Nonlinear System Identification

Oliver Nelles
Universität Siegen
233 PUBLICATIONS **4,730** CITATIONS

1 NONLINEAR SYSTEM IDENTIFICATION WITH NEURO-FUZZY METHODS

Oliver Nelles

1.1 INTRODUCTION

This chapter discusses nonlinear system identification with neuro-fuzzy methods. In a general part, summary and overview of the most important types of fuzzy models are given. Their properties, advantages, and drawbacks are illustrated. In a more specific part a new algorithm for the construction of Takagi-Sugeno fuzzy systems is presented in detail. It is successfully applied to the identification of two nonlinear dynamic real-world processes.

Nonlinear system identification is an important task in many disciplines. Nonlinear dynamic models are the foundation for e.g. prediction, simulation, model-based control and fault diagnosis. In most cases, a derivation of such models by first principles (physical, chemical, biological, etc. laws) is expensive, time-consuming and involves many unknown parameters and heuristics. Hence, methods for data-driven modeling (identification) are of great interest.

A large class of multiple-input single-output (MISO) nonlinear dynamic processes can be described in the discrete time domain by (see Leontaritis and

Chapter in: Da Ruan (ed.): Intelligent Hybrid Systems. Kluwer Academic Publishers, Dordrecht, 1997.

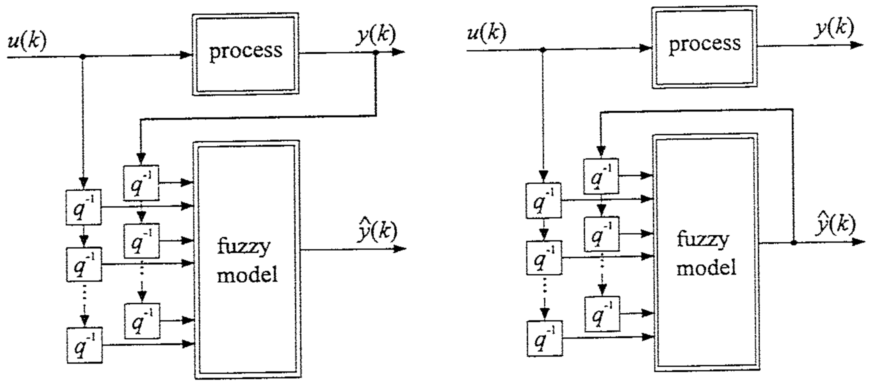

Figure 1.1 Series-parallel model for one-step-ahead prediction (left) and parallel model for simulation (right) for single-input single-output systems. q^{-1} denotes the time delay operator, i.e. $q^{-1}u(k) = u(k-1)$.

Billings, 1985):

$$y(k) = f\left(\underline{\varphi}(k)\right) \tag{1}$$

where the regression vector

$$\begin{aligned}\underline{\varphi}(k) = [\ & u_1(k-d_1-1)\ \dots\ u_1(k-d_1-nu_1)\ \dots \\ & u_m(k-d_m-1)\ \dots\ u_m(k-d_m-nu_m) \\ & y(k-1)\ \dots\ y(k-ny)\ ]^T\end{aligned} \tag{2}$$

is composed out of previous process inputs and outputs. $u_i(k)$ denotes the i-th input and $y(k)$ is the process output, d_i and nu_i are the dead times and dynamic orders of input i, and ny is the dynamic order of the output.

The unknown function $f(\cdot)$ in (1) has to be approximated from measurement data. Thus, the dynamic system identification problem is reformulated as a static function approximation problem. This, however, is only true if a one-step-ahead *prediction* model is required, that is the previous *process* outputs $y(k-i)$ are available during normal operation of the model. This is the case for e.g. weather or stock market prediction. A prediction model is also called a *series-parallel model*, compare Fig. 1.1 (left). If, however, the goal of modeling is not prediction but *simulation* no information about the process output is available during normal operation. Then, in (2) the process outputs $y(k-i)$ must be replaced with the model outputs $\hat{y}(k-i)$. Such a simulation model is also called a *parallel* model, compare Fig. 1.1 (right).

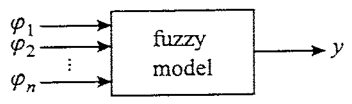

Figure 1.2 A fuzzy model with n inputs and one output.

This chapter is organized as follows. First, a section introduces different types of fuzzy models. The fuzzy basis function formulation, some important interpretation issues, and the preservation of prior knowledge for learning fuzzy systems are discussed. The next section introduces some fundamental issues in optimization and summarizes the three most popular approaches to learning fuzzy systems, namely the orthogonal least-squares method, neuro-fuzzy systems, and the application of global search techniques. The Section "Overcoming the Curse of Dimensionality" gives an overview about different strategies for dealing with high-dimensional input spaces. The next section entitled "Local Linear Model Trees (LOLIMOT)" proposes a new algorithm for the construction of Takagi-Sugeno fuzzy systems. More specifically the estimation of the rule consequents parameters, the optimization of the rule consequents structure, and the optimization of the rule premise structure are discussed. The succeeding two sections demonstrate the performance of this LOLIMOT approach with an application to real-world processes. First, a combustion engine turbocharger is identified with a mutliple-input single-output model. Second, a transport delay process with operating point dependent dead times is identified. Finally, a summary is given and some conclusions are drawn from the presented results.

1.2 FUZZY MODELS

This section discusses some fundamental issues on fuzzy systems. First, the rule structure of fuzzy systems is introduced. Then, different types of fuzzy systems are discussed and the fuzzy basis function formulation is explained. Finally, two subsections are devoted to interpretation issues in context with fuzzy systems.

A good overview on fuzzy systems can be found in Mendel, 1995 and Babuška and Verbruggen, 1996. The following brief introduction aims to clarify the terminology and notation. In fuzzy systems the relationship between an n-dimensional input $\underline{\varphi} = [\varphi_1 \ \varphi_2 \ \cdots \ \varphi_n]^T$ and the output y (see Fig. 1.2) is described in form of rules that may have the following form ("$\wedge$" denotes a conjunction):

$$R_1 : \text{ IF } \varphi_1 = A_{1,1} \;\wedge\; \varphi_2 = A_{1,2} \;\wedge\; \dots \;\wedge\; \varphi_n = A_{1,n} \;\text{ THEN } y = B_1$$
$$\vdots$$
$$R_M : \text{ IF } \varphi_1 = A_{M,1} \;\wedge\; \varphi_2 = A_{M,2} \;\wedge\; \dots \;\wedge\; \varphi_n = A_{M,n} \;\text{ THEN } y = B_M$$

Each of these M rules has a premise with n antecedents $\varphi_1 = A_{i,1}$ to $\varphi_n = A_{i,n}$ and a consequent $y = B_i$ where $A_{i,j}$ denote the input fuzzy sets and B_i the output fuzzy sets. If the output is multi-dimensional either the above rule consequents can be extended or simply one fuzzy system can be designed for each output. Rules that not only use the "AND" but also the "OR" connective can be translated into the rule type given above by standard techniques from crisp logic, see Mendel, 1995. If rules do not include all possible antecedents in their premise they are called incomplete, see Mendel, 1995. The unused input variables can be interpreted as irrelevant with respect to the rule consequent. Thus, the rule

$$\text{IF } \varphi_1 = A_{1,1} \;\text{ THEN } y = B_1$$

can be understood as

$$\text{IF } \varphi_1 = A_{1,1} \;\wedge\; \varphi_2 = \text{don't care} \;\wedge\; \dots \;\wedge\; \varphi_n = \text{don't care} \;\text{ THEN } y = B_1$$

It is shown later, that incomplete rules are very important in the context of high-dimensional input spaces. Fuzzy systems run into severe problems for high-dimensional input spaces due to the so-called curse of dimensionality (see Bellman, 1961), this is an exponential increase of complexity with increasing input dimension. All lattice-based approaches inherently suffer from this problem.

1.2.1 Types of Fuzzy Models

A fuzzy system may be applied to a classification or an approximation problem. In classification problems, real valued input features $\underline{\varphi}$ are mapped to discrete output values y, i.e. the classes. The rule consequents either realize an integer value representing the classes or the fuzzy system has one output for each class representing the estimated a-posteriori probability for the corresponding class.

In approximation problems real valued inputs $\underline{\varphi}$ are mapped to real valued outputs y. Then, the rule consequents can be described by fuzzy sets, which results in a truly linguistic model, e.g.:

$$\text{IF } \varphi_1 = \textit{medium} \;\wedge\; \varphi_2 = \textit{small} \;\wedge\; \dots \;\wedge\; \varphi_n = \textit{large} \;\text{ THEN } y = \textit{small}$$

Sometimes, a confidence is assigned additionally to each fuzzy rule to make the fuzzy system more flexible and less sensitive to the exact shape of the output membership functions (MFs).

If the fuzzy sets in this linguistic fuzzy model are reduced to real numbers a fuzzy system with singletons is obtained, e.g.:

$$\text{IF } \varphi_1 = \textit{medium} \ \wedge \ \varphi_2 = \textit{small} \ \wedge \ \dots \ \wedge \ \varphi_n = \textit{large} \ \text{ THEN } y = 3.4$$

Fuzzy systems with singletons are simple to implement and the singletons are easy to optimize. Therefore, they are very popular in many engineering applications. Furthermore, many defuzzification techniques and implication operators lead to equivalent results. However, the interpretation of fuzzy systems with singletons is difficult since each rule has its own (different) real-valued output. Under some conditions a fuzzy model with singletons and a linguistic fuzzy model with rule confidences are equivalent. Then, the singletons can be calculated from the rule confidences and output fuzzy sets and vice versa, see Brown and Harris, 1994. This procedure may be advantageous for interpretation of fuzzy systems with singletons.

A third very popular type of fuzzy system is the Takagi-Sugeno fuzzy system, where the rule consequents model some function $f(\cdot)$ of the input variables, e.g.:

$$\begin{aligned}\text{IF } \varphi_1 = \textit{medium} \ \wedge \ \varphi_2 = \textit{small} \ \wedge \ \dots \ \wedge \ \varphi_n = \textit{large}& \\ \text{THEN } y = f(\varphi_1, \varphi_2, \dots, \varphi_n)&\end{aligned}$$

Usually, $f(\cdot)$ is chosen as a linear function of the inputs: $y = w_0 + w_1\varphi_1 + \dots + w_n\varphi_n$. A Takagi-Sugeno fuzzy system simplifies to a fuzzy system with singletons if the function $f(\cdot)$ is constant. Generally, it can be said that more complex functions $f(\cdot)$ are able to model larger regions in the input space and therefore require less rules. However, the more complex $f(\cdot)$ is, the more parameters it includes and the less interpretable a Takagi-Sugeno fuzzy system becomes. On the other hand, Takagi-Sugeno fuzzy systems with linear consequents are especially popular for modeling dynamic systems. Then, the rule consequents can be interpreted as local linear models (transfer functions) with their own gains, poles and zeros. This makes Takagi-Sugeno fuzzy systems highly interpretable when dealing with dynamic system.

1.2.2 Fuzzy Basis Functions

The most popular fuzzy composition and defuzzification methods (see Mendel, 1995) lead to the following crisp output value of a Takagi-Sugeno fuzzy system:

$$y(\underline{\varphi}) = \frac{\sum_{i=1}^{M} \mu_i(\underline{\varphi}) \cdot f_i(\underline{\varphi})}{\sum_{i=1}^{M} \mu_i(\underline{\varphi})} \tag{3}$$

where μ_i denotes the degree of fulfillment of rule i. Strictly speaking, this is no real defuzzification since Takagi-Sugeno fuzzy systems do not have fuzzy sets in the consequents. It simply can be interpreted as a weighted average, that is each rule consequent is weighted according to the degree of rule fulfillment. This formula can also be applied to singleton fuzzy systems with $f_i(\underline{\varphi})$ = constant. For linguistic fuzzy systems similar formulas as (3) are obtained depending on the defuzzification method employed. Because overlapping of fuzzy sets must be taken into account linguistic fuzzy models require a higher computational effort than singleton or Takagi-Sugeno fuzzy models.

The degree of rule fulfillment $\mu_i(\underline{\varphi})$ has to be calculated as the conjunction of the corresponding degrees of antecedent fulfillment. Any t-norm can be applied for performing the conjunction, popular choices are the product- and min-operators. Thus, $\mu_i(\underline{\varphi})$ may be computed as $\mu_{i,1}(\varphi_1)$ and $\mu_{i,2}(\varphi_2)$ and ... and $\mu_{i,n}(\varphi_n)$, where $\mu_{i,1}(\varphi_1)$ is the degree of membership that φ_1 is medium, $\mu_{i,2}(\varphi_2)$ that φ_2 is small ... and $\mu_{i,n}(\varphi_n)$ that φ_n is large.

The output of a fuzzy system in (3) can be interpreted as a weighted sum of the following basis functions (see Kim and Mendel, 1995)

$$\Phi_i(\underline{\varphi}) = \mu_i(\underline{\varphi}) \Big/ \sum_{i=1}^{M} \mu_i(\underline{\varphi}) \tag{4}$$

and therefore (3) can be written as

$$y = \sum_{i=1}^{M} f_i(\underline{\varphi}) \cdot \Phi_i(\underline{\varphi}) \tag{5}$$

In this form it is easy to see that if the functions $f_i(\underline{\varphi})$ are linear in their parameters (as this is usually the case) then y is linear in those parameters as well. However, all parameters that determine the shape of the membership functions, such as position and width, influence $\Phi_i(\underline{\varphi})$ in a nonlinear way. Note, that although (5) formally looks like any other basis function expansion it differs in the following fact. If one basis function is excluded from or added to the sum in (5) then all other basis functions change their shape due to a modified normalization factor in the denominator of (4). This has important consequences for any rule selection procedure.

1.2.3 Interpretation of Fuzzy Models

The major difference between fuzzy systems and other nonlinear approximators is the possibility of interpretation in terms of rules. Therefore, it is of major importance to discuss the circumstances under which a fuzzy system is really interpretable. Clearly, this depends on the specific application. For example, as noted above, Takagi-Sugeno fuzzy systems are well interpretable when applied to dynamic process modeling but they are poorly interpretable when applied to static modeling. However, some general interpretation guidelines can be given and certainly good interpretation does not automatically follow from the existence of a rule structure. When optimizing fuzzy systems interpretation issues should always be considered.

The following factors may influence the interpretation of a fuzzy system:

- *Number of rules:* If the number of rules is too large the fuzzy system can be hardly understood by the user. Especially for systems with many inputs the number of rules often becomes overwhelmingly large if all antecedent combinations are realized.

- *Number of antecedents in the rule premise:* Rules with premises that have many, say more than three or four, antecedents are hard to interpret. In human languages most rules include only very few antecedents even if the total number of inputs relevant for the problem is large. As demonstrated in the Section "Fuzzy Systems" these kind of rules can be generated by introducing "don't cares".

- *Dimension of input fuzzy sets:* One way to avoid or at least to reduce the difficulties with high-dimensional input spaces and to decrease the number of rules, is to utilize high-dimensional input fuzzy sets, see e.g. Kroll, 1996. These approaches discard the lattice structure that is typical for fuzzy systems. However, it is exactly the conjunction of one-dimensional input fuzzy sets that make a fuzzy system easy interpretable. Multi-dimensional input fuzzy sets with more than three inputs certainly are beyond human imagination.

- *Order of fuzzy sets:* Fuzzy sets should be ordered such that e.g. very small is followed by small that is followed by medium and large etc. If a fuzzy system is developed with expert knowledge such an ordering of fuzzy sets is intuitive. In a successive optimization procedure, however, this ordering can be lost if no precautions are taken. Although it is, in principle, possible to re-label the fuzzy sets, this will lead to difficulties in the rule interpretation and the expert knowledge incorporated into the initial fuzzy system may get lost to a large extent.

- *Normalization of input membership functions (partition of unity):* Often the membership functions are chosen such that they sum up to one for each input, e.g. a 30 year old person may be considered young with a degree of membership of 0.2, be of middle age with 0.7 and old with 0.1. This property is intuitively appealing. If all membership functions sum up to one for each input and a complete rule base (all antecedent combination, i.e. full lattice) is implemented, it can be shown that the

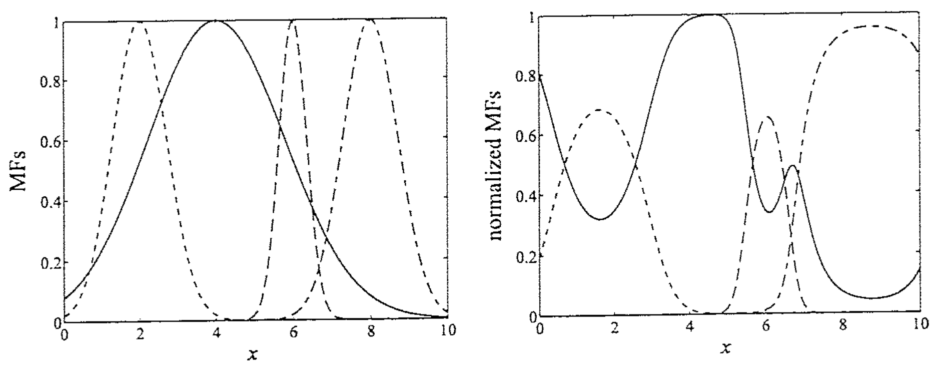

Figure 1.3 Gaussian membership functions with different standard deviations (left). Normalized Gaussian membership functions that sum up to one (right). The second membership function (solid line) has the largest standard deviation and therefore becomes dominant for $x \to \infty$ and $x \to -\infty$. Thus, the normalized membership functions become multimodal and non-local. Rules that include the second membership function do not only influence regions around its center $x = 4$ but also have a dominant effect around $x = 0$. This behavior is usually not expected by the user. Note, that due to the normalizing denominator in (1) or (2) a similar effect takes place if the non-normalized membership functions (left) are used.

denominator in (3) or (4) is equal to one. This property is called partition of unity and is generally considered to be advantageous (see Werntges, 1993). It does not hold if only a subset of the complete rule base is realized. The user of rule selection algorithms should always be aware of "strange" effects that might be caused by a modified denominator in (3) or (4). Thus, discarding or adding of rules may change the fuzzy system in a not easy-to-overview fashion.

There are two ways to achieve normalized (i.e. summing up to one) input fuzzy sets. One way is to choose membership functions that naturally employ this property, such as triangles with appropriate slopes. More generally B-splines of order m can be used (see Brown and Harris, 1994). Another way is to normalize arbitrary membership functions, e.g. Gaussians. Figure 1.3 shows an undesired effect that occurs if Gaussian membership functions do not have exactly the same width. Due to the normalization the rules may have non-local influence, which can be regarded as a highly unexpected and undesired property. Note, that if no explicit normalization is performed for all input fuzzy sets this normalization is automatically carried out by the denominator in (3) or (4), respectively.

All these points impose constraints on both, the type of fuzzy system employed and the optimization technique applied. In fact, depending on the specific application some of those points can be relaxed or become even more important. However, it is advised to keep those interpretation issues in mind when dealing with fuzzy systems.

1.2.4 *Preserving Prior Knowledge*

One issue in the optimization of fuzzy systems which needs more attention in future research is how to preserve prior knowledge during the optimization procedure. It is assumed that a fuzzy system is developed by expert knowledge and subsequently should be optimized based on data. A detailed discussion of this topic can be found in Lindskog, 1996.

The order of the input membership functions can be considered as "hard knowledge", i.e. it can either be violated or not. This order can easily be preserved by performing a constrained optimization. If genetic algorithms are applied these constraints can be elegantly incorporated by a relative coding, that is the position of each membership function is not coded with its absolute value but as a (always positive) distance from the neighboring membership function.

Further restrictions can be considered as "soft knowledge". The expert may like to restrict the membership functions in such a way, that they do not differ "too much" from his initially chosen ones, where "too much" is defined in some way by the expert. This can be done by the incorporation of penalty terms in the objective function. An interesting approach is suggested in Lindskog, 1996. The expert specifies the quality of his assumptions in form of fuzzy rules. These rules are then utilized to compute the penalty function.

In Lindskog, 1996 it is demonstrated that constrained compared to unconstrained optimization may not only lead to an easier interpretation of the fuzzy system. The performance may be higher as well because a better local optimum can be found. Since fuzzy systems are often overparameterized, i.e. have more parameters than reasonably can be estimated from the available amount of data (see the discussion of the bias/variance dilemma in the next section), constraining the flexibility of the fuzzy system can be advantageous with respect to performance as well. However, in some applications unconstrained optimization of a fuzzy system will lead to better performance since the constraints limit the flexibility of the fuzzy model.

All constraints impose restrictions on the search space, that is they reduce its size. Therefore, the rate of convergence of the applied optimization technique will increase with the number of constraints.

1.3 OPTIMIZATION OF FUZZY MODELS

This section summarizes some common approaches for the optimization of fuzzy models. After a fundamental discussion about the optimal complexity of a fuzzy system this section gives a brief overview on the following popular fuzzy optimization approaches: orthogonal least-squares learning, neuro-fuzzy systems, and the application of global search techniques.

It is helpful to clearly distinguish between four different categories of optimization techniques applied to fuzzy systems. Unsupervised methods, such as clustering, detect structures in the input data distribution, e.g. ellipses. Supervised methods take the desired output value into account and can be classified into linear and nonlinear techniques. Nonlinear optimization techniques can further be classified into nonlinear local methods that directly converge to a local optimum and nonlinear global methods that try to explore the search space globally. The strength of global methods is to search regions, while local methods are good in searching points. Thus, hybrid approaches that first run a global method for searching good performing regions in the search space and subsequently apply a local method for fast convergence to the final solution are becoming more and more popular, see Pedrycz, 1997.

While clustering and linear optimization methods usually are computationally inexpensive, nonlinear local techniques require higher and nonlinear global techniques much higher computational effort.

1.3.1 Basics in Fuzzy Model Optimization

One of the most important questions concerning the optimization of any model is: How complex or flexible should the model be? The complexity and flexibility of the fuzzy system, or any other model, directly relates to the number of parameters to be determined. Parameters of fuzzy systems typically are the positions and widths of the membership functions etc. The problem arises from the fact that the available data set usually is noisy and always finite. Therefore, it is not possible to estimate an arbitrary number of parameters from the data set. Although the performance on the training data set continuously improves with an increasing number of parameters, the performance on fresh data (a test data set) starts to decrease at some point. At this point the fuzzy system starts to fit noise (overfitting) contained in the training data and consequently the generalization performance starts to deteriorate. The optimal complexity of the fuzzy system is represented by this point.

The expected error of a fuzzy system on fresh data is composed out of two parts: a bias and a variance term (see Geman et al., 1992). The bias part describes the systematical error that is due to lacking model flexibility. Thus, the bias part is very high for fuzzy systems with only a few parameters and it

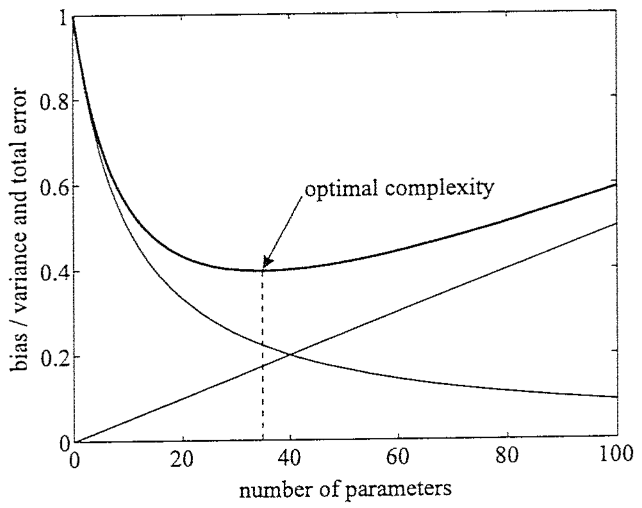

Figure 1.4 The bias error decreases while the variance error increases with growing model complexity (number of parameters). In this example a model with about 35 parameters has the best bias/variance trade-off.

goes to zero if the number of parameters tends to infinity. On the other hand, the variance part describes the error that follows from an incorrect parameter estimation due to the noisy and finite training data set. Consequently, the variance error is small for fuzzy systems with only a few parameters and it increases with the model flexibility. It can be shown under some mild conditions (see Ljung, 1987), that asymptotically (i.e. for very large data sets) the generalization error is proportional to the noise variance and one plus the number of parameters divided by the number of training data samples:

$$\text{squared generalization error} \sim \text{noise variance} \cdot \left(1 + \frac{\text{number of parameters}}{\text{number of training data samples}}\right) \quad (6)$$

It is conform with common sense that the parameter estimates improve with smaller noise variance, larger training data set and smaller number of parameters to be estimated. Obviously, the determination of the optimal fuzzy system complexity requires a trade-off between the bias and the variance error, Fig. 1.4. Because both error parts are in conflict this is called the bias/variance dilemma.

There exist two different approaches to perform the bias variance trade-off. One possibility is to choose a fuzzy system complexity by rule selection that is appropriate for the problem. Another way is to perform regularization. Regularization techniques reduce the variance error while increasing the bias error. Consequently, they allow to realize fuzzy systems with a large number of parameters. One of the most popular regularization techniques is the so-called "early stopping", that is the training procedure is not performed until convergence. It is stopped when the generalization error on a test data set has reached its minimum. Strictly speaking, if a regularization technique is performed the number of parameters in (6) must be replaced by the effective number of parameters, see Sjöberg, 1995. Loosely speaking, the effective number of parameters are those parameters which have converged to their final value; those are the significant (most important) parameters.

All methods that reduce the flexibility of the fuzzy system can be regarded as regularization techniques. In Johansen, 1996 different regularization techniques for Takagi-Sugeno fuzzy systems are analyzed. One very efficient possibility with respect to the required computational effort is to estimate the parameters of each rule consequent $f_i(\cdot)$ separately. This restricts the model flexibility because each rule consequent is forced to fit the data locally. Furthermore, all soft constraints imposed (see Subsection "Preserving Prior Knowledge") can be regarded as regularization as well.

1.3.2 Orthogonal Least-Squares Learning

The orthogonal least-squares algorithm as a mature forward subset selection method in statistics (see Miller, 1990) may offer a solution to this rule selection problem. It assumes a singleton fuzzy system and starts from the following linear least-squares problem obtained by the fuzzy basis function formulation in (5):

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \Phi_1(1) & \Phi_2(1) & \cdots & \Phi_M(1) \\ \Phi_1(2) & \Phi_2(2) & \cdots & \Phi_M(2) \\ \vdots & \vdots & & \vdots \\ \Phi_1(N) & \Phi_2(N) & \cdots & \Phi_M(N) \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_M \end{bmatrix} \tag{7}$$

where N is the number of training data samples, y is the desired fuzzy system output vector, $\Phi_i(\cdot)$ is the i-th fuzzy basis function and θ_i is the singleton that corresponds to the i-th rule. In order to optimize a fuzzy system that includes all M rules simply the above overdetermined (assuming $N > M$) linear equation system has to be solved in the least-squares sense. The orthogonal least-squares algorithm selects a subset of M_s rules out of the given M rules in (7). This approach was taken in Wang and Mendel, 1992 and is extended to Takagi-Sugeno fuzzy systems with linear functions in the rule consequents in Wang and Langari, 1995.

However, as mentioned in Hohensohn and Mendel, 1994 a serious problem arises. The fuzzy basis functions $\Phi_i(\cdot)$ are not independent from each other due to the denominator in (4). This violates an assumption of the linear subset selection methods. After the orthogonal least-squares algorithm has selected M_s rules out of (7), the basis functions are still normalized by a denominator in (4) that contains the fuzzy basis functions of all M rules. Thus, even after rule selection all rules are required for normalization in (4). Therefore, this approach makes no sense, neither in terms of computational demand (no rule is really discarded) nor in terms of interpretation.

As a remedy to these difficulties it is suggested in Hohensohn and Mendel, 1994 that after the rule selection procedure described above, in a second step, all non-selected fuzzy basis functions are discarded from the denominator in (4) as well. Since this modifies the shape of the selected basis functions the singletons are re-estimated in order to adapt to the new shape. However, from Fig. 1.3 it is clear that discarding the non-selected fuzzy basis functions may totally change the characteristics of the fuzzy rules. Therefore, the performance of the rule selection procedure can strongly deteriorate.

These difficulties can be overcome by applying a genetic algorithm to the subset selection problem. This is the most important motivation for applying genetic algorithms to the rule structure optimization problem.

1.3.3 Neuro-Fuzzy Approaches

The term neuro-fuzzy system originates from the application of neural network architectures for implementing a fuzzy system in order to utilize common neural network training algorithms. The most important systems are based on Kohonen feature maps (clustering), multi-layer perceptrons (gradient algorithms) or radial basis function networks (least-squares).

In many applications ellipsoidal clustering is performed in the input space as a preprocessing step. Each cluster in the input space may represent a rule and the input membership functions can be obtained by projecting the clusters onto the input axes. Only axis-orthogonal clusters can be reconstructed exactly by the conjunction of the input membership functions.

Unsupervised methods such as clustering are accompanied by the problem that the generated rules represent somehow the input data distribution. Without any information about the desired output there is no guarantee that the distribution of the rules will reflect the complexity of the underlying function. It is likely that many rules are placed in regions where the training data is dense and only few rules are placed where the training data is sparse. A desired feature is, however, that many rules are placed where the underlying function is complex and few rules are constructed where it can be approximated easily.

Therefore, most neuro-fuzzy algorithms that are based on clustering take the output-dimension into account and cluster in the $[\varphi_1 \ \varphi_2 \ \dots \ \varphi_n \ y]$-space, see Babuška and Verbruggen, 1996 and Dickerson and Kosko, 1996.

Other neuro-fuzzy systems rely on a multi-layer perceptron-like architectures and apply gradient-based training algorithms for membership function optimization, see e.g. Halgamuge and Glesner, 1994 and Nauck and Kruse, 1993. Pruning and growing algorithms for multi-layer perceptron structure optimization can be adopted in order to implement a fuzzy rule generation procedure.

Finally, the equivalence between normalized radial basis function networks and some kinds of fuzzy systems (see Hunt et al., 1996) can be utilized to optimize the fuzzy system.

1.3.4 Global Optimization Approaches

In contrast to the neuro-fuzzy methods, the global search techniques approach the problem of fuzzy system optimization in a more direct manner. They can be applied to both, parameter and structure optimization. Since many conventional optimization schemes are weak in structure optimization (see e.g. Section "Orthogonal Least-Squares Learning") this is the most promising domain for global search methods. If it is reasonable to apply them for parameter optimization depends on the specific problem and especially on the available prior knowledge.

This subsection gives a brief overview on global search techniques:

- *Evolutionary algorithms:* They are probabilistic search methods that employ a search technique based on ideas from natural genetics and evolutionary principles. Roughly, it can be distinguished between evolution strategies and genetic algorithms (see Davis, 1991). The typical features of evolution strategies are a real coding of the parameters, mutation as the most important operator, and a self-adaptation of the internal so-called strategy parameters, i.e. meta-parameters that influence the mutation rate. Genetic algorithms typically rely on binary coding and the cross-over operator is more important than mutation. Thus, evolution strategies may be more advantageous in the case of real-valued parameter optimization, while genetic algorithm may be ahead for combinatorial optimization problems such as rule selection.

- *Simulated annealing:* This is also a probabilistic search method. Its name stems from the following physical analogy that describes the ideas behind the algorithm: A warm particle is simulated in a potential field. Generally the particle moves down towards lower potential energy, but since it has a non-zero temperature, i.e. kinetic energy, it moves randomly and therefore occasionally jumps to higher potential energy. Thus, the particle is capable of escaping local minima and possibly finds a global one. The particle is annealed in this process, that is its temperature decreases gradually, so the probability of moving uphill decreases with time. In the context of optimization, the particle represents the parameter point in search space and the potential energy represents the objective function. In Ingber and Rosen, 1992 simulated annealing is compared with a GA for a number of benchmark problems.

- *Tabu search:* In contrast to evolutionary algorithms and simulated annealing, tabu search is more local and less probabilistic. It starts to search the space in a local neighborhood but prevents from being captured in a local minimum by systematically storing previously examined points, see Battiti, 1996. Up to now, tabu search is successfully applied mainly for operations research problems and an application in the context of fuzzy systems needs further studies.

1.4 OVERCOMING THE CURSE OF DIMENSIONALITY

Several popular algorithms try to reduce or to avoid the curse of dimensionality, which is one of the most important difficulties in dealing with fuzzy systems. For fuzzy systems the following strategies exist, compare Brown et al., 1995:

- *Additive models:* As shown in Fig. 1.5, the idea is to construct the model output as a sum of several low-dimensional fuzzy systems. Since low-dimensional fuzzy systems allow an easier interpretation such an approach is very appealing. The performance depends upon how good the process can be described in terms of additive models.

  The most common additive modeling algorithm is ASMOD (see Kalvi, 1993 and Bossley, 1996). It starts with an empty model and then iteratively constructs a fuzzy system applying the following three operations. It introduces new one-dimensional fuzzy subsystems, increases the number of membership functions and extends the flexibility (and dimensionality) of fuzzy subsystems by incorporating new inputs.

- *Hierarchical models:* Most fuzzy systems are flat in the sense that the rules directly relate the inputs to the outputs without intermediate states. Introducing hierarchical fuzzy systems as depicted in Fig. 1.6 is an interesting alternative. The interpretation of such hierarchical fuzzy systems is very close to human thinking if the intermediate states (ξ_i) have significance in the real world. Due to the highly nonlinear structure training of hierarchical fuzzy systems is a difficult task, see Raju and Zhou, 1993.

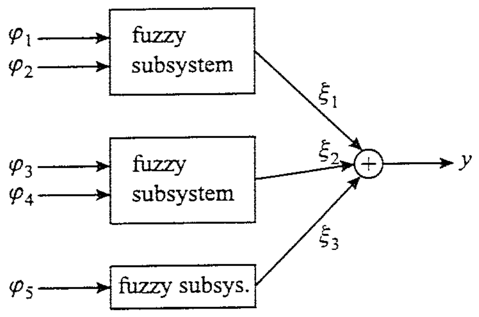

Figure 1.5 An additive fuzzy model composed out of three fuzzy subsystems.

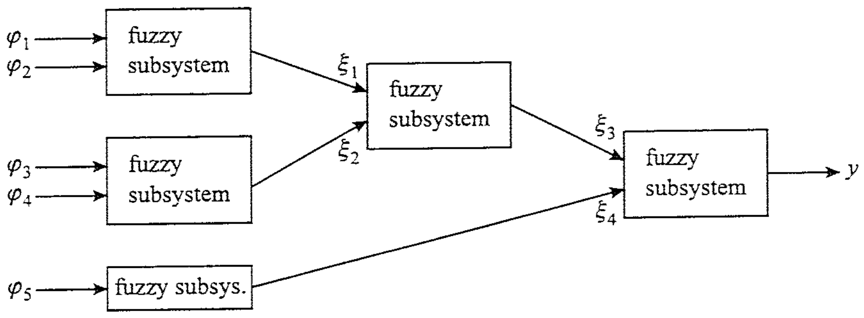

Figure 1.6 A hierarchical fuzzy model composed out of five fuzzy subsystems.

- *Non-lattice partitioning:* The exponential increase in complexity with the input dimension is a direct consequence of the lattice structure that is common for most fuzzy systems. As mentioned in Section "Fuzzy Systems", incomplete rules are one possibility to overcome the curse of dimensionality. In Krone, 1996 the so-called ROSA algorithm is discussed which uses incomplete rules for a singleton fuzzy system. In Nelles and Isermann, 1996 the LOLIMOT tree construction algorithm is introduced for Takagi-Sugeno fuzzy systems. Figure 1.7 shows the partitioning of the input space for lattice-based and tree-constructed fuzzy systems. Note, that an axis-orthogonal partitioning is necessary in order to keep the fuzzy system interpretable in terms of one-dimensional membership functions for each input.

- *Multi-resolution lattice:* Another possibility to deal with (but not to avoid) the curse of dimensionality is to generate multiple rule lattices

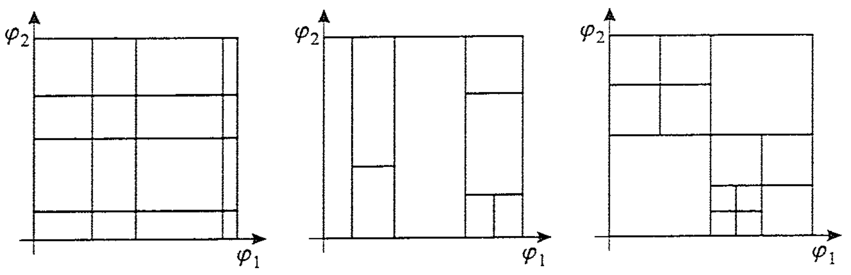

Figure 1.7 Lattice structure (left), k-d-tree partitioning (center), and quad-tree partitioning (right).

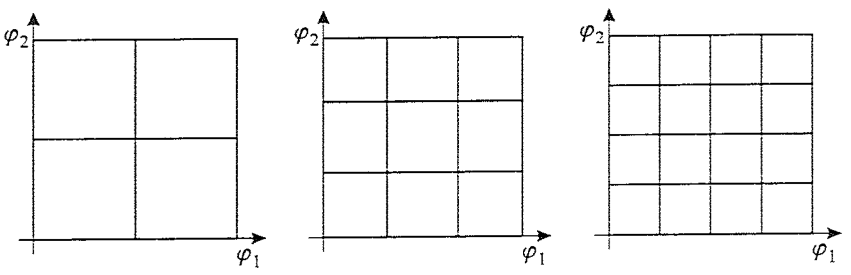

Figure 1.8 2×2 rules lattice (left), 3×3 rules lattice (center), and 4×4 rules lattice (right).

of different resolution (or granularity) (see Ishibuchi et al., 1994), see Fig. 1.8. In a second step a search technique (e.g. a GA) is applied for rule selection. However, the number of potential rules becomes so huge for high-dimensional input spaces that this approach is feasible only for moderately sized input dimensions.

1.5 LOCAL LINEAR MODEL TREES (LOLIMOT)

In this section a powerful approch for the construction of Takagi-Sugeno fuzzy models is introduced. It is based on a combination of a tree construction algorithm and local linear least-squares estimation schemes. The tree construction algorithm optimizes the rule premises and overcomes the curse of dimensionality by a non-lattice input space partitioning. The local estimation approach is very fast and robust against noisy data sets. It is a form of implicit regularization, compare Subsection "Basics in Fuzzy Model Optimization".

For an illustration of the following mathematical descriptions the Figs. 1.9 and 1.10 depict the functioning of Takagi-Sugeno fuzzy modes for the one- and two-dimensional case respectively.

A general Takagi-Sugeno fuzzy model with different input spaces for the rule premises and rule consequents can be described by M rules in the following form:

$$R_j : \text{IF } z_1 = A_{j,1} \wedge z_2 = A_{j,2} \wedge \ldots \wedge z_{nz} = A_{j,nz} \quad \text{THEN } y(k) = w_{j,0} + w_{j,1}x_1 + \ldots + w_{j,nx}x_{nx}$$

where $A_{j,i}$ is a fuzzy set defined on the universe of discourse of input i. In most applications of Takagi-Sugeno fuzzy models the nz-dimensional vector $\underline{z}(k) = [z_1 \ z_2 \ \ldots \ z_{nz}]^T$ which appears in the rule premises is identical to the nx-dimensional vector $\underline{x}(k) = [x_1 \ x_2 \ \ldots \ x_{nx}]^T$ that appears in the rule consequents. This kind of fuzzy model is a universal approximator of the function $f(\cdot)$ in (1) if $\underline{z}(k) = \underline{x}(k) = \underline{\varphi}(k)$. However, many practical considerations may lead to different choices of the vectors $\underline{x}(k)$ and $\underline{z}(k)$. In the following, the argument "(k)" is discarded for better readability.

A Takagi-Sugeno fuzzy system can be interpreted as follows. The rule consequents realize local linear models (hyper-planes) in the nx-dimensional space spanned by $\underline{x}$. Each rule premise constructs an nz-dimensional validity function in the space spanned by $\underline{z}$ by the conjunction of all nz antecedent fuzzy sets. The validity functions define the regions where the local linear models are valid. Therefore, the output of such a Takagi-Sugeno fuzzy model with M rules, Gaussian membership functions and product-operator as t-norm can be written as

$$y(k) = \sum_{j=1}^{M} (w_{j,0} + w_{j,1}x_1 + \ldots + w_{j,nx}x_{nx}) \, \Phi_j(\underline{z}, \underline{c}_j, \underline{\sigma}_j) \tag{8}$$

where c_j and σ_j are the centers and standard deviations of the input fuzzy sets. This yields the validity function $\Phi_j(\cdot)$ for rule j:

$$\Phi_j(\underline{z}, \underline{c}_j, \underline{\sigma}_j) = \frac{\mu_j}{\sum_{i=1}^{M} \mu_i} \tag{9}$$

with μ_j as the degree of fulfillment of rule j:

$$\mu_j = \exp\left(-\frac{1}{2}\frac{(z_1 - c_{j,1})^2}{\sigma_{j,1}^2}\right) \cdot \ldots \cdot \exp\left(-\frac{1}{2}\frac{(z_{nz} - c_{j,nz})^2}{\sigma_{j,nz}^2}\right) \tag{10}$$

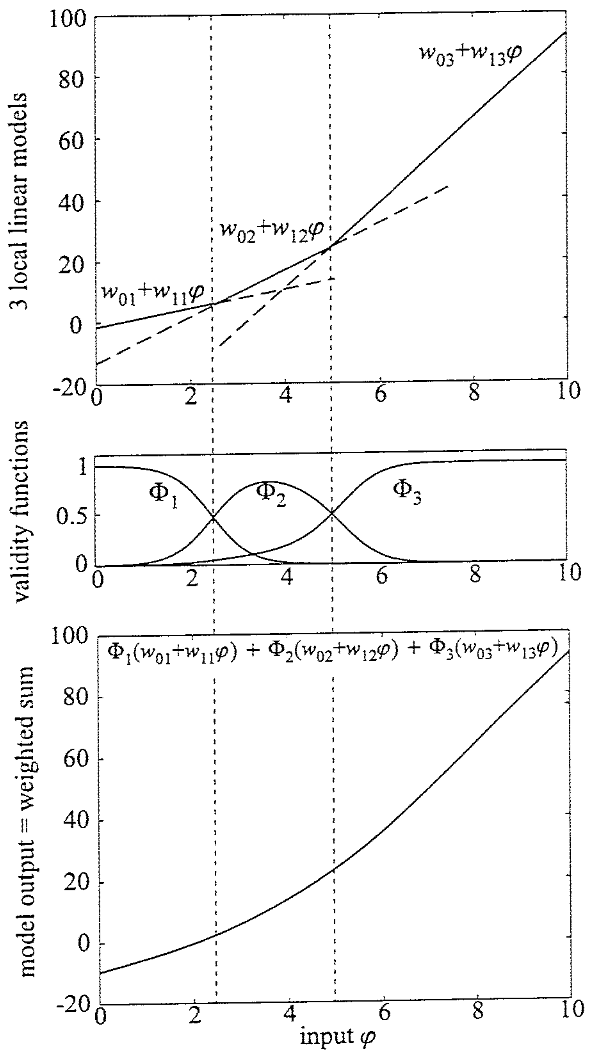

Figure 1.9 A Takagi-Sugeno fuzzy model for the one-dimensional case. The local linear models (top) are weighted with their validity fucntions (middle) and sum up to the model output (bottom).

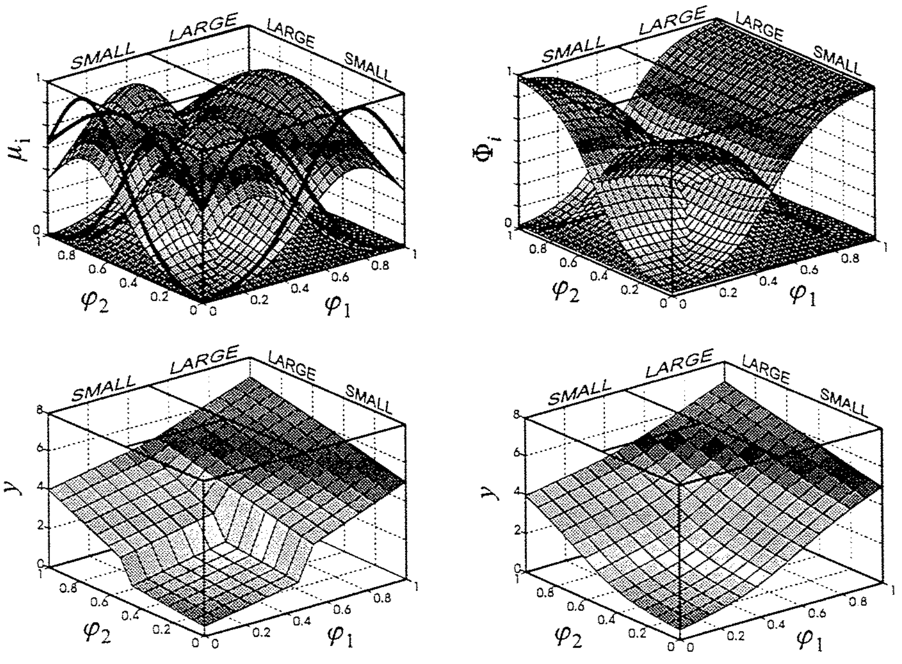

Figure 1.10 A Takagi-Sugeno fuzzy model for the two-dimensional case: two-dimensional membership functions are constructed by the conjunction (multiplication) of the one-dimensional membership functions (top left), ther normalized validity functions (top right), the local linear models (bottom left), and the smoothed model output obtained by a weighting of the local linear models with the validity functions (bottom right).

The Takagi-Sugeno fuzzy model is equivalent (see Hunt et al., 1996) to the local model network analyzed in Murray-Smith, 1994 and Johansen and Foss, 1993. However, for a proper interpretation in terms of fuzzy rules the validity functions $\Phi_j(\cdot)$ have to be constructed as conjunction of the one-dimensional fuzzy sets $A_{j,i}$. There is no equivalent restriction in the local model network approach.

In the following, the local linear model tree (LOLIMOT) algorithm for the construction of Takagi-Sugeno fuzzy models is presented. It consists of an inner loop which optimizes the rule consequents and an outer loop which optimizes the premise structure. First, the estimation of the rule consequents parameters is discussed. Next, the selection of an appropriate structure for the rule consequents is analyzed. Finally, the tree construction for the determination of the rule premise structure is introduced. Strictly speaking, LOLIMOT is a learning algorithm for Takagi-Sugeno fuzzy systems. For the sake of simplicity the whole neuro-fuzzy system constructed by this algorithm in the following is called LOLIMOT as well.

1.5.1 Parameter Estimation of the Rule Consequents

The output of a Takagi-Sugeno fuzzy model is linear in the rule consequent parameters $w_{j,0}, w_{j,1}, \ldots, w_{j,xn}$. Consequently, these parameters can be estimated by a linear least-squares technique (see Golub and Loan, 1987). The following discussion concerns the estimation of the rule consequents assuming known rule premises.

For parameter estimation a global or local approach can be taken. With the *global* approach the parameters of all rule consequents are estimated in one matrix pseudo-inversion. The fuzzy model output at time instant k in (8) can be written as

$$y(k) = \underline{\psi}(k)^T \cdot \underline{w} \tag{11}$$

with the following parameter and regression vectors

$$\underline{w} = [w_{1,0} \;\; w_{1,1} \;\; \cdots \;\; w_{1,nx} \;\; \cdots \;\; w_{M,0} \;\; w_{M,1} \;\; \cdots \;\; w_{M,nx}]^T \tag{12}$$

$$\underline{\psi} = [\psi_1 \;\; x_1\psi_1 \;\; \cdots \;\; x_{nx}\psi_1 \;\; \cdots \;\; \psi_M \;\; x_1\psi_M \;\; \cdots \;\; x_{nx}\psi_M]^T \tag{13}$$

For N measured data samples the least-squares solution for the parameters is

$$\underline{w} = \left(\underline{\Psi}^T \underline{\Psi}\right)^{-1} \underline{\Psi}^T \underline{y}_d \tag{14}$$

where $\underline{y}_d = [y_d(1) \;\; y_d(2) \;\; \ldots \;\; y_d(N)]^T$ is the vector of desired model outputs, i.e. the measured process outputs, and $\underline{\Psi} = [\underline{\psi}(1) \;\; \underline{\psi}(2) \;\; \ldots \;\; \underline{\psi}(N)]^T$ is the $(N \times M \cdot (nx+1))$-dimensional regression matrix. Due to the cubic complexity $\mathcal{O}([M \cdot (nx+1)]^3)$ global parameter estimation becomes computationally expensive for fuzzy systems with many rules M. Furthermore, the number of parameters becomes large and some regularization technique, see Sjöberg, 1995, may be required in order to find a good bias/variance trade-off (see Geman et al., 1992).

The *local* parameter estimation approach does not estimate all $M \times (nx+1)$ parameters simultaneously. Rather, it divides this task into M estimations of $(nx+1)$ parameters, that is the parameters for each rule consequent are determined separately, compare Murray-Smith, 1994. This is possible because the local linear models in the rule consequents have little interaction due to the localness of the validity functions. The local estimation approach neglects the overlap between the local linear models. It is based on a separate linear regression model for each rule. The (non-weighted) output generated by rule j is

$$y(k) = \underline{\psi}_j(k)^T \cdot \underline{w}_j \tag{15}$$

with parameter and regression vectors

$$\underline{w}_j = [w_{j,0} \;\; w_{j,1} \;\; \ldots \;\; w_{j,nx}]^T \tag{16}$$

$$\underline{\psi}_j = [1 \;\; x_1 \;\; \ldots \;\; x_{nx}]^T \tag{17}$$

For estimating the parameter vector $\underline{w}_j$ of each rule consequent the data have to be weighted with their validity function $\Phi_j(\underline{z})$. Thus, data close to the center of the validity function is highly relevant for the corresponding rule conclusion while data far away in the input space is non-relevant. The centers of the validity functions can be regarded as dynamic operating points of the local linear models. With this local approach the consequent parameters of rule j can be computed as

$$\underline{w}_j = \left(\underline{\Psi}_j^T \underline{Q}_j \underline{\Psi}_j\right)^{-1} \underline{\Psi}_j^T \underline{Q}_j \underline{y}_d \tag{18}$$

where $Q_j = \text{diag}\left(\Phi_j(z(1)), \Phi_j(z(2)), \ldots, \Phi_j(z(N))\right)$ is the $(N \times N)$-dimensional diagonal weighting matrix that contains the validity values for all data samples with respect to model j. Note, that in contrast to the global estimation approach, the regression matrix $\underline{\Psi}_j = [\underline{\psi}_j(1) \;\; \underline{\psi}_j(2) \;\; \ldots \;\; \underline{\psi}_j(N)]^T$ is only of dimension $N \times (nx+1)$. Therefore, the complexity $\mathcal{O}(M \times (nx+1)^3)$ is linear in the number of rules M. Besides the huge reduction in computational demand as pointed out in Murray-Smith, 1994 the local estimation approach has some appealing properties. It forces the local linear models to fit the data locally, while this is not guaranteed (especially for large standard deviations) for global estimation due to compensation effects. Furthermore, local estimation has a regularization effect, i.e. it lowers the variance error while increasing the bias error, see Murray-Smith, 1994 for details. Due to these properties, for most applications, see Nelles and Fischer, 1996, Nelles et al., 1996, Hecker et al., 1997, and Nelles et al., 1997, the local estimation approach is not only less computational demanding but leads to better performing models as well. Therefore, the local estimation approach is pursued in this chapter.

Note, that all linear least-squares approaches optimize the prediction error, that is the difference between the desired output and the one-step-ahead prediction of the fuzzy model. However, for most applications the fuzzy model is run in parallel to the process, i.e. it is used for simulation not for prediction. Thus, the parameter estimates generally are biased in the presence of additive output noise (see Isermann, 1992). For highly disturbed processes the application of instrumental variable or extended least-squares methods is recommended (see Ljung, 1987).

1.5.2 Structure Estimation of the Rule Consequents

So far, only the estimation of the rule consequent parameters has been discussed. This subsection is devoted to structure optimization of the rule consequents. The fundamental importance of such methods stems from the fact that the dead times d_i and dynamic orders nu_i and ny in (2) usually are unknown. Typically, the dead times and orders are determined based on physical insights and a trial-and error approach. However, such a procedure becomes very tedious, especially for multi-variable processes. Thus, an automatic selection of the relevant regressors is highly desirable.

Because the local estimation approach is taken, the structure optimization is performed separately for each rule consequent. This allows to model different dead times and dynamic orders in different regions of the input space.

Since (15) is a linear regression problem, very efficient structure optimization algorithms exist, for an overview see Miller, 1990. Those methods select a subset of significant regressors from a set of possibly relevant (candidate) regressors provided by the user. Note, that the most obvious approach would be to go through all possible regressors and to choose the best combination. However, due to the combinatorial explosion of possibilities such a strategy is totally infeasible even for moderately sized problems.

One of the most simple and popular structure optimization techniques is the forward selection method with an orthogonal least-squares (OLS) algorithm. It has been successfully applied for the selection of radial basis functions, fuzzy basis functions or polynomials in Chen et al., 1991, Wang and Mendel, 1992 and Kortmann, 1989, respectively. The exact algorithm can be found in these references. It simply has to be extended to weighted least-squares.

The idea of the OLS algorithm is to start with an empty model (no regressors) and to select the regressors one by one out of the set of candidates. At each iteration the candidate which leads to the highest improvement in modeling quality (in the least-squares sense) is chosen. This procedure is stopped on some termination criterion, such as maximal model complexity (number of regressors), an error bound or some statistical test. In the first iteration, the OLS checks all candidate regressors and selects the best performing one. For example, from C candidates the regressor number i may be selected. In the second iteration, the following $C-1$ regressor combinations are possible: $(i, 1)$, $(i, 2)$, ... $(i, i-1)$, $(i, i+1)$, ..., (i, C). In principle, the best one could be found by computing all those two-parameter models. For such a strategy, at iteration j, a j-dimensional linear optimization problem would have to be solved $C-j+1$ times. The OLS avoids this vast computational demand. In each iteration all candidates are made orthogonal to the selected regressors. This directly allows to extract the improvement obtained by each candidate. In general, the OLS will not select the optimal regressors, since a maximum improvement in each iteration does not guarantee the overall global optimum, because no regressors are removed from the model. More advanced methods like stepwise regression add and discard regressors but they require a higher computational effort, see Miller, 1990. The OLS performs well in most applications.

For the application of the OLS algorithm the user estimates lower and upper bounds for the dead times and the dynamic orders from physical insights and practicability considerations. Furthermore, a termination criterion is chosen and subsequently the OLS automatically selects the significant regressors. For the termination criterion a difficulty arises from the fact, that (15) is a linear regression problem and consequently with respect to linear dynamic models, it describes a one-step prediction (series-parallel) model. If auto-regressive terms such as $y(k-i)$ are in the set of candidate regressors in almost all cases $y(k-1)$ will be selected in the first OLS iteration. This term may explain about 99% of the output since $\hat{y}(k) = y(k-1)$ is quite a good prediction model, especially if the sampling period is small. However, such a model is infeasible for simulation since it does not include any information about the process input $u(k)$. Therefore, statistical tests based on prediction error models do not provide a good stop criterion for output error models. Consequently, an upper bound on the rule consequents' complexity, i.e. a maximum number of regressors, is chosen as termination criterion in this chapter.

1.5.3 *Estimation of the Rule Premise Structure*

The previous two subsections discussed the parameter and structure optimization of the rule consequents under the assumption of a known premise structure. The determination of the premise structure can be regarded as a much more difficult problem because it is highly nonlinear. Several strategies have been introduced in order to solve this problem, see Sugeno and Kang, 1988, Johansen and Foss, 1993, and Babuška and Verbruggen, 1996. Here the new approach proposed in Nelles and Isermann, 1996 is applied. It is based on a tree-construction algorithm similar to the ideas in Johansen and Foss, 1993 or Friedman, 1991. Because this local linear model tree (LOLIMOT) algorithm utilizes local linear parameter estimation and optionally local OLS it leads to

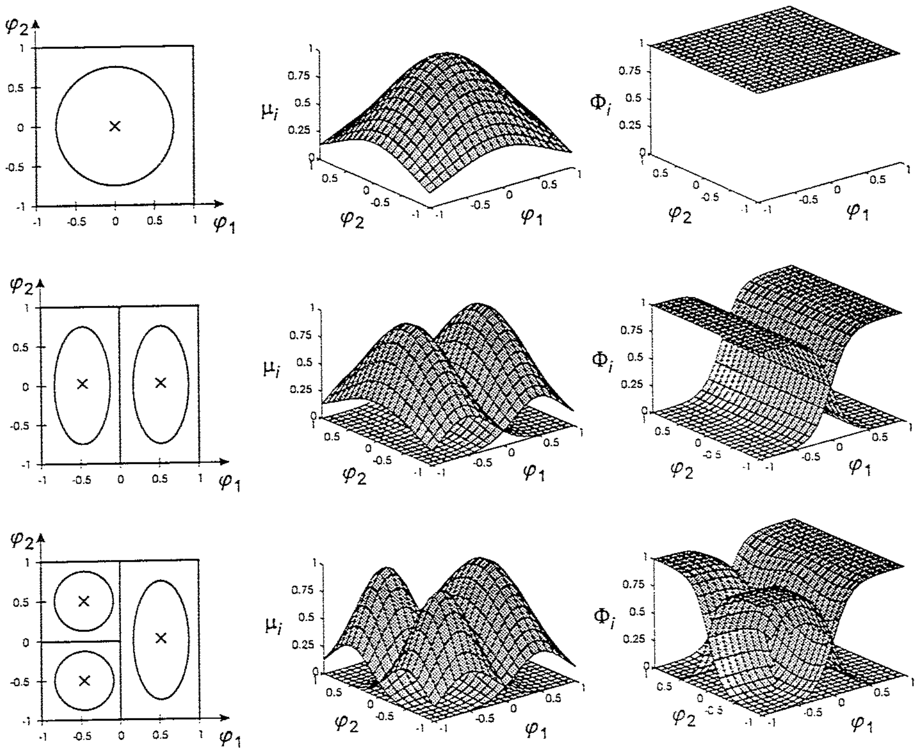

Figure 1.11 The Gaussian membership functions and the normalized validity functions for three different partitionings of the input space.

very fast training.

The LOLIMOT algorithm partitions the input space in hyper-rectangles. Each local linear model or each rule respectively belongs to one hyper-rectangle in whose center the validity function is placed. The standard deviations are set proportional to the size of the hyper-rectangle. This makes the size of the validity region of a local linear model proportional to its hyper-rectangle extension. A local linear model may be valid over a wide operating range of one input variable but only in a small area of another one. Figure 1.11 illustrates the (two-dimensional) Gaussian membership functions μ_i and the normalized Gaussian validity functions Φ_i for three different partitionings of the input space.

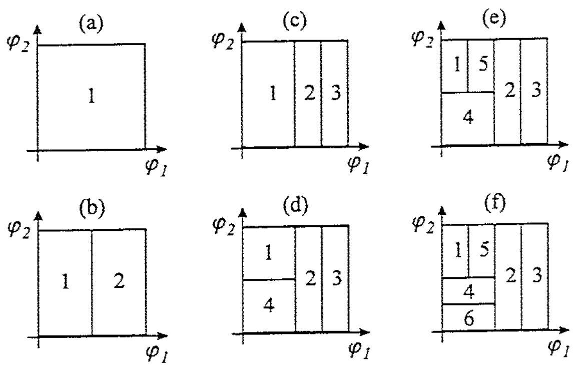

Figure 1.12 Six iterations of the LOLIMOT algorithm.

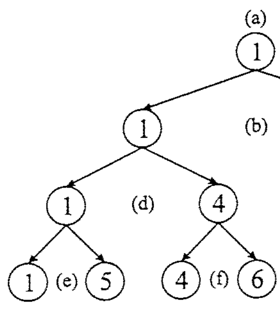

Figure 1.13 Tree structure of Fig. 1.12.

The structure of the hyper-rectangles is constructed as follows, compare Fig. 1.12. The LOLIMOT algorithm starts with one (global) linear model, that is with a single rule. In each iteration one rule is added to the Takagi-Sugeno fuzzy model until some termination criterion is met. A new rule is generated by dividing the "worst" performing local linear model into two halves. The "worst" performing local linear model is found by a comparison of the local error measures that are calculated by weighting the fuzzy model output error with the corresponding validity functions. Axes-orthogonal cuts in all dimensions are tested and the one with the highest performance improvement is chosen. Figure 1.12 shows six iterations of the LOLIMOT algorithm for a two-dimensional input space ($nz = 2$). It can be represented in a tree structure, see Fig. 1.13.

The following features make the LOLIMOT algorithm very fast. Because at each iteration only the worst performing local linear model is considered for division, the number of model structures investigated at each iteration remains constant. Furthermore, only $2nz$ rule consequents have to be optimized in each iteration since for each of the nz cuts two local linear models must be determined. This follows directly from the local estimation philosophy. The parameters of all other rule consequents remain unchanged. Consider e.g. Fig. 1.12 (d): Here model 1 is assumed to be the worst performing and it is divided into two halves in z_1- and z_2-dimension respectively. For each of both alternatives the two rule consequents are optimized according to the previous two subsections leaving the rules 2, 3 and 4 unchanged. The division in z_1 has lead to a smaller output error of the Takagi-Sugeno fuzzy model and is selected as can be seen in Fig. 1.12 (e).

The LOLIMOT algorithm is especially well suited for the identification of dynamic systems since the criterion for structure optimization is the output error. Therefore, an accumulation of errors or even instability due to the feedback is detected and the corresponding model can be avoided. Furthermore, overfitting can be detected because the evaluation of the model's performance implies generalization since it is run as a parallel model while parameter tuning applies a series-parallel model. Another feature of LOLIMOT is its linear extrapolation behavior that in the authors' experience is advantageous with respect to dynamic systems compared to the constant extrapolation behavior of multi-layer perceptrons or normalized radial basis function networks.

1.5.4 Interpretation of Takagi-Sugeno Fuzzy Models

Takagi-Sugeno fuzzy models for nonlinear dynamic processes offer a very good interpretability. Each rule realizes a local linear model by its rule consequent while the premise constructs the corresponding validity function. Each local linear model represents a discrete time transfer function where the dynamic operating point is described by the validity function's center. Thus, the user can easily extract local gains, poles and zeros form these transfer functions what often gives important insights into the process behavior.

The rule structure generated by the LOLIMOT algorithm reveals information about the type of nonlinearity. For example if the output depends highly nonlinear on an input variable many cuts in this dimension can be expected. Moreover, the distribution of the local linear models illustrates the highly nonlinear and basically linear regions in the input space. Because this approach offers an extremely good interpretability the incorporation of prior knowledge in many forms is remarkably easy. For example, the vector z, which spans the space for the validity functions, often can be chosen as a small subset of the elements in x. The more restrictions are imposed on z from prior knowledge, the faster the training of the fuzzy model becomes.

1.6 IDENTIFICATION OF A COMBUSTION ENGINE TURBOCHARGER

In this and the next section the previously introduced LOLIMOT algorithm is applied for identification of real-world processes. The following features of LOLIMOT are demonstrated:

- Fast training.
- Little user interaction leads to fast overall development times.
- Easy choice of model complexity, that is number of rules.
- Good interpretability allows the user to gain confindence in the model.

Because the combustion engine turbocharger is a multivariable process it is important to consider some general properties of multivariable modeling according to (1) and (2). Models according to (1) and (2) incorporate a feedback of previous outputs $y(k-i)$, i.e. they have an autoregressive part. This means, that all m physical process inputs $u_1, u_2, ..., u_m$ share the same "denominator" dynamics. This is equivalent to linear MISO systems where the denominators of the transfer functions from all inputs to the same output are chosen to be identical in order to apply a linear least-squares parameter estimation. The restrictions imposed by this kind of model can be compensated by increasing the dynamic orders nu_i and ny.

1.6.1 Process Description

Figure 1.14 schematically represents the charging process of a Diesel engine by an exhaust turbocharger. The exhaust enthalpy is used by the turbine to drive the compressor, which aspirates and precompresses the fresh air in the cylinder. Thus the turbocharger allows a higher compression ratio increasing the power of the engine while its stroke volume remains the same. This is important in the middle speed range. The charging process has a nonlinear input/output behavior as well as a strong dependency of the dynamic parameters on the operating point. This is known by physical insights and can be observed by the inferior performance of a linear process model.

In general, the static behavior of the turbocharger may be sufficiently described by characteristic maps (look-up tables) of compressor and turbine. However, if the dynamics of the turbocharger need to be considered, basic mechanical and thermodynamical modeling is required, see Zinner, 1985, Boy, 1980, and Pucher, 1985. Practical applications have shown that these methods are capable of reproducing the characteristic dynamic behavior of the turbocharger. The model quality, however, essentially depends on the accurate

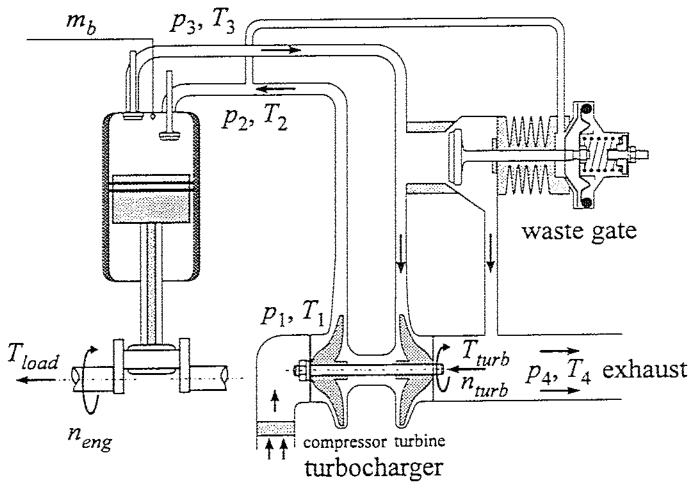

Figure 1.14 Scheme of the combustion engine turbocharger.

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. Here, simple, easy-to-identify input/output models suitable for real-time simulation are required. Therefore, in the following a novel approach for the development of a dynamic model of an exhaust turbocharger will be proposed utilizing the LOLIMOT approach and complying with the demanded properties to a high degree. Only the recorded data of input/output measurements of the real process are required. Thus, no theoretical knowledge of the process is necessary. For a hardware-in-the-loop simulation, the rate of injection mb 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 was generated by a special driving cycle to excite the system with all amplitudes and frequencies of interest, see Fig. 1.15. The measurements were recorded on a flat test track. Also, in order to operate the engine in high load ranges the truck was driven with the biggest possible load. For validation, special driving cycles were recorded, which reproduce realistic conditions in urban (see Fig. 1.16) and interstate traffic (see Fig. 1.17).

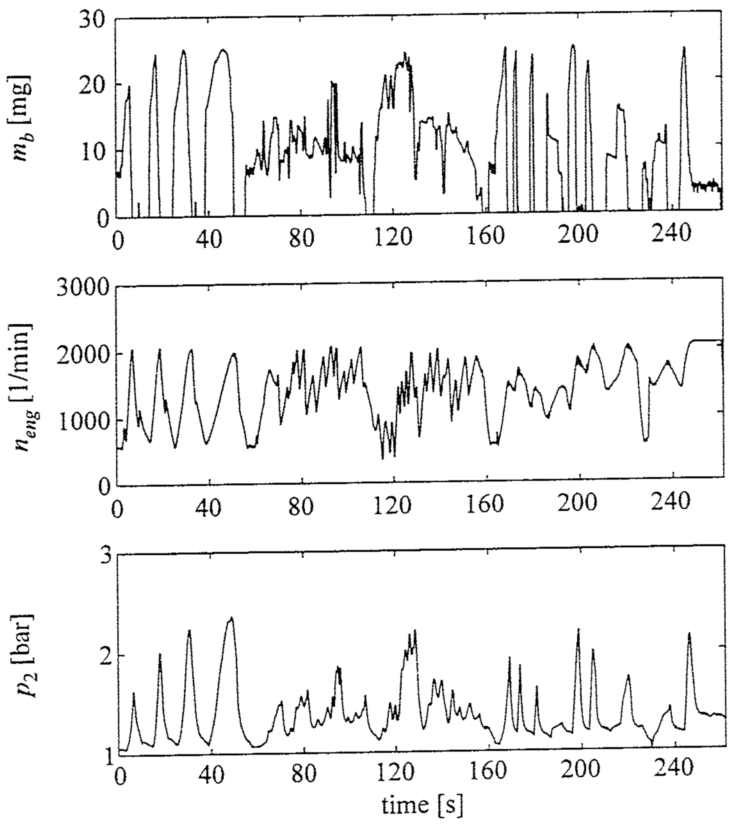

Figure 1.15 Training data for the turbocharger.

1.6.2 Identification with LOLIMOT

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

$$\begin{aligned} p_2(k) &= f\,(m_b(k-1),\, m_b(k-2),\, n_{eng}(k-1),\, n_{eng}(k-2), \\ &\quad p_2(k-1),\, p_2(k-2)) \end{aligned} \tag{19}$$

LOLIMOT was trained for 20 local linear models. Figure 1.18 shows the decrease of the output error in dependency of the number of local linear models, i.e. the iterations of LOLIMOT. Since for more than ten local linear models

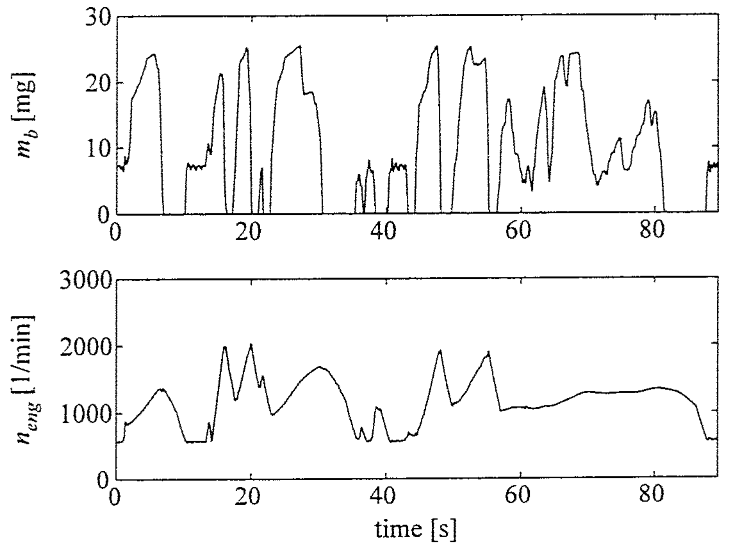

Figure 1.16 Urban traffic validation data.

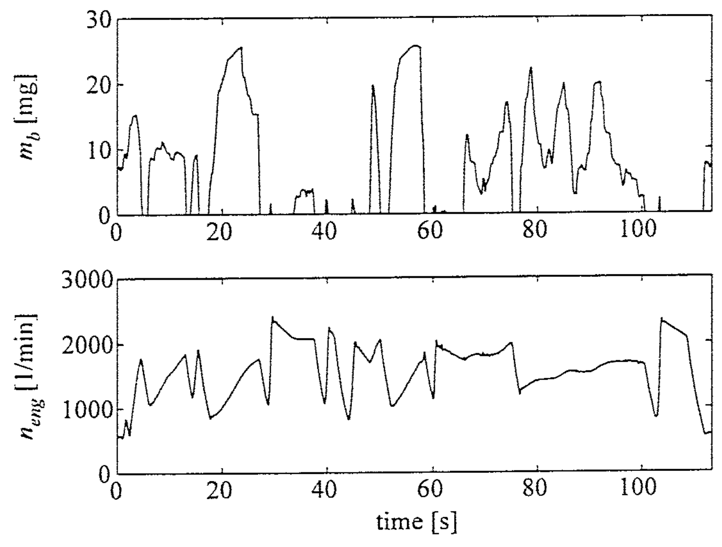

Figure 1.17 Interstate traffic validation data.

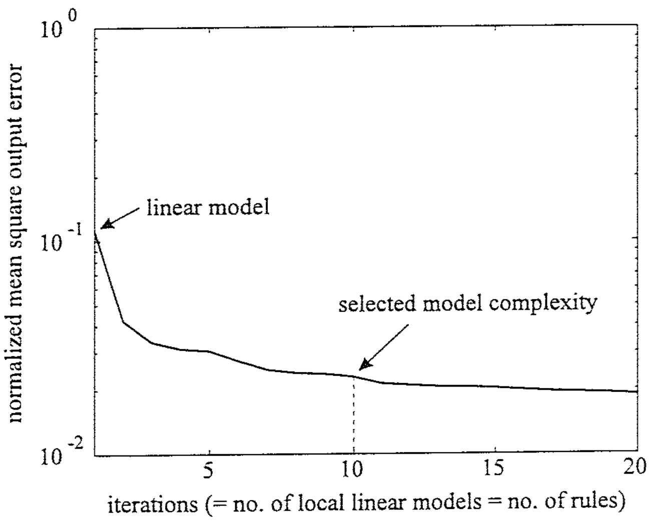

Figure 1.18 Rate of convergence of LOLIMOT.

no substantial improvement could be reached by a further increase in model complexity this LOLIMOT neuro-fuzzy model is selected.

It is interesting to note that LOLIMOT performs cuts only in the physical input dimensions, that is $m_b(k-1)$ and $n_{eng}(k-1)$ but not in the physical outputs $p_2(k-1)$ and $p_2(k-2)$. As shown in Nelles and Isermann, 1996 this reveals that the turbocharger can be approximately modeled by a Hammerstein structure, that is a static nonlinearity followed by a linear dynamic system. Although this information is not necessary for the construction of the LOLIMOT neuro-fuzzy model, it can be exploited by reducing the dimensionality of the validity functions' input space. Then the input space of the local linear models still contains all regressors: $\underline{x}(k) = [m_b(k-1)\ m_b(k-2)\ n_{eng}(k-1)\ n_{eng}(k-2)\ p_2(k-1)\ p_2(k-2)]^T$, while the input space of the validity functions contains only: $\underline{z}(k) = [m_b(k-1)\ n_{eng}(k-1)]^T$. In this case, LOLIMOT trains three times faster because it has to partition an only two-dimensional instead of a six-dimensional input space. The training with incorporation of this prior knowledge takes 7.5s on a Pentium 133MHz PC.

1.6.3 Results

Figures 1.19 and 1.20 and 1.21 show the performance of the LOLIMOT neuro-fuzzy model on the training data and on the urban and interstate traffic validation data, respectively. The model is operating in parallel to the process (simulation). The process and model output are almost indistinguishable in these figures with a maximum output error below 0.1 bar.

An appealing feature of the LOLIMOT neuro-fuzzy model is the interpretability of the obtained results. All ten local linear models have one slow and one fast stable real pole. The fast poles range from 0.12 to 0.37 (representing time constants between 0.01s and 0.02s) and the slow poles range from 0.71 to 0.84 (representing time constants between 0.58s and 1.15s) in the z-domain. These time constants are consistent with the physical knowledge about the turbocharger. Inspecting the poles of the local linear models not only tells something about the dynamics of the process and about a possible operating point dependency of the dynamics. It also can indicate overfitting, that is a too flexible model, if e.g. unstable poles or poles on the negative real axis which have no counterpart in continuous time are identified. The highest gain of all local linear models is five times higher than the lowest gain. This underlines the strongly nonlinear static and dynamic process characteristics. It is interesting to note that LOLIMOT models with higher dynamic order or with significantly more local linear models reveal overfitting since they typically have non-interpretable poles between -1 and 0 (i.e. without corresponding poles in continuous time) in the z-domain.

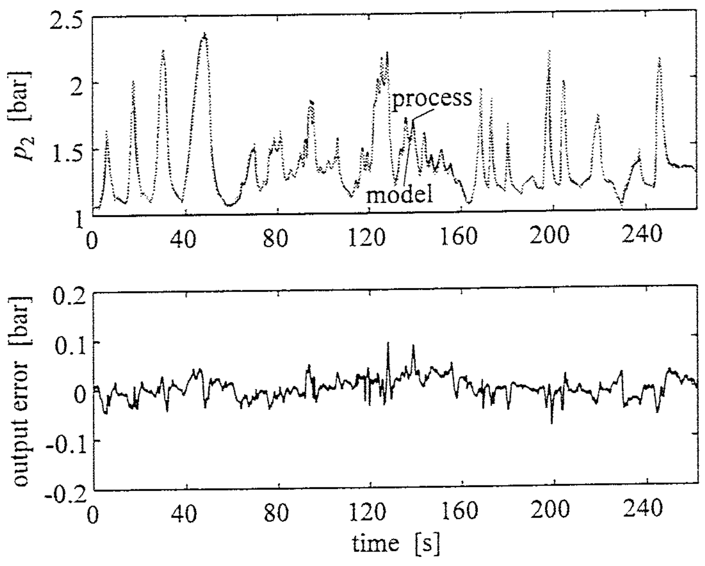

Figure 1.19 Fuzzy model performance on the training data.

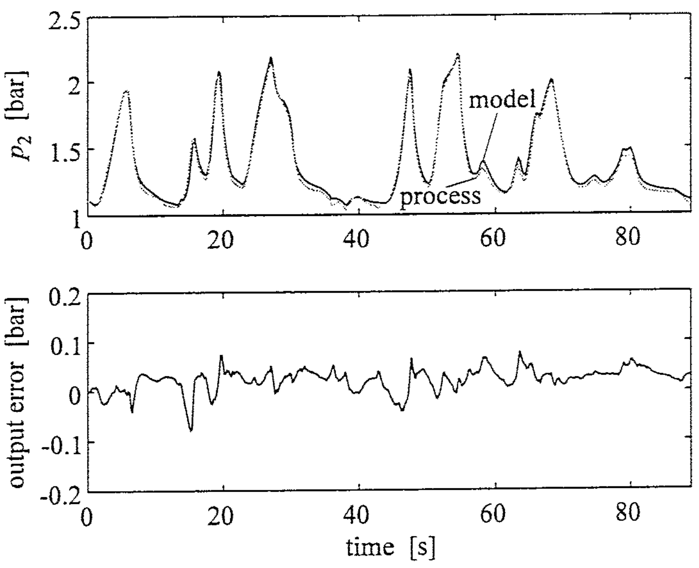

Figure 1.20 Fuzzy model performance on the urban traffic validation data.

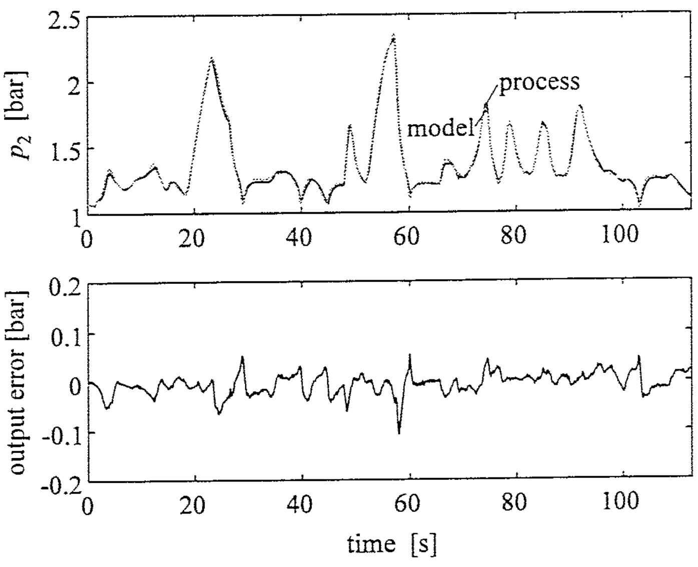

Figure 1.21 Fuzzy model performance on the interstate traffic validation data.

1.7 IDENTIFICATION OF A TRANSPORT DELAY PROCESS

The transport delay process belongs to an industrial-scale heat exchanger pilot plant. It is very well suited in order to illustrate the automatical structure selection of the rule conclusions by the orthogonal least-squares (OLS) algorithm. Although the process has operating point dependent dead times and time constants it is easy to understand. The fuzzy model identified with LOLIMOT allows a good interpretation and is conform with physical insights about the process.

In addition to the previous application the following features of LOLIMOT are demonstrated in this section:

- Incorporation of prior knowledge by the selection of different variables (regressors) for the rule premises and rule consequents.
- Automatic selection of significant (important) variables (regressors) in the rule consequents.

1.7.1 Process Description

The performance of the proposed approach is demonstrated in the following by modeling the transport of heated water within a pipe.

The process under investigation is part of the pilot plant depicted in Fig. 1.22. The plant consists of a closed water loop which is heated on one side by a steam heated heat exchanger and cooled on the opposite side by an air cooler. The amount of heat supplied to the water can be controlled by changing the steam flow. The amount of heat taken away is controlled by changing the speed of the air fan. Furthermore, the water flow in the pipe system can be changed in a wide range.

Now, the task is to model the temperature T_{out} at the steam heated heat exchanger inlet dependent on the temperature T_{in} at the cooler outlet and the water flow.

Two different measurement data sets were acquired for training and validation. Each has a duration of 120min. (For the sake of clarity only intervals of 60min length are shown in the following figures.) The sampling time was $T_0 = 1\text{s}$ in all experiments.

1.7.2 Identification with LOLIMOT

From simple physical considerations the following basic relations are known:

- The temperature T_{out} is, with a certain time delay, about equal to the temperature T_{in}. Thus, the static gain of the process can be assumed

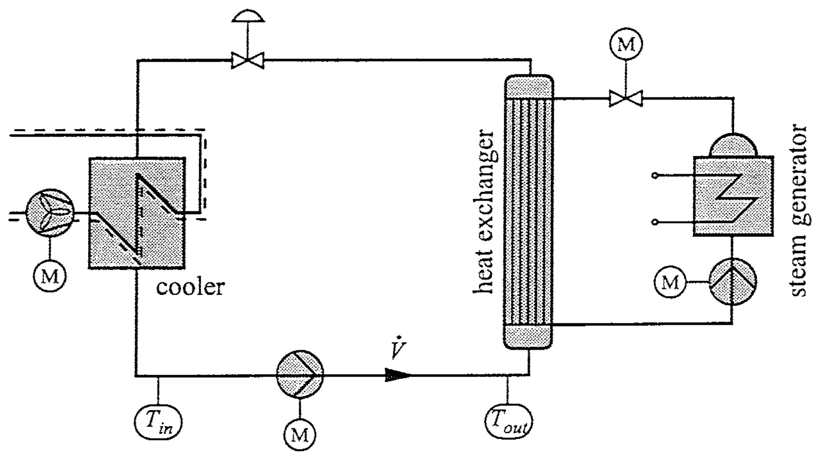

Figure 1.22 Scheme of the transport delay process which is part of a heat exchanger.

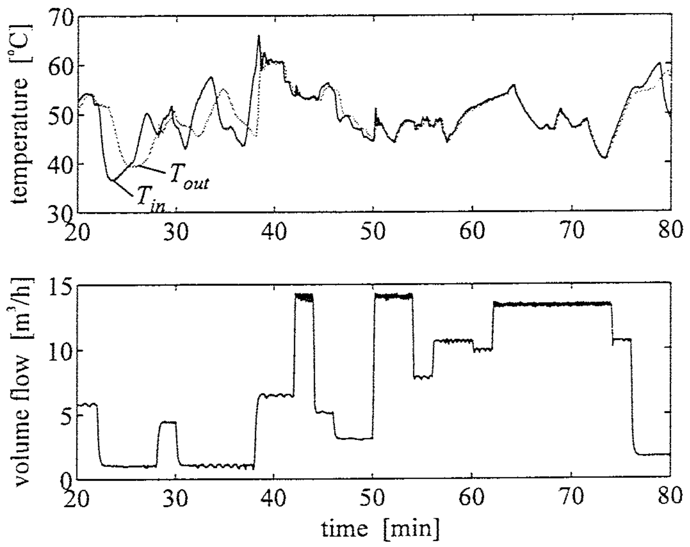

Figure 1.23 An one hour extract of the two hours training data set.

to be around one. In fact, a value slightly smaller than one has to be expected because the water temperature is higher than the environment temperature and some energy will be lost to the environment.

- The flow rate of the water determines the dead time (large dead time at low flow rates, small dead time at high flow rates). These relations are also very obvious from Fig. 1.23.
- Especially at low flow rates deviations from a pure dead time behavior can be seen which are probably due to energy storage in the water pipe. Thus, a first order time-lag model is reasonable.

This analysis implies the following model structure:

$$T_{out}(k) = f\left(\dot{V}(k-1), T_{in}(k-1), T_{in}(k-2), \ldots, T_{in}(k-d_{max}), T_{out}(k-1)\right) \tag{20}$$

From Fig. 1.23 the maximum dead time d_{max} can be estimated to be smaller than 100s. The function $f(\cdot)$ is assumed to depend only on the flow rate. Therefore the input space of the validity functions $\Phi(\cdot)$ in (8) is one-dimensional ($nz = 1$), i.e. $\underline{z}(k) = \dot{V}(k-1)$. For the rule conclusions the following 16 terms are presented to the structure determination algorithm:

$$\begin{aligned} \underline{x}(k) \quad = \quad & [\, T_{in}(k-1),\ T_{in}(k-2),\ T_{in}(k-4),\ T_{in}(k-8),\ T_{in}(k-16), \\ & T_{in}(k-24),\ T_{in}(k-32),\ \ldots,\ T_{in}(k-96),\ T_{out}(k-1)\,]^T \end{aligned} \tag{21}$$

In principle, all terms with dead times varying from 1 to 100 could be presented to the OLS structure determination algorithm. From previous experience, however, this leads only to very small improvements in the model accuracy at the cost of significantly increased training effort. The OLS algorithm was allowed to select two regressors out of the terms given in (21) for each rule conclusion. Therefore, only two parameters have to be estimated for each rule. Note, that no offset is modelled because it is clear from physical insights that there exists no offset between T_{in} and T_{out}.

Within one minute on a Pentium 133MHz PC all LOLIMOT models with 1, 2, ..., 10 rules are trained. Figure 1.24 shows that starting with a model consisting of five rules further intersections do not lead to significant improvements. Therefore this model is selected.

1.7.3 Results

It comprises the membership functions depicted in Fig. 1.25 and the following local linear models for the rule conclusions:

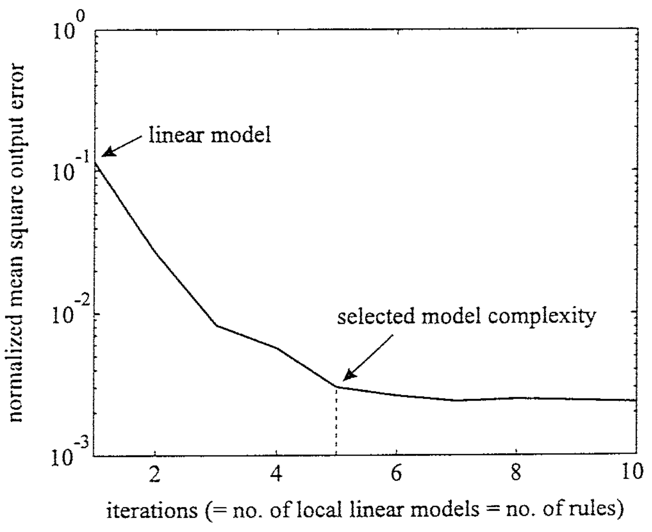

Figure 1.24 Rate of convergence of LOLIMOT.

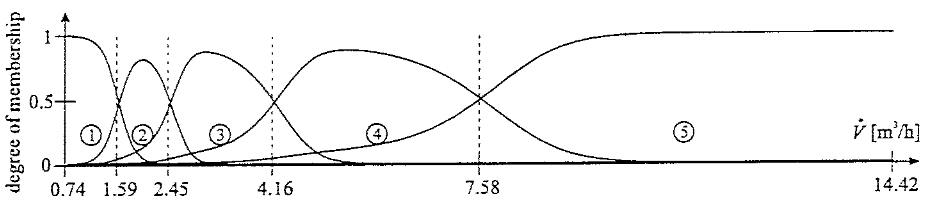

Figure 1.25 Membership functions for the volume flow.

1. IF $\dot{V}$ is extremely small THEN
$$T_{out}(k) = 0.0165\,T_{in}(k-48) + 0.9835\,T_{out}(k-1)$$
2. IF $\dot{V}$ is very small THEN
$$T_{out}(k) = 0.0229\,T_{in}(k-32) + 0.9769\,T_{out}(k-1)$$
3. IF $\dot{V}$ is small THEN
$$T_{out}(k) = 0.0231\,T_{in}(k-16) + 0.9767\,T_{out}(k-1)$$
4. IF $\dot{V}$ is medium THEN
$$T_{out}(k) = 0.0677\,T_{in}(k-8) + 0.9320\,T_{out}(k-1)$$
5. IF $\dot{V}$ is large THEN
$$T_{out}(k) = 0.1212\,T_{in}(k-4) + 0.8783\,T_{out}(k-1)$$

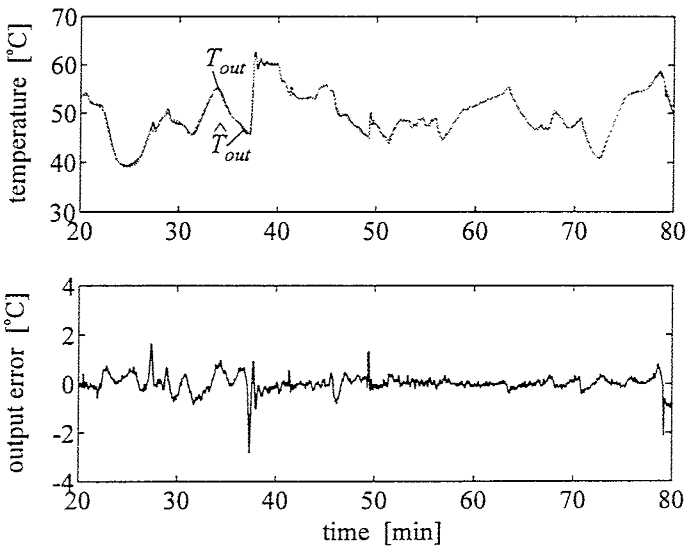

Figure 1.26 Performance on the training data.

This result can be easily interpreted. LOLIMOT has recognized that the dead times and time constants change much stronger at low flow rates than at higher ones. This can be seen from the intersections of the flow rate in Fig. 1.25 (fine divisions for low flow rates, coarse divisions for high flow rates). For all local linear models, the OLS algorithm selects one delayed input temperature and the previous output temperature. The dead times change in a wide interval and are – as expected – large for low flow rates and small for high flow rates. The same holds for the time constants (computed out of the poles of the local linear models), which vary between 60s (model 1) and 7.7s (model 5). For all local linear models a gain slightly smaller than one was identified, which as well corresponds to the physical insights. Figures 1.26 and 1.27 show the identification performance of the model operating in parallel to the process (simulation).

Figure 1.28 depicts three responses of the LOLIMOT model for a step of T_{in} from 45°C to 55°C. It can be clearly seen that the time constants and the dead times decrease for increasing flow rates. In the static operating points T_{out} is slightly smaller than T_{in} as expected due to a energy loss during the water transport. However, from physical considerations the curve for $\dot{V} = 1\text{m}^3/\text{h}$ should be below the plot for $\dot{V} = 3\text{m}^3/\text{h}$. This small modeling error is probably caused by insufficient static excitation for very low flow rates in

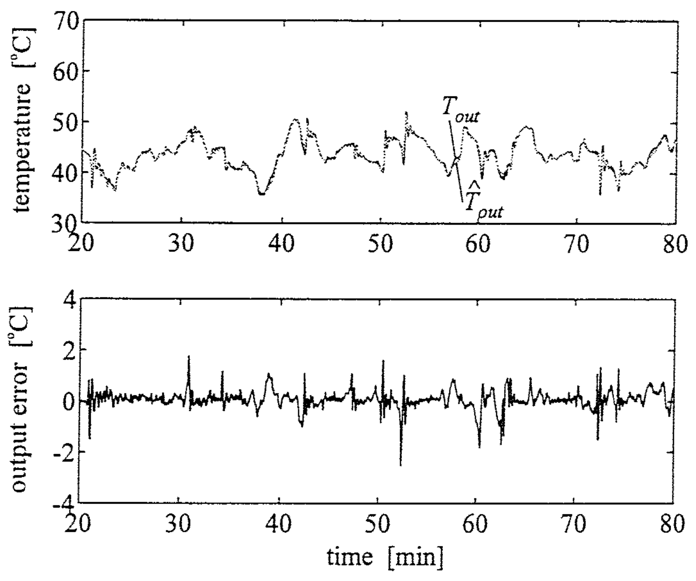

Figure 1.27 Performance on the validation data.

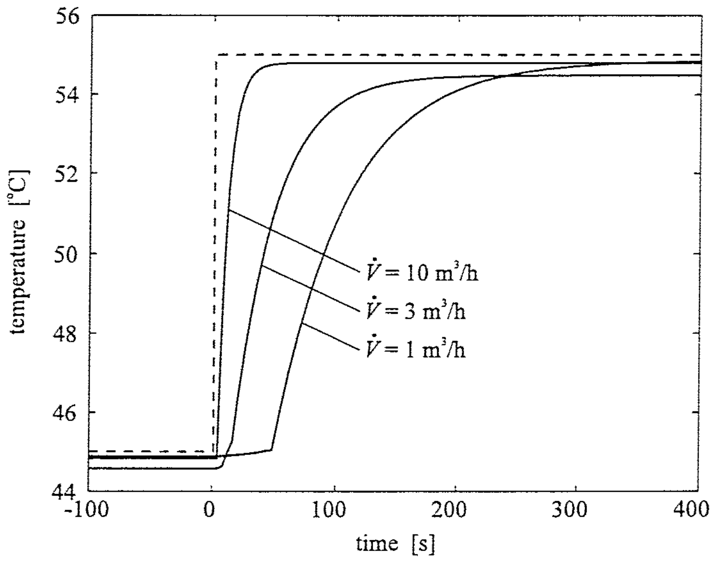

Figure 1.28 Step responses of the transport delay LOLIMOT model for three different water flows: $\dot{V} = 1\text{m}^3/\text{h}$, $\dot{V} = 3\text{m}^3/\text{h}$, and $\dot{V} = 10\text{m}^3/\text{h}$.

the training data because the time constants and dead times are very large. Another interesting aspect can be observed by examining the LOLIMOT model behavior of the step response with $\dot{V} = 1\text{m}^3/\text{h}$. The dead time is around 48s (compare the rules). The other four rules represent smaller dead times. Because *all* rules are active, the model's response is not exactly zero for 48 sampling periods. Nevertheless, this effect is negligible since the validity values of these rules are very small.

The strength of the proposed approach becomes very obvious if the development time is taken into account until the final model was determined. All results presented in this section were gained within half a day after the experimental data were acquired. The first model which performed only a little worse than the final result was gained after only one hour. This is due to the high degree of automation of the proposed approach.

1.8 CONCLUSIONS

The purpose of this chaper has been twofold. On the one hand an overview on neuro-fuzzy modeling techniques has been given. On the other hand a special type of neuro-fuzzy model (LOLIMOT) has been discussed in detail and was applied to complex real-world processes. The local linear model tree approach constructs Takagi-Sugeno fuzzy models from measurement data. It combines efficient local linear parameter estimation or subset selection schemes for optimization of the rule consequents with a tree construction algorithm for structure optimization of the rule premises. This method has been applied to identification of a combustion engine turbocharger and a transport delay process which belongs to an industrial-scale thermal plant. The obtained results underline the important features of LOLIMOT such as: fast training, little user interaction which implies short development times, easy choice of the model complexity and good interpretability of the obtained models.

In future research further improvements can be expected if more advanced techniques known from linear dynamic system identification are transferred to the LOLIMOT neuro-fuzzy model. Thus, it is possible to utilize instrumental variable methods and generalized orthonormal basis functions, see Nelles, 1997. Another important goal for future research is the comparision of alternative modeling approaches. For first steps towards this goal refer to Isermann et al., 1997 which includes a comparison of LOLIMOT with internal dynamic neural networks, i.e. networks where dynamic elements are built into the network structure.

References

Babuška, R. and Verbruggen, H. (1996). An overview of fuzzy modeling for control. *Control Engineering Practice*, 4(11).

Battiti, R. (1996). *Reactive Search: Toward Self-Tuning Heuristics in Modern Heuristic Search Methods.* John Wiley & Sons, London.

Bellman, R. (1961). *Adaptive Control Processes.* Princeton University Press, N.J.

Bossley, K. (1996). Neurofuzzy construction algorithms. *ISIS Technical Report ISIS-TR1.*

Boy, P. (1980). *Beitrag zur Berechnung des instationären Betriebsverhaltens von mittelschnelllaufenden Schiffsdieselmotoren.* Dissertationen TU Hannover, Hannover.

Brown, M., Bossley, K., Mills, D., and C.J.Harris (1995). High dimensional neurofuzzy systems: Overcoming the curse of dimensionality. *FUZZ-IEEE/IFES.*

Brown, M. and Harris, C. (1994). *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, New York.

Chen, S., Cowan, C., and Grant, P. (1991). Orthogonal least-squares learning algorithm for radial basis function networks. *IEEE Transacations on Neural Networks*, 2(2).

Davis, L. (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

Dickerson, J. and Kosko, B. (1996). Fuzzy function approximation with ellipsoidal rules. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 26(4).

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics.*

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4.

Golub, G. and Loan, C. V. (1987). *Matrix Computations.* Mathematical Sciences. The Johns Hopkins University Press.

Halgamuge, S. and Glesner, M. (1994). Fuzzy neural fusion techniques for industrial applications. In *ACM Symposium on Applied Computing*, Phoenix.

Hecker, O., Nelles, O., and Moseler, O. (1997). Nonlinear system identification and predictive control of a heat exchanger based on local linear model trees. In *American Control Conference (ACC)*, Albuquerque.

Hohensohn, J. and Mendel, J. (1994). Two-pass orthogonal least-squares algorihtm to train and reduce fuzzy logic systems. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, New Orleans.

Hunt, K., Haas, R., and Murray-Smith, R. (1996). Extending the functional equivalence of radial basis functions networks and fuzzy infernce systems. *IEEE Transactions on Neural Networks*, 7(3).

Ingber, L. and Rosen, R. (1992). Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical Computer Modelling*, 16:87–100.

Isermann, R. (1992). *Identifikation dynamischer Syteme - Band 1*. Springer, Berlin.

Isermann, R., Ernst, S., and Nelles, O. (1997). Identification with dynamic neural networks — architectures, comparisons, applications —, plenary. In *IFAC Symposium on System Identification (SYSID)*, Fukuoka.

Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1994). Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms. *Fuzzy Sets & Systems*, 65.

Johansen, T. (1996). Robust identification of takagi-sugeno-kang fuzzy models using regularization. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, New Orleans.

Johansen, T. and Foss, B. (1993). Constructing narmax models using armax models. *International Journal of Control*, 58(5).

Kalvi, T. (1993). Asmod - an algorithm for adaptive spline modelling of observation data. *International Journal of Control*, 58(4).

Kim, H. and Mendel, J. (1995). Fuzzy basis functions: Comparisons with other basis functions. *IEEE Transactions on Fuzzy Systems*, 3(2).

Kortmann, M. (1989). *Die Identifikation nichtlinearer Ein- und Mehrgr ensysteme auf der Basis nichtlinearer Modellans tze*. Number 177. VDI Verlag, Reihe 8: Me -, Steuerungs- und Regelungstechnik, Düsseldorf.

Kroll, A. (1996). Identification of functional fuzzy models using multidimensional reference fuzzy sets. *Fuzzy Sets & Systems*, 80.

Krone, A. (1996). Advanced rule reduction concepts for optimizing efficiency of knowledge extraction. In *European Congress on Intelligent Techniques and Soft Computing (EUFIT)*, Aachen.

Leontaritis, I. and Billings, S. (1985). Input-output parametric models for nonlinear systems, part 1: Deterministic nonlinear systems. *International Journal of Control*, 41.

Lindskog, P. (1996). *Methods, Algorithms and Tools for System Identification Based on Prior Knowledge*. Number 436. Link ping Studies in Science and Technology. Dissertations, Linköping.

Ljung, L. (1987). *System Identification — Theory for the User*. Prentice-Hall, Englewood Cliffs, N.J.

Mendel, J. (1995). Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE*, 83(3).

Miller, A. (1990). *Subset Selection in Regression.* Statistics and Applied Probability. Chapman and Hall, London.

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

Nauck, D. and Kruse, R. (1993). Fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In *IEEE International Conference on Neural Networks (ICNN)*, San Francisco.

Nelles, O. (1997). Orthonormal basis functions for nonlinear system identification with local linear model trees (lolimot). In *IFAC Symposium on System Identification (SYSID)*, Fukuoka.

Nelles, O. and Fischer, M. (1996). Local linear model trees (lolimot) for nonlinear system identification of a cooling blast. In *European Congress on Intelligent Techniques and Soft Computing (EUFIT)*, Aachen.

Nelles, O., Hecker, O., and Isermann, R. (1997). Automatic model selection in local linear model trees for nonlinear system identification of a transport delay process. In *IFAC Symposium on System Identification (SYSID)*, Fukuoka.

Nelles, O. and Isermann, R. (1996). Basis function networks for interpolation of local linear models. In *IEEE Conference on Decision and Control (CDC)*, Kobe.

Nelles, O., Sinsel, S., and Isermann, R. (1996). Local basis function networks for identification of a turbocharger. In *IEE UKACC International Conference on Control*. Exeter.

Pedrycz, W., editor (1997). *Fuzzy Evolutionary Computation.* Kluwer Academic Publishers.

Pucher, H. (1985). *Aufladung von Verbrennungsmotoren.* Expert-Verlag, Sindelfingen.

Raju, G. and Zhou, J. (1993). Adaptive hierarchical fuzzy controller. *IEEE Transactions on Systems, Man and Cybernetics*, 23(4).

Sjöberg, J. (1995). *Non-linear System Identification with Neural Networks.* Number 381. Link ping Studies in Science and Technology. Dissertations, Linköping.

Sugeno, M. and Kang, G. (1988). Structure identification of fuzzy model. *Fuzzy Sets & Systems*, 28(1).

Wang, L. and Langari, R. (1995). Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques. *IEEE Transactions on Fuzzy Systems*, 3(4).

Wang, L.-X. and Mendel, J. (1992). Fuzzy basis function, universal approximation and orthogonal least-squares learning. *IEEE Transactions on Neural Networks*, 3(5).

Werntges, H. (1993). Partitions of unity improve neural function approximators. In *IEEE International Conference on Neural Networks (ICNN)*, San Francisco.

Zinner, K. (1985). *Aufladung von Verbrennungsmotoren*. Springer, Berlin.