Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Automated model generation in the field of electrical automotive driveline components

-

Daniel Regulin Michael Schneider Sebastian Rehberger Birgit Vogel-Heuser

Institute of Automation and Information Systems, Technische Universität München, Garching, 85748 Germany

(e-mail: {regulin, schneider, rehberger, vogel-heuser}@ais.mw.tum.de)

Abstract: Model based techniques are increasingly applied to develop mechatronic systems. The automated modeling methodology supports the model synthesis of electrical components included in the automotive driveline. Since the time to market of vehicles is decreasing steadily and the demand for efficiency and quality is increasing, model based methods are required to enable the possibility of simultaneous engineering. The proposed solution is based on available measurement series of previously validated components as the lambda sensor and the electrical water pump. A model generation library enables the assembling of analogous models according to a predefined component classification. To increase the transparency and quality of the models experts can implement extensions and predefine parameters of the physical software models. Subsequently a parameter estimation identifies the values of the parameters by the objective to minimize the difference between measured values and simulation results. Applying this method the model dependant difference between simulation results and measured reference values could be minimized to 8.6% considering the lambda sensor.

Keywords: model approximation, objectoriented modeling, physical models, computeraided engineering, vehicular components.

1. INTRODUCTION

The difficulties in development of complex mechatronic systems is increasing steadily. This trend is caused by decreasing design and construction cycles as well as increasing expectations regarding quality and efficiency. Especially components of the electrified driveline need to be adapted to the different vehicle variants. Currently the operational strategy is developed by preliminary tests and evaluation of the measured values. Based on the experience of experts the component behavior is adapted to the predefined characteristics. Since this iterative procedure applying real systems demands high financial and time investments model based methods can help to increase the efficiency (Karlberg et al. [2013]). Scientists present many methods and possibilities which lead to fulfill the raised requirements. Representative for numerous methods Kalawsky et al. [2013] shows the approach of model based simultaneous engineering encombassing a hardware-in-the-loop-analysis applying Matlab/ Simulink and the xPC Target. Kalawsky et al. points out the challenge of combining domain specific modeling tools and the effort to build new models. Since evaluations in an early state of design cannot be executed using existing physical systems, the model driven development is applied and adjusted to the real system in later stages of engineering by parameter tuning.

Physical models represent the real system by software components which are based on mathematical equations using an equal parameter set compared to the physical component. The connection of represented component blocks enables the design of complex mechatronical systems. Thus evaluations regarding dynamic behavior, energy consumption and thermal characteristics are possible. Many development environments e.g. Matlab/ Simulink and Dymola provide a graphical user interface which facilitates the modeling process and comprehension even for people without knowledge of differential equations.

This paper presents a method which enables the possibility to generate models in an automated way. Especially in the field of automotive systems components are characterized by a high rate of reuse. Since measurement values for these proven components are available, the objective of this work is to use existing measurement information and transfer the generated knowledge into models.

An automated modeling process for physical systems requires both, the model composition of classified physical systems and the parameter estimation process to achieve a realistic behavior. In addition an important issue turns out by the possibility to extend the generated model by the knowledge and experience of professional users. Brommundt et al. [2012] presents one way how to apply object orientated modeling and graphical editors to simulate complex mechatronic systems. Barth and Fay [2013] uses and recommends object orientation for an automatic generation of simulation models based on data exchange and P&I-diagrams. It is shown how simulations can lower fault rates by early identification of failure by generation of physical models. According to Fritzson [2004] the Modelica-standard supplies the paradigms multi-physical

domains and object oriented modeling using an equation based language. In Brommundt et al. [2012] the advantages of object oriented modeling methods for a simplified construction of complex models is explained. The Modelica modeling language primarily developed by the continuous systems community in Europe offers a freely-available Modelica Standard Library which is supported by simulation environments e.g. OpenModelica, Wolfram System Modeler, Dymola etc.. These editors contain technical system items of different domains. Furthermore it supports the creation of customized libraries, the extension of the Modelica Standard Library or the purchase of licenses for commercial libraries, for example PowerTrain-library, VehicleDynamics-library or ElectricPower-library.
The Dymola-editor supports the graphical connection of different objects using input and output ports. Beside numerical Data, physical quantities such as voltage, current, pressure, temperature, etc. are potential choices. These features offer the possibility to use inheritance for reusing existing components and adaption of those characteristics to fulfill newly defined requirements. An exemplary model demonstrates how a multi-physical domain application of a floating wind turbine can be modeled in an effective way, without the user being an expert in multi-body dynamics or object oriented modeling. Regarding these prerequisites Modelica should be applied in this approach to generate the analogous models.
To achieve a satisfying model behavior, parametrization is the second main subject of the automated model synthesis. Chen et al. [2011] presents the process of parameter estimation for transfer functions and evaluates the estimation results of different approaches. In addition increasing estimation options are supported by tools e.g. Matlab/Simulink. Ljung and Singh [2012] introduces the eighth version of the *system identification toolbox* which enables an easy access to execute different identification methods. Applying these tools for estimation of system characteristics enables an easy way of model synthesis. In addition the Wiener Model structures can be applied to characterize mechatronic systems. Wills et al. [2011] presents algorithms for blind identification of Wiener Models where the input signal is not known and the measurements are corrupted by noise. The results of the referred methods are transfer functions or state space matrices, whose parameters can hardly be assigned to the to real parameters of a physical system. Since Regarding the method of physical modeling, the process of parameter estimation has to be extended. Elmqvist et al. [2005] describes the design optimization of physical models using the Modelica development environment Dymola and the *design optimization toolbox*. Since the parameter estimation is based on large data series and the level of automation should be increased, Matlab and Simulink turn out as an appropriate framework. The data processing, evaluation, the Modelica model import facility as well as the object oriented programming language are prerequisites for an automated model generation.

2. CONCEPT OF THE AUTOMATED MODEL SYNTHESIS

The following chapter presents the concept of the automated model synthesis containing architectural characteristics of the models and the tool chain. In addition the synthesis process concluding mathematical context is described.

2.1 Classification of heterogenous system components

Conventional modeling processes try to identify each part and its connection within one component and represent the behavior by related blocks containing the parameter an equation set. In contrast to this method the approach of automated model synthesis classifies different domains of a mechatronical system. To define the influencing domains for a system component, it is necessary to interview experts on the field of component topology. Focusing electrical automotive parts of the driveline following domain categories could be considered:

- mechanics
- electric-mechanical
- electric-thermal
- thermal-fluid

Each domain is represented by one generic equivalent model and its appropriate connectors. The model for a specified component can be generated by combining generic blocks of the corresponding domains.

2.2 Model architecture in Modelica

The paradigm object orientation allows programmers to build complex software systems in short periods of time and ensure a high level of maintainability.

According to Sinha et al. [2001] high complexity and variety of mechatronic systems may lead to high effort applying procedural or functional languages. In addition the prerequisites for an efficient reuse are not supported. Regarding these considerations an object oriented approach should be applied to design the model generation library.

The architecture was composed based on results of previous work of Regulin et al. [2013]. According to the identified domains and interrelationships the generic blocks are organized in domain specific packages containing the related basic class. Due to a simple library of equivalent models has been initialized. Inheritance helps to set up extensions by implementation of expert knowledge about parameters or the component setup.
To generate a component model the user selects the model blocks according to the predefined parts of the classification. The compatibility among the blocks is guaranteed by standardized interfaces which have to be connected. Finally the representing model for a component is composed.
An advantage of this architecture turns out by the possibility to use the basic classes in case of ignorance about a component and the application of adapted extensions to increase the accurateness by user knowledge.

2.3 Block interfaces

The applied blocks of the Modelica standard library are based on mathematical equations. In contrast to Matlab/Simulink Modelica modeling language supports a bidirectional connection of interfaces. From a semantic point of view these connectors contribute further equations including all submodels of the linked classes. Pepper et al.

Pepper et al. [2011] defines the semantic link among classes linked by connectors and the influence on the equations. To generalize the blocks, standardized interfaces and units are required.

Regarding the presented approach using different domain specific packages and models, the accurate connection of the required equations is important to build executable component models. The standardization of block interfaces is based on the in- and outputs of the basic classes and could be identified as follows:

- flange -transmission of mechanical torque
- data -exchange of numerical information
- heat port -transmission of thermal values

Applying these connectors all focussed components of the electrified driveline can be generated.

2.4 Composition of identified partial models and mathematical context

The following section presents the developed basic classes including parameters which are initialized by variables.

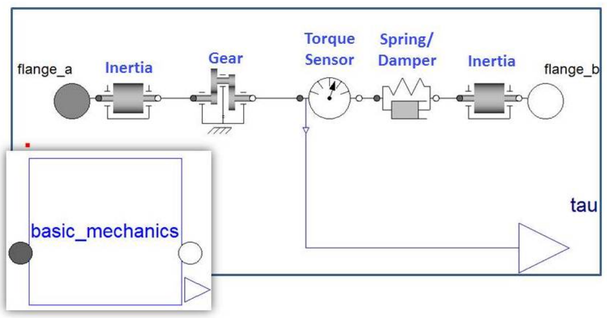

Fig. 1. Class Basic_mechanics

A view under the top layerof the *Basic_mechanics* class in figure 1 shows the components contained in the model. In general mechanical systems are characterized by masses as well as torsional forces. Limited stiffness and damping characteristics which can usually be substituted applying a spring-damper-system. Since one mass combined with a spring-damper-system can only be applied for one oscillation mode it is necessary to increase the dimension of freedom by a second mass in many cases. In addition the difference of rotary speed between input and output can be adapted by a transmission ratio. Applying this model the behavior of many rotational mechanical systems can be approximated. An adaption to very simple structures, for example shafts, is possible by parameter setting e.g. $J_2 = 0\ kg \cdot m^2$. Subsequently following parameters have to be determined for a simulation:

- transmission ratio r
- inertia J_1 representing the one mass of the system
- inertia J_2 representing the second mass of the system
- stiffness c representing the mechnical stiffness
- absorption d representing the machanical absorption

The following equations represent the mathematical context:

$$J_1\ddot{\phi}_1 \cdot r = c(\phi_{J_1} - \phi_{J_2}) + d(\dot{\phi}_{J_1} - \dot{\phi}_{J_2}) \quad (1)$$

$$J_2\ddot{\phi}_2 = c(\phi_{J_2} - \phi_{J_1}) + d(\dot{\phi}_{J_2} - \dot{\phi}_{J_1}) \quad (2)$$

To approximate the behavior of electrical systems applying physical models it is necessary to define the output. Electricity can be transformed into thermal, mechanical or chemical energy. Since the connectors of adjacent model parts may need a specific type of energy the basic electric classes are classified and grouped in separate packages. Figure 2 shows the *Basic_electric_mechanical* class which contains of controlled power supply, an inductance, a resistance, as well as an *electro motoric force* block (emf) which transforms electrical into mechanical rotational energy in relation to the constant k. The interface to the mechanical system is realized by the mechanical connector. According to the equations, the desired motor power can be adjusted using a controllable voltage source, which concurrently represents the interface to the system controller. Kirchhoff's laws can be applied to describe the behavior of the resulting circuit considering the following boundary conditions:

- electrical current i
- inertia Φ representing the converted mechanical energy
- resistance R representing the entire electrical resistance
- inductance L representing the entire electrical inductance
- inducted countervoltage u_{ind}
- constant k which defines the torque ratio as well as the iducted voltage into the electric circuit

The mathematical decription is given by 3.

$$u(t) = R \cdot i + L \cdot \frac{di}{dt} + u_{ind} = R \cdot i + L \cdot \frac{di}{dt} + k \cdot \dot{\phi}_{motor} \quad (3)$$

Since only the electrical parameters are focussed in this basic class the resulting torque can simply expressed by 4.

$$k \cdot i = M_{mech}. \quad (4)$$

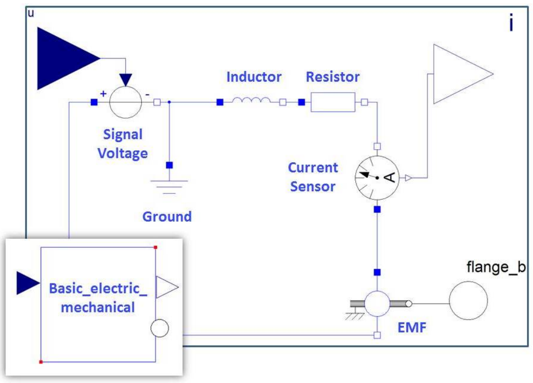

Fig. 2. Class Basic_electric_mechanic

The convertion of electrical to thermal energy is an often occurring phenomenon in mechatronic systems. Modelica supports the thermal connection of blocks by an appropriate interface. The *Basic_electric_thermal* class illustrated in figure 3 shows the model architecture. According to Joule's first law a temperature dependent resistive heater emits thermal energy respective the electric current and a material specific thermal coefficient. A evaluation is based on the following parameters:

- electrical current i

- resistance R representing the entire electrical resistance
- thermal coefficient α material and temperature dependent parameter
- current temperature t
- reference temperature t_0

Equation 5 represents the thermal, 6 the electrical current dependeny:

$$R(t) = R_{t_0} \cdot (1 + \alpha_{t_0} \cdot (t - t_0)) \tag{5}$$

$$P_{therm} = R(t) \cdot i^2 \tag{6}$$

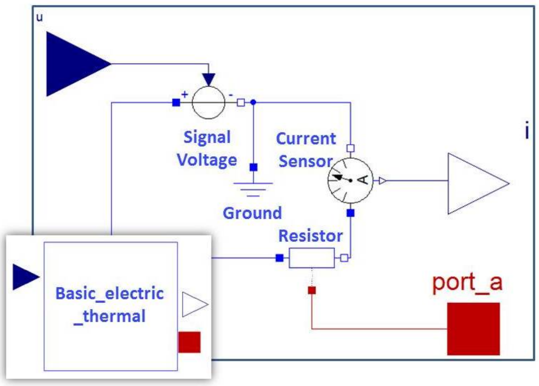

Fig. 3. Class Basic_electric_thermal

A similar approach can deliver a model representing the conversion of mechanical to thermal energy; since no focused component requires this transformation it is not implemented yet.

The *Basic_thermal_fluid* class typically can be applied to describe the system behavior of cooling or heating processes [4]. A heat capacity specifies the required amount of thermal energy for changing the temperature defined by the surrounding model parts. The temperatur T_2 influences the thermal energy which leads to the temperature T_1. According to the principle of convection, a temperature gradient ∂_{T_1,T_2} and a convective thermal conductance G_c define the heat transfer between the heat ports. Surrounding conditions are restricted by the following issues:

- temperature T_1 (solid heat port); T_2 (fluid heat port)
- convective thermal conductance G_c
- heat flow rate Q_{flow}
- heat transfer coefficient c_o
- convection surface A
- thermal capacity C

The equations below express the descripted charackteristics:

$$Q_{flow} = G_c \cdot (T_1 - T_2) \tag{7}$$

$$G_c = c_o \cdot A \cdot \left(\frac{T_2}{3,6}\right)^{0.78} \tag{8}$$

The generical analogous models are organized in a newly developed Modelica model generation library which enables the easy synthesis of component specific models according to the classification [current contend fig. 5].

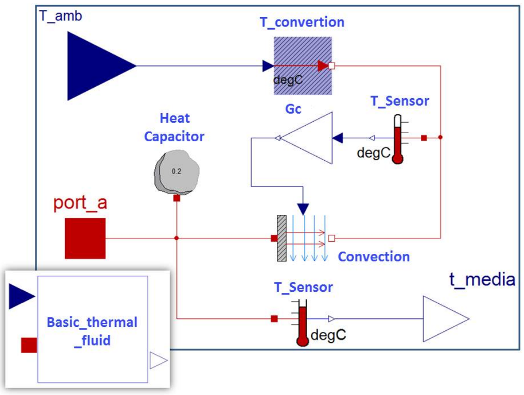

Fig. 4. Class Basic_thermal_fluid

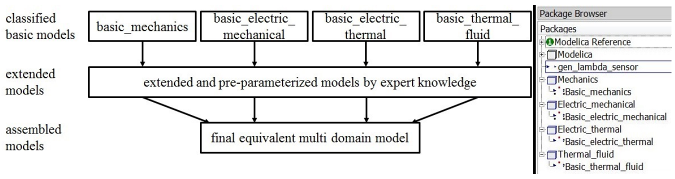

Fig. 5. Modelica library for model generation

2.5 *Parameter estimation*

To implement the correct characteristics, the parameters of the analogous model have to be defined. This step has been automated using Matlab optimizers and Simulink. An appropriate framework supports the simulation, optimization and parameter estimation process [fig. 6]. Figure illustrates the developed architecture of the parameter estimation program in Matlab.

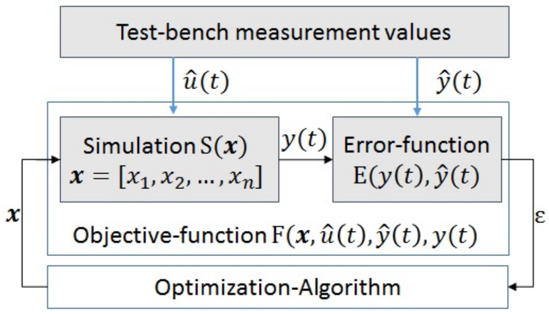
Test-bench measurement values
$\hat{u}(t)$
$\hat{y}(t)$
Simulation S($\boldsymbol{x}$)
$\boldsymbol{x} = [x_1, x_2, \dots, x_n]$
$y(t)$
Error-function
E($y(t), \hat{y}(t)$)
$\boldsymbol{x}$
Objective-function F($\boldsymbol{x}, \hat{u}(t), \hat{y}(t), y(t)$)
ε
Optimization-Algorithm

Fig. 6. Architecture of the parameter estimation program

Model import to Simulink The Simulation $S(\boldsymbol{x})$ delivers result which is the evaluation criterion for calculating the error ϵ [fig. 6]. Since the model parameters $\boldsymbol{x}$ are varied optimization algorithm, each iteration requires a new simulation. The interface to transfer the Modelica model designed in Dymola into Matlab is the Dymola-block of the Simulink library. This model import block enables the possibility to compile the Dymola model file into a S-fuction which is executable by the Simulink solvers. According to a successful compiling process the model parameters as well as the in- and outputs are added to the Dymola block in Simulink automatically. The control data sequence should be imported from the Matlab workspace into the Simulink model applying the *simin* block. For recording the model outputs the block *out* should be used.

Error function An parameter estimation of the created model requires an objective function to perform the optimization process. The aim is finding a set of parameters $\boldsymbol{x}$ with the result $\boldsymbol{y}^{model}$ which causes a minimal error to a reference signal $\hat{\boldsymbol{y}}$. The mismatch between the signals can be computed by mean square error (MSE) function

$$\epsilon_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y^{model}(t_i) - \hat{y}(t_i))^2. \tag{9}$$

As time reference for the computation of the MSE the time steps of the simulation of model are used

$$t_i = t_i^{model}. \tag{10}$$

Since the reference signals usually are sampled on a test bench, data series are not discretized at equally time stamps in comparison to the simulation. According to Molnar et al. [2003] the measurement series has to be interpolated. For each time step of the simulation i a corresponding measured time index j has to be found.

$$\left\{i,j \quad | \quad \hat{t}_j < t_i^{model} \leq \hat{t}_{j+1}\right\} \tag{11}$$

Since (11) is fulfilled the reference signal can be approximated by linear the interpolation.

$$\hat{y}(t_i) = \hat{y}_j + (\hat{y}_{j+1} - \hat{y}_j)\cdot\frac{t_i - \hat{t}_j}{\hat{t}_{j+1} - \hat{t}_j} \tag{12}$$

The error function (9) applies the interpolated signal to quantify the mismatch ϵ.

Optimization The process of parameter estimation is executed, applying the *fminsearch* simplex search method of Lagarias et al. [1998] implemented to Matlab. The calculated error ϵ represents the optimization input. Subsequently the optimizer varies the selected main parameters of the model to minimize the difference between measured values and simulation results.

Continuous simulation of the model can be described as non-linear and not differentiable mathematical functions. In combination with the error function (9) which evaluated each time after the computation of the simulation the overall optimization problem results to

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{t}}). \tag{13}$$

The used optimization algorithm based on the Nelder-Mead method Wright [2010] is suitable non-linear and not differentiable problems. A genetic algorithm could also be applied to solve the optimization task and will be subject of following research activities on this topic.

3. CASE STUDY LAMBDA SENSOR HEATING

This section shows the functionality of the method applied to the heating of a lambda or oxygen probe which is influenced by thermodynamical and electrical effects. The sensor delivers the oxygen proportion of the vehicular exhaust emissions to control the air-fuel ratio. Since a minimum temperature of 400 *degreeC* to 700 *degreeC* is a prerequisite for correct operation and the exhaust temperature during the start period cannot provide the required conditions, the electrical heating is necessary for calculating the quantity of fuel injection. To analyze the behavior during the engineering process and proof the measured values of driving cycles on test beds a software model is required.

3.1 Model generation for the lambda sensor heating

To generate the model for the lambda sensor heating the related generically model parts have to be chosen out of the model generation library. In this case two model parts, the *Basic_electric_thermal* and *Basic_thermal_fluid* class have to be applied. According to the interface *heat_port* of the two blocks one connector is required to build the entire analogous model. Subsequently needed in- and outputs are defined by the interfaces:

- control signal input *ctrl_var*
- time variant exhaust teperature input *T_amb*
- electrical current output *i_mod*

Figure 7 illustrates the model including interfaces and connectors.

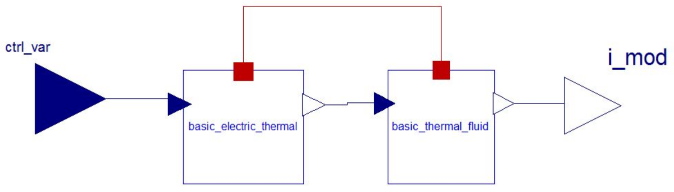

Fig. 7. Generated model of the lambda sensor heating

3.2 Parameter estimation

In case of the lambda senor heating the following parameters are varied and identified to the listed values:

Varied Parameter	Estimation result
Resistance R	0.6Ω
Thermal coefficient α	0.1
Thermal capacity C	$6.6\frac{J}{K}$
Heat transfer coefficient c_o	$11.1\frac{W}{K}$

Since the resulting value depends on the relation among these parameters, a predefinition of particular ones based on expert knowledge helps to increase the quality of the estimated characteristics.

3.3 Evaluation of the simulation results

Since the model behavior is adapted to the characteristics of the real component it can be used to evaluate the component states according to different control signal inputs. The quality of the model which can be rated by the difference of measured data and simulation results applying real control values as model input. A good conformity regarding the signal sequence can be detected by comparing the simulation results and measured values; the quantified difference constitutes 8.6% [fig.8]. Additionaly the modeling method was proofed by evaluation of further simulation results of an electrical water pump; a difference between measured values an simulation results of 9.3% could be quantified.

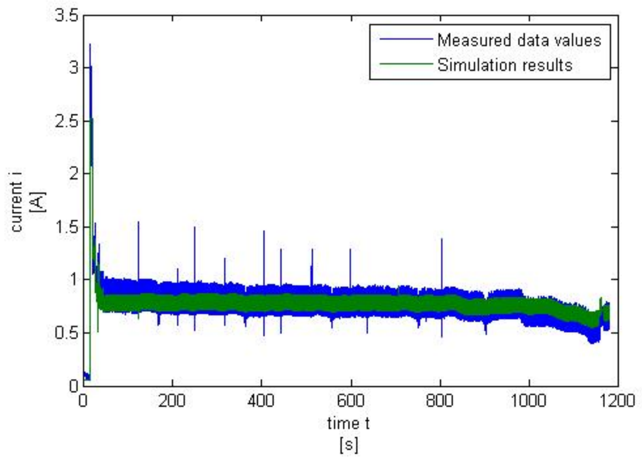

Fig. 8. Simulation results of the lambda sensor heating

4. CONCLUSION AND OUTLOOK

In this paper, a method for automated model generation has been shown and the required prerequisites in the field of automotive. Classified partial models of different domains can be assembled according to the mechanism of action considering the composition of the real component. The Modelica development environment Dymola is used to generate analogous models by drag and drop out of a newly developed model generation library. Beside the model set up including the mathematical background the automatic parametrization of characteristic parameters is the second key point of the presented method. Since Matlab is recommended for data processing, optimization and programming it is applied to estimate the corresponding parameters. Thus an model import to Simulink using the *Dymola Block* is required. To avoid limitations caused by predefined toolbox algorithms and to increase the transparency, the method is implemented using the object oriented programming in Matlab. The presented concept has been evaluated within a case study on a lambda sensor heating and an electrical water pump. Based on previously measured data series of existing nodes a fully satisfying approximation could be reached by automatic optimization of the model parameters.
Further evaluations have verified the functionality of the presented method. It is intended to increase the level of automation to reduce the generation time as well as the tool dependent work steps of the model transfer and setup to prepare the estimation.
An evaluation of experts has shown the relevance of the presented method to reduce the development time of vehicular physical models. In addition it reduces the obstacles to a model based work even for inexperienced poeple. The method was introduced for a model based evaluation of measured data series.

ACKNOWLEDGEMENTS

Represented research and development work is carried out in the project Energiemanagement III within the framework of the cooperation CAR@TUM (Munich Centre of Automotive Research) between the BMW Group and the Technical University Munich. This particular research was supported by BMW Group Research and Technology. The responsibility for this publication is held by authors only.

REFERENCES

Mike Barth and Alexander Fay. Automated generation of simulation models for control code tests. *Control Engineering Practice*, 21(2):218 – 230, 2013.

Matthias Brommundt, Michael Muskulus, Mareike Strach, Michael Strobel, and Fabian Vorpahl. Experiences with object-oriented and equation based modeling of a floating support structure for wind turbines in modelica. In *Proceedings of Simulation Conference, 2012 Winter*, pages 1–12, 2012.

Tianshi Chen, Henrik Ohlsson, and Lennart Ljung. On the estimation of transfer functions, regularizations and gaussian processes - revisited. In *World Congress*, volume 18, pages 2303–2308, 2011.

Hidling Elmqvist, Hans Olsson, Sven Erik Mattsson, Dag Brück, Christian Schweiger, Hans-Dieter Joos, and Martin Otter. Optimization for design and parameter estimation. In *Proceedings*, pages 255–266, 2005.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley-IEEE Press, 2004. ISBN 0-471-471631.

R.S. Kalawsky, J. O'Brien, S. Chong, C. Wong, H. Jia, H. Pan, and P.R. Moore. Bridging the gaps in a model-based system engineering workflow by encompassing hardware-in-the-loop simulation. *IEEE Systems Journal*, 7(4):593–605, 2013.

M. Karlberg, M. Lfstrand, S. Sandberg, and M. Lundin. State of the art in simulation-driven design. *International Journal of Product Development*, 18(1):68–87, 2013. cited By (since 1996)0.

Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

Lennart Ljung and Rajiv Singh. Version 8 of the matlab system identification toolbox. In *System Identification*, volume 16, pages 1826–1831, 2012.

K. Molnar, L. Sujbert, and G. Peceli. Synchronization of sampling in distributed signal processing systems. In *IEEE International Symposium on Intelligent Signal Processing, 2003*, pages 21–26, 2003.

Peter Pepper, Alexandra Mehlhase, Christoph Höger, and Lena Scholz. A compositional semantics for modelica-style variable-structuremodeling. In *EOOLT*, pages 45–54, 2011.

Daniel Regulin, Christopher Krooß, Sebastian Rehberger, and Birgit Vogel-Heuser. Efficient modeling of mechatronic systems regarding variety and complexity in the field of automotive. In *Conference of the IEEE Industrial Electronics Society, 2013*, 2013.

Rajarishi Sinha, Christiaan JJ Paredis, Vei-Chung Liang, and Pradeep K Khosla. Modeling and simulation methods for design of engineering systems. *J. Comput. Inf. Sci. Eng.*, 1(1):84–91, 2001.

Adrian Wills, Thomas B Schön, Lennart Ljung, and Brett Ninness. Blind identification of wiener models. In *Proceedings of the 18th IFAC World Congress, Milan, Italy*, 2011.

Margaret H Wright. Nelder, mead, and the other simplex method. *Documenta Mathematica*, (7), 2010.