Int. J. Appl. Math. Comput. Sci., 2000, Vol.10, No.4, 835–855

COMPARISON OF TWO CONSTRUCTION ALGORITHMS FOR TAKAGI-SUGENO FUZZY MODELS

OLIVER NELLES*, ALEXANDER FINK*
ROBERT BABUŠKA**, MAGNE SETNES**

This paper compares two different approaches to the construction of Takagi-Sugeno fuzzy models from data. These models approximate nonlinear systems by means of interpolation between local linear models. The main issue in the construction of Takagi-Sugeno models is the decomposition of the operating space into validity regions for the local models. The way this decomposition is done influences the complexity, accuracy and transparency of the obtained model. The first of the presented methods, the local linear model tree (LOLIMOT) algorithm generates incrementally the fuzzy model by axis-orthogonal decomposition of the input space. In the other method, product-space fuzzy clustering (the Gustafson-Kessel algorithm) is used to partition the available data into fuzzy subsets. The fundamental advantages and drawbacks of both the alternative strategies are pointed out. Their properties and real-world applicability are illustrated by building a dynamic model of a truck Diesel engine turbocharger.

Keywords: modeling, identification, Takagi-Sugeno fuzzy models, local linear models, turbocharger

1. Introduction

Nonlinear static and dynamic models are necessary, for instance, in prediction, simulation, model-based control, and fault diagnosis. In most cases, the derivation of such models by first principles (physical, chemical, biological and other laws) is expensive, time-consuming and involves many unknown parameters and heuristics. Hence, methods for data-driven modeling and identification are of great interest. A wide class of nonlinear dynamic processes with p inputs u_i and one output y can be described in the discrete time domain by input-output regression models (Leontaritis and Billings, 1985)

$$y(k) = f\big(\boldsymbol{\xi}(k)\big) \tag{1}$$

* Darmstadt University of Technology, Institute of Automatic Control, Laboratory of Control Systems and Process Automation, Landgraf-Georg-Str. 4, D–64283 Darmstadt, Germany,
e-mail: {ONelles, AFink}@iat.tu-darmstadt.de

** Delft University of Technology, Faculty of Information Technology and Systems, Control Laboratory, PO Box 5031, 2600 GA Delft, The Netherlands,
e-mail: {r.babuska, m.setnes}@its.tudelft.nl

with the regressor defined by

$$\boldsymbol{\xi}(k) = \big[u_1(k-1) \ \ldots \ u_1(k-m_1) \ \ldots \ u_p(k-1) \ \ldots \ u_p(k-m_p) \\ y(k-1) \ \ldots \ y(k-n)\big]^T.$$

Here, k denotes the discrete time and the dynamic order of the system is represented by the maximal lags m_i and n. The unknown function $f(\cdot)$ in (1) can be approximated from measurement data by Takagi-Sugeno fuzzy models. They are briefly introduced in the subsequent section. Sections 3 and 4 discuss two alternative strategies for the identification of Takagi-Sugeno (TS) fuzzy models and point out their fundamental properties. Finally, an application example is given to demonstrate the identification of a truck Diesel engine turbocharger by using the both methods.

2. Takagi-Sugeno Fuzzy Models

In this paper, the unknown function $f(\cdot)$ in (1) is approximated by Takagi-Sugeno type fuzzy models (Takagi and Sugeno, 1985). The rule base comprises M rules of the form

$$R_j : \textbf{If } z_1 \text{ is } A_{j,1} \text{ AND } \ldots \text{ AND } z_{nz} \text{ is } A_{j,nz} \\ \textbf{then } y(k) = w_{j,0} + w_{j,1}x_1 + \cdots + w_{j,nx}x_{nx}, \tag{2}$$

$j = 1 \ldots M$, where $A_{j,i}$ is a fuzzy set defined on the universe of discourse of input i. Both the nz-dimensional vector $\mathbf{z}(k) = [z_1 \ z_2 \ \ldots \ z_{nz}]^T$ in the rule premise and the nx-dimensional vector $\mathbf{x}(k) = [x_1 \ x_2 \ \ldots \ x_{nx}]^T$ in the consequent contain subsets of the elements of $\boldsymbol{\xi}(k)$. The rule consequents represent local linear models (LLMs) which are linear in the parameters $w_{j,i}$. For a dynamic model according to (1), these LLMs are linear difference equations. The additional constants $w_{j,0}$ define the operating points. This type of fuzzy model is a universal approximator of the function $f(\cdot)$ under the condition that the premise input space includes all regressors, i.e., $\mathbf{z}(k) = \boldsymbol{\xi}(k)$.

The output of the Takagi-Sugeno fuzzy system with M rules is computed as

$$y(k) = \sum_{j=1}^{M} (w_{j,0} + w_{j,1}x_1 + \cdots + w_{j,nx}x_{nx}) \, \Phi_j(\mathbf{z}) \tag{3}$$

with the validity functions Φ_j, see Fig. 1. These validity functions are normalized, i.e. $\sum_{j=1}^{M} \Phi_j(\mathbf{z}) = 1$ for all premise inputs $\mathbf{z}$. This normalization is achieved by

$$\Phi_j(\mathbf{z}) = \frac{\mu_j(\mathbf{z})}{\sum_{i=1}^{M} \mu_i(\mathbf{z})}, \tag{4}$$

where $\mu_j(\mathbf{z})$ represent the multi-dimensional premise membership functions (MSFs) of the fuzzy model. Murray-Smith (1994) refers to this type of model as a local model network which interpolates local linear models by overlapping local basis functions.

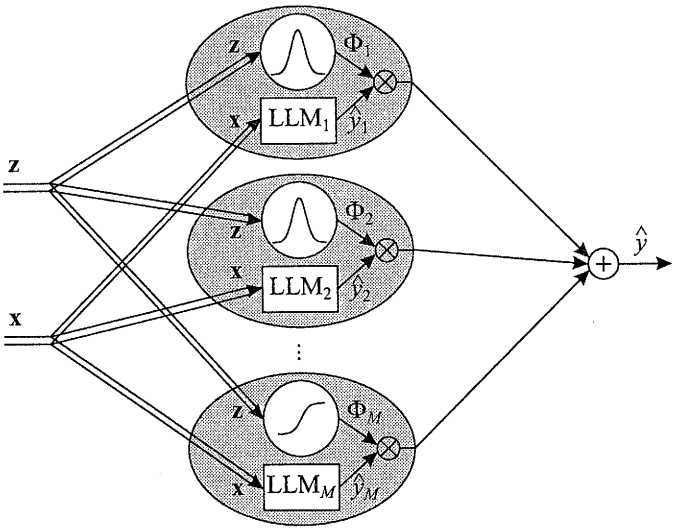


Fig. 1. Architecture of a Takagi-Sugeno fuzzy model with M rules for consequent inputs $\mathbf{x}$ and premise inputs $\mathbf{z}$.

The task of fuzzy model construction is to determine both the nonlinear parameters of the membership functions and the linear parameters of the local models (Johansen and Foss, 1993). In general, there are two ways to obtain this information. Possibly, human experts are able to formulate their process knowledge in fuzzy rules. Unfortunately, this usually delivers only a rough idea of the plant behavior, as humans cannot sense all the details and might not be able to quantitatively express the observations. Therefore, numerous approaches have been proposed which compute nonlinear dynamic fuzzy models from input-output measurement data (Babuška and Verbruggen, 1996).

- *Grid partitioning:* The number of input MSFs per input are typically chosen based on prior knowledge. This approach severely suffers from the curse of dimensionality. To weaken its sensitivity to the input space dimensionality, the grid can be reduced to the regions where enough data are available or a multi-resolution grid can be used (Ishibuchi *et al.*, 1994). All grid-based approaches are restricted to very low-dimensional problems and do not exploit the local complexity of the process.

- *Input space clustering:* The validity functions are placed according to the input data distribution (Strokbro *et al.*, 1990). Since the local process complexity (nonlinearity) is ignored, this simple approach usually does not perform well.

- *Nonlinear local optimization:* Originally, the input MSFs and the rule consequent parameters have been optimized simultaneously. The current state-of-the-art method, however, is to optimize the rule premise parameters by nonlinear local optimization and the rule consequent parameters by global least squares in a nested or staggered approach as in ANFIS (adaptive neuro-fuzzy inference

system) (Jang, 1993). This approach is computationally expensive, but typically yields very accurate results. However, a large number of parameters are optimized and overfitting often becomes a serious problem.

- *Orthogonal Least Squares (OLS):* The OLS algorithm can be used to select important rules. However, severe restrictions apply due to the normalization which changes the regressors during their selection (Kortmann, 1997; Wang, 1994). Because of the normalization, the OLS cannot unfold its full efficiency and thus this approach is computationally demanding. Furthermore, the fuzzy logic interpretation diminishes since the projection to univariate membership functions is not possible.

- *Genetic algorithms:* In order to circumvent the difficulties connected to the OLS algorithm, genetic algorithms can be applied for structure search (Tanaka *et al.*, 1994). Evolutionary algorithms offer a wide spectrum of different approaches. All of them, however, suffer from relatively slow convergence.

- *Heuristic construction algorithms:* This widely applied class of algorithms increases the complexity of the local linear neuro-fuzzy model during training. They start with a coarse partitioning of the input space (typically with a single rule, i.e., one global linear model) and refine the model by increasing the resolution of the input space partitioning. These approaches can be distinguished for very flexible strategies which allow for an (almost) arbitrary partitioning of the input space (Murray-Smith, 1994) or slightly less flexible axis-oblique decomposition strategies (Ernst, 1998) on one hand and the axis-orthogonal strategies which restrict the search to rectangular shapes, see (Nelles, 1999) and Section 3, on the other hand.

- *Product space clustering:* One of the most popular approaches applies the Gustafson-Kessel clustering algorithm to find hyperplanes in the product space. It is (initially) assumed that the rule premise and consequent spaces are equivalent ($\mathbf{x} = \mathbf{z}$) and hyperplanes are sought in the space spanned by $[x_1\ x_2\ \cdots\ x_{nx}\ y]$, see (Babuška, 1998) and Section 4.

The remainder of this paper compares an axis-orthogonal tree construction and a product space clustering strategy.

3. Local Linear Model Tree (LOLIMOT)

The local linear model tree (LOLIMOT) algorithm proposed in (Nelles and Isermann, 1996) utilizes Gaussian membership functions

$$\mu_j(\mathbf{z}) = \exp\left(-\frac{1}{2}\frac{(z_1 - c_{j,1})^2}{\sigma_{j,1}^2}\right) \exp\left(-\frac{1}{2}\frac{(z_2 - c_{j,2})^2}{\sigma_{j,2}^2}\right) \cdots \exp\left(-\frac{1}{2}\frac{(z_{nz} - c_{j,nz})^2}{\sigma_{j,nz}^2}\right), \tag{5}$$

where $c_{j,l}$ denote the centers and $\sigma_{j,l}$ stand for the standard deviations in dimension l for the membership function associated with rule j.

LOLIMOT splits up the identification procedure into two parts. In an outer loop of the algorithm, the premise structure and the corresponding membership functions are determined by a tree construction algorithm. In a nested inner loop, the consequent parameters of the rules are optimized.

3.1. Rule Premise Construction

The input space is decomposed into hyper-rectangles by axis-orthogonal splits. Each local linear model belongs to one hyper-rectangle in the center of which the basis function is placed. The standard deviations are chosen proportional to the size of the hyper-rectangle. This makes the size of the validity region of each local linear model proportional to its hyper-rectangle extension. At each iteration, the local linear model i with the worst local error measure

$$I_i = \sum_{j=1}^{N} \Phi_i(\mathbf{z}(j))\, e^2(j), \quad e(j) = y(j) - \hat{y}(j) \tag{6}$$

is subdivided by splitting it into two halves. Splits in all dimensions are tested and the one with the highest performance improvement is chosen. The LOLIMOT algorithm can be summarized as follows:

1. *Start with an initial model:* Construct the validity functions for the initially given input space partitioning and estimate the LLM parameters by the local weighted least squares algorithm. Set M to the initial number of LLMs. If no input space partitioning is available *a priori*, then set $M = 1$ and start with a single LLM which in fact is a global linear model since its validity function covers the whole input space with $\Phi_1(\mathbf{z}) = 1$.

2. *Find worst LLM:* Calculate a local loss function for each of the LLMs. The local loss functions can be computed by weighting the squared model errors with the degree of validity of the corresponding local model according to (6). Find the worst performing LLM, i.e. $\max_i(I_i)$, and denote by b the corresponding index.

3. *Check all divisions:* The LLM b is considered for further refinement. The hyperrectangle of this LLM is split into two halves with an axis-orthogonal split. Divisions in all dimensions are tried. For each division $\mathrm{dim} = 1, \ldots, nz$ the following steps are carried out:

   (a) Construction of the multi-dimensional membership functions for both hyperrectangles.

   (b) Construction of all validity functions.

   (c) Local estimation of the rule consequent parameters for both the newly generated LLMs.

   (d) Calculation of the loss function for the current overall model.

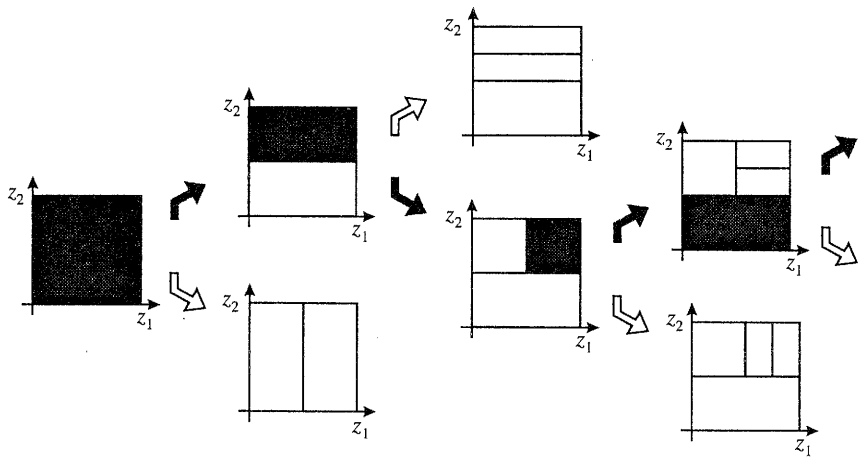


Fig. 2. Tree construction and input space decomposition by LOLIMOT.

4. *Find best division:* The best of the nz alternatives checked in Step 3 is selected. The validity functions constructed in Step 3a and the LLMs optimized in Step 3c are adopted for the model. The number of LLMs is incremented: $M \leftarrow M + 1$.

5. *Test for convergence:* If the termination criterion is met, then stop, otherwise go to Step 2.

For an illustration of the first four iterations of LOLIMOT, refer to Fig. 2.

3.2. Rule Consequent Optimization

In the inner loop, the linear regressors and parameters of the local models are selected and estimated by an orthogonal weighted least squares (OLS) algorithm. The LOLIMOT algorithm owes its high performance to the local estimation of the linear parameters in the rule consequents. When the premise structure has been determined, M linear optimization problems are solved separately. As argued in (Nelles, 1999), the local estimation is superior to the global estimation in many applications for the following reasons:

- *Fast training:* Due to the significantly lower computational complexity of local estimation, training becomes very fast. This advantage increases quadratically with the number of neurons.

- *Regularization effect:* The number of effective parameters with local estimation is decreased. The conditioning, i.e., the eigenvalue spread, of the Hessian of the loss function is smaller and thus the parameter variances are reduced. Consequently, the reduced flexibility of the model results in a higher bias (systematic) error while it decreases the variance (stochastic) error in comparison with global

estimation. Global estimation is more likely to overfit the data due to compensation effects between neighboring local models, whereas local estimation treats the local models separately. These properties of the local approach are advantageous when the available training data are noisy and/or sparsely distributed, which is usually the case for high-dimensional input spaces.

- *Interpretation:* The locally estimated parameters can be individually interpreted as a description of the identified process behavior in the regime represented by the corresponding validity function. The parameters of the local linear models are not sensitive with respect to the overlap of the validity functions. These properties easily allow us to gain insights in the process behavior. Globally estimated parameters can only be interpreted by taking the interaction with the neighboring models into account.

- *Online learning:* The local estimation approach offers considerable advantages when applied in a recursive algorithm for on-line learning. Besides the lower computational complexity and the improved numerical stability due to the better condition of the Hessian, local on-line learning allows us to solve the so-called stability/plasticity dilemma (Carpenter and Grossberg, 1988).

- *Higher flexibility:* The local estimation enables a wide range of optimization approaches that are not feasible with the global estimation approach. For example, some local models may be linearly parameterized while others are nonlinearly parameterized. Then, linear LS estimation can be applied to the former and nonlinear optimization to the latter. Another example is the use of local linear models with different structures of the dynamics. Local estimation allows us to specify and realize individually desired local model complexities. Furthermore, different loss functions can be specified individually for each local model.

The major drawback of local as opposed to global estimation is that it does not achieve the optimal parameters in the least squares sense. This is significant only if a large amount of high quality data are available or models of low complexity are sufficient because then overfitting is no issue.

The non-weighted output of the j-th rule can be written down as

$$y_j = \psi_j^T \mathbf{w}_j \tag{7}$$

with parameter and regression vectors

$$\begin{cases} \mathbf{w}_j = [w_{j,0}\ w_{j,1}\ \dots\ w_{j,nx}]^T, \\ \psi_j = [\psi_1\ \psi_2\ \dots\ \psi_{nx+1}]^T = [1\ x_1\ \dots\ x_{nx}]^T. \end{cases} \tag{8}$$

The optimal parameters $\mathbf{w}_j$ of each local linear model j are estimated from N data samples by local weighted least squares

$$\mathbf{w}_j = \left(\mathbf{\Psi}_j^T \mathbf{Q}_j \mathbf{\Psi}_j\right)^{-1} \mathbf{\Psi}_j^T \mathbf{Q}_j \mathbf{y}_\mathrm{d}, \tag{9}$$

where $\mathbf{\Psi}_j = [\psi_j(1)\ \psi_j(2)\ \dots\ \psi_j(N)]^T$ is the regression matrix and $\mathbf{y}_\mathrm{d} = [y_\mathrm{d}(1)\ y_\mathrm{d}(2)\ \dots\ y_\mathrm{d}(N)]^T$ is the vector of desired model outputs.

The weighting matrix $\mathbf{Q}_j = \text{diag}\{\Phi_j(z(1)), \Phi_j(z(2)), \ldots, \Phi_j(z(N))\}$ contains the values of the validity function which yields the local property of the estimation. Data samples located close to the center of the respective validity function exert stronger influence on the parameter estimates than data points which are far away in the input space. Hence, in contrast to global parameter estimation, it can be guaranteed that the local models represent the local behavior of the nonlinear system and that they do not suffer from compensation effects. One should notice that the overlap of the local models is neglected by the estimation scheme which might deteriorate the model's accuracy if the standard deviation of the Gaussian membership functions is chosen too large.

The main features of LOLIMOT can be summarized as follows:

1. *Gaussian membership functions* are used.
2. The *parameter estimation* is performed *locally*. This yields a regularization effect that reduces the variance error at the price of an increased bias error. The computational complexity grows only linearly with the number of fuzzy rules.
3. The *input space decomposition* is performed in an *axis-orthogonal* manner. This approach is very simple and allows for a computationally effective implementation. However, the restriction to axis-orthogonal splits with the ratio 1:1 leads to suboptimal results and thus LOLIMOT constructs models with relatively many rules.
4. Different *input spaces* for the rule premises and consequents can be used.
5. The computational complexity only grows in a cubic manner with the consequent space dimensionality $\dim(\mathbf{x})$ and linearly with the premise input space dimensionality $\dim(\mathbf{z})$. This overcomes the *curse of dimensionality*.
6. The axis-orthogonal input space partitioning allows for a direct *interpretation* in terms of *fuzzy logic*.
7. It is a *growing* algorithm, i.e., it incrementally increases the number of rules in each iteration.
8. It is not restricted to local *linear* models. Rather, any type of local model can be incorporated and different local model structures can be used in one Takagi-Sugeno fuzzy system.

For more details about the LOLIMOT algorithm refer to (Nelles, 1999).

4. Product Space Clustering

Identification methods based on fuzzy clustering originate from data analysis and pattern recognition, where the concept of graded membership is used to represent the degree to which a given object, represented as a vector of features, is similar to

some prototypical object. The degree of similarity can be calculated using a suitable distance measure. Based on the similarity, feature vectors can be clustered such that the vectors within a cluster are as similar (close) as possible, and vectors from different clusters are as dissimilar as possible.

In system identification, fuzzy clustering can be applied to discover operating regions in which a nonlinear system can be locally approximated by linear submodels (Babuška, 1998). For this purpose, clustering is applied in the product space of the regressors and of the regressand (output). The prototypes are defined as linear subspaces (Bezdek, 1981) or the clusters are ellipsoids with adaptively determined shapes. The latter approach is followed in this paper, using the Gustafson-Kessel (GK) algorithm (Gustafson and Kessel, 1979) which is briefly outlined below.

4.1. The Gustafson-Kessel Algorithm

Clustering is applied to data vectors, each of which consists of n measured variables, grouped into an n-dimensional column vector $\mathbf{z}_k = [z_{1k}, \ldots, z_{nk}]^T$, $\mathbf{z}_k \in \mathbb{R}^n$. A set of N observations is denoted by $\mathbf{Z} = \{\mathbf{z}_k | k = 1, 2, \ldots, N\}$, and is represented as an $n \times N$ matrix:

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nN} \end{bmatrix}. \tag{10}$$

When clustering is applied to the modeling and identification of dynamic systems, the columns of $\mathbf{Z}$ contain samples of time signals, and the rows are typically the input and output variables observed in the system (position, velocity, temperature, etc.). In order to represent the system's dynamics, past values of the variables may be included in $\mathbf{Z}$ as well (see the illustrative example later in this section).

The objective is to partition $\mathbf{Z}$ into K fuzzy subsets (clusters). These subsets are defined by their membership (characteristic) functions, represented in the *partition matrix* $\mathbf{U} = [\mu_{ik}]_{K \times N}$. The i-th row of this matrix contains values of the membership function of the i-th fuzzy subset A_i of $\mathbf{Z}$. The partition matrix satisfies the following conditions:

$$\mu_{ik} \in [0, 1], \qquad 1 \leq i \leq K, \quad 1 \leq k \leq N, \tag{11a}$$

$$\sum_{i=1}^{K} \mu_{ik} = 1, \qquad 1 \leq k \leq N, \tag{11b}$$

$$0 < \sum_{k=1}^{N} \mu_{ik} < N, \qquad 1 \leq i \leq K. \tag{11c}$$

Equation (11a) expresses the well-known fact that the membership degrees are real numbers from the interval $[0, 1]$. Condition (11b) constrains the sum of each column

to 1, and thus the total membership of each $\mathbf{z}_k$ in all clusters equals one. Equation (11c) means that none of the fuzzy subsets is empty nor it contains all the data.

The fuzzy partition matrix is obtained by the GK algorithm, which is based on the minimization of the well-known *fuzzy c-means* functional:

$$J(\mathbf{Z};\mathbf{U},\mathbf{V},\{\mathbf{A}_i\}) = \sum_{i=1}^{K}\sum_{k=1}^{N}(\mu_{ik})^m D^2_{ik\mathbf{A}_i}, \tag{12}$$

where $\mathbf{U} = [\mu_{ik}]$ is the fuzzy partition matrix of $\mathbf{Z}$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$, $\mathbf{v}_i \in \mathbb{R}^n$ is a vector of *cluster prototypes* (centers) which have to be determined, and

$$D^2_{ik\mathbf{A}_i} = (\mathbf{z}_k - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{z}_k - \mathbf{v}_i) \tag{13}$$

is the squared inner-product distance norm. Matrices $\mathbf{A}_i$ are computed in the optimization algorithm using the local covariance of the data around each cluster center. This allows each cluster to adapt the distance norm to the local distribution of the data. If the data samples are distributed along a nonlinear hypersurface, the GK algorithm will find clusters that are local linear approximations of this hypersurface. The overlap of the clusters is controlled by the user-defined parameter $m \in [1, \infty)$.

The minimization of (12) represents a nonlinear optimization problem that is usually solved by using the Picard iteration through the first-order conditions for stationary points of (12). This algorithm is stated (without derivation) in Algorithm 1 below.

The number of clusters, K, the 'fuzziness' exponent, m, and the termination tolerance, ϵ must be specified before clustering. The number of clusters can either be selected *a priori* or automatically determined by iterative merging or insertion of clusters (Kaymak and Babuška, 1995; Krishnapuram and Freg, 1992) or by using cluster validity measures (Bezdek, 1981; Gath and Geva, 1989; Pal and Bezdek, 1995). An example of a validity measure suitable for the construction of TS models is the flatness index (Babuška and Verbruggen, 1995). It is defined as the ratio between the smallest and the largest eigenvalue of the cluster covariance matrix:

$$t_i = \frac{\lambda_{in}}{\lambda_{i1}}. \tag{14}$$

When clustering data which describe a functional relationship, the clusters are flat. Consequently, the smallest eigenvalue λ_{in} of the covariance matrix (see Step 2 in Algorithm 1) is considerably smaller than the remaining eigenvalues. Hence, the index attains low values for clusters which are large and flat. The aggregate measure for the entire partition is given by

$$t_A = \frac{1}{K}\sum_{i=1}^{K}\frac{\lambda_{in}}{\lambda_{i1}}. \tag{15}$$

The prediction error of the model is as follows:

$$e = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2, \tag{16}$$

Algorithm 1. Gustafson-Kessel (GK) algorithm.

Given a data set $\mathbf{Z}$, choose the number of clusters $1 < K < N$, the weighting exponent $m > 1$ and the termination tolerance $\epsilon > 0$. Initialize the partition matrix randomly, such that it satisfies conditions (11).

Repeat for $l = 1, 2, \ldots$

Step 1. Compute cluster prototypes (means):

$$\mathbf{v}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m \mathbf{z}_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \leq i \leq K.$$

Step 2. Compute the cluster covariance matrices:

$$\mathbf{F}_i = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m (\mathbf{z}_k - \mathbf{v}_i^{(l)})(\mathbf{z}_k - \mathbf{v}_i^{(l)})^T}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, \quad 1 \leq i \leq K.$$

Step 3. Compute the distances:

$$D_{ik\mathbf{A}_i}^2 = \left(\mathbf{z}_k - \mathbf{v}_i^{(l)}\right)^T \left[\det(\mathbf{F}_i)^{1/n} \mathbf{F}_i^{-1}\right] \left(\mathbf{z}_k - \mathbf{v}_i^{(l)}\right),$$
$$1 \leq i \leq K, \quad 1 \leq k \leq N.$$

Step 4. Update the partition matrix:

for $1 \leq k \leq N$

if $D_{ik\mathbf{A}_i} > 0$ for $1 \leq i \leq K$,

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{K} (D_{ik\mathbf{A}_i} / D_{jk\mathbf{A}_j})^{2/(m-1)}},$$

otherwise

$$\mu_{ik}^{(l)} = 0 \text{ if } D_{ik\mathbf{A}_i} > 0, \text{ and } \mu_{ik}^{(l)} \in [0, 1] \text{ with } \sum_{i=1}^{K} \mu_{ik}^{(l)} = 1.$$

until $\|\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}\| < \epsilon$.

where y_k and $\hat{y}_k$ are the true data and the predicted output, respectively, and N is the number of data items, one can combine the average flatness index (15) with the prediction error (16) to obtain the validity measure

$$\nu = t\, e, \tag{17}$$

which prefers a few flat clusters to a larger number of small ones, if both the settings lead to approximately the same prediction error. This approach conceptually resembles the use of information criteria in linear system identification (Akaike, 1974).

The weighting exponent m influences the fuzziness of the resulting partition. As m approaches one from above, the partition becomes hard (non-fuzzy) ($\mu_{ik} \in \{0, 1\}$) and as $m \to \infty$, the partition becomes completely fuzzy ($\mu_{ik} = 1/K$). The standard value $m = 2$ is used in this paper.

The GK algorithm stops iterating when the norm of the difference between $\mathbf{U}$ in two successive iterations is smaller than the termination parameter ϵ. For the maximum norm $\max_{ik}(|\mu_{ik}^{(l)} - \mu_{ik}^{(l-1)}|)$, the usual choice is $\epsilon = 0.01$.

4.2. Rule Extraction from Clusters

The premise membership functions and the consequent parameters of the Takagi-Sugeno model are computed from the obtained fuzzy partition (Fig. 3).

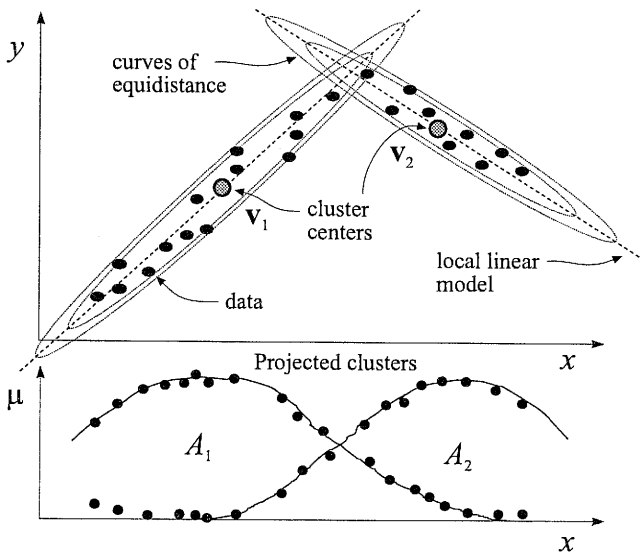


Fig. 3. Hyperellipsoidal fuzzy clusters.

Each obtained cluster results in one rule in the Takagi-Sugeno model. In this example, we obtain the following two rules:

$$\textbf{If } x \text{ is } A_1 \textbf{ then } y = a_1 x + b_1,$$

$$\textbf{If } x \text{ is } A_2 \textbf{ then } y = a_2 x + b_2.$$

The membership functions for the premise fuzzy sets are generated by point-wise projection of the partition matrix onto the premise variables. These point-wise defined fuzzy sets are then approximated by piecewise exponential membership functions:

$$\mu(x;\sigma_1,\alpha_1,\sigma_2,\alpha_2) = \begin{cases} \exp\left(-\left[\dfrac{x(k)-\alpha_1}{2\sigma_1}\right]^2\right) & \text{if } x(k) < \alpha_1, \\ \exp\left(-\left[\dfrac{x(k)-\alpha_2}{2\sigma_2}\right]^2\right) & \text{if } x(k) > \alpha_2, \\ 1 & \text{otherwise.} \end{cases} \tag{18}$$

The fit is obtained by numerically optimizing the parameters σ_1, α_1, σ_2 and α_2. The degree of fulfilment $\mu_j(x)$ of the premise is computed as the product of the individual membership functions (18). The consequent parameters for each rule are obtained by solving the least squares problem (9) for each rule.

The principle of identification in the product space directly applies to the dynamic model (1). To see this, consider an illustrative example of system identification from a time series generated by a nonlinear autoregressive system (Ikoma and Hirota, 1993):

$$x(k+1) = f\big(x(k)\big) + \epsilon(k), \quad f(x) = \begin{cases} 2x - 2, & 0.5 \le x, \\ -2x, & -0.5 < x < 0.5, \\ 2x + 2, & x \le -0.5. \end{cases} \tag{19}$$

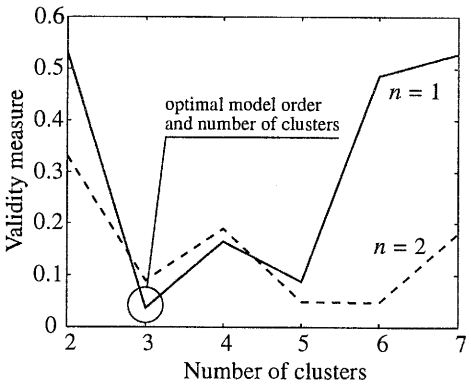
Here, $\epsilon(k)$ is a sequence of i.i.d. $N(0,\sigma^2)$ random variables with $\sigma = 0.3$. From the generated data $x(k)$, $k = 0,\ldots,200$, with an initial condition $x(0) = 0.1$, the first 100 points are used for identification and the rest for model validation. By means of fuzzy clustering, a TS model will be obtained. It is assumed that the only prior knowledge is that the data were generated by a nonlinear autoregressive system:

$$x(k+1) = f\big(x(k), x(k-1), \ldots, x(k-n+1)\big), \tag{20}$$

where n is the system order. Here, $\boldsymbol{\xi}(k) = [x(k), x(k-1), \ldots, x(k-n+1)]^T$ is the regression vector and $x(k+1)$ is the response variable. The matrix $\mathbf{Z}$ to be clustered is constructed from the identification data as follows:

$$\mathbf{Z} = \begin{bmatrix} x(n) & x(n+1) & \cdots & x(N-1) \\ \cdots & \cdots & \cdots & \cdots \\ x(1) & x(2) & \cdots & x(N-n) \\ x(n+1) & x(n+2) & \cdots & x(N) \end{bmatrix}. \tag{21}$$

To identify the system, we need to find the order n and to approximate the function f by a TS affine model. The order of the system and the number of clusters are determined by computing the cluster validity measure (17) for a range of model orders $n = 1, 2 \ldots, 5$ and the number of clusters $K = 2, 3 \ldots, 7$. The results are shown in matrix form in Fig. 4(b). The optimum (printed in boldface) was obtained for $n = 1$ and $K = 3$, which corresponds to (19). In Fig. 4(a), ν is also plotted as a function of K for orders $n = 1, 2$. Note that this function may have several local minima, of which the first is usually chosen in order to obtain a simple model with few rules.



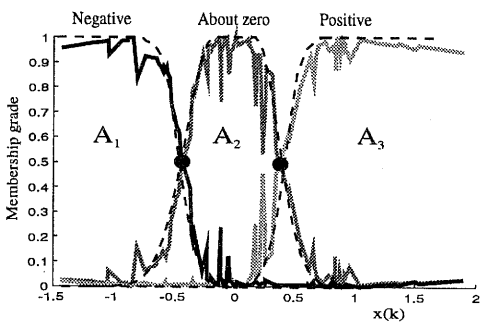
(a)

number of clusters ν	model order 1	2	3	4	5
2	0.53	0.33	0.50	4.64	1.27
3	**0.03**	0.08	0.07	0.21	2.45
4	0.16	0.19	5.62	0.36	1.60
5	0.08	0.04	0.06	0.18	0.27
6	0.48	0.04	0.43	0.22	0.51
7	0.52	0.18	2.07	0.12	0.13

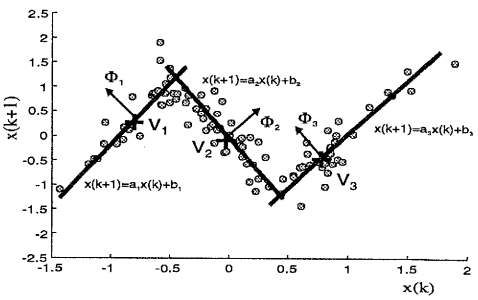
(b)

Fig. 4. The validity measure for different model orders and different number of clusters.

Figure 5(a) shows the projection of the obtained clusters onto the variable $x(k)$ for the correct system order $n = 1$ and the number of clusters $K = 3$.



(a) Fuzzy partition projected onto $x(k)$.



(b) Local linear models extracted from the clusters.

Fig. 5. Results of fuzzy clustering for $n = 1$ and $K = 3$. Part (a) shows the membership functions obtained by projecting the partition matrix onto $x(k)$. Part (b) gives the cluster prototypes v_i, the orientation of the eigenvectors Φ_i and the direction of the affine consequent models (lines).

By the least squares we derive the parameters a_i and b_i of the TS model shown below. After labeling these fuzzy sets as NEGATIVE, ABOUT ZERO and POSITIVE (cf. Fig. 5(a)), the obtained TS models can be written as

$$\begin{aligned}
&\textbf{If } x(k) \text{ is NEGATIVE} && \textbf{then } x(k+1) = 2.371x(k) + 1.237,\\
&\textbf{If } x(k) \text{ is ABOUT ZERO} && \textbf{then } x(k+1) = -2.109x(k) + 0.057,\\
&\textbf{If } x(k) \text{ is POSITIVE} && \textbf{then } x(k+1) = 2.267x(k) - 2.112.
\end{aligned}$$

The estimated consequent parameters correspond approximately to the definition of the line segments in the deterministic part of (19). Also the partition of the premise domain is in agreement with the definition of the system.

It should be noted that the transparency of the model obtained in the above way may be hampered by redundancy present in the form of many overlapping (compatible) membership functions (due to the projection). Similarity measures are used in order to assess the compatibility (pair-wise similarity) of fuzzy sets in the rule base, in order to identify sets that can be merged. Fuzzy sets estimated from data can also be similar to the universal set, thus adding no information to the model. Such sets can be removed from the premise of the rules. These operations reduce the number of fuzzy sets in the model. Reduction of the rule base follows when the premises of some rules become equal. Such rules are combined into one rule. The compatibility between the fuzzy sets A_{lj} and A_{mj} in the rules R_l and R_m, respectively, is assessed by the following fuzzy analog of the Jaccard index (Dubois and Prade, 1980):

$$c_{jlm} = \frac{|A_{lj} \cap A_{mj}|}{|A_{lj} \cup A_{mj}|}, \tag{22}$$

where $l, m = 1, 2, \ldots, L$, and $c_{jlm} \in [0, 1]$. The $\cap$ and $\cup$ operators are the intersection and the union, respectively, and $|\cdot|$ denotes the cardinality of a fuzzy set. The measure c_{jlm} is computed for discretized domains.

The main features of the clustering-based approach can be summarized as follows:

1. The decomposition of the premise space is performed in the product space of the regressors. This is a more general and powerful way than the axis-orthogonal decomposition of LOLIMOT and related methods. The latter techniques give suboptimal results, since a greedy strategy is used for the selection of variables to split and also the position of the split is fixed or sought in an suboptimal way.

2. Initially, the same variables must be used for both the premises and the consequents. In the second step, however, subsets of these variables may be obtained as a result of the simplification.

3. The computational complexity grows in a cubic manner with the sum of the premise and consequent space dimensions.

4. Similarly to LOLIMOT, a regularization effect is achieved that reduces the variance error at the cost of an increased bias error.

5. As clusters obtained in the product space may be of different shapes and orientations, the interpretation of the premise may be less accurate than with LOLIMOT.

5. Identification of a Turbocharger

This section presents the identification of an exhaust gas turbocharger with the goal of hardware-in-the-loop simulation of Diesel engines. Figure 6(a) schematically represents the charging process of a Diesel engine by an exhaust gas turbocharger. The charging process has nonlinear input-output behavior and it is characterized by a strong dependence of the dynamic parameters on the operating point. This is known by physical insights.

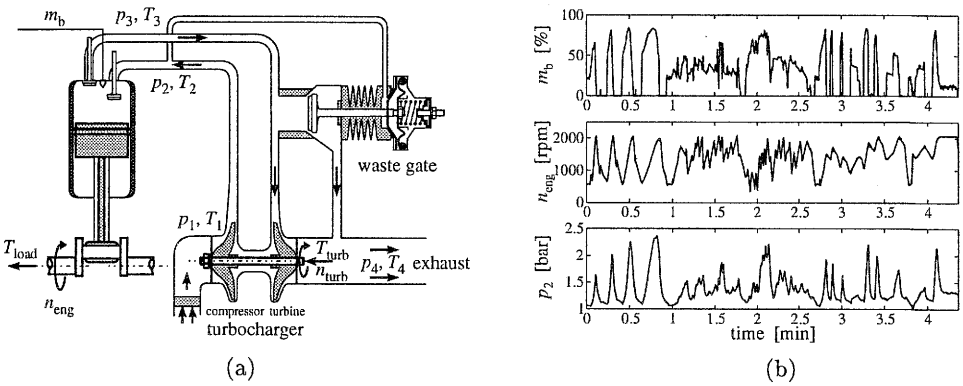


Fig. 6. (a) Scheme of the combustion engine turbocharger. (b) Training data.

In general, the static behavior of the turbocharger can be described sufficiently well by characteristic maps (look-up tables) of the compressor and the turbine. However, if the dynamics of the turbocharger need to be considered, basic mechanical modeling of the thermodynamics is required, see (Zinner, 1985). The quality of theoretical models, however, essentially depends on the accurate knowledge of several process parameters which have to be laboriously derived or estimated in most cases by analogy considerations. Another disadvantage is the considerable computational effort due to the complexity of those methods.

For these reasons, such methods are considered to be inconsistent with the requirement of typical control engineering applications such as controller design, fault diagnosis and hardware-in-the-loop simulations. Therefore, the two alternative strategies for building Takagi-Sugeno fuzzy models, which comply with the demanded properties to a high degree, are realized in the following. Only the recorded data of input-output measurements of the real process are required. Thus, no deeper theoretical knowledge of the process is necessary. The rate of injection m_b and the engine speed n_{eng} are chosen as inputs while the charging pressure p_2 is the output. The sampling time is $T_0 = 0.2s$.

The training data were generated by a special driving cycle to excite the system with all amplitudes and frequencies of interest, see Fig. 6(b). The measurements were recorded on a flat test track. In order to make the engine operate in high load ranges, the truck was driven with the highest possible load. For validation, driving cycles were recorded, which reproduce realistic conditions in interstate and urban traffic.

It was found by a trial-and-error approach (starting with the first order and increasing the order in each trial) that the turbocharger can be described sufficiently well by assuming a second-order model. Therefore, the charging pressure $p_2(k)$ at time instant k is modeled by the following relationship:

$$p_2(k) = f\Big(m_\mathrm{b}(k),\, m_\mathrm{b}(k-1),\, m_\mathrm{b}(k-2),\, n_\mathrm{eng}(k),\, n_\mathrm{eng}(k-1),$$
$$n_\mathrm{eng}(k-2),\, p_2(k-1),\, p_2(k-2)\Big). \tag{23}$$

First, the LOLIMOT algorithm is used to model the dynamic behavior of the turbocharger. Then, the identification by product space clustering is presented and the results are compared.

Figure 7(a) shows the convergence curve of the LOLIMOT identification algorithm. The fuzzy model containing $M = 10$ local linear models is chosen because it describes the process behavior accurately enough and more complex models do not yield further significant improvements. From prior experiments, it has been found out that it is sufficient to partition the premise input space only in the inputs $m_\mathrm{b}(k-1)$ and $n_\mathrm{eng}(k-1)$ while all regressors in (23) should be used in the consequents. This is an example of a reduced, low-order premise input space as discussed in Section 3, which reduces the complexity of the identification problem. The input space decomposition of the model with 10 rules is depicted in Fig. 7(b), which can be easily visualized since it is only two-dimensional. It should be noted that the input space is not partitioned in the output p_2, and consequently the process behavior is linear in its output.

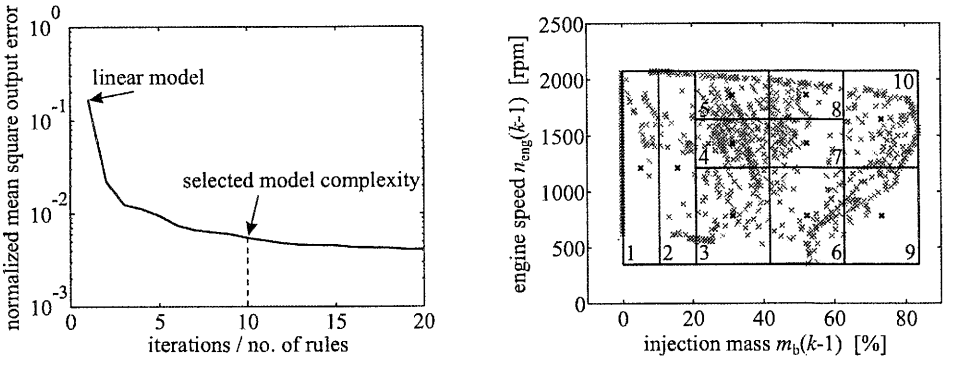


Fig. 7. (a) Convergence curve of the LOLIMOT algorithm. (b) Premise space decomposition and distribution of the training data.

The identification results are shown in Fig. 8. In Figs. 8(a) and (b), the model performance is depicted for the training data and one validation data set, respectively. It can be seen that the maximum output error is below the 0.1 bar for both the data sets. The static behavior calculated from the nonlinear dynamic turbocharger model is shown in Fig. 8(c). Unfortunately, static measurements from the turbocharger can only be gathered with a great effort on a test stand and are currently not available for comparison. However, the static mapping looks reasonable.

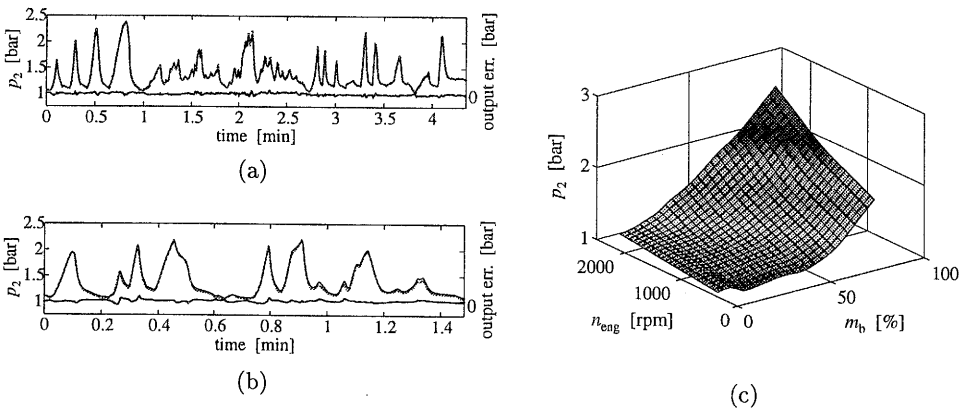


Fig. 8. The LOLIMOT model performance on (a) the training data, (b) the urban traffic validation data, and (c) the static model behavior.

For comparison, the product space clustering approach only requires three rules due to its higher flexibility. The identification results are shown in Fig. 9.

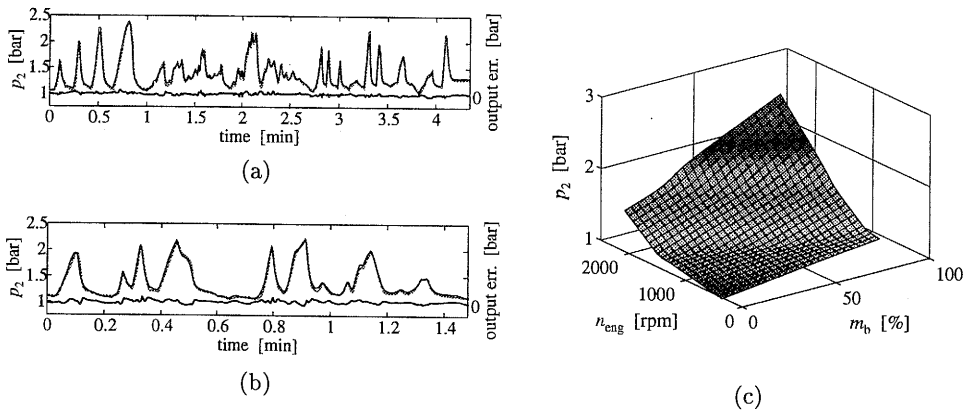


Fig. 9. Performance of the product space clustering model on (a) the training data, (b) the urban traffic validation data, and (c) the static model behavior.

Table 1. Comparison of the computational demands (MATLAB implementation).

	Total training time (sec)	Training (MFLOPS)	Prediction (FLOPS)
LOLIMOT	1.78	15.58	322
Clustering	6.68	39.01	238

The computational demand for training is comparable for both the approaches (Table 1). The number of floating point operations (FLOPS) for the training is for 1309 samples in the training data set. The number of FLOPS for prediction is calculated for one-step-ahead prediction per data sample. Here, one can see that the clustering-based model is cheaper to evaluate, as it only consists of three rules (as compared to 10 of the LOLIMOT model).

Note that for several reasons the numbers can only be used as a rough indication. First of all, the implementations have been optimized for neither speed nor processor load. Secondly, MATLAB counts every operation as a FLOP, regardless of the true numerical type of the arguments. Finally, the two methods were implemented by different programmers and may thus significantly differ in the efficiency of computations.

The prediction performance is similar for both the identification approaches. Table 2 gives the comparison for the training set and for the two validation data sets using the root-mean-squared error measure. Recall that the model constructed through clustering only consists of three rules and has thus a better performance/size ratio. Again, one should be careful in extrapolating this result to other applications or even data sets. Generally, one can expect that LOLIMOT will become superior for larger input dimensions.

Table 2. Root mean squared error (RMSE) on the training (T) and validation (V_1 and V_2) data sets.

	T	V_1	V_2
LOLIMOT	0.022	0.022	0.032
Clustering	0.022	0.023	0.032

6. Conclusions

Two different methods for the identification of Takagi-Sugeno fuzzy models from data have been presented and compared. The local linear model tree (LOLIMOT) algorithm partitions the input space by axis-orthogonal cuts, whereas the fuzzy clustering approach utilizes the Gustafson-Kessel algorithm to find linear submodels in the

product space. The identification results obtained for the turbocharger indicate that approximately the same predictive accuracy is obtained with both the methods and also the computational complexity of the identification methods is comparable. For real-time simulations (such as in predictive control), however, the model obtained through clustering may be preferred, as it consists of fewer rules and is thus less expensive to simulate. An interesting topic for future research is the interpretation in terms of local linear submodels for the two techniques.

References

Akaike H. (1974): *A new look at the statistical model identification.* — IEEE Trans. Automat. Contr., Vol.19, No.6, pp.716–723.

Babuška R. (1998): *Fuzzy Modeling for Control.* — Boston: Kluwer Academic Publishers, USA.

Babuška R. and Verbruggen H.B. (1995): *New approach to constructing fuzzy relational models from data.* — Proc. 3rd Europ. Congress *Intelligent Techniques and Soft Computing EUFIT'95*, Aachen, Germany, pp.583–587.

Babuška R. and Verbruggen H.B. (1996): *An overview of fuzzy modeling for control.* — Contr. Eng. Practice, Vol.4, No.11, pp.1593–1606.

Bezdek J.C. (1981): *Pattern Recognition with Fuzzy Objective Function.* — New York: Plenum Press, USA.

Carpenter G. and Grossberg S. (1988): *The ART of adaptive pattern recognition by a self-organizing neural network.* — IEEE Comp. Vol.21, No.3, pp.77–88.

Dubois D. and Prade H. (1980): *Fuzzy Sets and Systems: Theory and Applications.* — New York: Academic Press.

Ernst S. (1998): *Hinging hyperplane trees for approximation and identification.* — Proc. IEEE Conf. *Decision and Control (CDC)*, Tampa, USA, pp.1261–1277.

Gath I. and Geva A.B. (1989): *Unsupervised optimal fuzzy clustering.* — IEEE Trans. Pattern Anal. Mach. Intell. Vol.11, No.7, pp.773–781.

Gustafson D.E. and Kessel W.C. (1979): *Fuzzy clustering with a fuzzy covariance matrix.* — Proc. IEEE Conf. *Decision and Control (CDC)*, San Diego, USA, pp.761–766.

Ikoma N. and Hirota K. (1993): *Nonlinear autoregressive model based on fuzzy relation.* — Inform. Sci., Vol.71, pp.131–144, New York, USA.

Ishibuchi H., Nozaki K., Yamamoto N. and Tanaka H. (1994): *Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms.* — Fuzzy Sets Syst., Vol.65, No.3, pp.237–253.

Jang J.S.R. (1993): *ANFIS: Adaptive-network-based fuzzy inference systems.* — IEEE Trans. Syst. Man Cybern., Vol.23, No.3, pp.665–685.

Johansen T.A. and Foss B.A. (1993): *Constructing NARMAX models using ARMAX models.* — Int. J. Contr., Vol.58, No.5, pp.1125–1153.

Kaymak U. and Babuška R. (1995): *Compatible cluster merging for fuzzy modeling.* — Proc. FUZZ-IEEE/IFES'95, Yokohama, Japan, pp.897–904.

Kortmann P. (1997): *Fuzzy-Modelle zur Systemidentifikation.* — Düseldorf: VDI-Verlag, Germany, Reihe 8: Mess-, Steuerungs- und Regelungstechnik, Nr. 647.

Krishnapuram R. and Freg C.-P. (1992): *Fitting an unknown number of lines and planes to image data through compatible cluster merging.* — Pattern Recogn., Vol.25, No.4, pp.385–400.

Leontaritis I.J. and Billings S.A. (1985): *Input-output parametric models for nonlinear systems, part 1: Deterministic nonlinear systems.* — Int. J. Contr., Vol.41, No.2, pp.329–344.

Murray-Smith R. (1994): *A Local Model Network Approach to Nonlinear Modeling.* — Ph.D. Thesis, University of Strathclyde.

Nelles O. (1999): *Nonlinear System Identification with Local Linear Neuro-Fuzzy Models.* — Ph.D. Thesis, TU Darmstadt, Automatisierungstechnik series. Shaker Verlag, Aachen, Germany.

Nelles O. and Isermann R. (1996): *Basis function networks for interpolation of local linear models.* — Proc. IEEE Conf. *Decision and Control (CDC)*, Kobe, Japan, pp.470–475.

Pal N.R. and Bezdek J.C. (1995): *On cluster validity for the fuzzy c-means model.* — IEEE Trans. Fuzzy Syst., Vol.3, No.3, pp.370–379.

Strokbro K., Umberger D.K. and Hertz J.A. (1990): *Exploiting neurons with localized receptive fields to learn chaos.* — J. Compl. Syst., Vol.4, No.3, pp.603–622.

Takagi T. and Sugeno M. (1985): *Fuzzy identification of systems and its application to modelling and control.* — IEEE Trans. Syst. Man Cybern., Vol.15, No.1, pp.116–132.

Tanaka M., Ye J. and Tanino T. (1994): *Identification of nonlinear systems using fuzzy logic and genetic algorithms.* — Proc. IFAC Symp. *System Identification*, Copenhagen, Denmark, pp.301–306.

Wang L.-X. (1994): *Adaptive Fuzzy Systems and Control. Design and Stability Analysis.* — Englewood Cliffs: Prentice Hall.

Zinner K.A. (1985): *Aufladung von Verbrennungsmotoren.* — Berlin: Springer.