World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:6, 2011

Applications of Prediction and Identification Using Adaptive DCMAC Neural Networks

Yu-Lin Liao and Ya-Fu Peng

***Abstract*—**An adaptive dynamic cerebellar model articulation controller (DCMAC) neural network used for solving the prediction and identification problem is proposed in this paper. The proposed DCMAC has superior capability to the conventional cerebellar model articulation controller (CMAC) neural network in efficient learning mechanism, guaranteed system stability and dynamic response. The recurrent network is embedded in the DCMAC by adding feedback connections in the association memory space so that the DCMAC captures the dynamic response, where the feedback units act as memory elements. The dynamic gradient descent method is adopted to adjust DCMAC parameters on-line. Moreover, the analytical method based on a Lyapunov function is proposed to determine the learning-rates of DCMAC so that the variable optimal learning-rates are derived to achieve most rapid convergence of identifying error. Finally, the adaptive DCMAC is applied in two computer simulations. Simulation results show that accurate identifying response and superior dynamic performance can be obtained because of the powerful on-line learning capability of the proposed DCMAC.

***Keywords*—**adaptive, cerebellar model articulation controller, CMAC, prediction, identification

I. Introduction

RECENTYL, many researches have been done on the applications of neural networks (NNs) for prediction, identification and control of dynamic systems [1]-[5]. The most useful property of NNs is their ability to approximate arbitrary linear or nonlinear mapping through learning. Based on their approximation ability, the NNs have been used for approximation of control system dynamics or controllers. According to the structure, the NNs can be mainly classified as feedforward neural networks (FNNs) [2], [3] and recurrent neural networks (RNNs) [4], [5]. RNN has capabilities superior to FNN, such as the dynamic response and information storing ability [4], [5]. Since a RNN has an internal feedback loop, it captures the dynamic response of system with external feedback through delays. Thus, the RNN is a dynamic mapping and demonstrates good control performance in presence of unmodelled dynamics. However, no matter FNNs or RNNs, the learning is slow since all the weights are updated during each learning cycle. Therefore, the effectiveness of NN is limited in problems requiring on-line learning.

The cerebellar model articulation controller (CMAC) has been adopted widely for the closed-loop control of complex dynamical systems owing to its fast learning property, good generalization capability, and simple computation compared with the multiplayer perceptron with backpropagation algorithm [6]-[8]. The CMAC is a non-fully connected perceptron-like associative memory network with overlapping receptive-fields. The application of CMAC is not only limited to control problem but also to model-free function approximation. This network has been already validated that it can approximate a nonlinear function over a domain of interest to any desired accuracy. The advantages of using CMAC over conventional NN in many practical applications have been presented in recent literatures [7]-[10]. However, the major drawback of existing CMACs is that their application domain is limited to static problem.

To accomplish the mentioned motivation, a dynamic cerebellar model articulation controller (DCMAC) neural network is proposed in this study used for solving the prediction and identification problem. The DCMAC comprises the delayed self-recurrent units in the association memory space. Thus it captures the dynamic response. The DCMAC parameters are on-line tuned by the derived adaptive laws. Moreover, the analytical method based on a Lyapunov function is proposed to determine the learning-rates of DCMAC so that the variable optimal learning-rates are derived to achieve most rapid convergence of identifying error. Finally, simulation results show that accurate identifying response and superior dynamic performance can be obtained because of the powerful on-line learning capability of the proposed DCMAC.

II. Structure of the DCMAC Neural Network

A. Description of DCMAC

A dynamic cerebellar model articulation controller (DCMAC) is proposed and shown in Fig. 1, in which z^{-1} denotes a time delay. This DCMAC is composed of input space, association memory space, recurrent unit, receptive-field space, weight memory space, output space and recurrent weights. The signal propagation and the basic function in each space are introduced as follows.

Y.-L. Liao is with the Department of Business Administration, Ching-Yun University, Taoyuan 320, Taiwan, R.O.C. (phone: 886-3-4581196 ext 7107; fax:886-3-4684014; e-mail: celine@mail.cyu.edu.tw).

Y.-F. Peng is with the Department of Electrical Engineering, Ching-Yun University Taoyuan 320, Taiwan, R.O.C. (phone: 886-3-4581196 ext 5337; fax:886-3-4594937; e-mail: yfpeng@cyu.edu.tw).

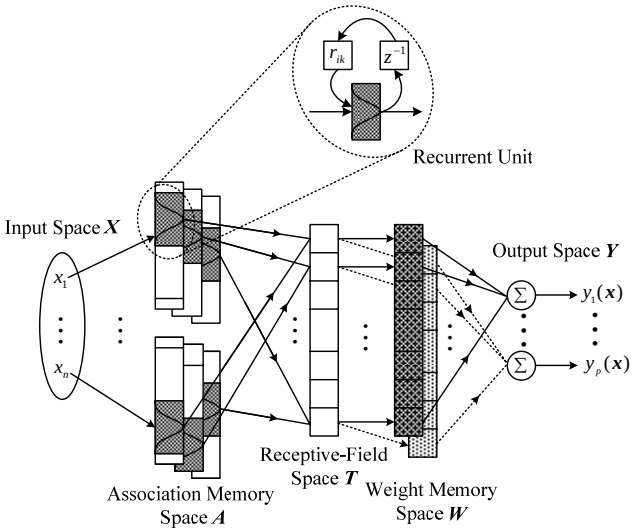

Fig. 1 Architecture of DCMAC.

*1) Input space **X**:* For a given $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \in \Re^n$, each input state variable x_i must be quantized into discrete regions (called *elements*) according to given control space. The number of elements, n_E, is termed as a resolution.

*2) Association memory space **A**:* Several elements can be accumulated as a *block*, the number of blocks, n_B, in DCMAC is usually greater than two. The ***A*** denotes an association memory space with n_A ($n_A = n \times n_B$) components. In this space, each block performs a *receptive-field basis function*, which can be defined as rectangular [6] or triangular or any continuously bounded function (e.g., Gaussian [11], [12] or B-spline [7], [13]). The Gaussian function is adopted here as the receptive-field basis function, which can be represented as

$$\phi_{ik} = \exp\left[\frac{-(x_{ri} - m_{ik})^2}{v_{ik}^2}\right], \quad \text{for } k = 1, 2, \cdots n_B \tag{1}$$

where ϕ_{ik} represents the *k*th block of the *i*th input x_i with the mean m_{ik} and variance v_{ik}. In addition, the input of this block for discrete time *N* can be represented as

$$x_{ri}(N) = x_i(N) + r_{ik}\, \phi_{ik}(N-1) \tag{2}$$

where r_{ik} is the recurrent weight of the recurrent unit. It is clear that the input of this block contains the memory terms $\phi_{ik}(N-1)$, which store the past information of the network. This is the apparent difference between the proposed DCMAC and the conventional CMAC. Figure 2 depicts the schematic diagram of two-dimensional DCMAC operations with $n_E = 9$ and $\rho = 4$ (ρ is the number of elements in a complete block), where x_1 is divided into blocks *A*, *B* and *C*, and x_2 is divided into blocks *a*, *b* and *c*. By shifting each variable an element, different blocks can be obtained. For instance, blocks *D*, *E* and *G* for x_1, and blocks *d*, *e* and *g* for x_2 are possible shifted elements. Each block in this space has three adjustable parameters m_{ik}, v_{ik} and r_{ik}.

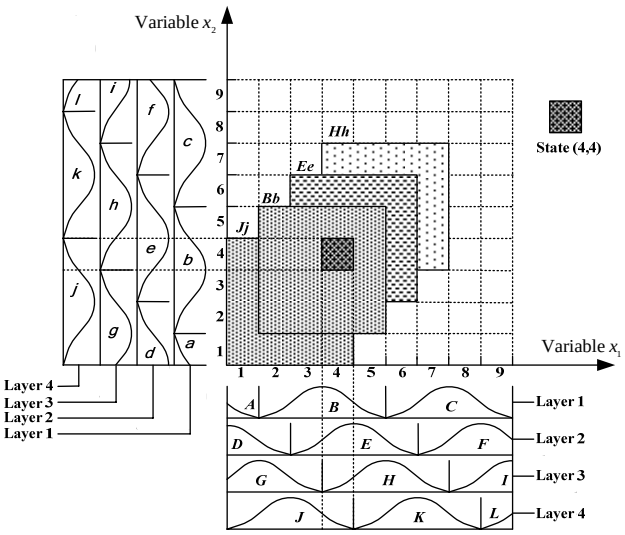

Fig. 2. Two-dimensional DCMAC with $\rho = 4$ and $n_E = 9$.

*3) Receptive-field space **T**:* Areas formed by blocks, named as *Aa*, *Bb* and *Cc* are called *receptive-fields*. The number of receptive-field, n_R, is equal to n_B in this study. Each location of ***T*** corresponds to a wavelet receptive-field. The *multidimensional wavelet receptive-field function* is defined as

$$b_k(\boldsymbol{x}_r, \boldsymbol{m}_k, \boldsymbol{v}_k) = \prod_{i=1}^{n} \phi_{ik} = \exp\left[-\left(\sum_{i=1}^{n} \frac{(x_{ri} - m_{ik})^2}{v_{ik}^2}\right)\right] \tag{3}$$

where b_k is associated with the *k*th wavelet receptive-field, $\boldsymbol{x}_r = [x_{r1}, x_{r2}, \cdots, x_{rn}]^T \in \Re^n$, $\boldsymbol{m}_k = [m_{1k}, m_{2k}, \cdots, m_{nk}]^T \in \Re^n$ and $\boldsymbol{v}_k = [v_{1k}, v_{2k}, \cdots, v_{nk}]^T \in \Re^n$. The multidimensional wavelet receptive-field function can be expressed in a vector form as

$$\Gamma(\boldsymbol{x}_r, \boldsymbol{m}, \boldsymbol{v}) = [b_1, b_2, \cdots, b_k, \cdots, b_{n_R}]^T \tag{4}$$

where $\boldsymbol{m} = [\boldsymbol{m}_1^T, \boldsymbol{m}_2^T, \cdots, \boldsymbol{m}_k^T, \cdots, \boldsymbol{m}_{n_R}^T]^T \in \Re^{nn_R}$ and $\boldsymbol{v} = [\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \cdots, \boldsymbol{v}_k^T, \cdots, \boldsymbol{v}_{n_R}^T]^T \in \Re^{nn_R}$. In the DCMAC scheme, no receptive-field is formed by the combination of different layers such as "*A*, *B*, *C* " and "*d*, *e*, *f* ". Therefore, *Dd*, *Ff* and *Gg* are new receptive-fields resulting from different blocks (see Fig. 2). With this kind of quantization and receptive-field composition, each state is covered by ρ different receptive-fields.

*4) Weight memory space **W**:* Each location of ***T*** to a particular adjustable value in the weight memory space can be expressed as

$$\boldsymbol{w} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_o, \cdots, \boldsymbol{w}_p] = \begin{bmatrix} w_{11} & \cdots & w_{1o} & \cdots & w_{1p} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ w_{k1} & \cdots & w_{ko} & \cdots & w_{kp} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n_R 1} & \cdots & w_{n_R o} & \cdots & w_{n_R p} \end{bmatrix}$$

where $\boldsymbol{w}_o = [w_{1o}, w_{2o}, \cdots, w_{ko}, \cdots, w_{n_R o}]^T \in \Re^{n_R}$, and w_{ko} denotes the connecting weight value of the *o*th output associated with the *k*th wavelet receptive-field. The weight w_{ko} is initialized from zero and is automatically adjusted during on-line operation.

*5) Output space **Y**:* The output of DCMAC is the algebraic sum of the activated weights in the weight memory space, and is expressed as

$$y_o = \boldsymbol{w}_o^T \boldsymbol{\Gamma}(\boldsymbol{x}_r, \boldsymbol{m}, \boldsymbol{v}) = \sum_{k=1}^{n_R} w_{ko}\, b_k(\boldsymbol{x}_r, \boldsymbol{m}_k, \boldsymbol{v}_k) \tag{5}$$

The outputs of the DCMAC can be expressed in a vector notation as

$$\boldsymbol{y} = [y_1, y_2, \cdots, y_o, \cdots, y_p]^T = \boldsymbol{w}^T \boldsymbol{\Gamma} \tag{6}$$

In a two-dimensional case shown in Fig. 2, the output of the DCMAC is the sum of the value in receptive-fields *Bb*, *Ee*, *Hh* and *Jj*, where the input state is (4,4). The architecture of DCMAC used in this paper is designed to possess the advantage of simple structure with dynamic characteristics.

B. Online Training Methodology

Selections of the recurrent weights and translations and dilations of the mother wavelet functions will significantly affect the performance of DCMAC. Inappropriate recurrent weights and wavelet functions will degrade the DCMAC learning performance. For achieving effective learning, an on-line learning algorithm, which is derived using the supervised gradient descent method, is introduced for DCMAC so that it can real-time adjust the recurrent weights and translations and dilations of the wavelet functions. Define the cost function E as

$$E(N) = \frac{1}{2}[y_d(N) - y_o(N)]^2 = \frac{1}{2} e_m^2(N) \tag{7}$$

where y_d denotes the desired output value. In the output space, the learning algorithm based on gradient descent method for w_k, can be derived as

$$\Delta w_k(N) = -\eta_w \frac{\partial E}{\partial w_k} = -\eta_w \frac{\partial E}{\partial y_o}\frac{\partial y_o}{\partial w_k} = \eta_w e_m(N) b_k \tag{8}$$

where the positive factor η_w is the learning-rate for the output weights w_k. The connective weights can be updated according to the following equation:

$$w_k(N+1) = w_k(N) + \Delta w_k(N) \tag{9}$$

Moreover, the translations, dilations and recurrent weights of the wavelet functions can be also adjusted in the following equation:

$$\begin{aligned}\Delta m_{ik}(N) &= -\eta_m \frac{\partial E}{\partial m_{ik}} = -\eta_m \frac{\partial E}{\partial y_o}\frac{\partial y_o}{\partial b_k}\frac{\partial b_k}{\partial \phi_{ik}}\frac{\partial \phi_{ik}}{\partial m_{ik}} \\ &= \eta_m e_m(N) w_k b_k \frac{2(x_{ri} - m_{ik})}{v_{ik}^2}\end{aligned} \tag{10}$$

$$\begin{aligned}\Delta v_{ik}(N) &= -\eta_v \frac{\partial E}{\partial v_{ik}} = -\eta_v \frac{\partial E}{\partial y_o}\frac{\partial y_o}{\partial b_k}\frac{\partial b_k}{\partial \phi_{ik}}\frac{\partial \phi_{ik}}{\partial v_{ik}} \\ &= \eta_v e_m(N) w_k b_k \frac{2(x_{ri} - m_{ik})^2}{v_{ik}^3}\end{aligned} \tag{11}$$

$$\begin{aligned}\Delta r_{ik}(N) &= -\eta_r \frac{\partial E}{\partial r_{ik}} = -\eta_r \frac{\partial E}{\partial y_o}\frac{\partial y_o}{\partial b_k}\frac{\partial b_k}{\partial \phi_{ik}}\frac{\partial \phi_{ik}}{\partial x_{ri}}\frac{\partial x_{ri}}{\partial r_{ik}} \\ &= \eta_r e_m(N) w_k b_k \frac{2(m_{ik} - x_{ri})}{v_{ik}^2}\phi_{ik}(N-1)\end{aligned} \tag{12}$$

where the positive factors η_m, η_v and η_r are the learning-rates for the translations, dilations and recurrent weights, respectively. Then the updated laws of translations, dilations and recurrent weights are given as follows:

$$m_{ik}(N+1) = m_{ik}(N) + \Delta m_{ik}(N) \tag{13}$$

$$v_{ik}(N+1) = v_{ik}(N) + \Delta v_{ik}(N) \tag{14}$$

$$r_{ik}(N+1) = r_{ik}(N) + \Delta r_{ik}(N) \tag{15}$$

To avoid the local minimum result and increase the convergence rate, the corresponding time-varying learning rates should be designed.

C. Convergence Analyses

The learning laws of (8), (10), (11) and (12) call for a proper choice of the learning-rates η_w, η_m, η_v and η_r. For a small value of learning-rates, the convergence is easy to be guaranteed; however, the learning speed is slow. On the other hand if learning-rates are too large, the learning mechanism may become unstable. In order to train the DCMAC effectively, the variable learning-rates, which guarantee convergence of the output error, are derived in the following. The convergence analyses in this study are to derive specific learning-rates for specific types of network parameters to assure convergence of the output error [5].

Theorem 1: Let η_s be the learning-rates for the DCMAC and let $\partial y_o / \partial \boldsymbol{s} = \boldsymbol{P}_s(N)$ for $s = w$, m, v and r. Then the convergence of identifying error is guaranteed if η_s is chosen as

$$0 < \eta_s < \frac{2}{\| \boldsymbol{P}_s(N) \|^2} \tag{16}$$

where $\|\cdot\|$ is the Euclidean norm. Moreover, the variable optimal-rates which achieve the most rapid convergence can be obtained as

$$\eta_s^* = \frac{1}{\| \boldsymbol{P}_s(N) \|^2} \tag{17}$$

Proof : Since

$\boldsymbol{P}_s(N) = \dfrac{\partial y_o}{\partial \boldsymbol{s}}$ for $s = w$, m, v and r, it reveals that

$$\boldsymbol{P}_w(N) = \frac{\partial y_o}{\partial \boldsymbol{w}} = [\frac{\partial y_o}{\partial w_1}, \cdots, \frac{\partial y_o}{\partial w_k}, \cdots, \frac{\partial y_o}{\partial w_{n_R}}]^T \tag{18}$$

$$\boldsymbol{P}_m(N) = \frac{\partial y_o}{\partial \boldsymbol{m}} = [\frac{\partial y_o}{\partial m_{11}}, \cdots, \frac{\partial y_o}{\partial m_{ik}}, \cdots, \frac{\partial y_o}{\partial m_{nn_R}}]^T \tag{19}$$

$$\boldsymbol{P}_v(N) = \frac{\partial y_o}{\partial \boldsymbol{v}} = [\frac{\partial y_o}{\partial v_{11}}, \cdots, \frac{\partial y_o}{\partial v_{ik}}, \cdots, \frac{\partial y_o}{\partial v_{nn_R}}]^T \tag{20}$$

$$\boldsymbol{P}_r(N) = \frac{\partial y_o}{\partial \boldsymbol{r}} = [\frac{\partial y_o}{\partial r_{11}}, \cdots, \frac{\partial y_o}{\partial r_{ik}}, \cdots, \frac{\partial y_o}{\partial r_{nn_R}}]^T \tag{21}$$

where $\dfrac{\partial y_o}{\partial w_k} = b_k$, $\dfrac{\partial y_o}{\partial m_{ik}} = w_k b_k \dfrac{2(x_{ri} - m_{ik})}{v_{ik}^2}$, $\dfrac{\partial y_o}{\partial v_{ik}} = w_k b_k \dfrac{2(x_{ri} - m_{ik})^2}{v_{ik}^3}$ and $\dfrac{\partial y_o}{\partial r_{ik}} = w_k b_k \dfrac{2(m_{ik} - x_{ri})}{v_{ik}^2} \phi_{ik}(N-1)$.

Define Lyapunov function as

$$V(N) = \frac{1}{2} e_m^2(N) \tag{22}$$

then change of the Lyapunov function is obtained as

$$\Delta V(N) = V(N+1) - V(N) = \frac{1}{2}[e_m^2(N+1) - e_m^2(N)] \tag{23}$$

The error difference can be represented by

$$e_m(N+1) = e_m(N) + \Delta e_m(N) = e_m(N) + [\frac{\partial e_m(N)}{\partial \boldsymbol{s}}]^T \Delta \boldsymbol{s} \tag{24}$$

it is obtained that

$$\frac{\partial e_m}{\partial \boldsymbol{s}} = \frac{\partial e_m}{\partial y_o} \frac{\partial y_o}{\partial \boldsymbol{s}} = -\boldsymbol{P}_s(N) \tag{25}$$

Thus

$$\begin{aligned} e_m(N+1) &= e_m(N) - [\boldsymbol{P}_s(N)]^T \eta_s e_m(N) \boldsymbol{P}_s(N) \\ &= e_m(N)[1 - \eta_s \| \boldsymbol{P}_s(N) \|^2] \end{aligned} \tag{26}$$

From (23) and (26), $\Delta V(N)$ can be represented as

$$\Delta V(N) = \frac{1}{2} \eta_s e_m^2(N) \| \boldsymbol{P}_s(N) \|^2 [\eta_s \| \boldsymbol{P}_s(N) \|^2 - 2] \tag{27}$$

Remark: If η_s is chosen as $0 < \eta_s < \dfrac{2}{\| \boldsymbol{P}_s(N) \|^2}$, $\Delta V(N)$ in (27) is less than 0. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ is guaranteed. Thus, the tracking error $e_m(N)$ will converge to zero as $t \to \infty$. Moreover, the optimal learning-rates which achieve the most rapid convergence are corresponding to $2\eta_s^* \| \boldsymbol{P}_s(N) \|^2 - 2 = 0$, i.e., $\eta_s^* = \dfrac{1}{\| \boldsymbol{P}_s(N) \|^2}$, which comes from the derivative of (27) with respect to η_s and equals to zero. This shows an interesting result for the variable optimal learning-rates which can be on-line adjusted at each instant.

III. Illustrative Examples

To demonstrate the performance of the proposed DCMAC for temporal problems, this section presents two examples and performance contrasts with cerebellar model articulation controller (CMAC, non-recurrent unit, i.e., $r_{ik} = 0$). The first example involves predicting a time sequence and the second example involves identifying a nonlinear dynamic system.

A. Example 1: Prediction of Time Sequence

To clearly verify that the proposed adaptive DCMAC can learn the temporal relationship, a simple time sequence prediction problem from [14], [15] is used for testing in the following example. The test bed used for this example is shown in Fig. 3. The test bed is an "8" shape made up of a series with 12 points that are to be presented to the network in the order shown.

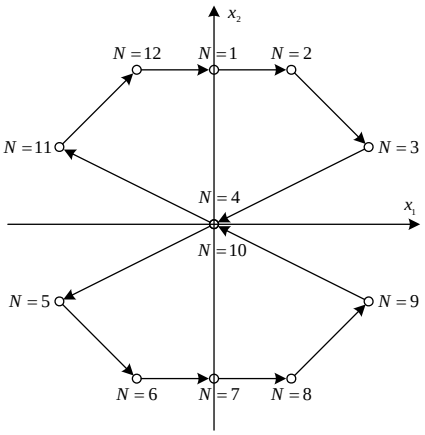

Fig. 3 Test bed for the sample prediction experiment in Example 1.

The adaptive DCMAC is asked to predict the succeeding point for every presented point. Obviously, this task cannot be accomplished by a static network because the point at coordinate has two successors: point 5 and point 11. The DCMAC must determine the successor of (0, 0) based on its predecessor; specifically, if the predecessor is 3, then the successor is 5, whereas if the predecessor is 9, the successor is 11.

In this example, the DCMAC contains only 2 input nodes, which were activated with the two dimensional coordinate of the current point, and two output nodes, representing the two dimensional coordinates of the predicted point. The training process was continued for 1000 epochs. The predicted values are shown in Fig. 4(c) (solid line: desired output; dotted line: DCMAC). We also applied the CMAC to this time prediction problem. The results of prediction using the CMAC with 2

inputs after training are shown in Fig. 4(a), verifying that a feedforward CMAC cannot accurate predict owing to its static mapping. To solve the problem using the feedforward CMAC above, 4 inputs must be fed into the network. Fig. 4(b) shows that the CMAC with 4 inputs can make effective predictions, but some time prediction points cannot be matched exactly. Fig. 4(d) shows the mean square error (MSE) for the DCMAC, CMAC with two inputs and CMAC with four inputs. From the simulation results shown in Fig. 4(d), we can see that the CMAC is inappropriate for time sequence prediction because of its static mapping.

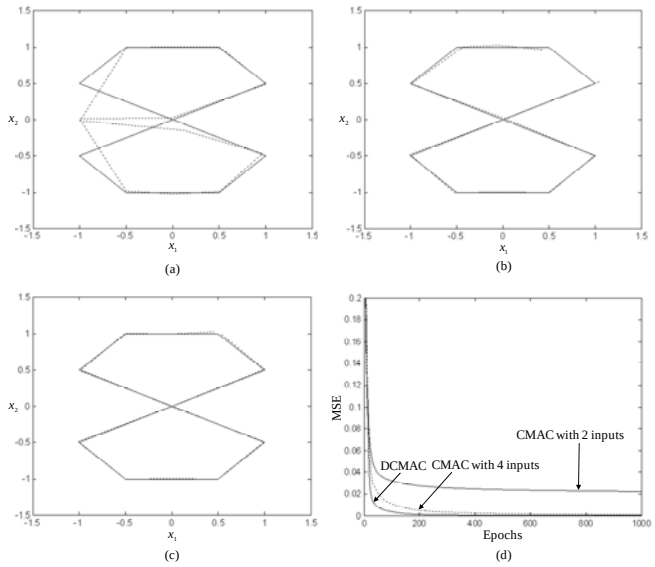

Fig. 4. Simulation results of time sequence prediction (solid line: desired output). (a) Results of prediction using the CMAC with 2 inputs. (b) Results of prediction using the CMAC with 4 inputs.. (c) Results of prediction using the DCMAC (d) MSE of the DCMAC, CMAC with 2 inputs and CMAC with 4 inputs.

B. Example 2: Identification of a nonlinear dynamic system

In this example, the nonlinear plant with multiple time-delay is described as [15], [16].

$$y_d(N+1) = f(y_d(N), y_d(N-1), y_d(N-2), u(N), u(N-1))$$

where $f(x_1, x_2, x_3, x_4, x_5) = \dfrac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2}$

Here, the current output of the plant depends on three previous outputs and two previous inputs. In [15] and [16], the FNN, with five input nodes for feeding the appropriate past values of y_d and u were used. In this paper, only two values, $y_d(N)$ and $u(N)$, are fed into the adaptive DCMAC to determine the output. In training the adaptive DCMAC, we used 100 epochs. The testing input signal $u(N)$ as the following equation is used to determine the identification results,

$$u(N) = \begin{cases} \sin\left(\dfrac{\pi N}{25}\right), & 0 < N < 250 \\ 1, & 250 \le N < 500 \\ -1, & 500 \le N < 750 \\ 0.3\sin\left(\dfrac{\pi N}{25}\right) + 0.1\sin\left(\dfrac{\pi N}{32}\right) & \\ \quad + 0.6\sin\left(\dfrac{\pi N}{10}\right), & 750 \le N < 1000 \end{cases}$$

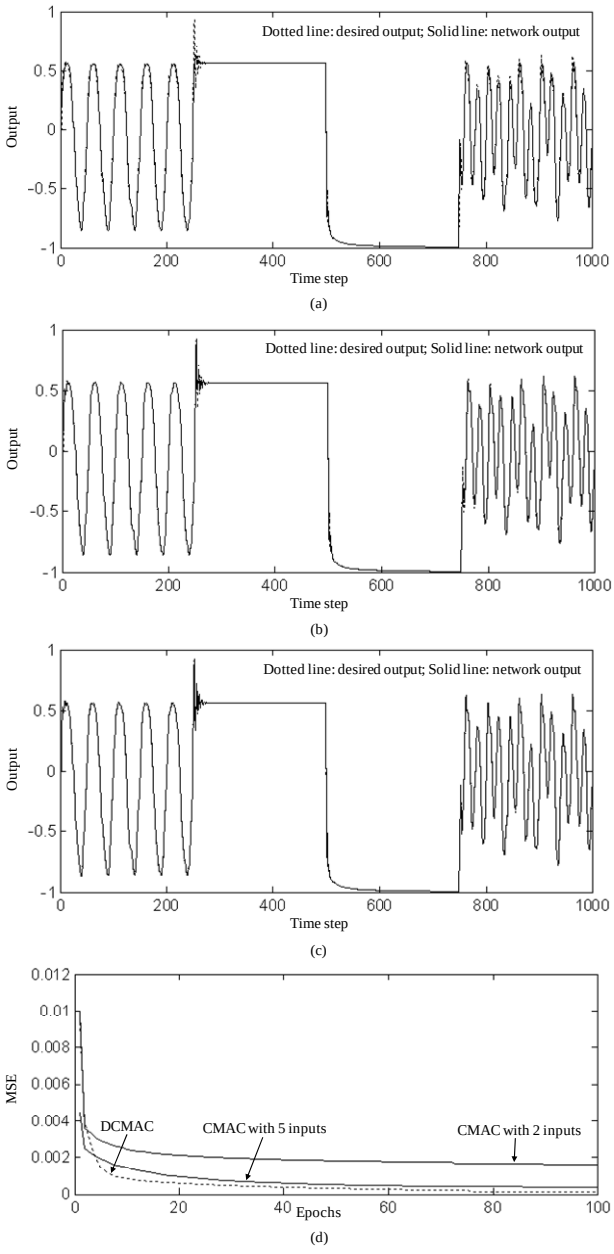

Fig. 5. Simulation results for dynamic system identification. (a) Results of identification using the CMAC with 2 inputs. (b) Results of identification using the CMAC with 5 inputs. (c) Results of identification using the DCMAC. (d) MSE of the DCMAC, CMAC with 2 inputs and CMAC with 5 inputs.

Fig. 5(a) shows the identification results using the CMAC with 2 inputs. Fig. 5(b) shows the identification results using the CMAC with 5 inputs. The results of identification the

DCMAC are shown in Fig. 5(c). Fig. 5(d) presents the MSE of the DCMAC, CMAC with 2 inputs and CMAC with 5 inputs. This simulation demonstrates that the DCMAC has the smaller network structure for identification. In addition, we observe that the identification error of the DCMAC is less than that of the CMAC.

IV. Conclusions

This study developed the dynamic cerebellar model articulation controller (DCMAC). The DCMAC expands on the powerful ability of CMAC (non-recurrent unit, i.e., $r_{ik} = 0$) to overcome temporal problems in prediction and identification. The recurrent network is embedded in the DCMAC by adding feedback connections in the mother wavelet association memory space so that the DCMAC captures the dynamic response, where the feedback units act as memory elements. The dynamic gradient descent method is adopted to adjust DCMAC parameters on-line. Moreover, the analytical method based on a Lyapunov function is proposed to determine the learning-rates of adaptive DCMAC so that the variable optimal learning-rates are derived to achieve most rapid convergence of identifying error. Simulation results show that accurate identifying response and superior dynamic performance can be obtained because of the powerful on-line learning capability of the proposed DCMAC.

References

[1] A. Agarwal, "A systematic classification of neural-network-based control," *IEEE Contr. Syst. Mag.*, vol. 17, pp. 75-93, 1997.
[2] J. G. Kuschewski, S. Hui, and S. H. Zak, "Application of feed-forward neural networks to dynamical system identification and control," *IEEE Trans. Contr. Syst.*, vol. 1, no. 1, pp. 37-49, 1993.
[3] C. M. Lin and C. F. Hsu, "Neural-network-based adaptive control for induction servomotor drive system", *IEEE Trans. Ind. Electron.*, vol. 49, no. 1, pp. 115-123, 2002.
[4] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Network*, vol. 6, no. 1, pp. 144-156, 1995.
[5] T. W. S. Chow, and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE Trans. Ind. Electron.*, vol. 45, no. 1, pp. 151-161, 1998.
[6] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *J. Dyn. Syst., Measurement, Contr.*, vol. 97, no. 3, pp. 220-227, 1975.
[7] S. H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher-order CMAC neural networks," *IEEE Contr. Syst. Mag.*, vol. 12, no. 2, pp. 23-30, 1992.
[8] K. S. Hwang, and C. S. Lin, "Smooth trajectory tracking of three-link robot: a self-organizing CMAC approach," *IEEE Trans. Syst., Man, and Cybern., pt. B*, vol. 28, no. 5, pp. 680-692, 1998.
[9] F. J. Gonzalez-Serrano, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Generalizing CMAC architecture and training," *IEEE Trans. Neural Networks*, vol. 9, no. 6, pp. 1509-1514, 1998.
[10] J. C. Jan and S. L. Hung, "High-order MS_CMAC neural network," *IEEE Trans. Neural Networks*, vol. 12, no. 3, pp. 598-603, 2001.
[11] C. T. Chiang and C. S. Lin, "CMAC with general basis functions," *Neural Networks*, vol. 9, no. 7, pp. 1199-1211, 1996.
[12] Y. H. Kim, and F. L. Lewis, "Optimal design of CMAC neural-network controller for robot manipulators," *IEEE Trans. Syst., Man, Cybern.*, pt. C, vol. 30, no. 1, pp. 22-31, 2000.
[13] S. Jagannathan, "Discrete-time CMAC NN control of feedback linearizable nonlinear systems under a persistence of excitation," *IEEE Trans. Neural Networks*, vol. 10, no. 1, pp. 128-137, 1999.
[14] C.J. Lin and C.C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, pp. 2144-2154, 2004.
[15] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 349-366, 2000.
[16] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.