Intelligent Automation and Soft Computing, Vol. 7, No. 2, pp. 145-157, 2001

FUZZIFICATION USING SPACE-FILLING CURVES

M. ELSHAFEI
Speech Recognition Applications
Mitel Corporation
350 Legget Dr., Kanata K2K 2W7, Canada
e-mail: moustafa_elshafei@mitel.com

AND

M.S. AHMED
E/E Engineering
DaimlerChrysler Corporation
CIMS #484-02-15
800 Chrysler Drive
Auburn Hills, MI 48326, USA
e-mail: msa7@daimlerchrysler.com

ABSTRACT—This paper introduces the Hilbert Space-Filling Curves SFC and outlines algorithms, which demonstrate the SFC efficient self-organizing features. We then propose a SFC fuzzy model based on clustering the object space. The SFC fuzzy model is utilized to construct a fuzzy feedback controller. It is shown that the self-replicating feature of SFC can be utilized to construct successively improving control strategies based on symmetric decision trees. The proposed SFC technique achieves a dramatic reduction in the complexity of fuzzy controllers by reducing the multi-dimensional fuzzification problem to a one-dimensional space.

Key Words: Space-filling curves, Vector Quantization, Fuzzy Logic, Artificial Intelligence, Fuzzy Control, nonlinear systems.

1. INTRODUCTION

The objective of this paper is to present a new model for fuzzy controller design based on Hilbert Space-filling Curve (SFC) technique. The SFC represents a powerful method for mapping a high dimensional signal vector into a reduced dimensionality feature vector or to a single dimensional vector. The method has already shown significant potential in a number of areas, and we believe it could have a competing position to the Artificial Neural Networks (ANN) due its computational efficiency and its inherent self-organizing characteristics. While the SFC technique has been accepted as a powerful method for clustering and vector quantization, Barthholdi and Platzman [1], in this paper we will demonstrate its potential in the design of fuzzy controllers.

First, in Section 2, we introduce the space-filling curve method. In Section 3, we outline algorithms for clustering and pattern classifications, and then we introduce the proposed SFC fuzzy inference system. Finally in section 4, we apply the SFC technique to the design of fuzzy controllers, present some simulation results, and propose some extensions of the method.

2. SPACE-FILLING CURVES

A Space-Filling Curve SFC is a means of travelling through an n-dimensional space via a single railroad. The SFC scans an n-dimensional space and reduces it to a one-dimensional line, and it has the characteristic that points that are close together on the resulting single dimensional curve are also close

together on the n-dimensional space. Of course, there is some loss of information on the closeness of points because a single dimension can not be one-to-one equivalent to a higher dimensional space. However, this loss can be quantified and confined to an acceptable level. Space-filling curves were first described by the mathematicians Peano [2], and Hilbert [3], as "topological monsters" since they seem contrary to intuition that a lower-dimensional space can be mapped continuously onto a space of higher dimension. Space-filling curves are part of a family of fractal curves, which have a self-similar feature; the curve is made up of a few symbols or "Cells", which are repeated in an orderly fashion until they fill the whole space.

An example of a space-filling curve on the plane is illustrated in Figure 1. It was first presented by Hilbert [3]. As shown in the figure, one partitions the interval [0,1] into 4 congruent intervals and the unit square into 4 congruent squares which are to be enumerated so that each two consecutively numbered squares have an edge in common, see Figure 1(a). If the entire interval [0-1] in one dimension can be mapped onto the squares, the intervals (0-1/4), ((1/4-1/2), (1/2-3/4), (3/4-1) respectively, can be mapped into the squares labeled as "0","1","2", and "3". Again, partition every subinterval into 4 congruent intervals, each sub-square into 4 congruent squares, enumerate the squares so that the preceding mapping is preserved and that each two consecutive squares have an edge in common, as shown in Figure 1(b), and repeat the previous procedure. The next step is depicted in Figure 1(c).

Proceeding in this manner, we obtain a mapping that is, by construction, is continuous. This also follows from the fact that each point in the original square lies in a sequence of closed nested squares the sides of which shrink to zero and that corresponds to a sequence of nested intervals, the lengths of which shrink to zero. In the limit the curve forms a continuous curve which passes through every point of the given area.

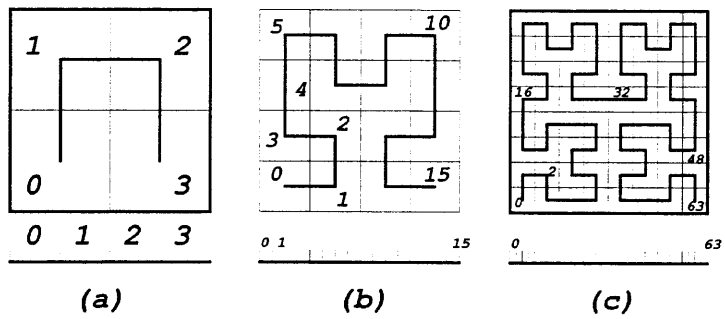

Figure 1. Generating the Hilbert space-filling curve.

Starting from Figure 1(a), it can be noticed that upon division of the basic cell, the offspring preserve the same shape of the basic cell, except possibly for their orientation or rotation. Second, the orientation and the locations of the offspring still preserve the general shape of the mother cell. Third, the orientation of a child cell depends on its location and the orientation of the mother cell. For the shown Hilbert curve, we can identify 4 different orientations of the basic cell and describe their division as the following:

$$A = \cap = \begin{bmatrix} \cap & \cap \\ \supset & \subset \end{bmatrix};$$

$$B = \supset = \begin{bmatrix} \cup & \supset \\ \cap & \supset \end{bmatrix};$$

$$C = \subset = \begin{bmatrix} \subset & \cup \\ \subset & \cap \end{bmatrix};$$

$$D = \cup = \begin{bmatrix} \supset & \subset \\ \cup & \cup \end{bmatrix} \quad (1)$$

Let us call the orientation of a cell as the "state" of the cell. Accordingly, the state of a child cell depends on its location as well as the state of its mother cell. This kind of cell division may be described by finite automata. We can define the finite automata as

$$\mathbf{M}=(\mathbf{Q}, \Sigma, \Phi, \Theta, \mathbf{Y}) \quad (2)$$

Where

Q : Finite set of states.

Σ : Finite nonempty set called the inputs, the set of possible position indices.

Φ : $\mathbf{Q} \times \Sigma \rightarrow \mathbf{Q}$: State transition mapping.

Y : is a finite nonempty set called the output or the set of possible orders of the child in the offspring family.

Θ : $\mathbf{Q} \times \Sigma \rightarrow \mathbf{Y}$: The output mapping.

The output mapping will be used to determine recursively the position on the line of a given n-dimensional vector. As an example, referring to Figure 1, Tables I.a and I.b

$$\mathbf{Q} = \{ A, B, C, D\}$$

$$\Sigma = \{ 00, 01, 10, 11\}$$

$$\mathbf{Y} = \{0, 1, 2, 3\}$$

The state transition is determined with the help of Table I.b, for example

$$\mathbf{Q} = \{A, B, C, D\};\ \Sigma = \{ 00, 01, 10, 11\}, \text{ and } \quad \mathbf{Y} = \{0, 1, 2, 3 \}$$

The state transition is determined with the help of Table I(b), for example

$$\Phi(A, 01) \rightarrow C \qquad ; \Phi(C, 11) \rightarrow D$$

Θ is determined with the help of Table I(a), for example

$$\Theta(A, 01) \rightarrow 3\ ; \qquad \Theta(C, 11) \rightarrow 0$$

Table I(a). Output mapping.

	A	B	C	D
00	0	0	2	2
01	3	1	3	1
10	1	3	1	3
11	2	2	0	0

Table I(b). State transition mapping.

	A	B	C	D
00	B	A	C	D
01	C	B	A	D
10	A	D	C	B
11	A	B	D	C

Let $X = (x_1, x_2, ..., x_n)$ and assume its components are quantized to m bits, i.e.

$$x_j = x_{j1}x_{j2}.....x_{jn}; \quad x_{ji} = 0,1$$

And let us define the sequence of vectors

$$u_i = .x_{1i}x_{2i}.....x_{ni}; \quad i = 1,2,...,m; \quad x_{ji} = 0,1$$

The position on the unit interval S as an $n*m$ bit binary number may be written as

$S = s_1 s_2 ... s_m$; Where s_j is an n bit string.

Now, given a space-filling curve defined by its initial generating state $q = q_0$ and an n-dimensional vector X, we can state the procedure to find its image S on the normalized space-filling curve S using $n*m$ bits accuracy as the following:

The SFC Mapping Algorithm:

Let $q = q_0$; The initial orientation of the basic cell.

for i= 1 to m

$s_i = \Theta(q, u_i)$; u_i and s_i as defined before.

$q = \Phi(q, u_i)$

next;

The normalized space-filling mapping S can be written as $S = s_1 s_2 ... s_m$.

For two dimensional space, $s_i = b_{i1} b_{i0}$, b_{ij} =0,1. Accordingly s_i takes only values in {0,1,2,3}. s_0 gives the position of X in the unit square in terms of the sub-squares 0, 1, 2, 3 as shown in Figure 1. Similarly, s_1 gives the position of X in the sub-square defined previously by s_0. Proceeding in this manner, s_i points to the position of X in terms of the sub-square of s_{i-1}, and so on. This feature will be used later on to derive decision trees and successive membership functions in the design of the SFC fuzzy model.

Bially [4] described a state diagram and a tabulation method for the generation of space-filling curves. The algorithm was demonstrated on 2D and 3D curves. Butz [5] derived an algorithm for performing the inverse mapping, i.e. given a point $S \in \mathbf{S}$ on the curve find the corresponding vector $X \in R^n$.

Other types of space-filling curves were proposed in the literatures. Patrick el. al. [6] considered multi-dimensional SFC for two types of curves: Dovetail mapping, and column mapping. The main advantage of these mappings is the simplicity of their generation in multi-dimensional spaces. Column mapping works only for odd modulo numbers as 1,3,5 etc. Dovetail is applicable for any modulo arithmetic, but the curve is discontinuous.

Bratholdi and Platzman [1,7,8] studied a family of space-filling circuits (closed curves) which turn to be very attractive for solving combinatorial problems in which N points are given and some combinatorial structure of maximal or minimal cost is sought, e.g., travelling salesman problem, weighted matching, and commercial routing problems. The main draw back of such a family of curves is that they are not one to one (a curve may visit a point more than one time). They also showed that the Space-filling heuristics require in general O(N log N) arithmetic operations and O(N) memory registers, where N is the number of data points. It was also proved that inserting a point or deleting a point from an N-point data requires only O(log N) arithmetic operations.

Extension of the 2-D SFC to higher dimensional spaces may also be achieved by Successive Compression (SC) or by Binary Compression (BC) of subspaces [9]. In SC the first and the second coordinates of the vectors are mapped to a single coordinate by the 2-D SFC. Next, the third coordinate is merged with the obtained coordinate point to produce a 3-D single point on the line, and so on. In BC, the coordinates of the vectors are compressed in pairs using 2-D SFC to obtain a new vector space with dimension n/2. Then the coordinates of the new vectors are compressed in pairs again to reduce the dimensionality of the space by one half. The process may then be repeated as many times until each vector is reduced to a single point on the line. Clearly, this latter method can proceed as described if $n=2^k$ for some integer k. The complexity of the two methods is the same. However, the BC is more attractive for parallel processing. Combinations of the two techniques, 2D, and 3D provide a flexible variety of mappings that can be tailored to suit specific applications.

3. FUZZY CLASSIFICATION USING SFC

In this section we first introduce the frame work of fuzzy clustering and classifications. We then provide SFC algorithms to achieve these goals. Finally, we introduce the SFC fuzzy inference engine model.

3.1 Fuzzy/Hard Clustering

Let us assume that we are given a finite set of data

$$\mathbf{X} = \{X_1, X_2, \ldots . X_N; X_j \in R^n\}$$

Let us call the space of such data as the input space or the object space. The clustering of **X** is the assignment (hard or fuzzy) of labels to the objects generating **X**. That is to identify K " natural subgroups" in **X**. This process is called ***unsupervised learning*** or self learning. For each of these subgroups or subset $\mathbf{C}_k$, k=1,2, K; we also associate a vector C_k, representing the center of gravity, or the centroid of the subset. The collection of these centroids constructs an n x K matrix $\mathbf{C}_B$, called the codebook. Moreover, for each vector $X_i \in \mathbf{X}$, we associate a vector $V_i \in R^K$, which indicates the membership values of the X_i in each set of the above mentioned partition, or in other words the membership values of the vector X_i such that

$$0 \le v_{ik} \le 1 ; \; k=1,2,\ldots K. \tag{3}$$

For constrained fuzzy membership, we may require

$$0 \le v_{ik} \le 1; \qquad \sum_k v_{ik} = 1 \tag{4}$$

In hard clustering, only one element of the vectors V_i is allowed to be one, while all the other elements will be zero, i.e.

$$v_{ik} \in \{0,1\}; \qquad \sum_k v_{ik} = 1 \tag{5}$$

3.2 Fuzzy/Hard Classification

Let **V** be the set of all vectors satisfying equation (3). A fuzzy classifier on $\Re^n$ is any function D; $D : \Re^n \rightarrow \mathbf{V}$. If the classifier is constrained, then we further require that the range of D satisfy equation (4). Finally for a hard classifier, the membership values are also required to satisfy condition (5) for each data vector X.

The difference between clustering and classification is that clustering algorithms label finite data sets $\mathbf{X} \in \Re^n$; whereas a classifier is capable of labeling every data point in the entire space. Classifiers are usually designed with labeled data, hence may require (but not always) ***supervised learning***. In supervised classifier design, labeled data set **X** is usually partitioned into a training or design set $\mathbf{X}_D$ with a hard/fuzzy label matrix $\mathbf{V}_D$, and a test set $\mathbf{X}_T$.

Testing of the classifier means submitting $\mathbf{X}_T$ to D and counting the errors assuming hard labels for the points in $\mathbf{X}_T$.

In Vector Quantization (VQ), K is usually chosen as $K = 2^b$. It is then required to determine the index k, represented by b bits, of the vector C_k from the codebook which is the nearest to a given arbitrary vector X.

During training, the clustering algorithm seeks to find the optimal K centroids using a large $\mathbf{X}_D$ set. The optimality is usually defined in terms of a cost function representing the overall distortion [10]. Classically, codebook training is performed using the so-called k-means clustering algorithms. Recently, ANNs [11-14] have been proposed for clustering and pattern classification, e.g. Kohonen Clustering Nets [14].

3.3 Clustering Using SFC

The most powerful feature of SFC is its ability to perform clustering and pattern classification very efficiently. It allows also to adaptively update the centroids as more input vectors arrive at a computational cost of O($N \log N$) [7]. Figure 2 illustrates the key idea behind SFC clustering, where grouping of vectors is based on clustering of their SFC positions on the unit interval.

The clustering problem can be solved efficiently using the following algorithms.

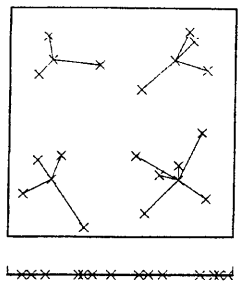

Figure 2. Clustering using SFC.

The K-Centers (KC) Algorithm

Choose K of N points, $N >> K$, so that the maximum distance between any point and its closest center is minimized.

Step 1: For each point, X, calculate, via a space-filling curve, its corresponding position, S_x on the unit interval.

Step 2: Divide the unit interval into K identical intervals, and define the jth subgroup to be

$$\mathbf{C}_j = \{X \in \mathbf{X}; \quad S_x \in [\frac{j-1}{K}, \frac{j}{K})\}$$

Take the centroids to be those points closest to the centers of the subintervals.

K-Balanced Clustering (KBC) Algorithm

The KBC algorithm divides the N points into K groups of equal population, and defines the centroids to be the median of each group.

Step 1: For each point X, calculate, via a space-filling curve, its corresponding position, S_x on the unit interval.

Step 2: Sort the points according to their corresponding positions on the unit interval, let i_x be the index of the vector X in the sorted table.

Step 3: From the sorted table divide the points into K groups, each consists of N/K points. Define the jth subgroup to be

$$\mathbf{C}_j = \{X \in \mathbf{X}; i_x \in [\frac{N}{K} * (j-1) + 1, \frac{N}{K} * j)\} \; ; \quad j=1,2,\ldots K. \tag{7}$$

Choose the K centroids to be the middle point of each group, i.e. those points of rank $[N/2K]$, $[3N/2K]$, ... $[(2K\text{-}1)N/2K]$.

Among the ramifications of the first two algorithms is to define the codebook vectors as the center of gravity of the selected subsets. Alternatively, the codebook vectors may be taken as those vectors, which correspond, to the mean value of the points in each subinterval.

3.4 Adaptive Clustering Using SFC

The SFC approach turns out to be very efficient as well when the population of the points of the codebook needs to be continuously updated as in adaptive codebooks. Since clustering is basically done on one dimensional space, it becomes feasible to assign a new vector to the appropriate subgroup and the re-compute its centroid. In finite horizon adaptive clustering, clustering is based on a moving window of the last N vectors. The training vectors are stored in a first-in first-out memory of size N. Updating of the centroids involves at most adding or adding/deleting one element from each group at a computational cost of O($N \log N$) [7].

3.5 SFC Fuzzy Model

The SFC fuzzy model, Figure 3, can be described by the following:

$$\{\mathbf{X}, \Psi, \mathbf{C}_\mathbf{B}, \mathbf{G}, W, \mathbf{U}\}$$

Where X is the object space or the input space, Ψ is the SFC mapping $\Psi : \Re^n \rightarrow \mathbf{S}$. $\mathbf{C}_\mathbf{B}$ is the codebook matrix, G is a set of K membership functions $\{g_k(.)\}$; $g_k : \mathbf{S} \rightarrow \Re$; k=1,2,...,K. W is the output mapping (consequences) $W : \Re^n \rightarrow \Re^m$, U is the output space consisting of vectors of dimension m.

The construction of the model starts by selecting appropriate K membership functions, performing the SFC clustering of the training data, and computing the set of the output weights as will be explained later. The proposed SFC fuzzy model, see Figure 3, works as the following:

1- For each input vector X, calculate, via a space-filling curve, its corresponding position, S_x on the unit interval.
2- Compute the fuzzy network output vector as

$$U(X) = \sum_{k=1}^{K} g_k(S_x) W(C_k) \tag{8}$$

Where C_k is the centroid of the Kth cluster. The kth membership function is normally dependent on λ_k, the image of the centroid vector C_k on the unit interval.
3- If the algorithm is adaptive, update the members of each subgroup, and compute new centrods $\{C_k\}$ of each changed group and update the output mapping W.

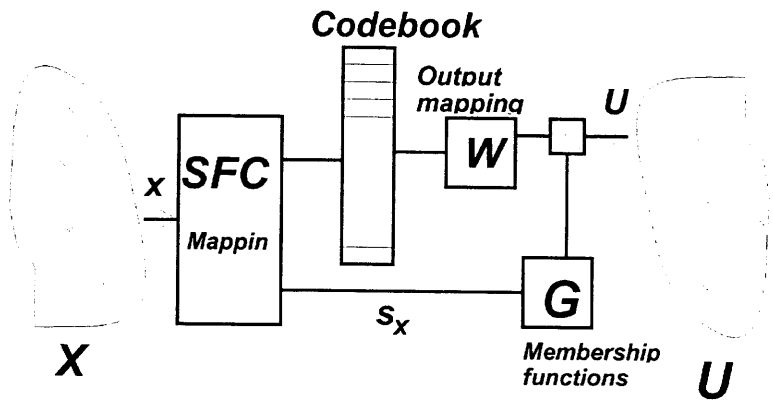

Figure 3. SFC Fuzzy model.

3.6 Training of the SFC Fuzzy model

The SFC fuzzy model presented above shares some similarities with the Tagaki-Sugeno fuzzy model [15]. The fuzzification layer and multiplication layers are implicitly included in the clustering of the input space. The clustering stage may also be viewed as a self-organizing unsupervised learning phase in which input vectors are grouped according to their SFC signatures. The membership functions are determined collectively on the SFC mapping of the input vector. The output-mapping layer is similar to the Sugeno's fuzzy output layer.

The SFC fuzzy model has the added advantage of controlling the function mapping resolution by successive partitioning of the input space. This feature will be demonstrated later on in section 4. The fuzzification parameters and the output mapping parameters need to be determined either adaptively, or by some prior training.

In supervised training, we assume that we have a training set (X_i, U_i) for i=1,2,...N. The set is collected by observing expert's decisions or from a reference model which generates the decision U_i for each input vector X_i. The training procedure may proceed as the following:
1- For each vector X_i generate its SFC mapping S_{xi}.
2- Cluster the input space by grouping the SFC mapping using one of the algorithms discussed before. Let $\{\mathbf{C}_k, C_k, \lambda_k\}$, k =1,2,..K be the generated clustered sets, their centroids, and the SFC mapping of their centroids respectively.
3- Select K membership functions $\{g_k(.)\}$; $g_k : \mathbf{S} \rightarrow \Re$.
4- Select the optimal weights W_k, k = 1,2 ..K to minimize

$$J = \sum_{i=1}^{N} (U_i - \sum_{k=1}^{K} g_k(S_{xi}) W_k)^T (U_i - \sum_{k=1}^{K} g_k(S_{xi}) W_k) \tag{9}$$

The solution of (9) can be obtained by the standard least square problem solving techniques or gradient-descent-algorithms [11].

Several observations can be made here. First, for rectangular membership functions W_k becomes the mean value of the output vectors taken over the set $\mathbf{C}_k$. Second, the membership functions are implicitly/or explicitly functions of the centroids of the clustered sets. For example, for triangular membership functions the vertex is centered at λ_k, and for Gaussian membership functions the mean value $m_k = \lambda_k$. Other types of membership functions may as well be defined, such as

$$g_k(S_x) = \frac{1}{1+\alpha_k(S_x-\lambda_k)^{2n_k}} \tag{10}$$

Third, in adaptive training, for each new set of input vectors, a new set of centroids and new weights are computed. The speed of adaptation can be controlled by changing the size of the memory holding the latest N vectors. A larger value of N causes slow adaptation, while a small value of N causes fast adaptation.

4. SFC FUZZY CONTROLLER

In this section we bring two examples to illustrate the proposed SFC fuzzy model as applied to the design of feedback fuzzy controllers. In the first example we utilized a simple 2D SFC fuzzy model which provides closed loop system characteristics comparable with a pole placement linear control technique. The second example illustrates the concept of successive approximation SFC fuzzy model in which the control value is iteratively refined by a quaternion decision tree in a nonlinear system.

Example 1:

The design of SFC fuzzy controller will be demonstrated using the following DC motor example. Let the pulse transfer function of a DC motor be given by

$$H(z) = \frac{K_p(1-b)}{(z-1)(z-a)}, \quad \text{where} \tag{11}$$

$a = e^{-T}$, $K_p = a - 1 + T$, $b = 1 - \dfrac{T(1-a)}{K_p}$, and $T = 0.25$ sec. is the sampling period.

A controller using pole placement based input-output model is illustrated in [16, pp. 296-299]. A pole placement controller with cancellation of the process zero produces unacceptable fluctuation in the control input. Here, we compare our fuzzy controller with the pole placement controller with no cancellation of process zero. The inputs to our SFC fuzzy controller have been the regulation error $e(t) = y_d(t) - y(t)$, and the difference in error $de(t) = e(t) - e(t-T)$. Both inputs are normalized in the range [0,1]. The unnormalized space-filling curve is divided into 4 clusters corresponding to the squares (0-10), (11-31), (32-52), and (53-63) respectively, as depicted in Figure 4. These clusters correspond roughly to the error regions shown in Figure 4(b). The membership functions are taken triangular as shown in Figure 4(c). The SFC fuzzy control law is given by

$$u(X) = \sum_{k=1}^{4} g_k(s_x) w_k \tag{12}$$

Where $X = (e(t), de(t))$, $\{g_k(.)\}$ are the membership functions, and $\{ w_k\}$ are the weights. The performance of the SFC fuzzy controller and the pole placement controller are shown in Figure 5. The corresponding control inputs are shown in figure 6. The figures clearly indicate that the performances are comparable. The values of w_1, w_2, w_3, and w_4 of equation (12) have been selected experimentally. The controller can easily be tuned up to the desired closed loop performance. For example, faster response can be achieved by increasing w_3, and w_4 by trading overshoot and/or the magnitude of the control input. On the other hand, decreasing the value of w_2 increases the settling time, but also reduces the magnitude of the

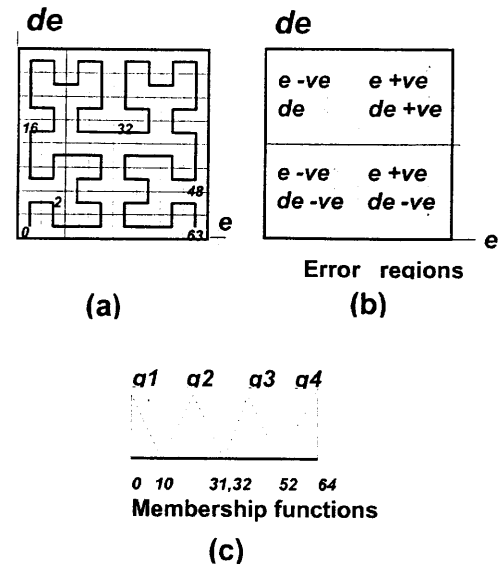

Figure 4. SFC fuzzy controller example.

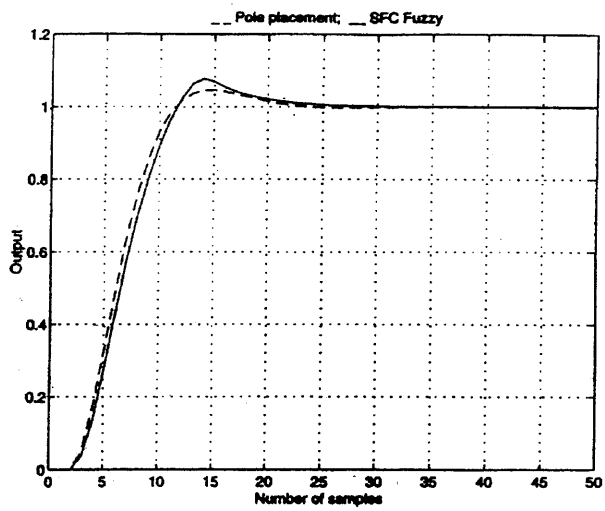

Figure 5. Performance of the pole-placement and the SFC fuzzy controller.

negative control input. Smooth control action can also be achieved by increasing the resolution of the space-filling curve mapping. Finer control of the steady state error and overshoot can be achieved by proper scaling of the error. Error dynamic range scaling may also be achieved by mapping the input first using a sigmoidal function of the following form

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \tag{13}$$

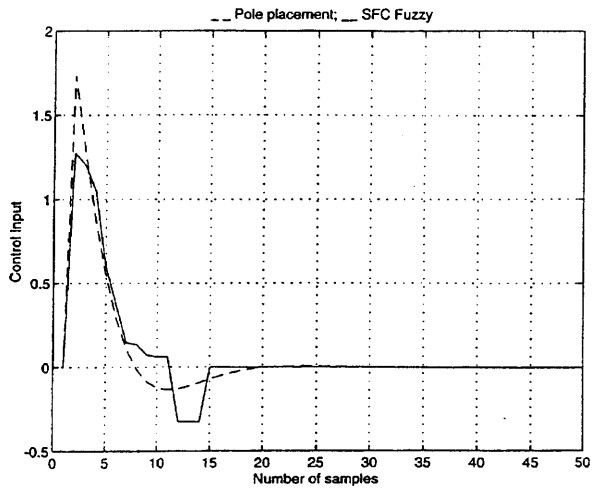

Figure 6. Control input for the pole-placement and SFC fuzzy controller.

The use of the sigmoidal function could improve the error resolution resulting in smoother control and reduced steady state error.

Example 2:

In fuzzy inference system, as more rules are involved a finer approximation as well as better modeling accuracy is likely to be achieved. In this example we propose a successive approximation control strategy. The control value is successively adjusted based on successive partitioning of the input space. The use of tree structures to organize the rules for pattern matching has been proposed in [17]. In binary boxtrees the top-level grid of the input space partition is coarsely partitioned into two fuzzy boxes, which can be further partitioned by finer fuzzy grid. Several criteria for performing such tree partitions have also been discussed by Sun.

The particular nature of the SFC mapping can help us to provide an efficient partitioning of the input space in a quaternion tree. Recall from Section 2 that

$$S_x = s_1 s_2 s_3 \ldots\ldots \tag{14}$$

For two dimensional space, s_i=b_{i1} b_{i0}, with $b_{ij} \in \{0,1\}$. s_1 gives the position of X in terms of the quadrants 0,1,2,3 as shown in Figure 1. We will call the selected rectangular $\mathbf{R}_{s1}$. Similarly, s_2 gives the position of X in the quadrant defined previously by s_1. Let us denote the selected sub-quadrant by $\mathbf{R}_{s1,s2}$. Proceeding in this manner, s_j points to the position of X inside the sub-quadrant assigned by s_{j-1}, where j is the resolution step, and so on. The successive approximation control strategy can now be achieved using a quaternion decision tree, where in each level one out of four possible regions is selected. The successive approximation control strategy can be constructed by the use of the following iteration

$$u^j = u^{j-1} + \beta\alpha^j \Big(\sum_{R_j} g_k^j(S_x) w_k^j\Big) \tag{15}$$

Where β and α are constants, $\beta > 0$, $0 < \alpha \leq 1$, j is the resolution level, w_k^j are appropriate weights corresponding to the ***consequences*** of the zero-order Takagi-Sugeno fuzzy model, and R_j= $\mathbf{R}_{s1,s2,s3..sj}$. The correction factor at the jth level is computed by applying the fuzzy rules defined on the region $\mathbf{R}_{s1,s2,..sj}$.

Unlike Takagi-Sugeno [15], the rules here are arranged in a specific tree structure, which enables the FLC output to be refined as we go down into the decision tree. Thus refinement of the fuzzy mapping can be done by an add-on basis, while in the traditional fuzzy systems one has to start all over again.

To demonstrate the concept, we consider here the problem of balancing an inverted pendulum. It is a complex nonlinear system, which is inherently unstable. The objective of the controller is to restore the inverted pendulum to its vertical (unstable) equilibrium position. The model for the inverted pendulum is given by Jang [18]

$$\theta'' = \frac{g\sin(\theta) + \left[\frac{-u - mL\theta'^2\sin(\theta)}{m_c + m}\right]\cos(\theta)}{\left[\frac{4}{3} - \frac{m\cos^2(\theta)}{m_c + m}\right]L} \tag{16}$$

Where g : is the gravitational acceleration, L : is the length of the stick, m : is the mass of the stick, m_c: is the mass of the cart, u : is the control input, and θ : is the angular displacement of the rod from the vertical position. m, m_c, and L have been chosen to be 0.2 kg, 1.0 kg, and 0.5 meters respectively. The sampling time was 0.02 second. For the simulation a 4th order Runge-Kutta method has been used. A linear control rule can be obtained by assuming a small deviation from the equilibrium position, linearizing the system model, and applying a pole placement output feedback. The linear control law is given by

$$u(t) = K_p\theta(t) + K_d(\theta(t) - \theta(t-1)).$$

Where K_p and K_d are the proportional and the derivative gains respectively. The performance of this controller is shown in Figure 7. For a small initial angle of amplitude 30 degrees. However, when the initial angle was increased to 60 degrees the closed loop system became unstable and the stick fell down, as shown in the same figure.

In the proposed SFC controller we assume that only the angle θ is available for measurement. The inputs to the SFC controller are the normalized signal $x_1(t) = \theta(t)$ and the normalized difference $x_2(t) = x_1(t) - x_1(t\text{-}1)$.

The normalization is obtained using the sigmoidal function as explained prior to equation (11). The controller this time generates the control output in a successive approximation strategy. For simplicity the membership functions are taken to be rectangular. The initial control law is given simply by $u^1 = W(s_1)$, where $W(:)$ is a constant consequence vector. This crude control value can be improved further if the position of X is refined by successive partition via the iteration

$$u^j = u^{j-1} + \beta\alpha^j W(s_1, s_2, \ldots s_{j-1}; s_j) \tag{17}$$

Where $W(s_1, s_2, \ldots s_{j-1}; :)$ is a weight vector corresponding to the region

$R_j = \mathbf{R}_{s1,s2,s3.sj\text{-}1}$. In this particular example the initial consequence vector is taken to be W = [-12, -1.2, 12, 1.2], and the subsequent weight vectors are taken to be constants over all the partitions of the four principal quadrants, i.e.

$$W(s_1, s_2, \ldots s_{j-1}; s_j) = W(s_1; s_j),$$

where

$$W(0;:) = [2\ \ 1\ \ 0\ \ 1],\ W(1,:) = [1\ \ 2\ \ 1\ \ 0],$$

$$W(2,:) = [0\ \ 1\ \ 2\ \ 1],\ \text{and}\ W(3,:) = [1\ 0\ \ 1\ \ 2].$$

The control system behavior for α =0.5 at the initial conditions of θ =30° and θ =60° are shown in Figure 8. The controller can be coarsely tuned by properly choosing the weights $W(s_1)$. Since we expect symmetric behavior for positive and negative errors, we need only to find $W(0)$ and $W(1)$ experimentally. Faster response can be achieved if we increase $W(0)$ by trading overshoot and/or the magnitude of the

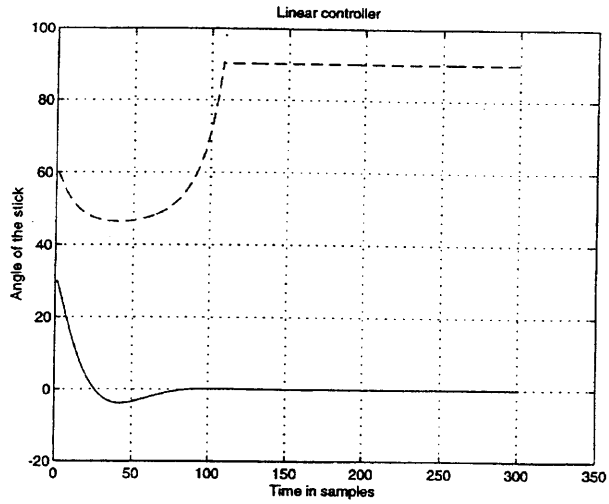

Figure 7. Performance of the linear controller at initial angles of 30 and 60 degrees.

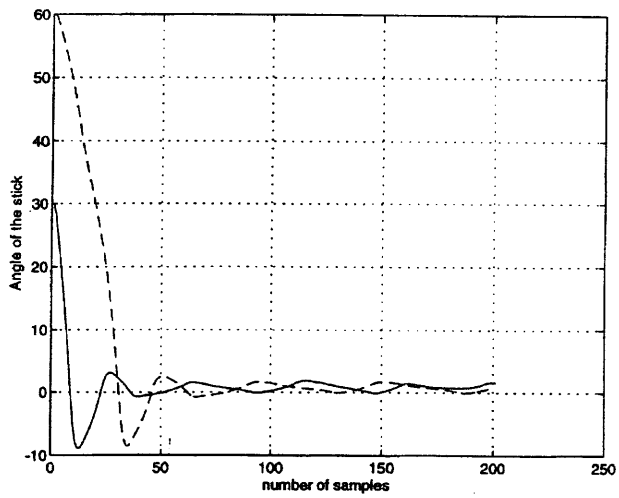

Figure 8. Performance of the SFC fuzzy controller.

control input. On the other hand, decreasing the value of $W(1)$ increases the settling time, but decreases the demand for control fluctuations.

The simulation study on the classical inverted pendulum problem establishes the effectiveness of the proposition. The SFC fuzzy controller managed to control the system in a region where the linear controller failed to regulate the system.

5. CONCLUSION

The paper demonstrates the potential of SFC heuristics as an efficient method for fuzzy controller design. The SFC reduces the problem of multidimensional fuzzification to a fuzzification in one dimension. SFC allows fuzzification to be based on multidimensional regions and clustering rather than single

dimensional intervals for each variable. It is believed that SFC could play a competing position to the ANN due to its efficient self-organizing characteristics.

REFERENCES

[1] Bartholdi, III, J.J. and Platzman, L.K., "A Fast Heuristic Based on Space-filling Curves for Minimum-Weight Matching in the Plane", *Information Processing Letters*, Vol. 17, 1983, pp. 177-180.

[2] Peano, G., "Sur une courbe qui remplit toute en aire plane", *Math. Ann.*, Vol. 36, 1890.

[3] Hilbert, D., "Uber die steitige Abbildung einer Linie auf ein Flachenstuck", *Math. Ann.*, Vol. 38, 1891.

[4] Bially, T., "Space-filling Curves: Their Generation and Their Application to Bandwidth Reduction", *IEEE Trans. Inform. Theory,* Vol. IT-15, No. 6, 1969, pp. 658-664.

[5] Butz, A.R., "Alternative Algorithm for Hilbert's Space-filling Curve", *IEEE Trans. Comput.*, C-20, 1971, pp. 424-426.

[6] Patrick, E.A., Anderson, D.R., and Bechtel,F.K., "Mapping Multidimensional Space to One Dimension for Computer Output Display", *IEEE Trans. Comput.*, Vol. C-17, No. 10, 1968, pp. 949-953.

[7] Bartholdi, III, J.J. and Platzman, L.K., "Heuristics based on Space-filling Curves for Combinatorial Problems in Euclidean Space", *Management Science*, Vol. 34, No. 3, 1988, pp. 291-305.

[8] Bartholdi, III, J.J. and Platzman, L.K., "Space-filling Curves and the Planer Travelling Salesman Problem", *Journal of ACM*, Vol.36, No.4, 1989, pp. 719-737.

[9] Elshafei-Ahmed, M., Al-Suwaiyel, M. I., and Akyldiz, Y., "Fast Algorithm For Vector Quantization of Speech", *Proc. the 6th Int. Conf. on Comput. Theory and Appl.*, 1996, pp. 485-488.

[10] Gersho, A., and Gray, R.M., *Vector Quantization and Signal Compression*, Boston, Mass., Kluwer Academic Publisher, 1992.

[11] Haykin, S., *NEURAL NETWORKS a Comprehensive Foundation*, New York: Macmillan College Publishing Company, 1994.

[12] Wasserman, P.D., *NEURAL COMPUTING: Theory and Practice*, New York: Van Nostrand and Reinhold, 1989.

[13] Widrow, B., Winter, R.G., and Baxter, R.A., "Layered Neural Nets for Pattern Recognition", *IEEE Trans. Acoust. Speech and Signal Proc.*, Vol. 36, 1988, pp. 1109-1118.

[14] Kohonen, T., *Self-Organization and Associative Memory*, 3rd ed., North Holand: Springer Verlag, 1989.

[15] Tagaki, T. and Sugeno, M., "Fuzzy Identification of Systems and its Applications to Modeling and Control", *IEEE Trans. Systems, Man, and Cybernatics*, Vol. SMC-15, 1985, pp. 116-132.

[16] Astrom, K.J. and Wittenmark, B., *Computer Controlled Systems; Theory and Design,* 2nd ed, New Jersey: Prentice-Hall International, Inc., 1990.

[17] Sun, C.T., "Rule-Based Structure Identification in an Adaptive-Network-Based Fuzzy Inference System", *IEEE Trans. on Fuzzy Systems*, Vol. 2, No. 1, 1994, pp. 64-73.

[18] Jang, J.R. and Sun, C., "Neuro-Fuzzy Modeling and Control", *Proceedings of the IEEE*, Vol. 83, No. 3, 1995, pp. 378-405.