in "International Journal of Intelligent Systems", Vol. 13, Number 2/3, February/March, 1998, pp 257-286.

Constructing Fuzzy Controllers with B-Spline Models – Principles and Applications

Jianwei Zhang and Alois Knoll
Faculty of Technology, University of Bielefeld,
33501 Bielefeld, Germany

Abstract

In this paper we present an approach to designing a novel type of fuzzy controller. B-spline basis functions are used for input variables and fuzzy singletons for output variables to specify linguistic terms. "Product" is chosen as the fuzzy conjunction, and "centroid" as the defuzzification method. By appropriately designing the rule base, a fuzzy controller can be interpreted as a B-spline interpolator. Such a fuzzy controller may learn to approximate any known data sequences and to minimise a certain cost function. By choosing such a function appropriately, the learning process can be made to converge rapidly. We applied this approach to the problems of function approximation and both supervised and unsupervised learning of mobile robots. Experiments validate the advantages of this approach.

1 Introduction

Recently, fuzzy logic control (FLC) has been successfully applied to a wide range of control problems and has demonstrated some advantages, e.g., in efficiency of developing control software, appropriate processing of imprecise sensor data, and real-time requirements [10, 13]. However, as pointed out in [3], one obstacle to wide acceptance in industrial applications is that "it is still not clear how membership functions, defuzzification procedures, domain discretization, and normalization coefficients contribute, either in combination or as stand-alone factors, to the performance of the FLC." Two important related issues are:

Quality of fuzzy controllers. In practical applications, the smoothness of the controller output is one of the most important design requirements. Generally, the smoothness can be measured by how many times an output variable can be differentiated with respect to the input variables. The smoothness criterion is applied to the control of very complex systems, such as the speed control of automated trains, as well as to simple actuators like electrical motors, whose life expectancy depends directly on the smoothness of the controller output. Unfortunately, in general cases, a high degree of smoothness cannot be guaranteed and is frequently hard to determine for a given controller.

Guidelines for choosing membership functions. Up to now, there exist no convincing guidelines for the successful design of fuzzy controllers. This pertains in particular to the choice of a concrete membership function. In various fuzzy control applications, membership functions of triangular or trapezoidal shape are utilised because of the simplicity of specification and the satisfying results. But the question still remains: Can the control performance be improved by choosing a certain set of membership functions?

These two issues can be addressed by comparing B-spline models with a standard fuzzy logic controller. In our previous work [14], we compared splines and a fuzzy controller with *single-input–single-output* (SISO) structures. In this paper, the *multi-input–single-output* (MISO)[1] controller is considered. Periodical nonuniform B-spline basis functions are interpreted as membership functions (MFs). Furthermore, aspects of function approximation and learning control of mobile robots are discussed.

2 Some Previous Work

2.1 Advances in Fuzzy Control

Several authors have shown that fuzzy controllers are universal approximators. Wang [11] presents a universal approximator by using Gaussian membership functions, product fuzzy conjunction and "centroid"[2] defuzzification. Buckley [2] has shown that input-output fuzzy controllers are universal approximators. Kosko and Dickerson [6] proved that an additive fuzzy system uniformly approximates $f : X \to Y$ if X is compact and f is continuous.

Two successful applications in commercial controller and process control are given in [13], one is the OMRON temperature controller, the other is a gas-fired water heater. The membership functions are selected as triangles and each pair overlaps. Can these be generalised as design rules? The work in [7] shows that triangular membership functions with a 1/2 overlap level produce a reconstruction error of zero. Further questions are: Are there other forms of suitable membership functions? Should the overlap of the fuzzy sets for linguistic terms meet certain constraints?

2.2 Popularity of B-Splines

To solve the problem of numerical approximation for smoothing statistical data, "basis splines" (B-Splines) were introduced by Schoenberg [9]. B-splines were used later by Riesenfeld [8] and Gordon [4] in *Computer Aided Geometric Design* (CAGD) for curve and surface representation. Because of their versatility based on only low-order polynomials and their straightforward computation, B-splines have become more and more popular. Nowadays, B-spline techniques represent one of the most important trends in CAD/CAM; they have been extensively applied in modelling free shape curves and surfaces. Recently, splines have also been proposed for neural network modelling and control [1, 12].

Although fuzzy techniques lend themselves to on-line control and B-splines have been used mainly in off-line modelling, some interesting common points can still be found. In our previous paper [14] we pointed out that B-spline basis functions and parametric membership functions of a linguistic variable are both convex, overlapping set functions. Splines and fuzzy controllers possess good interpolation features. The synthesis of a smooth curve with spline functions can easily be associated with the defuzzification process. These points are the main motivation for our work on utilising B-splines to design fuzzy controllers.

3 Construction Principles

We consider the membership functions that are used in the context of specifying linguistic terms ("values" or "labels") of input variables of a fuzzy controller. In the following, basis functions of periodical *Nonuniform B-Splines* (NUBS) are summarised and compared with a fuzzy controller. We also use *B-functions* for the NUBS basis functions.

[1]A multi-input–multi-output (MIMO) rule base is normally divided into several MISO rule bases.

[2]Synonyms: Takagi-Sugeno IDM (inference and defuzzification method), Tsukamoto-method, "weighted-mean."

3.1 B-Spline Basis Functions Defined on a Single Variable

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) $N_{i,k}$ of order k is defined as

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

with $i = 0, 1, \ldots, m - k$.

The important properties of B-functions are

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

3.2 Membership Functions of B-Function Type

The B-functions are employed to specify the linguistic terms, and knots are chosen to be different from each other (periodical model). Visually, the selection of k (the order of the B-functions) determines the following factors of the fuzzy sets for modelling the linguistic terms (Table 1). In this table, the *width* of a fuzzy set is measured by the number of knot intervals and the *overlap* degree by how many fuzzy sets are defined on each knot interval.

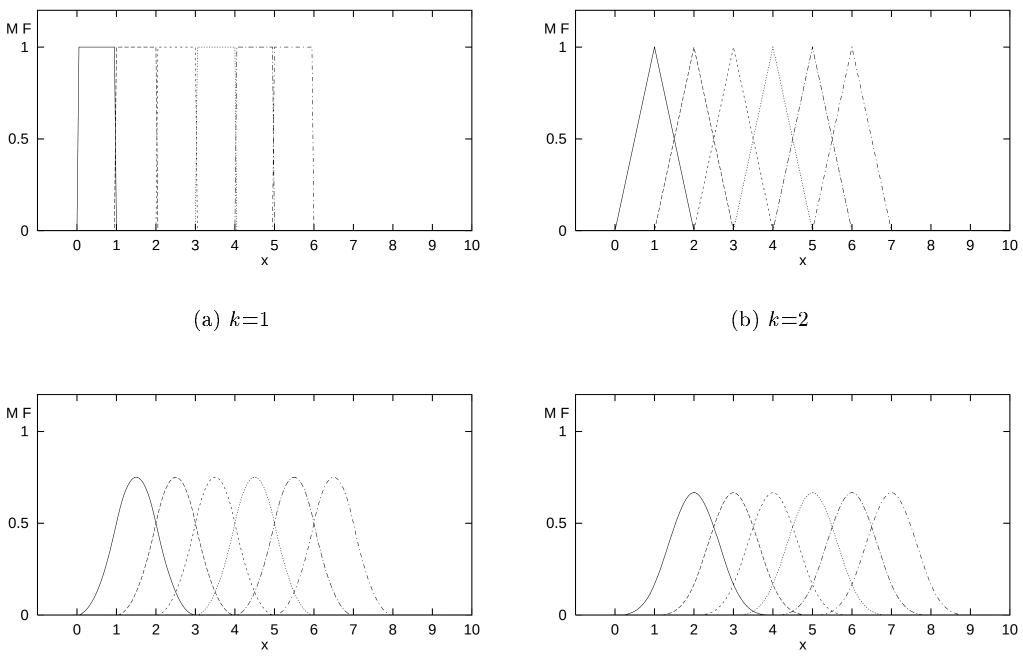
Order k	1	2	3	4	...
Degree	0	1	2	3	...
Shape	Rectangular Fig. 1(a)	Triangular Fig. 1(b)	Quadratic Fig. 1(c)	Cubic Fig. 1(d)	... ...
Width	1	2	3	4	...
Overlap	1	2	3	4	...

Table 1: The visual effect of fuzzy sets depends mainly on the order of the B-functions.

3.3 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*. To maintain the "partition of unity" for all $x \in [x_0, x_m]$, 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 illustrated in Figure 2:

- In the case of order 2, no marginal B-function is needed (Fig. 2(a)).



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

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

Figure 1: Fuzzy sets defined by B-spline basis functions of different orders.

- In the 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.

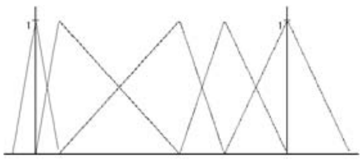
3.4 Core and Marginal Rules

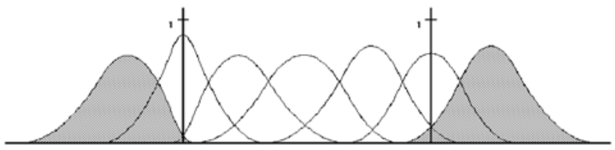
We define the *core rules* as linguistic rules that use real linguistic terms. If virtual linguistic terms appear in the premise, to maintain the output continuity at both ends of the universe of x, additional rules are needed to describe the control action for these cases. Since these rules use the virtual linguistic terms that 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 premise (Fig. 3).

3.5 A B-Spline Interpolator

Since a MIMO rule base is normally divided into several MISO rule bases, we consider only the MISO case. Under the following conditions:

- Periodical B-spline basis functions as membership functions for inputs.


(a) Order 2


(b) Order 3

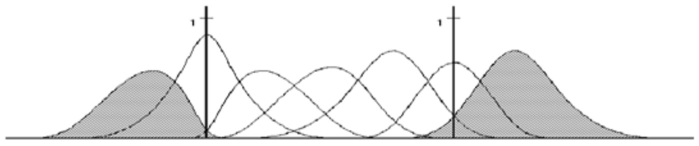

(c) Order 4

Figure 2: Non-uniform B-functions of different orders defined for real and virtual linguistic terms by the same knot vector (virtual linguistic terms: shaded).

- 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.
- 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 $x_1, x_2, \ldots, x_n$, rules with the n conjunctive terms in the premise are given in the following form:

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

where

- x_j: the jth input ($j = 1, \ldots, n$).
- k_j: the order of the B-spline basis functions used for x_j.
- $N^j_{i_j,k_j}$: the ith linguistic term of x_j defined by B-spline basis functions.
- $i_j = 0, \ldots, m_j$, representing how fine the jth input is fuzzy partitioned.

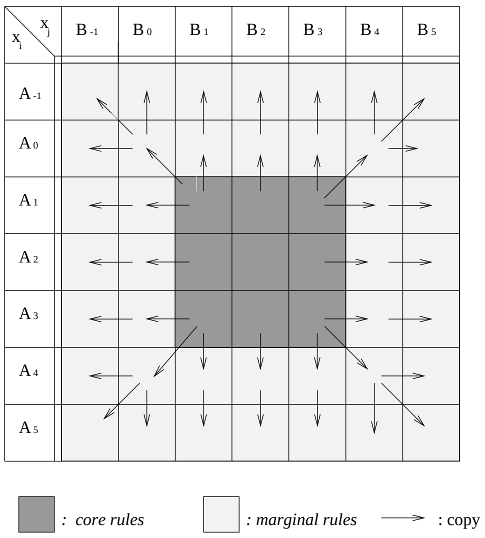


Figure 3: The outputs of the marginal rules are copied from that of the neighbouring core rules.

- $Y_{i_1 i_2 \ldots i_n}$: the control vertex (deBoor points) of $Rule(i_1, i_2, \ldots, i_n)$.

Then the output y of a MISO fuzzy controller is

$$y = \frac{\sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j))}{\sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j)} \quad (1)$$

$$= \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j)) \quad (2)$$

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 variables $x_j, j = 1, \ldots, n$.
- 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.

If the order of the B-functions and the number of linguistic terms used in the premise are chosen, 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.

3.6 Example of a SISO System

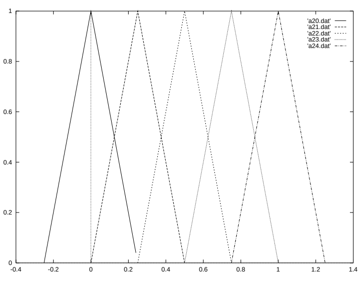
Assume that the input variable is x and y is the output variable. The input x is covered by the B-functions $N_{i,k}$. For specifying linguistic terms of y, the simple fuzzy singletons Y_i are used.

The *Core Rule Set* (CRS) can be described as

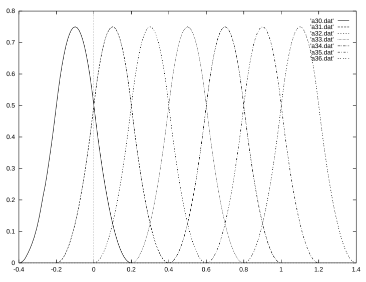
$$CRS = \{Rule(i) : \text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } Y_i \mid \quad i = 0, \ldots, m\}$$

If virtual linguistic terms appear in the premise, to maintain the output continuity at both ends of the real universe of discourse x, the *Marginal Rule Set* (MRS), which contains the left (right) virtual linguistic terms may use the repeated output values Y_0 and Y_m. The whole rule set RS is then $RS = CRS \cup MRS$.

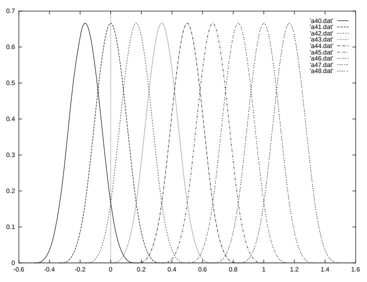
We consider a core rule set of five rules: $CRS = \{\text{IF } \ x \text{ is } A_i \text{ THEN } y \text{ is } Y_i, i = 1, \ldots, 5\}$, where the universe of discourse of x is defined on $[0, 1]$ and Y_1 to Y_5 are fuzzy singletons with the following values: 0.5, 1.0, 0.3, 0.55, 0.2.



(a) order 2



(b) order 3

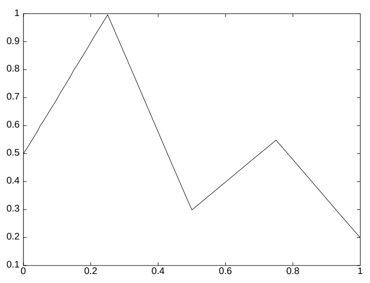


(c) order 4

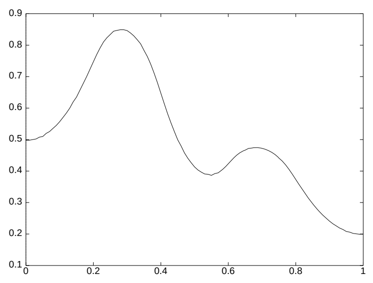
Figure 4: Membership functions used for the input variable x.

The linguistic terms used for input x are shown in Figure 4(a)-(c). For B-functions of order 3 and 4, one virtual linguistic term is added at the left and one at the right end. If the two virtual linguistic terms for the case of order 3 or 4 are denoted A_0 and A_6, two marginal rules can be constructed by copying the output values Y_1 and Y_5: $MRS = \{\text{IF } \ x \text{ is } A_0 \text{ THEN } \ y \text{ is } \ Y_1; \text{IF } \ x \text{ is } A_6 \text{ THEN } \ y \text{ is } \ Y_5\}$.

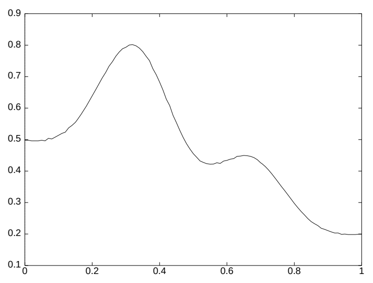
The trajectories of the output with respect to input are depicted in Figure 5(a)–(c). They are (a) C^0-continuous; (b) C^1-continuous; (c) C^2-continuous.



(a) order 2



(b) order 3



(c) order 4

Figure 5: Trajectory of the output y with respect to the input x.

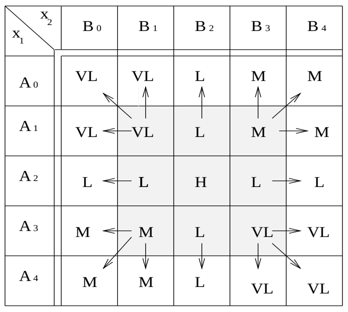
3.7 Example of a MISO System

Two input variables x_1 and x_2 are covered by three real linguistic terms, represented by $\{A_1, A_2, A_3\}$ and $\{B_1, B_2, B_3\}$, which denote “low,” “middle,” and “high,” respectively. A core rule set consisting of nine rules is shown in Figure 6(a). For the output variable y, fuzzy singletons are defined to represent "VL" (very low), "L" (low), "M" (middle), and "H" (high).

A_1, A_2, A_3 and B_1, B_2, B_3 are defined with adjacent uniform B-functions of order 2, 3, and 4, similar to Figure 4 (a)-(c). If B-functions of order 3 or 4 are used, one virtual linguistic term A_0 (B_0) is added left-adjacent to A_1 (B_1), another A_4 (B_4) is added right-adjacent to A_3 (B_3). Marginal rules that have one of the terms A_0, A_4, B_0 and B_4 in the premise are assigned the output singletons of the nearest core rule (Fig. 6 (b)).

$x_1 \backslash x_2$	B_1	B_2	B_3
A_1	VL	L	M
A_2	L	H	L
A_3	M	L	VL

(a) Core rule set for B-functions of order 2

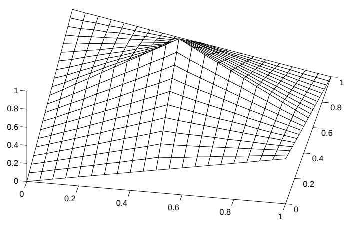


$x_1 \backslash x_2$	B_0	B_1	B_2	B_3	B_4
A_0	VL	VL	L	M	M
A_1	VL	VL	L	M	M
A_2	L	L	H	L	L
A_3	M	M	L	VL	VL
A_4	M	M	L	VL	VL

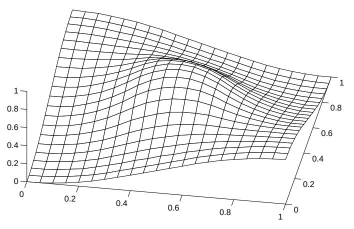
(b) Core and marginal rule set for order 3 or 4

Figure 6: Rule bases for MFs of different orders.

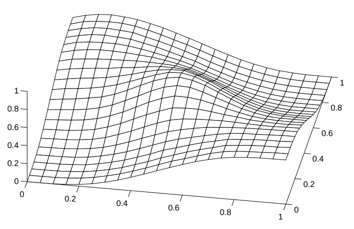
The control surfaces describing the relation of y with x_1 and x_2 are shown in Figure 7(a), (b), (c). The differentiability of the three cases is (a) y is continuous; (b) $\partial y/\partial x_1$ and $\partial y/\partial x_2$ are continuous; (c) $\partial^2 y/\partial x_1^2$ and $\partial^2 y/\partial x_2^2$ are continuous.



(a) Using order 2



(b) Using order 3



(c) Using order 4

Figure 7: The control surfaces of an example with 2 inputs (x_1, x_2) / 1 output (y).

4 Application A: Supervised Learning

Supervised learning assumes that a "teacher" provides the complete desired system output for each input datum. Based on the complete set of these input/output vectors, B-spline type fuzzy controllers can be trained very rapidly. Computing parameters of such a B-spline fuzzy system is divided into two steps: for the IF-part and for the THEN-part. Considering the granularity of the input space and the maximum point distribution of the control space if known, the fuzzy sets can be generated by using the recursive computation of B-spline basis functions. We developed an algorithm for adapting the knots of the IF-part, which is a modified algorithm for self-organising neural networks. However, if sufficient B-functions are used for the inputs, the local modification of the knots has only negligible influence on the control surface. Therefore, in the following, we concentrate on the control vertices of the THEN-part, which can be automatically achieved through a learning procedure.

4.1 Learning Algorithm

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

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

The squared error is computed as

$$E = \frac{1}{2}(y_r - y_d)^2 \tag{3}$$

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 (3) as small as possible, i.e.,

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

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

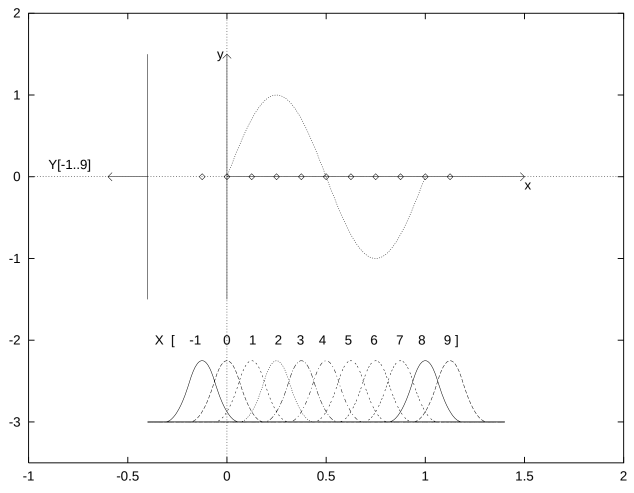
$$\begin{aligned} \Delta Y_{i_1,\ldots,i_n} &= -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} && (5) \\ &= -\epsilon (y_r - y_d) \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j) && (6) \end{aligned}$$

where ϵ is the learning rate, $0 < \epsilon \leq 1$.

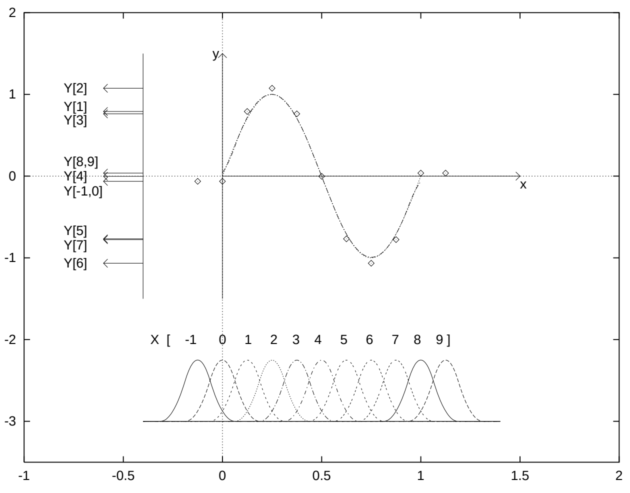
The gradient descent method guarantees that the learning algorithm converges to the global minimum of the error function because the second partial differentiation with respect to $Y_{i_1,i_2,\ldots,i_n}$ is constant:

$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\ldots,i_n}} = \epsilon (\prod_{j=1}^{n} N^j_{i_j,k_j}(x_j))^2 \ \geq 0 \tag{7}$$

This means that the error function (3) is convex in the space $Y_{i_1,i_2,\ldots,i_n}$ and therefore possesses only one (global) minimum.



(a) Before optimisation



(b) After optimisation

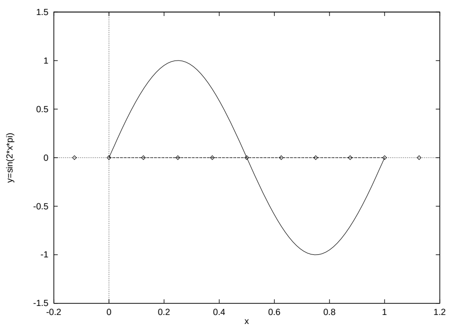
Figure 8: Mapping the input x to the output y with B-functions as MFs.

4.2 Function Approximation

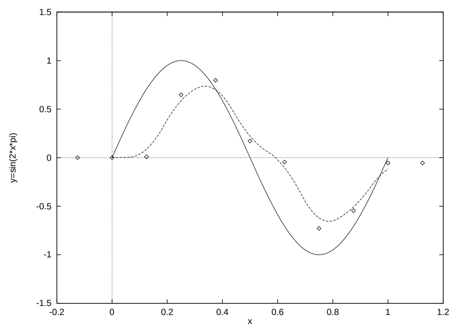
4.2.1 A One-Input–One-Output Controller

A function $y = sin(2\pi x)$ is to be approximated with a fuzzy controller. Figure 8 depicts the mapping of the input x to the output y, where x is covered with B-functions of order 3 and fuzzy singletons are defined on y. The initial positions of the fuzzy singletons are arbitrarily chosen (e.g., as zero, see Fig. 8(a)). The output curve and the fuzzy singletons after the self-optimisation process are illustrated in Figure 8(b).

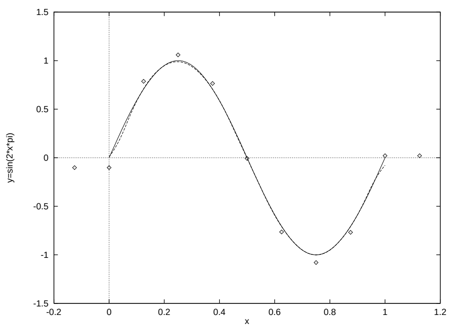
Figure 9 show several intermediate steps during the optimisation. The RMSE (*root squared mean error*) curves for the approximation are shown in Figure 10.



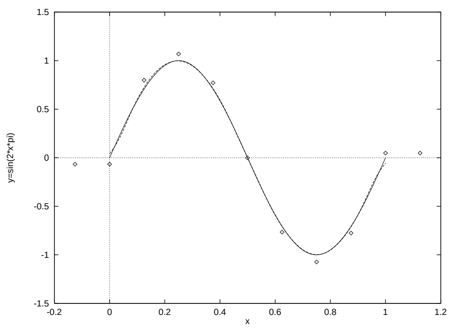
(a) step 0



(b) step 10



(c) step 200



(d) step 500

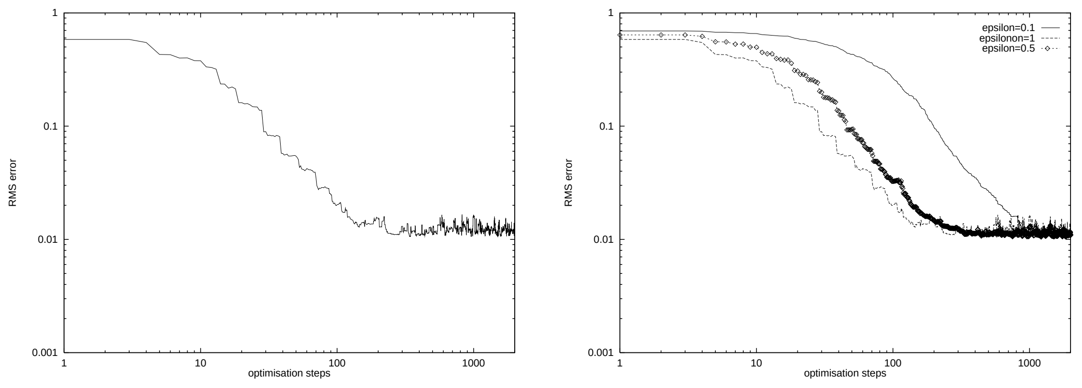
Figure 9: Optimisation of positions of the fuzzy singletons defined on the output ($\epsilon = 1$).

4.2.2 An Example with Two Input Variables

A 2D example is implemented to approximate the function $z = sin(2\pi x) \cdot cos(\pi y)$, where $-1 \leq x \leq 1$ and $0 \leq y \leq 1$. Figure 11(a) and (b) shows the membership functions defining the real and virtual linguistic terms of x and y. The control surfaces in several intermediate steps of the optimisation can be seen in Figure 12. Figure 13 illustrates the automatically generated control vertices. Figure 14 depicts the RSME curve.

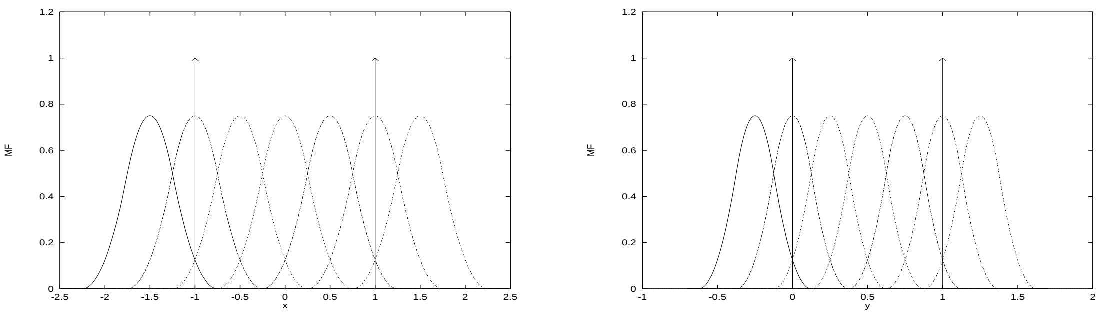
4.2.3 Examples of ANFIS

The following examples were implemented (for details of these functions see [5]):



(a) $\epsilon = 1$ (b) Different values of ϵ.

Figure 10: RSME with respect to the optimisation steps.



(a) variable x (b) variable y

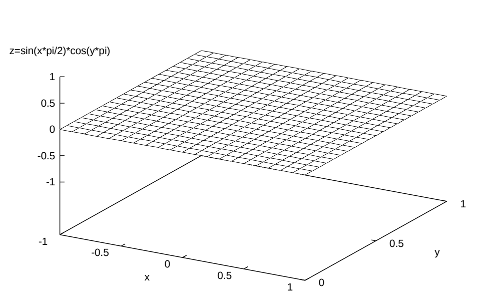
Figure 11: B-functions as MFs for the input variables.

- $z = \frac{\sin(x)}{x} * \frac{\sin(y)}{y}$.
- $f(x, y, z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$.
- $x(t+1) = \frac{0.2x(t-r)}{1+x^{10}(t-r)} + 0.9x(t)$.
- Identifying 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.

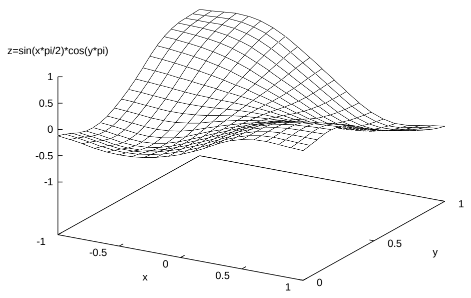
The implementations show that the B-spline fuzzy controllers can approximate all these functions to a certain precision if the order of B-functions is suitably chosen and the partition of the input space is fine enough. Generally, our approach needs less computation time than ANFIS.

4.3 Supervised Learning in Robotics

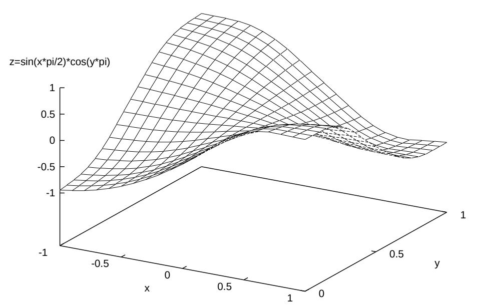
We successfully applied this approach to supervised learning to the "truck backer-upper" (see [11]) and "inverse kinematics" problem in robotics. Figure 15 to Figure 17 show an example of the automatically learned control space by achieving a "truck backer-upper" solution.



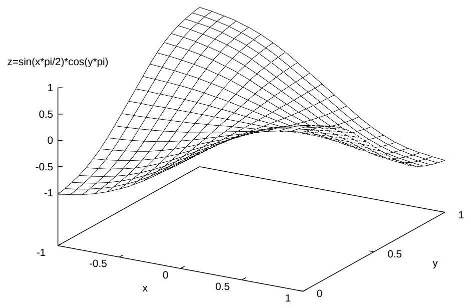
(a) step 0



(b) step 50



(c) step 200



(d) step 1000

Figure 12: The control surfaces during the optimisation.

The truck starts at an arbitrary position ((x, y) and ϕ) and should be steered backward so as to reach the goal position ($x = 10, y$) and $\phi = 90^\circ$ (Fig. 15). Therefore, the fuzzy controller has two inputs, $0 \leq x \leq 20$ and $-90^\circ \leq \phi \leq 270^\circ$, and one output, $-40^\circ \leq \theta \leq 40^\circ$.

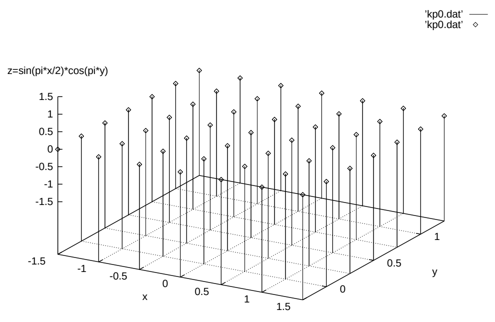
The kinematics of the truck are known:

$$\begin{aligned}
x(t+1) &= x(t) + \cos[\phi(t) + \theta(t)] + \sin[\phi(t)]\sin[\phi(t)] \\
y(t+1) &= y(t) + \sin[\phi(t) + \theta(t)] - \sin[\phi(t)]\cos[\phi(t)] \\
\phi(t+1) &= \phi(t) - \sin^{-1}\left[\frac{2\sin(\theta(t))}{b}\right]
\end{aligned}$$

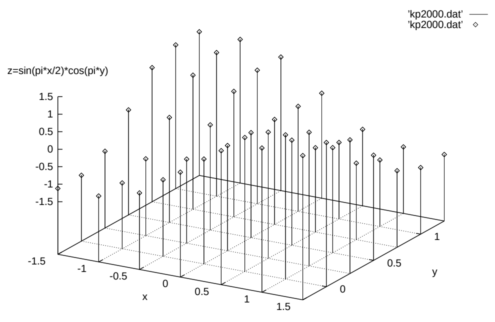
where b is the length of the truck.

In the following implementation, each input is covered with five B-functions of order three. Based on the above kinematics description, we can prepare 14 “training trajectories.” Each trajectory is a sequence of data vector $(x^i(t),\ y^i(t),\ \phi^i(t),\ \theta^i(t))$, where $t = 0, \ldots, t^i_{max}$, which supplies one control process to steer the truck from the start position $(x^i(0), y^i(0), \phi^i(0))$ to the goal position $(x^i(t^i_{max}) = 10, y^i(t^i_{max}), \phi^i(t^i_{max}) = 90)$.

We train the control vertices as follows:



(a) The intial values



(b) After 2000 learning steps

Figure 13: The control vertices Z_{ij}

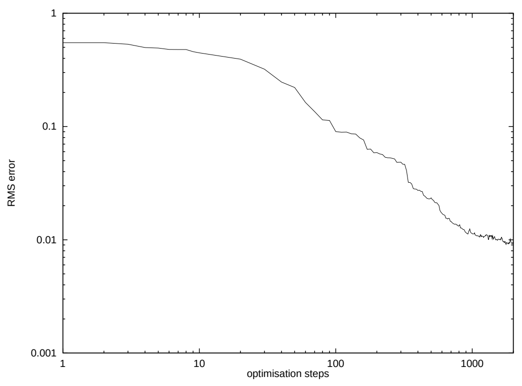


Figure 14: The RMSE curve for approximation of $z = sin(2\pi x) \cdot cos(\pi y)$.

```
For each trajectory i:
    For each position at $t = 1$ to $t=t^i_{max}$:
            Calculate $\theta(t)$ based on $(x^i(t), \phi^i(t))$;
            Modify the control vertices with the known value of $\theta^i(t)$.
```

Figure 16 shows the control surface of this control problem. Figure 17 depicts two backward trajectories from two different start posistions after learning.

5 Application B: Unsupervised Learning Control of a Mobile Robot

The proposed learning approach was also applied to a real mobile robot system *Khepera*. Our earlier work with *Khepera* was to use fuzzy control to integrate planning and control [15]. In the following, we show how the control vertices of a fuzzy controller can be generated automatically through learning.

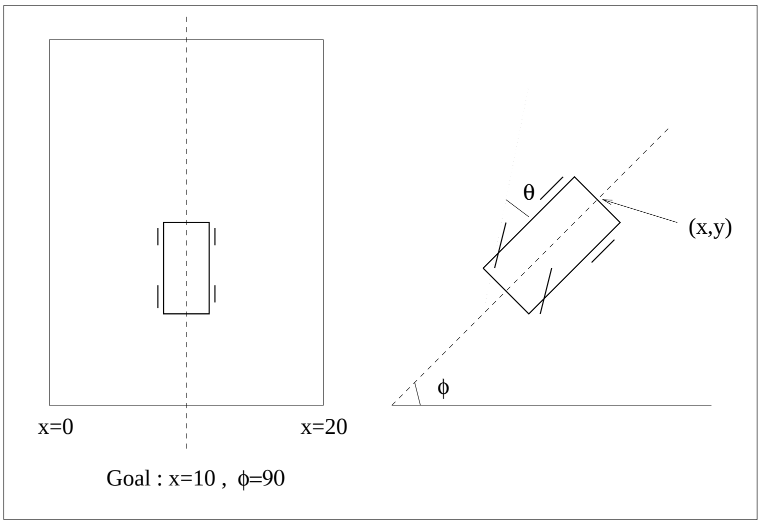


Figure 15: The start and goal position of the truck.

5.1 Sensors and Control Variables

The robot perceives its environment with six infrared sensors, which can be grouped into three variables, *SensL, SensF*, and *SensR*. The original sensor values are from 0 to 1023. To reduce the uncertainties, the range of the sensor values is scaled down to the interval $[0, 100]$, using 5 as the increment, i.e., the sensor values (controller inputs) are 0, 5, 10, ..., 90, 95, 100. All of these inputs are covered with seven B-functions as linguistic terms. Since the robot is equipped with only limited on-board computation capability, the order 2 of the basis functions was chosen so as to make real-time learning possible.

The two wheels of the robot are controlled independently by two motors. The output of the controller is the steering angle w. A positive value of w steers the robot to the right, and a negative value to steer it to the left. The velocities of the two wheels can then be computed as

- *For the left wheel:* $v + w$
- *For the right wheel:* $v - w$

where v is the current front velocity of the robot (Fig. 18).

5.2 Modifying Control Vertices

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 that the evaluation function, denoted by $F(\cdot)$, is monotonic, i.e., if state A is better than B, then $F(A) \geq F(B)$. The adaptation of the control vertices can be performed with a representation similar to that in supervised learning. Assume that the desired state is A_d. Analogous to formula (6) in Section 4.1, the change of control vertices can be written as

$$\Delta Y_{i_1,\ldots,i_n} = S \cdot \epsilon \cdot |F(B) - F(A_d)| \cdot \prod_{j=1}^{n} N_{i_j,k_j}(x_j) \tag{8}$$

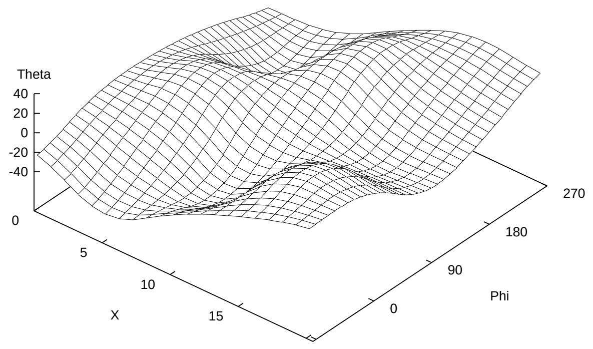


Figure 16: Solution of the "truck backer-upper" problem: output θ after learning (θ was initialised as zero for all inputs).

where

$$S = sign(F(A) - F(B)) \;\cdot\; sign(F(B) - F(A_d)) \;\cdot\; sign(y) \tag{9}$$

represents the direction to modify the control vertex.

5.3 Learning to Avoid Obstacles

A test environment for collision avoidance is shown in Figure 19. Before trying to develop the evaluation function, we first discuss the situations that the robot should deal with and the evaluation function:

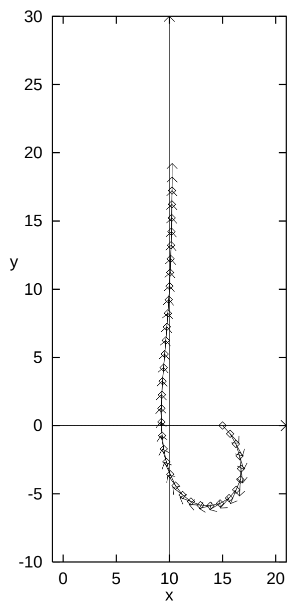
- $F(SensL, SensF, SensR) = -(SensL + SensF + SensR)$, if $SensF$ is large (the robot should then try to minimise the sum of all three sensor readings).
- $F(SensL, SensF, SensR) = -|SensL - SensR|$, if $SensF$ is small, $SensL$ or $SensR$ is not zero (the robot should try to keep the difference of $SensL$ and $SensR$ as small as possible).
- $F(SensL, SensF, SensR) = 0$ (no obstacles present).
- Otherwise (no reasonable evaluation function can be found): simply turn left.

This evaluation function supplies a large negative value if the robot has reacted the wrong way; it is positive if the robot steers correctly but with too small a steering angle. Figure 20 shows the learning results after 1000 learning steps.

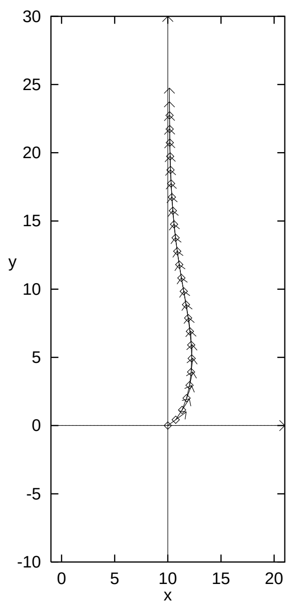
To demonstrate that the robot has learned correctly with the above approach, we perform the following experiment. We change the roles of the wheels, i.e., we let the wheels be controlled by

- *For the left wheel:* $v - w$,
- *For the right wheel:* $v + w$.

The results we get are the inverted ones of Figure 20 – exactly what we expected.



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



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

Figure 17: Two motion trajectories produced by the trained fuzzy controller.

5.4 Learning to Track Contours

In this mode of operation, the robot does not move away from an obstacle, but tries to "keep the obstacle in the right eye." The robot reacts only if it sees something in its "right eye" or in its "front eye." The evaluation function depends on the last and new sensor values as well:

- $F(SensF, SensR) = (SensF + |SensR - 80|)$, if the robot sees something in the front or on the right.
- Zero, otherwise.

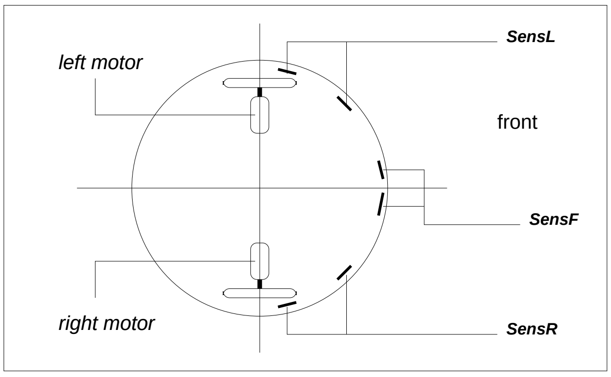


Figure 18: Sensors and motors of the Khepera robot.

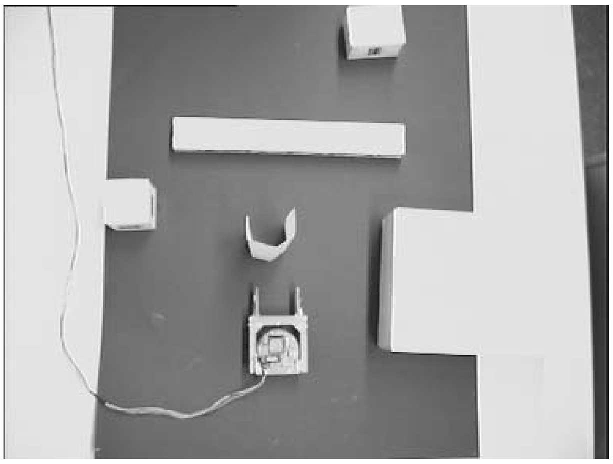


Figure 19: The robot and the obstacles in the environment.

After learning, the robot can track any contour of objects. The learning results of the control surface are shown in Figgure 21.

6 Discussion and Conclusions

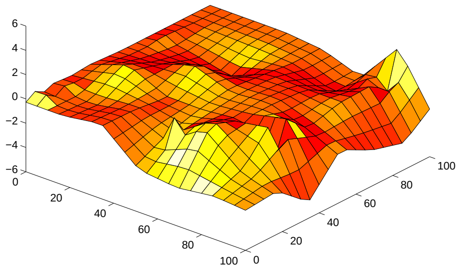
Some issues related to the construction procedures of a fuzzy controller are:

Computation of membership functions.
The B-functions are piecewise polynomials. Coefficients of nonuniform B-functions of any order can be computed recursively. The solutions for lower order B-functions can be derived explicitly. Therefore, they could easily be included in fuzzy development tools to facilitate the modelling of membership functions of such types of controllers.

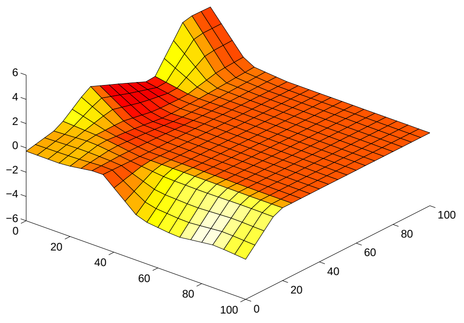
Choosing control vertices.
Note that the control vertices are only identical with the output values for interpolation if the order $k = 2$ (this agrees with the conclusion in [7]). For $k > 2$, control vertices are points near the interpolation point; they “control” the output curve to form a certain shape inside the convex hull of them. The larger k is, the greater the difference between control vertices and interpolation points. Normally, when rules are formulated using the “IF–THEN” convention, the singleton values of the output are initialised qualitatively in a manner enabling the controller to reach these values approximately; they can be optimised locally by fine-tuning.

Critera for selecting order k.
Obviously, if C^{k-2}-continuity is necessary, the order of B-functions should be at least k. However, too large a value of k leads to more marginal linguistic terms and thus more rules, as well as a larger disparity of control vertices and data points. In most applications, C^1- or C^2-continuity is sufficient. Then, B-functions of order 3 and 4 besides those of order 2 with triangular shape could well be suitable for modelling membership functions.

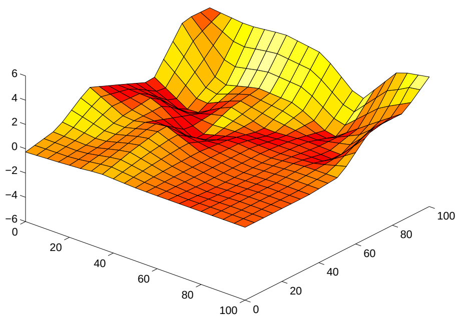
Optimal Partitioning of the Input Space.
To convert a fuzzy system to a B-spline interpolator, it should first be answered how the knots should be placed in the input space. An intuitive answer is to fix the knots where the output has its extrema. If such information is available (e.g., by approximating an analytically representable function), we can apply this principle to select the knots. If the output of a control system cannot



(a) w to $SensL = 0$, $SensF$, $SensR$



(b) w to $SensL$, $SensF = 0$, $SensR$



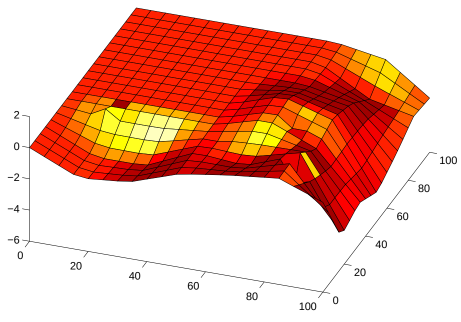
(c) w to $SensL$, $SensF$, $SensR = 0$

Figure 20: The control surfaces after 1000 learning steps.

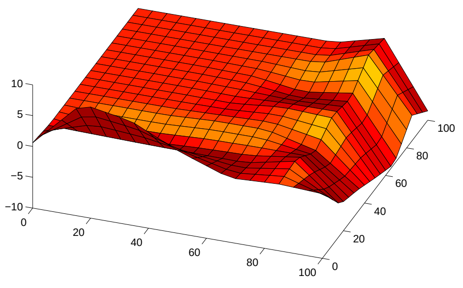
be analytically represented or be even unknown, we can adaptively compute the knots with an approach similar to the optimisation of a self-organising neural network. In this way, the optimal partitioning of the input space can be automatically achieved.

The transformation shown in Section 3 is conceptually important because it gprovides a construction method for data approximation using fuzzy controllers. The advantage of the fuzzy control idea over the pure B-spline interpolation lies mainly in its linguistic modelling ability: interpolation data can be prepared by using natural language with the help of expert knowledge. Furthermore, the interpolation procedure becomes transparent because it can also be interpreted in fuzzy logic "IF–THEN" form.

Experiments show the feasibility of such type of fuzzy controller, with B-functions as membership functions of input variables, singletons as membership functions of output variables, "product" as fuzzy conjunction, and centroid as defuzzification method. If the rule table is complete, then by adding certain more marginal rules, smoothness of the controller output can be achieved by selecting the proper order of B-functions. B-spline fuzzy controllers are exact in that no information is lost after the defuzzification. Although the number of control vertices to be optimised can be quite large in our approach, learning of such a fuzzy controller converges rapidly, especially for the supervised learning due to the properties of local support and one-minimum of the error function. Therefore, the computation time of the learning can be significantly reduced.



(a) After 100 learning steps



(b) After 1000 learning steps

Figure 21: The control surfaces for contour tracking.

Acknowledgements

The authors thank K. V. Le for implementing different versions of fuzzy controllers and Dr. X. Li of ProSTEP GmbH, Germany, for help with B-splines.

References

[1] M. Brown and C. J. Harris. *Neural networks for modelling and control*, chapter I, pages 17–55. In "Advances in Intelligent Control", Ed., C. J. Harris, Taylor & Francis, London, 1994.

[2] J. J. Buckley and Y. Hayashi. Fuzzy input-output controllers are universal approximators. *Fuzzy Sets and Systems*, 58:273–278, 1993.

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

[4] W. J. Gordon and R. F. Riesenfeld. B-spline curves and surfaces. In R. E. Barnhill and R. F. Riesenfeld, editors, *Computer Aided Geometric Design*. Academic Press, 1974.

[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] W. Pedrycz. Why triangular membership functions? *Fuzzy Sets and Systems*, 64:21–30, 1994.

[8] R. F. Riesenfeld. *Applications of B-Spline approximation to geometric problems of computer-aided design.* PhD thesis, Syracuse University, 1973.

[9] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics*, 4:45–99, 112–141, 1946.

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

[11] L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, 1994.

[12] H. W. Werntges. Partition of unity improve neural function approximators. In *Proceedings of IEEE International Conference on Neural Networks, San Francisco*, volume 2, pages 914–918, 1993.

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

[14] J. Zhang, J. Raczkowsky, and A. Herp. Emulation of spline curves and its application in robot motion control. *Proceedings of IEEE International Conference on Fuzzy Systems, Orlando*, pages 831–836, 1994.

[15] J. Zhang, F. Wille, and A. Knoll. Modular design of fuzzy controller integrating deliberative and reactive strategies. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, 1996.