Chapter 2
Interpretability of Hinging Hyperplanes

The hinging hyperplane model was proposed by Breiman [20]. This type of nonlinear model is often referenced in the literature since it suffers from convergency and range problems [19, 33–35]. Methods such as a penalty of the hinging angle were proposed to improve Breiman's algorithm [18]; alternatively, the Gauss-Newton algorithm can be used to obtain the final nonlinear model [34]. Several application examples have also been published in the literature; e.g., it can be used in the identification of piecewise affine systems via mixed-integer programming [36], and this model also lends itself to forming hierarchical models [19].

In this chapter a much more applicable algorithm is proposed for hinging hyperplane identification. The key idea is that in a special case ($c = 2$), the fuzzy c-regression method (FCRM) [37] can be used for identifying hinging hyperplane models. To ensure that two local linear models used by the fuzzy c-regression algorithm form a hinging hyperplane function, it has to be guaranteed that local models intersect each other in the operating regime of the model. The proposed constrained FCRM algorithm is able to identify one hinging hyperplane model; therefore, to generate more complex regression trees, the described method should be recursively applied. Hinging hyperplane models containing two linear submodels divide the operating region of the model into two parts, since hinging hyperplane functions define a linear separating function in the input space of the hinging hyperplane function. These separations result in a regression tree where branches correspond to linear divisions of the operating regime based on the hinge of the hyperplanes at a given node. This type of partitioning can be considered as the crisp version of a fuzzy regression-based tree described in [38]. Fortunately, in the case of a hinging hyperplane-based regression tree there is no need to select the best splitting variable at a given node, but, on the other hand, it is not as interpretable as regression trees utilizing univariate decisions at nodes.

To support the analysis and building of this special model structure, novel model performance and complexity measures are presented in this work. Special attention is given to modeling and controlling nonlinear dynamical systems. Therefore, an application example related to the Box-Jenkins gas furnace benchmark identification

T. Kenesei and J. Abonyi, *Interpretability of Computational Intelligence-Based Regression Models*, SpringerBriefs in Computer Science,
DOI 10.1007/978-3-319-21942-4_2

problem is added. It will also be shown, that thanks to the piecewise linear model structure, the resulting regression tree can be easily utilized in model predictive control. A detailed application example related to the model predictive control of a water heater will demonstrate the benefits of the proposed framework.

A critical step in the application of model-based control is the development of a suitable model for the process dynamics. This difficulty stems from lack of knowledge or understanding of the process to be controlled. Fuzzy modeling has been proven to be effective for the approximation of uncertain nonlinear processes. Recently, nonlinear black-box techniques using fuzzy and neuro-fuzzy modeling have received a great deal of attention [39]. Readers interested in industrial applications can find an excellent overview in [40]. Details of relevant model-based control applications are well presented in [41, 42].

Most nonlinear identification methods are based on the NARX (Nonlinear AutoRegressive with eXogenous input) model [8]. The use of NARX black-box models for high-order dynamic processes in some cases are impractical. Data-driven identification techniques alone may yield unrealistic NARX models in terms of steady-state characteristics, local behavior and unreliable parameter values. Moreover, the identified model can exhibit regimes which are not found in the original system [42]. This is typically due to insufficient information content of the identification data and the overparametrization of the model. This problem can be remedied by incorporating prior knowledge into the identification method by constraining the parameters of the model [43]. Another way to reduce the effects of overparametrization is to restrict the structure of the NARX model, using, for instance, the Nonlinear Additive AutoRegressive with eXogenous input (NAARX) model [44]. In this book a different approach is proposed; a hierarchical set of local linear models are identified to handle complex systems dynamics.

Operating regime-based modeling is a widely applied technique for identification of these nonlinear systems. There are two approaches building operating regime-based models. An additive model uses the sum of certain basis functions to represent a non-linear system, while partitioning approach partitions the input space recursively to increase modeling accuracy locally [18]. Models generated by this approach are often represented by trees [45]. Piecewise linear systems [46] can be easily represented in a regression tree structure [47]. A special type of regression tree is called the locally linear model tree, which combines a heuristic strategy for input space decomposition with a local linear least squares optimization (like LOLIMOT [1]). These models are hierarchical models consisting of nodes and branches. Internal nodes represent tests on input variables of the model, and branches correspond to outcomes of the tests. Leaf (terminal) nodes contains regression models in the case of regression trees.

Thanks to the structured representation of the local linear models, hinging hyperplanes lend themselves to a straightforward incorporation into model-based control schemes. In this chapter this beneficial property is demonstrated in the design of an instantaneous linearization-based model predictive control algorithm [32].

This chapter organized as follows: the next section discusses how hinging hyperplane function approximation is done with the FCRM identification approach. The

description of the tree growing algorithm and the measures proposed to support model building are given in Sect. 2.2. In Sect. 2.3, application examples are presented, while Sect. 2.4 concludes the chapter.

2.1 Identification of Hinging Hyperplanes

2.1.1 Hinging Hyperplanes

The following section gives a brief description of the hinging hyperplane approach on the basis of [18, 34, 48], followed by a description of how the constraints can be incorporated into FCRM clustering.

For a sufficiently smooth function $f(\mathbf{x}_k)$, which can be linear or nonlinear, assume that regression data $\{\mathbf{x}_k, y_k\}$ is available for $k = 1, \ldots, N$. Function $f(\mathbf{x}_k)$ can be represented as the sum of a series of hinging hyperplane functions $h_i(\mathbf{x}_k)\, i = 1, 2, \ldots, K$. Breiman [20] proved that we can use hinging hyperplanes to approximate continuous functions on compact sets, guaranteeing a bounded approximation error

$$\|e_n\| = \|f - \sum_{i=1}^{K} h_i(\mathbf{x})\| \leq (2R)^4\, c^2/K, \tag{2.1}$$

where K is the number of hinging hyperplane functions, R is the radius of the sphere in which the compact set is contained, and c is such that

$$\int \|w\|^2 |f\,(w)\,| dw = c < \infty. \tag{2.2}$$

The approximation with hinging hyperplane functions can get arbitrarily close if a sufficiently large number of hinging hyperplane functions are used. The sum of the hinging hyperplane functions, $\sum_{i=1}^{K} h_i(\mathbf{x}_k)$, constitutes a continuous piecewise linear function. The number of input variables n in each hinging hyperplane function and the number of hinging hyperplane functions K are two variables to be determined. The explicit form for representing a function $f(\mathbf{x}_k)$ with hinging hyperplane functions becomes (see Fig. 2.1)

$$f(\mathbf{x}_k) = \sum_{i=1}^{K} h_i(\mathbf{x}_k) = \sum_{i=1}^{K} \langle \max \mid \min \rangle \left(\mathbf{x}_k^T \theta_{1,i}, \mathbf{x}_k^T \theta_{2,i} \right), \tag{2.3}$$

where $\langle \max \mid \min \rangle$ means max or min.

Suppose two hyperplanes are given by:

$$y_k = \mathbf{x}_k^T \theta_1,\, y_k = \mathbf{x}_k^T \theta_2, \tag{2.4}$$

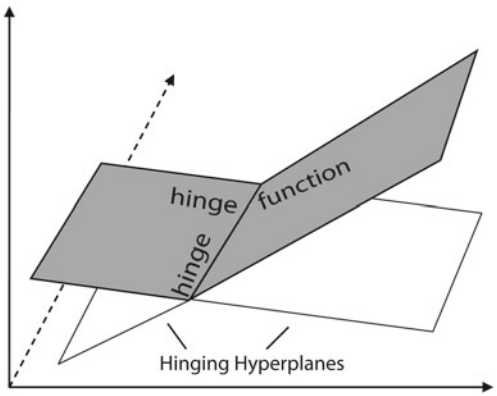

Fig. 2.1 Basic hinging hyperplane definitions

where $\mathbf{x}_k = \left[x_{k,0}, x_{k,1}, x_{k,2}, \ldots, x_{k,n}\right]$, $x_{k,0} \equiv 1$, is the kth regressor vector and y_k is the kth output variable. These two hyperplanes are continuously joined together at $\{\mathbf{x} : \mathbf{x}^T (\theta_1 - \theta_2) = 0\}$, as can be seen in Fig. 2.1. As a result they are called *hinging hyperplanes*. The joint $\triangle = \theta_1 - \theta_2$, is defined as *hinge* for the two hyperplanes $y_k = \mathbf{x}_k^T \theta_1$ and $y_k = \mathbf{x}_k^T \theta_2$. The solid/shaded parts of the two hyperplanes are explicitly given by

$$y_k = \max(\mathbf{x}_k^T \theta_1, \mathbf{x}_k^T \theta_2) \text{ or } y_k = \min(\mathbf{x}_k^T \theta_1, \mathbf{x}_k^T \theta_2). \tag{2.5}$$

The hinging hyperplane method has some interesting advantages for nonlinear function approximation and identification:

1. Hinging hyperplane functions could be located by a simple computationally efficient method. In fact, hinging hyperplane models are piecewise linear models; the linear models are usually obtained by repeated use of the linear least squares method, which is very efficient. The aim is to improve the whole identification method with more sophisticated ideas.
2. For nonlinear functions that resemble hinging hyperplane functions, the hinging hyperplane method has very good and fast convergence properties.

The hinging hyperplane method practically combines some advantages of neural networks (in particular, the ability to handle very large dimensional inputs) and constructive wavelet-based estimators (availability of very fast training algorithms).

The essential hinging hyperplane search problem can be viewed as an extension of the linear least squares regression problem. Linear least squares regression aims to find the best parameter vector $\widehat{\theta}$ by minimizing a quadratic cost function with the regression model that gives the best linear approximation to y. For nonsingular data matrix $\mathbf{X}$, the linear least squares estimate $y = \mathbf{x}^T \theta$ is always uniquely available. The hinging hyperplane search problem, on the other hand, aims to find the two parameter vectors θ_1 and θ_2, defined by

$$[\theta_1, \theta_2] = \arg\min_{\theta_1,\theta_2} \sum_{k=1}^{N} \left[\langle \max \mid \min \rangle \left(y_k - \mathbf{x}_k^T \theta_1,\, y_k - \mathbf{x}_k^T \theta_2 \right) \right]^2 . \tag{2.6}$$

A brute force application of the Gauss-Newton method can solve the above described optimization problem. However, two problems exist [18]:

1. High computational requirement. The Gauss-Newton method is computationally intensive. In addition, since the cost function is not continuously differentiable, the gradients required by the Gauss-Newton method cannot be given analytically. Numerical evaluation is thus needed, which has high computational requirement.
2. Local minima. There is no guarantee that the global minimum can be obtained. Therefore, an appropriate initial condition is crucial.

2.1.2 Improvements in Hinging Hyperplane Identification

The proposed identification algorithm applies a much simpler optimization method, the so-called alternating optimization, which is a heuristic optimization technique and has been applied for several decades for many purposes; therefore, it is an exhaustively tested method in nonlinear parameter and structure identification as well. Within the hinging hyperplane function approximation approach, the two linear submodels can be identified by the weighted linear least squares approach, but their operating regimes (where they are valid) are still an open question.

For that purpose, the fuzzy c-regression model (further referred as FCRM and proposed by Hathaway and Bezdek [37]) was used. This technique is able to partition the data and determine the parameters of the linear submodels simultaneously. With the application of alternating optimization techniques and by taking advantage of the linearity in $(y_k - \mathbf{x}_k^T \theta_1)$ and $(y_k - \mathbf{x}_k^T \theta_2)$, an effective approach is given for hinging hyperplane function identification; hence the FCRM method for a special case ($c = 2$) is able to identify hinging hyperplanes. The proposed procedure is attractive from the local minima point of view as well, because in this way, although the problem is not avoided, it is transformed into a deeply discussed problem, namely the cluster validity problem.

The following quadratic cost function can be applied for the FCRM method:

$$E_m(\mathbf{U}, \{\theta_i\}) = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{i,k})^m E_{i,k}(\theta_i) \,, \tag{2.7}$$

where $m \in \langle 1, \infty)$ denotes a weighting exponent which determines the fuzziness of the resulting clusters, while θ_i represents the parameters of local models and $\mu_{i,k} \in \mathbf{U}$ is the membership degree, which could be interpreted as a weight representing the extent to which the value predicted by the model $f_i(\mathbf{x}_k, \theta_i)$ matches y_k. The prediction error is defined by

$$E_{i,k} = \left(y_k - f_i(\mathbf{x}_k; \theta_i)\right)^2, \tag{2.8}$$

but other measures can be applied as well, provided they fulfill the minimizer property stated by Hathaway and Bezdek [37].

One possible approach to the minimization of the objective function (2.7) is the group coordinate minimization method that results in the following algorithm:

- **Initialization** Given a set of data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, specify c, the structure of the regression models (2.8) and the error measure (2.7). Choose a weighting exponent $m > 1$ and a termination tolerance $\varepsilon > 0$. Initialize the partition matrix randomly.
- **Repeat** For $l = 1, 2, \ldots$
- Step 1 Calculate values for the model parameters θ_i that minimize the cost function $E_m(\mathbf{U}, \{\theta_i\})$.
- Step 2 Update the partition matrix

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} (E_{i,k}/E_{j,k})^{2/(m-1)}}, \ 1 \le i \le c, \ 1 \le k \le N \tag{2.9}$$

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

A specific situation arises when the regression functions f_i are linear in the parameters θ_i, $f_i(\mathbf{x}_k; \theta_i) = \mathbf{x}_{i,k}^T \theta_i$, where $\mathbf{x}_{i,k}$ is a known arbitrary function of $\mathbf{x}_k$. In this case, the parameters can be obtained as a solution of a set of the weighted least squares problem where the membership degrees of the fuzzy partition matrix $\mathbf{U}$ serve as the weights.

The N data pairs and the membership degrees are arranged in the following matrices:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{i,1}^T \\ \mathbf{x}_{i,2}^T \\ \vdots \\ \mathbf{x}_{i,N}^T \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \ \Phi_i = \begin{bmatrix} \mu_{i,1} & 0 & \cdots & 0 \\ 0 & \mu_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_{i,N} \end{bmatrix}. \tag{2.10}$$

The optimal parameters θ_i are then computed by

$$\theta_i = [\mathbf{X}^T \Phi_i \mathbf{X}]^{-1} \mathbf{X}^T \Phi_i \mathbf{y}. \tag{2.11}$$

Applying $c = 2$ during FCRM identification, these models can be used as base identifiers for hinging hyperplane functions. For hinging hyperplane function identification purposes, two prototypes have to be used by FCRM ($c = 2$), and these prototypes must be linear regression models. However, these linear submodels have to intersect each other within the operating regime covered by the known data points (within the hypercube expanded by the data). This is a crucial problem in the hinging hyperplane identification area [18]. To take into account this point of view as well,

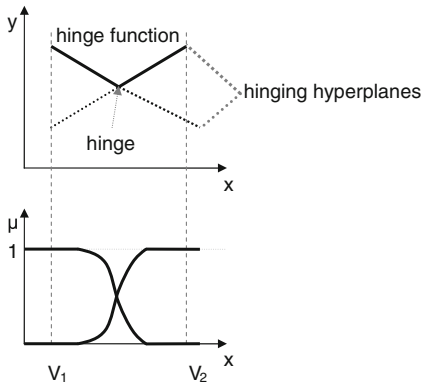

Fig. 2.2 Hinging hyperplane identification restrictions

constraints have to be taken into consideration as follows. Cluster centers $\mathbf{v}_i$ can also be computed from the result of FCRM as the weighted average of the known input data points:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} \mathbf{x}_k \mu_{i,k}}{\sum_{k=1}^{N} \mu_{i,k}}, \tag{2.12}$$

where the membership degree $\mu_{i,k}$ is interpreted as a weight representing the extent to which the value predicted by the model matches y_k. These cluster centers are located in the 'middle' of the operating regime of the two linear submodels. Because the two hyperplanes must cross each other, the following criteria can be specified (see Fig. 2.2):

$$\begin{aligned} &\mathbf{v}_1(\theta_1 - \theta_2) < 0 \text{ and } \mathbf{v}_2(\theta_1 - \theta_2) > 0 \text{ or} \\ &\mathbf{v}_1(\theta_1 - \theta_2) > 0 \text{ and } \mathbf{v}_2(\theta_1 - \theta_2) < 0. \end{aligned} \tag{2.13}$$

These relative constraints can be used to take into account the constraints above:

$$\lambda_{rel,1,2} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \leq 0 \text{ where } \lambda_{rel,1,2} = \begin{bmatrix} \mathbf{v}_1 & -\mathbf{v}_1 \\ -\mathbf{v}_2 & \mathbf{v}_2 \end{bmatrix}. \tag{2.14}$$

When linear equality and inequality constraints are defined on these prototypes, quadratic programming (QP) has to be used instead of the least squares method. This optimization problem still can be solved effectively compared to other constrained nonlinear optimization algorithms.

Local linear constraints applied to fuzzy models can be grouped into the following categories according to their validity region:

- **Local constraints** are valid only for the parameters of a regression model, $\lambda_i \theta_i \leq \omega_i$.
- **Global constraints** are related to all of the regression models, $\lambda_{gl} \theta_i \leq \omega_{gl}$, $i = 1, \ldots, c$.

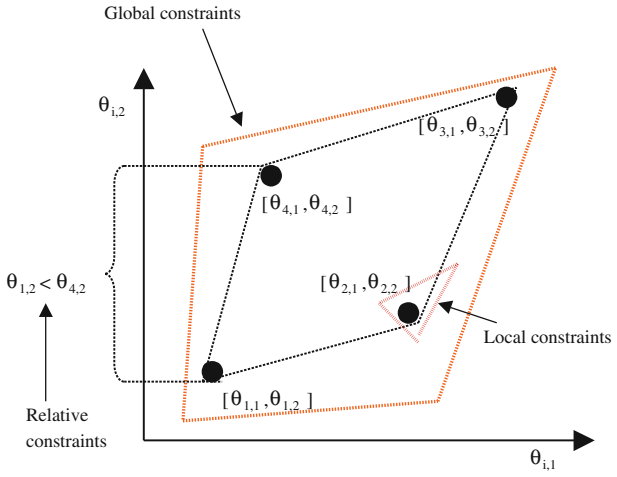

Fig. 2.3 Hinging hyperplane model with four local constraints and two parameters

- **Relative constraints** define the relative magnitude of the parameters of two or more regression models:

$$\lambda_{rel,i,j} \begin{bmatrix} \theta_i \\ \theta_j \end{bmatrix} \leq \omega_{rel,i,j}. \tag{2.15}$$

An example of these types of constraints is illustrated in Fig. 2.3.

In order to handle relative constraints, the set of weighted optimization problems has to be solved simultaneously. Hence, the constrained optimization problem is formulated as follows:

$$\min_{\theta} \left\{ \frac{1}{2} \theta^T \mathbf{H} \theta + \mathbf{c}^T \theta \right\}, \tag{2.16}$$

with $\mathbf{H} = 2\mathbf{X}'^T \Phi \mathbf{X}'$, $\mathbf{c} = -2\mathbf{X}'^T \Phi \mathbf{y}'$, where

$$\mathbf{y}' = \begin{bmatrix} \mathbf{y} \\ \mathbf{y} \\ \vdots \\ \mathbf{y} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_c \end{bmatrix}, \tag{2.17}$$

$$\mathbf{X}' = \begin{bmatrix} \mathbf{X}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{X}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X}_c \end{bmatrix}, \quad \Phi = \begin{bmatrix} \Phi_1 & 0 & \cdots & 0 \\ 0 & \Phi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Phi_c \end{bmatrix}, \tag{2.18}$$

where Φ_i contains local membership values and the constraints on θ:

$$\Lambda\theta \leq \omega, \tag{2.19}$$

with

$$\lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_c \\ \lambda_{gl} & 0 & \cdots & 0 \\ 0 & \lambda_{gl} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{gl} \\ \{\lambda_{rel}\} & & & \end{bmatrix}, \ \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_c \\ \omega_{gl} \\ \omega_{gl} \\ \vdots \\ \omega_{gl} \\ \{\omega_{rel}\} \end{bmatrix}. \tag{2.20}$$

Referring back to Fig. 2.1, it can be concluded that with this method both parts of the intersected hyperplanes are described and the part ($\langle max|min\rangle$) that describes the training data in the most accurate way is selected.

2.2 Hinging Hyperplane-Based Binary Trees

So far, the hinging hyperplane function identification method has been presented. The proposed technique can be used to determine the parameters of one hinging hyperplane function. The classical hinging hyperplane approach can be interpreted by identifying K hinging hyperplane models consisting of global model pairs, since these operating regimes cover the whole N dataset. This representation leads to several problems during model identification and also renders model interpretability more difficult. To overcome this problem, a tree structure is proposed where the data is recursively partitioned into subsets, while each subset is used to form models of lower levels of the tree. The concept is illustrated in Fig. 2.4, where the membership functions and the identified hinging hyperplane models are also shown.

During the identification the following phenomena can be taken into consideration (and can be considered as benefits too):

- When using the hinging hyperplane function there is no need to find splitting variables at the nonterminal nodes, since this procedure is based on the hinge.
- A populated tree is always a binary tree, either balanced or non-balanced, depending on the algorithm (greedy or non-greedy). It is based on a binary tree; the hinge splitting the $\mathbf{x}$ data pertains to the left side of the hinge and θ_1 always goes to the left child; and the right side behaves the similarly. For example, given a simple symmetrical binary tree structure model, the first level contains one hinging

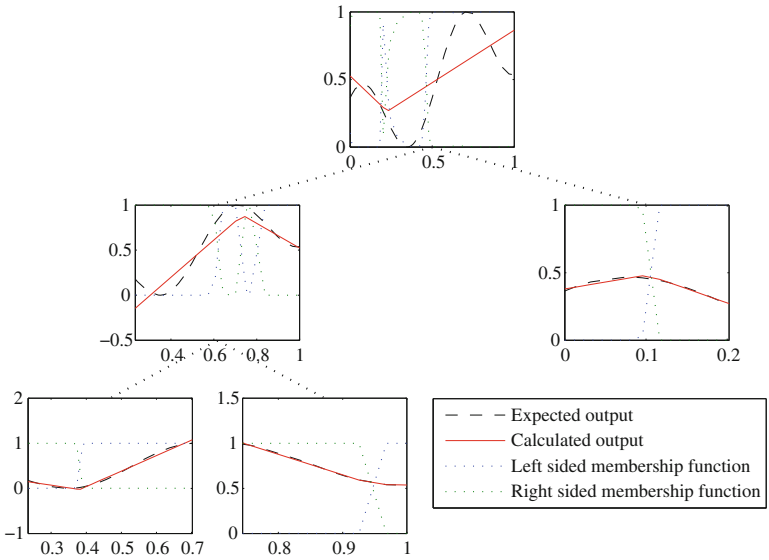

Fig. 2.4 Hinging hyperplane-based regression tree for basic data sample in case of greedy algorithm

hyperplane function, the second level contains two hinging hyperplane functions, the third level contains four hinges, and in general the kth level contains $2^{(k-1)}$ hinging hyperplane functions.

Concluding the above and obtaining the parameters θ during recursive identification, the following cost function has to be minimized:

$$E(\{\theta_i\}, \pi) = \sum_{i=1}^{K} \pi_i E_{m_i}(\theta_i), \tag{2.21}$$

where K is the number of the hinge functions (nodes), and π is the binary ($\pi_i \in 0, 1$) terminal set, indicating that the given node is a final linear model ($\pi_i = 1$), and can be incorporated as a terminal node of the identified piecewise model.

A growing algorithm can be either balanced or greedy. In the balanced case the identification algorithm builds the tree till the desired stopping criteria, while the greedy one continues the tree building by choosing a node for splitting that performs worst during the building procedure. Hence, this operating regime needs more local models for better model performance. For a greedy algorithm, the crucial item is the selection of a good stopping criterion. Any of the following can be used to determine whether to continue the tree growing process or stop the procedure:

1. The loss function becomes zero. This corresponds to the situation where the size of the data set is less than or equal to the dimension of the hinge. Since the hinging hyperplanes are located by linear least squares, when the number of data points is

equal to the number of parameters to be determined, the result is exact, provided the matrix is not singular.

2. $E = E_1 + E_2$, where $E_i = \sum_{k=1}^{N} \mu_{i,k} \left(y_k - f_i(\mathbf{x}_k; \theta_i)\right)^2$ represents the performances of the left-and right-hand side models of the hinge. During the growth of the binary tree, the loss function is always nonincreasing, so E should be always smaller than the performance of the parent node. When no decrease is observed in the loss function, the growing should be stopped.
3. The tree-building process reaches the predefined tree depth.
4. All of the identified terminal nodes' performances meet an accuracy level (ε, the error rate). In this case, it is not necessary to specify the depth of the tree, but it can cause overfitting of the model.

The algorithm results are represented in Fig. 2.4, where $L = 3$, $K = 5$, and $\pi = [0, 0, 1, 1, 1]$. In Fig. 2.5 a three-dimensional example is shown. The function

$$y = \frac{\sin\left(\sqrt{x_1^2 + x_2^2 + \varepsilon}\right)}{\sqrt{x_1^2 + x_2^2 + \varepsilon}} \tag{2.22}$$

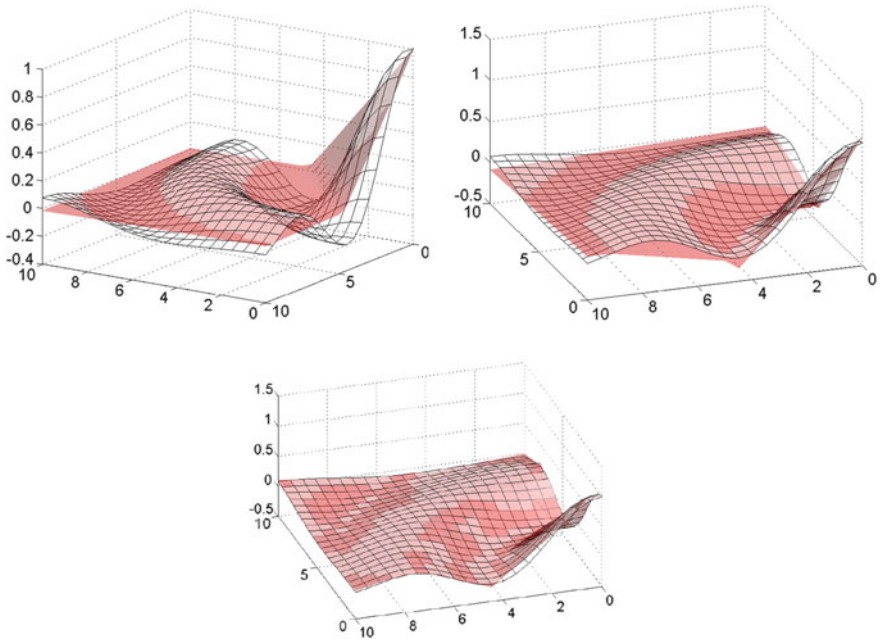

Fig. 2.5 Modeling a 3D function with hinging hyperplane-based tree

has been approximated by a hinging hyperplane-based tree. In Fig. 2.5 it is shown how the approximation becomes much more smooth with applying one, two, and four levels and greedy building method.

The generation of the binary tree structured model is not the only thing important; to construct a greedy algorithm and to measure the identified model, node performance must be determined during the identification procedure. It can be defined in different ways:

- *Modeling performance of the nodes*

  The well-known regression performance estimators can be used for node performance measurement; in this work, root mean squared prediction error (RMSE) was used:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}. \tag{2.23}$$

- *Condition of the nodes*

  It is described that with two prototype clusters ($c = 2$) and apriori knowledge (*constraints*), the FCRM method is able to identify hinging hyperplanes; hence, membership degree $\mu_{i,k}$ has information at a node about how many data points are going to the slitted prototypes. Comparing this data with the information about the hinge-based node splitting rule (how many data samples are described by the θ_1, θ_2 parameter vectors), we can assign a certain condition (ρ) to a node:

$$\rho_n = 1 - \frac{\|m_1 - \sum_{k=1}^{N_n} \mu_{1,k}\|}{\sum_{i=1}^{2}\sum_{k=1}^{N_n} \mu_{i,k}}, \tag{2.24}$$

  where m_1 is the cardinality based on θ^+, while N_n represents the number of samples at node π_i.

We can consider ρ as a measurement of the FCRM hinging hyperplane identification perfection. The closer ρ is to 1, the better the identification. The hinge does not "override" the $\mu_{i,k}$ membership degrees. This measure is very similar to the one that was introduced in [49] and was used for identifying parameter similarity. In Figs. 2.6 and 2.7 RMSE and ρ results of identifying Eq. 2.22 node by node (axis x) with the depth of four levels can be seen. In Fig. 2.6, the non-greedy case is examined, while Fig. 2.7 shows the performance of the greedy algorithm. It is apparent that for tree-building purposes, cardinality-based splitting is a very good approach.

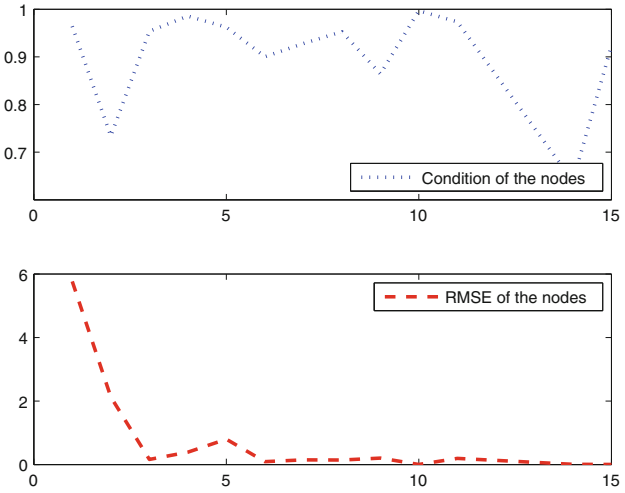

Fig. 2.6 Node by node ρ and RMSE results for non-greedy tree building

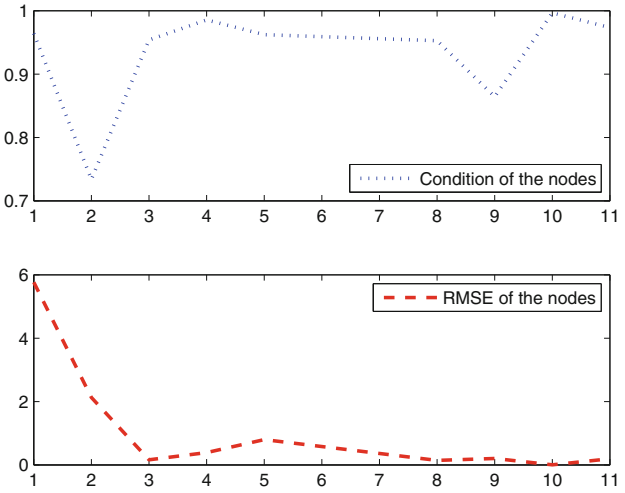

Fig. 2.7 Node by node ρ and RMSE results for greedy tree building

2.3 Application Examples

Accuracy and transparency of the proposed algorithm are shown based on multiple datasets, two real life and two synthetic ones, followed by examples from the area of dynamic system identification.

2.3.1 Benchmark Data

All datasets have been used before, most of them have originated from well-known data repositories. Performance of the models is measured by the root mean squared prediction error (RMSE; see Eq. 2.23).

Real life datasets:

- *Abalone* Dataset from UCI machine learning repository[1] used to predict the age of abalone from physical measurements. Contains 4,177 cases with eight attributes (one nominal and seven continuous).
- *Kin8nm* Data containing information on the forward kinematics of an eight link robot arm from the DVELVE repository. Contains 8,192 cases with eight continuous attributes.

Synthetic datasets:

- *Fried* Artificial dataset used by Friedman [50] containing ten continuous attributes with independent values uniformly distributed in the interval [0, 1]. The value of the output variable is obtained with the equation:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0, 1). \tag{2.25}$$

- *3DSin* Artificial dataset containing two continuous predictor attributes uniformly distributed in interval $[-3, 3]$, with the output defined as

$$y = 3\sin(x_1)\sin(x_2). \tag{2.26}$$

3,000 data points were generated using these equations.

For the robust testing of the performance of the model-building algorithm, ten-fold cross-validation method is utilized with data normalized to zero mean and unit variance. Table 2.1 shows the performance on these datasets compared to the results of other algorithms and can be found in [38]. Moreover, Table 2.2 contains the min, mean, and max error rates of the ten-fold cross-validation results for the hinging hyperplane algorithm with the calculated standard deviation values. The comparative algorithms are fuzzy-based (*FRT-Fuzzy Regression Tree, FMID-Fuzzy Model*

[1] FTP address: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/.

Table 2.1 Comparison of RMSE results of different algorithms

Data	Sample	HH	CART	FMID	FRT
Fried	Train	0.87	0.84	2.41(4)	0.69
	Test	0.92(8)	2.12(495.6)	2.41(12)	0.7(15)
3Dsin	Train	0.17	0.09	0.50(4)	0.18
	Test	0.18(11)	0.17(323.1)	0.31(12)	0.18(12)
Abalone	Train	2.62	1.19	2.20(4)	2.18
	Test	2.88(8)	2.87(664.8)	2.19(12)	2.19(4)
Kinman	Train	0.15	0.09	0.20(4)	0.15
	Test	0.16(6)	0.23(453.9)	0.20(12)	0.15(20)

Numbers in brackets are the number of models

Table 2.2 10-fold cross validation report for hinging hyperplanes based tree

Data	Sample	MIN	MEAN	MAX	Standard dev.
Fried	Train	0.5822	0.8677	1.2107	0.227
	Test	0.6226	0.9208	1.2673	0.2337
3Dsin	Train	0.0906	0.1741	0.3162	0.0714
	Test	0.0838	0.178	0.342	0.0801
Abalone	Train	2.3496	2.6241	2.9256	0.1532
	Test	2.3242	2.8803	3.451	0.3445
Kinman	Train	0.1433	0.1515	0.1595	0.0054
	Test	0.1464	0.1579	0.1729	0.0092

Identification) and classical regression tree-based (*CART-Classification and Regression Tree*). It can be concluded that the performance of the introduced algorithm is in line with even the other methods, that of with a moderate number of terminal nodes in the identified model tree. Results are consistent; even for the worst performance of the ten-fold cross-validation.

2.3.2 Application to Dynamical Systems

The following subsection shows results on the identification first-order nonlinear dynamic system and describes performance of the proposed technique in model predictive control.

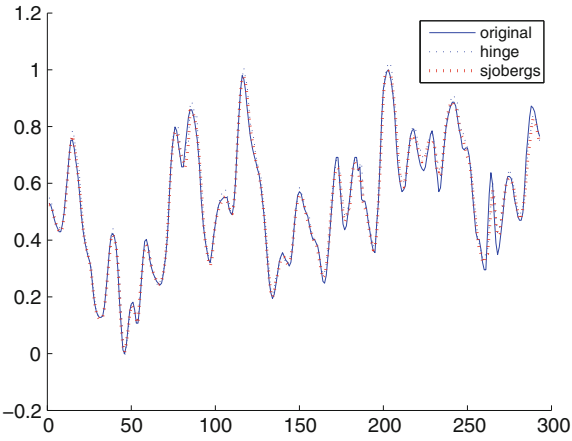

Fig. 2.8 Identification of the Box-Jenkins gas furnace model with hinging hyperplanes

2.3.2.1 Identification of the Box-Jenkins Gas Furnace

The well-known Box-Jenkins furnace data benchmark is used to illustrate the proposed modeling approach and to compare its effectiveness with other methods. The data set consists of 296 pairs of input-output observations taken from a laboratory furnace with a sampling time of nine seconds. The process input is the methane flow rate and the output is the percentage of CO_2 in the off-gas. A number of researchers concluded that a proper structure of a dynamic model for this system is

$$y(k+1) = f(y(k), u(k-3)). \tag{2.27}$$

The approximation power of the model can be seen in Fig. 2.8 and Table 2.3. Comparing its results with those of other techniques in [51] can conclude that the modeling performance is in line with that of other techniques with a moderate number of identified hinging hyperplanes.

So far, *a general nonlinear modeling technique* was presented and a new identification approach was given for hinging hyperplane-based nonlinear models: $\widehat{y} = f(\mathbf{x}(k), \theta)$, where $f(.)$ represents the hinging hyperplane-based tree structured model and $\mathbf{x}(k)$ represents the input vector of the model. To identify a discrete-time *input-output model* for a dynamical system, the dynamic model structure has to be

Table 2.3 RMSE results of the generated models

Method	Training	Testing	Free run	HH ♯
Proposed technique	0.0266	0.0311	0.0374	4
Sjoberg model	0.0336	0.0342	0.0351	4

chosen or determined beforehand. A possible structure often applied is the nonlinear autoregressive model with exogenous input (NARX), where the input vector of the model $\mathbf{x}(k)$ contains the delayed inputs and outputs of the system to be modeled [32]. In several practical cases a simpler and more specific model structure can be used to approximate the behavior of the system, which fits better the structure of the system. Therefore, it can be an advantage for the identification approach (models with simpler structures can be identified), and this model can be more accurate. One such special case of the NARX model is the Hammerstein model, where the same static nonlinearity f is defined for all of the delayed control inputs (for the sake of simplicity, SISO models are considered):

$$\widehat{y} = \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{j=1}^{n_b} b_j, f(u(k-j)) \tag{2.28}$$

where $y()$ and $u()$ are the output and input of the system, respectively, and n_a and n_b are the output and input orders of the model. The parameters of the blocks of the Hammerstein model (static nonlinearity and linear dynamics) can be identified by the proposed method simultaneously if the same linear dynamic behavior can be guaranteed by all of the local hinging hyperplane-based models. It can be done in an elegant way utilizing the flexibility of the proposed identification approach: global constraints can be formulated for the a_i and b_j parameters of the local models (for a detailed discussion on what constraints have to be formulated, see [32]). In the following, the hinging hyperplane modeling technique is applied on a Hammerstein type system. It will be shown why it is an effective tool for the above-mentioned purpose.

2.3.2.2 Application to Model Predictive Control

The modeling of a simulated water heater (Fig. 2.9) is used to illustrate the advantages of the proposed hinging hyperplane-based models. The water flows through a pair of metal pipes containing a cartridge heater.

The outlet temperature, T_{out}, of the water can be varied by adjusting the heating signal, u, of the cartridge heater (see [32] or Appendix D for details). The performance of the cartridge heater is given by:

$$Q(u) = Q_M \left[u - \frac{\sin(2\pi u)}{2\pi} \right] \tag{2.29}$$

where Q_M is the maximal power and u is the heating signal (voltage). As the equation above shows, the heating performance is a static nonlinear function of the heating signal. Hence, the Hammerstein model is a good match to this process. The aim is to construct a dynamic prediction model from data for the output temperature (the dependent variable, $y = T_{out}$) as a function of the control input: the heating

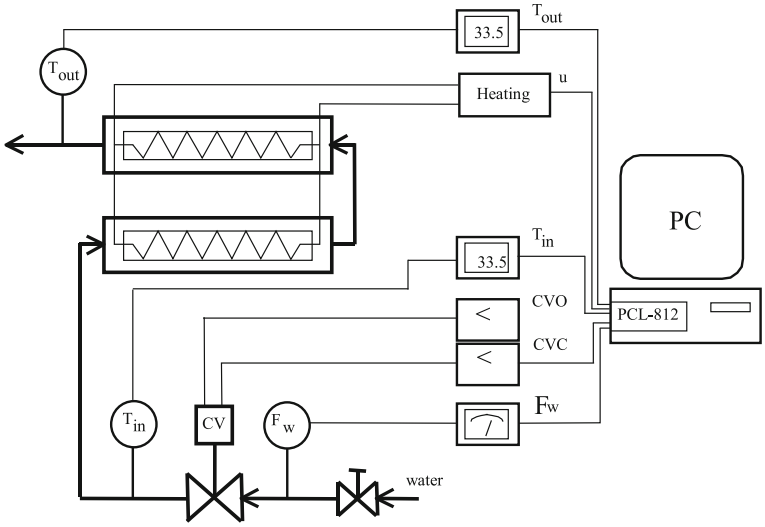

Fig. 2.9 The water heater

signal. The parameters of the Hammerstein model were chosen as $n_a = n_b = 2$. The performance of this modeling technique will be compared to that of linear and feedforward neural network models. The modeling performances can be seen in Table 2.4. Modeling errors were also calculated based on RMSE (see Eq. (2.23)). In this example, a hinging hyperplane function-based tree with four leaves was generated. For robust testing of the model-building algorithm performance, ten-fold cross-validation method was used. For comparison, a feedforward neural net and linear model was also trained and tested using the same data. The neural net contains one hidden layer with 4 neurons using tanh basis functions. As can be seen from the results, the training and test errors are comparable with the errors of the proposed method. A very rigorous test of NARX models is free run simulation because the errors can be cumulated. It can be also seen in Fig. 2.10 that the identified models (the proposed ones, linear models and the neural nets) perform very good also in free run simulation (the system output and the simulated ones can hardly be distinguished). Although the neural net seems to be more robust in this example, the proposed hinging hyperplane model is much more interpretable than the neural net [1]. This confirms that both

Table 2.4 RMSE results of the generated models

Method	Training	Testing	Free run
Linear model	0.0393	0.0449	0.387
Neural network	0.0338	0.0403	0.356
Proposed method	0.0367	0.0417	0.359

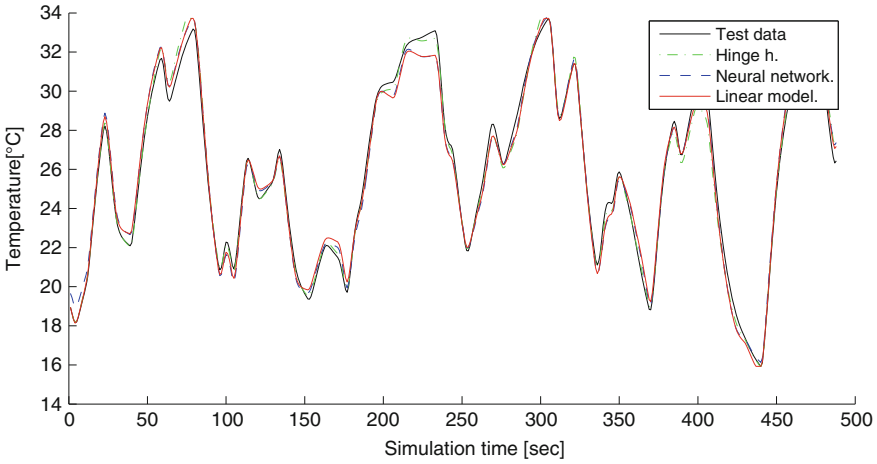

Fig. 2.10 Free run simulation of the water heater (proposed hinging hyperplane model, neural network, linear model)

the proposed clustering-based constrained optimization strategy and the hierarchical model structure have advantages over the classical gradient-based optimization of global hinging hyperplane models.

In the following, this model will be applied for model predictive control. Details of model-based control of fuzzy and operating regime models can be found in [41, 42]. Among the wide range of possible solutions, a model predictive controller (MPC) was designed. Figure 2.11 shows the structure of an MPC controller.

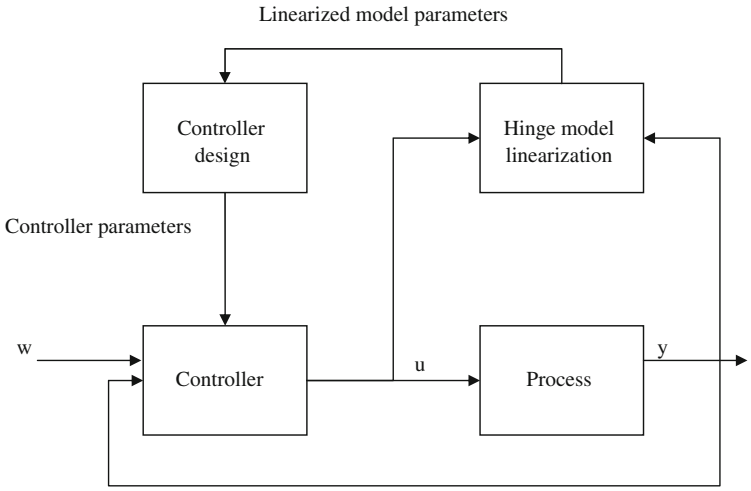

Fig. 2.11 Structure of the MPC controller

Real-time control needs low computational complexity. Hence a time-varying linear MPC has been designed based on time-varying parameters of a linear model extracted at every time instant from the regression tree. This scheme has been widely studied and similarities to the nonlinear optimization based control solutions including convergence have also been shown. In [32] it was shown that there are two ways to obtain a linear model from a nonlinear operating regime-based (fuzzy) model. Taylor expansion-based linearization assumes global interpretation of the model, while the linear parameter varying (LPV) model extraction approach considers the model as an interpolating system between local linear time-invariant (LTI) models were the dynamic effect of the interpolation is negligible. Fortunately, thanks to hinging hyperplane models and tree-structured representation, the proposed model perfectly supports the interpretation, that hinging hyperplanes define local linear models and their operating regimes. Since these local models do not overlap, the negative effect of the interpolating functions do not have to be taken into account.

The classical model predictive controller computes an optimal control sequence by minimizing the following cost quadratic cost function:

$$J\left(H_p, H_c, \lambda\right) = \sum_{j=1}^{H_p} \left(w\left(k+j\right) - \hat{y}\left(k+j\right)\right)^2 + \lambda \sum_{j=1}^{H_c} \Delta u^2\left(k+j-1\right), \quad (2.30)$$

where $\hat{y}\left(k+j\right)$ denotes the predicted process output, H_p is the maximum costing or prediction horizon, H_c is the control horizon, and λ is a weighting coefficient. According to the receding horizon principle, only the first element of the optimized control sequence is applied $u(k)$, and this optimization is performed at every time instant. This scheme allows real-time control, provides feedback on model errors, handles unmeasured disturbances, and supports the previously mentioned iterative liberalization scheme. Details about the convergence and possible extensions of this control scheme can be found in [33].

The key equation of MPC is the prediction of the model:

$$\hat{\mathbf{y}} = \mathbf{S}\Delta\bar{\mathbf{u}} + \mathbf{p}, \quad (2.31)$$

where the model prediction equation is given in its vector-based form as $\Delta\bar{\mathbf{u}} = [\Delta u(k), \ldots, \Delta u(k+H_c)]$, and $\mathbf{p} = \left[p_1, p_2, \ldots, p_{Hp}\right]$ and $\hat{\mathbf{y}} = [\hat{y}(k+1), \ldots, \hat{y}(k+H_p)]$ and the $\mathbf{S}$ containing the parameters of a step-response model is an $\left(H_p\right) \times H_c$ matrix with 0 entries $s_{i,j}$ for $j - i > 1$:

$$\mathbf{S} = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ s_2 & s_1 & 0 & 0 \\ \vdots & & \ddots & \\ s_{H_p} & s_{H_p-1} & \cdots & s_{H_p-H_c} \end{bmatrix}. \quad (2.32)$$

When constraints are considered, the minimum of the cost function can be found by quadratic optimization with linear constraints:

$$\min_{\bar{\mathbf{u}}} \left\{ (\mathbf{S}\Delta\bar{\mathbf{u}} + \mathbf{p} - w)^T (\mathbf{S}\Delta\bar{\mathbf{u}} + \mathbf{p} - w) + \lambda \Delta\bar{\mathbf{u}}^T \Delta\bar{\mathbf{u}} \right\}, \tag{2.33}$$

$$\min_{\bar{\mathbf{u}}} \left\{ \frac{1}{2} \Delta\bar{\mathbf{u}}^T \mathbf{H} \Delta\bar{\mathbf{u}} + \mathbf{d}\Delta\bar{\mathbf{u}} \right\},$$

with $\mathbf{H} = 2\left(\mathbf{S}^T\mathbf{S} + \lambda\mathbf{I}\right)$, $\mathbf{d} = -2\left(\mathbf{S}^T(\mathbf{w} - \mathbf{p})\right)$, where $\mathbf{I}$ is an $(H_c \times H_c)$ unity matrix.

The constraints defined on u and Δu can be formulated with the following inequality:

$$\begin{pmatrix} \mathbf{I}_{\Delta\bar{\mathbf{u}}} \\ -\mathbf{I}_{\Delta\bar{\mathbf{u}}} \\ \mathbf{I}_{H_c} \\ -\mathbf{I}_{H_c} \end{pmatrix} \Delta\bar{\mathbf{u}} \leq \begin{pmatrix} \mathbf{u}_{\max} - \mathbf{I}_{\bar{\mathbf{u}}} u(k-1) \\ -u_{\min} + \mathbf{I}_{\bar{\mathbf{u}}} u(k-1) \\ \Delta\mathbf{u}_{\max} \\ -\Delta u_{\min} \end{pmatrix}, \tag{2.34}$$

where $\mathbf{I}_{H_c}$ and $\mathbf{I}_{\bar{\mathbf{u}}}$ is an $H_c \times H_c$ unity matrix, $\mathbf{I}_{\Delta\bar{\mathbf{u}}}$ is an $H_c \times H_c$ lower triangular matrix with all elements equal to 1, and $\Delta\mathbf{u}_{\min}, u_{\max}, \mathbf{u}_{\min}, \mathbf{u}_{\max}$ are H_c-vectors, with the constraints $\Delta u_{\min}$, $\Delta u_{\max}$, $u_{\min}$, $u_{\max}$, respectively.

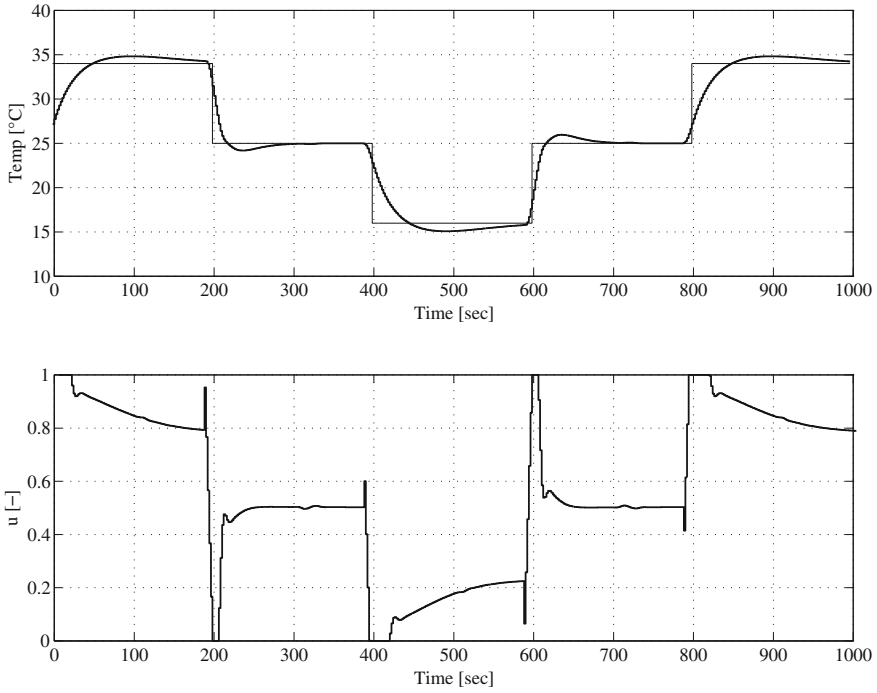

Fig. 2.12 Performance of the MPC based on linear model

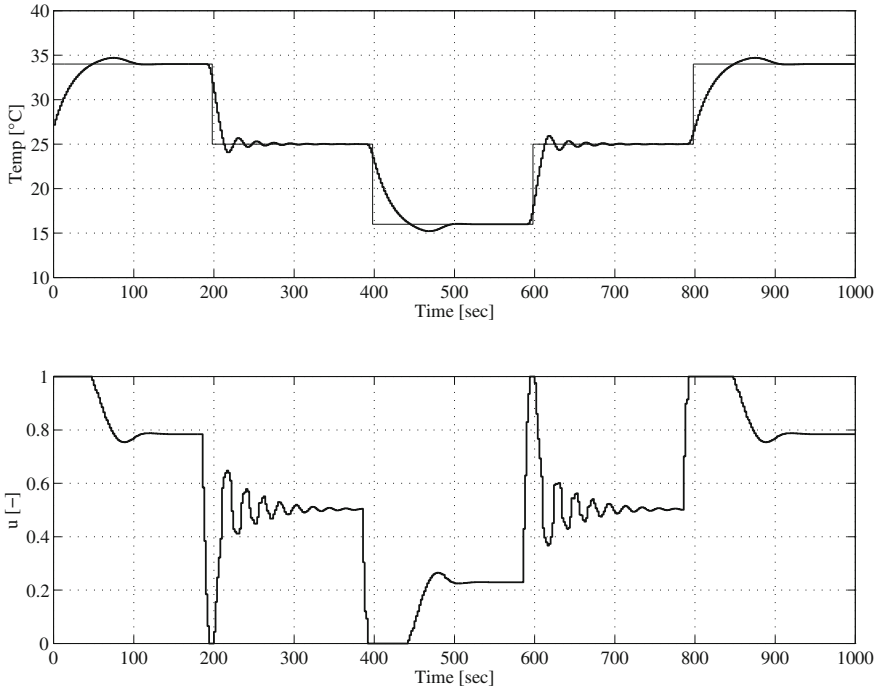

Fig. 2.13 Performance of the MPC based on neural network model

To handle modeling error, the MPC is applied in the well-known internal model control (IMC) scheme where the setpoint of the controller is shifted by the filtered modeling error. For this purpose, a first-order linear filter is used:

$$e_{mf}(k) = \alpha e_m(k) + (1-\alpha)\, e_{mf}(k-1)\,, \tag{2.35}$$

where $0 \leq \alpha < 1$ is determined such that a compromise between performance and robustness is achieved. Effective suppressing of the steady-state modeling error can be achieved by proper tuning of this filter. The best parameters are found for the controller: $H_p = 9$, $H_c = 2$, $\lambda = 20$ and $\alpha = 0.95$. Simulation results for the hinging hyperplane based model, the affine neural network model and the linear model are shown in Figs. 2.12, 2.13 and 2.14.

At the operation region edges, the MPC based on the linear model resulted in undesirable overshoots and undershoots. This is a direct consequence of a bad estimation of the nonlinear gain of the system in these regions. This overestimation of the system gain by the linear model is also seen in the sluggish control action. In contrast, the MPC based on the nonlinear models shows superior performance over the whole operating region. Among these, the MPC based on the hinging hyperplane model results in the smallest overshoot with the fastest change in the control signal.

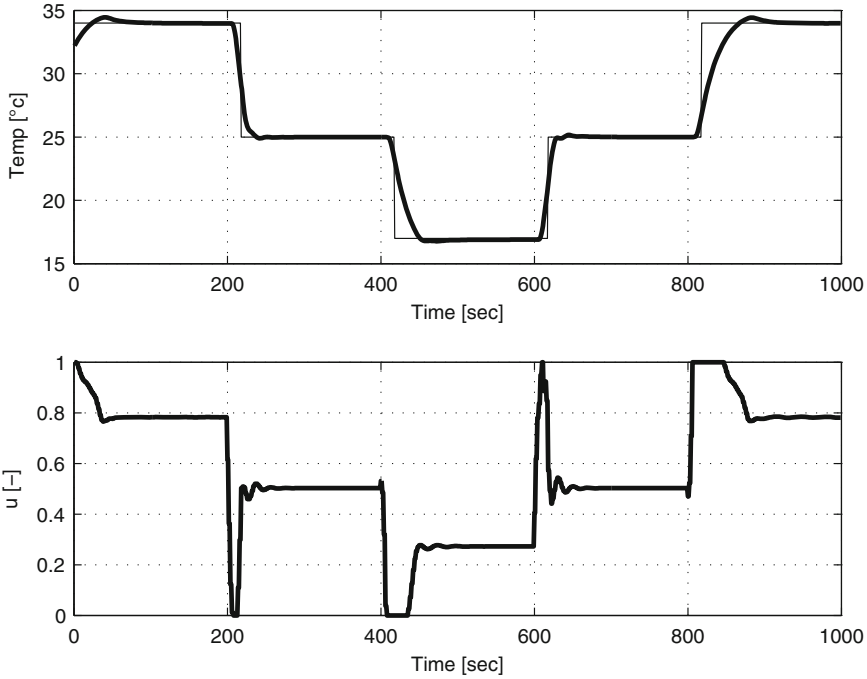

Fig. 2.14 Performance of the MPC based on hinging hyperplane

Table 2.5 Simulation results (SSE, sum squared tracking error; CE, sum square of the control actions)

The applied model in GPC	SSE	CE
Linear model	1085	1.61
Neural network model	956	1.39
Hinging hyperplane model	966	0.58

Notice also that the oscillatory behavior of the neural network model-based MPC is due to the bad prediction of the steady-state gain of the system around the middle region. However, as can be seen from Table 2.5, both nonlinear models achieved approximately the same summed squared tracking error (SSE), although a smaller control effort (CE) was needed for the hinging hyperplane-based MPC.

2.4 Conclusions

A novel framework for hinging hyperplane-based data-driven modeling has been developed. Fuzzy c-regression clustering can be used to identify the parameters of two hyperplanes. A hierarchical regression tree is obtained by the recursive cluster-

ing of the data. The complexity of the model is controlled by the proposed model performance measure. The resulting piecewise linear model can be effectively used to represent nonlinear dynamical systems. The resulting linear parameter varying (LPV) model can be easily utilized in model-based control.

To illustrate the advantages of the proposed approach, benchmark datasets were modeled and a simulation example presented for the identification and model predictive control of a laboratory water heater.

The results show that, with the use of the proposed modeling framework, accurate and transparent nonlinear models can be identified since the complexity and the accuracy of the model can be easily controlled. The local linear models can be easily interpreted and utilized to represent operating regimes of nonlinear dynamical systems. Based on this interpretation, effective model-based control applications can be designed.

http://www.springer.com/978-3-319-21941-7

Interpretability of Computational Intelligence-Based Regression Models

Kenesei, T.; Abonyi, J.

2015, X, 82 p. 34 illus., 14 illus. in color., Softcover
ISBN: 978-3-319-21941-7