Total Vehicle Concept Design using Computational Intelligence

Dipl.-Ing. Maximilian Helbing
Chair of Vehicle Mechatronics
Institute of Automobile Engineering
Technische Universität Dresden
George-Bähr-Str 1c
01069 Dresden, Germany
Phone: +49-351-463-39566
Email: maximilian.helbing@tu-dresden.de

Prof. Dr.-Ing. Bernard Bäker
Head of the Chair of Vehicle Mechatronics
Institute of Automobile Engineering
Technische Universität Dresden
George-Bähr-Str 1c
01069 Dresden, Germany
Phone: +49-351-463-34832

Dipl.-Ing. Stephan Schiffer
BMW Group
Total Vehicle
Electrified Vehicles, Concepts
Knorrstraße 147
80788 München, Germany
Phone: +49-89-382-17611
Email: stephan.st.schiffer@bmw.de

***Abstract*—This paper evaluates five methods from the field of computational intelligence and machine learning with regard to their suitability for total vehicle concept design. These methods are used to mimic the input-output relation of a complex model, based on a limited number of expensive simulations (metamodeling). Important steps of this model building process will be investigated and possibilities for model evaluation will be discussed. As the theoretical background is provided, Artificial Neural Network (ANNs), Multivariate Adaptive Regression Splines (MARS), Gaussian Process Models (GPMs), Regression Trees (RTREES) and Support Vector Machines (SVMs) are implemented and compared regarding their accuracy and speed of modeling and computation. A setting from an early stage of total vehicle concept design serves as use case.**
It can be summarized that the implemented metamodels, except RTREES and SVMs, achieve good approximation accuracy. The ANN proved to be superior method compared to MARS and GPMs. However, the training of the ANN took the most time, which may be of relevance if the modeling process has to be carried out very frequently. Then GPM or MARS offer a good compromise between training duration and accuracy.

I. Introduction

The vehicle concept design is characterized by an iterative procedure, in which different parties, e.g. the development parties at an OEM, gradually approach the best possible solution. All relevant technical variables that affect the vehicle concept and subsequent vehicle characteristics must be considered. At the same time, concept engineers are confronted with an increasing variety of variants and system complexity with simultaneously ever shorter development cycles. It is a great challenge to efficiently evaluate the potential of a concept idea, espacially in an early development phase, when there is usually little valid information on individual components [1], [2], [3]. Decisions made at this early stage of development have a profound impact on later development processes. Subsequent changes in concepts are usually associated with high technical and financial risks. Therefore, methods of concept development and evaluation are of particular importance and represent a strategic success factor [4].

Computer simulations have long established themselves as an integral part of the tool chain of a concept developer. According to Siebertz [5] simulations are thereby not used as simple virtual testing, but for structured parameter variation, to create knowledge from numerous simulations. Although computing performance is increasing steadily, a sensible use of simulations and the efficient utilization of the available computing ressources is a major aspect. In parallel to the computational performance, the model complexity and the number of model parameters increase in order to improve the model quality [5], [6], [7]. Given this, complex model analyses can quickly become very time-intensive if it comes to multiple-objective optimization problems and / or given a large design spaces. To lower the computational burden *approximations* of the complex models can be used. This approximations describe only significant effects of the complex model by replacing computer-intensive functions with fast-calculable analytical functions [7]. The creation of this approximation is performed using data obtained with the complex model. This approximations are called *metamodels* (or surrogate, approximation models) and have been successfully used in the field of engineering in order to supplement very high-intensity finite elements or elaborate aerodynamic simulations.

Although Siebertz [5] sees a clear trend toward metamodelling, and Simpson et. al [7] describes metamodels as a valuable tool that allows rapid analysis within a large design space, with only minor disadvantages in accuracy and according to Wang et al. [6] they can support a wide range of activities in engineering design, there are relatively few applications in vehicle concept design. Here, Eghtessad [2] and Moses et al. [1] are to be mentioned. They used metamodels to improve the overall computation efficiency in order to find the optimal drivetrain respectively vehicle concept design [1], [2]. Although, many techniques for metamodeling have been developed in various disciplines, such as statistics, mathematics, computer science, their application in the automotive development process is marginal. As Wang already implies [6], rapid development driven by many disciplines impedes keeping the overview and selecting the best method for application.

This paper explores the methodology of metamodels for

the concept design of vehicles in an early development phase. We will discuss the necessary steps for the development of metamodels and give specific examples of both the methods and the evaluation techniques. Here the trade-off between accuracy and time efficiency is specifically addressed as the effort in modeling and not only the model use is taken into account. The analysis, which this paper is based upon, uses well established software and toolboxes for the realization of the different methods. Thus, the mathematical background is not covered in detail, but rather referenced in the extensive literature already existing for these methods.

Following this assumption, the contribution is structured as follows. Section II gives a brief introduction to the concept of metamodeling and its basics steps. Section III covers the sampling methods in order to generate data for metamodeling and thereby describes the requirements of deterministic computer simulations as well as the evaluation of the generated design of experiment. Once the basis for the data generation is created, IV presents selected methods, that are able to approximate complex system behaviour given a limited data set. Based on these results, section V presents a use case in which the sampling method of section III and the modeling methods of IV are implemented and evaluated.

II. Brief Introduction to Metamodels and their Potential for the Vehicle Concept Design

This section describes the methodology of metamodels, how they are created, and what they can be used for. But first, the term is to be clarified.

Metamodels are used as a surrogate of a complex model [7].

A **Metamodel** is a simplified model of a computation-intensive computer model (reference model). It approximates selected reference model outputs as a simple analytical function of the input variables, describes them independent of the reference model and provides a (much) faster calculation [5], [6], [7], [8].

So, the basic idea is to provide an approximation for the relationship between system variables (inputs) $\mathbf{x}$ and system response (outputs) $\mathbf{y}$ of the reference model (see Eq. (1)).

$$\mathbf{y} = f(\mathbf{x}) \tag{1}$$

Since the original model relation $f(\mathbf{x})$ is usually unknown the approximation can be generally described as [7]:

$$\hat{\mathbf{y}} = g(\mathbf{x}) \tag{2}$$

$$\mathbf{y} = \hat{\mathbf{y}} + \epsilon, \tag{3}$$

where $g(\mathbf{x})$ representing the approximation function and $\hat{\mathbf{y}}$ is the approximated system response, that differs by the error ϵ (approximation and/or measurement error) from the original responses $\mathbf{y}$.

The process of generating a metamodel i.e. the generation of $g(\mathbf{x})$ is called metamodeling, which is structured in figure 1. In order to build the input-output approximation $g(\mathbf{x})$, data of the reference model is required. Since it is

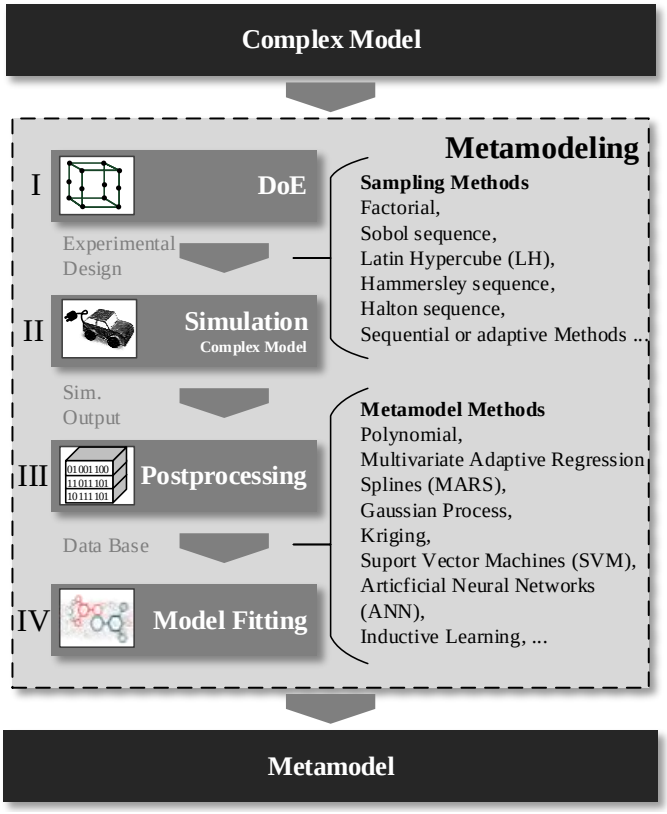


Figure 1: Metamodeling

based on data, Simpson et al. [7] refer to it as a *statistical approximation*. This is the origin of metamodeling, as it has developed from the classical *Design of Experiments* (DoE) theory, in which - mostly low order - polynomials were used as metamodels [7], [6]. To obtain data, simulations[1] have to be performed. As mentioned befor these simulations can be very time consuming. Therefore, the simulation of the reference model is preceded by the generation of an experimental plan. Several DoE methods exists, that shall provide an experimental plan ensuring both a time-efficient data generation as well as enough information to create an accurate metamodel. After the experimental design is created and the necessary simulations were performed, the generated simulation results have to be prepared to generate a data set suitable for metamodeling (postprocessing). Subsequently, using this data base, the desired metamodel can finally be build, which is refered to as training or model fitting. The most common method for this input-output approximation is the use a polynomal regression as metamodel and to perform the modelfit (find suitable polynom parameters) with the method of least squares.

The process presented in Fig. 1 is carried out in order to increase the computational efficiency. That in turn enables a variety of analysis in a short amount of time, which can be

[1]Due to the connection to the DoE, the use of 'computer experiment' is common and will be used in this paper as synonym for simulation.

grouped into three categories:

1) graphical representation,
2) design space exploration and
3) optimization.

Assume an accurate metamodel is available, then the relation between model in- and outputs can be quickly visualized, which makes it an efficient tool to explore large design spaces. Both aspects can help to provide a deeper understanding of the model and its related system. When it comes to multiple-objective optimization problems or large design spaces, a metamodel together with optimization methods can be utilized to effciently find the optimal variable settings [1], [2], [5], [7]. An introduction to the metamodel-based-design-optimization (MBDO) and the different MBDO strategies can be found in [6].

All three basic application areas are of relevance for the use in vehicle concept design. This can examplarily be illustrated for the powertrain concept design of electrified vehicles. Feasible technical variables of the powertrain concept design are for example the selection, scaling and allocation of powertrain components (e.g. electric machine(s), battery system, transmission etc.), which have been comprehensively studied [1], [2], [9], [10], [11], [12], [13], [14], [15], [16]. The studies mainly differ concerning their applied simulation tools and evaluation criteria. Eghtessad [2] and Moses [1] optimize powertrains of Battery Electric vehicles (BEV)s and assess possible solutions regarding the criteria costs, energy efficiency and driving performance. In addition to those criteria Fuchs [10] considers interactions between installation space and powertrain component allocation. An even broader evaluation framework is used by Reupold [11]. Besides the aforementioned criteria applied in [10], [1] and [2] acoustics, complexity and safety are additionally included within the powertrain concept design of a BEV [11]. Similar criteria are incorporated finding the best HEVs powertrain concept [13], [14], [12].

This results in numerous interactions which makes it difficult for developers to efficiently evaluate the full potential of a concept. Owing to the large number of parameters and interactions and due to the computational burden of the used models, the problem formulation must necessarily be simplified by introducing constraints and assumptions. Eghtessad [2] and Moses [1] realized the potential of metamodeling and used it to substitute the computationally complex consumption simulation and have successfully used it within an optimization process. Metamodels, however, have the potential to process many input parameters very quickly which makes it very interesting for use in vehicle development. This leads to two questions: first, how can the data be generated in order to build a metamodel, and second, how can a metamodel be generated using that data? Following the aim of this paper we will focus on the practical application of the metamodeling, giving a brief introduction to the sampling methods (sec. III) and meta modelling methods (sec. IV).

III. Experimental design for computer experiments

A. Brief overview of space-filling sampling methods for computer experiments

As described in sec. II the prerequisite for the creation and usage of metamodels is the existence of information, i.e. data. For technical modeling, a distinction is made between two types of modeling, the theoretical and the experimental modeling [17], [18]. The metamodeling is clearly assigned to the latter category, with the peculiarity that the data is not derived from experiments on real components or systems (physical experiments), but from computer experiments (simulations). Regardless of whether physical or computer experiments are to be performed, an experimental design is essential.

Experimental Design The sequence of experiments to be performed is called experimental design. It consists of a set of factors (design variables = inputs) and associated values (input settings). The experimental design can be represented as a $\mathrm{n_s} \times \mathrm{n_f}$ matrix $\mathbf{X}$, where each row is one experiment run (sample) and each column is one particular factor setting [7].

$$\mathbf{X} = \begin{array}{c} \begin{matrix} \text{factor 1} & \text{factor 2} & \text{factor 3} & \dots & \text{factor } \mathrm{n_f} \end{matrix} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,\mathrm{nf}} \\ x_{2,1} & & & & \\ x_{3,1} & & & & \\ \vdots & & & \ddots & \vdots \\ x_{ns,1} & \dots & & & x_{\mathrm{ns,nf}} \end{bmatrix} \end{array} \begin{array}{l} \text{sample 1} \\ \text{sample 2} \\ \text{sample 3} \\ \vdots \\ \text{sample } \mathrm{n_s} \end{array}$$

Given the number of factors n_f, the goal is to create an efficient set of computer runs $(\mathbf{x_1},\mathbf{x_2},...,\mathbf{x}_{\mathrm{ns}})$ (samples) [5], [7]. An ideal experimental design for metamdeling provides a maximum ammount of information while ensuring a minimal number of expensive simulations[2]. The classic DoE methods were developed for physical experiments and tend to concentrate samples at the edges as well as at the center of the experimental space [6]. However, some methods of experimental design are not applicable to computer experiments. Instead, the special characteristics of computer experiments must be considered in order to generate an optimal design. The main differences are that computer experiments, unlike physical experiments, are deterministic and therefore not subject to random errors[3] and factor settings can easily be adjusted. As a consequence classical methods, e.g. *D-optimality designs*, *central composite design* and in fact all methods which include clasical notions of experimental blocking, replication and randomization are inappropriate [5], [6], [7]. Instead, there is consensus in the literature that experimental designs for deterministic computer experiments in order to generate metamodels without prior knowledge about the model behavoiur, should be *space filling* [2], [5], [6], [7], [8], [19], [20], [21].

[2]In case of physical experiments the minimum number of experiments not necessarily correlates with the minimum effort in execution of the experiment

[3]Explanations and additional literature concerning *numerical noise* can be found in [6], [7].

Hence, this section focuses on the creation of space-filling experimental designs in order to generate a proper data set as prerequisite for metamodeling. From the sources mentioned above, the following sampling methods were selected for investigation.

- Full factorial design
- Latin hypercube design (LHD)
- SOBOL sequence
- HALTON sequence
- HAMMERSLEY sequence

The **full factorial design** is propably the most basic experimental design method in which the number of samples is the product of the number of settings for each factor [7]. Although the full factorial design does not belong to the space-filling design methods, it is to be considered here as it is an easy-to-understand type of experimental design, which is still used in industry. For detailed information and further variants of the factorial design, such as *fractional factorial design*, can be obtained, for example, in [5], [7]. Because of the exponentially increasing number of experiments with increasing number of factors (n_f) and the problem of *pseudo-repetitions*, more efficient methods have been established.

Due to their speed and ease of implementation **Latin hypercube designs** (LHDs) have become a very popular experimental design technique. A LHD constructs samples by dividing each dimension (n_f) in m equal sections. Each of these sections is assigned exactly one value for each dimension, which guarantees that each sample is at least $\frac{2}{m}\sqrt{2}$ away from the nearest neighbor, assuming a $[-1, 1]^{n_f}$ design space [8]. In contrast to the classic *Monte Carlo Method*, where the sample points are randomly choosen from the entire design space, this provides more space uniform design. Although, LHDs still provide no deterministic but a random design, many researchers are working on finding an optimal LHDs, e.g. [19], [22] and further literatur is listed in [6], [7].

SOBOL, **HALTON** and **HAMMERSLEY** sequence belong to a more general category called low discrepancy sequences or low discrepancy procedures [5], [6], [8]. All low discrepancy procedures are also known as quasi-random or quasi-Monte Carlo methods, as they generate a experimental design according to specific algorithms. This results in an experimental design that at first glance may seem to be randomly determined (in the case of a sufficiently high number of sample points), but is in fact deterministically computed. The computations of a HALTON, HAMMERSLEY and SOBOL sequence are based on prime numbers to generate 'quasi' random samples. Deeper insight on this methods as well as their algorithms can be found e.g. in [5].

In order to find a proper experimental design, suitable as prerequiste for the metamodeling, the methods are implemented in MATLAB. The implementation of fullfactorial design, LHD, SOBOL- as well as the HALTON design is done using the MATLAB Statistics and Machine Learning Toolbox. An implementation of the HAMMERSLEY sequence is available in [23] wich is licenced under GNU LGPL. The resulting experimental designs for two of $n_f = 5$ factors and $n_s = 1024$ sample points are visualized in Fig. 2. To choose a suitable design, different criteria exist of which two will be addressed in the next section.

B. Evaluation of space-filling sampling methods for computer experiments

Concerning the evaluation of space-filling sampling methods, two aspects play an essential role, the number of experiments and the knowledge to be derived from them.

According to their number of samples (n_s) experimental designs are categorized in *unsaturated*, *saturated* and *supersaturated* designs [7]. Most designs are unsaturated, as they have at least two more samples n_s as the number of factors n_f. If the design contains exactly one more sample point than factors ($n_s = n_f + 1$), it is called saturated, and if $n_s \leq n_f + 1$ the experimental design is supersaturated. In the use case presented in Sec. V a unsaturated design is choosen, as it is recommended in literature as long as the experiments/simulations are not excessively elaborate [5], [6], [7], [20]. Preliminary n_s was set to $n_s = 1024$. With this setting, a sufficient amount of data could be generated, both to ensure adequate model accuracy and to enable a sufficiently extensive validation while limiting the simulation effort for the given reference model (see sec. IV and sec. V).

The information content that can be obtained with a certain experimental design is linked to the space-filling criterion. This assumes that no information on the model behavior exists before the experiment is carried out.

Under this assumption, it is expected that the highest information content is provided by equally spreading out the sample points around the entire design space (space-filling). Consequently, the evaluation of the maximum amount of information turns into evaluating the space-filling criterion. An overview of and discussion on different space-filling criteria is given for example in [5], [8].

In order to compare the design methods (see III-A) and finally select one, two measurands were choosen – the AUDZE-EGLAIS (AE) and *centered discrepancy* (CD). Both measurands enable an assessment of the space-filling criterion of experimental designs.

The discrepancy can be described as a kind of "density ratio". In this case, the space covered by the entire test field and the number of samples located therein are set in relation to a subspace of the test field and the samples remaining therein [2], [5]. A distinction is made between different types of discrepancy values, depending on the selection of the design spaces to be set in relation to one another. For more information and different definitions for discrepancy, please refer to [5], [24]. The formula for centered discrepancy (CD)

used in this paper is defined as follows [24]:

$$[CD(\mathbf{X})]^2 = \left(\frac{13}{12}\right)^{n_f} - \frac{2}{n_i}\sum_{i=1}^{n_i}\prod_{j=1}^{n_f}\left[1+\frac{1}{2}|x_{ij}-0.5| - \frac{1}{2}|x_{ij}-0.5|^2\right] + \frac{1}{n_i^2}\sum_{i=1}^{n_i}\sum_{k=1}^{n_i}\prod_{j=1}^{n_f}\left[1+\frac{1}{2}|x_{ij}-0.5| + \frac{1}{2}|x_{kj}-0.5| - \frac{1}{2}|x_{ij}-x_{kj}|\right] \quad (4)$$

where n_s is the number of samples (here $n_s = 1024$), n_f is the number of factors (here $n_f = 5$) and x_{ij} is the setting for factor j at simulation run number i. A low value for CD indicates a good space-filling design.

The AUDZE-EGLAIS-criterion can be explained with physics as it is related to the potential energy among the design points [20]. The potential Energy E^{AE} is proportional to the inverse of the squared distance between the sample points, so that:

$$E^{AE} = \sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{1}{d(x_i,x_j)^2}, \quad (5)$$

where $d(x_i, x_j)$ is the EUCLIDEAN distance between point $\mathbf{x}_i$ and $\mathbf{x}_j$, defined as:

$$d(x_i,x_j) = \sqrt{\sum_{l=1}^{n_f}(x_{il}-x_{jl})^2} \quad (6)$$

Again, a good space-filling property is indicated by a small value for E^{AE}. Quite similiar to the AUDZE-EGLAIS-criterion EUCLIDEAN *maximin* which is another very common space-filling criterion. It states that the minimal EUCLIDEAN distance $\min_{i\neq j} d(x_i, x_j)$ between two sample points should be maximized for a optimal space-filling design [2], [5], [6], [7], [8], [20]. In fact the implementation of the LHDs in the MATLAB Statistics and Machine Learning Toolbox uses the maximin criterion by default to generate an "optimal" LHD. So the best LHD according to the maximin-criterion out of p iterations (default: $p = 5$) is selected.

The comparison of the generated experimental designs (see Fig. 2) according to the centered discrepancy (see Eq. (4)) and the AUDZE-EGLAIS-criterion (see Eq. (5)) is presented in Fig. 3. As shown in the bar plots, the fullfactorial design is clearly outperformed by the low-discrepancy procedures. Although all low-discrepancy procedures show similarly good space-filling characteristics the SOBOL design is selected for further use in the context of this paper, as it shows the best result in CD and the second best for E^{AE}.

It should be noted that although the SOBOL design shows good space-filling characteristics, it does not make it to the best design. One reason is the number of samples which is defined before the metamodel is build. This so called *one-stage-sampling* may generate good space-filling designs but still contains too much (or too few) samples for the purpose of accurate metamodeling. Common research therefore aims to find sequential design methods that generate samples adaptivaly without defining the number of samples a priori [6], [8], [20]. In order to optimize the design of experiment many other evaluation criteria, like orthogonality-based criteria exist. A compact overview and comparison of different criteria can be found in [20].

The DoE continues to have a great importance in research, as the generation of data through simulation consumes the majority of time in the model building process With this short survey of DoE for computer experiments, a suitable space-filling design was selected in order to efficiently create an accurate metamodel. The following section is intended to provide a brief overview of metamodeling methods and gives an introduction to quality measures for model validation.

IV. METHODS FOR METAMODELING

After discussing in the previous sections how metamodels are created and the data required for the modeling process can efficiently be generated, this section adresses the question which methods are suitable for building a metamodel.

A. Requirements and how to meet them intelligently

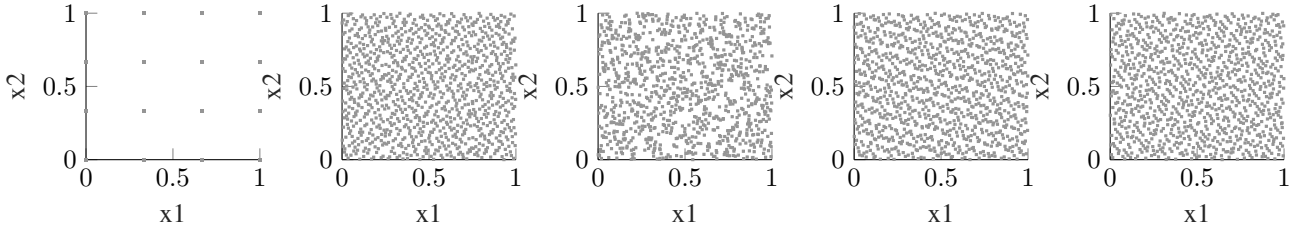
The following requirements on the method and the model obtained therefrom are provided [2], [5]:

1) low computational burden,
2) high model accuracy,
3) automatic, adaptive and flexible modeling

The use of metamodels is only justified if the computational burden can be reduced 1) while ensuring a satisfactorily accurate approximation of the reference model responses 2). In fact this is the main driver for metamodeling as allready mentioned in section II. Furthermore, the method should be able to automatically analyse the provided data and flexibily adapt to complex system behaviour of the reference model 3). This requirement is important because no knowledge of the relationship between (many) input factors and the outputs to be considered is known before the model is built [5]. Methods which presuppose a setting of the input-output behavior entail the risk that, in the case of an incorrect a priori definition, the response can not be computed exactly.

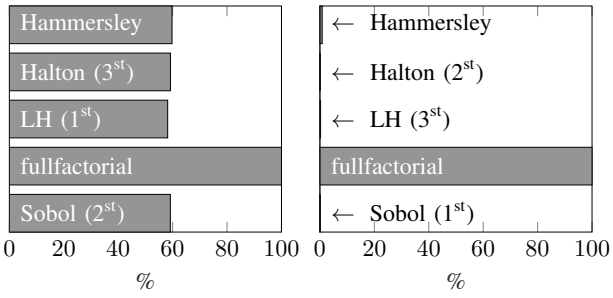
If, for example, a linear behavior is predefined for a simple polynomial regression but the system shows a quadratic relationship, this leads to poor accuracy.

However, the goal of a metamodel is to mimic the reference model by *learning* the mapping from inputs $\mathbf{x}$ to outputs $\mathbf{y}$, based on a limited number of expensive simulations [8]. In case of discrete outputs $\mathbf{y}$ this task is called *classification* and *regression* if $\mathbf{y}$ is continous [25]. To solve this problems many methods from different disciplines and fields of research exist. In addition to the classic methods like linear or polynomial regression, spline regression, interpolation techniques, new, *intelligent* methods are rapidly expanding the portfolio. The terms *learning* and *intelligence* open up a broad spectrum of information. As Kroll [26] states, *Computational Intelligence* is understood to be the scientific domain, which develops



(a) Full factorial design (b) SOBOL design (c) LHD (d) HALTON design (e) HAMMERSLEY design

Figure 2: Sample designs (fullfactorial and four common low-discrepancy designs) with 1024 samples for two factors (x_1,x_2) in a $[0, 1]^5$ design space



(a) Audze-Eglais (b) Centered discrepancy

Figure 3: Design evaluation subject to space-filling criterion using Audze-Eglais and centered discrepancy, both normalized and higlighting the top 3

algorithmes for solving complex problems based on data and inspired by nature. These algorithms mainly include the fuzzy systems, evolutionary algorithms and artificial neural networks (ANN) [26]. The latter has a wide range of application possibilities, including pattern recognition / clustering, and modeling. Although the concept of an intelligent system, an intelligent method or technical intellegince is not precisely defined [26], [27], the ability to learn is always an important characteristic. This in turn leads to the concept of machine learning, which, according to Murphy [25], can be defined as *"[...] a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty"*.

It is not the goal nor is the claim of this paper to examine these domains in greater detail.Therefore, please refer to [25], [27], [28], [29]. However, a common incentive is to find methods for the efficient processing and representation of knowledge [28]. Hence, the following frequently used methods are selected for further investigation:

1) Artificial Neural Network (ANN)
2) Multivariate Adaptive Regression Splines (MARS)
3) Regression Tree (RTREE)
4) Gaussian Process Model (GPM)
5) Support Vector Machine (SVM)

B. Methods

An **Artificial Neural Network (ANN)** is an information processing algorithm whose structure and function is inspired by the nervous system, especially the brain of animals and humans [5], [26], [28]. They consist of nonlinear elements, so called neurons, which exchange information in a directed graph. For this purpose, the networks are trained by sample data $(\mathbf{x}, \mathbf{y})$. During these training processes (model fitting) the connections between the neurons, i.e. the regression coefficients (weights) are adjusted. For this purpose usually a back-propagation algorithm is used. A main issue while working with ANN is the definition of the networks architecture, which means the assembling of the neurons [7]. The most familiar form is the feedforward multilayer perceptrons (MLPs) network. Herein neurons arranged in parallel form a layer and each layer is fully connected to the next layer. The number of neurons of the first and last layer is defined by the number of input (n_f) and output variables. The layers between them are called hidden layers, and the number of neurons within the hidden layers has to be predefined. Since the ideal choice of architecture is often difficult from the outset, Siebertz [5] recommends modeling of various networks and then selecting the one with best accuracy. A guidline for the determination of the number neurons $n_{N_{1HL}}$ of a network with one hidden layer is [5]:

$$n_{N_{1HL}} = \sqrt{n_f \cdot n_y}, \tag{7}$$

where n_f is the number of inputs and n_y is the number of outputs. In a MLP network one hidden layer is sufficiant for most applications, as it is considered as a 'universal function approximator' [5], [30].

Due to their high flexibility and capability of handling complex and high dimensionality data, ANN are suited for a wide range of applications, like pattern recognition (speech, handwriting, image), clustering and modeling of nonlinear. For further information please refere to [5], [7], [28]

Multivariate Adaptive Regression Splines (MARS) is a flexible, non-parametric regression technique that recursively devides the design space and uses splines for piecewise approximation of the input-output relation. MARS automaticaly models interaction between variables, produces models with continous derivations and can handle high-dimensional data.

In order to limit the number of parameters to fit for high-dimensional data, linear splines are prefered [7], [25]. The method was introduced by Friedman in 1991 and is intended here as a representative of the 'classical' regression method [7], [25], [31]. At this point, another method, the so-called response surface models (RSM), shall be mentioned. RSM which are originally developed from DoE for statistical modeling are probably the most researched and most commonly used technique for metamodeling [6]. Both MARS and RMS use polynomials (mostly low order) for approximation, but in contrast to MARS, RMS provide an global instead of a piecewise approximation $g(\mathbf{x})$. As a consequence RMS are easy to consruct but provide limited accuracy for complex data [6].

Regression Tree (RTREE) are closely related to MARS, but instead of piecewise-linear spline approximations, they use a piecewise constant approximation [7]. That leads to a sequence of condition-action or if-else rules, which can be represented by a tree. Because of this, RTREEs are easy to interpret and are able to handle discrete and continuous data as well as combination of both. Their scalability to large data and the automatic variable selection makes RTREEs a popular method for classification and regression applications. The mathematics of this method are given by [25].

GAUSSIAN Process Model (GPM) have recently attracted much attention as a method for metamodeling [6]. In a GPM the input-output relation is approximated with a global polynomial and a normally distributed GAUSSIAN random process [7]. Depending on its implementation (selection of the correlation function), the GPM is able to interpolate the given data set exactly or to filter noisy data [6]. As in case of metamodeling, the data is not subject to random erros or noise the GPM is a noise-free regression model. [5], [6], [7]

Another popular method are **Support Vector Machines (SVMs)** which have proven as robust classifiers able to handle high-dimensional data while automatically fitting to the given data set [32]. In addition to their classification ability, SVMs can be applied for high dimensional regression (Support Vector Regression - SVR). While they are able to provide a similiar prediction quality than ANNs, according to Murphy [25] the training process is less time-consuming since they use a convex objective function. For an introduction to mathematics of SVMs please see [25]. A comprehensive work on convex optimization is given by [33] and if you prefer a more compact introduction you may see [34].

Each method has its associated fitting/training method. In classic polynomial functions usually the method of least squares is used. Fitting an ANN is usually done by back-propagation algorithms. The automatic, adaptive and flexible modeling is strongly connected to this fitting methods. However, presenting the mathematics for the associated fitting methods as well as equations for the listed methods is far beyond the scope of this paper. In addition, the given list of methods merely represent a selection of a large number of methods and does not claim to be complete. For a survey of metamodeling methods please see [6], [7]. A comprehensive discussion on methods from the field of machine learning is provided by [25]. An introduction to computational intelligence is given by [26] or [28] and a more comprehensive literature is [29].

Since it is not possible to determine from the outset which is the most suitable method, it is necessary to examine different methods for the associated application and finally select the appropriate one – which is sometimes refered to as the *no free lunch theorem* [2], [5], [7], [25]. For this selection, a model validation with correspondingly objective evaluation criteria is necessary. Some of these validation methods are presented in the next section.

C. Model Validation

Due to the high flexibility and adaptability of many methods for the metamodel creation, a high model quality even for complex systems is possible [5]. Each of the methods presented in Sec. IV-B is capable of fulfilling the requirements explained in IV-A and therefore suitable for metamodeling. This section is intended to present common quality criteria for the model evaluation, and in doing so, address some of the special features of the validation of metamodels. One of these special features is the problem of creating a metamodel which is excellently fitted to the given data points, but has a low prediction quality for new variable combinations. The reasons for this can be incorrectly chosen variables, the misinterpretation of the system behaviour based on the available data, overfitting to the existing data, and not least the improper use of meta modeling methods (e.g. specification of the ANN architecture) [5], [7].

Therefore, proper model validation is essential. The neglect of adequate validation is, according to Simpson, one of the most common mistakes [5]. Ususal quality measures such as least square residuals $(y_i - \hat{y}_i)^2$, which are typically used in case of simple polynomial regression models, are not sufficient for complex metamodels based on PC experiments. For example, a Kriging or Gaussian Process model always[4] contains exactly the data points that were used for model creation, which is why the assessment by means of residuals at these points is not useful. In the literature, it is agreed that the validation of the models is to be carried out with a data set which was not part of the data set used for model creation [1], [5], [6], [7]. Given a test data set $S_2\{X_{\text{test}}, Y_{\text{test}}\}$ consisting of n_{test} input-output data pairs $(\mathbf{x}_{\text{test}}; \mathbf{y}_{\text{test}})$ which were not subset of the training data, various quality measures for metamodel validation exist. Four frequently used measurands are the root mean square error (RMSE), the maximum absolute error (MAX), the maximum error (MAX_{rel}) and the R square value (R^2) [1], [2], [5], [6], [7].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{\text{test}}} \left(y_{\text{test},i} - \hat{y}_{\text{test},i}\right)^2}{n_{\text{test}}}} \tag{8}$$

While the average accuracy of the global model can be assessed using RMSE (see Eq. (8)), MAX (see Eq. (9))

[4]unless an inexact interpolation is wanted, which may be very rare in case of deterministic computer experimental data

provides a measure for the local accuracy [6], which is why both are often used in combination [1], [2]. Low values for RMSE and MAX indicate good model acuracy. Naturally MAX is quite similiar to $\mathrm{MAX_{rel}}$ (see (10)) except that the latter provides a unitless measure, which sometimes facilitates the interpretation.

$$MAX \quad = \max |y_{\mathrm{test},i} - \hat{y}_{\mathrm{test},i}| \tag{9}$$

$$MAX_{rel} = \max \left| \frac{y_{\mathrm{test},i} - \hat{y}_{\mathrm{test},i}}{y_{\mathrm{test},i}} \right| \tag{10}$$

Additionally, the R square value can be used. The closer R square is to 1, the more accurate the model is, where $\bar{y}_{\mathrm{test}}$ is the mean of test data outputs $\mathbf{y}_{\mathrm{test}}$. As Wang et al. [6] point out, Eq. (11) here is computed for the additional test data, which is different from the traditional use of R square.

$$R^2 = 1 - \frac{\sum_{i=1}^{n_{\mathrm{test}}} (y_{\mathrm{test},i} - \hat{y}_{\mathrm{test},i})^2}{\sum_{i=1}^{n_{\mathrm{test}}} (y_{\mathrm{test},i} - \bar{y}_{\mathrm{test}})^2} \tag{11}$$

Another common method to ensure good model quality is *cross validation*, beeing mainly classified into three different types - *repeated random sub-sampling validation*, *k-fold cross-validation*, *leave-p-out cross-validation* [5]. However, there seems to be no clear consensus in the literature about the usage of cross-validation for assessing metamodel quality. Although Siebertz [5] describes it as proven criterion and both Eghtessad [2] and Krause et al. [35] use it, Wang et al. [6] consider cross validation rather inappropriate. The research underlying this paper indicates, that cross-validation can be used when only a small data set is available and generating additional test data is very costly. Since this is not the case the current application (see Sec. V), the data set is split into two subsets. One subset $S_1\{X_{\mathrm{train}}, Y_{\mathrm{train}}\}$ is used exclusively for model training (training set) and the second subset $S_2\{X_{\mathrm{test}}, Y_{\mathrm{test}}\}$ is for validation of the fitted model (validation set) (see Fig. 4). Thus, the training set for model creation and the validation set for model evaluation are identical for every method discussed in this paper in order to evaluate their performance and enhance comparability [25]. Further commonly used measurands are the mean squared error (MSE), the normalized mean squared error (MSNE) or the number of metamodel outputs $y_t\hat{e}st$ with a higher relativ error than 5 % [1], [2]. A more comprehensive discussion on model validation can be found for example in [5], [6], [7], [36]. With regard to their extrapolation behavior, all methods tend to significantly lose accuracy, as soon as the area enclosed by the training points is left. Although the extrapolation error depends on the model used and the behavior of the real system, it is not recommended to use metamodels for extrapolation [5].

Finally, it should be noted that from our experience all validation methods refer to the data obtained from the reference model. The validation of the metamodel is therefore always subject to the quality of its reference.

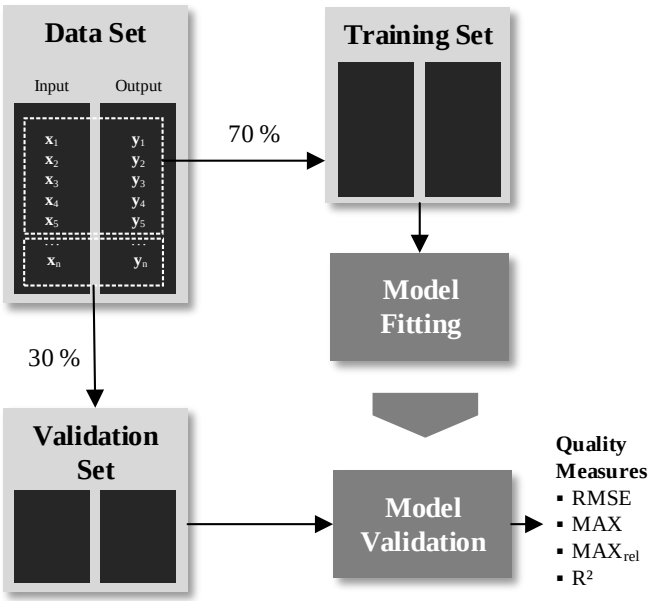


Figure 4: Splitting data into two independent subsets, one exclusively for model fitting and the other for model validation

V. Application of Metamodeling for Total Vehicle Concept Design

This section compares the metamodeling techniques listed in Sec. IV-B based on the measures of quality given in IV-C. As one main goal for metamodeling is to lower the computational burden, this comparison will additionally take the aspect time into consideration. This includes, on the one hand, the time that can be saved in the simulation of the metamodel compared to the reference model and, on the other hand, the time required for modeling and training data generation. But before this, first the application settings will be clarified.

A. Application Settings

The application of the metamodeling techniques for total vehicle concept design will focus on an early stage of development. In this early stage, technical measures, such as the use of a newly developed drive train component or the influence of new materials, e.g. lightweight materials for the car body design, have to be assessed. This usually results in numerous interactions which makes it difficult for the developer to efficiently evaluate the full potential of a concept. Also, in the early stage of development, information about newly developed components or technical measures is scarce. As a result, the variaty of some selected basic parameters in the vehicle, such as the variation of the vehicle weight, the change in engine performance or efficiency, is analysed as a substitute for the technical measures [1].

In the presented use case the following five vehicle parameters were choosen for variation:

$$\mathbf{X} = \begin{bmatrix} \underset{\text{Vehicle weight}}{\mathbf{x}_1} & \underset{\text{Drag coefficient}}{\mathbf{x}_2} & \underset{\text{Frontal area}}{\mathbf{x}_3} & \underset{\text{HV Load}}{\mathbf{x}_4} & \underset{\text{Rolling resistance coefficient}}{\mathbf{x}_5} \end{bmatrix}$$

Each input was varied around its reference value in the range of $\pm 20./\%$ except the additional high voltage (HV) load, which was varied from -500 to $+1500\,\mathrm{W}$. For parameter variation the experimental design generated with the SOBOL sequence was used, as it shows good space-filling characteristics (see Sec. III-A).

The responses that the metamodel approximates are:

$$\mathbf{Y} = \begin{bmatrix} \underset{\text{Consumption WLTC}}{\mathbf{y}_1} & \underset{\text{Consumption NEDC}}{\mathbf{y}_2} & \underset{\text{Acceleration Time 0-100 km/h}}{\mathbf{y}_3} \end{bmatrix}$$

The outputs $\mathbf{y_1}$ and $\mathbf{y_2}$ represent the amount of energy consumpt during the *Worldwide Harmonized Light-Duty Vehicles Test Cycle*[5] (WLTC) and the *New European Driving Cycle* (NEDC). As measure for the acceleration performance the amount of time the vehicle needs to acceleration from 0 to 100 km/h (T_{0-100}) is the third output $\mathbf{y_3}$.

As a reference serves a longitudinal dynamics model of a battery electric vehicle implemented in Modelica/Dymola. The implementation of the metamodeling methods as well as the evaluation is carried out in MATLAB. With the *Statistics and Machine Learning Toolbox* modeling of the ANN, the GPM, the SVM as well as the RTREEs are done while the *Adaptive Regression Splines Toolbox* for MATLAB/Octave licenced under GNU GPL is used to generate the MARS model, which is available here [37]. The application setting is summerized in Fig. 5 and system configuration is listed in Table: II (see Appendix: A). As it is recommended by Siebertz [5] one metamodel for each output y was build. Regarding the application setup, this results in three metamodels for each method.

B. Accuracy Evaluation

After training the metamodels with the training set (see Sec. IV-C) each metamodel was tested with a validation data set which was not used for modelfit. Fig. 6 a) - c) show the results for all three metamodels and each metamodeling method. As illustrated in the bar graphs, the RTREE has the lowest accuracy, both in the consumption approximation for the WLTC and NEDC as well as for the approximation of the acceleration time T_{0-100}. For both drive cycles this methods computes a maximum error of approximatly $\mathrm{MAX}_{\mathrm{rel}} \approx 15\,\%$, while the other methods provide results less than 1 % RMSNE and an maximum error $\mathrm{MAX}_{\mathrm{rel}} < 3,5\,\%$. The excellent result of the ANN is to be emphasized, as it produces maximimum errors $\mathrm{MAX}_{\mathrm{rel}} < 0,05\,\%$ for consumption approximation and maximal errors $\mathrm{MAX}_{\mathrm{rel}} < 0,04\,\%$ for acceleration time approximation. Even the very good results in global approximation of the GPM are outperformed by the ANN (see RMSE and RMSNE). This excellent accuracy is confirmed by the R^2 value (see Fig. 6 d)). For detailed model validation results please see Table: I. In addition to the evaluation of the approximation quality, the following section presents the time savings in more detail.

[5]Class 3

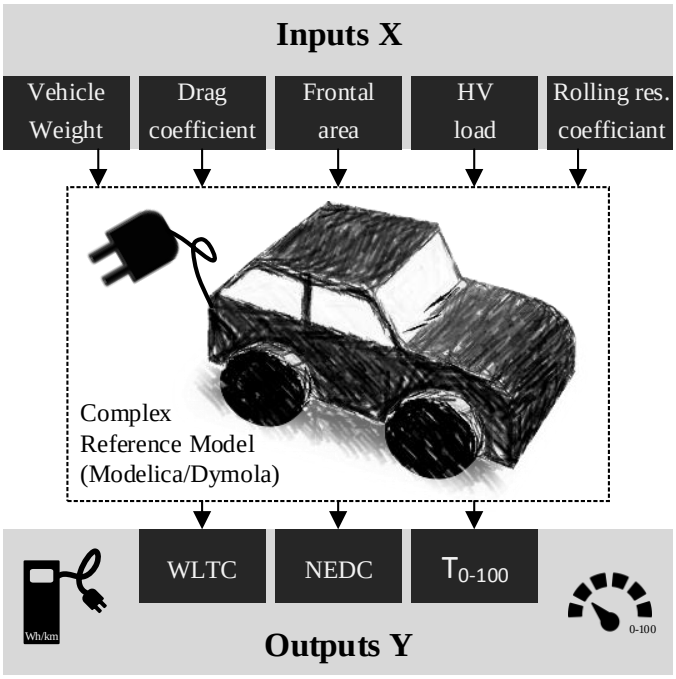
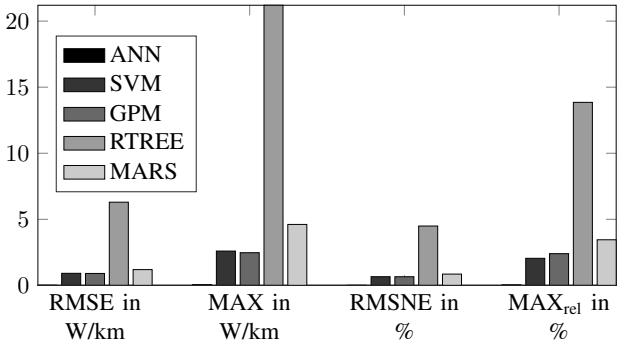


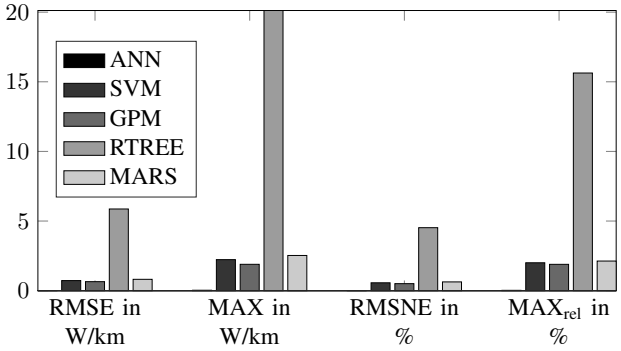
Figure 5: Application of metamodeling showing the input and output settings to perform simulations on a complex reference model in order to generate training

Table I: Accuracy evaluation measures (best result **bold**)

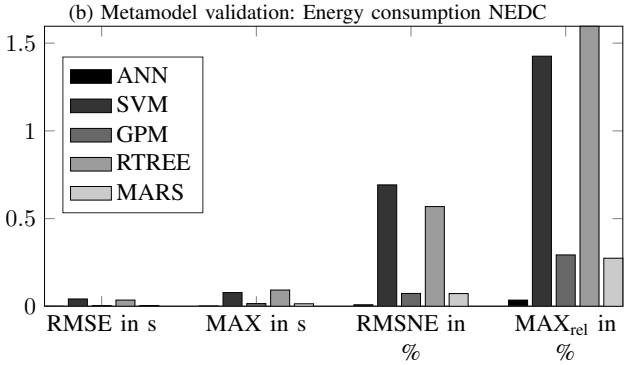
		ANN	SVM	GPM	RTREE	MARS	unit
WLTC	RMSE	**0,010**	0,915	0,904	6,296	1,195	W/km
	MAX	**0,056**	2,598	2,473	21,200	4,614	W/km
	RMSNE	**0,008**	0,658	0,656	4,492	0,856	%
	$\mathrm{MAX}_{\mathrm{rel}}$	**0,042**	2,050	2,401	13,851	3,452	%
	R^2	**1,000**	0,996	0,996	0,819	0,993	-
NEDC	RMSE	**0,008**	0,723	0,646	5,865	0,815	W/km
	MAX	**0,030**	2,227	1,892	20,118	2,527	W/km
	RMSNE	**0,006**	0,565	0,503	4,523	0,629	%
	$\mathrm{MAX}_{\mathrm{rel}}$	**0,024**	2,003	1,891	15,630	2,127	%
	R^2	**1,000**	0,998	0,998	0,857	0,997	-
T0-100	RMSE	**<0,001**	0,042	0,005	0,036	0,005	s
	MAX	**0,002**	0,08	0,016	0,093	0,015	s
	RMSNE	**0,009**	0,692	0,073	0,569	0,073	%
	$\mathrm{MAX}_{\mathrm{rel}}$	**0,036**	1,425	0,293	1,596	0,275	%
	R^2	**1,000**	0,995	0,999	0,996	0,999	-



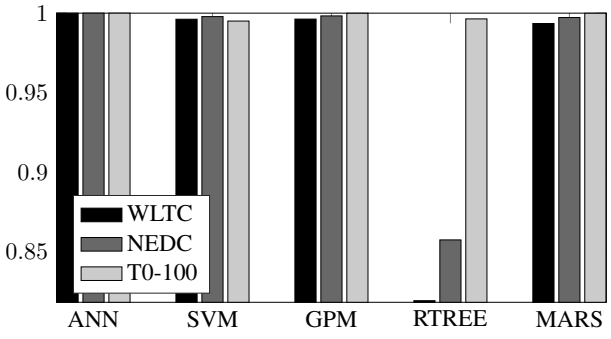
(a) Metamodel validation: Energy consumption WLTC



(b) Metamodel validation: Energy consumption NEDC



(c) Metamodel validation: Acceleration time 0-100 km/h



(d) Metamodel validation via R square value

Figure 6: Metamodel validation

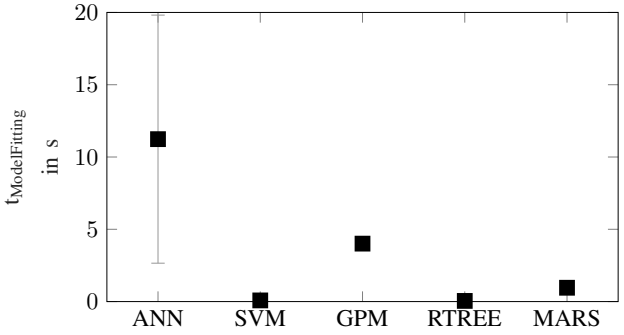


Figure 7: Mean time for metamodel fitting which includes the training of three models (one per model output y_i) and errorbars indicating the standard deviation

C. Time Evaluation

At first we will examine how time-consuming the model fitting process is. For this purpose, each method is used to create ten models and the average time duration per model fitting as well as the standard deviation were investigated. As can be seen in Fig. 7 the training process of the ANN is the longest and also subject to a relatively large uncertainties, which is due to the non-deterministic training process of the ANN. Nevertheless the modeling process can be carried out in seconds on a notebook with standard equipment (see Table: II). In average the ANN needs more than 10 times as much training time as the SVM or RTREE, which are trained in less than $1\,\mathrm{s}$. Next we take a look at the overall time-balance, consisting of the sampling time, the simulation time for the complex reference model (in order to obtain the data set) and the time for model fitting. This time, which must be invested for the metamodeling process (see Fig. 1), we set against a 'fictitious' time-saving. This means, that we also want to investigate how much faster a metamodel is in comparison to the reference model. Therfore, the ANN is selected as an example method.

As already known, the training of an ANN takes an average $11\,\mathrm{s}$. The generation of the experimental design by means of the SOBOL sequence (see sec. III-A) takes less than one second and is therefore negligible. The generation of the data set, i.e. the simulation of the reference model, takes the majority of the time as it lasts more than $2000\,\mathrm{min}$ ($\approx 33\,\mathrm{h}$). Fig.8 shows this overall time-balance. Computing the same number of simulation ($3 \cdot 1024 = 3072$) with the generated ANN takes approximately $10\,\mathrm{ms}$, which accelerates the calculation by a factor of 10^7. Please remind this analysis being an example without the claim of generalization. Nevertheless, it can be seen that with metamodeling considerable time saving are feasible, once the necessary data is obtained.

VI. Conclusion and Future Prospects

In this paper, metamodeling for vehicle concept design is investigated.

After discussing the overall process of metamodel building (Sec. II), important steps of this model building process were

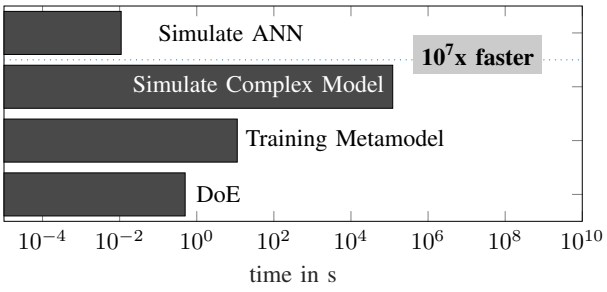


Figure 8: Overall time-balance showing the time invest for the metamodel building process consisting of the sampling time (DoE), the simulation time for the complex reference model, the time for model fitting and a 'fictitious' time-saving while computing the same number of simulations with the metamodel (ANN) (10^7 x faster than complex model)

investigated. On the one hand this involves data generation, which requires a suitable experimental plan, and, on the other hand, methods that are able to mimic complex input-output relations based on this data. For both aspects, common methods were listed, described and possibilities for evaluation were discussed (see Sec. III - IV). In this, current methods of DoE and the intelligent modeling from the field of computational intelligence and (supervised) machine learning were presented.

With this theoretical background, a suitable experimental design using the SOBOL sequence is generated and the methods for modeling are selected (see sec. V). The implemented models are then compared with the help of objective measures regarding their approximation accuracy. In addition, the aspect of time saving was considered, since this is one of the main drivers for metamodeling.

It can be summarized that the implemented metamodels, with the exception of Regression Trees (RTREES) and Support Vector Machines (SVMs), achieve good approximation accuracy. The Artificial Neural Network (ANN), implemented as a feedforward MLP with one hidden layer and 6 neurons, proved to be superior method compared to Multivariate Adaptive Regression Splines (MARS) and Gaussian Process Models (GPMs). However, the training of the ANN took the most time. This may be of relevance if the modeling process has to be carried out very frequently. Then GPM or MARS offer a good compromise between training duration and accuracy.

The high accuracy and the very fast execution time make ANN a very versatile method for vehicle concept design. This confirms earlier investigations of [2] and [1]. In contrast to their work [1], [2], both the inputs and the outputs are supplemented by additional factors. In addition to discrete inputs (e.g. number of electric machines), more complex outputs are to be tested for the evaluation of vehicle longitudinal performance as presented in [38]. It will be interesting to use adaptive, sequential DoE instead of the one-shot sampling shown here to speed up the modeling process. Based on this, future research will focus on creating a fast analysis module for efficient optimization in order to find an optimal powertrain configuration which is both energy efficient and attractive to the user.

APPENDIX
SYSTEM CONFIGURATION

Table II: System Configuration

CPU	Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz (DualCore)
RAM	12,0 GB
Graphic	Grafik Intel(R) HD Graphics 5500
OS	Windows 7 64-Bit
MATLAB	R2016b 64-Bit
	Neural Network Toolbox ver. 9.1 (R2016b)
	Statistics and Machine Learning Toolbox ver. 11.0 (R2016b)
	Adaptive Regression Splines Toolbox for MATLAB/Octave ver. 1.13.0 (GNU GPL) [37]
	Hammersley sequence ver. 2016/08 (GNU LGPL) [23]
Dymola	2016 64-Bit

REFERENCES

[1] S. Moses, C. Gühmann, and J. Jäkel, "Optimierung von fahrzeugkonzepten in der frühen entwicklungsphase mit hilfe genetischer algorithmen und künstlicher neuronaler netze", in *Proceedings. 21. workshop computational intelligence, dortmund, 1. - 2. dezember 2011*, F. Hoffmann and E. Hüllermeier, Eds., KIT Scientific Publ, 2011, pp. 77–92, ISBN: 9783866447431.

[2] M. Eghtessad, "Optimale antriebsstrangkonfigurationen für elektrofahrzeuge", PhD thesis, Technische Universität, Braunschweig, 2014.

[3] J. Fuchs, "Analyse der wechselwirkungen und entwicklungspotentiale in der auslegung elektrifizierter fahrzeugkonzepte", PhD thesis, Technische Universität, München, 2015.

[4] H.-H. Braess and U. Seiffert, *Vieweg handbuch kraftfahrzeugtechnik*, 7., aktual. Aufl. 2013, ser. SpringerLink : Bücher. Wiesbaden: Springer Vieweg, 2013, ISBN: 978-365-80169-1-3. DOI: 10.1007/978-3-658-01691-3.

[5] K. Siebertz, D. van Bebber, and T. Hochkirchen, *Statistische versuchsplanung: Design of experiments (doe)*, ser. VDI-Buch. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010, ISBN: 3642054935.

[6] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization", *Journal of mechanical design*, vol. 129, no. 4, p. 370, 2007, ISSN: 10500472. DOI: 10.1115/1.2429697.

[7] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, "Metamodels for computer-based engineering design: Survey and recommendations", *Engineering with computers*, vol. 17, no. 2, pp. 129–150, 2001, ISSN: 0177-0667. DOI: 10.1007/PL00007198.

[8] K. Crombecq, E. Laermans, and T. Dhaene, "Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling", *European journal of operational research*, vol. 214, no. 3, pp. 683–696, 2011, ISSN: 03772217. DOI: 10.1016/j.ejor.2011.05.032.

[9] S. Matz and M. Lienkamp, "Optimisation of vehicle concepts in a multimodal environment with regard to user benefit", in *Conference on future automotive technology: Focus electromobility*, 2013.

[10] J. Fuchs, "Analyse der wechselwirkungen und entwicklungspotentiale in der auslegung elektrifizierter fahrzeugkonzepte", PhD thesis, Göttingen, 2015.

[11] P. Reupold, "Lösungsraumanalyse für hauptantriebsstränge in batterieelektrischen straßenfahrzeugen", PhD thesis, Technische Universität, München, 2014. [Online]. Available: http://mediatum.ub.tum.de?id=1120693.

[12] Liang Li, Yahui Zhang, Chao Yang, Xiaohong Jiao, Lipeng Zhang, and Jian Song, "Hybrid genetic algorithm-based optimization of powertrain and control parameters of plug-in hybrid electric bus", *Journal of the franklin institute*, vol. 352, no. 3, pp. 776–801, 2015, ISSN: 0016-0032. DOI: 10.1016/j.jfranklin.2014.10.016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016003214002968.

[13] Lianghong Wu, Yaonan Wang, Xiaofang Yuan, and Zhenlong Chen, "Multiobjective optimization of hev fuel economy and emissions using the self-adaptive differential evolution algorithm", *Ieee transactions on vehicular technology*, vol. 60, no. 6, pp. 2458–2470, 2011, ISSN: 0018-9545; 1939-9359. DOI: 10.1109/TVT.2011.2157186.

[14] V. T. Long and N. V. Nhan, "Bees-algorithm-based optimization of component size and control strategy parameters for parallel hybrid electric vehicles", *International journal of automotive technology*, vol. 13, no. 7, pp. 1177–1183, 2012, ISSN: 1229-9138. DOI: 10.1007/s12239-012-0121-5. [Online]. Available: http://dx.doi.org/10.1007/s12239-012-0121-5.

[15] C. Appel, S. Freudenstein, and C. Temmen, "Comprehensive simulation for powertrain electrification", *Mtz worldwide*, vol. 75, no. 2, pp. 10–17, 2014, ISSN: 2192-9114. DOI: 10.1007/s38313-014-0016-0.

[16] C. Appel, *Comprehensive and cross-domain vehicle simulation for electrification: Vehicle & powertrain system simulation*, Frankfurt, 2014.

[17] P. A. Fritzson, *Introduction to modeling and simulation of technical and physical systems with modelica*, Online-Ausg, ser. IEEE Xplore Digital Library. Hoboken, N.J: Wiley, 2011, ISBN: 1-11-809424-7.

[18] L. Guzzella and A. Sciarretta, *Vehicle propulsion systems: Introduction to modeling and optimization*, 3rd ed. Springer, 2013, ISBN: 978-3-642-35912-5. [Online]. Available: http://slub.eblib.com/patron/FullRecord.aspx?p=337428.

[19] Husslage, Bart G. M., G. Rennen, van Dam, Edwin R., and D. den Hertog, "Space-filling latin hypercube designs for computer experiments", *Optimization and engineering*, vol. 12, no. 4, pp. 611–630, 2011, ISSN: 1573-2924. DOI: 10.1007/s11081-010-9129-8. [Online]. Available: http://dx.doi.org/10.1007/s11081-010-9129-8.

[20] E. Janouchová and A. Kučerová, "Competitive comparison of optimal designs of experiments for sampling-based sensitivity analysis", *Computers & structures*, vol. 124, pp. 47–60, 2013, ISSN: 00457949. DOI: 10.1016/j.compstruc.2013.04.009.

[21] O. T. Kajero, T. Chen, Y. Yao, Y.-C. Chuang, and D. S. H. Wong, "Meta-modelling in chemical process system engineering", *Journal of the taiwan institute of chemical engineers*, 2016, ISSN: 18761070. DOI: 10.1016/j.jtice.2016.10.042.

[22] M. Liefvendahl and R. Stocki, "A study on algorithms for optimization of latin hypercubes", *Journal of statistical planning and inference*, vol. 136, no. 9, pp. 3231–3247, 2006, ISSN: 03783758. DOI: 10.1016/j.jspi.2005.01.007.

[23] J. Burkardt, *Hammersley - the hammersley quasi monte carlo (qmc) sequence*. [Online]. Available: http://people.sc.fsu.edu/~jburkardt/m_src/hammersley/hammersley.html.

[24] F. Hickernell, "A generalized discrepancy and quadrature error bound", *Mathematics of computation of the american mathematical society*, vol. 67, no. 221, pp. 299–322, 1998.

[25] K. P. Murphy, *Machine learning: A probabilistic perspective*. MIT Press, 2012, ISBN: 978-0-262-01802-9.

[26] A. Kroll, *Computational intelligence: Eine einführung in probleme, methoden und technische anwendungen*. München: Oldenbourg Verlag, 2013, ISBN: 9783486989786.

[27] J. Lunze, *Künstliche intelligenz für ingenieure*, 2., völlig überarbeitete Auflage. München: Oldenbourg Verlag, 2010, ISBN: 9783486702224.

[28] R. Kruse, C. Borgelt, C. Braune, F. Klawonn, C. Möwes, and M. Steinbrecher, *Computational intelligence: Eine methodische einführung in künstliche neuronale netze, evolutionäre algorithmen, fuzzy-systeme und bayes-netze*, 2. Aufl. 2015, ser. SpringerLink : Bücher. Wiesbaden: Springer Vieweg, 2015, ISBN: 3658109041.

[29] S. J. Russell and P. Norvig, *Künstliche intelligenz: Ein moderner ansatz*, 3., aktualisierte Aufl., ser. Pearson-Studium. München [u.a.]: Pearson, 2012, ISBN: 3868940987.

[30] F. Hoffmann and E. Hüllermeier, Eds., *Proceedings. 21. workshop computational intelligence, dortmund, 1. - 2. dezember 2011*, KIT Scientific Publ, 2011, ISBN: 9783866447431. [Online]. Available: https://books.google.de/books?id=hBLTh2XP3zQC.

[31] Friedman, H., Jerome, "Multivariate adaptive regression splines (mars)", *The annals of statistics*, no. 19, 1991.

[32] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features", in *Machine learning: Ecml-98*, ser. Lecture Notes in Computer Science, J. G. Carbonell, J. Siekmann, G. Goos, J. Hartmanis, J. van Leeuwen, C. Nédellec, and C. Rouveirol, Eds., vol. 1398, Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142, ISBN: 978-3-540-64417-0. DOI: 10.1007/BFb0026683.

[33] S. P. Boyd and L. Vandenberghe, *Convex optimization*, 9. print. Cambridge [u.a.]: Cambridge Univ. Pr, 2011, ISBN: 0521833787.

[34] T. Nueesch, P. Elbert, M. Flankl, C. Onder, and L. Guzzella, "Convex optimization for the energy management of hybrid electric vehicles considering engine start and gearshift costs", *Energies*, vol. 7, no. 2, pp. 834–856, 2014, ISSN: 1996-1073. DOI: 10.3390/en7020834.

[35] D. Krause, K. Paetzold, and S. Wartzack, Eds., *Design for x: Beiträge zum 26. dfx symposium, bd. 26*, Hamburg: TuTech Verlag, 2015, ISBN: 3941492934.

[36] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments", *Statistical science*, pp. 409–423, 1989.

[37] G. Jekabsons, *Regression software for matlab/octave*, 3.09.2016. [Online]. Available: http://www.cs.rtu.lv/jekabsons/regression.html.

[38] M. Helbing, B. Bäker, and S. Schiffer, "Electrified powertrain design of road vehicles: New evaluation framework", in *International conference on electrical systems for aircraft, railway, ship propulsion and road vehicles and the international transportation electrification conference*, ESARS-ITEC, Ed., 2016.