to gather information about the (global) behavior of the engine quantities, which is called exploration. After that, when the basic behavior of the system is known, the optimization can exploit this information and search for the optimal areas of the objectives, which is called exploitation. Since the goal is to perform as few measurements as possible, the stage of the exploration

should be as short as possible. However, if the exploration is aborted too early, a wrong estimation of the true system behavior can lead the exploitation stage to ignore areas, which can contain optimal points. Hence, if the exploitation is started too soon, it is possible that only local optimums are found and the global optimums are missed. This conflict is known as the trade-off between exploration and exploitation, and it is well known in the field of global optimization.

In order to deal with this trade-off, in this thesis a two-stage approach was developed.
In the first stage exploration is performed. This task is referred to as online modeling in this thesis, and it is aborted, when the predictions of the model about the global engine behavior can be trusted. In the next section it will be discussed how this abort criterion can be determined.
After that, in the second stage exploitation is performed and this task is referred to as online optimization in this thesis.

This two-stage approach is different to the mbminimize concept of BMW. At mbminimize the whole optimization is divided into more than two stages, in order to achieve a smooth progress from exploration to exploitation. On the one hand, these additional stages were integrated into mbminimize due to the special choice of the modeling via a committee of MLP networks. On the other hand, this smooth progress can be useful in order to reduce the total number of measurements, since also the measurements of the optimization stage can further improve the prediction of the global engine behavior.
However, in order to avoid that the optimization is searching too early for a local minimum and missing the global one, a lot of a priori knowledge is required for the planning of the different stages, and therefore in the mbminimize concept the scheduling of the stages is performed manually [53]. In contrast, in this thesis the ambition is to develop a model-based online optimization, which can be fully automated.

6.2 Online Modeling

In the first stage, the model is improved stepwise in several iterations with update measurements from the test bench, until the quality of the model is appropriate. Therefore, this stage is named online modeling. According to figure 1.7, the basic functionality of the online modeling is given in algorithm 2.
At the beginning, the initial measurement, which was discussed in section 5.4, is performed on the test bench. With these measurements the initial model is calculated. After that, further update points are determined, and if the abort criterion is not fulfilled, then the procedure is repeated again.

As mentioned above, the ambition of online modeling is to perform exploration. Hence, the global engine behavior should be roughly approximated by the model at the end of the online modeling. In order to guarantee that one can trust the model prediction, it has to be assured that the model estimates the true system behavior correctly.

Algorithm 2 Online Modeling

input: initial test plan, objective function f and abort criterion
output: model

1: **set**: update points = initial test plan
2: **repeat**
3: measurement of the update point(s) on the test bench
4: calculate model out of all measurements
5: determine further update point(s) through optimization of the objective function $f(model)$, which depends on the model
6: **until** abort criterion = true

In the context of a Gaussian process modeling, this means that the hyperparameters $\boldsymbol{\Theta}$ are estimated correctly. Further, this does not necessarily imply that the model error (difference between the predicted mean (expected value) (4.21) and the true system behavior) has to be small, but rather this implies that the model error correlates with the predicted model uncertainty (variance) (4.22). Hence, at a high predicted variance it is acceptable that the model error is quite high, but at a small predicted variance the model should be close to the true system behavior.

Two tasks are substantial for the success of the online modeling. The choice of the update points and the selection of the abort criterion. Both tasks are discussed in the next sections.
Compared to state of the art models for online modeling, through the use of GP models new features are available, which enable a better performance than the traditional approaches. Hence, in the next sections mainly new techniques are discussed, which are suitable for an online modeling with Gaussian processes.

6.2.1 Choice of Update Points

The ambition is to choose the update points in a way that the remaining uncertainty of the model about the true system behavior is decreasing as fast as possible.

Hence, a possible approach would be to find the update point, which minimizes the remaining uncertainty of the hyperparameters $\boldsymbol{\Theta}$ at most. Therefore, the conditional entropy $H(\boldsymbol{\Theta}|\mathcal{D}_{\text{new}})$ of the hyperparameters $\boldsymbol{\Theta}$ given the estimated new measurements $\mathcal{D}_{\text{new}}$ has to be minimized, which is somewhat similar to the IAGO approach [128, 129]. The problem of this approach is that the calculation of the conditional entropy is analytically intractable, and therefore sampling methods, such as MCMC methods, have to be used [128], which are computationally too expensive for engine calibration tasks, as already discussed in section 4.3.1.1. Hence, this method was not implemented in this thesis.

In [92] other methods for this problem were examined for the mbminimize approach, and the most suitable method was the selection of update points where the variance (5.10) of the model committee is maximum, as discussed in section 5.5. Hence, also in this work the

selection of update points, where the predicted variance (4.22) is maximum, was examined, and it was found out that this approach works well for the regarded problems. The reason for this good performance is, that these update points largely reduce the uncertainty of the model, since they provide additional information at an area, which had a high (predicted) uncertainty before these update measurements were observed.

Sometimes it is useful to calculate more than a single update point at a time, e.g. when the required computing time of modeling and optimization is proportionately high compared to the measurement time on the test bench. For these cases a procedure has to be developed, which defines how these multiple update points can be calculated meaningfully.
At mbminimize the multiple updates are determined by a distance measure. After the determination of the first update point at the maximum variance, a constraint is integrated in the further optimizations, which assures that the distance between the different update points is higher than a minimum distance [92].
However, in this work another procedure has been found to be more effective. After the determination of the first update point at the maximum variance, this update point is added to the previous measurement in such a way, that the measurement value of this update is set to the expected value of the model. With this procedure the expected behavior of the system is integrated into the virtual measurement. Afterwards, with this virtual measurement the model is trained again, and the next update point is determined by the maximum variance of the new model. Usually (e.g. for MLP networks) the training of the updated model would again cause much computing time, and therefore this procedure would be computationally too expensive for engine calibration tasks. But if a GP model is used, computing time can be saved by neglecting the optimization of the hyperparameters Θ. Instead of that, the hyperparameters Θ of the previous model are used, since these parameters are not expected to change very much by a single update. With this procedure a better performance compared to the mbminimize concept had been obtained at a low computational cost.

6.2.2 Abort Criteria

In the state of the art online optimizations the online modeling stage is aborted, if a maximum number of update points is reached, which is defined manually. Compared to this, in this thesis abort criteria are presented, which can be calculated in a fully automatic way. In the following different abort criteria are examined, which achieved a good performance on practical problems in engine calibration.

Since these abort criteria evaluate the model quality, they indicate how much the model prediction can be trusted. Hence, these criteria are not only important for online modeling, but they can also provide a useful information of the model quality for model-based offline optimization.

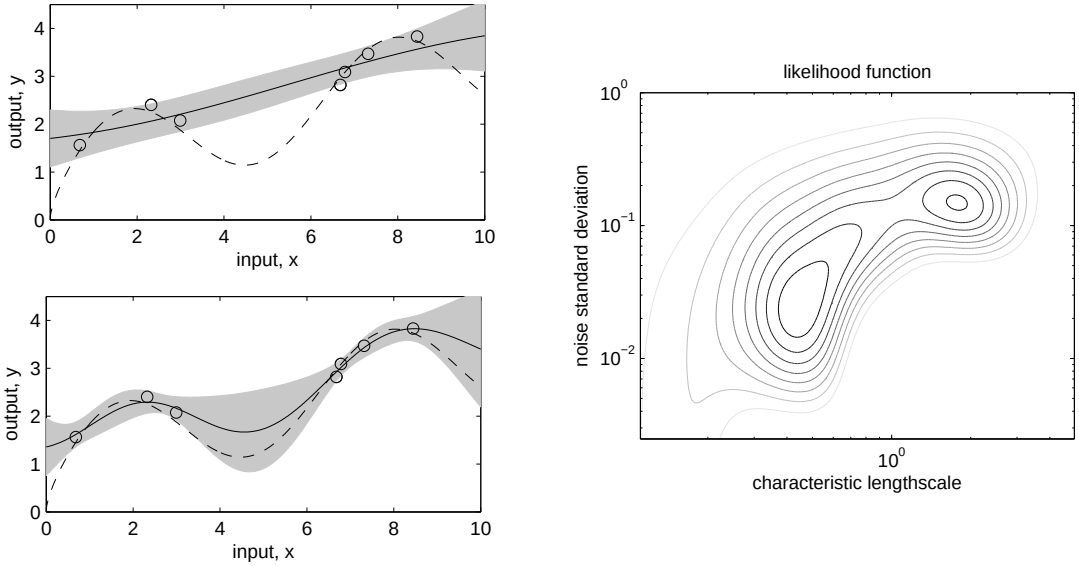
6.2.2.1 Marginal Likelihood Probability Density Function

Due to the use of Gaussian processes for online modeling, the marginal likelihood (4.18) or (2.49), respectively, of the hyperparameters $\boldsymbol{\Theta}$ of the model can be calculated. Compared to other state of the art approaches, where such an information of the probability density function of the model parameters is not available, here it can be used to evaluate the reliability of the model. Since this method has been found to give outstanding results in practical applications, it is discussed more deeply in this section.

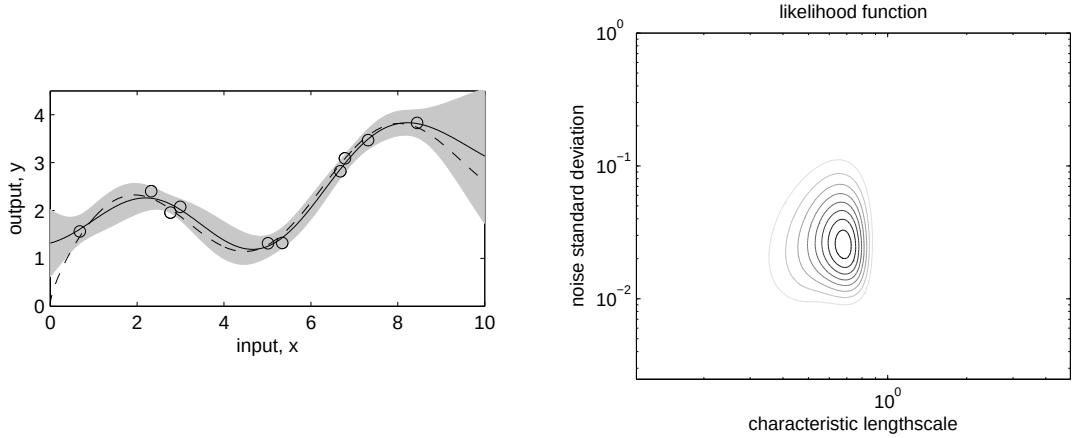
The basic idea of this new approach is illustrated in a simple theoretical example in figure 6.1. In this figure the progress of the marginal likelihood during an online optimization is illustrated.
In this demonstration a one-dimensional example is considered, which is similar to the example in figure 4.3. In the left column the function values are plotted over the input space and in the right column the marginal likelihood probability density function is drawn, which can be calculated from (2.49). For simplicity, a GP model with a normal noise assumption is considered, since this model has fewer hyperparameters. As the squared exponential kernel (2.44) is used for modeling, the GP contains the three hyperparameters $\{\theta_\sigma^2, \theta_l, \sigma^2\}$. Again for simplicity, in order to display the marginal likelihood as a two dimensional contour plot, the hyperparameter θ_σ^2 is fixed, and the likelihood is plotted over the characteristic length-scale θ_l and the noise standard deviation σ.
In the left column, the function (4.23) (dashed line) can be seen, from which training data (circles) is sampled and shifted by random noise. With this data Gaussian process models were calculated. The predicted mean (solid line) represents the estimated function value, and with the predicted variance a 95% confidence interval is drawn.
In the three parts (a), (b) and (c) the number of training data is increased.

In the upper part (a), 7 training points were sampled, and with this data the marginal likelihood (2.49) was calculated and drawn in the right plot of figure 6.1 (a). It can be seen that the likelihood of this data has two local optimums. With the values of the hyperparameters of the two local optimums, two GP models can be calculated. The first model has a higher estimated length-scale and a higher estimated noise, and it is drawn in the upper left part of figure 6.1 (a). The second model has a smaller estimated length-scale and a smaller estimated noise, and it is drawn in the lower left part of figure 6.1 (a). Hence, both models interpret the same data in a different way.
By considering the training data points and the two different models in figure 6.1 (a), it becomes clear that one cannot identify, which model is the ”correct” one. There are simply too few measurements to decide that, and therefore both models are probable. The key point is that this lack of information can be observed in the likelihood function. *If the likelihood function is multi-modal, then more different model behaviors are possible, and therefore more data is needed to identify the appropriate one, which correlates with the true system behavior at most.*

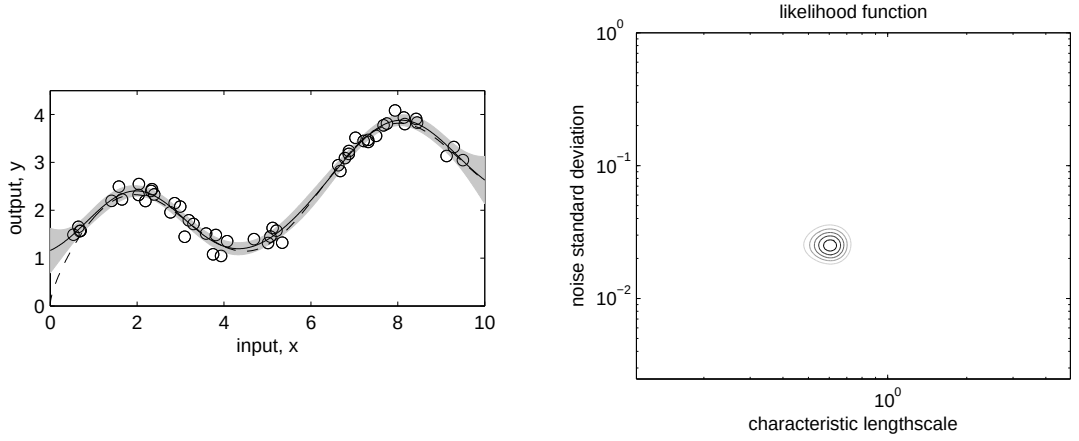
In the middle part (b) of figure 6.1, three additional training data points have been added to the



(a) Multi-modal likelihood optimum due to too few information of system behavior - two probable models



(b) Unimodal likelihood optimum with a broader distribution



(c) Unimodal likelihood optimum with a sharp distribution

Figure 6.1: Theoretical example: progress of the marginal likelihood during online modeling.

previous seven points of part (a). With these ten points the likelihood is calculated and drawn in the right plot of figure 6.1 (b). It can be seen that the likelihood of this data has now only one optimum (unimodal). Therefore, now only one optimal set of hyperparameters exists, and with this set the GP model is calculated and drawn in the left plot of figure 6.1 (b).
It can clearly be seen, that the reduction of two probable models in (a) to one suitable model in (b) seems reasonable, since the additional three measurements pointed out that the flat model in (a) seems unlikely. Hence, these three further data points provided additional information, and therefore it becomes more obvious how the true system behavior can be approximated correctly. However, since the likelihood function is still a broad distribution, there is still space for the decision which hyperparameters should be chosen.

In the lower part (c) of figure 6.1, forty additional training data points have been added to the previous ten points of part (b). With these fifty points it can clearly be seen, that the true system behavior can be determined very precisely. Therefore the hyperparameters of the model can be determined very accurately, which can also be observed from the sharp distribution of the likelihood function.
The more data points are observed, the more precisely the model parameters can be determined, and therefore the sharper the likelihood distribution appears.

This information of the likelihood distribution can be used as an abort criterion for online modeling. If the likelihood distribution is multi-modal, then further update points should be measured at the test bench. But if the likelihood is unimodal, then the online modeling can be aborted.
This approach worked well on practical problems, and in figure 6.2 a practical example is examined.
In figure 6.2 the practical data set of section 3.7 is considered. From the total set of 755 measurements (for simplicity, outliers were removed), a subset was selected and used for a multistart optimization of the likelihood of GP models for consumption, NOx and soot. With this multistart optimization the different local optimums of the likelihood function could be determined. Then the number of the training data points of the subset was increased and the procedure was performed again. The top left plot in figure 6.2 shows the progress of the number of local optimums of the likelihood function for consumption, NOx and soot over an increasing number of measurements. Similar to figure 6.1, it can be seen that the likelihood functions have many local optimums with a few amount of training data. If the number of training data is increased, only a single likelihood optimum exists.
This should be compared to the progress of the hyperparameters (which were taken at the maximum likelihood) in the other plots of figure 6.2. In the initial phase, when more than a single likelihood optimum was found, the optimal hyperparameters were strongly oscillating. After that, when only a single likelihood optimum exists, the optimal hyperparameters are not changing very much.

Hence, at the point when only a single likelihood optimum exists, the model prediction can be trusted, since the optimal hyperparameters are (approximately) determined, and the online modeling can be aborted. It was found out that this approach has an outstanding reliability in

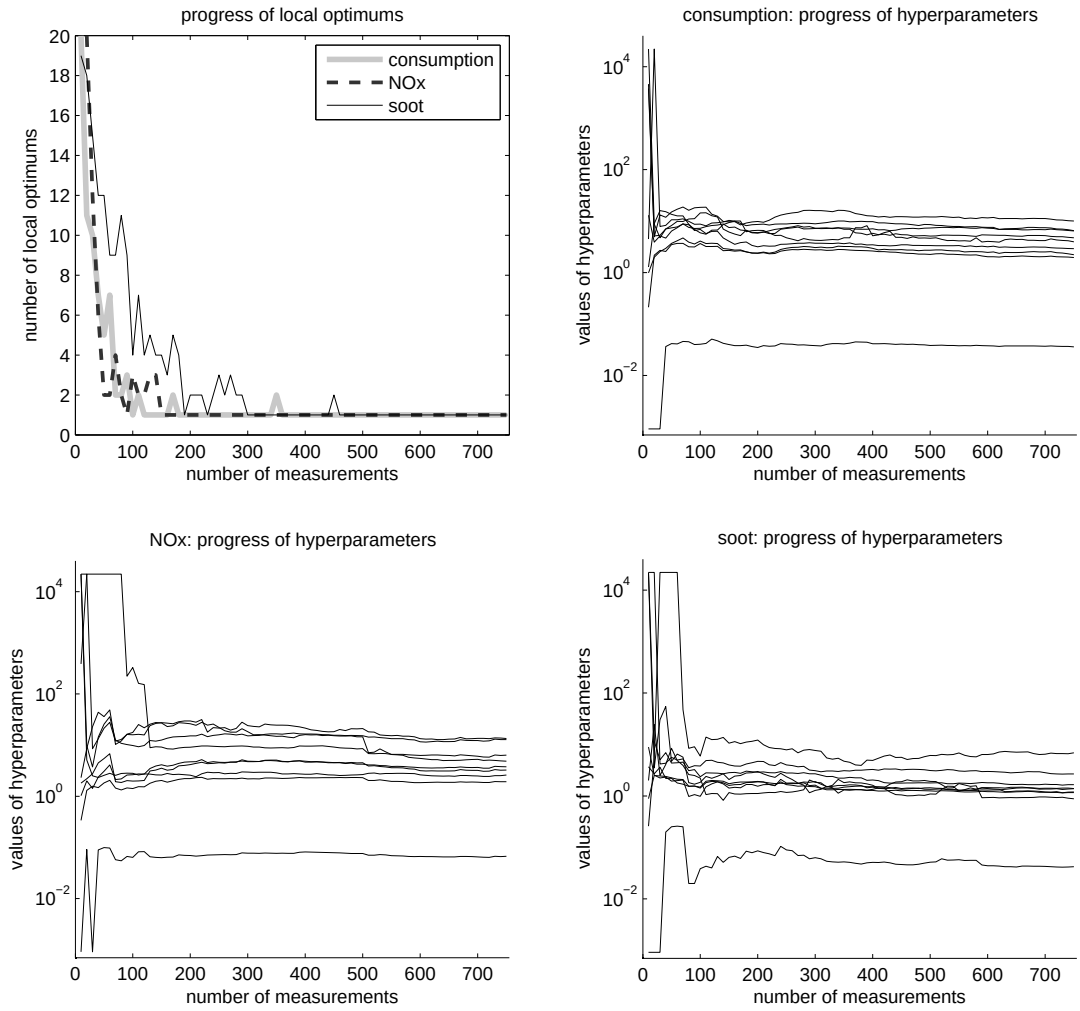


Figure 6.2: Progress of the number of local likelihood optimums and of the hyperparameters over an increasing number of measurements

practical applications and requires only a very few number of measurements.
Clearly, this technique can additionally be improved, if the shape of the likelihood function is further analyzed. As in figure 6.1 the broadness of the likelihood distribution can be evaluated in practical applications, in order to estimate the model quality and the accuracy of the prediction.

6.2.2.2 Other Methods and Discussion

During this work also various other different abort criteria have been examined. Some abort criteria, which have been considered suitable for calibration tasks, can be summarized as follows:

- examination of the marginal likelihood probability density function (subsection 6.2.2.1)
  - multistart optimization of the likelihood finds only a single optimum
  - the likelihood function is strongly peaked around the optimal value (sharp probability density function)
- examination of the progress of the cross-validation error (and in the limit: the leave-one-out cross-validation error)
- maximum variance (4.22) or (2.52), respectively, of the model is smaller than a maximum allowable variance
- maximum number of update points is reached

The choice of an appropriate abort criterion depends on the specific application.
If the primary goal is the optimization of different objectives (e.g. consumption, NOx and soot) and the reduction of the required measurements, then a suitable abort criterion of the online modeling is based on the unimodality of the marginal likelihood function. This approach requires only very few measurements and assures that the true system behavior is correctly interpreted by the model by an accurate determination of the hyperparameters, as shown above. After the online modeling, the online optimization should be performed, where the optimal areas of the objectives are determined.
Nevertheless, sometimes there are calibration tasks where the problems (and therefore also the objectives) are not known a priori, before measurements are performed on the test bench. Hence, since an online optimization cannot be scheduled before the objectives are defined, the ambition of these applications is the identification of the real system behavior with accurate models, which can be calculated after an online modeling. But if an abort criterion based on the unimodality of the likelihood is used, then there may be still large areas in the model with a high predicted variance after the online modeling.

Therefore, in order to obtain more accurate models after the online modeling, in these applications an increased time on the test bench is accepted, and other abort criteria may be suitable, which require more measurements.
A standard procedure to measure the model quality is the calculation of the cross-validation error. This quantity gives a very reliable interpretation of the approximation quality of the model, and it can be further extended, in order to integrate also the predicted variance (4.22) or (2.52) respectively, see [51]. Further, in practical applications during this work good results had been achieved with this technique. A drawback of this method is an increased computational cost, which can be reduced for GP models, if the optimal hyperparameters are only calculated once with the whole data set [51].
Another interesting approach is to abort the online modeling, if the predicted variance (4.22) or (2.52), respectively, of the model is smaller than a maximum allowable variance. This method assures that also the maximum model error is bounded, and therefore it guarantees that the model expectation is close to the real system behavior. In addition, this method can be further enhanced, if the maximum allowed variance is a function of the expected value of

the model. With this extension it can be achieved that only few measurements are taken at unsuitable engine states, and more measurements are taken at appropriate settings of the engine. Since this approach is somewhat similar to online optimization, it can only be used if some desired objectives (e.g. low consumption) are known a priori.

Moreover, it is often useful to combine different abort criteria, in order to meet the requirements of the calibration tasks. But regardless of the application, it should always be assured that the likelihood distribution is unimodal, since otherwise the interpretation of the measured data is not unique, and therefore the model cannot give a clear approximation of the engine behavior.

6.3 Online Optimization

At the end of the first stage, the online modeling stage, the global engine behavior is roughly approximated by the model. After that, the second stage, called online optimization, can be performed, where the optimal areas of the system are determined precisely with as few measurements as possible.
The combination of both stages to a framework for calibration tasks has various advantages compared to model-based offline optimization.
In model-based offline optimization a test plan is used for the measurement, which has to be fixed in advance. In contrast, model-based online optimization iteratively uses the information of the previous measurements, which are received by a permanent interaction with the test bench, in order to calculate the next optimal measuring point. This strategy increases the efficiency of the test bench utilization, since only the most useful measurements are made and other measuring points, which are not of great interest, are neglected. Further, the number of measurements (and therefore time and costs on the test bench) can be reduced to an optimal amount with model-based online optimization, since the algorithm is able to determine if already enough measurements are taken and the measurement on the test bed can be stopped.

The model, which is obtained at the end of the model-based online optimization, is very accurate in the optimal areas, since the density of the measurements is high in these regions. Therefore, it is assured that the calibration engineer can trust the models and use them for generating the engine operating maps, and it is avoided that the calibration engineer is forced to perform a further measurement, as this can be the case with model-based offline optimization, if the verification of the optimum fails, see section 1.2.2.1.

In the next subsections different algorithms for online optimization are discussed, and the most suitable one is determined. In subsection 6.3.1 single-objective online optimization is examined and the disadvantages of this state of the art approach are illustrated. In order to overcome these drawbacks, a new model-based multi-objective online optimization was developed, which is discussed in subsection 6.3.2. In the last two subsections a theoretical example and a practical application are examined.

6.3.1 Single-Objective Online Optimization

The ambition of a single-objective online optimization for base calibration is to find the global optimum of a single engine quantity (e.g. consumption) with as few measurements as possible. This is achieved by a model-based optimization, where the estimated improvement of the objective, which can be calculated from the model, is high. The other objectives (e.g. emissions and smoothness of the engine maps) are considered by an integration into constraints.
In this way, the optimum value of the main objective is searched at every operating point under consideration of the constraints.

A state of the art approach for stationary base engine calibration is mbminimize, which was discussed in section 5.5.
Also several other single-objective online optimizations were examined in this work: the EGO (efficient global optimization) approach with probability of improvement [50], the classical EGO with expected improvement [51], the EGO with generalized expected improvement [106] and the Multiple-EGO with generalized expected improvement [95]. In order to abbreviate this section and due to the general drawbacks of single-objective online optimization, which will be discussed soon, these algorithms are not examined in detail in this thesis.
During this work it was found out that the Multiple-EGO approach provides the best performance, and therefore this technique was implemented. Surprisingly, also the developers of the mbminimize technique admitted that the EGO approach achieves a better performance than mbminimize for problems without noise [92]. But since the developers of mbminimize were not able to extend the EGO approach in order that it can cope with measurement noise, they could not use this superior optimization technique for engine calibration problems [92]. However, it should be mentioned that the extension of a GP for interpolation (called kriging) to a GP for regression is straightforward, see section 2.4.2, and therefore it is rather simple to apply EGO to engine calibration tasks. The superior performance of EGO compared to mbminimize is not surprising, since mbminimize uses a committee of MLP models, which is converging to a GP in the limit of an infinite number of MLP's, each with an infinite number of neurons, see section 3.5.

6.3.1.1 Drawbacks of Single-Objective Online Optimization

The basic problem of single-objective online optimization is that only one objective is optimized, while all other objectives have to be integrated into constraints. This can become a problem if there is a trade-off between different objectives, where a compromise has to be chosen carefully.
A typical example of a practical application is the optimization of several adjustment parameters in the part load area of a diesel engine. For this problem mainly consumption and emissions (NOx, soot, HC,...) have to be minimized, whereas at the same time a smooth characteristics of the resulting engine operating maps has to be assured. Hence, at every operating point a compromise of these different objectives has to be found. Often, also different calibration engineers from different departments are involved in this process, and therefore this

compromise is typically obtained in an iterative way. Thus, the calibration engineers require models of the different objectives, which have a good quality in the Pareto optimal areas, in order to choose this compromise.

But these Pareto optimal models cannot be obtained from a single-objective online optimization. At a single objective online optimization the compromise of the different objectives has to be chosen *in advance*. Hence, a lot of *a priori information* is required to do that.
In the mbminimize concept the constraints of the emissions are fixed in advance, before the online optimization is started. This can be done, since gasoline engines are considered, where the emissions are not a big problem [53] (these gasoline engines are without direct injection, so that the soot emissions are not very high). Further, the restriction to a single objective is slightly reduced in mbminimize, as not only the optimal areas of the objective are determined, but also the regions around a local optimum are intensively measured [53]. In addition, the developers of mbminimize have already admitted that a consideration of a multi-objective problem would be more suitable [53, 54, 92], but this has not been realized until now.

Another approach is to perform a single-objective optimization of consumption, with the constraint that the summarized emissions over a driving cycle have to be lower than a (legislative) restriction. This single-objective optimization is useful for a subsequent offline optimization in the office. However, this approach cannot be performed automatically by an online optimization, since for a number of reasons nearly always a manual interaction from the calibration engineers is required:
First, as discussed in [35], this optimization does not consider a smooth characteristics of the engine operating maps. Further, the solution of this optimization is only optimal for the specific driving cycle, and the obtained measurements of the single-objective online optimization may not be useful for other applications (other driving cycles). In addition, the treatment of the summarized emissions over a definite number of operating points does not avoid the problem that a good compromise at every operating point has to be found. Clearly, by using a single aggregate objective function (a weighted linear sum of the objectives), which is constant over all considered operating points, the solution can become arbitrarily bad, see [19].
For these reasons, nearly always a subsequent manual interaction from the calibration engineers is required, in order to obtain suitable engine operating maps.

Hence, in comparison to single-objective online optimization, where a lot of constraints have to be defined a priori, before the measurement is started, in this work another approach has been developed. This multi-objective online optimization approach supports the calibration engineers by providing models, which have a high precision in the Pareto optimal areas, and with these models the engineers can determine suitable compromises for the different trade-offs and the final engine operating maps via a subsequent offline optimization in the office.
This approach does only require the definition of the most critical engine quantities (e.g. consumption, NOx and soot for a diesel engine) and the basic goal, which should be achieved (e.g. minimization of consumption, NOx and soot). With these definitions the multi-objective online optimization determines automatically the Pareto optimal areas of the engine. Therefore, only a few a priori information of the process is required and the final decisions of the calibration engineers can be performed offline.

6.3.2 Multi-Objective Online Optimization

Although there exists no online optimization which can consider more than a single objective in engine calibration, in other fields of research, especially in global optimization and machine learning, multi-objective online optimizations have already been developed. Hence, the ambition in this thesis was to identify and combine the most promising state of the art techniques from other fields of research, in order to obtain an approach, which is most suitable for engine calibration tasks.

In the next subsection the hypervolume measure (or often called $\mathcal{S}$ metric) is introduced, from which we will make extensive use in the following. In subsection 6.3.2.2 different state of the art approaches for multi-objective online optimization are compared against each other, and in section 6.3.2.3 a new approach is presented, which achieved the best performance.

6.3.2.1 The Hypervolume Measure ($\mathcal{S}$ metric)

In this subsection the hypervolume measure is discussed. This value will be crucial for the further multi-objective approaches, since it has been found out to be a superior indicator to identify the next update measurement in this thesis and also in the literature [22, 94].
The hypervolume measure was introduced by [138], and later [28] defined it as the Lebesgue measure $\mathbf{\Lambda}_{\mathrm{LE}}$ of the union of hyper-rectangles defined by a set of non-dominated solution vectors $\mathbf{A}$ and a reference solution vector $\mathbf{y}^{\max}$ that is dominated by all solution vectors in $\mathbf{A}$:

$$\mathcal{S}(\mathbf{A}) := \mathbf{\Lambda}_{\mathrm{LE}}\left(\bigcup_{\mathbf{y}\in\mathbf{A}} \{\mathbf{y}' | \mathbf{y} \prec \mathbf{y}' \prec \mathbf{y}^{\max}\}\right). \tag{6.1}$$

An illustration of the hypervolume measure of a problem with two objectives is given in figure 6.3. In this example the set of non-dominated solution vectors $\mathbf{A}$ consists of the four points $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(4)}\}$ and the gray area indicates the hypervolume measure $\mathcal{S}(\mathbf{A})$.

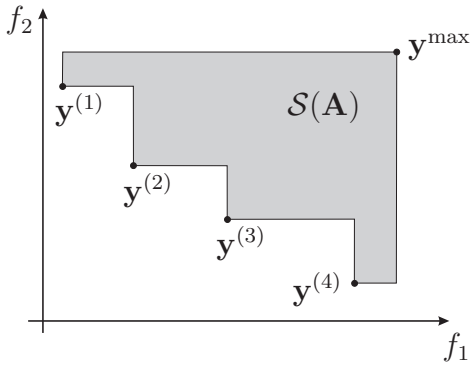


Figure 6.3: Illustration of the hypervolume measure ($\mathcal{S}$ metric).

Clearly, the better the set $\mathbf{A}$ approximates the Pareto frontier, the higher the value $\mathcal{S}(\mathbf{A})$ will be. Mathematically this was proven by [28], where it was shown that the detection of a set

$\mathbf{A}$ that maximizes $\mathcal{S}(\mathbf{A})$ is equivalent to the detection of the Pareto optimal set for any finite search space. Therefore, the hypervolume measure can be used for an indication how accurate the Pareto frontier is detected.
A drawback of the $\mathcal{S}$ metric is that the evaluation of $\mathcal{S}(\mathbf{A})$ is computationally expensive, if the number of elements in $\mathbf{A}$ or the number of objectives d_{Obj} is high. The computational effort for an evaluation of the hypervolume increases exponentially with the number of objectives d_{Obj} with recent implementations [29]. This should be compared to the computational cost of the NSGA-II approach, which increases only linear with the number of objectives d_{Obj} [20]. Hence, it will be a crucial point in the further discussion to make use of the superior performance of the hypervolume measure for multi-objective optimization, but at the same time to keep the computational effort moderate.

6.3.2.2 Comparison of State of the Art Approaches in the Context of Engine Calibration

During this work numerous different techniques had been examined: the Multi-EGO approach [48], the ParEGO technique [56], the model-based MA-SMS-EMOA method [22, 23], the SMS-EGO approach [94] and other variations of these techniques. In order to evaluate the performance of these methods, various test problems [136] were implemented and different assessment criteria were examined. Instead of going into detail on this comparison, the focus in this thesis lies on the presentation of a new approach, which is a combination of the Multi-EGO and MA-SMS-EMOA method, and on the application of this technique to engine calibration problems. Hence, only the main results of this comparison are summarized in a very short form.

A general problem of all approaches is that they were developed for an optimization of an expensive computer simulation and not for problems where noise on the measurement data occurs, like in engine calibration. Therefore, the objectives of this comparison were quite different to other comparisons which can be found in the literature. However, some similar results could be obtained.

A key result of the comparison was that the approaches, which use the hypervolume measure to identify the next update point, had a significantly better performance than the other techniques. Independent from this work, this fact was already observed by [22, 94]. However, as discussed above, the computational burden of the evaluation of the hypervolume measure can become prohibitive, if it is performed too often.
The SMS-EGO approach achieves the best performance in theory [94], but it is also evaluating the $\mathcal{S}$ metric very often. For some applications this may be acceptable, but for engine calibration tasks the computational effort of this approach is too high.
The performance of the Multi-EGO and the ParEGO technique is not as good as the MA-SMS-EMOA approach, since the MA-SMS-EMOA uses the hypervolume measure for optimization, while the Multi-EGO and the ParEGO technique use the expected improvement criterion [51], which is not so effective [22]. However, the Multi-EGO has an interesting property, since it

uses a NSGA-II optimization, in order to obtain a preselection for the suitable update points. This reduces the computational effort of this approach.
Hence, in this work the Multi-EGO and the MA-SMS-EMOA technique were combined to a new approach, in order to exploit the advantages of both techniques. In this new approach, similar to the Multi-EGO method, at first a NSGA-II optimization is performed, in order to obtain a preselection of appropriate update points at a low computational cost. After that, similar to the MA-SMS-EMOA approach, the update point is chosen, which maximizes the hypervolume measure. With this new technique, which is discussed in the next section, the best performance for calibration tasks could be obtained.

6.3.2.3 A New Approach

The basic functionality of the new approach for model-based online optimization for engine calibration is given in algorithm 3.

Algorithm 3 Multi-Objective Online Optimization (for Engine Calibration)

input: initial measurement (e.g. from online modeling), objectives (e.g. minimization of consumption, NOx and soot) and abort criterion
output: models of objectives

1: calculate initial models out of initial measurement
2: **repeat**
3: determine further update point(s) through the new optimization approach in two steps:
- determine a suitable preselection of further update point(s) through performing a multi-objective optimization
- determine suitable update point(s) out of the preselection through the hypervolume measure

4: measurement of the update point(s) on the test bench
5: calculate models out of all measurements
6: **until** abort criterion = true

As in the online modeling in section 6.2, for the online optimization one or more update points can be determined in every stage. In the next sections only a single update point is considered, but it should be mentioned that the procedure of determining more than a single update point is equivalent to the suggested technique in section 6.2.1.
In the following, the two steps of the new approach, the determination of the preselection and the determination of the final update point, are discussed. After that, different abort criteria of the multi-objective online modeling are examined.

Determination of the Preselection
Compared to other state of the art approaches (e.g. MA-SMS-EMOA), where the hypervolume measure is evaluated for every potential update point, this new approach is performing a preselection of suitable update points through a multi-objective optimization with the NSGA-II algorithm. This procedure is computationally cheaper than an evaluation of the hypervol-

ume measure for all potential update points.
In this work it has been found that the lower confidence bounds $lcb_{\omega,i}$ of the models are appropriate objectives f_i for the multi-objective optimization (5.7). These lower confidence bounds are given by [50]

$$lcb_{\omega,i}(\mathbf{x}) := \mathbb{E}_i(\mathbf{x}) - \omega \cdot \sqrt{\mathbb{V}_i(\mathbf{x})} \tag{6.2}$$

where $\mathbb{E}_i(\mathbf{x})$ and $\mathbb{V}_i(\mathbf{x})$ are the predicted mean (4.21) and the predicted variance (4.22) of the i-th model. The value ω defines how many standard deviations of the predicted uncertainty of the model should be subtracted from the predicted mean. Hence, it indicates how much the predicted uncertainty is weighted against the mean value of the model [50]. In this thesis in all applications $\omega = 2$ is chosen, since good results could be obtained with this setting.

Determination of Suitable Update Point(s)
Having found the Pareto frontier with the NSGA-II in the previous stage, now the most suitable update point is determined out of this preselection through the hypervolume measure. Since the results of the NSGA-II optimization are already the lower confidence bounds of the models, the selection of the maximum hypervolume measure of these points corresponds to the selection of the maximum potential improvement LBI_ω [22], which is given by

$$\text{LBI}_\omega(\mathbf{y}_{\text{pot}}) := \mathcal{S}\left(\mathbf{y}_{\text{pot}} \cup \mathbf{T}_{\text{op},j}\right) - \mathcal{S}\left(\mathbf{T}_{\text{op},j}\right) \tag{6.3}$$

where $\mathbf{y}_{\text{pot}}$ is a solution vector of the NSGA-II optimization and $\mathbf{T}_{\text{op},j}$ are the measurements of the objectives at the considered operating point of the j-th update. Equation (6.3) is evaluated for every single solution vector of the NSGA-II optimization, and the vector which maximizes LBI_ω is chosen for the next update point.

Abort Criteria
Abort criteria which have been found to be useful in this thesis are:

- precision of the models in the Pareto optimal areas is sufficient; this means that the maximum variance of the models in the Pareto optimal areas is smaller than a maximal allowable variance
- potential of increasing the hypervolume is lower than a minimal allowable value
- maximal number of update points is reached

As for the online modeling, the choice of an appropriate abort criterion for online optimization depends on the specific application. In the following subsections we will see that a combination of the abort criteria is suitable for practical applications.

6.3.2.4 A Theoretical Example

In this section a theoretical example of a multi-objective online optimization is considered. The advantage of this theoretical example is that the true Pareto frontier of this problem is known, and therefore we are able to demonstrate the convergence of the optimization to the Pareto optimal areas.
However, the conditions of this example should be as close as possible to practical conditions in engine calibration, in order to draw reasonable conclusions for practical applications. Therefore, the regarded system which should be optimized consists of models of engine quantities. Assuming that these models predict the true engine behavior precisely, with this procedure the performance of the multi-objective online optimization can be analyzed under practical conditions.

In this example the consumption and NOx measurements of section 3.7.1 are examined. A detailed description of these measurements was given in section 3.7, where it was also shown that the models of these quantities approximate the true engine behavior precisely.
Hence, in this theoretical example an online optimization with two objectives is considered, and therefore the objective space is two-dimensional. In the next section, where a practical application to a diesel engine is examined, a three-dimensional objective space is regarded. Generally, the algorithms in this thesis can cope with an arbitrary high number of objectives, but it should be mentioned that the complexity of the optimization problem (and therefore also the number of required measurements) increases if more objectives are considered.

In figure 6.4 the theoretical example is illustrated.
With the total set of the 780 consumption and NOx measurements, models were calculated, which represent the true engine behavior, as mentioned above. From these models the Pareto frontier was determined by an extensive multi-objective optimization. After that, a subset of 200 measurements was randomly selected and used as an initial measurement for the online optimization. In this online optimization the operating point at engine speed $n_{engine} = 1525$ rpm and engine torque $M_{engine} = 780$ Nm is considered. The initial measurements at this operating point are marked in the plots of figure 6.4 by crosses, and the Pareto frontier at this operating point consists of two connected sets, which are indicated by the two solid lines.
After the initialization, the multi-objective online optimization is performed, and in the plots of figure 6.4 the progress of the update measurements (stars) is illustrated. In order to simulate real conditions for the optimization, a measurement noise is added to all update points. Further, the Pareto optimal measurements in the example are marked by circles in figure 6.4.

Overall, 100 update points were simulated. It can be seen that the update points are fast converging towards the true Pareto frontier, and that only a few update points are required in order to identify the Pareto optimal areas. In addition, it can be determined that nearly no update point is dominated by the initial measurements, but all initial measurements are dominated by the update points. Further, it should be mentioned that more measurements are on the Pareto frontier, if the number of update measurements increases, which means

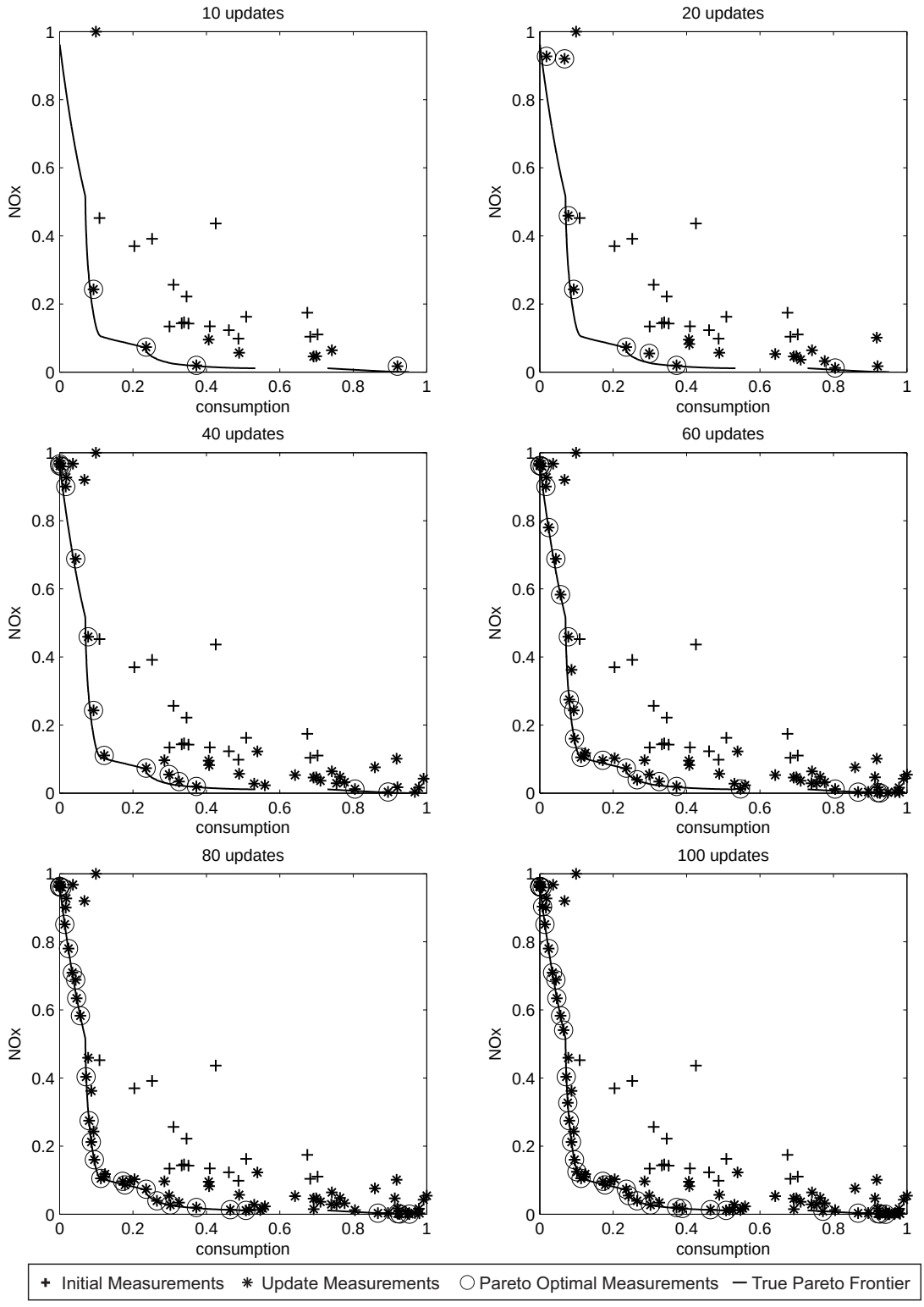


Figure 6.4: Theoretical example of a multi-objective online optimization with two objectives. In the different figures the progress of the online optimization is shown.

that the density of the Pareto optimal measurements increases. This is also indicated by the uniform distribution of the Pareto optimal update points. Hence, dependent on the number of update measurements the Pareto frontier can be approximated to arbitrary accuracy with the multi-objective online optimization. Moreover, it can be seen that also the edges of the Pareto frontier were determined precisely.

The fact, that the optimization results are getting better with an increasing number of measurements, is also indicated in figure 6.5.

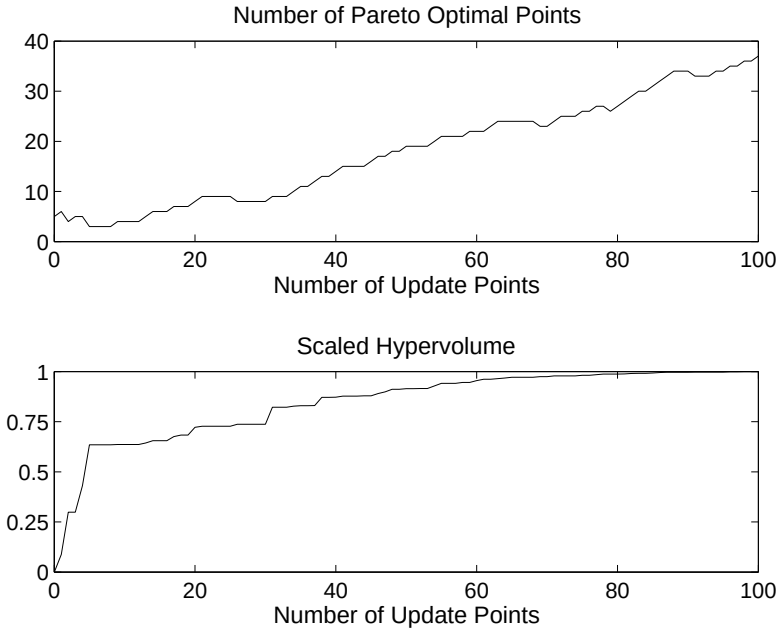


Figure 6.5: Number of Pareto optimal update points and the scaled hypervolume for the model-based multi-objective online optimization for the theoretical example of figure 6.4.

In figure 6.5 the number of Pareto optimal points and the hypervolume (scaled to an interval of [0 1]) are plotted over the number of update points. It can be seen that the scaled hypervolume increases rapidly at the beginning, which indicates that the update points are fast converging towards the true Pareto frontier. After that, the limits of the Pareto frontier are explored and the number of Pareto optimal points increases, which illustrates that the density of the Pareto optimal measurements increases.

Hence, with this theoretical example it could be demonstrated that the new multi-objective online optimization is fast converging towards the optimal areas and that the resulting measurements of the optimization are well distributed.

6.3.2.5 A Practical Application to a Diesel Engine

After this theoretical example, in this section a practical application of the multi-objective online optimization is examined. For this task the modeling and optimization algorithms have to be in a permanent interaction with the test bench. Therefore, a communication software was implemented, which bidirectionally transfers the data between the automation software of the test bench and the methods described above. This software and other aspects of implementation are discussed in chapter 7. Here, we want to examine the accuracy and the performance of the model-based multi-objective online optimization in practice.

The goal of this application was the optimization of consumption of a diesel engine in the part load area, by simultaneous consideration of the NOx and soot emissions, in order to meet the emission standards. The adjustment parameters, which should be optimized, were the main injection time, injection pressure and quantity and time of the post injection. Since a global modeling is performed, also engine speed and torque are taken as model inputs. This leads to an 6-dimensional input space.
As the exact values for the constraints for NOx and soot at each operation point are not known in advance, a multi-objective online optimization was performed. With this multi-objective online optimization the engineer can either directly choose one of the Pareto optimal measurements as a compromise, or these measurements can be used in order to improve the model quality in the Pareto optimal areas, which reduces the uncertainty of the optimal solution, if the Pareto frontier is dense enough.

For the initial data 483 measurements were taken at 33 operating points. With these measurements the initial global models were calculated and the abort criterion of the online modeling of section 6.2.2.1 had been checked. After this, the multi-objective online optimization was performed on different operating points. Since the results on the different operating points are very similar, here we consider a single operating point at engine speed $n_{engine} = 1300$ rpm and engine torque $M_{engine} = 215$ Nm as an example. The results for other operating points are given in appendix A.

The objectives were the minimization of consumption, NOx and soot. This leads to a three-dimensional objective space and a two-dimensional Pareto frontier. The measurements in the objective space at the discussed operating point are plotted in figure 6.6. In the left column a three-dimensional view of the objective space is shown. Since the positions of the measurements in this three-dimensional plot are hard to determine, also two-dimensional projections of the objective space are plotted in the middle and right columns. With these projections it is easier to identify the optimal measurements, which should be close to the low values of consumption, NOx and soot. At the top row the initial measurement is plotted. The Pareto optimal measurements are marked with circles. In the lower rows the progress of the multi-objective online optimization is shown. The second row displays the measurements after 6 update steps of the online optimization. The third and fourth row show the development after 12 and 18 update measurements.
It can be seen that during the progress of the optimization routine, more and more measure-

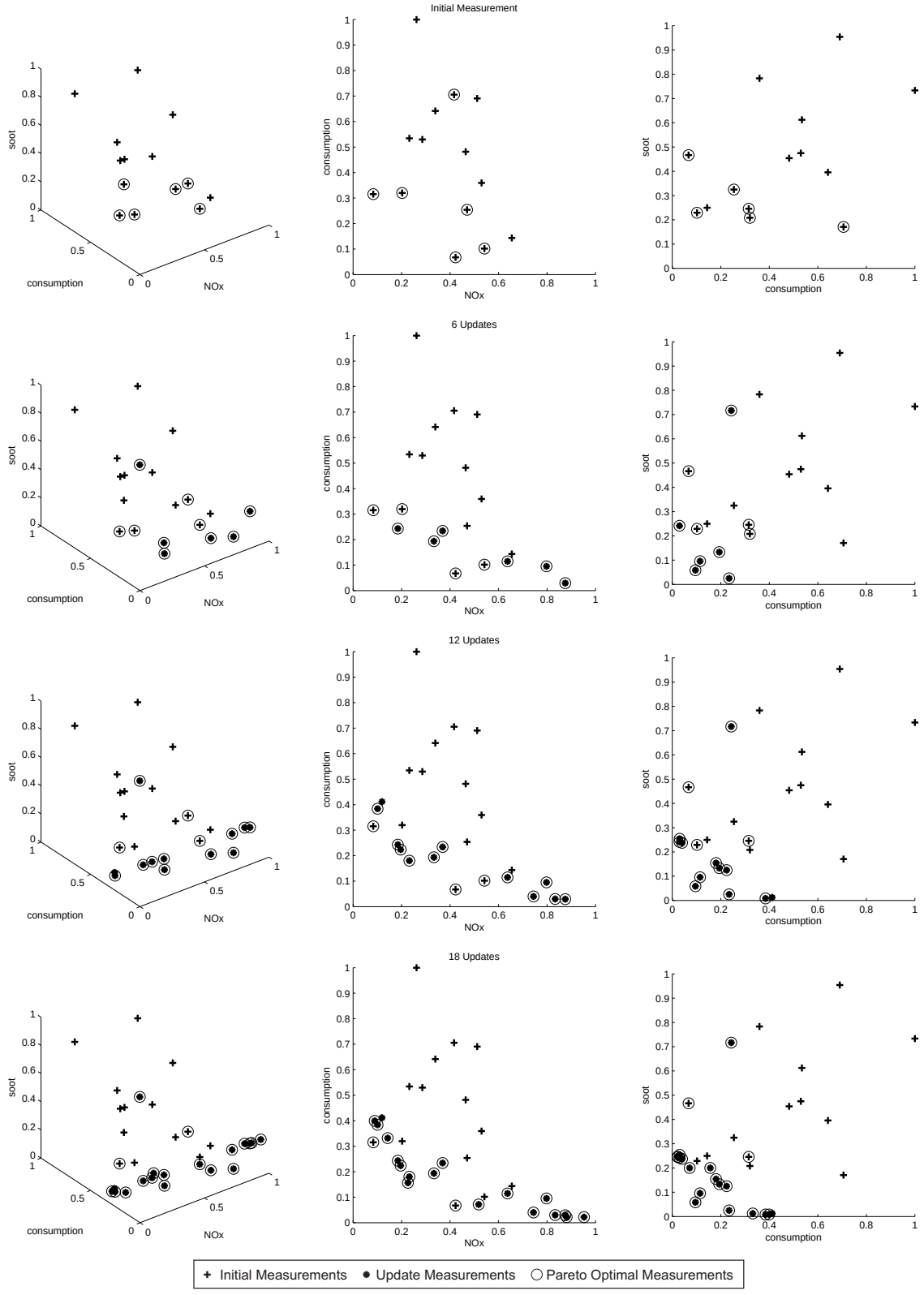


Figure 6.6: Practical model-based multi-objective online optimization. Here the consumption, NOx and soot measurements at the operating point $(n_{engine}, M_{engine}) = (1300\ \text{rpm}, 215\ \text{Nm})$ are plotted in the objective space. On the left a three-dimensional view of the objective space is shown, on the middle and right two-dimensional projections of the objective space are shown.

ments are taken in Pareto optimal areas. Further, the update points are well distributed in the objective space and try to cover the whole Pareto frontier evenly. Hence, with more and more update points the measurements which indicate the Pareto frontier will get more and more dense. This makes it easy for the engineer to get a good impression of the optimal areas, and therefore a good compromise for the trade-off between consumption, NOx and soot can easily be chosen.

The fact, that the Pareto frontier will get more dense with more update points, is also indicated in figure 6.7.

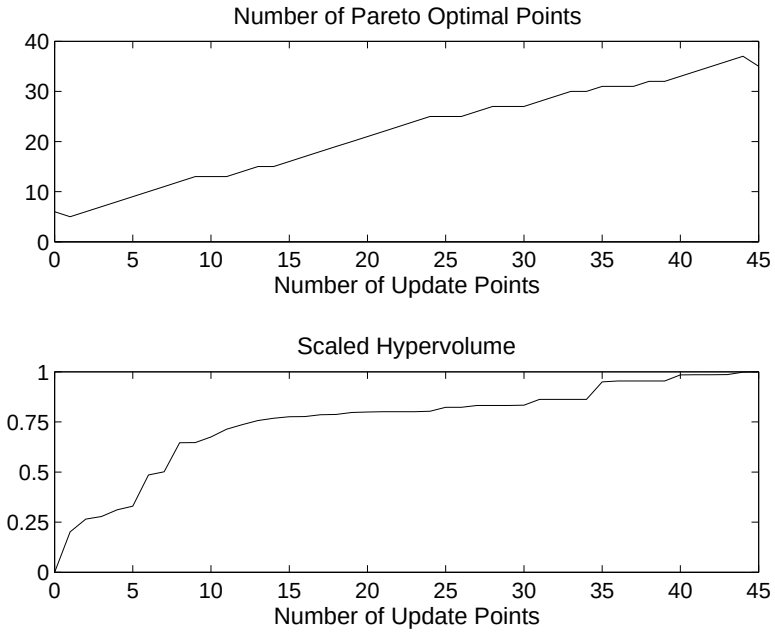


Figure 6.7: Number of Pareto optimal update points and the scaled hypervolume for the model-based multi-objective online optimization at the operating point $(n_{engine}, M_{engine}) = (1300\ \text{rpm}, 215\ \text{Nm})$.

In this figure the number of Pareto optimal points during 45 updates of the multi-objective online optimization is plotted. From this plot it can be seen that most of the update measurements are Pareto optimal and therefore the Pareto frontier can be approximated more and more accurately during the optimization.
Further, in figure 6.7 the hypervolume, which is scaled to the interval [0 1], is plotted against the number of update points. This plot indicates the convergence of the optimization routine. It can be seen that most of the optimization of the hypervolume can be achieved in the first updates. After 6 updates over 50% improvement of the hypervolume can be obtained and after 15 updates over 80% of the hypervolume can be achieved. Hence, in a practical realization only a few updates would have been needed to get an impression of the full potential for the optimization. The later updates cannot improve the hypervolume very much, but they allow a dense representation of the Pareto frontier.

Clearly, the convergence rate of the optimization may vary if the complexity of the optimization problem is changed (e.g. if more adjustment parameters should be optimized or if there is more noise on the measurement data).
But obviously, it can be seen that the model-based multi-objective online optimization performs well under practical conditions.

Besides the advantages of the multi-objective online optimization, the whole process of measurement, modeling and optimization in a loop demonstrates the robustness and accuracy of the whole concept. At any time for every update point, the modeling and optimization is performed in a fully automatic way without any manual interaction. Since the online optimization achieved accurate results, this means that also the modeling, which is recalculated in every single update step, is very accurate. This is in contrast to other state of the art modeling approaches for stationary base engine calibration, where a manual interaction is often required, in order to tune the models to get an accurate prediction. Hence, these results also emphasize the use of Gaussian processes for modeling in engine calibration.

6.4 Conclusion and Discussion

In this chapter a model-based online optimization has been presented. The whole approach consists of two stages: an online modeling and an online optimization stage. For both stages new techniques were presented, which have a better performance for engine calibration tasks than state of the art approaches from other fields of research.

Clearly, the utilization of the presented framework for model-based online optimization is most useful, if a complex calibration problem is considered, with e.g. many adjustment parameters and difficult engine quantities (like soot).
If a simple calibration problem is considered, then a few measurements from a simple test plan and a unique model-based offline optimization may be sufficient. However, for highly complex problems either numerous measurements are required for an accurate offline optimization, or many process loops are necessary, where the test planning, measurement, modeling and optimization are performed iteratively, in order to converge to the optimal areas. Nevertheless, these iterative process loops require various manual interactions from the calibration engineers, which are time- and cost-intensive. In contrast, the presented framework for model-based online optimization can be performed in a fully automatic way, with a high quality of the results at a low amount of measurements.
Hence, this approach assists the calibration engineers. Without this framework the calibration engineers not only have to identify a suitable setting for the engine adjustment parameters, but they also have to perform extensive additional work, in order to design appropriate test plans and to verify the models and the optimization results. These new techniques take the pressure off the calibration engineers, so that they can concentrate on finding a suitable compromise out of the Pareto optimal solutions.

The advantages and properties of the new model-based online optimization approach can be

summarized as follows:

- *Optimal utilization of the test bench time.* Compared to model-based offline optimization, only the most important measurements are performed with this approach. Therefore time and costs on the test bench can be remarkably reduced.
- It is *avoided that additional measurements are required* at the end of the online optimization. In model-based offline optimization the verification of the optimum may fail, if the quality of the model at the optimal area is not sufficient, and therefore an additional measurement has to be performed, which increases the precision of the model. Hence, the calibration engineer is forced to design a new test plan and repeat the measurement, modeling and optimization. Clearly, this is wasteful of time and resources and does not increase the user acceptance of calibration tools. In contrast, with the presented framework the final model automatically has a high accuracy at the optimal areas, and therefore the risk that an additional measurement is required is minimized.
- *Only few a priori information is required.* The (unique) state of the art approach for model-based online optimization in engine calibration is the single-objective online optimization of mbminimize. This approach requires that the compromises of different objectives have to be chosen in advance, before the online optimization is performed on the test bench. Hence, a lot of a priori information is required, which is often not available. In comparison, in this thesis a multi-objective online optimization is presented. With this approach the Pareto optimal areas of the calibration problem are identified, and the calibration engineer is able to choose a compromise afterwards, by a subsequent offline optimization in the office. Hence, only the main objectives of the calibration problem have to be defined a priori.

Chapter 7

IMPLEMENTATION

This chapter focuses on the implementation of all approaches, which were mentioned above, into a calibration tool for model-based offline and online optimization. Hence, the following sections emphasize that the techniques in this thesis were not only examined under a theoretical viewpoint with a practical verification, but also that these approaches are already implemented for a further usage.
The main implementation was performed in MATLAB, but some computationally expensive routines were implemented in C/C++ and Fortran and integrated in the main implementation over MEX interfaces.

All algorithms which have been examined in this thesis were integrated into a toolbox named KASIO, which is discussed in section 7.1. In order that a calibration engineer can use these algorithms, a GUI called PAoptimizer-Matlab-Edition was realized, which is discussed in section 7.2. In section 7.3 the connection to the test bench and the realization of the online optimization is examined.

7.1 The KASIO Toolbox

During this work nearly all mentioned algorithms in this thesis were implemented and tested. The final choice of the most useful algorithms had been integrated in a toolbox named KASIO, which stands for KRATZER System Identification and Optimization toolbox.

The KASIO toolbox consists of two layers: an object-oriented layer and a layer of computationally expensive implementations. The user of the toolbox only has to interact with the object-oriented layer, which can be easily operated through simple interfaces. After an error checking of the user inputs, the computationally demanding routines in the other layer are automatically executed. Hence, with these two layers a simple and error-free use is possible.

The content of the KASIO toolbox is given in table 7.1.

Classes in KASIO:	
KASIO_VP	class: design of experiments
KASIO_MB	class: modeling
KASIO_OP	class: optimization
KASIO_OBJECTIVE	class: definition of objective functions and constraints
KASIO_MBOO	class: model-based online optimization
KASIO_VP - class for design of experiments	
FullFak	full factorial design
Random	random (uniform) design
LatHyp	latin hypercube design
Dopt	D-optimal design via k-exchange
KASIO_MB - class for modeling	
Gauss_process	Gaussian process with normal noise assumption
Gauss_process_rob	Gaussian process with Student's-t noise assumption
MLP_BR	MLP network with Bayesian regularization
Poly_free_Coeff	polynomial stepwise regression
Poly	polynomial regression with a priori specification of the considered coefficients
(SVM	support vector machine - not yet integrated)
KASIO_OP - class for optimization	
Single_Obj_glob	single-objective optimization for the global optimum
Single_Obj_Mult	single-objective optimization with a multistart approach
Multi_Obj	multi-objective optimization
KASIO_MBOO - class for model-based online optimization	
Single_MBOO	single-objective online optimization
Multi_MBOO	multi-objective online optimization

Table 7.1: Content of the KASIO toolbox - 01.06.2012.

7.2 The PAoptimizer-Matlab-Edition

In order that a calibration engineer can use the algorithms of the KASIO toolbox, a GUI named PAoptimizer-Matlab-Edition was realized.

With this GUI the calibration engineer can manage different applications in the project directory, store and load numerous settings, import the measurement data out of xls-files and perform a modeling and optimization. Further, various different visualization techniques were integrated into the PAoptimizer-Matlab-Edition, such as intersection plots, measured-predicted plots, 3-D model plots and measurement analyzing plots. In addition, via the PAoptimizer-Matlab-Edition the user is able to connect to the test bench and to perform an online optimization, which is discussed in the next section.
Figure 7.1 shows some screenshots of the PAoptimizer-Matlab-Edition.

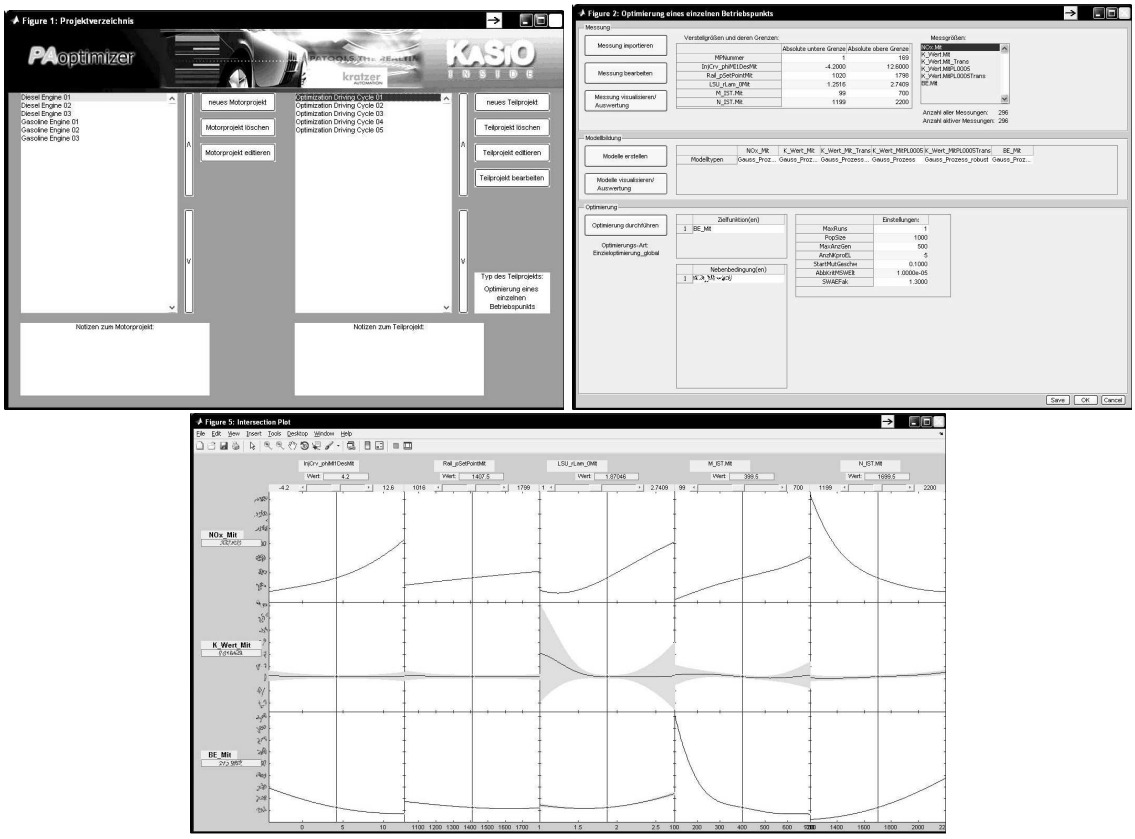


Figure 7.1: Screenshots of the PAoptimizer-Matlab-Edition

7.3 Connection to the Test Bench and Online Optimization

During this work the test benches were automated with the PAtools Software Suite from KRATZER AUTOMATION AG. The PAtools Real-Time-System is the interface to the test stand and performs the execution of the test program.
In order to perform an online optimization, the modeling and optimization algorithms have to be in a permanent interaction with the test bench. Hence, a communication software was implemented and integrated in the PAoptimizer-Matlab-Edition, which bidirectionally transfers the data between the PAtools Real-Time-System and the online optimization procedure.
After the specification, the online optimization passes the settings of the next measuring point and further parameters to the PAtools Real-Time-System, which is performing the measurement. The limit monitoring and other critical tasks are performed in the real time system. At the end of the measurement, the measured quantities are received by the online optimization, which decides if further measurements have to be performed or if the procedure can be stopped.

Chapter 8

CONCLUSION AND FUTURE WORK

This thesis has presented a complete framework for modeling and optimization for stationary base engine calibration. The results range from clear modeling recommendations from a comprehensive comparison, new robust modeling approaches to new model-based online optimization concepts. These new contributions enhance the performance for modeling and optimization, and therefore they are able to reduce time and costs on the test bench, improve the reliability of modeling and optimization results, assist the calibration engineers and increase the user acceptance of model-based techniques in engine calibration.

Based on the presented requirements on the modeling, the determination of the most suitable modeling technique for stationary base calibration was performed, which was an important open question in this field of research. The Gaussian process model could be identified as the most promising type of modeling. With the Gaussian process approach, the highest precision with a low amount of measurements can be achieved, the modeling can be fully automated with the maximum marginal likelihood method, a dependable performance on complex problems can be obtained and an accurate prediction of the uncertainty of the model can be estimated.

However, recent approaches in engine calibration do not consider outliers in the measurements and an automatic adaption to bad distributed data. Therefore, based on the results of the model comparison, the most promising modeling technique was enhanced in order that a new robust modeling framework for stationary base engine calibration could be obtained. An automatic transformation of the measurement data ensures that the modeling assumptions on the data distributions are met. Even if outliers are contained in the data set, a robust Gaussian process formulation guarantees that the modeling is asymptotically unbiased, meaning that the model is tending to the real engine behavior as the number of measurements tends towards infinity. It was shown that this new framework has an outstanding performance on challenging practical data sets.

Since state of the art model-based online optimizations for engine calibration do not use a fully probabilistic approach and can only handle a single objective function, a new, improved online optimization approach was presented. As a Gaussian process modeling is used, additional information, such as an accurate prediction of the variance and the marginal likelihood probability density function of the model parameters, can be exploited for the online modeling, in order to obtain an increased performance at a lower amount of measurements compared

to other approaches. With the new multi-objective online optimization more objectives can be regarded and the Pareto optimal areas can be determined. A fast convergence under practical conditions was demonstrated.

Not only the requirements of the calibration process have been regarded throughout the thesis, but also the needs of the calibration engineers have been considered during the development of each new approach. The robust modeling technique can be performed in a fully automatic way and the model-based online optimization can be easily parameterized. Both techniques assist the calibration engineers by providing models with an increased accuracy, since, on the one hand, an improved modeling technique is used and, on the other hand, additional measurements are placed in the Pareto optimal areas. Hence, instead of removing outliers manually from the data set, investigating in complex mathematical issues, or performing a verification of models and optimization results, with these contributions the calibration engineers can concentrate on their main tasks.

The following areas of further research are proposed to further increase the performance of the concepts presented in this thesis:

- **MCMC methods:** As discussed throughout the thesis, MCMC methods provide a better performance than other approximation techniques. However, traditional approaches for MCMC approximation are computationally too expensive for engine calibration tasks. Nevertheless, if more computing power will become available in the future and more sophisticated MCMC algorithms can be developed, then an increased performance for outlier robust Gaussian process models and for online optimizations may be achieved.
- **Smooth ECU maps:** An explicit integration of the smoothness of the ECU maps as an additional objective could improve the multi-objective online optimization. A possible approach for the consideration of smooth engine operating maps in an optimization is, for example, given in [54, 92], and this technique could be integrated in the model-based online optimization.
- **Dynamic engine calibration:** Most of the presented approaches in this thesis can also be applied to dynamic engine calibration problems. By doing so, the reduction of the computational effort of the modeling techniques will be a major challenge. However, as already indicated in chapter 3, a lot of modeling approaches exist, such as sparse kernel machines, which have a lower computational effort than a full Gaussian process modeling, and it is believed that these techniques can be combined with the presented online optimization and robust modeling approaches.

Appendix A

FURTHER EXPERIMENTAL RESULTS OF SECTION 6.3.2.5

As mentioned in section 6.3.2.5, in this appendix further experimental results of the practical application of the model-based multi-objective online optimization are given.
A detailed description of the optimization procedure was given in section 6.3.2.5. Here, only the results at other operating points are presented. Figure A.1 considers the operating point at engine speed $n_{engine} = 900$ rpm and engine torque $M_{engine} = 930$ Nm, figure A.2 considers the operating point at engine speed $n_{engine} = 900$ rpm and engine torque $M_{engine} = 1264$ Nm and figure A.5 considers the operating point at engine speed $n_{engine} = 1100$ rpm and engine torque $M_{engine} = 1244$ Nm. In the figures A.3, A.4 and A.6 the number of Pareto optimal points and the scaled hypervolume is plotted of the applications of the figures A.1, A.2 and A.5.

As in the other operating point in section 6.3.2.5, it can be seen that the online optimization is converging fast to the Pareto frontier. Further, the update points are well distributed in the objective space and try to cover the whole Pareto frontier evenly.

The results of these additional experiments re-emphasize the advantages of the multi-objective online optimization, which were discussed extensively in chapter 6.

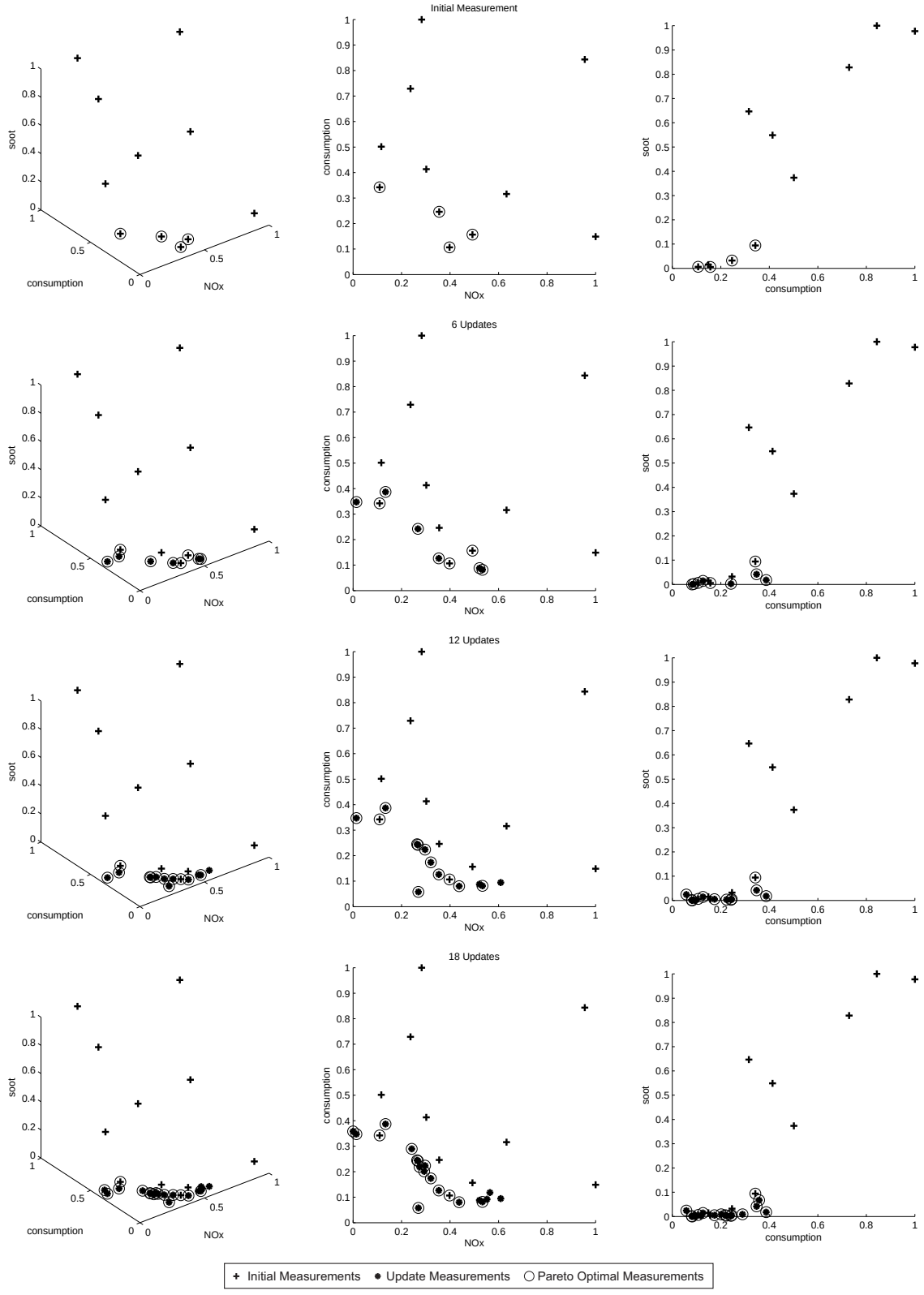


Figure A.1: Practical model-based multi-objective online optimization. Here the consumption, NOx and soot measurements at the operating point $(n_{engine}, M_{engine}) = (900\,\text{rpm}, 930\,\text{Nm})$ are plotted in the objective space. At the left a three-dimensional view of the objective space is shown, in the middle and right two-dimensional projections of the objective space are shown.

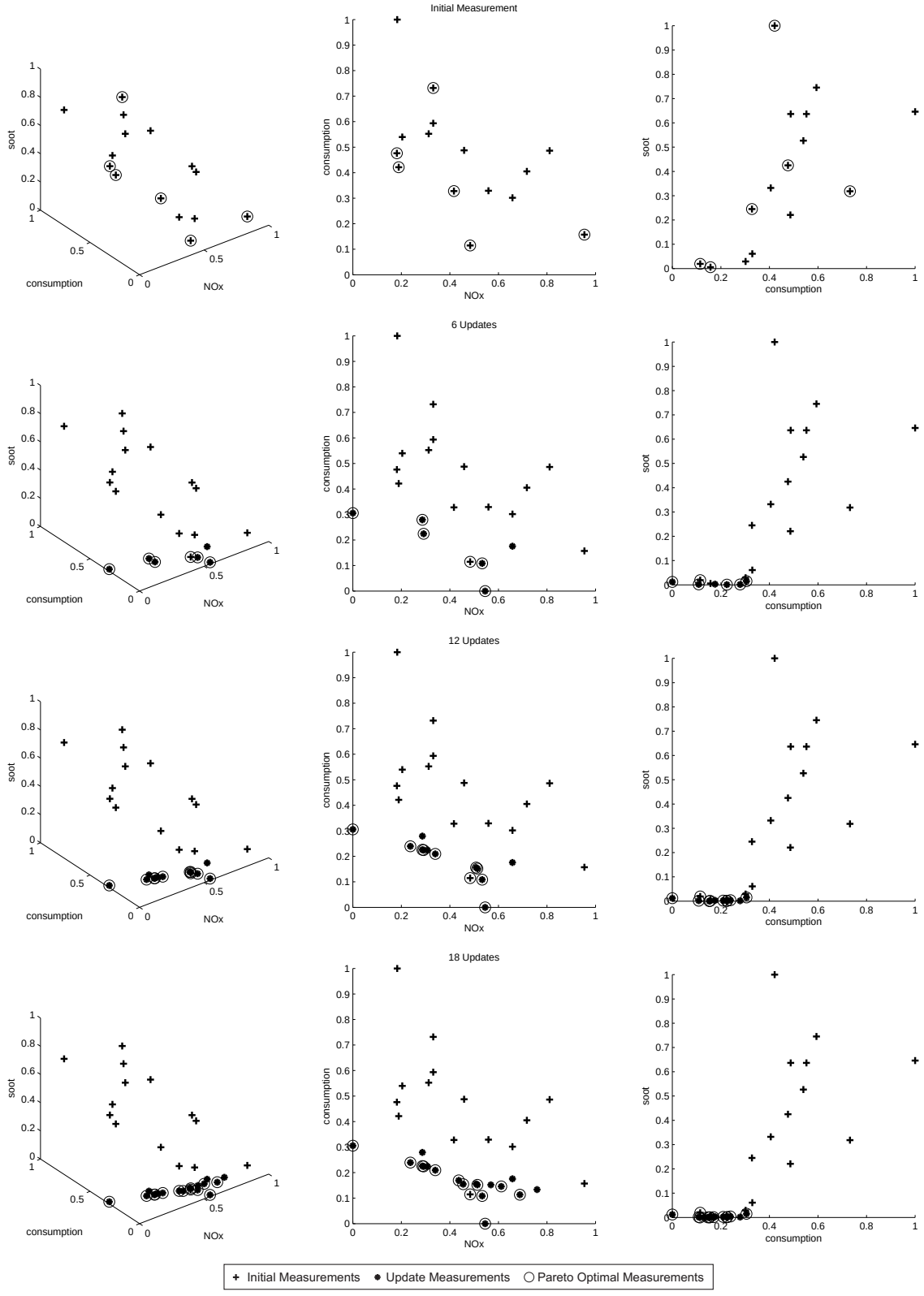


Figure A.2: Practical model-based multi-objective online optimization. Here the consumption, NOx and soot measurements at the operating point $(n_{engine}, M_{engine}) = (900\ \text{rpm}, 1264\ \text{Nm})$ are plotted in the objective space. At the left a three-dimensional view of the objective space is shown, in the middle and right two-dimensional projections of the objective space are shown.

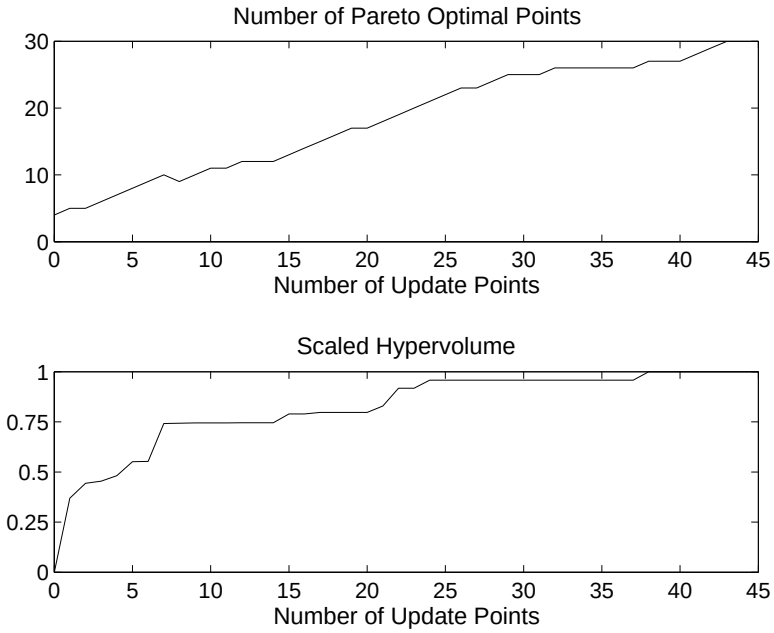


Figure A.3: Number of Pareto optimal update points and the scaled hypervolume for the model-based multi-objective online optimization at the operating point $(n_{engine}, M_{engine}) = (900 \text{ rpm}, 930 \text{ Nm})$.

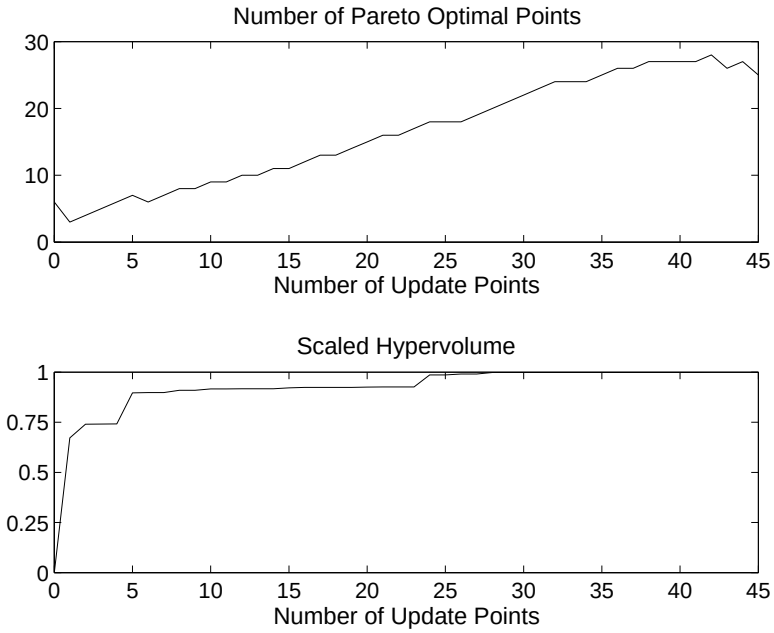


Figure A.4: Number of Pareto optimal update points and the scaled hypervolume for the model-based multi-objective online optimization at the operating point $(n_{engine}, M_{engine}) = (900 \text{ rpm}, 1264 \text{ Nm})$.

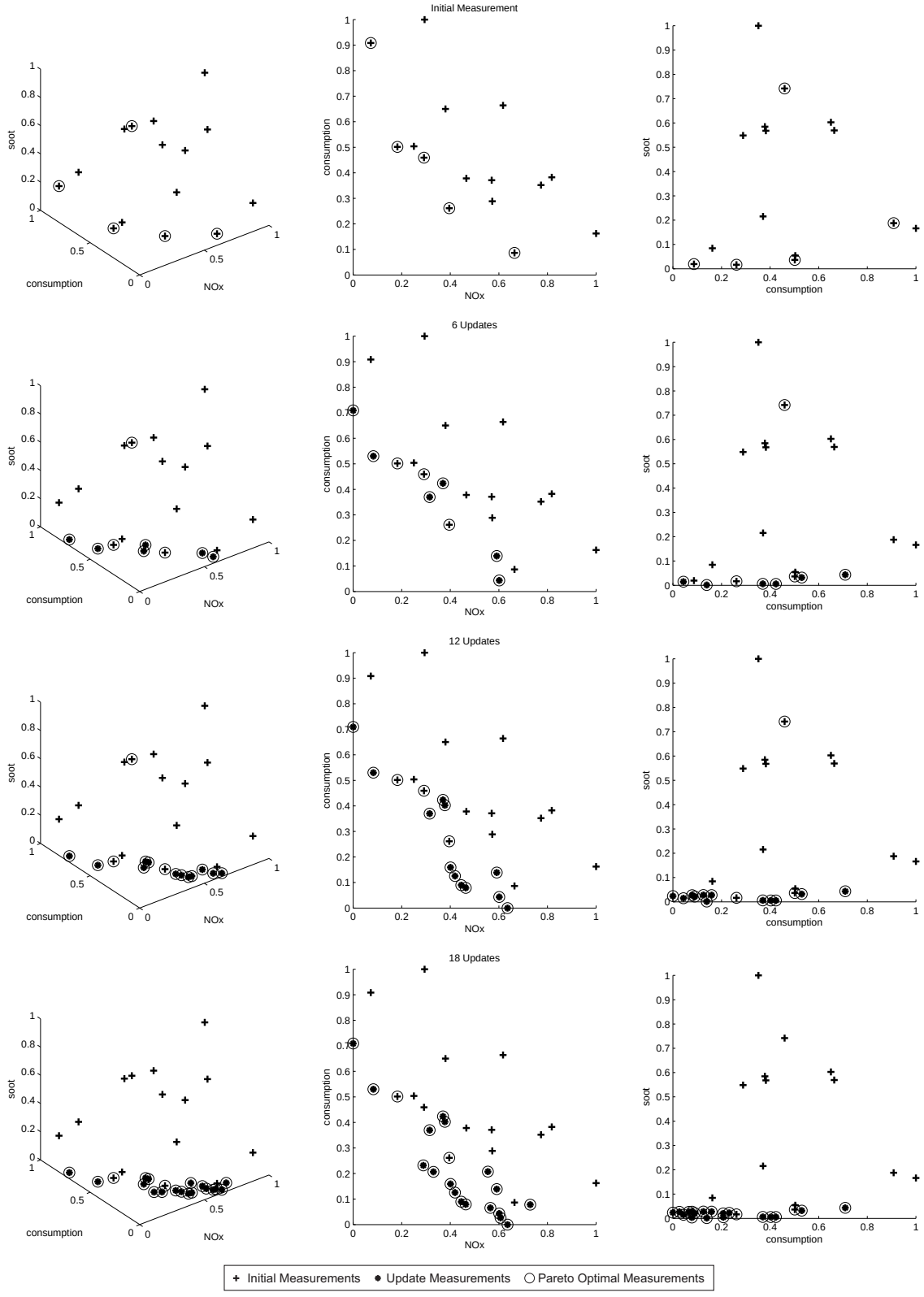


Figure A.5: Practical model-based multi-objective online optimization. Here the consumption, NOx and soot measurements at the operating point $(n_{engine}, M_{engine}) = (1100\ \text{rpm}, 1244\ \text{Nm})$ are plotted in the objective space. At the left a three-dimensional view of the objective space is shown, in the middle and right two-dimensional projections of the objective space are shown.

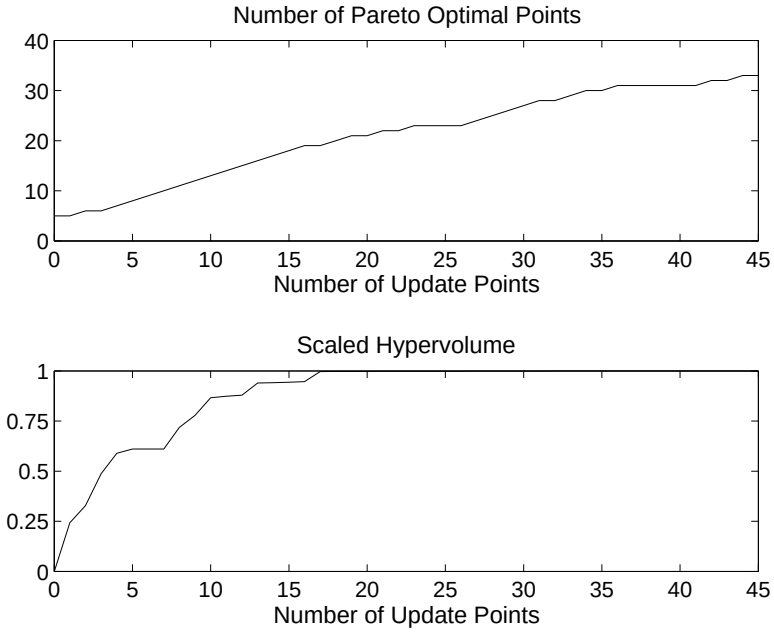


Figure A.6: Number of Pareto optimal update points and the scaled hypervolume for the model-based multi-objective online optimization at the operating point $(n_{engine}, M_{engine}) = (1100\ \text{rpm}, 1244\ \text{Nm})$.

BIBLIOGRAPHY

[1] A&D Company. Automated Calibration with ORION - Presentation at Testing Expo, retrieved from http://www.testing-expo.com/usa/08conf/pdfs/day_1/6_A&D_Don%20Nutter.pdf, Status: May 2012.

[2] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[4] AVL. AVL CAMEO, retrieved from https://www.avl.com/cameo, Status: May 2012.

[5] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[6] H. Bandemer and A. Bellmann. *Statistische Versuchsplanung*. Teubner Verlag, 1994.

[7] B. Berger and F. Rauscher. Evaluation of Gaussian Processes for Black-Box Engine Modelling. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[8] B. Berger and F. Rauscher. Robust Gaussian Process Modelling for Engine Calibration. In *Proceedings of the 7th Vienna International Conference on Mathematical Modelling (MATHMOD)*, 2012.

[9] B. Berger, F. Rauscher, and B. Lohmann. Analysing Gaussian Processes for Stationary Black-Box Combustion Engine Modelling. In *Proceedings of the 18th IFAC World Congress, Milano, Italy*, 2011.

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[12] BOSCH. *Kraftfahrtechnisches Taschenbuch, 25. Auflage*. Vieweg Verlag, 2003.

[13] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society*, 26:211–252, 1964.

[14] L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.

[15] R. H. Byrd, P. Lu, and J. Nocedal. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[16] R. D. Cook and C. J. Nachtsheim. A Comparison of Algorithms for Constructing Exact D-Optimal Designs. *Technometrics*, 22(3):315–324, 1980.

[17] N. Cressie. *Statistics for Spatial Data*. Wiley, 1993.

[18] B. De Finetti. The Bayesian approach to the rejection of outliers. In *Proceedings of the fourth Berkeley Symposium on Mathematical Statistics and Probability*, pages 199–210. University of California Press, 1961.

[19] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[21] M. Deflorian. *Versuchsplanung und Methoden zur Identifikation zeitkontinuierlicher Zustandsraummodelle am Beispiel des Verbrennungsmotors*. Ph.D. thesis TU-München, 2012.

[22] M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. Ph.D. thesis TU-Dortmund, 2005.

[23] M. Emmerich, N. Beume, and B. Naujoks. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, 2005.

[24] A. Emtage, D. Sampson, and K. Röpke. Proposal for a Generic Interface between Test Bench Automation System and DoE Modelling Application. In *Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development*, 2009.

[25] S. Ernst. Hinging hyperplane trees for approximation and identification. In *Proceedings of the 37th IEEE Conference on Decision and Control*, 1998.

[26] L. Fahrmeir, T. Kneib, and S. Lang. *Regression (Modelle, Methoden und Anwendungen)*. Springer, 2009.

[27] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[28] M. Fleischer. The Measure of Pareto Optima. Applications to Multi-Objective Metaheuristics. In *Evolutionary Multi-Criterion Optimization - EMO'2003, Faro, Portugal*, pages 519–533, 2003.

[29] C. M. Fonseca, L. Paquete, and M. López-Ibánez. An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *2006 IEEE Congress on Evolutionary Computation - CEC*, pages 1157–1163, 2006.

[30] F.D. Foresee and M.T. Hagan. Gauss-Newton approximation to Bayesian learning. In *Proceedings of the 1997 International Joint Conference on Neural Networks*, volume 3, pages 1930–1935, 1997.

[31] D. Furch and M. Link. Paoptimizer - ein neues werkzeug zur modellgestützten kennfeldoptimierung von motorsteuergeräten. *MTZ - Motortechnische Zeitschrift*, 12:1047–1050, 2002.

[32] J. Geweke. Bayesian Treatment of the Independent Student-t Linear Model. *Journal of Applied Econometrics*, 8:519–540, 1993.

[33] K. Gschweitl, H. Pluegl, T. Fortuna, and R. Leithgoeb. Steigerung der Effizienz in der modellbasierten Motorenapplikation durch die neue CAMEO Online DoE-Toolbox. *ATZ - Automobiltechnische Zeitschrift*, 103:636–643, 2001.

[34] C. Gühmann and J. M. Riedel. Comparison of Identification Methods for Nonlinear Dynamic Systems. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[35] M. Hafner. *Modellbasierte stationäre und dynamische Optimierung von Verbrennungsmotoren am Motorenprüfstand unter Verwendung neuronaler Netze*. Ph.D. thesis TU-Darmstadt, 2002.

[36] M. Hafner, M. Schler, O. Nelles, and R. Isermann. Fast neural networks for diesel engine control design. *Control Engineering Practice*, 8:1211–1221, 2000.

[37] M. Hafner, M. Weber, and R. Isermann. Model-based control design for IC-engines on dynamometers: The toolbox 'OptiMot'. In *Proceedings of the 15th IFAC World Congress*, 2002.

[38] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[39] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[40] J. Hämmerle. Entwurf von robusten Online-Modellbildungsverfahren für die Motorapplikation. Master's thesis, Technische Universität München, 2011. Supervisor: B. Berger.

[41] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[42] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6(8):1069–1072, 1993.

[43] S. K. Hyde. Likelihood Based Inference on the Box-Cox Family of Transformations: SAS and Matlab Programs. Technical report, Department of Mathematical Sciences, Montana State University, 1999.

[44] IAV GmbH. IAV EasyDoE Toolsuite, retrieved from http://www.iav.com/engineering/methods-tools/produkte/easydoe-toolsuite, Status: March 2012.

[45] R. Isermann. *Modellgestützte Steuerung, Regelung und Diagnose von Verbrennungsmotoren*. Springer Berlin Heidelberg, 2003.

[46] S. Jakubek and C. Hametner. Identification of neurofuzzy models using GTLS parameter estimation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(5):1121–1133, 2009.

[47] S. Jakubek and N. Keuth. Optimierte Neuro-Fuzzy-Modelle für Auslegungsprozesse und Simulation im Automotive-Bereich. *at-Automatisierungstechnik*, 53:425–433, 2005.

[48] S. Jeong and S. Obayashi. Efficient Global Optimization (EGO) for Multi-Objective Problem and Data Mining. In *the 2005 IEEE Congress on Evolutionary Computation*, 2005.

[49] M. E. Johnson and C. J. Nachtsheim. Some Guidelines for Constructing Exact D-Optimal Designs on Convex Design Spaces. *Technometrics*, 25(3):271–277, 1983.

[50] D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[51] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.

[52] N. Keuth, M. Thomas, H. Pflügl, E. Martini, and S. Bergold. Utilization of the Slow Dynamic Slope methology for the calibration of the ECU-functions Air Charge Determination and Torque Prediction in the series production. In *Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development*, 2009.

[53] F. Klöpper. *Entwicklung und Einsatz modellgestützter Online-Methoden zur Parameteroptimierung von Verbrennungsmotoren am Prüfstand*. Ph.D. thesis TU-München, 2008.

[54] K. Knödler. *Methoden der restringierten Online-Optimierung zur Basisapplikation moderner Verbrennungsmotoren*. Ph.D. thesis University of Tübingen, 2004.

[55] K. Knödler, J. Poland, T. Fleischhauer, A. Mitterer, S. Ullmann, and A. Zell. Modellbasierte Online-Optimierung moderner Verbrennungsmotoren Teil 2: Grenzen des fahrbaren Suchraums. *MTZ - Motortechnische Zeitschrift*, 6:520–526, 2003.

[56] J. Knowles. ParEGO: A Hybrid Algorithm With On-Line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2005.

[57] H. Kötter, H. Sequenz, and R. Isermann. Ermittlung der Güte experimentell gewonnener Verbrennungsmotor-Modelle. In *Proceedings of the 2. Internationales Symposium für Entwicklungsmethodik*, 2007.

[58] T. Kruse, S. Kurz, and T. Lang. Modern Statistical Modeling and Evolutionary Optimization Methods for the Broad Use in ECU Calibration. In *Proceedings of the 6th IFAC Symposium "Advances in Automotive Control"*, 2010.

[59] T. Kruse, H. Ulmer, and U. Schulmeister. Einsatz neuer Modellier- und Optimierverfahren zur Applikation von Diesel- und Ottomotoren. In *Proceedings of the 4. Tagung DoE in der Motorenentwicklung*, Berlin, 2007.

[60] L. Kuipers and H. Niederreiter. *Uniform Distribution of Sequences*. Dover Pubn Inc, 2006.

[61] Malte Kuß. *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, TU Darmstadt, April 2006.

[62] S. Kwon, S. Park, C. Kim, and S. Yang. SCR Rapid Heat-Up Calibration with DoE & Virtual Power-Train Development System. In *Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development*, 2009.

[63] S. L. Lauritzen. Time Series Analysis in 1880: A Discussion of Contributions Made by T.N. Thiele. *International Statistical Review*, 49:319–331, 1981.

[64] L. Ljung. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR, December 1999.

[65] D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[66] D. J. C. MacKay. Bayesian methods for backprop networks. *Models of Neural Networks*, 3:211–254, 1994.

[67] D. J. C. MacKay. Introduction to gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–166. Springer, 1998.

[68] D. J. C. MacKay. *Information theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[69] L. Magee. Nonlocal Behavior in Polynomial Regressions. *The American Statistician*, 52:20–22, 1998.

[70] H. Markert, C. Schiepe, F. Streichert, U. Meister, R. Diener, and T. Gutjahr. Comparison of Alternative Approaches to Auto-Regressive Modelling of Dynamic Systems. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[71] E. Martini, H. Voss, S. Töpfer, and R. Isermann. Effiziente Motorapplikation mit lokal linearen neuronalen Netzen. *MTZ - Motortechnische Zeitschrift*, 5:406–413, 2003.

[72] M. D. McKay, W. J. Conover, and R. J. Beckman. 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.

[73] T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[74] T. J. Mitchell. An Algorithm for the Construction of ”D-Optimal” Experimental Designs. *Technometrics*, 16(2):203–210, 1974.

[75] T. J. Mitchell. Computer Construction of ”D-Optimal” First-Order Designs. *Technometrics*, 16(2):211–220, 1974.

[76] A. Mitterer. *Optimierung vielparametriger Systeme in der KFZ Antriebsentwicklung*. Ph.D. thesis TU-München, 2000.

[77] A. Mitterer and F. Zuber-Goos. Modellgestützte Kennfeldoptimierung - Ein neuer Ansatz zur Steigerung der Effizienz in der Steuergeräteapplikation. *ATZ - Automobiltechnische Zeitschrift*, 102:188–194, 2000.

[78] F. Mittermayer. *CFD - Simulation der Zündung und Verbrennung in einem vorkammergezündeten Großgasmotor*. Ph.D. thesis TU-München, 2011.

[79] J.J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. *Lecture notes in mathematics*, 630:105–116, 1977.

[80] R. M. Neal. Probabilistic Inference using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[81] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer. Lecture Notes in Statistics, 1996.

[82] Radford M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of statistics and Department of Computer Science, University of Toronto, 1997.

[83] O. Nelles. LOLIMOT - Lokale, lineare Modelle zur Identifikation nichtlinearer, dynamischer Systeme. *at - Automatisierungstechnik*, 45:163–174, 1997.

[84] O. Nelles. *Nonlinear System Identification*. Springer Verlag, Heidelberg, Germany, 2001.

[85] H. Nickisch and C. E. Rasmussen. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078, 2008.

[86] H. Niedernolte, F. Kloepper, A. Mitterer, and F. Schwarzer. Workflow for data evaluation during basic calibration of combustion engines. In *2006 IEEE International Conference on Control Applications*, pages 2060–2065, 2006.

[87] V. Nissen. *Einführung in Evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution*. Vieweg, Braunschweig, 1997.

[88] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[89] T. Oberndorfer, C. Höfer, C. Doppelbauer, and N. Buch. Intelligent Calibration Tool - A use case. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[90] A. O'Hagan. On outlier rejection phenomena in Bayes inference. *Royal Statistical Society*, Series B:358–367, 1979.

[91] K. Pfeil, R. Zimmerschied, and R. Isermann. Nichtlineare Identifikation des Luftpfads von aufgeladenen Dieselmotoren und automatisierter AGR-/VTG-Reglerentwurf. *at - Automatisierungstechnik*, 55:352–359, 2007.

[92] J. Poland. *Modellgestützte und Evolutionäre Optimierungsverfahren für die Motorentwicklung*. Ph.D. thesis University of Tübingen, 2002.

[93] J. Poland, K. Knödler, T. Fleischhauer, A. Mitterer, S. Ullmann, and A. Zell. Modellbasierte Online-Optimierung moderner Verbrennungsmotoren Teil 1: Aktives Lernen. *MTZ - Motortechnische Zeitschrift*, 5:432–437, 2003.

[94] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted S-Metric Selection. In *Parallel Problem Solving from Nature (PPSN)*, pages 784–794, 2008.

[95] W. Ponweiser, T. Wagner, and M. Vincze. Clustered Multiple Generalized Expected Improvement: A Novel Infill Sampling Criterion for Surrogate Models. In *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, pages 3515–3522, 2008.

[96] C. E. Rasmussen and H. Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

[97] C. E. Rasmussen and J. Quinonero-Candela. Healing the relevance vector machine through augmentation. In *Proceedings of the 22nd international conference on Machine learning*, pages 689–696. ACM Press, 2005.

[98] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[99] I. Rechenberg. *Evolutionsstrategie '94*. frommann-holzboog, Stuttgart, 1994.

[100] M. Reger, R. Diener, V. Imhof, T. Lang, A. Powrosnik, H. Schmidt, and H. Ulmer. Extension of classical DoE method to dynamic conditions by means of characteristic numbers. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[101] K. Röpke. *Design of Experiments (DoE) in Engine Development II*. Expert-Verlag, 2005.

[102] K. Röpke, M. Knaak, A. Nessler, and S. Schaum. Rapid Measurement: Grundbedatung eines Verbrennungsmotors innerhalb eines Tages? *MTZ - Motortechnische Zeitschrift*, 68(4):276–283, 2007.

[103] J. Rückert and M. Bross. Applikationsmethodik bei BMW – nicht nur, aber auch wirtschaftlich ein Gewinn. In *Proceedings of the 3rd International Symposium on Development Methodology*, 2009.

[104] D. Sampson. Use of Model-Based Calibration in the Test Cell. In *Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development*, 2009.

[105] D. Sampson, I. Noell, and L. Sheridan. New challenges in optimization for model-based calibration. In *Proceedings of the 6th Conference Design of Experiments (DoE) in Engine Development*, 2011.

[106] M. J. Sasena, P. Papalambros, and P. Goovaerts. Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. *Engineering Optimization*, 34(3):263–278, 2002.

[107] A. Schreiber and R. Isermann. Durchgängige Methodik zur stationären und dynamischen Vermessung und Modellbildung von Verbrennungsmotoren. In *Proceedings of the 3rd International Symposium on Development Methodology*, 2009.

[108] A. Schwarte, L. Hack, R. Isermann, H.-G. Nitzke, J. Jeschke, and J. Piewek. Automatisierte Applikation von Motorsteuergeräten mit kontinuierlicher Motorvermessung. In *Proceedings of AUTOREG*, 2004.

[109] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[110] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, 1977.

[111] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, New York, 1995.

[112] Matthias Seeger. Bayesian Inference and Optimal Design for the Sparse Linear Model. *Journal of Machine Learning Research*, 9:759–813, 2008.

[113] H. Sequenz, M. Mrosek, and R. Isermann. Stationary Global-Local Emission Models of a CR-Diesel Engine with Adaptive Regressor Selection for Measurements of Airpath and Combustion. In *Proceedings of the 6th IFAC Symposium "Advances in Automotive Control"*, 2010.

[114] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[115] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[116] A. Sung, F. Klöpper, A. Mitterer, and G. Wachtmeister. Modellbasierte Online-Optimierung in der Simulation und am Motorenprüfstand. *MTZ - Motortechnische Zeitschrift*, 68(1):42–48, 2007.

[117] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:116–132, 1985.

[118] The MathWorks, Inc.. Model-Based Calibration Toolbox - Calibrate Complex Powertrain Systems, retrieved from http://www.mathworks.com/products/mbc/, Status: March 2012.

[119] M. E. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[120] M. E. Tipping and N. D. Lawrence. Variational inference for Student-t models: Robust bayesian interpolation and generalised component analysis. *Neurocomputing*, 69:123–141, 2005.

[121] S. Töpfer. Approximation nichtlinearer Prozesse mit Hinging Hyperplane Baummodellen (Approximation of Nonlinear Processes with Hinging Hyperplane Trees). *at-Automatisierungstechnik*, 50:147–154, 2002.

[122] F. Triefenbach. Design of Experiments: The D-Optimal Approach and Its Implementation As a Computer Algorithm. Technical report, Department of Computing Science, Umea University, 2008.

[123] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reding, Massachusetts, 1977.

[124] S. Ullmann. Efficient Test Bed Automation. In *Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development*, 2009.

[125] J. G. van der Corput. Verteilungsfunktionen. *Proc. Ned. Akad. v. Wet.*, 38:813–821, 1935.

[126] J. Vanhatalo, P. Jylänki, and A. Vehtari. Gaussian process regression with Student-t likelihood. In *Advances in Neural Information Processing Systems*, volume 23, 2009.

[127] V. N. Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[128] J. Villemonteix, E. Vazquez, M. Sidorkiewicz, and E. Walter. Global optimization of expensive-to-evaluate functions: an empirical comparison of two sampling criteria. *Journal of Global Optimization*, 43(2):373–389, 2009.

[129] J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534, 2009.

[130] J. Warnatz, U. Maas, and R. W. Dibble. *Verbrennung*. Springer Verlag, third edition, 2001.

[131] M. West. Outlier Models and Prior Distributions in Bayesian Linear Regression. *Journal of the Royal Statistical Society. Series B.*, 46(3):431–439, 1984.

[132] D. H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[133] D. H. Wolpert. The supervised learning no-free-lunch theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 2001.

[134] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[135] C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[136] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[137] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), 2001.

[138] E. Zitzler and L. Thiele. Multiobjective Pptimization Using Evolutionary Algorithms – A Comparative Study. In *Parallel Problem Solving from Nature – PPSN V*, 1998.