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

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

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

\[
\phi_k = \exp \left( -\frac{(x_i - m_k)^2}{v_k} \right), \quad \text{for} \ k = 1,2,\ldots,n_s
\]  

where \( \phi_k \) represents the \( k \text{-} \text{th} \) block of the \( i \text{-} \text{th} \) input \( x_i \) with the mean \( m_k \) and variance \( v_k \). In addition, the input of this block for discrete time \( N \) can be represented as

\[
x_i(N) = x_i(N + r_k \phi_k(N - 1))
\]  

where \( r_k \) is the recurrent weight of the recurrent unit. It is clear that the input of this block contains the memory terms \( \phi_k(N - 1) \), which store the past information of the network. This is the apparent difference between the proposed DCMAC and the conventional CMAC.

This schematic diagram of two-dimensional DCMAC operations with \( n_x = 9 \) and \( \rho = 4 \) ( \( \rho \) is the number of elements in a complete block), where \( x_i \) is divided into blocks \( A, B \) and \( C \), and \( x_i \) 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 \( x_i \), and blocks \( d, e \) and \( g \) for \( x_i \) are possible shifted elements. Each block in this space has three adjustable parameters \( m_a, v_d \) and \( r_e \).

3) Receptive-field space \( T \): Areas formed by blocks, named as \( Aa, Bb \) and \( Cc \) are called \( \textit{receptive-fields} \). The number of receptive-field, \( n_s \), is equal to \( n_x \) in this study. Each location in \( T \) corresponds to a \( \textit{wavelet receptive-field} \). The \( \text{multidimensional wavelet receptive-field function} \) is defined as

\[
b_i(x, m, v) = \prod_{k=1}^{n_s} \phi_k = \exp \left( -\frac{\sum (x_i - m_k)^2}{v_k} \right)
\]  

where \( b_i \) is associated with the \( k \text{-} \text{th} \) wavelet receptive-field, \( x_i=[x_{i_1}, x_{i_2}, \cdots, x_{i_n}]^T \in \mathbb{R}^n \), \( m=[m_{i_1}, m_{i_2}, \cdots, m_{i_n}]^T \in \mathbb{R}^n \) and \( v_i=[v_{i_1}, v_{i_2}, \cdots, v_i] \in \mathbb{R}^n \). The multidimensional wavelet receptive-field function can be expressed in a vector form as

\[
\Gamma(x, m, v) = [b_1, b_2, \cdots, b_{n_s}]^T
\]  

where \( m = [m_{i_1}, m_{i_2}, \cdots, m_{i_n}] \in \mathbb{R}^{n_s} \) and \( v = [v_{i_1}, v_{i_2}, \cdots, v_i] \in \mathbb{R}^{n_s} \). 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 \( \rho \) 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

\[
w = [w_{11}, \cdots, w_{1n}, \cdots, w_{1n}] = [w_{11}, \cdots, w_{1n}, \cdots, w_{1n}] = \begin{bmatrix} w_{11} & \cdots & w_{1n} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{n1} & \cdots & w_{n1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{n1} & \cdots & w_{n1} & \cdots & w_{n1} \end{bmatrix}
\]
where $w_i = [w_{i1}, w_{i2}, \cdots, w_{i6}]^T \in \mathbb{R}^6$, and $w_{i0}$ denotes the connecting weight value of the $i$th output associated with the $k$th wavelet receptive-field. The weight $w_{i0}$ 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

$$y_i = w_i^T \Gamma(x_i, m, v) = \sum_{j=0}^{5} w_{ij} b_j(x_i, m, v)$$

(5)

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

$$y = [y_1, y_2, \cdots, y_i, \cdots, y_N]^T = w^T \Gamma$$

(6)

In a two-dimensional case shown in Fig. 2, the output of the DCMAC is the sum of the value in receptive-fields $Bb, Dc, Ee, Hh$ 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

$$E(N) = \frac{1}{2} \left[ y_o(N) - y_s(N) \right]^2 = \frac{1}{2} e_o^2(N)$$

(7)

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

$$\Delta w_i(N) = -\eta \frac{\partial E}{\partial w_i} = -\eta \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial \hat{b}_i} \hat{b}_i = \eta e_o(N) b_i$$

(8)

where the positive factor $\eta$ is the learning-rate for the output weights $w_i$. The connective weights can be updated according to the following equation:

$$w_i(N+1) = w_i(N) + \Delta w_i(N)$$

(9)

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

$$\Delta m_s(N) = -\eta \frac{\partial E}{\partial m_s} = -\eta \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial \hat{b}_i} \hat{b}_i \phi_x$$

$$= \eta e_o(N) w_j b_j \frac{2(x_j - m_s)}{\nu_s^2}$$

$$\Delta v_s(N) = -\eta \frac{\partial E}{\partial v_s} = -\eta \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial \hat{b}_i} \hat{b}_i \phi_x \phi_y$$

$$= \eta e_o(N) w_j b_j \frac{2(x_j - m_s)}{\nu_s^2}$$

$$\Delta r_s(N) = -\eta \frac{\partial E}{\partial r_s} = -\eta \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial \hat{b}_i} \hat{b}_i \phi_x \phi_y \phi_z$$

$$= \eta e_o(N) w_j b_j \frac{2(x_j - m_s)}{\nu_s^2}$$

(10)

where the positive factors $\eta_o, \eta_v, \eta$ and $\eta$ 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:

$$m_s(N+1) = m_s(N) + \Delta m_s(N)$$

(13)

$$v_s(N+1) = v_s(N) + \Delta v_s(N)$$

(14)

$$r_s(N+1) = r_s(N) + \Delta r_s(N)$$

(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 $\eta_o, \eta_v, \eta$ and $\eta$. 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 $\eta$ be the learning-rates for the DCMAC and let $\frac{\partial y_o}{\partial s} \partial s = P_o(N)$ for $s = w, m, v$ and $r$. Then the convergence of identifying error is guaranteed if $\eta$ is chosen as

$$0 < \eta < \frac{2}{\| P(N) \|}$$

(16)

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

$$\eta^* = \frac{1}{\| P(N) \|}$$

(17)
Proof: Since

$$P_s(N) = \frac{\partial y}{\partial s}$$ for s = w, m, v and r, it reveals that

$$P_s(N) = \frac{\partial y}{\partial w} = \left[ \frac{\partial y}{\partial w_1}, \ldots, \frac{\partial y}{\partial w_i}, \ldots, \frac{\partial y}{\partial w_{wa}} \right]'$$ (18)

$$P_s(N) = \frac{\partial y}{\partial m} = \left[ \frac{\partial y}{\partial m_1}, \ldots, \frac{\partial y}{\partial m_i}, \ldots, \frac{\partial y}{\partial m_{ma}} \right]'$$ (19)

$$P_s(N) = \frac{\partial y}{\partial v} = \left[ \frac{\partial y}{\partial v_1}, \ldots, \frac{\partial y}{\partial v_i}, \ldots, \frac{\partial y}{\partial v_{va}} \right]'$$ (20)

$$P_s(N) = \frac{\partial y}{\partial r} = \left[ \frac{\partial y}{\partial r_1}, \ldots, \frac{\partial y}{\partial r_i}, \ldots, \frac{\partial y}{\partial r_{ra}} \right]'$$ (21)

where

$$\frac{\partial y}{\partial v_i} = b_i, \quad \frac{\partial y}{\partial v_i} = w_i b_i \frac{2(x_i - m_i)}{v_i^3}$$,

and

$$\frac{\partial y}{\partial r_i} = w_i b_i \frac{2(m_i - x_i)}{v_i^3} \phi_i(N - 1).$$

Define Lyapunov function as

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

then change of the Lyapunov function is obtained as

$$\Delta V(N) = V(N + 1) - V(N) = \frac{1}{2} \left[ e_s^2(N + 1) - e_s^2(N) \right]$$ (23)

The error difference can be represented by

$$e_s(N + 1) = e_s(N) + \Delta e_s(N) = e_s(N) + \left[ \frac{\partial e_s(N)}{\partial s} \right] \Delta s$$ (24)

it is obtained that

$$\frac{\partial e_s}{\partial s} = \frac{\partial e_s}{\partial y_s} \frac{\partial y_s}{\partial s} = -P_s(N)$$ (25)

Thus

$$e_s(N + 1) = e_s(N) - \left[ P_s(N) \right]^T \eta e_s(N) P_s(N)$$ (26)

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

$$\Delta V(N) = \frac{1}{2} \eta e_s^2(N) \left[ P_s(N) \right]^T \left[ \eta \left[ P_s(N) \right]^T - 2 \right]$$ (27)

Remark: If $\eta$ is chosen as $0 < \eta < \frac{2}{\|P_s(N)\|^2}$, then change of the Lyapunov function is less than 0. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ is guaranteed. Thus, the tracking error $e_s(N)$ will converge to zero as $t \to \infty$. Moreover, the optimal learning-rates which achieve the most rapid convergence are corresponding to $2\eta \|P_s(N)\| - 2 = 0$, i.e., $\eta = \frac{1}{\|P_s(N)\|^2}$, which comes from the derivative of (27) with respect to $\eta$ 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., $r_o = 0$). 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.](image)

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 accurately 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.

\[
u(N) = \begin{cases} 
\sin \left( \frac{\pi N}{25} \right) & 0 < N < 250 \\
1 & 250 \leq N < 500 \\
-1 & 500 \leq N < 750 \\
0.3 \sin \left( \frac{\pi N}{25} \right) + 0.1 \sin \left( \frac{\pi N}{32} \right) + 0.6 \sin \left( \frac{\pi N}{10} \right) & 750 \leq N < 1000 
\end{cases}
\]

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

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,

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., \( r_n = 0 \)) 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