A New Type of Fuzzy Logic System for Adaptive Modelling and Control

J. Zhang, A. Knoll and K. V. Le

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

Abstract. We present a new type of fuzzy controller constructed with the B-spline model and its applications in modelling and control. Unlike the other normalised parameterised set functions for defining fuzzy sets, B-spline basis functions do not necessarily span from membership value 0 to 1, but possess the property "partition of unity". These B-spline basis functions are automatically determined after the input space is partitioned. By using "product" as fuzzy conjunction, "centroid" as defuzzification, "fuzzy singletons" for modelling output variables and adding *marginal linguistic terms*, fuzzy controllers can be constructed which have advantages like smoothness, automatic design and intuitive interpretation of controller parameters. Furthermore, both theoretical analysis and experimental results show the rapid convergence for tasks of data approximation and unsupervised learning with this type of fuzzy controller.

1 Introduction

In most fuzzy systems, linguistic terms are defined with *fuzzy numbers*, i.e. normalised, closed, convex fuzzy sets. In approximate reasoning, usually only the qualitative information is referred to, thus the final result is not very sensitive to the shape and the height of the fuzzy sets. However, if a fuzzy logic system is applied in modelling or control problems, the shape as well as the position of fuzzy sets are implicitly or explicitly synthesised 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 true fuzzy sets for modelling linguistic terms for each input. The input space is partioned into overlapping cells, which reflects the vague modelling of linguistic concepts on one side and enables the continuous transition of output values on the other side.

The IF-part of a rule is generally modelled as

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

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

* Published in *Proceedings of the International Conference on Computational Intelligence*, Dortmund, 1997. Springer Verlag.

While discrete representation of fuzzy sets avoids the on-line function evaluation on a fuzzy hardware chip, parameterised representation is adopted in fuzzy controllers running on a general-purpose, non-fuzzy computer architecture. In up-to-date control applications, mainly triangle and trapezoid set functions are used. Recently, Fuzzy Basis Functions based on Gaussian functions are also proposed for function approximation, [7]. In [6] several exotic functions like "Cauchy", "*sinc*", "Laplace", "Logistic", "Hyperbolic Tangent" are introduced and their abilities of function approximation are compared[1]. However, all the above set functions need additional special parameters apart from the partition positions (called *knots* in the following) on the universe of discourse of each input. Since the knots are the only 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 is based on the idea of directly using symbolic rules for control tasks. A rule has the form

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

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 modelling the output y. The aggregation of output values of all the firing rules are realised either by the "max"-operator [5] or simple addition [4], where the second method is a small variation of the first one and even more simple to compute.

Another important type of fuzzy controllers is based on the TSK (Tagaki-Sugeno-Kang) model. A rule using a TSK model of order 1 looks like:

$$\boxed{\begin{array}{ll} \text{IF} & (x_1 \text{ is } A^1_{i_1}) \text{ and } (x_2 \text{ is } A^2_{i_2}) \text{ and } \ldots \text{ and } (x_n \text{ is } A^n_{i_n}) \\ \text{THEN} & y = a^i_0 + a^i_1 x_1 + \cdots + a^i_n x_n, \end{array}}$$

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 the data acquired by observing a skilled human operator's control action. The 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, the knowledge acquisition with this model is indirect and not intuitive.

We propose an approach which can systematically build the fuzzy sets for linguistic terms of the IF-part while the fuzzy sets of the THEN-part can be adapted through learning. The model of linguistic terms in our approach is based on B-spline basis functions, a special set of piecewise polynomial curves.

[1] The experimental results in [6] show that the non-convex functions *sinx* works generally better for a quick and accurate function approximation. Nevertheless, this function possesses more than one peak and thus cannot be assigned an appropriate linguistic meaning.

2 B-Spline Basis Functions as Fuzzy Sets

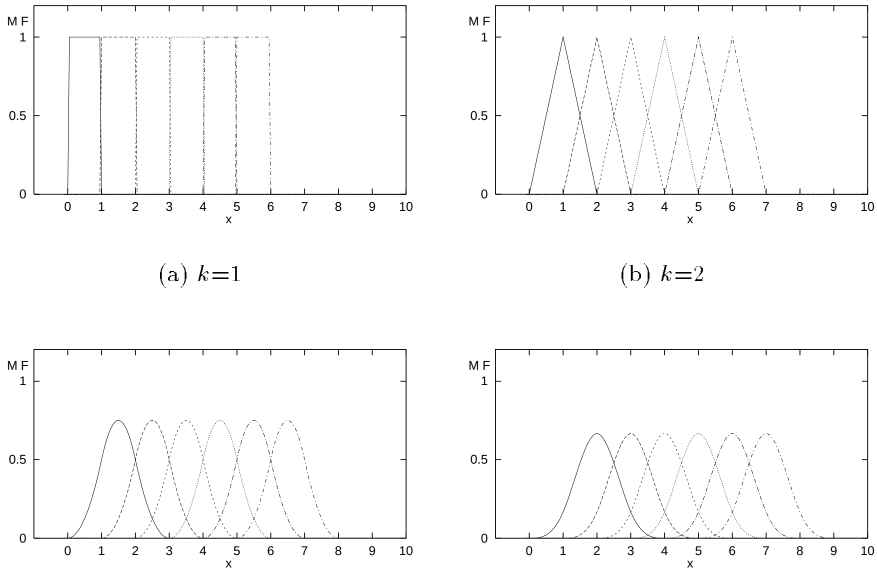
2.1 Definition

In our previous work [9] we compared the basis functions of periodical *Non-Uniform B-Splines* (NUBS) with a fuzzy controller. In this paper, we also follow the usage of this type of NUBS basis functions (*B-functions* for short).

Assume x is a general input variable of a control system which 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 $i-$th normalised B-spline basis function (B-function) $X_{i,k}$ of order k is defined as:

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

with $i = 0, 1, \ldots, m - k$. The B-functions of order 1, 2, 3, 4 are illustrated in Fig. 1.



(a) k=1 (b) k=2

(c) k=3 (d) k=4

Fig. 1. Fuzzy sets defined by B-spline basis functions of different orders (examples of uniform cases).

The important properties of B-functions are:

Partition of unity:	$\sum_{i=0}^{m} X_{i,k}(x) = 1$.
Positivity:	$X_{i,k}(x) \geq 0$.
Local support:	$X_{i,k}(x) = 0$ for $x \notin [x_i, x_{i+k}]$.
C^{k-2} *continuity:*	If the knots $\{x_i\}$ are pairwise different from each other, then $X_{i,k}(x) \in C^{k-2}$, i.e. $X_{i,k}(x)$ is $(k-2)$ times continuously differentiable.

2.2 Real and Virtual Linguistic Terms

It is assumed that linguistic terms are to be defined over $[x_0, x_m]$, the universe of an input variable x of a fuzzy controller. They are referred to as *real linguistic terms*. In order to maintain the "partition of unity" for all $x \in [x_0, x_m]$, some more B-functions should be added at both ends of $[x_0, x_m]$. They are called *marginal B-functions*, defining *virtual linguistic terms*. *Real* and *virtual* linguistic terms are denoted as A_i in Fig. 2:

- In case of order 2, no marginal B-function is needed, Fig. 2(a).
- In case of order 3 or 4, two marginal B-functions are needed, one for the left end and another for the right end, Fig. 2(b), (c).
- Generally, $((k+1)\ div\ 2)$ marginal B-functions are needed.

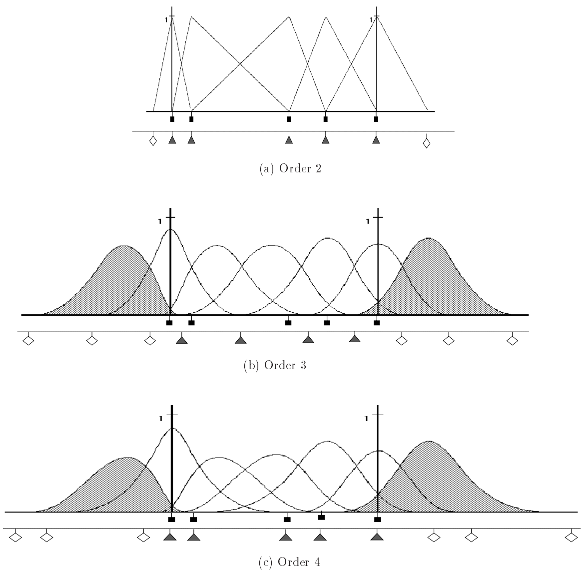


Fig. 2. Non-uniform B-functions of different orders defined for *real* and *virtual* linguistic terms by the same knot vector (*Peak supprot points*: □; *Iknots*: △; *Eknots*: ◊; *virtual linguistic terms*: shaded).

2.3 Peak 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 non-zero membership grades in A. If a B-function of order k ($k > 1$) is used for modelling a fuzzy set, it possesses only one peak which has the largest membership grade. The support point of this peak, denoted as the $Psupp$-point (*peak support point*), can be defined as $Psupp(A) = \{x|A(x) = maximum\}$, 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*) which lie within the universe of discourse and *Extended knots* (*Eknots*) which are generated at both ends of the universe for defining the marginal linguistic terms. Generally, $m - (k\ mod\ 2)$ interior knots are needed, where

- m is the number of the real linguistic terms, and
- k is the order of the B-functions ($k \leq m$).

If k is even, the interior knots coincide with the *Psupp*-points. If k is odd, the $m - 1$ interior knots can be determined by

$$Iknot_i \ = \ Psupp_i \ + \ \frac{Psupp_{i+1} \ - \ Psupp_i}{2}, \ i = 1, \ldots, m-1. \qquad (1)$$

At each end of the universe of discourse $[Psupp_1,\ Psupp_m]$, $((k\ +\ 1)\ div\ 2)$ *Eknots* can be determined by reflecting the *Iknots* with respect to $Psupp_1$ and $Psupp_m$. Altogether there are $k\ +\ m$ *knots*.

If the $Psupp$-points are chosen evenly, i.e. $Psupp_{i+2} - Psupp_{i+1} = Psupp_{i+1} - Psupp_i$ for $i = 1, \ldots, m-2$, the B-functions are *uniform* (Fig. 1), otherwise they are *non-uniform* (Fig. 2).

3 B-Spline Model for Fuzzy Controllers

3.1 Core and Marginal Rules

We define the *core rules* as linguistic rules which use *real linguistic terms*. If *virtual linguistic terms* appear in the IF-part, in order to maintain the output continuity at both ends of the universe of x, additional rules are needed to describe the control action for these cases. Since these rules use the *virtual linguistic terms* which are defined by membership functions neighbouring the ends of the universe of each variable, they are called *marginal rules*. The output value of each *marginal rule* is selected just as the output value of the "nearest" *core rule*, i.e. the rule using the directly adjacent linguistic terms in its IF-part (Fig. 3).

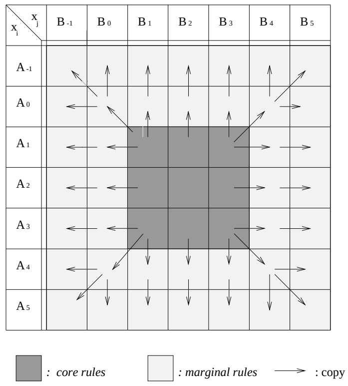


Fig. 3. The outputs of the marginal rules are copied from that of the neighbouring core rules.

3.2 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. Under the following conditions:

- periodical B-spline basis functions as membership functions for inputs;
- fuzzy singletons as membership functions for outputs;
- "product" as fuzzy conjunctions;
- "centroid" as defuzzification method;
- addition of "virtual linguistic terms" at both ends of each input variable and
- extension of the rule base for the "virtual linguistic terms" by copying the output values of the "nearest" neighbourhood;

the computation of the output of such a fuzzy controller is equivalent to that of a *general B-spline hypersurface*. Generally, we consider a MISO system with n inputs $x_1, x_2, \ldots, x_n$. A $Rule(i_1, i_2, \ldots, i_n)$ with the n conjunctive terms in the IF-part is given in the following form:

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

where

- x_j: the j-th input $(j = 1, \ldots, n)$,
- k_j: the order of the B-spline basis functions used for x_j,
- $X^j_{i_j,k_j}$: the i-th linguistic term of x_j defined by B-spline basis functions,
- $i_j = 1, \ldots, m_j$, representing how fine the j-th input is fuzzy partitioned,
- $Y_{i_1 i_2 \ldots i_n}$: the control vertex (deBoor points) of $Rule(i_1, i_2, \ldots, i_n)$.

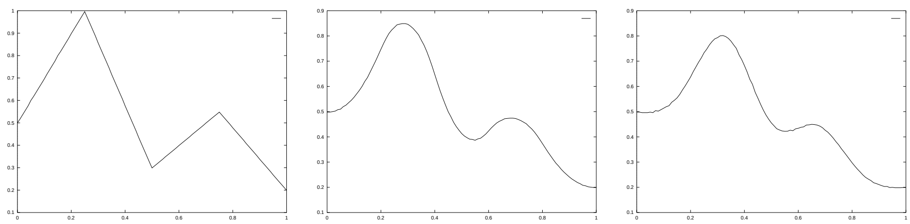
Then, the output y of a MISO fuzzy controller is:

$$y = \frac{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j))}{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)} \tag{2}$$

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

This is called a *general NUBS hypersurface*, which possesses the following properties:

- If the B-functions of order $k_1, k_2, \ldots, k_n$ are employed to specify the linguistic terms of the input variables $x_1, x_2, \ldots, x_n$, it can be guaranteed that the output variable y is $(k_j - 2)$ times continuously differentiable with respect to the input variable $x_j, j = 1, \ldots, n$ (see Fig. 4 for an example).
- If the input space is partitioned fine enough and at the correct positions, the interpolation with the B-spline hypersurface can reach a given precision.



(a) order 2: C^0-continuous

(b) order 3: C^1-continuous

(c) order 4: C^2-continuous

Fig. 4. Trajectory of the output y with respect to the input x. The core rule set of 5 rules: $CRS = \{\text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } Y_i, i = 1, \ldots, 5\}$, where the universe of discourse of x is defined on $[0, 1]$ and Y_1 to Y_5 are fuzzy singletons with the following values: 0.5, 1.0, 0.3, 0.55, 0.2.

We will show later in section 4 that the output of the fuzzy controller can be flexibly adapted to anticipated values by adjusting the positions of the fuzzy singletons (control vertices) of the *core rules* after the order of the B-functions and the linguistic terms used in the IF-part are chosen.

3.3 SISO Systems

To illustrate the corresponding procedures of fuzzification, inference and defuzzification with our B-spline fuzzy controller, we first consider a system with single input and single output (SISO), Fig. 5. For clarity, we suppose the input is covered with four B-functions of order 2.

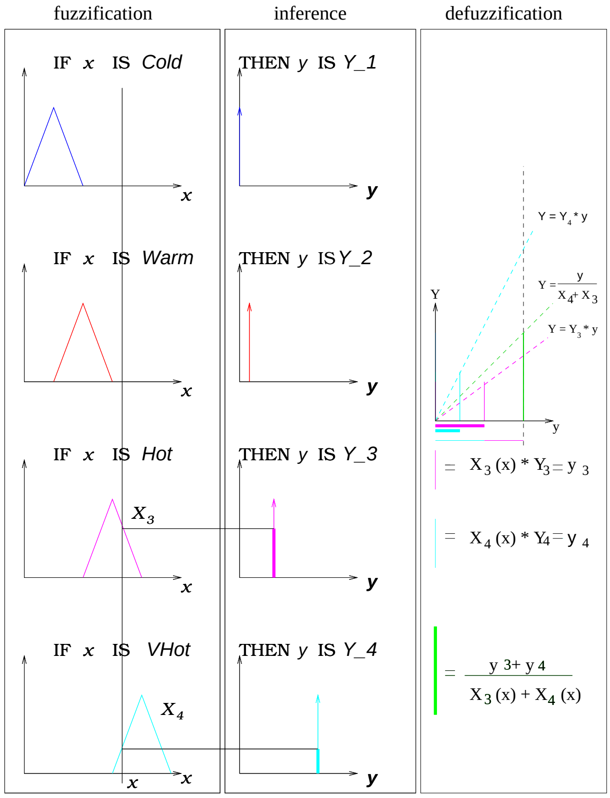


Fig. 5. A SISO system with B-functions of order 2 ($X_i(x)$: firing strength of rule i; y_i: the contribution of rule i to the output).

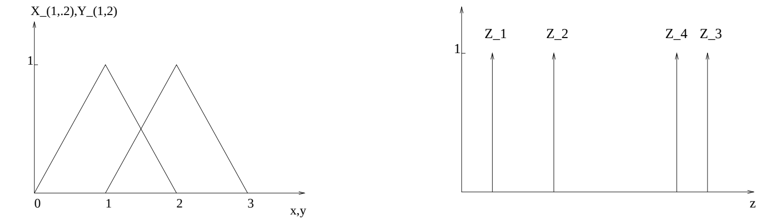
3.4 MISO Systems

We further illustrate the principle of the fuzzy controller by an example with two input variables (x and y) and an output z. The control vertices of the output are Z_1, Z_2, Z_3, Z_4. Fig. 6(a) and 6(b) depict the linguistic terms of the input and output.

The rule base consists of four rules:

Rule

1)	IF x	is	X_1	and y	is	Y_1	THEN z	is Z_1
2)	IF x	is	X_1	and y	is	Y_2	THEN z	is Z_2
3)	IF x	is	X_2	and y	is	Y_1	THEN z	is Z_3
4)	IF x	is	X_2	and y	is	Y_2	THEN z	is Z_4



(a) B-functions of order 2 covering the input x and y

(b) Singeltons defined on the output z

Fig. 6. Linguistic terms used in a Two-Input-One-Output system

Fig. 7 demonstrates the inference and Fig. 8 the defuzzification procedure by illustrating the computation of the crisp output z.

3.5 Rule Weighting

Some fuzzy control systems allow using a weight for each rule. That provides one more parameter to each rule, surely more flexibility for shaping control surface in a Mamdani type controller, but it results in more work for fine-tuning.

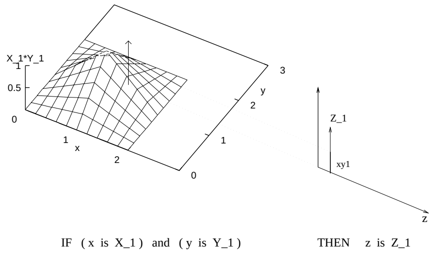
In fact, if we add one more rule weight to each rule in a B-spline fuzzy controller, then this controller corresponds a NURBS (*Non-Uniform Rational B-Spline*) model:

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

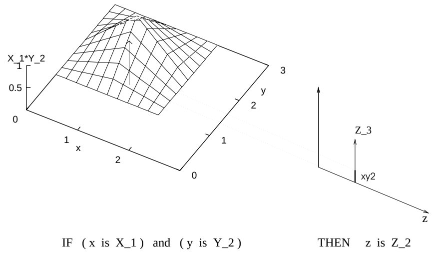
Experience of using B-splines in CAD shows that by using sufficient B-spline basis functions, NUBS of non-rational form may approximate any shape to a given precision, [2]. NURBS curves and surfaces are mainly used for exactly modelling special analytical functions like a circle, square, etc. The control 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 non-rational form for constructing fuzzy controllers.

3.6 Acceleration of Rule Evaluation

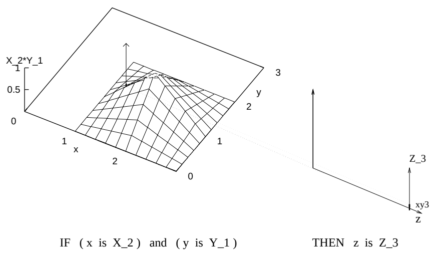
The index coding of the B-functions makes the evaluation of fuzzy rules highly efficient. For an input $x \in (x_i, x_{i+1})$, it is known that exact k (the order of B-functions) linguistic terms will be activated, i.e. k B-functions $X_{i,k}(x)$, $X_{i-1,k}(x)$, $\ldots$, $X_{i-k+1,k}(x) > 0$. All the other linguistic terms are unactivated. In the whole rule base with n inputs, exact k^n rules are firing for any input vector in the universe of discourse.



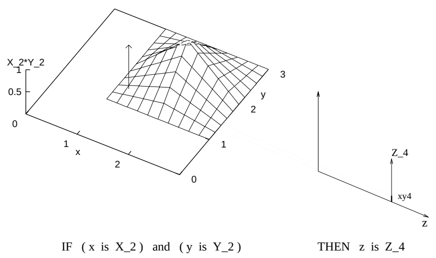
(a) Rule 1



(b) Rule 2



(c) Rule 3



(d) Rule 4

Fig. 7. Inference with a B-spline fuzzy controller with two inputs

4 Adaptation and Learning

4.1 Adaptation of Peak Support Points

We developed an algorithm for adapting the *Psupp*-points. This algorithm is a modified algorithm for self-organising neural networks. On the left and right neighbour of a *Psupp*-point $\mathbf{x}$, two new *Psupp*-points $\mathbf{x_l}$ and $\mathbf{x_r}$ are selected and the output values of the controller, noted as y_l and y_r, for these two points are computed. If the desired training data y^d is greater or smaller with a threshold $\theta \geq 0$ than both y_l and y_r, a modification of $\mathbf{x}$ is necessary.

Assume $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ *are the Psupp-points of the input variables and* $\mathbf{X_{1,k}}, \mathbf{X_{2,k}}, \ldots, \mathbf{X_{m,k}}$ *are their corresponding linguistic terms (B-functions of order* k*).*

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

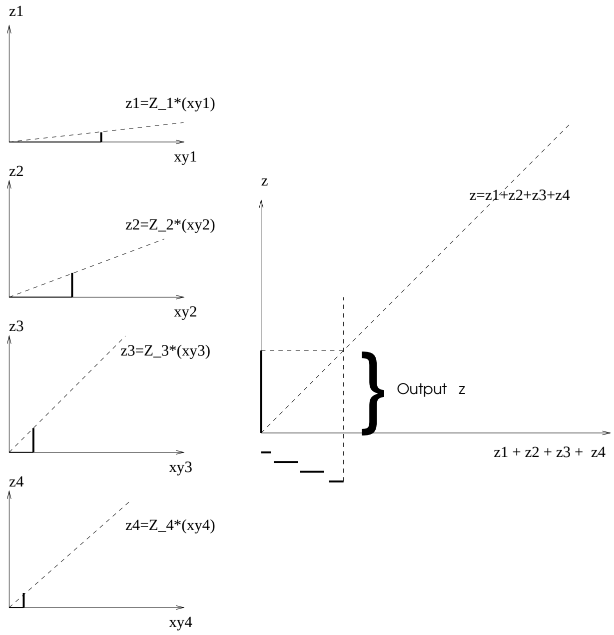


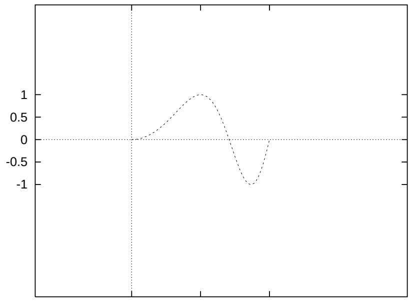
Fig. 8. Defuzzification by blending the control vertices with B-function values

2. *Select two neighbouring Psupp-points* $\mathbf{x_l}, \mathbf{x_r}$*, 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* $y_l \leq y_r$
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 (see section 4.2).*
5. *Continue with 1.*

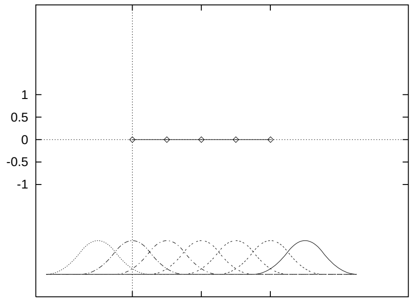
As an example we show the approximation of a function $\sin(\mathbf{2\pi x^2})$. The function has a minimum at $x = 0.86$ and a maximum at $x = 0.5$. At the beginning five *Psupp*-points are evenly distributed within the interval $[0, 1]$. All the five control vertices are set to 0. The training data 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 *Psupp*-points.

4.2 Supervised Learning of Control Vertices

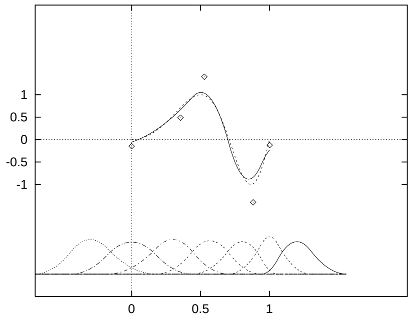
Assume $\{(\mathbf{X}, y_d)\}$ is a set of training data, where



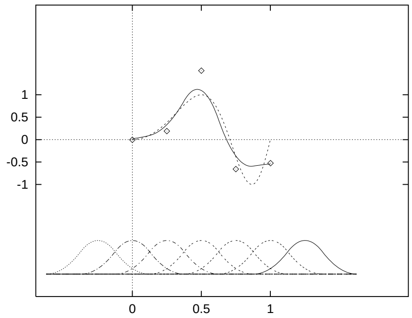
(a) The function $\sin(2\pi x^2)$



(b) The initial controller output



(c) After 2000 learning steps with *Psupp*-point adaptation



(d) After 2000 learning steps without *Psupp*-point adaptation

Fig. 9. Approximation of the function $y = \sin(2\pi x^2)$. The initial uniform B-functions are adapted to non-uniform ones.

- $\mathbf{X} = (x_1, x_2, \ldots, x_s)$: the input data vector,
- y_d : the desired output for $\mathbf{X}$.

The squared error is computed as:

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

where y_r is the current real output value during training.

The parameters to be found are $Y_{i_1, i_2, \ldots, i_n}$, which make the error in (4) as small as possible, i.e.

$$E = \frac{1}{2}(y_r - y_d)^2 \equiv \text{ MIN}. \tag{5}$$

Each control vertex $Y_{i_1,\ldots,i_n}$ can be modified by using the gradient descent method:

$$\Delta Y_{i_1,\ldots,i_n} = -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} \tag{6}$$

$$= \epsilon(y_r - y_d) \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) \tag{7}$$

where $0 < \epsilon \leq 1$.

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 always positive:

$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\ldots,i_n}} = \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) \geq 0. \tag{8}$$

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 (global) minimum.

4.3 Implemented Examples

The following examples were implemented (for details of these functions see [3,6]):

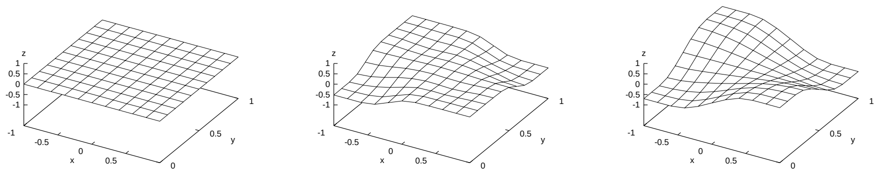
- Approximation of very non-linear functions with one, two and three variables (see Fig. 10 and 11 for an example of approximating a function with two variables).
- Parameter identification and time series prediction.
- Supervised learning of "Truck Backer-Upper" (see Fig. 12 for the learned control surfaces and Fig. 13 for two test examples) and "Inverse Kinematics".

The results show that B-spline fuzzy systems can achieve the same and sometimes better modelling effect as done by the ANFIS (Adaptive-Network-Based Fuzzy Inference System) [3] and "Additive Systems" [6] methods, but B-spline fuzzy systems converge faster in all these cases.

4.4 Unsupervised Learning

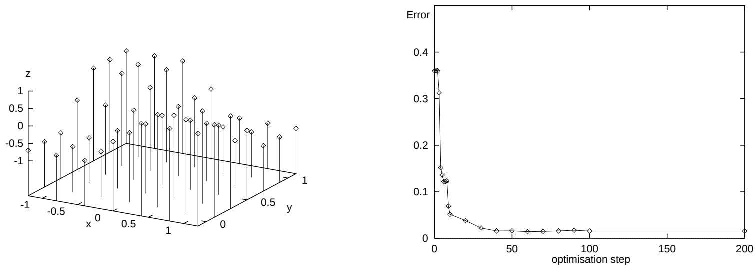
In unsupervised learning, it is usually possible to define an "evaluation function" if the desired data of the output are unknown. Such an evaluation function should describe how "good" the current system state $((x_1, x_2, \ldots, x_n), y)$ is. For each input vector, an output is generated. With this output, the system transits to another state. The new state is compared with the old one; an adaptation is performed if necessary.

Assume the evaluation function, denoted by $F(\cdot)$, possesses a bigger value for a better state, i.e. for two states A and B, if A is better than B, then



(a) step 0 (b) step 5 (c) step 100

Fig. 10. The control surfaces during the optimisation for approximating the function $z = sin(2\pi x) \cdot cos(\pi y)$, with $-1 \leq x \leq 1$ and $0 \leq y \leq 1$. The membership functions defining the real and virtual linguistic terms of x and y are 7 B-functions of order 3 for both x and y.



(a) Automatically generated control vertices Z_{ij}

(b) Convergence of the approximation error

Fig. 11. Learning results after 100 optimisation steps of the problem shown in Fig. 10.

$F(A) \geq F(B)$. The adaptation of the control vertices can be performed with a similar representation as in supervised learning. Assume that the desired state is A_d. The change of control vertices can be written as:

$$\Delta Y_{i_1,\ldots,i_n} = S \ . \ \epsilon \ . \ |F(B) - F(A_d)| \ . \ \prod_{j=1}^{n} X_{i_j,k_j}(x_j). \tag{9}$$

where

$$S = sign(F(A) - F(B)) * sign(F(B) - F(A_d)) * sign(y) \tag{10}$$

represents the direction to modify the control vertex.

By appropriately defining certain cost functions, we have successfully applied the B-spline fuzzy systems to the following control problem:

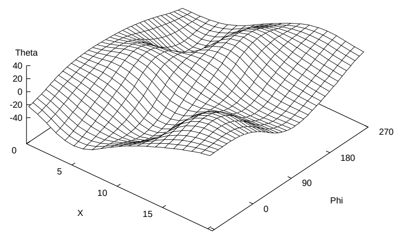


Fig. 12. Solution of the "truck backer-upper" problem: output θ after learning (θ was initialised as zero for all inputs). Both inputs are covered with 5 B-functions of order 3. Altogether 14 training trajectories are prepared.

– "Cart-pole-Balancing", Fig. 14;
– Collision-avoidance and contour-tracking of a real mobile robot, Fig. 15(a);
– Sensor-based screwing with two robot arms, Fig. 15(b).

5 Characteristics

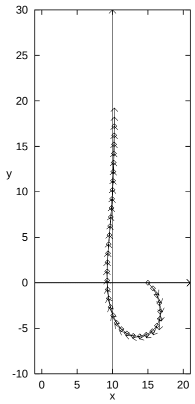
Computing parameters of such a B-spline fuzzy system is divided into two steps: for the IF-part and for the THEN-part.

Characteristics of modelling the IF-part are:

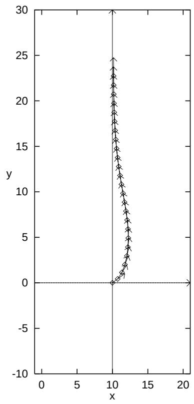
– Considering the granularity of the input space and the extrema distribution of the control space if known, the fuzzy sets can be generated using the recursive computation of B-spline basis functions. This approach provides an natural, automatic approach to generate the information granularity proposed by Zadeh [8].
– These fuzzy sets can be further adapted during the generation of the whole system.

Characteristics for generating the THEN-part are:

– Fuzzy singletons can be initialised with the values acquired from expert knowledge. These approximately determined parameters will be fine-tuned with learning algorithms.
– For supervised learning, the differentials of the Square-Error with respect to control vertices are convex functions. Therefore, rapid convergence for supervised learning is guaranteed, and a reinforced learning process can also converge quite well if the change of cost function is piecewise approximately linear with the change of control vertices.



(a) Starting position: $x = 15,\phi = -30$.



(b) Starting position: $x = 10,\phi = 30$.

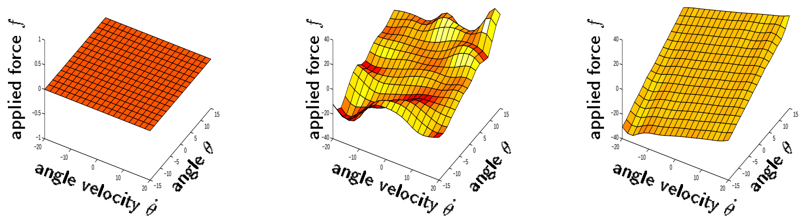
Fig. 13. Two motion trajectories produced by the trained fuzzy controller of the "truck backer-upper" problem.

– The control space changes locally while the control vertices are modified. Based on this feature, the control vertices can be optimised gradually, area-by-area. This is especially important for a high-dimensional control space with a large amount of parameters.

6 Discussions

The curse of dimensionality. Like all other types of fuzzy controllers, B-spline type controllers cannot avoid this problem, i.e. the number of rules increases exponentially with the number of inputs. However, as shown in section 3.6, the evaluation time of the whole rule base can be reduced form m^n to k^n, where m is the number of linguistic terms for input, k the order of B-functions and k is usually selected as 2 or 3 for most applications. The adaptation of knots also contributes to the efficient utilisation of linguistic terms. Additionally, the learning process of a MISO system does not suffer from the problem of local minima even for a high-dimensional control space thanks to the local influence of control vertices.

Conversion back to classic type. For some applications, e.g. data mining or qualitative analysis, we may find the large number of fuzzy singletons too numerical and want to approximately transform them into a small 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, which represents a fuzzy partition of the output variable, Fig. 16.



(a) at the beginning

(b) after 100 learning steps

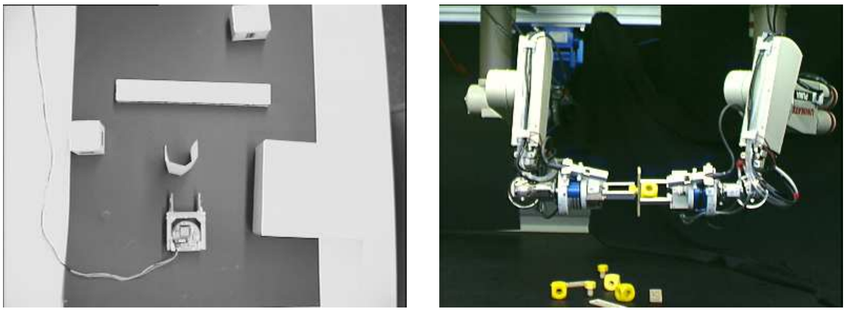
(c) after 3000 learning steps

Fig. 14. Applied force f to the cart-pole system

In this way, our approach can optimally combine the numerical interpolation with linguistic interpretation, it is thus very promising for a wide range of applications in adaptive modelling and control.

References

1. H. Bersini and G. Bontempi. Now comes the time to defuzzify neuro-fuzzy models. In *Proceedings of the FLINS Workshop on Intelligent Systems and Soft Computing for Nuclear Science and Industry*, pages 130–139, 1996.
2. 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.
3. J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man and Cybernetics*, 23(3):665–685, 1993.
4. 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.
5. E. H. Mamdani. Twenty years of fuzzy control: Experiences gained and lessons learned. *IEEE International Conference on Fuzzy Systems*, pages 339–344, 1993.
6. S. Mitaim and B. Kosko. What is the best shape of a fuzzy set in function approximation. In *IEEE International Conference on Fuzzy Systems*, 1996.
7. L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, 1994.
8. L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Trans. on Fuzzy Systems*, 4(2):103–111, 1996.
9. J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models. In *IEEE International Conference on Fuzzy Systems*, 1996.
10. J. Zhang, K. V. Lee, and A. Knoll. Unsupervised learning of control spaces based on b-spline models. Submitted to IEEE International Conference on Fuzzy Systems, 1997.
11. J. Zhang, Y. v. Collani, and A. Knoll. On-line learning of b-spline fuzzy controller to acquire sensor-based assembly skills. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.



(a) Motion control of a mobile robot. The robot learns to avoid obstacles and to track an unknow contour. For details see [10].

(b) Two cooperating manipulators for fixture-less assembly. Compensation motions are determined by a self-learning B-spline controller with force/moment sensor readings, [11].

Fig. 15. Applications of unsupervised learning in robot systems.

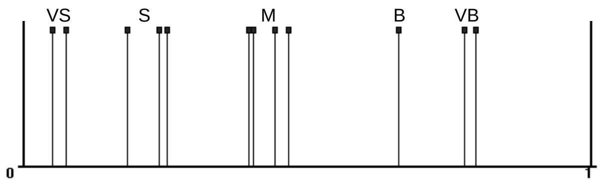


Fig. 16. All fuzzy singletons can be clustered into linguistic terms.