Vol. 4, No. 2 / June 2000
pp. 67 ~ 73

Application of CMAC to Structural Model Identification

By Dong-Hyawn Kim*, Byoung-Wan Kim** and In-Won Lee***

Abstract

Cerebellar model articulation controller (CMAC) is introduced and used for the identification of structural dynamic model. The speed of learning of CMAC is known to be fast. It can learn structural response within a few seconds. Therefore it is suitable for the real-time identification of structures. Real-time identification is required for the control of structure that may be damaged due to some severe loads, or may be changed in mechanical properties due to shrinkage or relaxation. In numerical examples, it is shown that CMAC trained with the dynamic response of three-story building can predict responses under earthquakes that were not trained with allowable error. Finally, CMAC has great application potential in structural and control engineering.

Keywords: *cerebellar model articulation controller (CMAC), system identification, training, dynamic response, hash mapping, noise.*

1. Introduction

In 1990s artificial neural network theory has been successfully applied to many research areas. Especially it was shown that neural network is powerful in the identification and control of structures with unknown dynamics (J. Ghaboussi *et al.*, 1995; H. M. Chen *et al.*, 1995). To design a robust controller, the mathematical model of a structure should first be obtained. If the model has some uncertainty or error, the controller may show bad performance or even unstable behavior. But in neural network based control, a mathematical model is not required. The structure is identified and controlled only by the learning of neural networks. Therefore, neural network is said to be a promising tool for the identification and control of uncertain, large and nonlinear civil structures.

In most applications, including the identification of structures, the multi-layer perceptron (MLP) has been widely used due to its simplicity in both structure and learning algorithm. The MLP basically learns implicit functional relation from input-output matches. For an instance of input-output pair, all of the weights of the MLP are affected and invoked. Therefore, the weights trained by one set of training data are interfered by another set of the training data. This kind of global learning can save the required memory and has good generalization characteristics, i.e., it can be used to predict the input-output relation that is not trained by the MLP. However, the learning speed becomes too slow in the global learning due to the interference problem. Because the speed of learning of MLP is so slow, it cannot be used for real-time applications, especially for the real-time identification of unknown structures. In controlling a structure, the dynamic properties can be changed due to heavy damage or aging of materials used. Therefore, another tool for learning the dynamic properties of the structure that shows different characteristics from the original one is required. Basically, the fast learning capability is demanded for the new tool.

There are lots of researches on the identification of structures. J. Ghaboussi *et al.*, (1995) and H. M. Chen *et al.*, (1995) simultaneously proposed the identification algorithm of structural dynamic model based on MLP. They showed MLP can serve as a good emulator which is used for the training of controller neural network. In Chen *et al.*'s study,

*Graduate Student, Department of Civil Engineering, Korea Advanced Institute of Science and Technology, 373-1, Yusong, Taejon, 305-701, Korea. (e-mail: jtkim@kaist.ac.kr)

**Graduate Student, Department of Civil Engineering, Korea Advanced Institute of Science and Technology, 373-1, Yusong, Taejon, 305-701, Korea.

***Member, Professor, Department of Civil Engineering, Korea Advanced Institute of Science and Technology, 373-1, Yusong, Taejon, 305-701, Korea

The manuscript for this paper was submitted for review on November 10, 1999.

off-line training process takes 59,200 epochs in reducing the root mean square error to an allowable level. In other words, it takes several minutes or hours in training the neural network. In J. Ghaboussi *et al.*'s study, the exact time spent on training the neural identifier was not shown because they trained it off-line in which learning time is not much of a concern. Probably, they spent more than minutes on training. After them, there have been researches on neural network based identification and control (H. M. Chen *et al.*, 1995; S. D. Snyder *et al.*, 1995; Y. Tang 1996; K. Nikzad *et al.*, 1996; K. Bani-Hani *et al.*, 1998). In all of their researches the MLP is used for identifier and trained off-line. The learning time was not short enough to be used for the real-time identifier.

A kind of neural network called cerebellar model articulation controller (CMAC) is introduced to explore the possibility of usage as a real-time identifier of structural model. CMAC was first proposed by J. S. Albus (1975) and has been improved by many researchers (S. H. Lane *et al.*, 1992; J. Nie *et al.*, 1993; C. T. Chiang *et al.*, 1996; F. J. Gonzalez-Serrano *et al.*, 1998; H. Hama *et al.*, 1998). It has a fast convergence in learning. Therefore, it has been mainly used in the control of electrical and mechanical systems that require the fast learning and operation speed (M. Michael *et al.*, 1994; K. M. Koo *et al.*, 1994).

S. L. Hung *et al.*, (1999) first used CMAC as civil engineers. They proposed Macro Structure CMAC (MS_CMAC), constructed by a number of CMACs, to estimate some coefficients for steel construction. It was shown that the learning speed of CMAC is quite fast in their applications.

CMAC is used for the identification of a dynamic structural model in this research. CMAC fed into the information of the states and a external load of one step ahead is trained to predict the next step response of structure of which the mathematical model is assumed to be unknown. A systematic procedure of CMAC is presented. In the numerical example, the response of three-story building under earthquakes is trained by CMAC, and another responses under untrained earthquakes are predicted. It is shown that the learning speed is fast enough to be used as a real-time identifier with almost little error though the input signals of CMAC are contaminated by noises.

2. Cerebellar Model Articulation Controller (CMAC)

CMAC is a kind of memory-mapping function similar to table look-up method. J. Albus (1975) proposed its original structure imitating human cerebellum. For an input to CMAC, only a few localized elements of memory are invoked to produce corresponding output. The update, or learning, of the stored memory elements is also done very locally, i.e., only the associate memory elements to a given input are updated. Therefore, the speed of learning is very fast and localized non-linearity can be efficiently trained. Since CMAC uses table look-up type memory mapping, the required space of memory is larger than that of MLP. Therefore a special algorithm to save the memory space is used, that is, the information of some adjacent memory elements are shared one another. This is the so-called hash-mapping scheme. A systematic procedure including hash-mapping scheme is presented in what follows.

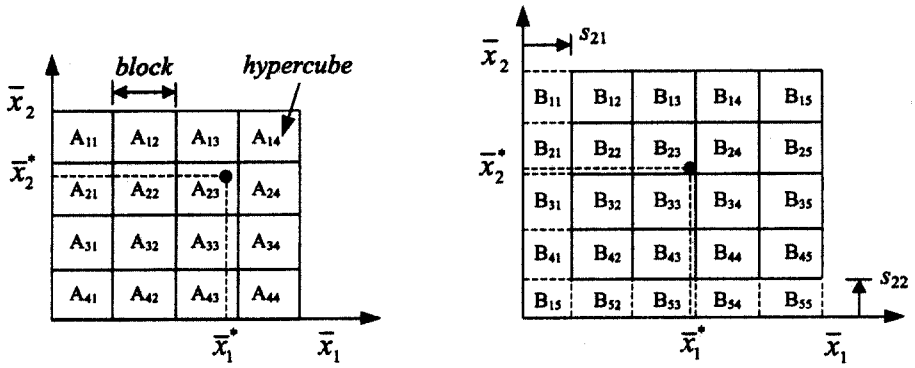
2.1 Moving Input Space

To ease the addressing of associated memory, each input space is moved so that it distributes from zero toward positive direction. If x_i is the i-th input to CMAC, it can be done through

$$\bar{x}_i = x_i - x_{i,\ min};\ i = 1, 2, \cdots, n \tag{1}$$

where $x_{i,min}$ is the minimum of x_i and n is the dimensionality of input space. Then, all of input spaces have minimum values of zero and all values remain in positive domain.

2.2 Addressing Associate Memory(Hash Mapping)



(a) the first way of quantization *(b) the second way of quantization*

Fig. 1. Quantization scheme for hash mapping

Let's consider the case of CMAC with two inputs and single output for conceptual understanding. Each input space, $\bar{x}_1$ and $\bar{x}_2$, is quantized to the size of *block* as shown in Fig. 1. Then, the blocks of each input construct the quantized areas that are called *hypercubes*. Hypercube can be seen as the physical memory. In this manner we have the first way of quantization as shown in Fig. 1(a). The second way of quantization can also be performed very easily by shifting the quantization mesh of the first way of quantization toward a vector $S_2=[s_{21}\ s_{22}]^T$ as shown in Fig. 1(b). The first subscript, 2, of the elements of S_2 means that it is belong to the second way of quantization. The second subscripts mean the direction of input space. That is, s_{21} stands for the shifting value of the first variable, $\bar{x}_1$, in the second way of quantization. The shifting vector S_1 equals $[0\ 0]^T$, because there is no shifting in the first way of quantization. In this example the number of quantization, the so-called *generalization width*, is two. For an input $\bar{X}^* = [x_1^*\ x_2^*]^T$ that is dotted in Fig. 1, hypercubes of A_{23} and B_{23} are selected on each quantization mesh. Then the weights corresponding to the hypercubes are activated. In general, when the generalization width is N_g and the dimensionality sion of input space is n, the indices of the activated weights in physical memory are calculated as

$$a_k = ceil\left(\frac{\bar{x}_1 - s_{k1}}{q_1}\right) + \sum_{i=2}^{n}\left[ceil\left(\frac{\bar{x}_i - s_{ki}}{q_i}\right)\cdot\prod_{j=1}^{i-1}(b_j+1)\right] + a_{k0}$$
$$k = 1, 2, \cdots, N \tag{2}$$

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

s_{ki} : the shifting value of the i-th variable on the k-th quantization,

q_i : the quantization interval of the i-th variable,

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

a_{k0}: the starting position of the address for the k-th quantization, can be expressed as

$$\begin{cases} a_{k-1,0} + \prod_{i=1}^{n}(b_i+1), & (k \geq 2) \\ 1 & (k \geq 1) \end{cases} \tag{3}$$

2.3 Calculating output and Learning

The calculation of output of CMAC is simple. The stored values in the activated addresses are summed to give an output as

$$u = \sum_{m=1}^{N_h} a_m W_m \tag{4}$$

where N_h is the total number of hypercubes or weights, and can be expressed as $N_g \cdot \Pi_{i=1}^{n}(b_i+1)$, and a_m equals one only at the activated, equals zero at the others, W_m is the weight stored at the m-th position.

To make an adequate output to a given input, the CMAC should be trained, i.e., the weights are updated so that the output is close to the desired one within allowable error. The learning rule is simply obtained by minimizing the squared error between the desired and the actual output as

$$W^{new} = W^{old} + \frac{\eta}{N_g}(d-u) \tag{5}$$

where d is the desired output and η is the learning rate.

2.4 The required size of memory

Since the basic idea of CMAC is to store information in table look-up fashion, the size of memory is large. To reduce its size, hash-mapping algorithm described in the previous section is used. If one have n inputs, and each of which can take on Ω different values, then Ω^n memory locations would be required to store the entire information. However, the size is reduced to $N_g(\Omega/N_g)^n$ in hash-mapping algorithm. The ratio is N_g^{1-n}. For example, if n equals 3, and N_g equals 50, the ratio will be $50^{-2}=4\times10^{-4}$. Therefore, it can be said that hash-mapping algorithm is essential to the implementation of CMAC into real world.

3. System Identification

Fig. 2 describes the learning method of the dynamic response of structure. The delayed signals from both the external load and the state of structure are fed into a CMAC. Then, the CMAC estimates the dynamic response of structure. The estimation error gives the information on how much the activated weights should be corrected. Estimation and update are repeated until the error goes below the allowable level. After that, the CMAC can predict the dynamic response of structure with allowable error.

A measure of error should be defined beforehand so that it is used as the stopping condition of learning. One of the widely used criteria is the sum of squared error (SSE).

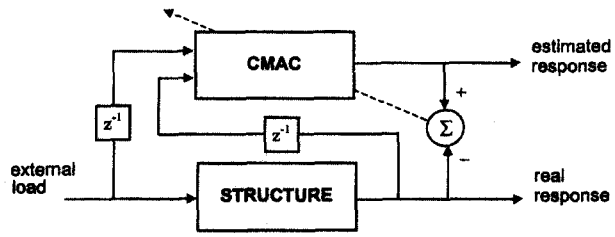
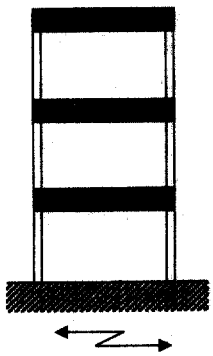


Fig. 2. Learning of dynamic response.



$$M = \begin{bmatrix} 9.806 & 0.0 & 0.0 \\ 0.0 & 9.806 & 0.0 \\ 0.0 & 0.0 & 9.806 \end{bmatrix} (N \cdot \sec^2/m)$$

$$C = \begin{bmatrix} 3.826 & -0.573 & 0.616 \\ -0.573 & 4.567 & -0.026 \\ 0.616 & -0.026 & 4.372 \end{bmatrix} (N \cdot \sec/m)$$

$$K = \begin{bmatrix} 2.740 & -1.641 & 0.369 \\ -1.641 & 3.021 & -1.624 \\ 0.369 & -1.624 & 1.333 \end{bmatrix} \times 10^4 \ (N/m)$$

Fig. 3. Model structure to be identified.

Sometimes, the normalized sum of squared error is used. However they cannot be a good measure because they are affected by the absolute magnitude of outputs or by the number of the predicted outputs. Therefore, Square-root of Averaged Error Norm (*SAEN*) is defined as

$$SAEN = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}\left(\frac{d_i - u_i}{d_{max}}\right)^2} \tag{6}$$

where N_T is the total number of training data. Since the prediction error is normalized and averaged, *SAEN* can be a good measure of the accuracy of prediction regardless of the absolute magnitude of the real response and the number of the predicted response data. Learning is repeated before *SAEN* goes below the desired value.

4. Numerical Examples

4.1 Structural Model

The three-story building structure shown in Fig. 3, which also appeared in the reference [T. T. Soon 1990] was used as a numerical example for identification. This structure was controlled by an active tendon system experimentally in the text. Mass(M), damping(C) and stiffness(K) matrices are presented in Fig. 3. The natural frequencies of the structure are 2.24, 6.83 and 11.53 (Hz), respectively. Modal damping factors are 1.62, 0.39 and 0.36% respectively.

This structure is exposed to some ground motions. Then, the relative displacement and velocity of the top roof are measured at each time step and they are fed into CMAC to estimate the displacement of next step.

4.2 Ground Motions

Ten ground motions listed in Table 1 are randomly chosen from the historical earthquakes and applied to the model structure. The first three ground motions are used in the learning of CMAC and other seven motions are used in the verification of trained CMAC. All motion data have the time interval of 0.02 second and have the duration time of 20 seconds. Therefore, the each motion data has 1000 dif-

Table 1. Ten ground accelerations

Abbr.	Earthquake	Year of Occurrence	Com-ponent	PGA (m/s^2)	Scaling (%)	PGA after scaling (m/s^2)
EC	El Centro	1940	S00E	3.417	100	3.417
NO	Northridge	1994	S65W	3.243	100	3.243
KC	Kern County	1952	N21E	1.527	100	1.527
SF	San Fernando	1971	S16E	9.567	30	2.870
SJ	San Jose	1955	N31W	1.002	200	2.004
BM	Borrego Mt.	1968	S00W	1.278	250	3.195
LB	Long Beach	1933	S08W	1.306	100	1.306
LP	Loma Prieta	1989	N90E	3.258	100	3.258
WW	Western Washington	1949	N86E	2.746	100	2.746
ML	Mammoth Lakes	1980	S90W	3.040	100	3.040

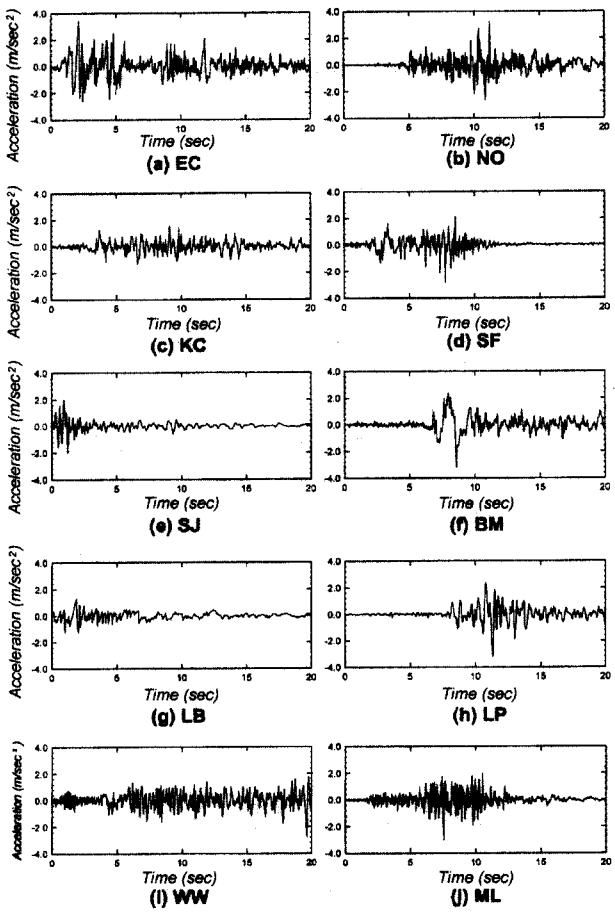


Fig. 4. Time history of ten ground accelerations.

ferent accelerations. Some ground motions having too high or low peak ground acceleration (PGA) are scaled down or up. For example, San Fernando (SF) earthquake of which PGA is 9.567 (m/s^2) was reduced to 30% of the original values. The wave-forms of the earthquakes in the time domain are presented in Fig. 4 and their frequency components are shown in Fig. 5.

4.3 Structure of CMAC and Its Training

There are three inputs to CMAC; the delayed signals of displacement and velocity of top roof and the ground acceleration. With these inputs, CMAC is trained to estimate the displacement of top roof. Quantization intervals for the displacement, the velocity and the ground acceleration are 0.015(m), 0.367(m/s) and 2.33(m/s^2), respectively. And the input spaces have 6, 3 and 3 blocks, respectively. Therefore, the total number of required weights are 50×(6+1)×(3+1)×(3+1)=5600. The number of generalization width is 50 and the learning rate is set to 0.9. The training data set is composed of the dynamic responses due to three earthquakes, EC, NO and KC. Therefore, the training data are composed of 3000 cases. And the learning is repeated until *SAEN* goes below 0.01. The epochs and the CPU-time spent on the learning with pentium 166MHz are summarized in Table 2.

Table 2. Epochs & learning time

Epochs	CPU Time
4	3.406 sec

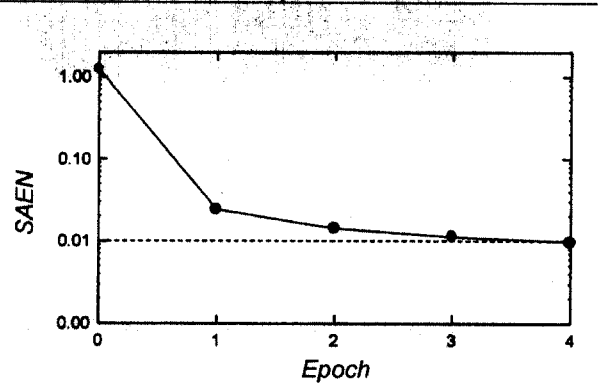


Fig. 6. *SAEN* during learning.

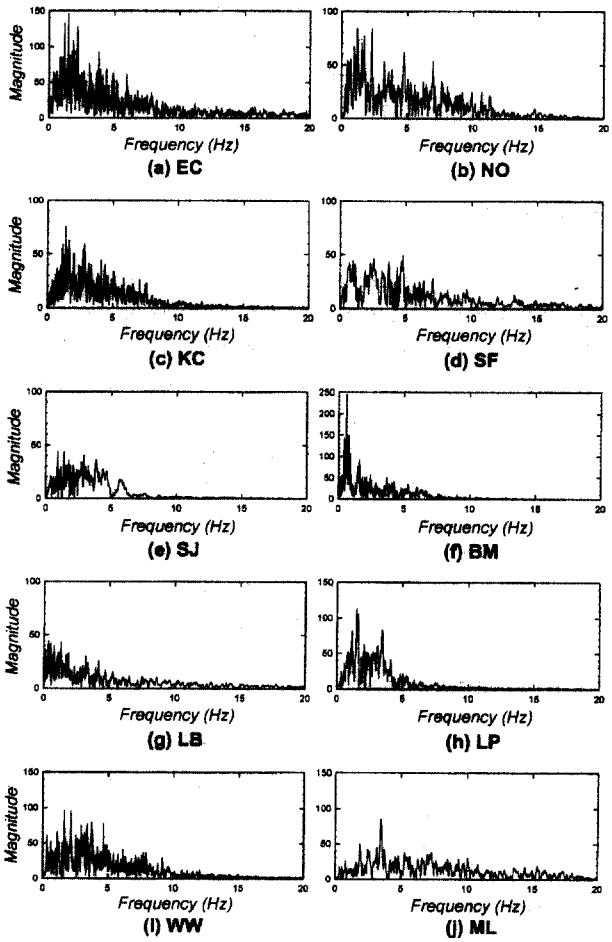


Fig. 5. Frequency components of ground accelerations.

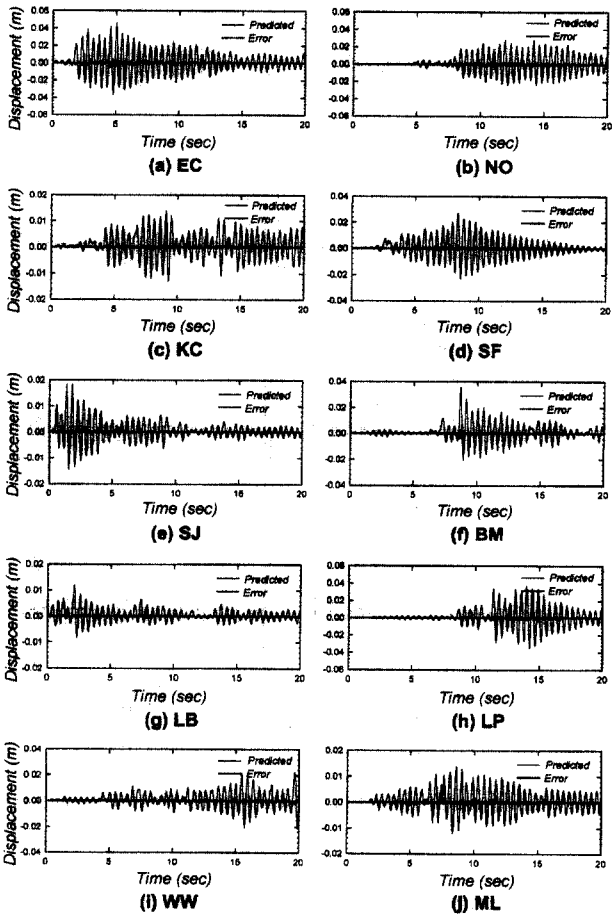


Fig. 7. Identified responses and errors (without noise).

Table 3. SAEN without noise (-10^{-2})

Load	EC	NO	KC	SF	SJ
SAEN	2.24	2.69	3.63	2.46	3.15
Load	BM	LB	LP	WW	ML
SAEN	2.26	4.17	2.25	2.96	3.85

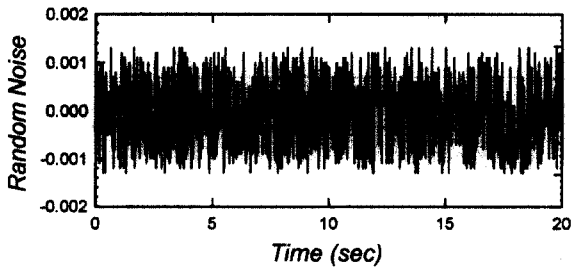


Fig. 8. Random noise signal.

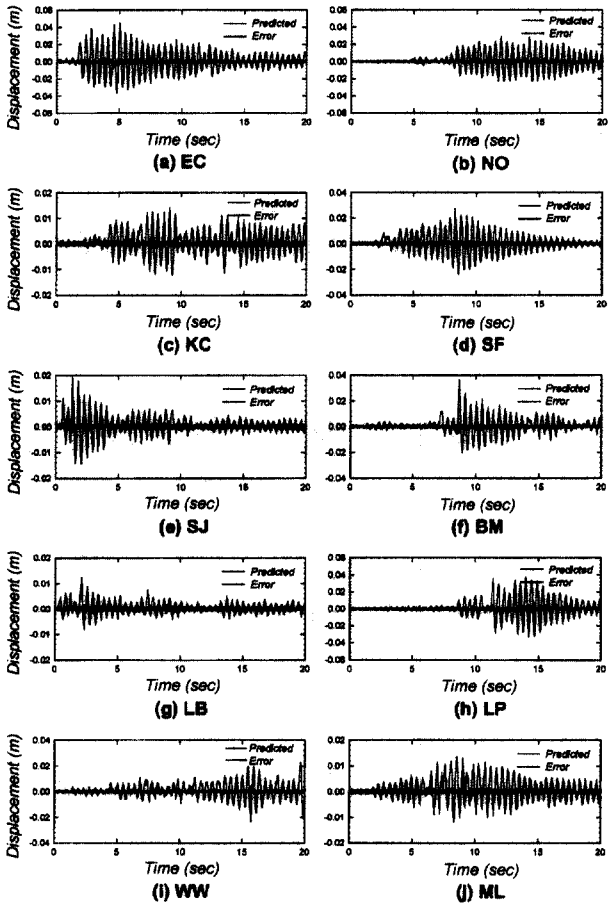


Fig. 9. Identified responses and errors (with noise).

expected, the learning speed was very fast and the convergence was excellent.

4.4 Identification without Noise

The performance of the trained CMAC was verified for the case of earthquakes which are not included in the learning. Fig. 7 shows the estimation results. The estimation was successful during not only trained earthquakes but untrained earthquakes. *SAEN* for the each case is summarized in Table 3. In this case, *SAENs* are all below 1.5.

Table 4. SAEN with 3% noise (-10^{-2})

Load	EC	NO	KC	SF	SJ
SAEN	1.26	1.47	1.15	0.79	0.91
Load	BM	LB	LP	WW	ML
SAEN	1.07	1.04	0.99	1.27	1.38

4.5 Identification with Noise

To test the robustness of the trained CMAC to signal noise, randomly generated noises were added to all the inputs of CMAC. The amplitudes of all random noises were set to 3% of the maximum of the input signals. A typical random noise signal added to the delayed signal of the displacement due to EC is shown in Fig. 8.

Fig. 9 shows the estimation results when the inputs have signal noises. Although the inputs of CMAC are contaminated with random noises, no remarkable error can be seen in the estimation. *SAEN* values, in this case, are listed in Table 4. The maximum of *SAEN* is the case of LB. However in other cases *SAENs* are below 4.0.

5. Conclusion

The systematic procedure of CMAC including the moving input space, the addressing, the training, and the calculating the output of CMAC were presented. Using CMAC, the identification of a dynamic structural model was accomplished. In the numerical example, it was shown that CMAC learned the dynamic response of structure very quickly. The responses under earthquakes that were not included in the training stage were predicted very accurately. It was also shown that the trained CMAC is robust to signal noises. Finally, it can be said that CMAC is also a very promising tool in structural engineering.

Acknowledgement

This work was partially supported by grant No. 97-0601-0301-3 from the Basic Research program of the Korea Science & Engineering Foundation (KOSEF). The financial support is gratefully acknowledged.

References

1. Albus, J. S. (1975). "A new approach to manipulator control-

ler: the cerebellar model articulation controller (CMAC)." *J. Dyn. Sys., Measurement & Control*, Trans. ASME, Vol. 97, No. 3, pp. 220-227.
2. Bani-Hani, K. and Ghaboussi, J. (1998). "Nonlinear structural control using neural networks." *J. Engrg. Mech. ASCE*, Vol. 124, No. 3, pp. 319-327.
3. Bani-Hani, K. and Ghaboussi, J. (1998). "Neural networks for structural control of a benchmark problem, active tendon system." *Earthquake Engrg. and Struct. Dynamics*, Vol. 27, pp. 1225-1245.
4. Chassiakos, A. G. and Masri, S. F. (1996). "Modelling unknown structural systems through the use of neural networks." *Earthquake Engrg. and Struct. Dynamics*, Vol. 25, pp. 117-128.
5. Chassiakos, A. G., Masri, S. F., Smyth, A. W. and Caughey T. K. (1998). "On-line identification of hysteretic systems." *J. Appl. Mech.*, Trans. ASME, Vol. 65, March, pp. 194-203.
6. Chen, H. M., Qi, G. Z., Yang, J. C. S. and Amini, F. (1995). "Neural network for structural dynamic model identification." *J. Engrg. Mech. ASCE*, Vol. 121, No. 12, pp. 1377-1381.
7. Chen, H. M., Tsai, K. H., Qi, G. Z. and Yang, J. C. S. (1995). "Neural network for structural control." *J. Computing in Civ. Engrg. ASCE*, Vol. 9, No. 2, pp. 168-176.
8. Chiang, C. T and Lin, C. S. (1996). "CMAC with general basis functions." *Neural Networks*, Vol. 9, No. 7, pp. 1199-1211.
9. Francisco, J. G., Anibal, R. F. and Antonio, A. (1998). "Generalizing CMAC architecture and training." *IEEE Trans. Neural Networks*, Vol. 9, No. 6, 1509-1514.
10. Ghaboussi, J. and Joghataie, A. (1995). "Active control of structures using neural networks." *J. Engrg. Mech. ASCE*, Vol. 121, No. 4, pp. 555-567.
11. Hama, H., Xing, C. and Liu, Z. (1998). "New high order association memory system based on Newtons forward interpolation." *IEICE Trans. Fundamentals*, E81-A(12), pp. 2688-2693.
12. He, Y. and Wu, J. (1998). "Control of structural seismic response by self-recurrent neural network (SRNN)." *Earthquake Engrg. and Struct. Dynamics*, Vol. 27, pp. 641-648.
13. Hung, S. L. and Jan, J. C. (1999). "MS_CMAC neural network learning model in structural engineering." *J. Computing in Civ. Engrg. ASCE*, Vol. 13, No. 1, pp. 1-11.
14. Kim, J. T., Jung, H. J. and Lee, I. W. (2000). "Optimal structural control using neural networks." *J. Engrg. Mech. ASCE*, Vol. 126, No. 2, pp. 201-205.
15. Koo, K. M. and Kim, J. H. (1994). "Robust control of robot manipulators with parametric uncertainty." *IEEE Trans. Automatic Control*, Vol. 39, No. 6, pp. 1230-1233.
16. Michael, M., James, S. and Cho, D. I. (1994). "Neural network control of automotive fuel-injection systems." *IEEE Control Systems*, June, pp. 31-36.
17. Nikzad, K., Ghaboussi, J. and Paul, S. L. (1996). "Actuator dynamics and delay compensation using neurocontrollers." *J. Engrg. Mech. ASCE*, Vol. 122, No. 10, pp. 966-975.
18. Soong, T. T. (1990). *Active structural control: theory and practice*, John Wiley & Sons, New York, N. Y.
19. Stephen, H. L., David, A. H. and Jack, J. G. (1992). "Theory and development of higher-order CMAC neural networks." *IEEE Control Systems*, April, pp. 23-30.
20. Tang, Y. (1996). "Active control of SDF systems using artificial neural networks." *Comp. and Struct.*, Vol. 60, No. 5, pp. 695-703.