(Forthcoming) *FUZZY SETS AND SYSTEMS, Vol. 101/2, January, 1999*

Designing Fuzzy Controllers by Rapid Learning

Jianwei Zhang and Alois Knoll

Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany

Abstract

We propose a learning approach to designing fuzzy controllers based on the B-spline model. Unlike other normalised parameterised set functions for defining fuzzy sets, B-spline basis functions do not necessarily span from membership values zero to one, but possess the property "partition of unity". B-spline basis functions can be automatically determined after each input is partitioned. Learning of a fuzzy controller based on B-spline basis functions is then equivalent to the adaptation of a B-spline interpolator. Parameters of the controller output of each rule can be adapted by using the gradient descent method. Optimal placements of the B-spline basis functions for specifying each input can be found by an algorithm working similarly to a self-organising neural network. Through comparative examples of function approximation we show that learning of such a fuzzy controller generally converges fast. This approach can be extended to the problems of supervised as well as unsupervised learning.

1 Introduction

In most fuzzy systems, linguistic terms are defined by *fuzzy numbers*, i.e. normalised, closed, convex fuzzy sets. In approximate reasoning, usually only qualitative information is referred to, thus the inference result is not very sensitive to the shape and the height of the fuzzy sets. However, if a fuzzy logic system is applied to modeling or control problems, parameters describing fuzzy sets are implicitly or explicitly used both in the inference procedure and in the defuzzification. Therefore, the specification of the fuzzy sets for both the IF- and THEN-part is worth being discussed in more detail.

IF-part. All fuzzy controllers employ real fuzzy sets for modeling linguistic terms for each input. The input space is partitioned into overlapping regions, which both reflects the vague modeling of linguistic concepts and enables the continuous transition of output values. The IF-part of a rule is generally described as:

$$(x_1 \text{ is } A^1_{i_1}) \text{ and } (x_2 \text{ is } A^2_{i_2}) \text{ and } \ldots \ (x_n \text{ is } A^n_{i_n})$$

where x_j is the jth input ($j = 1, \ldots, n$) and $A^j_{i_j}$ is the ith linguistic term defined on x_j. The "and"-operation is implemented with a t-norm, which is represented by "min" or "product" in most applications.

While a discrete representation of fuzzy sets is employed by some fuzzy hardware chips to avoid online function evaluation, parameterised representations are normally adopted in fuzzy controllers running on general-purpose, nonfuzzy computer architectures. In most control applications, triangle and trapezoid set functions are used. Recently, Gaussian fuzzy basis functions have also been proposed for function approximation, [12]. In [8] functions like "Cauchy", "*Sinc*", "Laplace", "Logistic", "Hyperbolic Tangent" are introduced and their abilities of function approximation are compared [1]. However, all these set functions need additional parameters beside the partition positions (called *knots* in the following) on each input's universe of discourse, e.g. the standard deviation of a Gaussian function. Since only the knots are the intrinsic parameters resulting from the partition of the input space, the selection and tuning of these additional parameters are neither natural nor intuitive.

THEN-part. The classical fuzzy controller of Mamdani type [7] is based on the idea of directly using symbolic rules for diverse control tasks (cf. the applications shown in [11,13]). As application areas grow, the *systematic* design of an optimal fuzzy controller becomes more and more important, as pointed out in [4]. A rule of a Mamdani type controller has the form:

$$\begin{array}{ll} \text{IF} & (x_1 \text{ is } A^1_{i_1}) \text{ and } (x_2 \text{ is } A^2_{i_2}) \text{ and } \ldots \text{ and } (x_n \text{ is } A^n_{i_n}) \\ \text{THEN} & y \text{ is } B_k \end{array}$$

where B_k is a fuzzy set with the same properties as that used in the "IF-part", $k = 1, \ldots, t$, and t is the total number of linguistic terms for modeling the output y. The aggregation of output values of all the firing rules are realised either by the "max"-operator [7] or by simple addition [6], where the later is a small variation of the former and even more simple to compute.

Another important type of fuzzy controllers is based on the TSK (Takagi-Sugeno-Kang) model [10]. Recently, TSK type fuzzy controllers have been used for function approximation and supervised learning, [12,5]. A rule using a TSK model of order 1 can be generally described as:

[1] The experimental results in [8] show that the nonconvex function *sinc* works generally better than the others for a quick and accurate function approximation. Nevertheless, this function possesses more than one peak and is difficult to be assigned an appropriate linguistic meaning.

IF $(x_1 \text{ is } A^1_{i_1})$ and $(x_2 \text{ is } A^2_{i_2})$ and ... and $(x_n \text{ is } A^n_{i_n})$

THEN $y = a^i_0 + a^i_1 x_1 + \cdots + a^i_n x_n$

where $a^i_0, a^i_1, \ldots, a^i_n$ are the coefficients of a simplified local linear model. These parameters can be identified by optimising a least squares performance index using training data. Recent work with TSK model shows that it is a suitable function approximator. However, some authors [1] pointed out that the TSK model is a multi-local-model black-box. Obviously, a general TSK model does not fully utilise the advantage of linguistic control because the polynomial combination of input variables cannot be easily extracted from the expert's intuitive knowledge.

We propose an approach that can build the fuzzy sets for linguistic terms of the IF-part systematically and adapt the control actions of the THEN-part through learning. Our model of linguistic terms is based on B-spline basis functions, a special set of piecewise polynomial curves.

2 Constructing Fuzzy Controllers with B-Splines

In principle, the evaluation of a fuzzy rule base is an *interpolation process*. Therefore, if we consider an automatic method for designing a fuzzy controller, it is useful to review the interpolation methods that use analytical functions. Lagrange polynomials supply a set of functions that can be used for interpolating a given number of data points. Newton polynomials can realise the same task but they can be recursively computed so that a new polynomial does not need to be totally re-calculated for new data. Bernstein polynomial functions are based on a parameter set and can also interpolate data quite well. However, with B-splines, the order of basis functions is independent of the number of interpolation data. Moreover, basis functions of the B-spline model can be used as a convenient tool to specify linguistic terms.

2.1 B-Spline Basis Functions

Assume x is a general input variable of a control system that is defined on the universe of discourse $[x_0, x_m]$. Given a sequence of ordered parameters (knots): $(x_0, x_1, x_2, \ldots, x_m)$, the ith normalised B-spline basis function (B-function) $X_{i,k}$ of order k is recursively defined as:

$$X_{i,k}(x) = \begin{cases} \begin{cases} 1 & \text{for } x_i \leq x < x_{i+1} \\ 0 & \text{otherwise} \end{cases} & \text{if } \; k = 1 \\ \frac{x - x_i}{x_{i+k-1} - x_i} X_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} X_{i+1,k-1}(x) & \text{if } \; k > 1 \end{cases} \tag{1}$$

with $i = 0, 1, \ldots, m - k$. Figure 1 shows B-functions from order $k = 1, 2, 3, 4$ which can be compared with fuzzy membership functions (MFs).

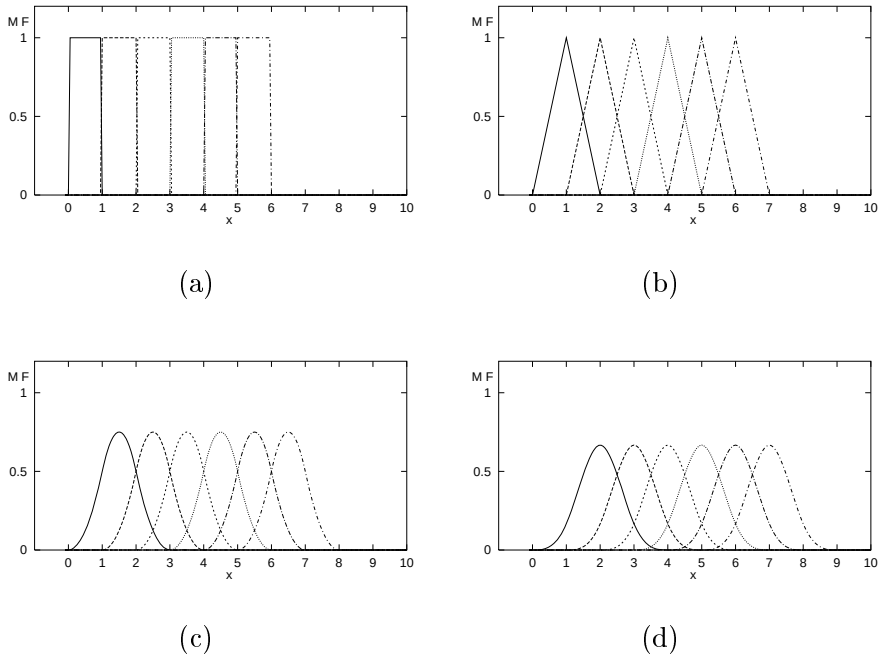


Fig. 1. Fuzzy sets defined by B-spline basis functions of different orders. (a) k=1. (b) k=2. (c) k=3. (d) k=4.

2.2 Core Support Points and Knots

In fuzzy set theory, the *support* of a fuzzy set A within a universal set X is the crisp set that contains all the elements of X that have nonzero membership grades in A. If a B-function of order $k > 1$ is used for modeling a fuzzy set, it has only one peak (with the largest membership grade). Such a support point of the interpolation position is defined as *core support point*, denoted by CSP [2] . If B-functions with uniformly distributed knots

[2] *Core support points* are also called interpolation abscissas in other B-spline applications.

are used, $CSP(A) = \{x|A(x) = maximum\}$, where A is defined by a B-function.

A B-function representing A_i is defined by the *knots*, the boundary points of the *support* of A_i. The complete *knots* consist of two parts, the *interior knots* (noted as *Iknots*) that lie within the universe of discourse and *Extended knots* (*Eknots*) that are generated at both ends of the universe for defining the marginal linguistic terms, [15]. Generally, $m - (k \ mod \ 2)$ interior knots are needed, where m is the number of the real linguistic terms, and $k \leq m$ is the order of the B-functions.

If k is even, the interior knots coincide with the CSPs (Fig. 2). If k is odd, the $m - 1$ interior knots can be determined by

$$Iknot_i \ = \ CSP_i \ + \ \frac{CSP_{i+1} \ - \ CSP_i}{2}; \ i = 1, \ldots, m-1. \tag{2}$$

At each end of the universe of discourse $[CSP_1, \ CSP_m]$, $((k \ + \ 1) \ div \ 2)$ *Eknots* can be determined by mirroring the *Iknots* with respect to CSP_1 and CSP_m. Altogether there are $k \ + \ m$ *knots*.

2.3 A B-Spline Interpolator

Since a MIMO (multiple-input–multiple-output) rule base is normally divided into several MISO (multiple-input–single-output) rule bases, we consider only the MISO case. In [15], we showed that under the following conditions the computation of the output of such a fuzzy controller is equivalent to that of a *general B-spline hypersurface*:

- The membership functions for the inputs are periodical B-spline basis functions;
- The membership functions for the outputs are fuzzy singletons;
- "Product" is used as fuzzy conjunctions;
- "Centroid" is the defuzzification method;
- "Virtual linguistic terms" are added at both ends of each input variable; and
- The rule base for the "virtual linguistic terms" is extended by copying the output values of the "nearest" neighbourhood.

Generally, we consider a MISO system with n inputs $x_1, x_2, \ldots, x_n$, rules with the n conjunctive terms in the premise given in the following form:

IF $(x_1$ is $X^1_{i_1,k_1})$ and $(x_2$ is $X^2_{i_2,k_2})$ and $\ldots$ and $(x_n$ is $X^n_{i_n,k_n})$ THEN y is $Y_{i_1 i_2 \ldots i_n}$

where

- x_j: the jth input $(j = 1, \ldots, n)$,

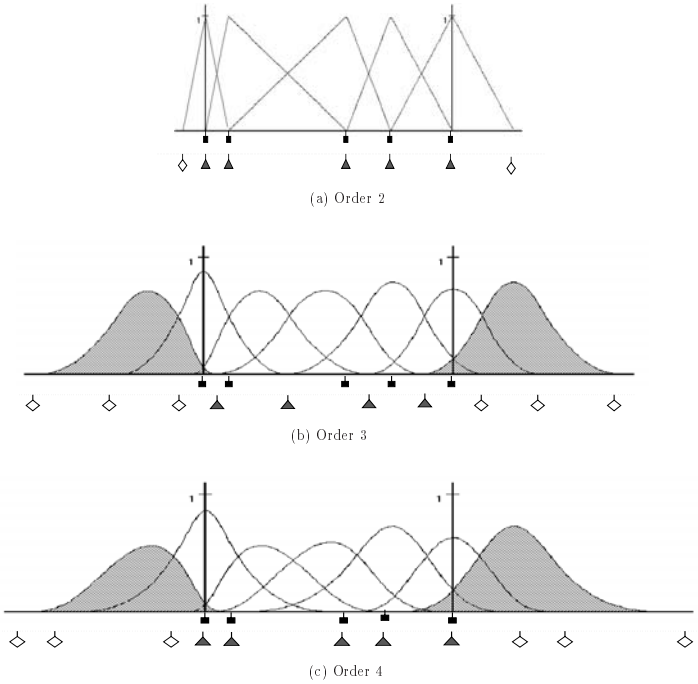


Fig. 2. Nonuniform B-functions of different orders defined by the same knot vector for *real* and *virtual* linguistic terms (*Core support points*: rectangle; *Iknots*: triangle; *Eknots*: diamond; *virtual linguistic terms*: shaded).

- k_j: the order of the B-spline basis functions used for x_j,
- $X^j_{i_j,k_j}$: the ith linguistic term of x_j defined by B-spline basis functions,
- $i_j = 1, \ldots, m_j$, representing how fine the jth input is fuzzy partitioned,
- $Y_{i_1 i_2 \ldots i_n}$: the control vertex (de Boor points) [2] of the output [3] if $Rule(i_1, i_2, \ldots, i_n)$ fires 100%.

Then, the output y of a MISO fuzzy controller is [15]:

$$\begin{aligned} y &= \frac{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j))}{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)} \\ &= \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)) \end{aligned} \tag{3}$$

[3] The THEN-part has the same form of a zero-order TSK model.

This is a multivariate B-spline model, representing a general NUBS (nonuniform B-spline) *hypersurface*.

2.4 Rule Weighting

Some fuzzy control systems use a weight for each rule. This enables more flexibility for shaping the control surface in a Mamdani type controller, but it results in more work for fine-tuning. In fact, if we add one more rule weight to each rule in a B-spline fuzzy controller, then this controller corresponds to a NURBS (*Non-Uniform Rational B-Spline*) model:

$$y = \frac{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} w_{i_1,\ldots,i_n} Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X_{i_j,k_j}^{j}(x_j)}{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} w_{i_1,\ldots,i_n} \prod_{j=1}^{n} X_{i_j,k_j}^{j}(x_j)}$$

Experience of using B-splines in CAD shows that by using sufficient B-spline basis functions, NUBS of nonrational form may approximate any shape to a given precision, [3]. NURBS curves and surfaces are mainly used for exactly modeling special analytical functions like a circle, square, etc. The control vertex of each rule in a B-Spline fuzzy controller plays the role of the rule weight as well as the control action. Therefore we adopt the NUBS of nonrational form for constructing fuzzy controllers.

2.5 Acceleration of Rule Evaluation

The index coding of the B-functions makes the evaluation of fuzzy rules highly efficient. For an input $x \in (x_i, x_{i+1})$, it is known that exactly k linguistic terms , i.e. k B-functions $X_{i,k}(x), X_{i-1,k}(x), \ldots, X_{i-k+1,k}(x) > 0$ will be activated. All the other linguistic terms are not unactivated. In the whole rule base with n inputs, exactly k^n rules fire for any given input vector in the universe of discourse.

3 Learning of B-spline Fuzzy Controllers

3.1 Properties of Parameter Learning

Computing parameters of such a B-spline fuzzy system is divided into two steps: one for the IF-part and the other one for the THEN-part.

Properties of modelling the IF-part are:

- Based on the granularity of the input space and the distribution of extrema in the control space (if known), the fuzzy sets can be generated using the recursive computation of B-spline basis functions. This approach provides an automatic approach to generate the information granularity as proposed by Zadeh [14].
- These fuzzy sets can be further adapted during the generation of the whole system by modifying the CSPs.

Properties for generating the THEN-part are:

- Fuzzy singletons represented by control vertices can be initialised with the values acquired from expert knowledge. These parameters will be fine-tuned by a learning algorithm.
- For supervised learning, we show in the following that the squared errors with respect to control vertices are convex functions. Therefore, rapid convergence for supervised learning is guaranteed. The control space changes locally due to the "local support" property of B-functions while the control vertices are modified. Based on this feature, the control vertices can be optimised gradually, area-by-area.

3.2 Supervised Learning of Control Vertices

A fuzzy system constructed with the above approach can be optimised with the gradient descent approach. Assume $\{(\mathbf{X}, y_d)\}$ is a set of training data, where $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ is a training data vector, and y_d is the desired output for $\mathbf{X}$.

We apply the following squared error function:

$$E = \frac{1}{2}(y_r - y_d)^2 \tag{4}$$

where y_r is the current real output value during training computed with (3).

The parameters to be found are $Y_{i_1,i_2,\ldots,i_n}$ that minimise the error in (4). Each control vertex $Y_{i_1,\ldots,i_n}$ can be modified by using the gradient descent method:

$$\begin{aligned} \Delta Y_{i_1,\ldots,i_n} &= -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} \\ &= -\epsilon (y_r - y_d) \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) \end{aligned} \tag{5}$$

where $0 < \epsilon \leq 1$ represents the learning rate.

The gradient descent method guarantees that the learning algorithm converges to the global minimum of the error function because the second partial differentiation with respect to $Y_{i_1,i_2,\ldots,i_n}$ is constant:

$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\ldots,i_n}} = -\epsilon \Big(\prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) \Big)^2 \geq 0. \tag{6}$$

This means that the error function (4) is convex in the space $Y_{i_1,i_2,\ldots,i_n}$ and therefore possesses only one minimum.

3.3 Adaptation of the Core Support Points

Since inserting of one more B-function results in only one more knot (or CSP), more accurate approximation can be achieved by using more B-functions. For the purpose of fully utilising a limited number of B-functions to achieve better approximation results, we developed an algorithm for finding the optimal placements of the *core support points* (CSPs). This algorithm can be viewed as a modified algorithm for self-organising neural networks, [9].

Applying a new training input vector $\mathbf{x}$, its left and right neighbour, two CSP vectors noted as $\mathbf{x_l}$ and $\mathbf{x_r}$, can be found and the output values of the controller at these two points, denoted as y_l and y_r, are computed. If the desired training data y^d at point $\mathbf{x}$ is larger or smaller than a threshold $\theta \geq 0$ compared with y_l (y_r), a modification of $\mathbf{x_l}$ ($\mathbf{x_r}$) is necessary. We assume $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ are the CSP vectors of the input data sequence, and $\mathbf{X_{1,k}}, \mathbf{X_{2,k}}, \ldots, \mathbf{X_{m,k}}$ are their corresponding linguistic terms (B-functions of order k). The adaptation algorithm can be described as follows:

(1) Apply a new training input–output pair $(\mathbf{x}, y^d)$.

(2) Find two neighbouring CSPs $\mathbf{x_l}, \mathbf{x_r}$, so that
$\mathbf{x_i} \leq \mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_r} \leq \mathbf{x_j}$, for $i = 1, \ldots, l-1$ and $j = r+1, \ldots, m$
Compute y_l, y_r. Assume that $y_l \leq y_r$ [4].
(3) If $y^d \leq y_l - \theta$:
Modify $\mathbf{x_l}$: $\mathbf{x_l} = \mathbf{x_l} + (\mathbf{x} - \mathbf{x_l}) \cdot \mathbf{X_{l,k}}(\mathbf{x})$
If $y_r + \theta \leq y^d$:
Modify $\mathbf{x_r}$: $\mathbf{x_r} = \mathbf{x_r} + (\mathbf{x} - \mathbf{x_r}) \cdot \mathbf{X_{r,k}}(\mathbf{x})$
(4) Optimise the control vertices of the output variables (according to the gradient descent method in section 3.2).
(5) If there are still training data, continue with (1).
(6) Terminate.

As an example we show the approximation of a function $\sin(\mathbf{2}\pi\mathbf{x^2})$. The function has a minimum at $x = 0.86$ and a maximum at $x = 0.5$. Initially, five CSPs are evenly distributed over the interval $[0, 1]$. All five control vertices are set to 0. The training data are randomly generated from the interval $[0, 1]$. In the following implementation, we select θ as 0.2. The dashed curve represents the desired values, the solid curve the output of the controller. The points depicted as "$\diamond$" are the control vertices, whose abscissas are the CSPs.

3.4 Steps for Developing a B-Spline Fuzzy Controller

The steps for developing a fuzzy controller with B-spline models can be summarised as follows (M: manually, A: automatically):

(1) Select inputs. (M/A)

(2) Select the order of the B-functions for each input variable. (M/A)

(3) Determine the CSPs for partitioning each input variable. (M/A)

(4) Compute the real and virtual linguistic terms for all inputs. (A)

(5) Initialise the control vertices for the output. (M/A)

(6) Adapt the control vertices. (A.)

[4] If $y_l > y_r$, then step (3) is symmetric.

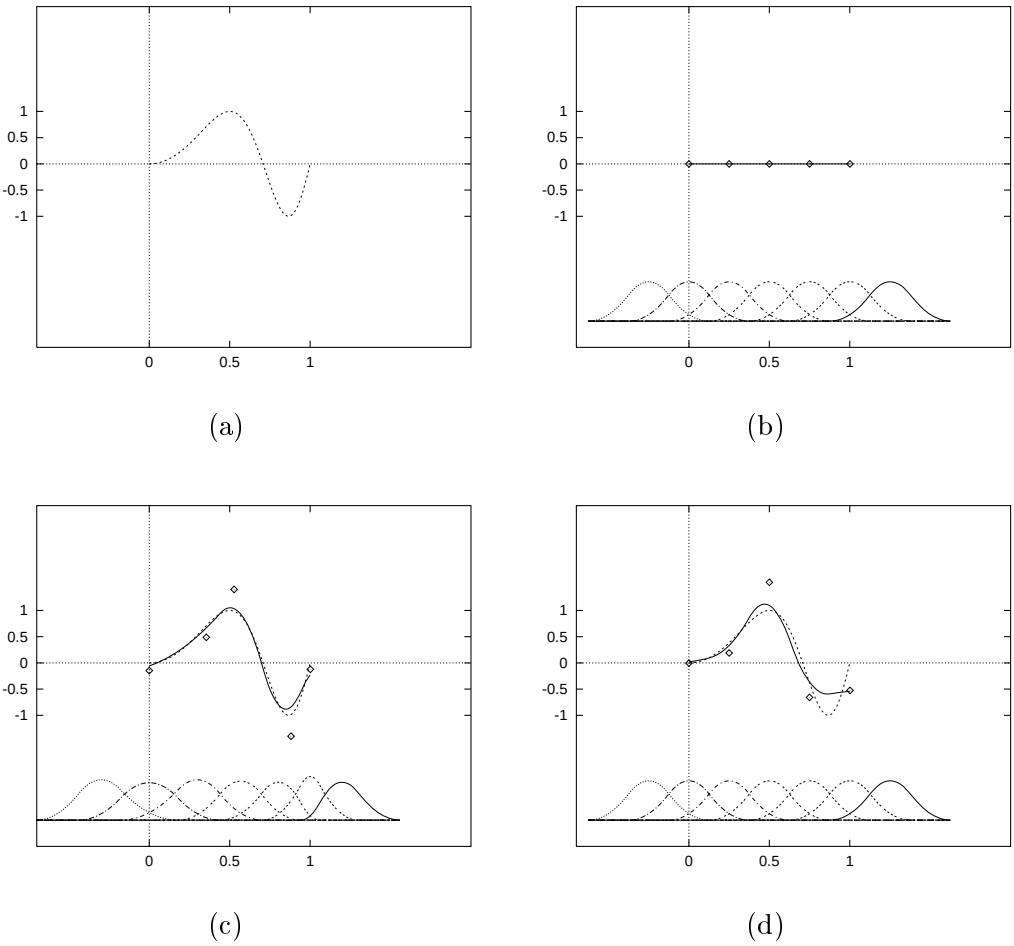


Fig. 3. Approximation of the function $y = \sin(2\pi x^2)$ (the horizontal axis represents x, covered with B-functions, the vertical axis represents the output value y). (a) The function $\sin(2\pi x^2)$. (b) The initial controller output set as zero. (c) The output curve after 2000 epochs with CSP adaptation. (d) The output curve after 2000 epochs without CSP adaptation.

(7) If the results are satisfying, terminate. (A)

(8) Modify the CSPs for inputs (M/A), go to (4);

or Refine the granularity and use more training data (M/A), go to (3);

or Increase the order of B-functions (M/A), go to (3);

or Delete certain inputs and/or add new ones (M/A), go to (2).

In step (3) it is very important to know how the CSPs should be placed in the input space. An intuitive answer is to place the CSPs where the output has its extrema. If such information is available, we can apply this principle to select the CSPs. If the output of a control system is unknown, the CSPs can first be equally distributed and then adapted with the approach described in section 3.3. In step (5), the control vertices can be initialised with the approximate *a priori* values, e.g., empirical data from experts if available. Otherwise they can be just set to zero.

4 Numerical Examples

The examples in [5] and [8] were implemented with our approach to demonstrate the learning ability for function approximation and system identification. In the following figures, the modeling of the linguistic terms using B-functions are not extra shown. Instead we use the representation $a_1 \times a_2 \times a_3 \times \ldots$ to describe how many linguistic terms are applied for the input variable $x_1, x_2, x_3, \ldots$ of the used fuzzy controller. Our observation mainly focuses on the learning process of the input–output relation, i.e. the control surface, the training error, and the checking error.

4.1 Approximation of a Function with Two Variables $z = \frac{\sin(x)}{x} * \frac{\sin(y)}{y}$

The training data were uniformly selected from the area [-10,10]×[-10,10], altogether 121 training vectors $((x, y), z)$ were obtained. Since this function is symmetric, the two input variables are covered with the same number of linguistic terms of order three. Figure 4(a)–(d) illustrate the training results. It can be found that by using 11 B-functions with 14 parameters for defining knots for each input, the output approximates the original function quite accurately after only 20 epochs of learning (the ANFIS model in [5] needs 450 epochs).

4.2 Prediction of a Chaotic System

A chaotic time series is generated with the following discrete Mackey-Glass equation [5]:

$$x(t+1) = \frac{0.2x(t-r)}{1+x^{10}(t-r)} + 0.9x(t)$$

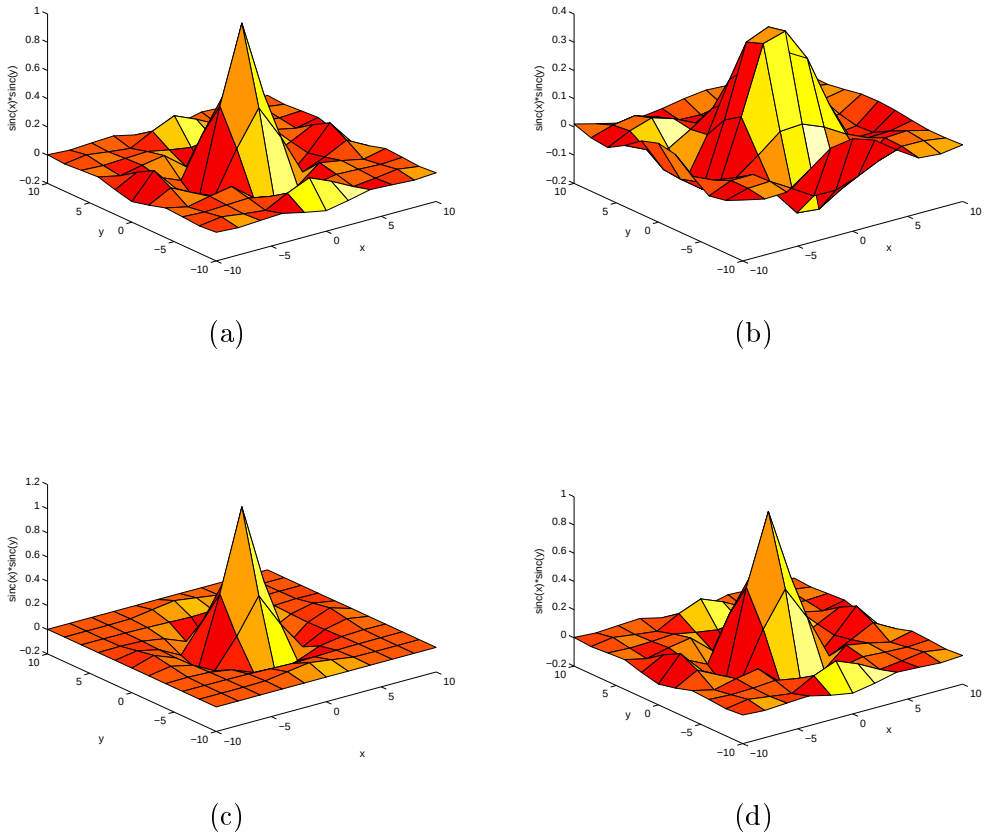
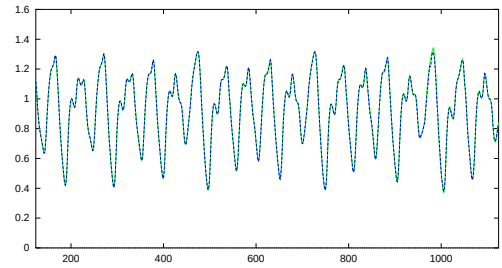


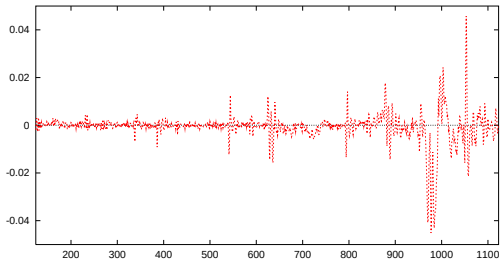
Fig. 4. Approximation of the function $z = sin(x)/x * sin(y)/y$. (a) Training data. (b) Output of a 5×5 controller after 1000 epochs. (c) Output of a 7×7 controller after 100 epochs. (d) Output of a 11×11 controller after 20 epochs.

1000 training data are selected using the method in [5]. 500 are used as training data, other 500 as checking data. The fuzzy controller has four inputs. The RMSE (Root Mean Square Error) measures the approximation error, see Figure 5. For this problem, our algorithm needs only 40 minutes on a SUN Sparc-4 workstation for the $10 \times 10 \times 10 \times 10$ fuzzy controller with 10,000 parameters.

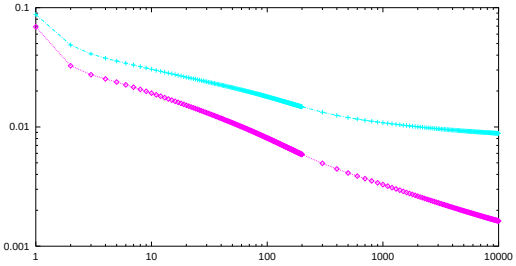
For this problem, our algorithm needs only 40 minutes on a SUN Sparc-4 workstation for the $10 \times 10 \times 10 \times 10$ fuzzy controller with 10,000 parameters.



(a)



(b)



(c)

Fig. 5. Emulation of a chaotic time series with a 10×10×10×10 fuzzy controller. (a) $x(t)$, the output of the trained controller. (b) Approximation error / time. (c) RMSE / epochs (the bottom curve: the training error; the top curve: the checking error).

4.3 Identification of a Nonlinear System

The task is to identify a nonlinear component in a dynamic system: $y(t+1) = 0.3y(t) + 0.6y(t-1) + f(u(t))$, where $y(t)$ is the output of moment t and $u(t)$ is the input, see [5].

In this simulation $f(\cdot)$ has the the form: $f(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u)$.

The input signal is:

$$u(t) = \begin{cases} \sin(2\pi t/250), & \text{for } t \leq 500, \\ \\ 0.5\sin(2\pi t/250) + 0.5\sin(2\pi t/25), & \text{for } t > 500. \end{cases}$$

B-spline fuzzy controllers of order three are trained with 250 data from $t = 1$ to $t = 250$. The data from $t = 251$ to $t = 750$ are used for checking. Figure 6 shows the results with 30 B-functions after one and ten training epochs. In Figure 6(a), (c) and (e), the lighter curves represent the desired data, while in Figure 6(b), (d) and (f), it is shown that these curves are accurately approximated.

4.4 Test of a Set of Sample Functions

We also tested our approach on the following sample functions used in [8]:

One-dimensional functions:

$$\begin{aligned}
f_1(x) &= 3x(x-1)(x-1.9)(x-0.7)(x+18), \quad \text{for} \quad -2 \leq x \leq 2 \\
f_2(x) &= 10\tan^{-1}\left(\frac{(x-0.2)(x-0.7)(x+0.8)}{(x+1.4)}\right), \quad \text{for} \quad -1 \leq x \leq 1 \\
f_3(x) &= \frac{100(x+0.95)(x+0.6)(x+0.4)(x-0.1)(x-0.4)(x-0.8)(x-0.9)}{(x+1.7)(x-2)^2}, \\
&\quad \text{for} \quad -1 \leq x \leq 1 \\
f_4(x) &= 8\sin(10x^2+5x+1), \quad \text{for} \quad -1 \leq x \leq 1 \\
f_5(x) &= 10\tan^{-1}\left(\frac{(x-0.2)(x-0.7)(x+0.8)}{(x+1.4)(x-1.1)x+0.7}\right), \quad \text{for} \quad -1 \leq x \leq 1 \\
f_6(x) &= 10\left(e^{-5|x|} + e^{-3|x-0.8|/10} + e^{-10|x+0.6|}\right), \quad \text{for} \quad -1 \leq x \leq 1
\end{aligned}$$

Two-dimensional functions:

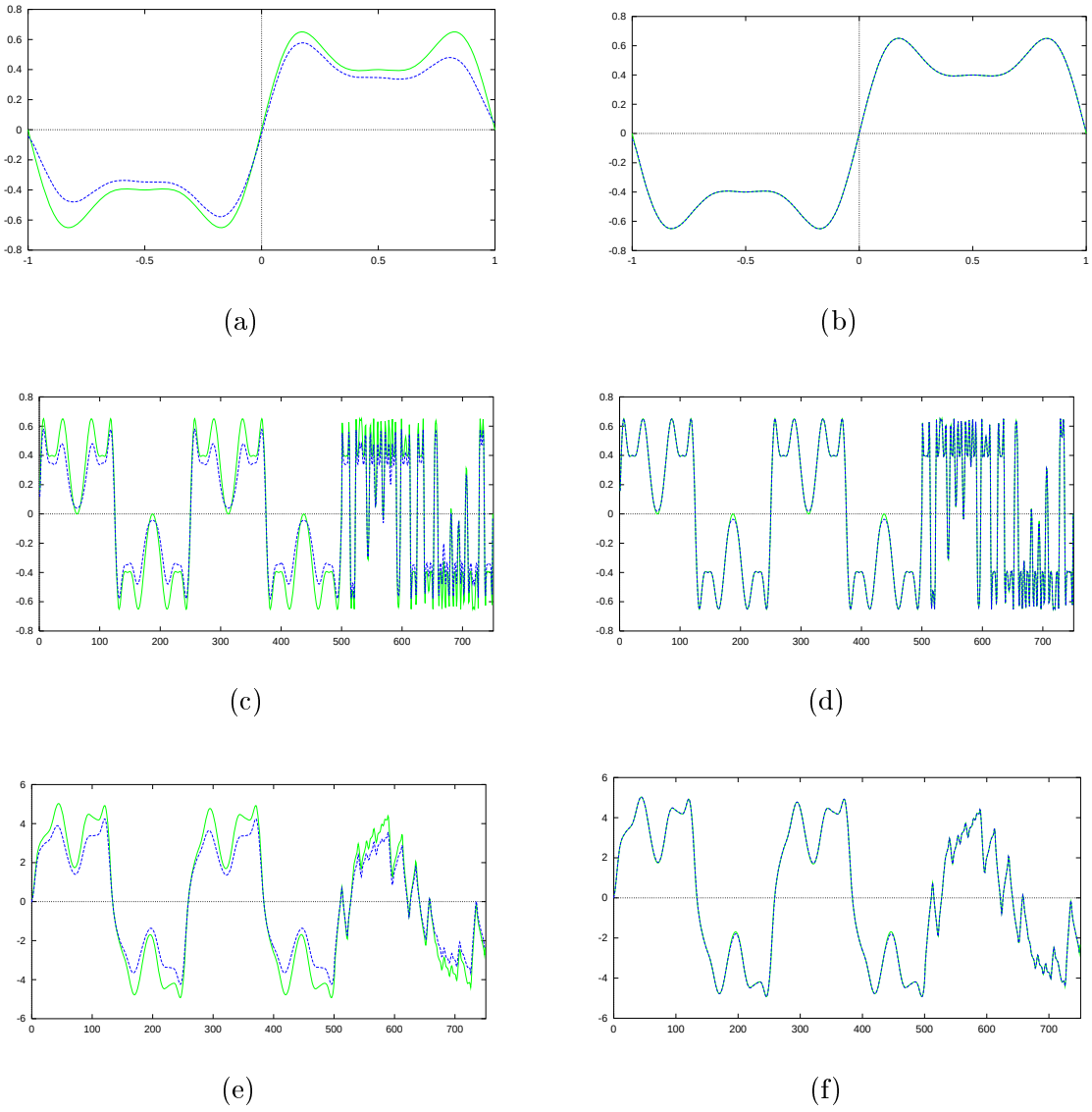


Fig. 6. Results of the fuzzy controller with 30 B-functions after learning. (a) The nonlinear component f(u) after 1 epoch. (b) $f(u)$ after 10 epochs. (c) $f(t)$ after 1 epoch. (d) $f(t)$ after 10 epochs. (e) System output $y(t+1)$ after 1 epoch. (f) System output $y(t+1)$ after 10 epochs.

$$g_1(x,y) = f_2(x) \times 10\left(\sin(4y+0.1) + \sin(11y-0.2) + \sin(14y) + \sin(17y+0.3)\right),$$
$$\text{for} \quad -1 \le x, y \le 1$$
$$g_2(x,y) = f_4(x) \times 2\left(e^{-\left(\frac{y-0.1}{0.25}\right)^2} - 0.8e^{-\left(\frac{y+0.75}{0.15}\right)^2} - 0.4e^{-\left(\frac{y-0.8}{0.1}\right)^2}\right),$$
$$\text{for} \quad -1 \le x, y \le 1$$
$$g_3(x,y) = f_1(x) \times \sin(y), \quad \text{for} \quad -2 \le x, y \le 2$$

Three-dimensional functions:

$$h_1(x,y,z) = 0.1\left(e^{-\frac{|x|}{0.2}} + e^{-\frac{|x-0.8|}{0.3}} + e^{-\frac{|x+0.6|}{0.1}}\right) \times \left(\tan^3(1.5y) + 10\tan^2(y) - 20\tan(0.7y)\right) \times \left(\arccos^3(z) - \arccos^2(-z) - \arccos(-z)\right), \quad \text{for} \quad -1 \le x,y,z \le 1$$

$$h_2(x,y,z) = e^{-(xy-0.7)(xz-0.5)} \times \frac{\sin\left(\frac{125}{(x+1.5)}\right)}{y+1.1} \times \frac{(5xy^3 - 6z^3)}{(xyz+2)} \tan^{-1}(10xy + z^3), \quad \text{for} \quad -1 \le x,y,z \le 1$$

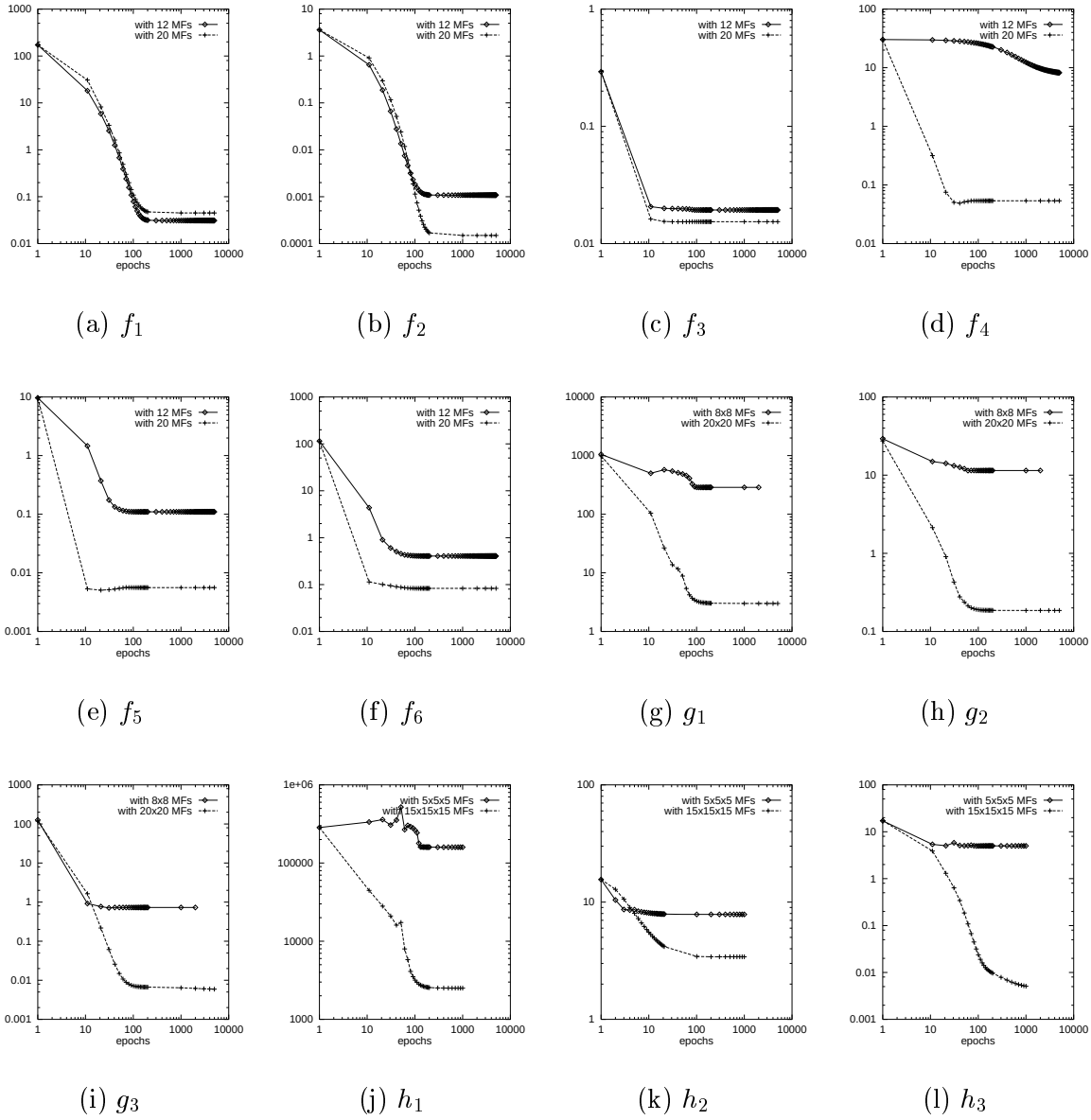
$$h_3(x,y,z) = 1000(x+0.95)(x+0.6)(x+0.4) \times (x-0.1)(x-0.4)(x-0.8)(x-0.9)(y+0.7) \times (y-0.35)(y-0.9)(z+0.7)(z+0.2)(z-0.4) \times (yz+0.6)(x+yz), \quad \text{for} \quad -1 \le x,y,z \le 1$$

These test functions were approximated with the fuzzy controller based on the model proposed in the above sections. To approximate each function, we first used the same number of linguistic terms as in [8], then some more. In both cases the computation times for evaluation of the rule base are the same. Here we need to emphasise that in our approach inserting one more MF results in only one more knot (or CSP), while adding one triangle MF results in three more parameters, one trapezoid four parameters, one Gaussian function two parameters, etc. Curves of the mean-squared error by using B-functions of order three are plotted in Figure 7. The comparison of our results in Figure 7 with the results in [8] shown that by using the same number of parameters for defining MFs, the B-spline model performs better in most cases.

5 Discussion and Conclusions

5.1 Supervised and Unsupervised Learning

The proposed approach is suitable for on-line learning thanks to learning rate. We successfully applied it to supervised learning of the "truck backer-upper" problem (see [12]) and "inverse kinematics" problem in robotics. The learning approach may also be generalised for unsupervised learning by designing a suitable *evaluation function* and using such information to modify the control vertices. We have applied this approach to mobile robot control and sensor-based assembly operations with robot arms. For further details see [17,16]. Our current work is on extending this approach to the learning problem of multivariate systems.



(a) f_1 (b) f_2 (c) f_3 (d) f_4

(e) f_5 (f) f_6 (g) g_1 (h) g_2

(i) g_3 (j) h_1 (k) h_2 (l) h_3

Fig. 7. Mean-squared error of approximating functions of f_1 to f_6 (one-dimensional), g_1 to g_3 (two-dimensional) and h_1 to h_3 (three-dimensional), by using B-functions of order three.

5.2 The Curse of Dimensionality

Like all other types of fuzzy controllers, B-spline type controllers cannot avoid the problem of the number of rules increasing exponentially with the number of inputs. However, as shown in section 2.5, the evaluation time of the whole rule base can be reduced from m^n

to k^n, where m is the number of linguistic terms for input and k the order of B-functions. k is usually selected to be two or three for most applications. The adaptation of CSPs also contributes to the efficient utilisation of linguistic terms.

5.3 Conversion Back to Mamdani Type

For some applications, e.g. data mining or qualitative analysis, we may find the large number of fuzzy singletons too complex to be interpreted and may want to approximately transform them into a smaller number of linguistic terms. Such a transformation can best be realised by fuzzy c-means clustering: given the number of linguistic terms we want, the fuzzy singletons can be grouped naturally into fuzzy sets. This represents a fuzzy partition of the output variable. In this way, our approach can optimally combine the numerical interpolation with linguistic interpretation.

5.4 Summary

We proposed a novel approach for constructing fuzzy controllers with B-spline basis functions and learning of the control vertices for the THEN-parts of fuzzy rules. If the rule table is complete, then, by adding certain more marginal rules, a high smoothness of the controller output can be achieved by selecting the proper order of basis functions. B-spline fuzzy controllers are exact, meaning that no information is lost after the defuzzification. Although the number of control vertices to be optimised can be quite large with our approach, the learning process of such a fuzzy controller converges rapidly thanks to the one-minimum property of the error function and the efficient evaluation of the rule base. Our implementations show that this approach is very promising for a wide range of applications in adaptive modeling and control.

References

[1] H. Bersini and G. Bontempi. Now comes the time to defuzzify neuro-fuzzy models. *Proceedings of the FLINS Workshop on Intelligent Systems and Soft Computing for Nuclear Science and Industry*, pages 130–139, 1996.

[2] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[3] T. Dokken, V. Skytt, and A. M. Ytrehus. *The role of NURBS in geometric modelling and CAD/CAM.* in: "Advanced Geometric Modelling for Engineering Applications", edited by Krause, F.-L.; Jansen, H. Elsevier, 1990.

[4] D. Driankov, H. Hellendoorn, and R. Palm. *Some Research Directions in Fuzzy Control*, chapter 11, pages 281–312. in: "Theoretical Aspects of Fuzzy Control", edited by H. T. Nyuen, M. Sugeno, R. Tong, and R. R. Yager, John Wiley & Sons, New York, 1995.

[5] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man and Cybernetics*, 23(3):665–685, 1993.

[6] B. Kosko and J. A. Dickerson. *Function Approximation with Additive Fuzzy Systems*, chapter 12, pages 313–347. in: "Theoretical Aspects of Fuzzy Control", edited by H. T. Nyuyen, M. Sugeno and R. R. Yager, John Wiley & Sons, 1995.

[7] E. H. Mamdani. Twenty years of fuzzy control: Experiences gained and lessons learned. *IEEE International Conference on Fuzzy Systems*, pages 339–344, 1993.

[8] S. Mitaim and B. Kosko. What is the best shape of a fuzzy set in function approximation. *IEEE International Conference on Fuzzy Systems*, 1996.

[9] H. Ritter, T. Martinetz, and K. Schulten. *Neuronale Netze.* Addison-Wesley, 1991.

[10] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modelling and control. *IEEE Transactions on System, Man and Cybernetics*, SMC-15(1):116–132, 1985.

[11] T. Terano, K. Asai, and M. Sugeno. *Applied Fuzzy Systems.* AP Professional, Cambridge, MA, 1994.

[12] L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[13] J. Yen, R. Langari, and L. A. Zadeh. *Industrial Applications of Fuzzy Logic and Intelligent Systems.* IEEE Press, 1994.

[14] L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, 1996.

[15] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems (Forthcoming)*, 1997.

[16] J. Zhang, K. V. Le, and A. Knoll. Unsupervised learning of control spaces based on B-spline models. *Proceedings of IEEE International Conference on Fuzzy Systems*, 1997.

[17] J. Zhang, Y. von Collani, and A. Knoll. On-line learning of sensor-based control for acquiring assembly skills. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.