20

System Identification Using Fuzzy Cerebellar Model Articulation Controllers

Cheng-Jian Lin* and Chun-Cheng Peng
National Chin-Yi University of Technology
Taiwan, R. O. C.

1. Introduction

Being an artificial neural network inspired by the cerebellum, the cerebellar model articulation controller (CMAC) was firstly developed in (Albus, 1975a, 1975b). With the advantages such as fast learning speed, high convergence rate, good generalization capability, and easier hardware implementation (Lin & Lee, 2009; Peng & Lin, 2011), the CMAC has been successfully applied to many fields; for example, identification (Lee et al., 2004), image coding (Iiguni, 1996), ultrasonic motors (Leu et al., 2010), grey relational analysis (Chang et al., 2010), pattern recognition (Glanz et al., 1991), robot control (Harmon et al., 2005; Mese, 2003; Miller et al., 1990), signal processing (Kolcz & Allinson, 1994), and diagnosis (Hung & Wang, 2004; Wang & Jiang, 2004). However, there are three main drawbacks of Albus' CMAC, i.e., larger required computing memory (Lee et al., 2007; Leu et al., 2010; Lin et al., 2008)), relatively poor ability of function approximation (Commuri & Lewis, 1997; Guo et al., 2002; Ker et al., 1997), and difficulty of adaptively selecting structural parameters (Hwang & Lin, 1998; Lee et al., 2003).

In order to tackle these disadvantages, several methods, such as online-based clustering (Kasabov & Song, 2002; Tung & Quek, 2002) for the above-mentioned first drawback, B-spline functions (Lane et al., 1992; Wu & Pratt, 1999) and fuzzy concepts (Jou, 1992; Chen, 2001; Guo et al., 2002; Ker et al., 1997; Lai & Wong, 2001; Zhang & Qian, 2000) for the second one, and competitive learning (Chow & Menozzi, 1994), clustering (Hwang & Lin, 1998) and Shannon's entropy and golden-section search (Lee et al., 2003) for the third one, were proposed. Among these approaches, further improvements were implemented by Lin et al. (2008) with self-constructing algorithm and Gaussian basis functions.

The rest of this chapter is organized as follows. Starting from the first CMAC model in 1975 the development processes, related learning algorithms and system identification examples of the fuzzy CMACs are briefly reviewed in section 2. Sections 3 and 4 respectively discuss the self constructing FCMAC (SC-FCMAC) and the powerful parametric FCMAC (P-FCMAC). Lastly, section 5 concludes this chapter, with suggested directions of further researches.

* Corresponding Author

2. From CMACs to fuzzy CMAC models

2.1 The traditional CMAC models

As mentioned in the previous section, the traditional CMAC model (Albus, 1975a, 1975b) has fast learning ability and good local generalization capability for approximating nonlinear functions. The basic idea of the CMAC model is to store learned data in overlapping regions in a way that the data can easily be recalled yet use less storage space. The action of storing weight information in the CMAC model is similar to that of the cerebellum in humans. Take a two-dimensional (2-D) input vector, or the so-called two-dimensional CMAC (2-D CMAC), as an example, while its structure is shown as in Fig. 1. The input vector is defined by two input variables, s_1 and s_2, which are quantized into three discrete regions, called blocks. It is noted that the width of the blocks affects the generalization capability of the CMAC. In the first method of quantization, s_1 and s_2 are divided into blocks A, B, and C, and blocks a, b, and c, respectively. The areas Aa, Ab, Ac, Ba, Bb, Bc, Ca, Cb, and Cc formed by quantized regions are called hypercubes. When each block is shifted by a small interval, different hypercubes can be obtained. In Fig. 1, there are 27 hypercubes used to distinguish 49 different states in the 2-D CMAC. For example, let the hypercubes Bb and Ee be addressed by the state (s_1, s_2) = (2, 2). Only these three hypercubes are set to 1, and the others are set to 0.

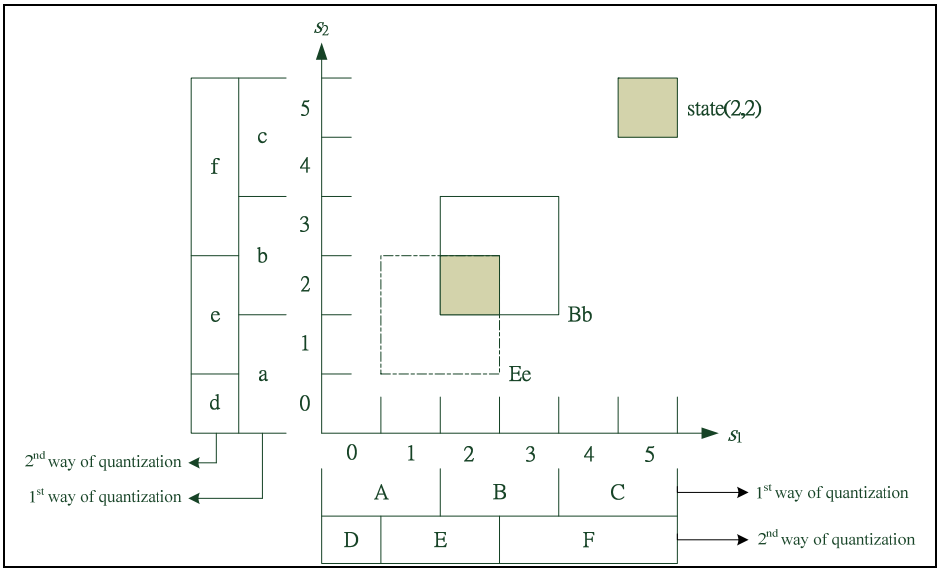


Fig. 1. Structure of a 2-D CMAC

In the work of (Mohajeri et al., 2009a) the hash coding and corresponding comparison with other neural networks for the CMACs were reviewed. There are 13 architectural improvements discussed in (Mohajeri et al., 2009a), including approaches of interpolation, numerous transfer functions (such as sigmoid, basic spline, general basis and wavelet ones), weighted regression, optimized weight smoothing, recurrence and generalization issue. Learning techniques such as neighborhood sequential training and random training were

firstly considered, while the developments of other five training schemes (i.e., credit assignment, gray relational, error norm, active deformable and Tikhonov ones) were mentioned as well. In order to reduce relative memory usages, proposed approaches of hierarchical and self-organizing CMACs were reasoned, whereas the fuzzy variation of the self-organizing CMAC will be further presented in the following section of this chapter.

2.2 Fuzzy CMAC models

Many researchers have integrated the fuzzy concept into the CMAC network, such as in (Chen 2001; Dai et al., 2010; Guo et al., 2002; Jou, 1992; Ker et al., 1997; Lai & Wong, 2001; Lee & Lin, 2005; Lee et al., 2007a; Lee et al., 2007b; Lin & Lee, 2008; Lin & Lee, 2009; Lin et al., 2008; Peng & Lin, 2011; Wang, 1994; Zhang & Qian, 2000). In general, they use membership functions rather than basis functions, and the resulting structures are then called fuzzy CMACs (FCMACs).

In addition, the work of (Mohajeri et al., 2009b) provides a review of FCMACs, including over 23 relative aspects such as membership function, memory layered structure, defuzzification and fuzzy systems, was provided. Even FCMACs have originally reduced memory requirement for the CMAC, further discussions of clustering (such as fuzzy C-mean, discrete incremental clustering and Bayesian Ying-Yang) and hierarchical approaches for reducing memory sizes of FCMACs themselves were overviewed in (Mohajeri et al., 2009b) as well. Furthermore, as divided in (Dai et al., 2010), there are two classes of FCMACs architectures, i.e., forward and feedback fuzzy neural networks, which is useful for beginners to have a big picture of the basic concept for the FCMACs.

In the following sections, being the example models in this chapter the self-constructing FCMAC (SC-FCMAC, Lee et al., 2007a) and the powerful parametric FCMAC (P-FCMAC, Lin & Lee, 2009) are reviewed, in order to provide readers the insight knowledge of how these FCMAC work. Companied by their corresponding architectures and learning schemes, illustrative examples of system identification are provided as well.

3. The self-constructing fuzzy CMAC

From relative architectures to learning algorithms this section provides a brief review and discussions of the self-constructing FCMAC (SC-FMAC, Lee et al., 2007a).

3.1 Architecture of the SC-FCMAC model

As illustrated in Fig. 2, the SC-FCMAC model (Lee et al., 2007a) consists of the input space partition, association memory selection, and defuzzification. Similar to the traditional CMAC model, the SC-FCMAC model approximates a nonlinear function $y = f(x)$ by applying the following two primary mappings:

$$S : X \Rightarrow A \tag{1}$$

$$P : A \Rightarrow D \tag{2}$$

where X is an s-dimensional input space, A is an N_A-dimensional association space, and D is a 1-dimensional (1-D) output space. These two mappings are realized by using fuzzy

operations. The function $S(x)$ maps each point x in the input space onto an association vector $\alpha = S(x) \in A$ that has N_L nonzero elements ($N_L < N_A$). Here, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{N_A})$, where $0 \le \alpha \le 1$ for all components in α is derived from the composition of the receptive field functions and sensory inputs. Different from the traditional CMAC model, several hypercubes are addressed by the input state x. The hypercube values are calculated by product operation through the strength of the receptive field functions for each input state.

In the SC-FCMAC model, we use the Gaussian basis function as the receptive field function and the fuzzy weight function for learning. Some learned information is stored in the fuzzy weight vector. The 1-D Gaussian basis function can be given as follows:

$$\mu(x) = e^{-((x-m)/\sigma)^2} \tag{3}$$

where x represents the specific input state, m represents the corresponding center, and σ represents the corresponding variance.

Let us consider a N_D-dimensional problem. A Gaussian basis function with N_D dimensions is given as follows:

$$\alpha_j = \prod_{i=1}^{N_D} e^{-((x_i - m_{ij})/\sigma_{ij})^2} \tag{4}$$

where Π represents the product operation, α_j represents the j-th element of the association memory selection vector, x_i represents the input value of the i-th dimension for a specific input state x, m_{ij} represents the center of the receptive field functions, σ_{ij} represents the variance of the receptive field functions, and N_D represents the number of the receptive field functions for each input state. The function $P(a)$ computes a scalar output y by projecting the association memory selection vector onto a vector of adjustable fuzzy weights. Each fuzzy weight is inferred to produce a partial fuzzy output using the value of its corresponding association memory selection vector as the input matching degree. The fuzzy weight is considered here so that the partial fuzzy output is defuzzified into a scalar output using standard volume-based centroid defuzzification (Kosko, 1997; Paul & Kumar, 2002). The term volume is used in a general sense to include multi-dimensional functions. For 2-D functions, the volume reduces to the area. If v_j is the volume of the consequent set and ζ_j is the weight of the scale α_j, then the general expression for defuzzification is

$$y = \frac{\sum_{j=1}^{N_L} \alpha_j w_j^m v_j \zeta_j}{\sum_{j=1}^{N_L} \alpha_j v_j \zeta_j} \tag{5}$$

where w_j^m is the mean value of the fuzzy weights and N_L is the number of hypercube cells. The volume v_j in this case is simply the area of the consequent weights, which are represented by Gaussian fuzzy sets. Therefore, $v_j = w_j^{\alpha}\sqrt{\pi}$, where w_j^{α} represents the variance of the fuzzy weights. If the weight ζ_j is considered to be one, as in this work, then the actual output y is derived as

$$y = \frac{\sum_{j=1}^{N_L} \alpha_j w_j^m w_j^\alpha}{\sum_{j=1}^{N_L} \alpha_j w_j^\alpha}. \tag{6}$$

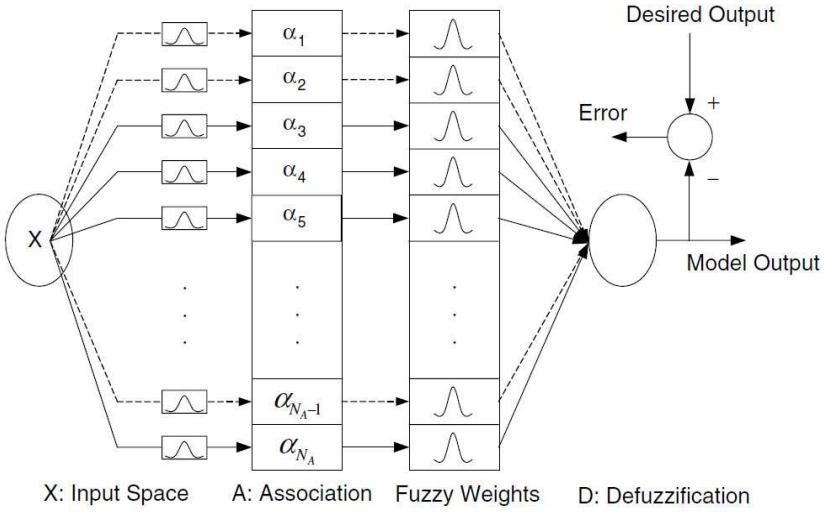


Fig. 2. Structure of the SC-FCMAC model (Lee et al., 2007)

3.2 Learning Algorithm of the SC-FCMAC

In this section, for completing the SC-FCMAC model (Lee et al., 2007a) the self-constructing learning algorithm, which consists of an input space partition scheme (i.e., scheme 1) and a parameter-learning scheme (i.e., scheme 2), is reviewed. First, the input space partition scheme is used to determine proper input space partitioning and to find the mean and the width of each receptive field function. This scheme is based on the self-clustering method (SCM) to appropriately determine the various distributions of the input training data. Second, the parameter-learning scheme is based on the gradient descent learning algorithm. To minimize a given cost function, the receptive field functions and the fuzzy weights are adjusted using the back-propagation algorithm. According to the requirements of the system, these parameters will be given proper values to represent the memory information. For the initial system, the values of the tuning parameters w_j^m and w_j^σ of the fuzzy weights are generated randomly, and the m and σ of the receptive field functions are generated by the proposed SCM clustering method.

3.2.1 The input space partition scheme

The receptive field functions can map input patterns. Hence, the discriminative ability of these new features is determined by the centers of the receptive field functions. To achieve good classification, centers are best selected based on their ability to provide large class separation.

An input space partition scheme, called the SCM, is used to implement scatter partitioning of the input space. Without any optimization, the proposed SCM is a fast, one-pass algorithm for a dynamic estimation of the number of hypercube cells in a set of data, and for finding the current centers of hypercube cells in the input data space. It is a distance-based connectionist-clustering algorithm. In any hypercube cell, the maximum distance between an example point and the hypercube cell center is less than a threshold value, which has been set as a clustering parameter and which would affect the number of hypercube cells to be estimated.

In the clustering process, the data examples come from a data stream, and the process starts with an empty set of hypercube cells. When a new hypercube cell is created, the hypercube cell center, C, is defined, and its hypercube cell distance and hypercube cell width, Dc and Wd, respectively, are initially set to zero. When more samples are presented one after another, some created hypercube cells will be updated by changing the positions of their centers and increasing the hypercube cell distances and hypercube cell width. Which hypercube cell will be updated and how much it will be changed depends on the position of the current example in the input space. A hypercube cell will not be updated any more when its hypercube cell distance, Dc, reaches the value that is equal to the threshold value D_{thr}.

Figure 3 shows a brief clustering process using the SCM in a two-input space. The detailed clustering process can be found in (Lee et al., 2007a).

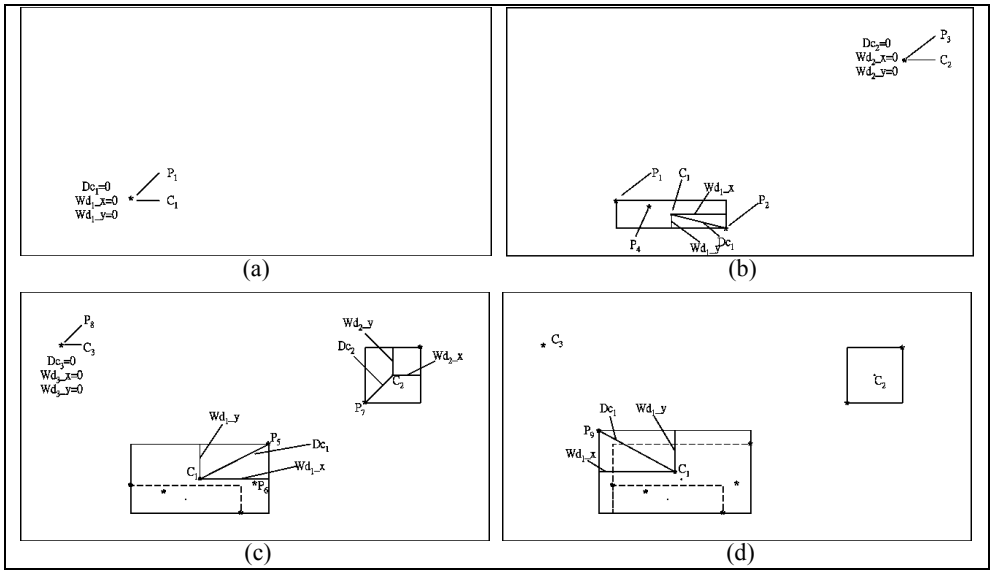


Fig. 3. A brief clustering process using the SCM with samples P_1 to P_9 in a 2-D space. (Notations: P_i for pattern, C_j for hypercube cell center, Dc_j is hypercube cell distance, Wd_j_x represents x-dimensions hypercube cell width, and Wd_j_y stands for y-dimensions hypercube cell width) (a) The example P_1 causes the SCM to create a new hypercube cell center C_1. (b) P_2: update hypercube cell center C_1, P_3: create a new hypercube cell center C_2, P_4: do nothing. (c) P_5: update hypercube cell C_1, P_6: do nothing, P_7: update hypercube cell center C_2, P_8: create a new hypercube cell C_3. (d) P_9: update hypercube cell C_1.

In this way, the maximum distance from any hypercube cell center to the examples that belong to this hypercube cell is not greater than the threshold value D_{thr}, though the algorithm does not keep any information on passed examples. The center and the jump positions of the receptive field functions are then defined by the following equation:

$$m_j = C_j,\ j = 1,2,...,n, \tag{7}$$

$$\theta_j^r = \frac{1}{\left((n_s+1)/2\right)} \cdot r \cdot D_j, \tag{8}$$

where $j = 1,2,...,n,$ and $r = 1,2,...,n_s$.

The threshold parameter D_{thr} is an important parameter in the input space partition scheme. A low threshold value leads to the learning of fine clusters (such that many hypercube cells are generated), whereas a high threshold value leads to the learning of coarse clusters (such that fewer hypercube cells are generated). Therefore, the selection of the threshold value D_{thr} critically affects the simulation results, and the threshold value is determined by practical experimentation or trial-and-error

3.2.2 The parameter-learning scheme

In the parameter-learning scheme, there are four adjustable parameters (m_{ij},σ_{ij}, w_j^m, and w_j^σ) that need to be tuned. The parameter-learning algorithm of the SC-FCMAC model uses the supervised gradient descent method to modify these parameters. When we consider the single output case for clarity, our goal is to minimize the cost function E, defined as follows:

$$E = \frac{1}{2}\left(y^d(t) - y(t)\right)^2, \tag{9}$$

where $y^d(t)$ denotes the desired output at time t and $y(t)$ denotes the actual output at time t. The parameter-learning algorithm, based on back-propagation, is defined as follows.

The fuzzy weight cells are updated according to the following equations:

$$w_j^m(t+1) = w_j^m(t) + \Delta w_j^m, \tag{10}$$

$$w_j^\sigma(t+1) = w_j^\sigma(t) + \Delta w_j^\sigma \tag{11}$$

where j denotes the j-th fuzzy weight cell for j=1,2,…N_L, w_j^m the mean of the fuzzy weights, and w_j^σ the variance of the fuzzy weights. The elements of the fuzzy weights are updated by the amount

$$\Delta w_j^m = \eta \cdot e \cdot \frac{\partial y}{\partial w_j^m} = \eta \cdot e \cdot \frac{\alpha_j w_j^\sigma}{\sum_{j=1}^{N_L} \alpha_j w_j^\sigma} \tag{12}$$

$$\begin{aligned}\Delta w_j^{\sigma} &= \eta \cdot e \cdot \frac{\partial y}{\partial w_j^{\sigma}} \\ &= \eta \cdot e \cdot \frac{\alpha_j w_j^{m} \sum_{j=1}^{N_L} \alpha_j w_j^{\sigma} - \alpha_j \sum_{j=1}^{N_L} \alpha_j w_j^{m} w_j^{\sigma}}{\left(\sum_{j=1}^{N_L} \alpha_j w_j^{\sigma}\right)^2}\end{aligned} \tag{13}$$

where η is the learning rate of the mean and the variance for the fuzzy weight functions between 0 and 1, and e is the error between the desired output and the actual output, $e = y^d - y$.

The receptive field functions are updated according to the following equations:

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij} \tag{14}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij} \tag{15}$$

where i denotes the ith input dimension for i=1,2,…,n, m_{ij} denotes the mean of the receptive field functions, and σ_{ij} denotes the variance of the receptive field functions. The parameters of the receptive field functions are updated by the amount

$$\begin{aligned}\Delta m_{ij} &= \eta \cdot e \cdot \frac{\partial y}{\partial \alpha_j} \cdot \frac{\partial \alpha_j}{\partial m_{ij}} \\ &= \eta \cdot e \cdot \frac{w_j^{m} w_j^{\sigma} \sum_{j=1}^{N_L} \alpha_j w_j^{\sigma} - w_j^{\sigma} \sum_{j=1}^{N_L} \alpha_j w_j^{m} w_j^{\sigma}}{\left(\sum_{j=1}^{N_L} \alpha_j w_j^{\sigma}\right)^2} \cdot \alpha_j \cdot \frac{2\left(x_i - m_{ij}\right)}{{\sigma_{ij}}^2}\end{aligned} \tag{16}$$

$$\begin{aligned}\Delta \sigma_{ij} &= \eta \cdot e \cdot \frac{\partial y}{\partial \alpha_j} \cdot \frac{\partial \alpha_j}{\partial \sigma_{ij}} \\ &= \eta \cdot e \cdot \frac{w_j^{m} w_j^{\sigma} \sum_{j=1}^{N_L} \alpha_j w_j^{\sigma} - w_j^{\sigma} \sum_{j=1}^{N_L} \alpha_j w_j^{m} w_j^{\sigma}}{\left(\sum_{j=1}^{N_L} \alpha_j w_j^{\sigma}\right)^2} \cdot \alpha_j \cdot \frac{2\left(x_i - m_{ij}\right)^2}{{\sigma_{ij}}^3}\end{aligned} \tag{17}$$

where η is the learning rate of the mean and the variance for the receptive field functions.

3.3 An example: Learning chaotic behaviors

A nonlinear system $y(t)$ with chaotic behaviors (Wang, 1994) is defined by the following equations, i.e.,

$$\dot{x}_1(t) = -x_1(t){x_2}^2(t) + 0.999 + 0.42\cos(1.75t) \tag{18}$$

$$\dot{x}_2(t) = x_1(t){x_2}^2(t) - x_2(t) \tag{19}$$

$$y(t) = \sin\left(x_1(t) + x_2(t)\right) \tag{20}$$

We solved the differential Eqs. (18) and (19) with t from t=0 to t=20 and with $x_1(0)$=1.0 and $x_2(0)$=1.0. We obtained 107 values of $x_1(t)$ and $x_2(t)$ (the chaotic glycolytic oscillator, Wang, 1994) and 107 values of $y(t)$. Figure 4 shows $y(t)$, which is the desired function to be learned by the SC-FCMAC model.

The input data were $x_1^p(t)$ and $x_2^p(t)$, and the output data was $y^p(t)$, for p=1,2,...,107. For this chaotic problem, the initial parameters η=0.1 and D_{thr} =1.3 were chosen. First, using the SCA clustering method, we obtained three hypercube cells. The learning scheme then entered parameter learning using the back-propagation algorithm. The parameter training process continued for 200 epochs, and the final trained rms (root mean square) error was 0.000474. The number of training epochs is determined by practical experimentation or trial-and-error tests.

We compared the SC-FCMAC model with other models (Lin et al., 2004; Lin et al., 2001). Figure 5(a) shows the learning curves of the SC-FCMAC model, the FCMAC model (Lin et al., 2004), and the SCFNN model (Lin et al., 2001). As shown in this figure, the learning curve that resulted from our method has a lower rms error. Trajectories of the desired output $y(t)$ and the SC-FCMAC model's output are shown in Figures 5(b)-5(d). A comparison analysis of the SC-FCMAC model, the FCMAC model (Lin et al., 2004), and the SCFNN model (Lin et al., 2001) is presented in Table 1. It can be concluded that the proposed model obtains better results than some of the other existing models (Lin et al., 2004; Lin et al., 2001).

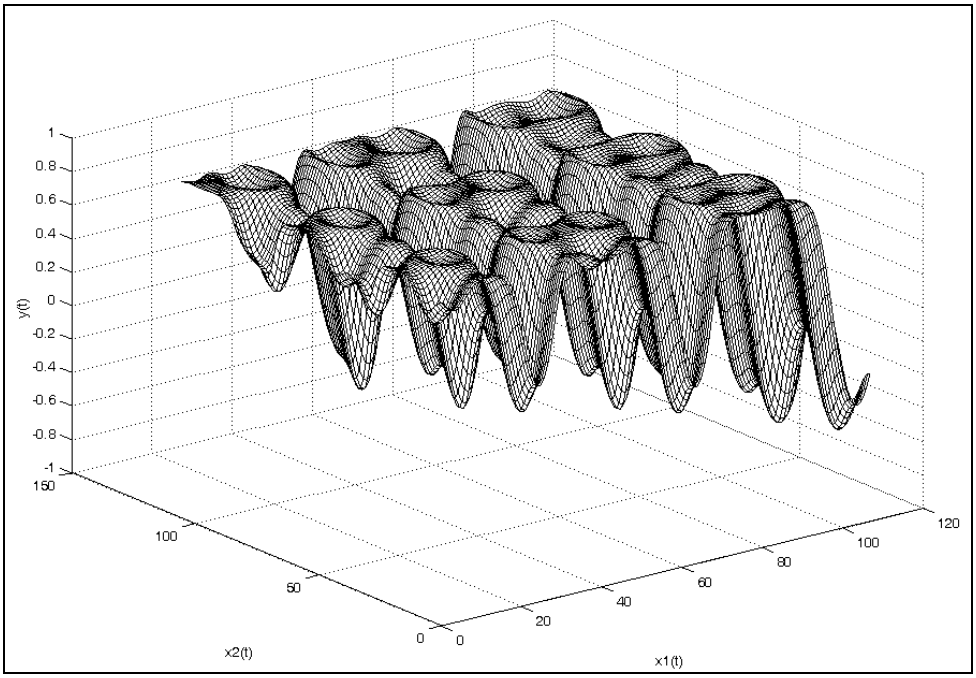
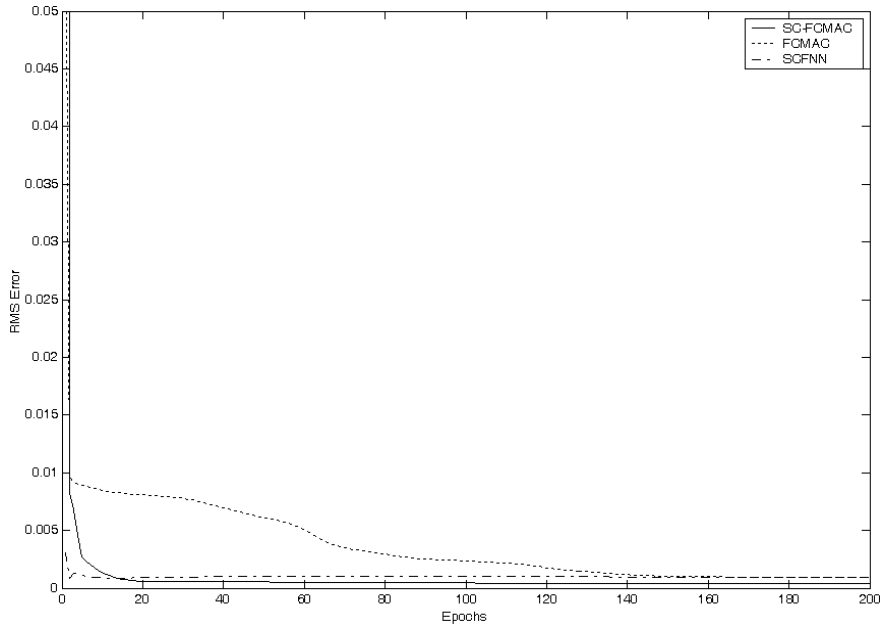
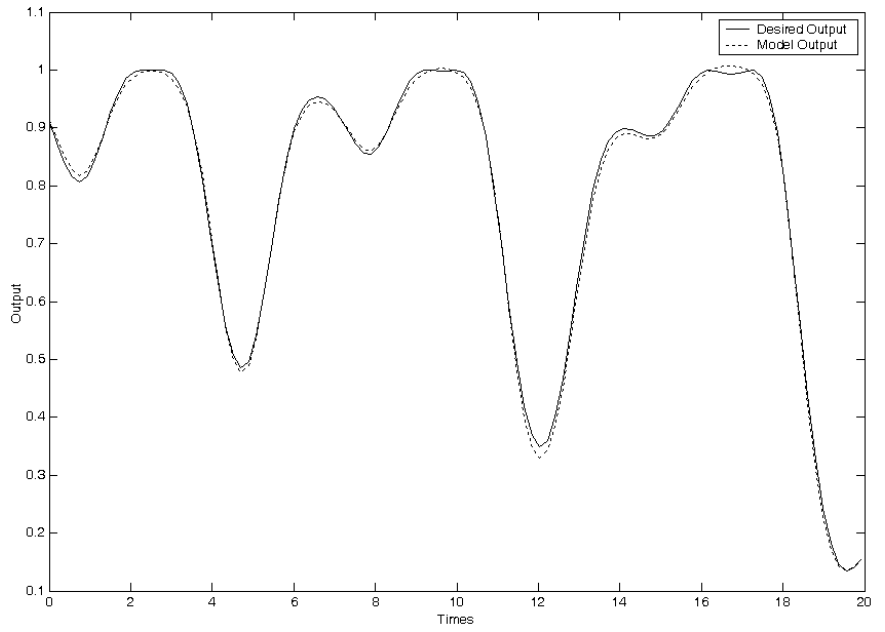


Fig. 4. The nonlinear system: $y(t) = \sin\left(x_1(t) + x_2(t)\right)$, defined by Eqs. (18)-(20).


SC-FCMAC
FCMAC
SCFNN
RMS Error
Epochs
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
20
40
60
80
100
120
140
160
180
200

(a)


Desired Output
Model Output
Output
Times
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8
10
12
14
16
18
20

(b)

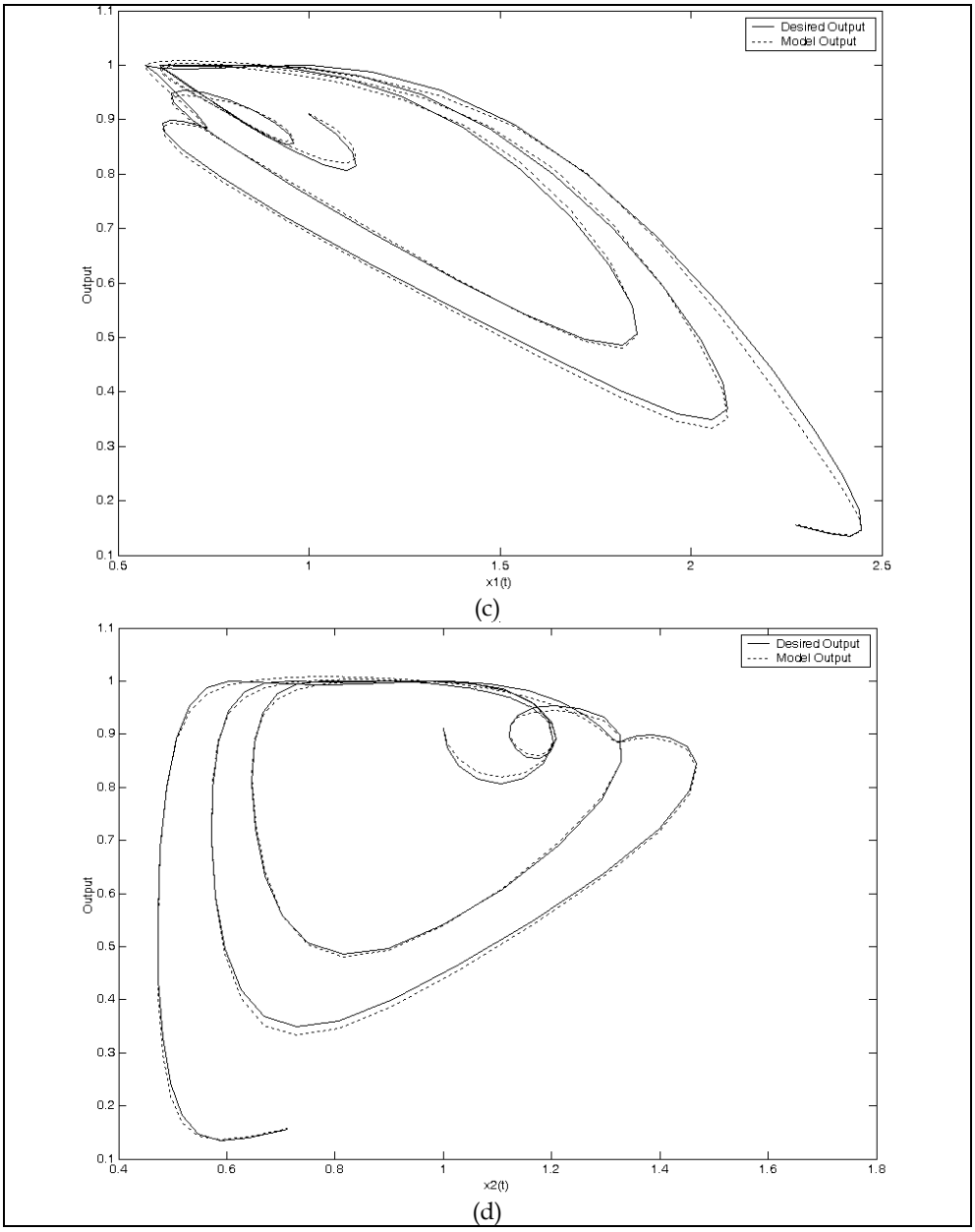


Fig. 5. Simulation results for learning chaotic behaviors. (a) Learning curves of the SC-FCMAC model, the FCMAC model (Lin et al., 2004), and the SCFNN model (Lin et al., 2001). (b) The desired output $y(t)$ and the SC-FCMAC model's output for time t dimension. (c) The desired output $y(t)$ and the SC-FCMAC model's output for $x_1(t)$ dimension. (d) The desired output $y(t)$ and the SC-FCMAC model's output for $x_2(t)$ dimension.

Items \ Models	SC-FCMAC	FCMAC (Lin et al., 2004)	SCFNN (Lin et al., 2001)
Training Steps	200	200	200
Parameters	18	18	20
Hypercube Cells	3	4	N/A
RMS errors	0.000474	0.000885	0.000908

Table 1. Comparisons of the SC-FCMAC model with some existing models for dynamic system identification

4. The parametric fuzzy CMAC (P-FCMAC)

In this section the architecture and learning algorithms of the parametric FCMAC (P-FCMAC, Lin & Lee, 2009) are reviewed, which mainly derived from the traditional CMAC and Takagi-Sugeno-Kang (TSK) parametric fuzzy inference system (Sugeno & Kang, 1988; Takagi & Sugeno, 1985). Since the SCM are inherent in the scheme of input-space partition for the P-FCMAC model, the performance of P-FCMAC is definitional better than the SC-FCMAC. Therefore, another system-identification problem is taken in order to explore the benefit of the P-FCMAC, fully and more fairly.

4.1 Architecture of the P-FCMAC model

As illustrated in Fig. 6, the P-FCMAC model consists of the input space partition, association memory selection, and defuzzification. The P-FCMAC network like the conventional CMAC network that also approximates a nonlinear function $y=f(x)$ by using two primary mappings, $S(x)$ and $P(\alpha)$. These two mappings are realized by fuzzy operations. The function $S(x)$ also maps each point x in the input space onto an association vector $\alpha = S(x) \in A$ that has N_L nonzero elements ($N_L < N_A$). Different from conventional CMAC network, the association vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{N_A})$, where $0 \le \alpha \le 1$ for all components in α, is derived from the composition of the receptive field functions and sensory inputs. Another, several hypercubes is addressed by the input state x that hypercube value is calculated by product operation through the strength of the receptive field functions for each input state. In the P-FCMAC network, we use Gaussian basis function as the receptive field functions and the linear parametric equation of the network input variance as the TSK-type output for learning. Some learned information is stored in the receptive field functions and TSK-type output vectors. A one-dimension Gaussian basis function can be given as defined in Eq. (3). Similar to section 2.2, if a N_D-dimensional problem is considered a Gaussian basis function with N_D dimensions is expressed as Eq. (4) defined.

Each element of the receptive field functions is inferred to produce a partial fuzzy output by applying the value of its corresponding association vector as input matching degree. The partial fuzzy output is defuzzified into a scalar output y by the centroid of area (COA) approach. Then the actual output y is derived as,

$$y = \frac{\sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i \right)}{\sum_{j=1}^{N_L} \alpha_j} \tag{21}$$

The j-th element of the TSK-type output vectors is described as

$$a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i \tag{22}$$

where a_{0j} and a_{ij} denote the scalar value, N_D the number of the input dimensions, N_L the number of hypercube cells, and x_i denotes the ith input dimension. Based on the above structure, a learning algorithm will be proposed to determine the proper network structure and its adjustable parameters.

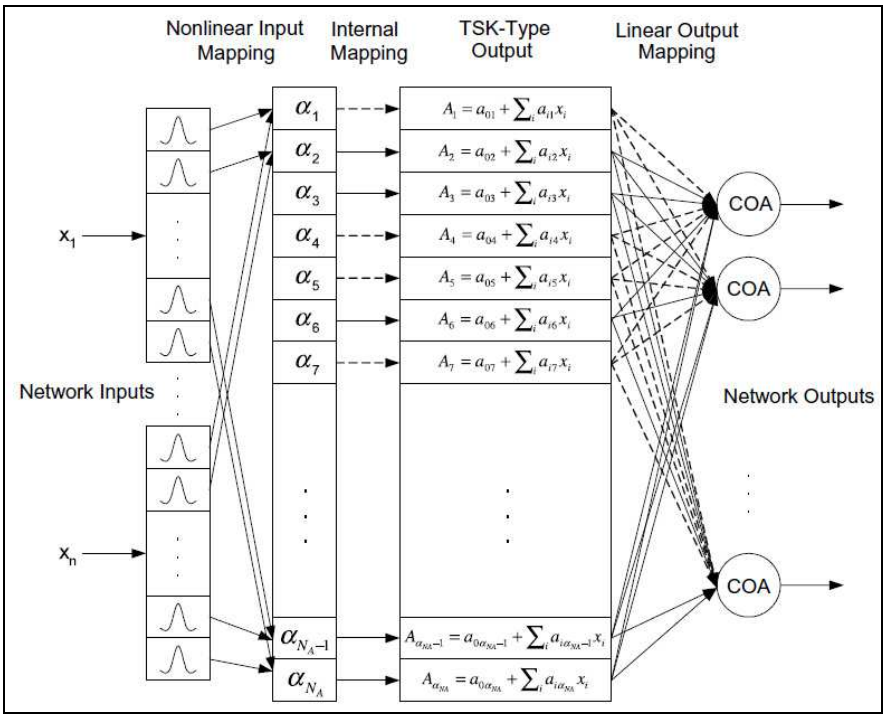


Fig. 6. Structure of the P-FCMAC model (Lin & Lee, 2009)

4.2 Learning algorithm of the P-FCMAC model

Similar to the SC-FCMAC model, the P-FCMAC's learning algorithm consists of an input space partition scheme and a parameter learning scheme. As the same SCM method was applied for input space partition, in the following paragraph the main focus is drawing to the scheme of parameter learning for the P-FCMAC, which is exhibited as in Figure 7.

First, the input space partition scheme (i.e., scheme 1) is used to determine proper input space partitioning and to find the mean and the width of each receptive field function. The input space partition is based on the SCM to appropriately determine the various distributions of the input training data. After the SCM, the number of hypercube cells is

determined. That is, we can obtain the initial m and σ of receptive field functions by using SCM. Second, the parameter learning scheme (i.e., scheme 2) is based on supervised learning algorithms. The gradient descent learning algorithm is used to adjust the free parameters. To minimize a given cost function, the m and σ of the receptive field functions and the parameters a_{0j} and a_{ij} of the TSK-type output vector are adjusted using the backpropagation algorithm. According to the requirements of the system, these parameters will be given proper values to represent the memory information. For the initial system, the values of the tuning parameters a_{0j} and a_{ij} of the element of the TSK-type output vector are generated randomly and the m and σ of receptive field functions are generated by the proposed SCM clustering method.

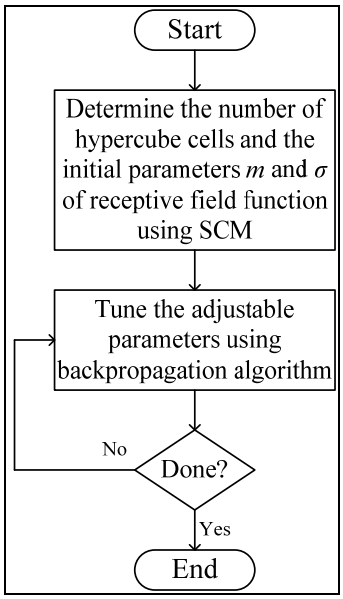


Fig. 7. Flowchart of the P-FCMAC model's learning scheme.

In the parameter learning scheme, there are four parameters need to be tuned, i.e. m_{ij}, σ_{ij}, a_{0j}, and a_{ij}. The total number of tuning parameters for the multi-input single-output P-FCMAC network is $2N_D*N_L+4N_L$, where N_D and N_L denote the number of inputs and hypercube cells, respectively. The parameter learning algorithm of the P-FCMAC network uses the supervised gradient descent method to modify these parameters. When we consider the single output case for clarity, our goal is to minimize the cost function E, defined as in Eq. (9).

Then their parameter learning algorithm, based on backpropagation, is described in detail as follows. The TSK-type outputs are updated according to the following equation:

$$a_{0j}(t+1) = a_{0j}(t) + \Delta a_{0j} \tag{23}$$

$$a_{ij}(t+1) = a_{ij}(t) + \Delta a_{ij} \tag{24}$$

where a_{0j} denotes the proper scalar, a_{ij} the proper scalar coefficient of the i-th input dimension, and j the j-th element of the TSK-type output vector for j=1,2,…,N_L. The elements of the TSK-type output vectors are updated by the amounts

$$\Delta a_{0j} = \eta \cdot e \cdot \frac{\partial y}{\partial a_{0j}} = \eta \cdot e \cdot \frac{\alpha_j}{\sum_{j=1}^{N_L} \alpha_j} \tag{25}$$

and

$$\Delta a_{ij} = \eta \cdot e \cdot \frac{\partial y}{\partial a_{ij}} = \eta \cdot e \cdot \frac{x_i \alpha_j}{\sum_{j=1}^{N_L} \alpha_j} \tag{26}$$

where η is the learning rate, between 0 and 1, and e is the error between the desired output and the actual output, $e = y^d - y$.

The receptive field functions are updated according to the following equation:

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij} \tag{27}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij} \tag{28}$$

where i denotes the i-th input dimension for i=1,2,…,n, m_{ij} the mean of the receptive field functions, and σ_{ij} the variance of the receptive field functions. The parameters of the receptive field functions are updated by the amounts

$$\begin{aligned} \Delta m_{ij} &= \eta \cdot e \cdot \frac{\partial y}{\partial \alpha_j} \cdot \frac{\partial \alpha_j}{\partial m_{ij}} \\ &= \eta \cdot e \cdot \frac{\left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right) \sum_{j=1}^{N_L} \alpha_j - \sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\left(\sum_{j=1}^{N_L} \alpha_j\right)^2} \cdot \alpha_j \cdot \frac{2\left(x_i - m_{ij}\right)}{{\sigma_{ij}}^2} \end{aligned} \tag{29}$$

and

$$\begin{aligned} \Delta \sigma_{ij} &= \eta \cdot e \cdot \frac{\partial y}{\partial \alpha_j} \cdot \frac{\partial \alpha_j}{\partial \sigma_{ij}} \\ &= \eta \cdot e \cdot \frac{\left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right) \sum_{j=1}^{N_L} \alpha_j - \sum_{j=1}^{N_L} \alpha_j \left(a_{0j} + \sum_{i=1}^{N_D} a_{ij} x_i\right)}{\left(\sum_{j=1}^{N_L} \alpha_j\right)^2} \cdot \alpha_j \cdot \frac{2\left(x_i - m_{ij}\right)^2}{{\sigma_{ij}}^3} \end{aligned} \tag{30}$$

where η is the learning rate of the mean and the variance for the receptive field functions.

4.3 An example: Identification of a nonlinear system

In this example, a nonlinear system with an unknown nonlinear function, which is approximated by the P-FCMAC network as shown in Figure 8(b), is a model. First, some of

training data from the unknown function are collected for an off-line initial learning process of the P-FCMAC network. After off-line learning, the trained P-FCMAC network is applied to the nonlinear system to replace the unknown nonlinear function for on-line test.

Consider a nonlinear system in (Wang, 1994) governed by the difference equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \tag{31}$$

We assume that the unknown nonlinear function has the form:

$$g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \tag{32}$$

For off-line learning, twenty-one training data pairs are provided in Table 2 using Eq. (31). The off-line learning configuration of the twenty-one training data points is shown in Figure 12 (a). And the on-line test configuration of the 1000 data points is shown in Figure 12 (b) that using the difference equation is defined as:

$$\hat{y}(k+1) = 0.3y(k) + 0.6y(k-1) + \hat{f}[u(k)] \tag{33}$$

where $\hat{f}[u(k)]$ is the approximated function for $g[u(k)]$ by the P-FCMAC network and $\alpha_0 = 0.3$, and $\alpha_1 = 0.6$. The error is defined as in Eq. (9)

In this example, the initial threshold value in the SCM is 0.15, and the learning rate is η=0.01. After the SCM clustering process, there are eleven hypercube cells generated. Using the first and second parameter learning schemes, the final trained error of the output approximates 0.00057 and 0.00024 after 300 epochs. The numbers of the adjustable parameters of the trained P-FCMAC network are 66.

For on-line testing, we assume that the series-parallel model shown in Figure 8 (b) is driven by $u(k) = \sin(2\pi k / 250)$. The test results of the P-FCMAC network are shown in Fig. 9(a) and (c) for the scheme-1 and scheme-2 methods. The errors between the desired output and the P-FCMAC network output are shown in Figs. 9(b) and (d) for the scheme-1 and scheme-2 methods. The learning curves of the scheme 1 and scheme 2 methods are shown in Fig. 10. Figures 9 can prove that the P-FCMAC network successfully approximates the unknown nonlinear function.

u	$g(u)$	$\hat{f}(u)$	u	$g(u)$	$\hat{f}(u)$
-1.0000	-0.0000	-0.000357	0.1000	0.5281	0.526161
-0.9000	-0.5281	-0.528020	0.2000	0.6380	0.640683
-0.8000	-0.6380	-0.628281	0.3000	0.4781	0.472271
-0.7000	-0.4781	-0.480252	0.4000	0.3943	0.400248
-0.6000	-0.3943	-0.392093	0.5000	0.4000	0.401699
-0.5000	-0.4000	-0.405011	0.6000	0.3943	0.390031
-0.4000	-0.3943	-0.390852	0.7000	0.4781	0.471195
-0.3000	-0.4781	-0.481167	0.8000	0.6380	0.648287
-0.2000	-0.6380	-0.635818	0.9000	0.5281	0.516553
-0.1000	-0.5281	-0.531204	1.0000	0.0000	0.007459
0.0000	0.0000	0.002119			

Table 2. Training data and approximated data obtained using the P-FCMAC network for 300 epochs

Table 3 shows the comparison the learning result among various models. The previous results were taken from (Wan & Li, 2003; Wang et al., 1995; Farag et al., 1998; Juang et al., 2000). The performance of the very compact fuzzy system obtained by the P-FCMAC network is better than all previous works.

Methods	Error	Methods	Error
P-FCMAC	0.00057 (Scheme 1) 0.00024 (Scheme 2)	Gradient Descent (Wang et al., 1995)	0.2841
SGA-SSCP (Wan & Li, 2003)	0.00028	MRDGA (Farag et al., 1998)	0.5221
Symbiotic Evolution (Juang et al., 2000)	0.1997	Genetic Algorithm et al. (Karr 1991)	0.67243

Table 3. Comparison results of the twenty-one training data for off-line learning.

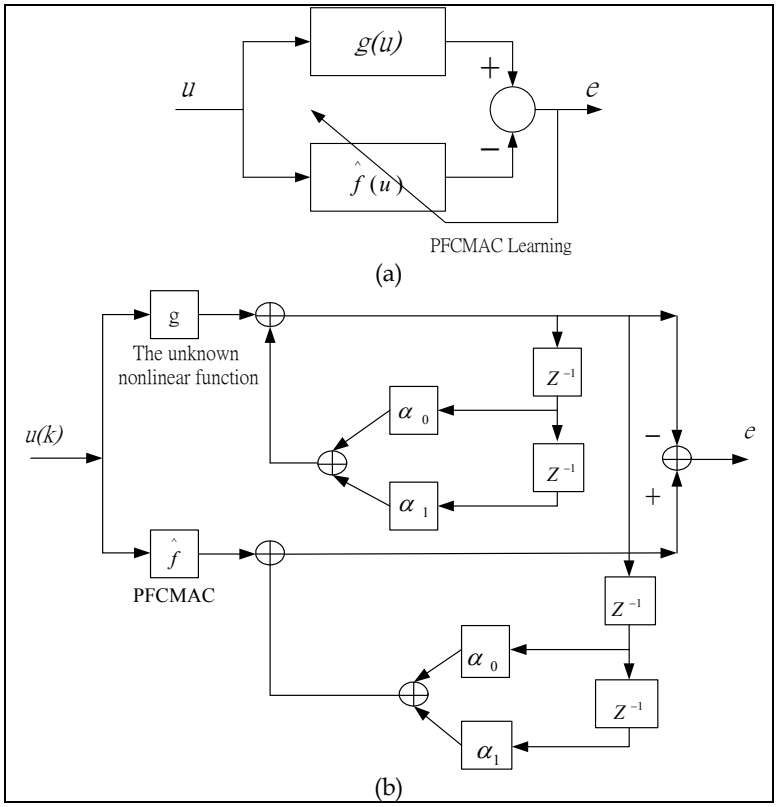


Fig. 8. The series-parallel identification model. (a) Off-line learning by twenty-one training data in Table IV; (b) On-line testing for real $u(k) = \sin(2\pi k / 250)$

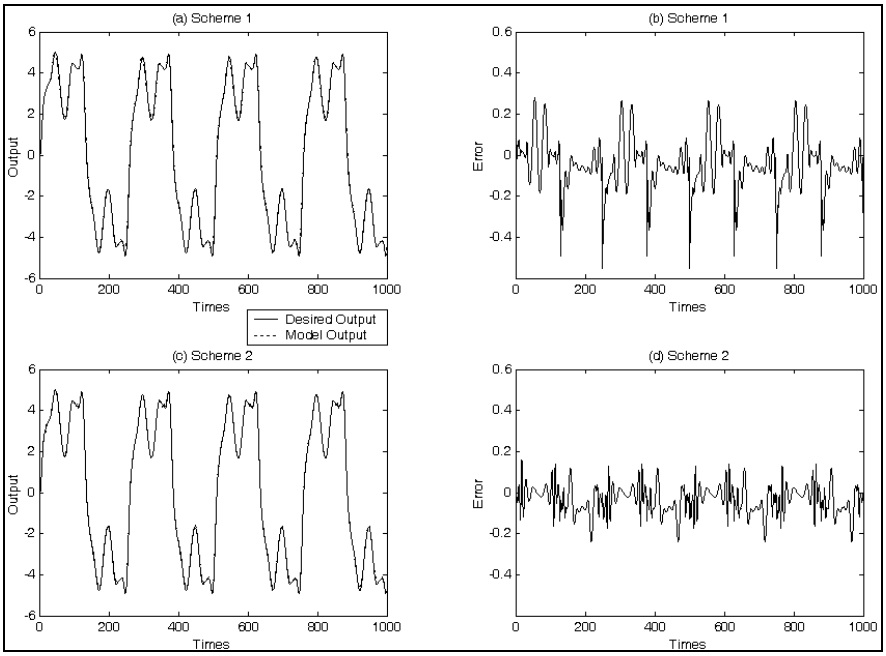


Fig. 9. Comparison of simulation results. (a) Outputs of the nonlinear system (solid line) and the identification model using the proposed network (dotted line) for the scheme 1 method. (b) Identification error of the approximated model for the scheme 1 method. (c) Outputs of the nonlinear system (solid line) and the identification model using the proposed network (dotted line) for the scheme 2 method. (d) Identification error of the approximated model for the scheme 2 method.

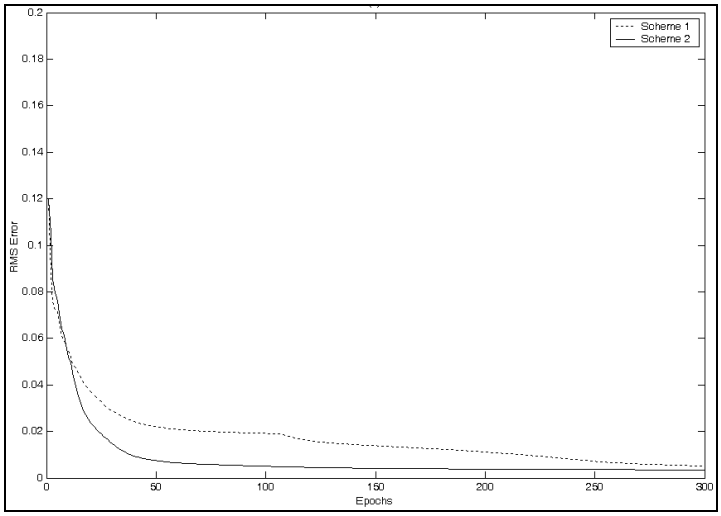


Fig. 10. Learning curves for the scheme 1 and scheme 2 parameter learning methods.

5. Conclusions

In this paper, starting from the discussion of traditional CMAC approach, two novel and latest developed fuzzy CMACs are reviewed. By summarizing the drawbacks of the CMAC model, relative improvement made in the literature have been addressed and presented. Via the exhibited self-constructing FCMAC (SC-FCMAC) and parametric FCMAC (P-FCMAC), not only the inference ability of FCMAC is demonstrated, but also presented the state-of-the art in the field of fuzzy inference systems.

6. References

Albus, J.S. (1975a). A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC). *Transactions of the ASME: Journal of Dynamic Systems Measurement and Control*, Vol. 97, (September 1975), pp. 220-227, ISSN 0022-0434

Albus, J. S. (1975b). Data Storage in the Cerebellar Model Articulation Controller (CMAC). *Transactions of the ASME: Journal of Dynamic Systems Measurement and Control*, Vol. 97, (September 1975), pp. 228-233, ISSN 0022-0434

Chang, P.-L.; Yang, Y.-K.; Shieh, H.-L.; Hsieh, F.-H. & Jeng, M.-D. (2010). Grey Relational Analysis based Approach for CMAC Learning. *International Journal of Innovative Computing, Information and Control*, Vol.6, No.9, (September 2010), pp. 4001-4018, ISSN 1349-4198

Chen, J.Y. (2001). A VSS-type FCMAC Controller, *Proceedings of IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 872-875, ISBN 0-7803-7293-X, December 2-5, 2001

Chow, M.Y. & Menozzi, A. (1994). A Self-organized CMAC Controller, *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 68-72, ISBN 0-7803-1978-8, Guangzhou, China, December 5-9, 1994

Commuri, S. & Lewis, F.L. (1997). CMAC Neural networks for Control of Nonlinear Dynamical Systems: Structure, Stability, and Passivity. *Automatica*, Vol.33, No.4, (April 1997), pp. 635-641, ISSN 0005-1098

Dai, Y.; Liu, L. & Zhao, X. (2010) Modeling Of Nonlinear Parameters on Ship with Fuzzy CMAC Neural Networks, *Proceedings of IEEE International Conference on Information and Automation*, pp. 2070-2075, ISBN 978-1-4244-5701-4, June 20-23, Harbin, China, 2010.

Farag, W.A.; Quintana, V.H. & Lambert-Torres, G. (1998). A Genetic-Based Neuro-fuzzy Approach for Modeling and Control of Dynamical Systems. *IEEE Transactions on Neural Networks*, Vol.9, No.5, (September 1998), pp. 756–767, ISSN 1045-9227

Glanz, F.H.; Miller, W.T. & Graft, L. G. (1991). An Overview of the CMAC Neural Network, *Proceedings of IEEE Conference on Neural Networks for Ocean Engineering,* pp. 301-308, ISBN 0-7803-0205-2, Washington, DC, USA, August 15-17, 1991

Guo, C.; Ye, Z.; Sun, Z.; Sarkar, P. & Jamshidi, M. (2002). A Hybrid Fuzzy Cerebellar Model Articulation Controller Based Autonomous Controller. *Computers & Electrical Engineering: an International Journal*, (January 2002), Vol.28, pp.1-16, ISSN 0045-7906

Harmon, F.G.; Frank, A.A. & Joshi, S.S. (2005). The Control of a Parallel Hybrid-electric Propulsion System for a Small Unmanned Aerial Vehicle Using a CMAC Neural Network. *Neural Networks*, Vol.18, No.5-6, (June-July 2005), pp. 772-780. ISSN 0893-6080

Hung, C.P. & Wang, M.H. (2004). Diagnosis of Incipient Faults in Power Transformers Using CMAC Neural Network Approach. *Electric Power Systems Research*, Vol.71, No.3, (November 2004), pp. 235-244, ISSN 378-7796

Hwang, K.S. & Lin, C.S. (1998). Smooth Trajectory Tracking of Three-link Robot: a Self-Organizing CMAC Approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.28, No.5, (October 1998), pp.680-692, ISSN 1083-4419

Iiguni, Y. (1996). Hierarchical Image Coding via Cerebellar Model Arithmetic Computers. *IEEE Transactions on Image Processing*, Vol.5, No.10, (October 1996), pp. 1393-1401, ISSN 1057-7149

Jou, C.C. (1992). A Fuzzy Cerebellar Model Articulation Controller, *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 1171-1178, ISBN 0-7803-0236-2, San Diego, CA, USA, March 8-12, 1992

Juang, C.F.; Lin, J.Y. & Lin, C.T. (2000). Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.30, No.2, (Apr. 2000), pp. 290-302, ISSN 1083-4419

Karr, C.L. (1991) Design of an Adaptive Fuzzy Logic Controller Using a Genetic Algorithm, *Proceedings of the 4th International Conference on Genetic Algorithms*, ISBN 1-55860-208-9, pp. 450-457, San Diego, CA, USA, July 1991

Kasabov, N.K. & Song, Q. (2002). DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction. *IEEE Transactions on Fuzzy Systems*, Vol.10, No.2, (April 2002), pp. 144-154, ISSN 1063-6706

Ker, J.S.; Hsu, C.C.; Huo, Y.H. & Liu, B.D. (1997). A Fuzzy CMAC Model for Color Reproduction. *Fuzzy Sets and Systems*, Vol.91, No.1, (October 1997), pp.53-68, ISSN 0165-0114

Kolcz, A. & Allinson, N.M. (1994). Application of the CMAC Input Encoding Scheme in the N-tuple Approximation Network, *IEE Proceedings of Computers and Digital Techniques*, Vol.141, No.3, (May 1994), pp. 177-183, ISSN 1350-2387

Kosko, B. (1997). *Fuzzy Engineering*, Prentice-Hall, ISBN 0-1312-4991-6, Englewood Cliffs, NJ.

Lai, H.R. & Wong, C.C. (2001). A Fuzzy CMAC Structure and Learning Method for Function Approximation, *Proceedings of IEEE International Conference on Fuzzy Systems*, Vol.1, pp. 436-439, ISBN 0-7803-7293-X, Melbourne, Australia, December 2-5, 2001

Lane, S. H., Handelman, D. A., & Gelfand, J. J. (1992). Theory and Development of Higher-Order CMAC Neural Networks. *IEEE Control Systems Magazine*, Vol.12, No.2, (April 1992), pp. 23-30, ISSN 1066-033X

Lee, C.-Y. & Lin, C.-J. (2005). A Self-Constructing Fuzzy CMAC Model for Learning Chaotic Behaviors. *GEST International Transactions on Computer Science and Engineering*, Vol.15, No.1, (July 2005), pp. 9-20, ISSN 1738-6438

Lee, C.-Y.; Lin, C.-J. & Chen, H.-J. (2007a). A Self-constructing Fuzzy CMAC Model and Its Applications. *Information Sciences: An International Journal*, Vol.177, No.1, (January 2007), pp. 264-280, ISSN 0020-0255

Lee, C.-Y.; Lin, C.-J. & Hsu, Y.-C. (2007b). A Parametric Fuzzy CMAC Model with Hybrid Evolutionary Learning Algorithms. *Journal of Multiple-Valued Logic and Soft Computing*, Vol.13, No.1-2, (February 2007), pp. 89-114, ISSN 1542-3980

Lee, H.M.; Chen, C.M. & Lu, Y.F. (2003). A Self-Organizing HCMAC Neural-Network Classifier. *IEEE Transactions on Neural Networks*, Vol.14, No.1, (January 2003), pp.15-27, ISSN 1045-9227

Lee, Z.J.; Wang, Y.P. & Su, S.F. (2004). A Genetic Algorithm based Robust Learning Credit Assignment Cerebellar Model Articulation Controller. *Applied Soft Computing,* Vol.4, No.4, (September 2004), pp. 357-367, ISSN 1568-4946

Leu, Y.-G.; Hong, C.-M.; Chen, Z.-R. & Liao, J.-H. (2010). Compact Cerebellar Model Articulation Controller for Ultrasonic Motors. *International Journal of Innovative Computing, Information and Control*, vol.6, No.12, (December 2010), pp. 5539-5552, ISSN 1349-4198

Lin, C.-J. & Lee, C.-Y. (2008). A Self-Organizing Recurrent Fuzzy CMAC Model for Dynamic System Identification. *International Journal of Intelligent Systems*, Vol.23, No.3, (March 2008), pp. 384-396, ISSN 1098-111X

Lin, C.-J. & Lee, C.-Y. (2009). A Novel Parametric Fuzzy CMAC Network and Its Applications. *Journal of Applied Soft Computing*, Vol.9, No.2, (March 2009), pp. 775-785, ISSN 1568-4946

Lin, C.-J.; Lee, C.-H. & Lee, C.-Y. (2008). A Novel Hybrid Learning Algorithm for Parametric Fuzzy CMAC Networks and Its Classification Applications. *Expert Systems with Applications*, Vol.35, No.4, (Nov. 2008), pp. 1711-1720, ISSN 0957-4174

Mese, E. (2003). A Rotor Position Estimator for Switched Reluctance Motors using CMAC. *Energy Conversion and Management*, Vol.44, No.8, (May 2003), pp. 1229-1245, ISSN 0196-8904

Miller, W.T.; Hewes, R.P.; Glanz, F.H. & Graft, L.G. (1990). Real-time Dynamic Control of an Industrial Manipulator using a Neural-Network Based Learning Controller. *IEEE Transactions on Robotics and Automation*, Vol.6, No.1, (February 1990), pp. 1-9, ISSN 1042-296X

Mohajeri, K.; Pishehvar, G. & Seifi, M. (2009a). CMAC Neural Networks Structures, *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 39-45, ISBN 978-1-4244-4808-1, Daejeon, December 15-18, 2009

Mohajeri, K.; Zakizadeh, M.; Moaveni, B. & Teshnehlab, M. (2009b). Fuzzy CMAC Structures, *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 2126-2131, ISBN 978-1-4244-3596-8, August 20-24, Jeju Island, Korea, 2009

Paul, S. & Kumar, S. (2002). Subsethood-Product Fuzzy Neural Inference System (SuPFuNIS). *IEEE Transactions on Neural Networks*, Vol.13, No.3, (May 2002), pp. 578-599, ISSN 1045-9227

Peng, C.-C. & Lin, C.-J. (2011). A Self-Organizing Fuzzy CMAC Model for Classification Applications. *Innovative Computing, Information and Control Express Letters*, Vol.5, No.7, (July 2011), pp. 2389-2394, ISSN 1881-803X

Sugeno, M. & Kang, G.T. (1988). Structure Identification of a Fuzzy Model. *Fuzzy Sets and Systems*, Vol.28, No.1, (October 1988), pp. 15-33, ISSN 0165-0114

Takagi, T. & Segeno, M. (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.SMC-15, (January/Feburary 1985), pp. 116-132, ISSN 1083-4419

Tung, W.L. & Quek, C. (2002). GenSoFNN: A Generic Self-Organizing Fuzzy Neural Network. *IEEE Transactions on Neural Networks,* Vol.13, No.5, (September 2002), pp. 1075-1086, ISSN 1045-9227

Wan, W.Y. & Li, Y.H. (2003). Evolutionary Learning of BMF Fuzzy-Neural Networks Using a Reduced-Form Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.33, No.6, (December 2003), pp. 966–976, ISSN 1083-4419

Wang, C.H.; Wang, W.Y.; Lee, T.T. & Tseng, P.S. (1995). Fuzzy B-Spline membership Function (BMF) and Its Applications in Fuzzy-Neural Control. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.25, No.5, (May 1995), pp. 841–851, ISSN 0018-9472

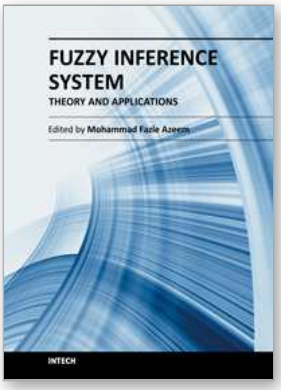
Wang, D.-Y.; Lin, C.-J. & Lee, C.-Y. (2008). A New Pseudo-Gaussian-Based Recurrent Fuzzy CMAC Model for Dynamic Systems Processing. *International Journal of Systems Science*, Vol.39, No.3, (March 2008), pp. 289-304, ISSN 0020-7721

Wang, L.X. (1994). *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice-Hall, ISBN 0-1309-9631-9, Englewood Cliffs, NJ.

Wang, S. & Jiang, Z. (2004). Valve Fault Detection and Diagnosis Based on CMAC Neural Networks. *Energy and Buildings*, Vol.36, No.6, (June 2004), pp. 599-610, ISSN 0378-7788

Wu, J. & Pratt, F. (1999). Self-Organizing CMAC Neural Networks and Adaptive Dynamic Control, *Proceedings of IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pp. 259-265, ISBN 0-7803-5665-9, Cambridge, MA, USA, September 15-17, 1999

Zhang, K. & Qian, F. (2000). Fuzzy CMAC and Its Application, *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, pp. 944-947, ISBN 0-7803-5995-X, Hefei, China, June 28-July 2, 2000



Fuzzy Inference System - Theory and Applications

Edited by Dr. Mohammad Fazle Azeem

ISBN 978-953-51-0525-1
Hard cover, 504 pages
Publisher InTech
Published online 09, May, 2012
Published in print edition May, 2012

This book is an attempt to accumulate the researches on diverse inter disciplinary field of engineering and management using Fuzzy Inference System (FIS). The book is organized in seven sections with twenty two chapters, covering a wide range of applications. Section I, caters theoretical aspects of FIS in chapter one. Section II, dealing with FIS applications to management related problems and consisting three chapters. Section III, accumulates six chapters to commemorate FIS application to mechanical and industrial engineering problems. Section IV, elaborates FIS application to image processing and cognition problems encompassing four chapters. Section V, describes FIS application to various power system engineering problem in three chapters. Section VI highlights the FIS application to system modeling and control problems and constitutes three chapters. Section VII accommodates two chapters and presents FIS application to civil engineering problem.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821