

Combustion Engine Modelling using an Evolving Local Model Network

Christoph Hametner* and Stefan Jakubek†

Christian Doppler Laboratory for Model Based Calibration Methodologies
Vienna University of Technology
1040 Vienna, Austria

*christoph.hametner@tuwien.ac.at

†stefan.jakubek@tuwien.ac.at

Abstract—In this paper a new evolving parameter estimation algorithm for a local model network under special consideration of combustion engine modelling is presented. For practical applications computational speed, incorporation of prior knowledge and the interpretability of the local models is of great interest. Accordingly, a robust and efficient online training algorithm with a particular focus on computational requirements involved in dynamic system identification of complex nonlinear processes is presented. The incremental construction of the model tree allows to gradually increase the model complexity while a proper initialisation of new model parameters is easily possible. The proposed evolving local model network is validated using real measurement data from a state-of-the-art 4-cylinder EURO5 diesel engine.

Index Terms—System identification, local model network, online learning, engine modelling

I. INTRODUCTION

Evolving model structures are becoming a more and more essential factor in the automotive industry. Recent publications have addressed automatic adaptation techniques and evolving schemes for data-based modelling, e. g. [1], [2], [3], [4]. Such models are trained *online* and it is necessary to adjust, refine, extend, and evolve the already trained model when new data are available, [1]. The model is evolved sample by sample and its complexity is gradually increased due to the fact that it is not practicable to completely rebuild the model using all measurement data. Accordingly, new rules are created and an update of the model parameters during the operation of the system is required.

In the context of data-driven modelling, fuzzy systems are an established approach for nonlinear system identification. These so-called *grey-box* models allow to reduce the model complexity, if physical (or other) insight into the nature of the object is available, [5]. In automotive industry, *local model networks* (LMN) are a widely used concept, e. g. [6], [7]. Local model networks offer a versatile structure for the identification of nonlinear static and dynamic systems and provide not only accurate but also *interpretable* models and therefore allow a better understanding of the true system than pure black-box models, e. g. [8], [9], [10].

Especially in engine modelling, evolving model structures are of great interest in order to provide an online model of the engine during testbed runs, see e. g. [11]. Internal combustion (IC) engines are typically calibrated on testbeds which involves

the determination of a number of functional relationships between measured input and output data. In this context, the *design of experiments* (DoE) for testbed runs is closely related to the model adaptation. Generally, *optimal* experiment design is targeted to retrieve the information which is required to properly parameterise a model with as little effort as possible, [12]. Typically, the experiment design is performed offline and prior to the testbed run based on prior knowledge about the process under consideration. The availability of an evolving model allows to optimise the experiment design online (model based) during the testbed run. Accordingly, *model based DoE* employs an intermediate model in combination with a statistical criterion to design optimal experiments, e. g. [13]. Model based DoE can also be used to take into account local process limits in terms of engine protection and combustion stability (i. e. the driveable area of the engine) which are not known in advance. In such cases a model which has been trained online allows to predict and avoid the violation of certain operational limits.

The main contribution of the present paper is a new evolving optimisation algorithm for a local model network that can build the model incrementally whenever new data are available. Important requirements for the practical applicability in engine modelling are low computational effort, proper initialisation, and robustness of the optimisation algorithm. The proposed online training algorithm is based on the local model network presented in [14]. Here, the global process description is obtained by recursively bisecting the input space. The bisection is carried out in such a way that the nonlinear process can reasonably accurately be described by the local models in the resulting subdomains. The use of an incremental tree has proven to be an effective model building strategy in both offline and online modelling, e. g. [15], [16], [17]. An important advantage of these hierarchically constructed models is their robustness and training speed.

The proposed algorithm is not restricted to axis orthogonal splits and offers a higher flexibility compared to e. g. [15], [6]. Especially in online system identification, where the operating range of a process (e. g. driveable area of the engine) is generally not known a priori, arbitrary (axis oblique) splits are required for an optimal partitioning of the input space. Thereby, the incremental construction of the model tree allows

to gradually increase the model complexity while a proper initialisation of new model parameters is easily possible. A difficult part in online learning of incremental trees is the decision when to grow the tree, see [16]. In this paper a splitting rule based on statistical measures is proposed.

The remainder of this paper is structured as follows: In section II the architecture of the local model network is shortly reviewed. In section III the online training algorithm is presented. In section IV a practical application of *dynamic* online system identification using real measurement data from a state-of-the-art diesel engine is presented.

II. LOCAL MODEL NETWORK ARCHITECTURE

Local model networks interpolate between different local models, each valid in a certain region of a input space. Each local model - indicated by subscript i - consists of two parts: The *validity function* $\Phi_i(\tilde{\mathbf{x}}(k))$ and its *model parameter vector* θ_i . Thereby, Φ_i defines the region of validity of the i -th local model.

The *local* estimate for the output is obtained by

$$\hat{y}_i(k) = \mathbf{x}^T(k)\theta_i, \quad (1)$$

where $\mathbf{x}^T(k)$ denotes the regressor vector. In dynamic system identification, the regressor vector $\mathbf{x}(k)$ comprises past system inputs and outputs. For static models, the regressor vector contains arbitrary functions (e.g. polynomials) of the input signals.

All local estimations $\hat{y}_i(k)$ are used to form the global model output $\hat{y}(k)$ by weighted aggregation

$$\hat{y}(k) = \sum_{i=1}^M \Phi_i(k)\hat{y}_i(k), \quad (2)$$

where

$$\Phi_i(k) = \Phi_i(\tilde{\mathbf{x}}(k)) \quad (3)$$

and M denotes the number of local linear models. Thereby, the elements in $\tilde{\mathbf{x}}(k)$ span the so-called partition space and are chosen on the basis of prior knowledge about the process and the expected structure of its nonlinearities. Especially in dynamic identification, the dimension of the partition space and furthermore the complexity of the optimisation problem can thus be reduced dramatically when prior knowledge about the nonlinear behaviour of the process is available, see also [6].

The computation of the validity functions Φ_i is based on a logistic discriminant tree. Exemplarily, a hierarchical model tree with three local models is depicted in Fig. 1. Each node corresponds to a split of the partition space into two parts and the free ends of the branches represent the actual local models with their parameter vector θ_i and their validity functions Φ_i . The overall nonlinear model thus comprises M local models and $M - 1$ nodes which determine their regions of validity.

Generally, for the representation of the discriminant function in the d -th node a logistic sigmoid activation function is chosen, c. f. [18]:

$$\varphi_d(\tilde{\mathbf{x}}(k)) = \frac{1}{1 + \exp(-a_d(\tilde{\mathbf{x}}(k)))} \quad (4)$$

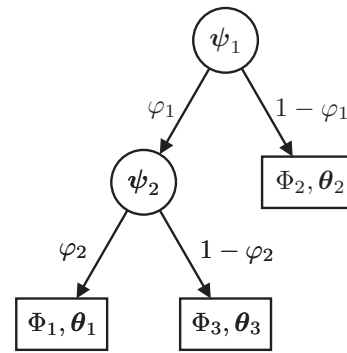


Fig. 1. Logistic discriminant tree

with

$$a_d(\tilde{\mathbf{x}}(k)) = [1 \quad \tilde{\mathbf{x}}^T(k)] \begin{bmatrix} \psi_{d0} \\ \tilde{\psi}_d \end{bmatrix}. \quad (5)$$

Here, $\tilde{\psi}_d^T = [\psi_{d1} \dots \psi_{dp}]$ denotes the weight vector and ψ_{d0} is called bias term. The discriminant functions φ_d are used to calculate the validity functions Φ_i , c. f. [19].

The validity functions for the layout in Fig. 1 are obtained by

$$\Phi_1 = \varphi_1\varphi_2, \quad (6)$$

$$\Phi_2 = 1 - \varphi_1, \quad (7)$$

$$\Phi_3 = \varphi_1(1 - \varphi_2). \quad (8)$$

A main advantage of this hierarchical discriminant tree is that a partition of unity throughout the partition space is guaranteed and no normalisation side effects like reactivations can occur.

III. EVOLVING LOCAL MODEL NETWORK

An incremental model construction allows to gradually increase the complexity of the local model network: When the number of local models M is increased by one, the *worst* local model (indexed by l) of the logistic discriminant tree in Fig. 1 is replaced by a new node and two adjoining local models are appended, see Fig. 2. On the one hand this strategy allows a proper initialisation of the new model parameters while on the other hand the computational demand is low.

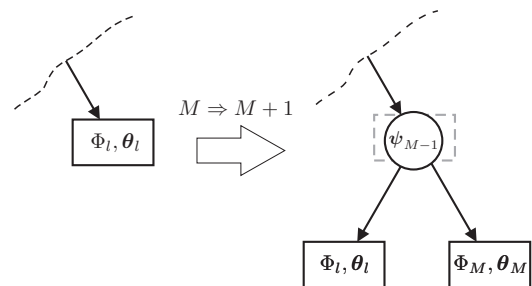


Fig. 2. Incremental tree construction

In order to reduce the computational demand, only the weight vector of the new node ψ_{M-1} is optimised while

all other weight vectors are retained. A disadvantage of this strategy is that the computation of the weight vector ψ_{M-1} and the consequent parameters substantially depends on previous splits. Due to the fact that each split is optimised subject to previous splits, a *global* optimum can generally not be guaranteed. However, a global (simultaneous) optimisation of *all* weight vectors and local model parameters of the LMN results in a much higher computational effort which is not desired and inapplicable for online engine modelling.

A flowchart of the *online* training algorithm is depicted in Fig. 3. The online identification task comprises four subtasks:

- *Build initial model*: The construction of the initial model is based on an *initial data block* using the batch training algorithm, c. f. [14].
- *Split optimisation and parameter update* (see section III-A): Optimisation and update of the actual node (discriminant function). Update of the consequent parameters of each local model (LM) using the new data sample.
- *Model extension* (see section III-B): Statistical assessment of the actual LMN in order to decide whether to extend the model tree.
- *Initialisation of new LMs* (see section III-C): Determine a proper initialisation of new parameters.

A. Split Optimisation and Parameter Update

In the online training, the *actual* number of data - indicated by N - is increased by one in each loop. The online adaptation of the local model network can be divided into two subtasks:

- 1) *Partitioning process*: Find an optimal weight vector $\psi_{M-1}^T = [\psi_{M-1,0} \quad \tilde{\psi}_{M-1}^T]$ for the new node.
- 2) *Local model parameter estimation*: Update of the consequent parameters of *all* local models.

In this section, the optimisation of the actual (most recent) split and the parameter update of all local model parameters is presented.

1) *Partitioning Process*: Due to the incremental model construction, only the weight vector of the actual node ψ_{M-1} is optimised while all other weight vectors are retained. The optimisation of the weight vector is based on the cost function

$$\begin{aligned} J &= \frac{1}{2} \sum_{k=1}^N \Phi_l(k) [y(k) - \hat{y}_l(k)]^2 + \\ &\quad + \Phi_M(k) [y(k) - \hat{y}_M(k)]^2 \\ &= \frac{1}{2} \sum_{k=1}^N e_l^2(k) + e_M^2(k), \end{aligned} \quad (9)$$

where

$$e_l(k) = \sqrt{\Phi_l(k)} [y(k) - \hat{y}_l(k)] \quad (10)$$

and

$$e_M(k) = \sqrt{\Phi_M(k)} [y(k) - \hat{y}_M(k)] \quad (11)$$

are the *weighted* local prediction errors of the adjoining models l and M , respectively.

Obviously, the weight vector ψ_{M-1} and the two adjoining models with their local model parameters θ_l, θ_M have to be optimised simultaneously, see also Fig. 1.

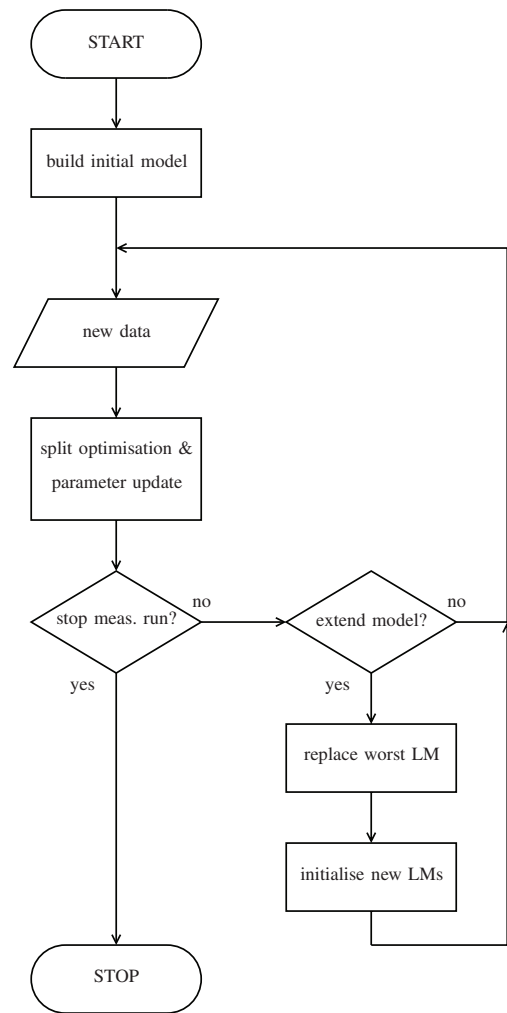


Fig. 3. Flowchart diagram for the online training

With $\Phi_l = \Phi_l(\psi_{M-1})$ and $\Phi_M = \Phi_M(\psi_{M-1})$ a direct solution for the optimisation problem with respect to θ_l, θ_M and ψ_{M-1} is not possible because the computation of the consequent parameters θ_l and θ_M depends on the weight vector ψ_{M-1} and vice versa. Consequently, a *nonlinear least squares* method is applied for the minimisation of (9). A recursive formulation of the nonlinear least squares algorithm, in order to reduce the computational demand, can be found in e. g. [20].

Another important issue in fuzzy modelling is the interpretability of the local models, [21], [10]. Only consequent parameters obtained by local estimation (weighted least squares) allow the interpretation of the consequent parameters as a local linearisation of the nonlinear system, [9]. The particular choice of (9) aims at a *locally* weighted least squares optimisation of the consequent parameters so that local interpretability is conserved.

2) *Parameter Update*: While the update of the consequent parameters θ_l and θ_M is based on a nonlinear least squares algorithm, the online training of the local model network also

requires the update of all other local model parameters when a new observation is available. Due to the fact that the validity functions, i.e. the weight vectors $\psi_j, j = 1, \dots, M - 2$, of these models are not modified during the split optimisation, these parameters can be updated using the conventional weighted least squares method. A recursive formulation of the weighted least squares algorithm is easily possible which helps to reduce the computational demand, see e.g. [22].

B. Model Extension

In online learning the number of local models strictly depends on the currently available data. Too many local models can result in overfitting or even loss of identifiability while a simplistic model yields an inaccurate description of the process. In this section a statistically motivated splitting rule is proposed in order to automatically detect when an extension of the model structure is reasonable.

The splitting rule is based on the assumption that the residual distribution of each local linear model follows a χ^2 distribution for Gaussian white noise and a proper choice of the model structure, i.e. the number of local models.

Using the χ^2 distribution the confidence interval for the variance of the model error of the i -th local model $\sigma_{e_i}^2$ is obtained as

$$\frac{DOF_i}{\chi_{1-\alpha/2, DOF_i}^2} s_i^2 \leq \sigma_{e_i}^2 \leq \frac{DOF_i}{\chi_{\alpha/2, DOF_i}^2} s_i^2, \quad (12)$$

with the error probability α and the remaining statistical degrees of freedom DOF_i . In (12) s_i^2 is the estimated variance of the model error based on the currently available data and $\sigma_{e_i}^2$ the *unknown* true variance.

In the online training the estimated variance and the degrees of freedom change over time as the number of data increases, i.e. in each iteration step new data (a new observation) are available. The χ^2 distribution is used to determine if a local model of the updated LMN performs significantly worse using the additional data sample. The hypothesis that the current local model structure properly describes the process in its region of validity is accepted with an error probability α , if the estimated variance does not exceed the corresponding confidence interval. Otherwise the hypothesis is rejected and the corresponding local model is replaced by two new local models.

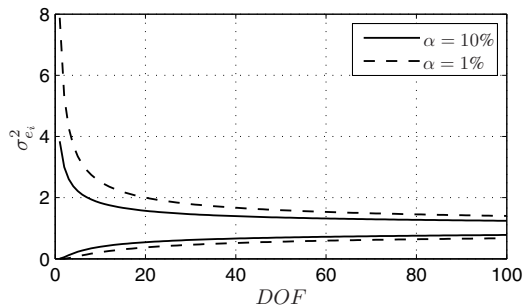


Fig. 4. Upper and lower bounds of the χ^2 distribution

In Fig. 4 the upper and lower bounds of the χ^2 distribution are depicted for $\alpha = 1\%$ and $\alpha = 10\%$, respectively. In the χ^2 distribution the probability α is a good means to control regularisation. Accordingly, a small error probability results in a small number of local models while the robustness of the parameter estimation process is increased. An increased error probability will in turn yield a more accurate description of the process with a higher number of local models while the robustness is decreased (overfitting).

C. Initialisation of new Local Models

The robustness of the training algorithm is closely related to the initialisation of new parameters when the model complexity is increased. Thus the initialisation of new local models, i.e. the consequent parameters θ_l and θ_M as well as the weight vector of the most recent split ψ_{M-1} , must not depend on a random variables.

The incremental model construction allows a proper initialisation of these parameters: The weight vector ψ_{M-1} of new node is chosen based on the weighted mean and the weighted covariance matrix of the observations with respect to the worst local model. The initialisation of the local model parameters θ_l and θ_M is then based on the conventional weighted least squares algorithm.

IV. EXAMPLE

In order to demonstrate the capabilities of the proposed evolving model tree construction scheme the online training of a complex *dynamic* nonlinear local model network using real measurement data from a state-of-the-art 4-cylinder EURO5 diesel engine is presented.

The inputs into the system are *engine speed* N [rpm] and *main injection mass* Q [mg/stroke]. Exemplarily, the *turbine speed* N_{Turbo} [rpm] is considered as the system output $y(k)$. Accordingly, the input-output dynamics are described by the nonlinear difference equation

$$N_{Turbo}(k) = f(x(k))$$

where the dynamic regressor vector in (1) is made up according to

$$x(k) = \begin{bmatrix} 1 \\ Q(k-1) \\ \vdots \\ Q(k-m) \\ N(k-1) \\ \vdots \\ N(k-m) \\ N_{Turbo}(k-1) \\ \vdots \\ N_{Turbo}(k-m) \end{bmatrix}.$$

The dynamic system order for the local models is chosen as $m = 3$.

The dynamic input and output data are depicted in Fig. 5. All in all 5000 data samples are available for the training of the local model network. Here, amplitude modulated pseudo random binary (APRB) signals are used for the input signals in order to ensure proper excitation of the system, see e.g. [6],

[23]. The bandwidth of the input signals is defined by their minimum and maximum hold times which are chosen in consideration of the expected system dynamics.

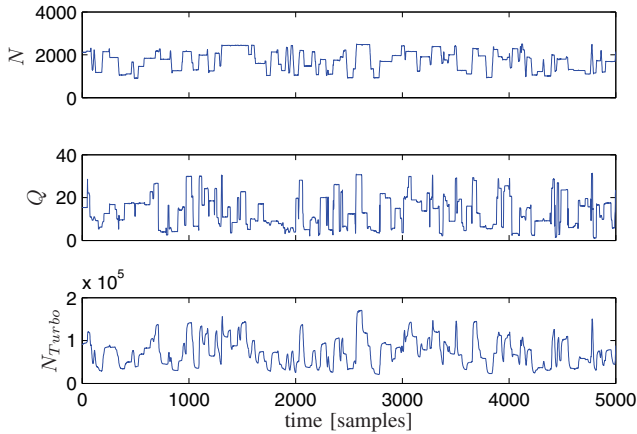


Fig. 5. Measured input and output data

The *initial* local model network with two local models is constructed using the first 1000 samples, see Fig. 6. The partition space is spanned by $N(k-1)$ and $Q(k-1)$ which are the main influence parameters on the highly nonlinear behaviour of the engine. In Fig. 6 a contour plot of the membership functions indicates the region of validity of the two local models. The trajectory of the initial data record is depicted in grey.

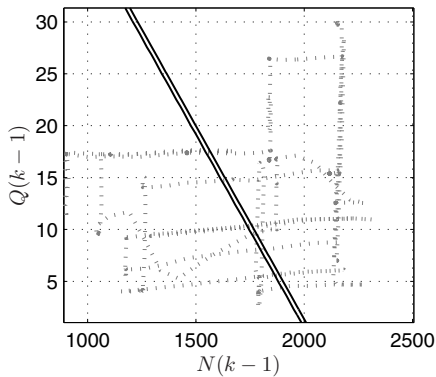


Fig. 6. Initial engine model with 1000 samples

In Fig. 7 the evolvement of the first two local models during the online training procedure is illustrated, c.f. Fig. 6. The validity functions (black) are continuously adapted to the growing data base using the online optimisation algorithm from section III-A. The lower plot in Fig. 7 illustrates how the estimated variances $\sigma_{e_i}^2$ of the two local models evolve as the number of data and the operating range of the LMN increase. Obviously, the estimated variance of the second local model exceeds the upper bound of the χ^2 distribution at some stage. According to section III-B, the second local model is then replaced by two new models in the next iteration step. Their

validity functions are indicated by the dashed black contour lines.

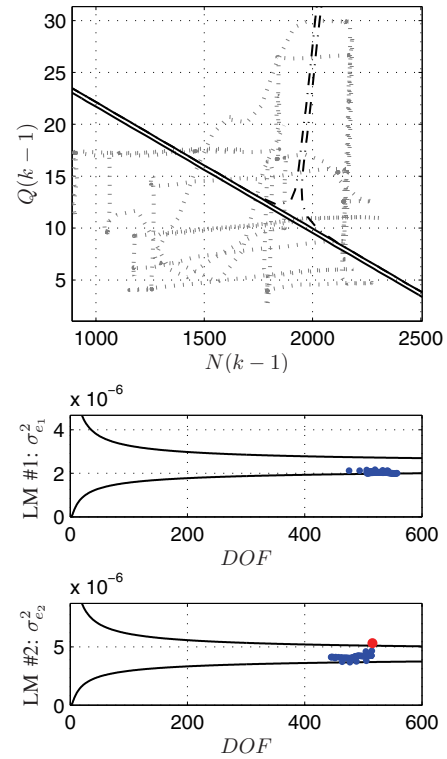


Fig. 7. Online training: Validity functions and χ^2 criterion

The sample-wise online training of the evolving local model tree using the χ^2 splitting rule from section III-B with $\alpha = 1\%$ results in a local model network with seven local models. The final model with the shape and extent of each validity function (black) as well as the training data record (grey) are shown in Fig. 8.

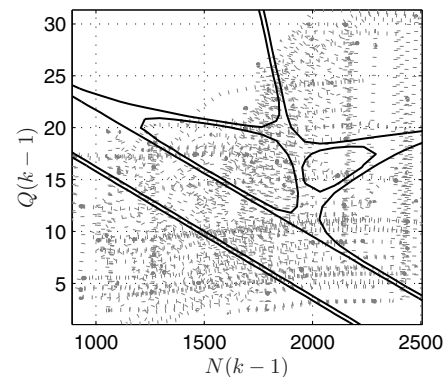


Fig. 8. Final model with 5000 samples and 7 local models

The performance of the local model network obtained from online training is highlighted using a dynamic generalisation data record. In Fig. 9 a section of the measured output and the simulated model output are compared. It is clearly visible that

the LMN accurately describes the nonlinear behaviour of the engine throughout the operating range of the engine.

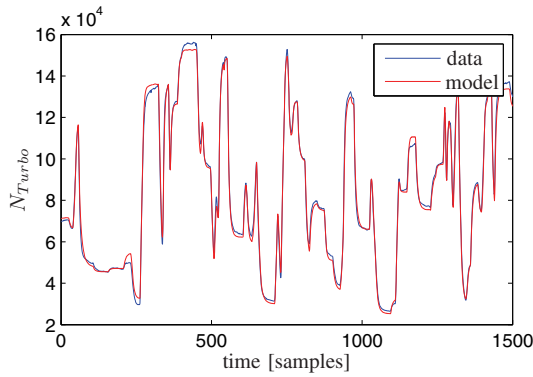


Fig. 9. Simulated model output using generalisation data

In table I the performance of the models obtained from the evolving online training and the classic batch algorithm is indicated by means of the R^2 statistics

$$R^2 = 1 - \frac{\sum_{k=1}^N (y(k) - \hat{y}(k))^2}{\sum_{k=1}^N (y(k) - \bar{y})^2}, \quad (13)$$

where $\bar{y}$ is the mean value of all $y(k)$.

TABLE I
COMPARISON OF ONLINE AND OFFLINE TRAINING

	online training	offline training
R^2	0.99277	0.99353

V. CONCLUSION AND OUTLOOK

In this paper an evolving local model network concept especially suited for engine modelling is presented. Especially in dynamic system identification the computational speed is an important requirement for online training. The incremental model construction allows a proper initialisation of the model parameters in order to increase the robustness while the computational demand is low. A new evolving optimisation algorithm for the online training of the local model network is proposed where the particular choice of the cost function for the split optimisation aims at obtaining locally interpretable models.

A statistically motivated splitting rule is proposed in order to automatically detect when an extension of the model structure is reasonable.

The practical applicability of the evolving local model network is validated by means of real dynamic measurement data from a state-of-the-art 4-cylinder diesel engine.

ACKNOWLEDGEMENTS

This work was supported by the Christian Doppler Research Association and AVL List GmbH, Graz.

REFERENCES

- [1] E. Lughofer, "FLEXFIS: A Robust Incremental Learning Approach for Evolving Takagi-Sugeno Fuzzy Models," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 6, pp. 1393–1410, dec. 2008.
- [2] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484–498, feb. 2004.
- [3] N. Kasabov and Q. Song, "Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *Fuzzy Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 144–154, apr 2002.
- [4] P. Angelov and X. Zhou, "Evolving fuzzy systems from data streams in real-time," in *Evolving Fuzzy Systems, 2006 International Symposium on*, sept. 2006, pp. 29–35.
- [5] L. Ljung, "Perspectives on system identification," *Proceedings of the 17th IFAC World Congress*, 2008.
- [6] O. Nelles, *Nonlinear System Identification*, 1st ed. Springer Verlag, 2002.
- [7] M. Hafner, M. Schler, O. Nelles, and R. Isermann, "Fast neural networks for diesel engine control design," *Control Engineering Practice*, vol. 8, no. 11, pp. 1211–1221, 2000.
- [8] R. Murray-Smith and T. A. Johansen, *Multiple Model Approaches to Modelling and Control*. Taylor & Francis, 1997.
- [9] J. Abonyi and R. Babuska, "Local and global identification and interpretation of parameters in takagi-sugeno fuzzy models," in *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, vol. 2, 2000, pp. 835–840 vol.2.
- [10] T. A. Johansen, R. Shorten, and R. Murray-Smith, "On the Interpretation and Identification of Dynamic Takagi-Sugeno Fuzzy Models," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 3, pp. 297–313, 2000.
- [11] P. Angelov, V. Giglio, C. Guardiola, E. Lughofer, and J. M. Lujan, "An approach to model-based fault detection in industrial measurement systems with application to engine test benches," *Measurement Science and Technology*, vol. 17, no. 7, p. 1809, 2006. [Online]. Available: <http://stacks.iop.org/0957-0233/17/i=7/a=020>
- [12] L. Pronzato, "Optimal experimental design and some related control problems," *Automatica*, vol. 44, no. 2, pp. 303–325, 2008.
- [13] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chemical Engineering Science*, vol. 63, no. 19, pp. 4846–4872, 2008.
- [14] C. Hametner and S. Jakubek, "Neuro-fuzzy modelling using a logistic discriminant tree," *American Control Conference, 2007. ACC '07*, pp. 864–869, July 2007.
- [15] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.
- [16] D. Potts and C. Sammut, "Online nonlinear system identification using linear model trees," in *Proceedings of the 16th IFAC World Congress*, 2005.
- [17] O. Nelles, "Local linear model trees for on-line identification of time-variant nonlinear dynamic systems," in *Artificial Neural Networks ICANN 96*, ser. Lecture Notes in Computer Science, C. von der Malsburg, W. von Seelen, J. Vorbruggen, and B. Sendhoff, Eds. Springer Berlin / Heidelberg, 1996, vol. 1112, pp. 115–120.
- [18] C. M. Bishop, *Neural networks for pattern recognition*. USA: Oxford University Press, 1995.
- [19] P. Pucar and M. Millnert, "Smooth hinging hyperplanes - an alternative to neural networks," in *Proceedings of the 3rd ECC*, 1995.
- [20] L. Ngia, J. Sjöberg, and M. Viberg, "Adaptive neural nets filter using a recursive levenberg-marquardt search direction," in *Signals, Systems Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, vol. 1, Nov. 1998, pp. 697–701 vol.1.
- [21] R. Babuska, *Fuzzy Modeling for Control*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.
- [22] N. R. Draper and H. Smith, *Applied Regression Analysis*, 3rd ed., ser. Wiley Series in Probability and Statistics. John Wiley & Sons, 1998.
- [23] R. Isermann, *Identifikation dynamischer Systeme 2*, 2nd ed. Springer Verlag, 1992.