Computational Structural Engineering
An International Journal
Vol. 3, No. 1, pp. 39~47 (2003)

Computational Structural Engineering Institute

Sensitivity Property of Generalized CMAC Neural Network

Dong-Hyawn Kim† and In-Won Lee*

†Senior Reserchet, Coastal Eng. & Horbor Reserch Division, KORDI
**Professor, Dept. of Civil Engineering, KAIST*

Received December 1999; Accepted February 2000

ABSTRACT

Generalized CMAC (GCMAC) is a type of neural network known to be fast in learning. The network may be useful in structural engineering applications such as the identification and the control of structures. The derivatives of a trained GCMAC is relatively poor in accuracy. Therefore to improve the accuracy, a new algorithm is proposed. If GCMAC is directly differentiated, the accuracy of the derivative is not satisfactory. This is due to the quantization of input space and the shape of basis function used. Using the periodicity of the predicted output by GCMAC, the derivative can be improved to the extent of having almost no error. Numerical examples are considered to show the accuracy of the proposed algorithm.

Keywords: cerebellar model articulation controller(cmac), training, neural-networks, basis function, derivative, finite difference, hashing, quantization.

1. Introduction

The cerebellar model articulation controller (CMAC) proposed by Albus(1975) has shown promise in the field of control engineering for the last two decades. Local learning characteristics and fast convergence are fascinating features of CMAC. CMAC has, however, inherent problems. The output of CMAC is discontinuous due to the quantization of the input space. To obtain a smooth output with CMAC, input space has to be densely quantized. This requires a substantial increase in memory. In addition, the derivatives of the trained CMAC are not obtained. The needs for continuous output with appropriate memory size and for the derivative of CMAC opened the second generation of CMAC. CMAC with spline functions enable continuous output and derivative of CMAC. Both continuous output and derivative of CMAC can be obtained thereafter. Spline receptive field function is an interpolation function that is used in the calculation of output at the intermediate space between quantization centers. Fuzzy membership function which is another type of interpolation function has been used by Junhong Nie, et al.(1993) and by U. D. Patel et al. (1995). They called it fuzzified CMAC, or simply FCMAC. Recently, generalized CMAC, so called GCMAC, has been proposed by C. T. Chiang et al.(1996) and F. J. Gonzalez-Serrano et al.(1998). The spline receptive field function is generalized as basis function for interpolation in their work. They showed that nonlinear smooth function can be approximated via basis function, and the derivative of output can be obtained directly by differentiating the basis function.

Although the derivative of CMAC can be obtained via interpolation function such as spline receptive field function, fuzzy membership function and basis function, the accuracy of derivatives is not satisfactory. Derivatives through the direct differentiation of basis function have inherent errors due to quantization of input space and the shape of basis function. Accuracy of derivative is important for control applications. If the accuracy of derivative of emulator CMAC is poor, learning speed of controller, which is trained with the help of emulator CMAC, becomes slower. Finally, the performance of on line controller also becomes poor. The calculation of derivatives with improved accuracy is performed in this study. Proposed algorithm is a finite difference scheme using the periodicity of output of GCMAC. GCMAC is used for

† Corresponding author
Tel:+82-31-400-6345 , Fax: +82-31-408-5823
E-mail address: eastlite@kordi.re.kr

function approximation and the derivative of approximated function is calculated by the proposed differentiation algorithm applied to trained GCMAC.

2. CMAC Structure

CMAC is an associate memory mapping structure imitating human cerebellum. For a given input state, a group of addresses indicating associated memory locations is invoked. The weights stored in the activated memory are added to give the output of CMAC. Therefore, the process of CMAC is divided into two mapping functions. One is memory-addressing mapping. The other is sum-up mapping.

Consider the memory-addressing mapping. To invoke some associated addresses, input space must be quantized into a finite number of regions. Two input variables, x_1 and x_2, may be quantized as shown in Fig. 1. Quantization intervals are called *blocks*, and quantized regions such as A_{11}, A_{12}, ..., and B_{11}, B_{12}, ..., are called *hypercubes* . First, input space is quantized as shown in Fig. 1(a). Then, by shifting quantization mesh toward s, a second method of quantization can be performed as shown in Fig. 1(b). With these methods, several number of quantization may be achieved. The number of quantization is called generalization width, or number of elements. This hashing algorithm can reduce the required size of memory. For N inputs, each taking Ω different values, Ω^N memory locations are required. Whereas, if the generalization width is N_g, the size is reduced to $N_g(\Omega/N_g)^N$ in a hashing algorithm.

In the case of two input variables shown in Fig. 1, hypercubes A_{23} and B_{32} are invoked for the state marked with symbol ✖. And generally, the addresses invoked by state $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T$ can be expressed as

$$a_k = ceil\left(\frac{x_1 - s_{k,1}}{q_1}\right) + \sum_{i=2}^{n}\left[ceil\left(\frac{x_1 - s_{k,1}}{q_1}\right)\cdot\prod_{j=1}^{i-1}(b_j+1)\right] + a_{k,0}$$

$$k = 1, 2, \dots, N_g \quad (1)$$

where

ceil(*y*) : the least integer greater than or equal to *y*,

$s_{k,i}$: shifting value of i-th variable on k-th element for hash mapping,

q_i : quantization interval of i-th variable,

b_j : number of blocks covering entire region of j-th variable,

$a_{k,0}$: starting position of address for k-th element and can be expressed as

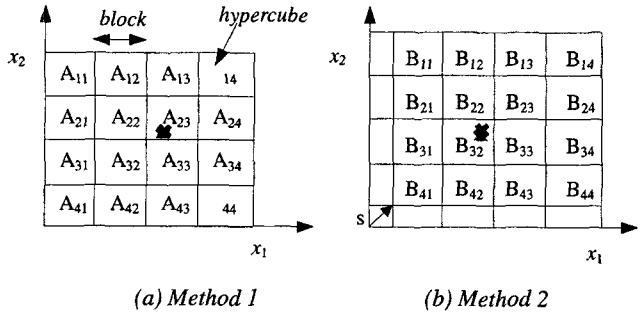


Fig. 1. Quantization scheme for hash mapping

$$\begin{cases} a_{k-1,0} + \prod_{i=1}^{n}(b_i+1), & (k \geq 2) \\ 1 & (k=1) \end{cases} \quad (2),$$

n : dimensionality of input space.

After the activation of associated addresses, CMAC produces its output by summing up the weights stored in the addresses using

$$u = \sum_{k=1}^{N_h} W(a_k) \quad (3)$$

where N_h is the total number of hypercubes(weights) which can be expressed as $N_g \cdot \Pi_{i=1}^{n}(b_i+1)$, a_k equals to one only at the activated, equals to zero at the others, and W is the weight. The conventional CMAC proposed by Albus[3] has the form given in Eq. (3) for output calculation. The weights of addresses invoked are simply added to produce output u. The inherent problem is that output is discontinuous and it cannot be differentiated because weights are constants in hypercubes.

Research efforts have generalized CMAC structure to overcome these problems. The most generalized form has been reported by C. T. Chiang *et al*[12], which is similar to the conventional CMAC except for the output calculation. The Generalized CMAC, the so called GCMAC, gives output in the following form

$$u = \sum_{k=1}^{N_h} \Phi_k(\mathbf{x}) W(a_k) \quad (4)$$

where

$$\Phi_k(\mathbf{x}) = \prod_{i=1}^{n} \phi_k(x_i) \quad (5)$$

In Eq. (5), $\phi_k(x_i)$ is called basis function and can have many types. In Chiang's paper, Gaussian function is used for basis function which can be written as

$$\phi_k(x_i) = \exp\left[-\left(\frac{x_i-\mu_{k,i}}{\sigma_{k,i}}\right)^2\right] \tag{6}$$

where μ_i and σ_i are the mean and standard deviation of x_i, respectively, in the quantization interval and can be expressed as

$$\mu_{k,i} = \left[2\cdot ceil\left(\frac{x_1-s_{k,1}}{q_1}\right)-1\right]\times\frac{q_i}{2}+s_{k,i} \tag{7}$$

$$\sigma_{k,i} = cq_i \tag{8}$$

The output of GCMAC has two fascinating features: continuity and differentiability. Because the basis function acts as an interpolator, output of GCMAC is continuous. In addition, derivative of output can be obtained by directly differentiating the basis function. To train GCMAC, weights are updated through the addition of incremental value as follows:

$$\Delta W(a_k) = \frac{\eta}{N_g}(f-u)\Phi_k(\mathbf{x}) \tag{9}$$

where η and f are learning rate and target function to be learned respectively.

3. Sensitivity Property of GCMAC

Although GCMAC can have continuous differentiable output, the accuracy of the derivative is not satisfactory. Derivative errors are mainly caused by the quantization of input and by the shape of basis function. If Gaussian function is used for basis, output of GCMAC have the form shown in Fig. 2(a).

Because the shape of output of GCMAC smoothly fluctuates as shown in Fig. 2(a), errors in the derivatives are amplified as shown in Fig. 2(b).

The shape of output of GCMAC reveals the periodicity of errors. Using this periodicity, error in the derivative can be reduced. To calculate the derivative of GCMAC, the following equation has been used until now.

$$\frac{\partial u}{\partial x_i} = \sum_{k=1}^{N_h}\frac{\partial\Phi_k}{\partial x_i}W(a_k) \tag{10}$$

where

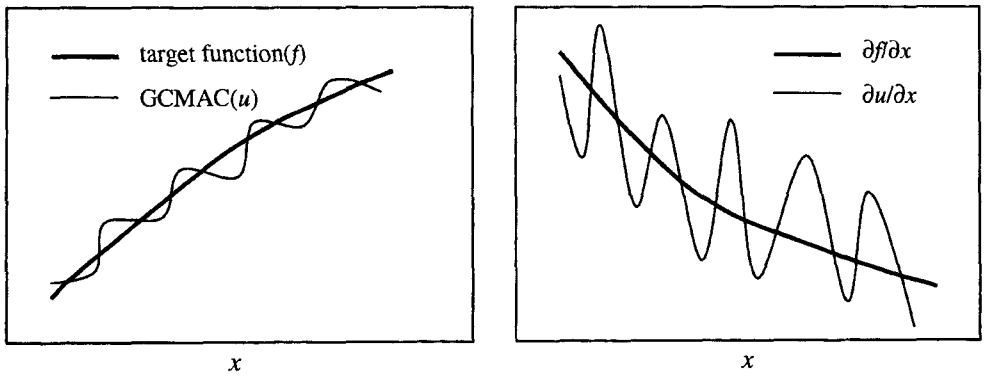
$$\frac{\partial\Phi_k}{\partial x_i} = \frac{\partial\phi_k}{\partial x_i}\prod_{\substack{j=1\\(j\neq i)}}^{n}\phi_k(x_j) \tag{11}$$

However, a finite difference scheme in the following equation is used for the derivative of GCMAC in this study:

$$\frac{\partial u}{\partial x_i}\cong\frac{u(x_i+\Delta p_i/2)-u(x_i-\Delta p_i/2)}{\Delta p_i} \tag{12}$$

where

$$u(x_i+\delta) = \sum_{k=1}^{N_h}\Phi_k(\hat{\mathbf{x}})W(a_k) \tag{13}$$



(a) Target function(f) vs. GCMAC(u) *(b) Derivatives of f and u*

Fig. 2. Error in GCMAC and the derivative

$$\hat{\mathbf{x}} = [x_1 \quad x_2 \quad \cdots \quad x_{i-1} \quad x+\delta \quad x_{i+1} \quad \cdots \quad x_n]^T, \tag{14}$$

p_i: period of GCMAC along x_i

The efficiency of finite difference scheme for derivative of GCMAC is shown in Fig. 3. At x_c, derivatives by finite difference are closer to the exact derivatives than those by direct differentiation. Since the period(p_i) is the directional distance along the x_i direction between the centers of the closest basis function in all elements, if the difference Δx_i of the finite difference calculation equals to this period, the error in the derivative is minimized.

4. Numerical Examples

Example 1 : Single Input Single Output(SISO) case

Single input single output function given by Eq. (15) is learned by GCMAC. The Gaussian function is used as basis with c=0.21. Quantization interval is 0.5(rad). The number of elements is 4. Therefore, $4\times\{ceil(2\pi/0.5)+1\}^2=56$ weights are used for the output of GCMAC. Shifting matrix in Eq. (16) is used for the hash mapping of addresses. After learning, outputs of GCMAC are almost exactly the same as function f given by

$$f = \exp\left[-\left(\frac{x-\pi}{2}\right)^2\right]\cdot\cos\left(\frac{3x}{2}\right), \quad 0\le x\le 2\pi \tag{15}$$

$$\begin{aligned} S = [s_{k,i}] &= [0 \quad r/4 \quad 2r/4 \quad 3r/4]^T \\ &= [0 \quad 0.125 \quad 0.25 \quad 0.375]^T \end{aligned} \tag{16}$$

Fig. 4(a) shows the function to be approximated and the predicted output by the trained GMAC. The prediction error between f and GCMAC is shown in Fig. 4(b). As expected, error shows periodic characteristics. Fig. 5 shows the windowed Fourier transform of the error from 2.0 rad to 3.0 rad. The first dominant peak at 8.0(rad^1) shows that the error has periodic signal with period 1/8 (rad) which exactly coincide with the quantization interval(0.5) divided by the number of elements along the x-direction(4). Therefore 0.125 is used as the difference value for the derivative of GCMAC in the finite difference calculation.

In Fig. 6, analytic differentiation(*AD*) shows fluctuating

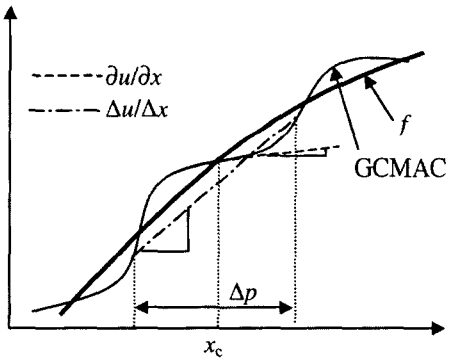


Fig. 3. Finite difference vs. direct differentiation

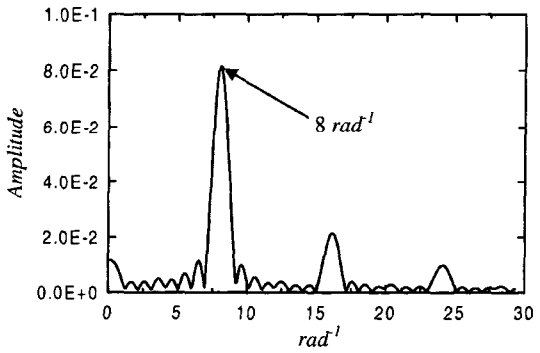
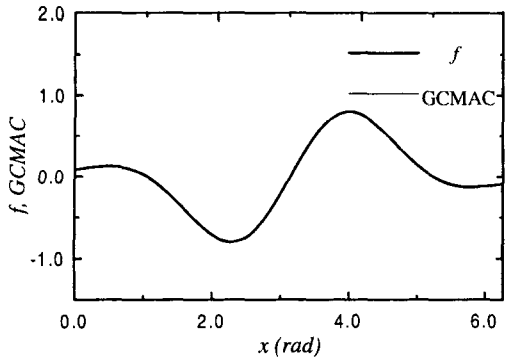
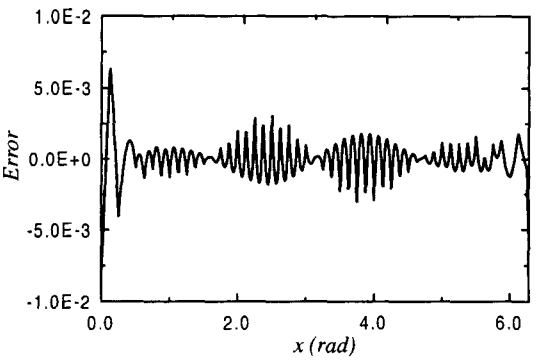


Fig. 5. Fourier transform of the error.



(a) GCMAC vs. f



(b) Error (f-GCMAC)

Fig. 4. Result of learning of f by GCMAC (SISO case)

error, but derivative using finite difference(*FD*) is almost the same as the exact derivative(*ED*). When the finite difference scheme is used for the interval $x_{\min} \le x \le x_{\min}+\Delta x$, and $x_{\max}-\Delta x \le x \le x_{\max}$, where the finite difference cannot be applied, the finite difference scheme is slightly modified as given by the following equations.

$$\frac{\partial u}{\partial u_i} \cong \frac{\Delta u}{\Delta x_i} = \frac{u(x_i+\Delta p_i/2)-u(x_{i,\min})}{x_i+\Delta p_i/2-x_{i,\min}}, \quad \text{if} \quad x_{i,\min} \le x_i \le x_{i,\min}+\Delta x_i \tag{17}$$

$$\frac{\partial u}{\partial u_i} \cong \frac{\Delta u}{\Delta x_i} = \frac{u(x_{i,\max})-u(x_i-\Delta p_i/2)}{x_{i,\max}-(x_i-\Delta p_i/2)}, \quad \text{if} \quad x_{i,\min}-\Delta x_i \le x_i \le x_{i,\max} \tag{18}$$

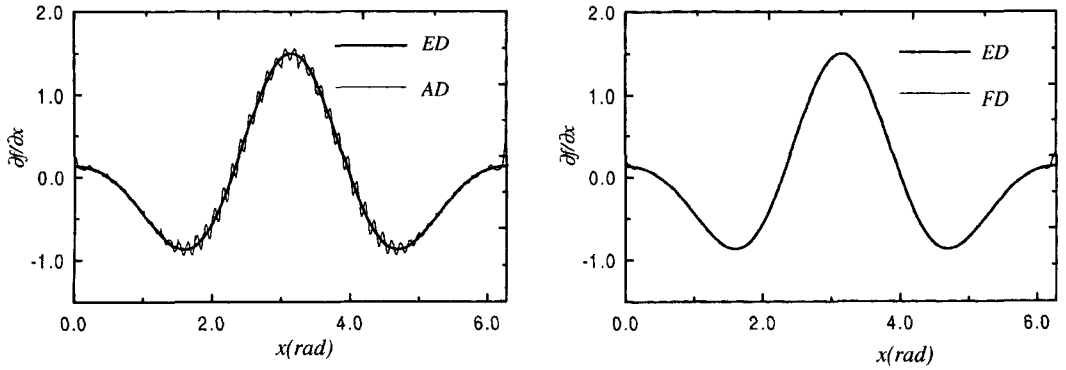
Possible use of another two basis functions are exploited for the learning of function *f*. One is linear interpolation function and the other is sinusoidal function which are expressed as

$$\phi_k(x_i) = 1-\left|\frac{x_i-\mu_{k,i}}{q_i/2}\right| \tag{19}$$

$$\phi_k(x_i) = \cos^2\left(\frac{\pi}{2}\cdot\frac{x_i-\mu_{k,i}}{q_i/2}\right) \tag{20}$$

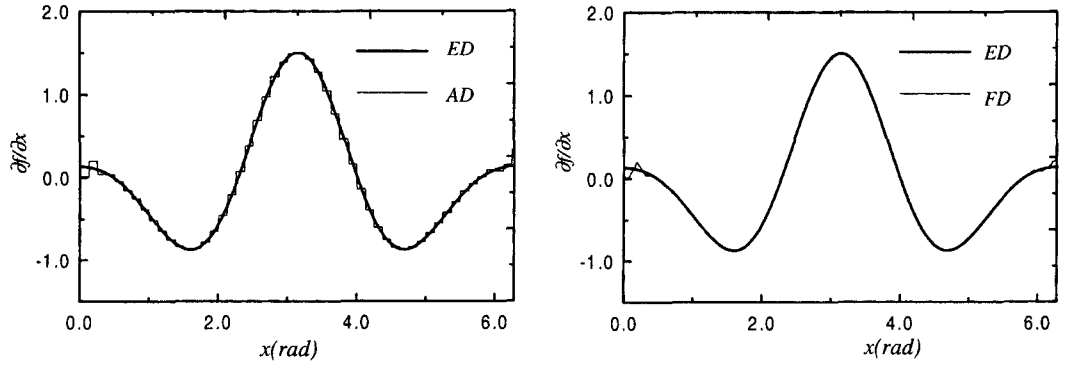
Fig. 7 shows derivative of *f* when linear basis function is used for GCMAC learning. As expected, piecewise continuous derivative is obtained because the derivative of linear basis function is piecewisely continuous. Fig. 8 shows the case when sinusoidal basis function is used. It shows that the finite difference scheme is excellent for the derivative calculation of GCMAC.

To find the effect of the finite difference value on the accuracy of derivatives, difference Δx is varied from 10% to 300 % of period of GCMAC, *p*. Figure 9 shows errors versus the variation of Δx. The parameter α in Fig. 9 is defined as $\Delta x/p$. ASE is defined as *accumulated square*



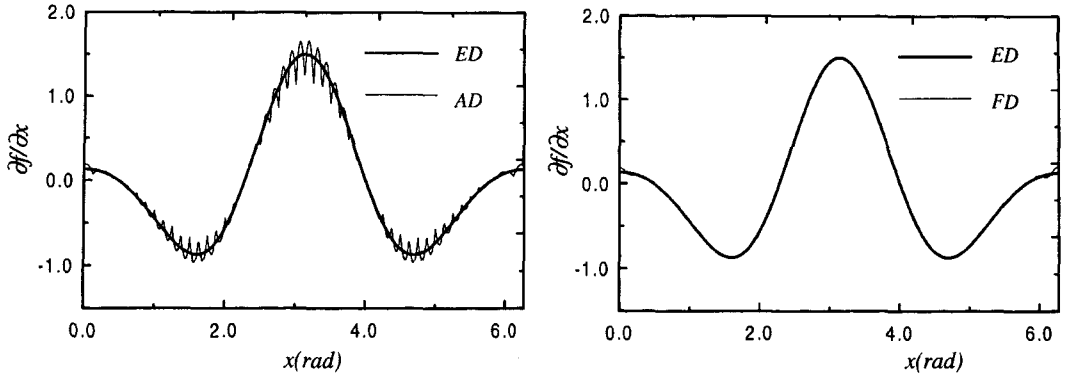
(a) Analytic differentiation *(b) Finite difference*

Fig. 6. Analytic vs. finite difference differentiation (Gaussian basis)



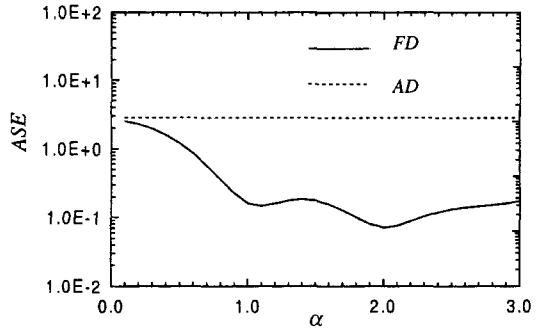
(a) Analytic differentiation *(b) Finite difference*

Fig. 7. Analytic vs. finite difference differentiation (linear basis)

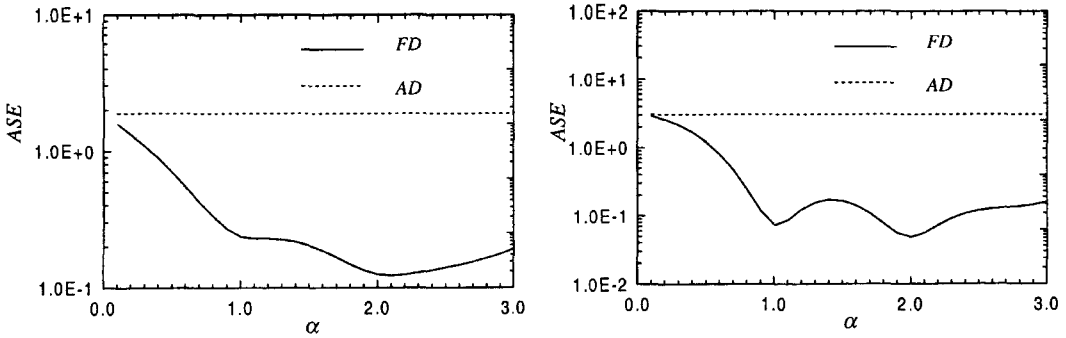


(a) Analytic differentiation *(b) Finite difference*

Fig. 8. Analytic vs. finite difference differentiation (sinusoidal basis)



(a) Gaussian basis function



(b) Linear basis function *(c) Sinusoidal basis function*

Fig. 9. Effect of finite difference on ASE.

errors for every 0.01 rad.

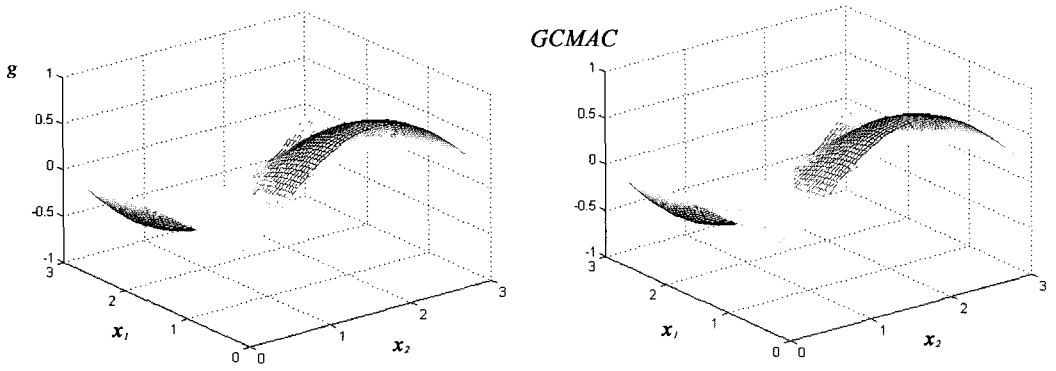
Error of the derivatives using finite difference(*FD*) is always under that of direct differentiation(*AD*) until α is less than 3. ASE has local minimum at α=1 where finite difference equals to the period p(=0.125). ASE has local minimum also at α=2, which means that error of GCMAC shows periodic output also $2p$. As a goes to zero, ASE approaches to that of analytic differentiation, since difference equation becomes analytic differentiation equation.

Example 2 : Multi Input Single Output(MISO) case

GCMAC learns the multi-input single-output functior given by Eq. (21). Fig. 10 shows the function g and the output of GCMAC after learning.

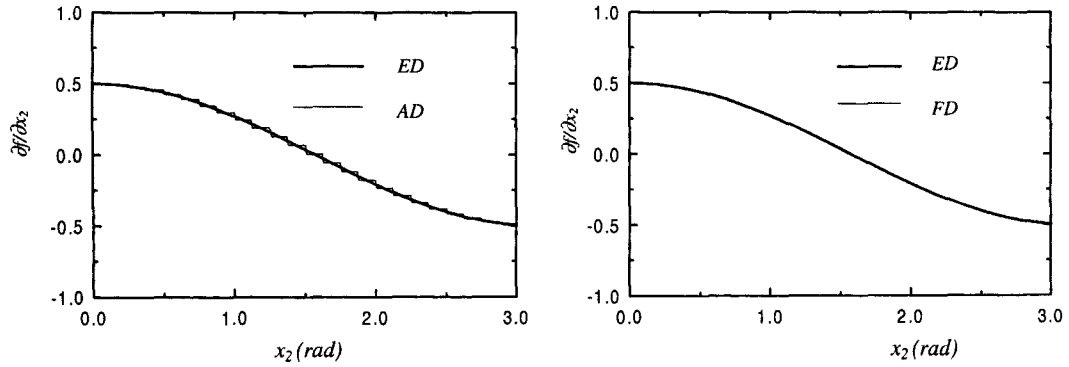
$$g(x_1,x_2) = cosx_1 \cdot sinx_2, \qquad 0 \le x_1, x_2 \le 3(rad) \qquad (21$$

Quantization interval is 0.5rad(q_1, q_2) for both x_1 and x_2



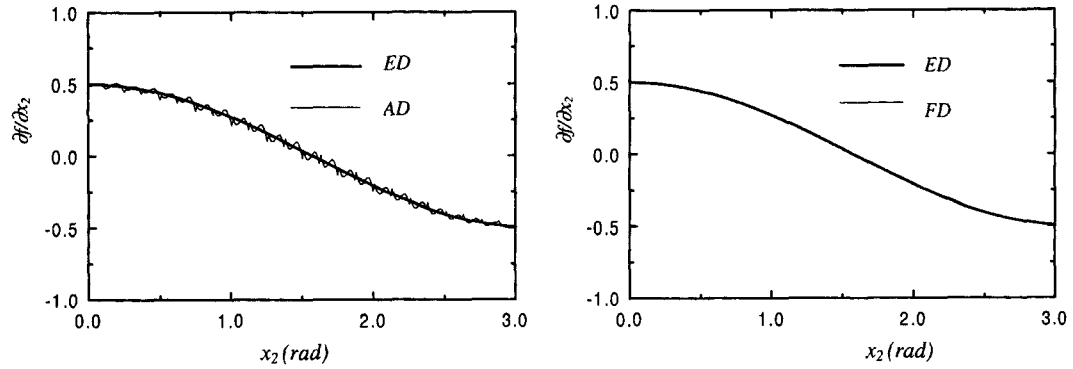
(a) Function to be learned (b) Output of GCMAC

Fig. 10. Function g and output of GCMAC



(a) Analytic differentiation (b) Finite difference

Fig. 11. Analytic vs. finite difference differentiation (linear basis, $x_j=\pi/3$)



(a) Analytic differentiation (b) Finite difference

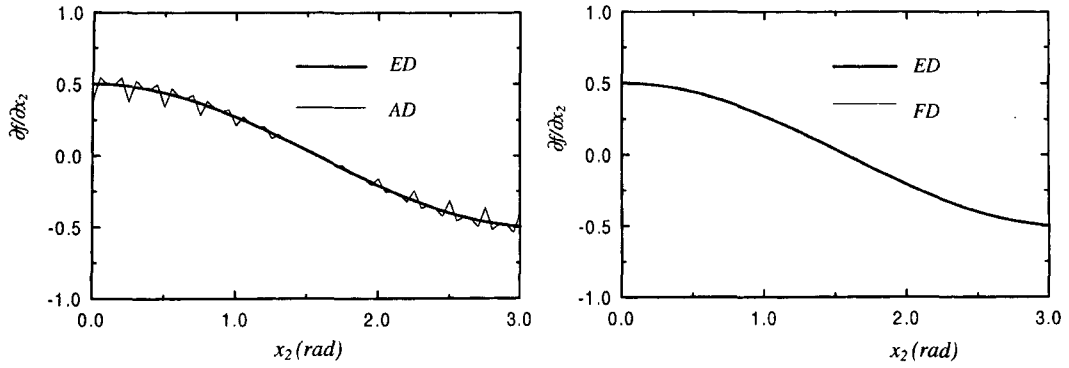
Fig. 12. Analytic vs. finite difference differentiation (Gaussian basis, $x_j=\pi/3$)

$6(N_g)$ elements are used for hash mapping. The shifting matrix $S(16\times2)$ is given by

$$S=[s_{k,i}]=\begin{bmatrix} \vdots & \vdots \\ \frac{q_1}{4}\times\mathrm{mod}(k-1,4) & \frac{q_2}{4}\times\mathrm{fix}\{(k-1)/4\} \\ \vdots & \vdots \end{bmatrix} \quad (22)$$

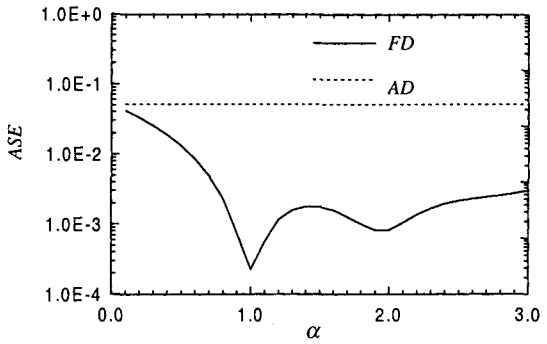
where mod(d,e) is the remainder of d divided by e, and fix(m) is the least integer less than or equal to m. The number of weights used is $\{ceil(3/q_1)+1\}\times\{ceil(3/q_2)+1\}\times N_g$, or 784. Three basis functions are all used respectively. Derivatives of GCMAC using basis functions are shown in Fig. 11~13.

Discontinuous errors are shown in analytic differenti-

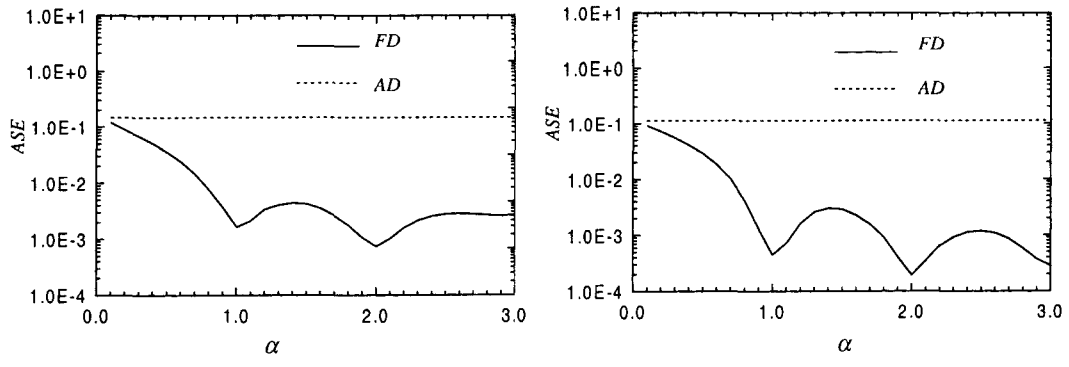


(a) Analytic differentiation *(b) Finite difference*

Fig. 13. Analytic vs. finite difference differentiation (sinusoidal basis, $x_1=\pi/3$)



(a) Linear basis function



(b) Gaussian basis function *(c) Sinusoidal basis function*

Fig. 14. Effect of finite difference on ASE ($x_1=\pi/3$)

ation, but almost exact derivative can be obtained by the proposed finite difference scheme.

For Gaussian basis and sinusoidal basis function analytic derivatives have fluctuating errors, but derivatives by the finite difference scheme are almost the same as the exact ones. Effect of difference, Δx_2, on the accuracy of derivative by finite difference scheme is shown in Fig. 14. When Δx_2 equals to p_2(01.25 rad) and $2p_2$ (0.25), ASE has its local minimum.

5. Conclusions

To improve the accuracy of the derivative of a trainec GCMAC, a finite difference scheme with the finite difference equal to the period of output of the trainec GCMAC is proposed. The period is the quantization interval divided by the generalization width along the directior toward which the derivative is calculated, or the distance between the centers of the nearest basis function in all ele-

ments. The error in the derivative also has the local minimum also at twice the period because the periodic nature appears at twice the period. Linear and sinusoidal basis function are exploited for training of GCMAC. Test results show that not only the Gaussian function but linear and sinusoidal basis function are also applicable to the proposed method.

References

Albus JS (1975) A new approach to manipulator control : the cerebellar model articulation controller, ASME Trans., J. Dyn. Sys. Meas. Cont., 97: 220-227.

Chiang CT, Lin CS (1996) CMAC with general basis function, Neural Networks, 9(7): 1199-1211.

Comoglio RF, Pandya AS (1992) Using a cerebellar model arithmetic computer neural network to control an autonomous underwater vehicle, Proceedings of the International Joint Conference on Neural Networks, 781-786.

Gonzalez-Serrano FJ, Figueiras AR, Artes-Rodriguez A (1998) Generalizing CMAC architecture and training, IEEE Trans. Neural Networks, 9(6): 1509-1514.

Hama H, Xing C, Liu Z (1998) New high-order association memory system based on Newtons forward interpolation, IEICE Trans. Fundamentals, E81-A(12): 2688-2693.(in Japanese)

Hung SH, Jan JC (1999) MS_CMAC Neural Network Learning Model in Structural Engineering, ASCE Journal of Computing in Civil Engineering, 13(1): 1-11.

Ker JS, Hsu CC, Kuo YH, Liu BD (1997) A fuzzy CMAC model for color reproduction, Fuzzy Sets and Systems, 91: 53-68.

Kim JT, Jung HJ, Lee IW (2000) Optimal Structural Control Using Neural Networks, ASCE Journal of Engineering Mechanics, 126 (2): 201-205.

Kim HS, Lin CS (1992) Use of adaptive resolution for better CMAC learning, Proceedings of the International Joint Conference on Neural Networks, 1517-1522.

Koo KM (1995) Robust position and force/position control of robot manipulator, Ph. D. thesis, department of civil engineering, KAIST, Taejon. (in Korean)

Larsen GA, Cetinkunt S, Donmez A (1995) CMAC neural network control for high precision motion control in the presence of large friction, ASME Trans., J. Dyn. Sys. Meas. Cont., 117: 415-420.

Lane SH, Handelman DA, Gelfand JJ (1992) Theory and development of higher-order CMAC neural networks, IEEE Control Systems Mag., April, 23-29.

Majors M, Stori J, Cho DI (1994) Neural network control of automative fuel-injection systems, IEEE Control Systems Mag., 14(3): 31-36.

Nie J, Linkens DA (1993) A fuzzified CMAC self-learning controller, Second IEEE Int. Conf. Fuzzy Sys., 500-505.

Patel UD, Carroll RL (1995) Self-organizing fuzzy CMAC neural network, Proceedings of the Artificial Neural Networks in Engineering(ANNIE 95), 97-102.

Parks PC, Militzer J (1992) A comparison of five algorithms for the training of CMAC memories for learning control systems, Automatica, 28(5): 1027-1035.

Shang C, Reay, DS, Williams BW (1996) Modified LMS adaptive algorithm for CMAC neural network based control of switched reluctance motors, Electronics Letters, 32(12): 1113-1115.