'89 KACC 1989. 10. 27~28

LEARNING PERFORMANCE AND DESIGN OF AN ADAPTIVE CONTROL FUNCTION GENERATOR: CMAC (Cerebellar Model Arithmetic Controller)

Heon Hwang and Dong Y. Choi

Robotics Lab., Automation Eng. Dept.
Korea Institute of Machinery and Metals
Sangnam-dong 66, Changwon, Kyungnam, Korea

ABSTRACT

As an adaptive control function generator, the CMAC (Cerebellar Model Arithmetic or Articulated Controller) based learning control has drawn a great attention to realize a rather robust real-time manipulator control under the various uncertainties. There remain, however, inherent problems to be solved in the CMAC application to robot motion control or perception of sensory information. To apply the CMAC to the various unmodeled or modeled systems more efficiently, It is necessary to analyze the effects of the CMAC control parameters on the trained net.

Although the CMAC control parameters such as size of the quantizing block, learning gain, input offset, and ranges of input variables play a key role in the learning performance and system memory requirement, these have not been fully investigated yet. These parameters should be determined, of course, considering the shape of the desired function to be trained and learning algorithms applied.

In this paper, the interrelation of these parameters with learning performance is investigated under the basic learning schemes presented by authors. Since an analytic approach only seems to be very difficult and even impossible for this purpose, various simulations have been performed with prespecified functions and their results were analyzed. A general step following design guide was set up according to the various simulation results.

1. INTRODUCTION

The adaptive controller for complicated robotic systems is confined mostly to the experimental stage because it requires heavy computing of real time parameter identification based on some performance criteria and management of sensitivity on sensory inputs. It usually involves a complex algorithm. For this reason, the robust adaptive controller based on the biological structure and function have drawn a great attention recently. How to achieve a great degree of the robustness, adaptation, real time control and easy learning is the major focus on this area.

Research and application of the artificial neural net to the robot system control and perception have become widely spreaded for a few years around the world with an aim of realizing the structure and function of biological organisms especially of human brain[1]. Through the massively parallel connection of processing elements with learning capability and fault-tolerance, the neural net approach is known to overcome the limitations and weakness posed by the conventional sequential information processing[2].

Anatomical and neurophisical studies of the cerebellum have led to a theory concerning the functional operations of the cerebellum. Some basic principles of how the cerebellum accomplishes motor behavior have been organized into a mathematical model, Cerebellar Model Articulation or Arithmetic Controller(CMAC) by Albus[3,4,5]. Since then, research on the CMAC based general learning controller has been attempted to control the various systems including robot because of the simple structured nature of the CMAC[2,6-9].

The learning convergence of the CMAC was proved by authors identifying the network as a kind of one layer linear associator having a linear activation function[10]. Two types of basic learning algorithms of the CMAC, sequential error correction(SEC) and random error correction(REC) under delta rule have been proposed and analyzed with different learning gains[10].

To apply the CMAC to the various unmodeled or modeled systems more efficiently, It is necessary to analyze the effects of the CMAC control parameters on the trained net. Parameters such as size of the quantizing block K, learning gain G, input offset, and ranges of input variables play a key role in the learning performance and system memory requirement. However, these have not been fully investigated yet. Values of control parameters are chosen in most cases on an ad hoc basis. Values of parameters should be determined, of course, by also considering the shape of desired function to be trained and learning algorithms applied.

In this paper, with predetermined input variable offset and ranges (refer to [10] for the offset and uniform quantized scheme handling various input ranges), the interrelation of quantizing size K and learning gain G is investigated with learning performance under two types of learning schemes, SEC and REC. The system memory required for the CMAC application depends on the quantizing size K and ranges of the CMAC input variables. The formular for the required system memory refers to the refernce [10].

Since an analytic approach only seems to be very difficult and even impossible for this purpose, various simulations have been performed with quite different shaped model functions and their results were analyzed and some of them are presented. A general step following design guide was set up through the characteristics of the CMAC network analyzed theoretically and experimentally.

2. SIMULATION

2.1 Basic Learning Algorithm

Equivalent learning period

Desired function to be learned:

$$P = Sin(X)$$

Input range: $0 \leq X < 360$ (deg)

Interval of sampled node inputs: 5 deg

Selected size of quantizing block:

5, 10, 20, 30, 40, 60

CMAC offset: 1 = 1 deg

(1) Batch Sequential Error Correction(SEC)

Learning gains which avoids divergence at the initial training are selected from

0.1 ~ 1.0 by 0.1

0.09 ~ 0.01 by 0.01

0.009 ~ 0.001 by 0.01

Number of training epoch: 0 ~ 100

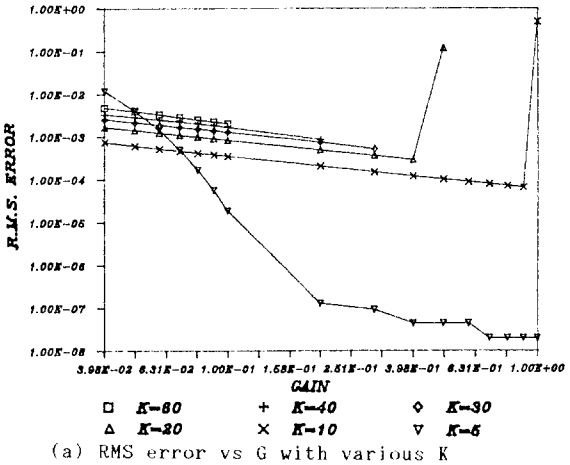
Since the sampled node interval was specified as 5 deg, the case of K=5 does not have any interference effect at all generating the orthogonal CMAC mapped binary pattern vectors. As a result, the function is trained completely by one shot when gain is one. It is not shown clearly in fig.1a because it was plotted at the training epoch of 100.

As K increases from five, values of the initial gains, which avoid the divergence, decrease. Gains located closely to the initial divergent gain converge at the early stage but show a diverging trend when the learning epoch increases[10].

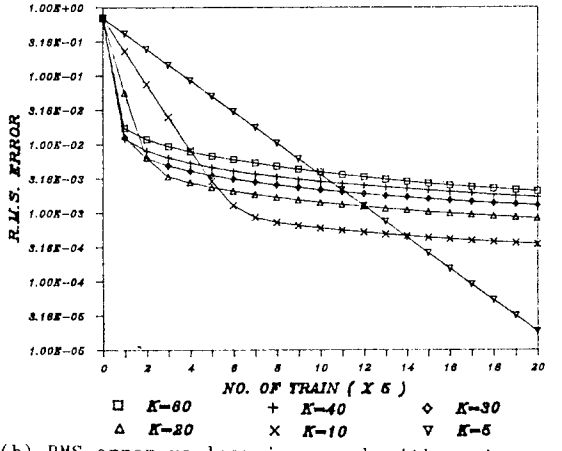
The distributing and interference effect can be seen in fig.1b. As values of K increases, the corrected delta values are distributed over the input space broadly resulting fast rms error convergence. However, the converging rate weakens earlier than smaller values of K as learning epoch increases because of the interference.

When K is equal to 5, the converging slope is logarithmically straight because of the orthogonality of the pattern vectors. The slope is getting steep as learning gain increases up to the infinite value when the value of gain is one.

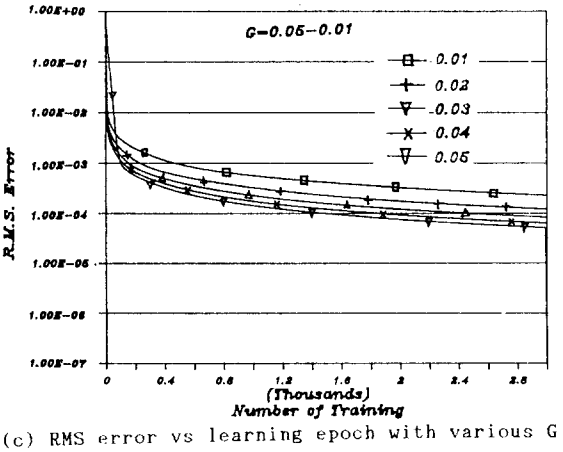
Fig.1c shows the extended learning epoch up to 3000 with K=40. The parallel trend of the slopes for the various gains is maintained until it reaches its global minimum. It is not plotted but converged learned rms errors from the unlimited learning epoch with G=0.05 and G=0.01 were 1.395131E-6 at learning epoch of 19000 and 7.898105E-6 at learning epoch of 55500 respectively. In a case of the batch SEC, we can see that with a fixed K higher value of gains converges faster and reaches lower global minimum once it is within the converging range.



(a) RMS error vs G with various K (learning epoch=100)



(b) RMS error vs learning epoch with various K (G=0.1)



(c) RMS error vs learning epoch with various G (K=40)

Fig.1 Batch type SEC learning for $P=Sin(X)$.

(2) On-Line Sequential Error Correction(SEC)

Selected learning gain:

0.2～1.0 by 0.2

Number of training epoch: 0～100

As batch type SEC does, on-line SEC has the same learning effect after one epoch training with K=5 as shown in fig.2a and fig.2b. Fig.2a shows the distributing and interference effects of K at the learning epoch of 100. When the value of K is large, the learning effect is interfered more with large values of G. However, this fact is rather slowly occured with small values of K.

With smaller gain, the learned performance is rather low because of the less distributing effects. From this we can see that as K increases the shape of the learned rms error trend becomes a rather bowl type. This bowl type shape is flattened as learning epoch increases more and more.

At the early stage of learning, although it is not shown clearly in Fig.2b because of the plotting interval of learning epoch, large K shows a better learning. It is noted, however, the size of K should be proper to the shape of the function to be trained.

Fig.2c shows trends of the various gains with a fixed K of 40 when learning epoch is extended up to 3000. It shoud be noted the smaller gain catches larger gain as learning epoch increases. This is contrary to the result of the batch type SEC. Note, however, practically the learning period is also critical to the system performance as well. For this reason, it is not recommended to reduce the gain value small based on the result of fig.2c when applying on-line SEC learning.

(3) Random Error Correction(REC)

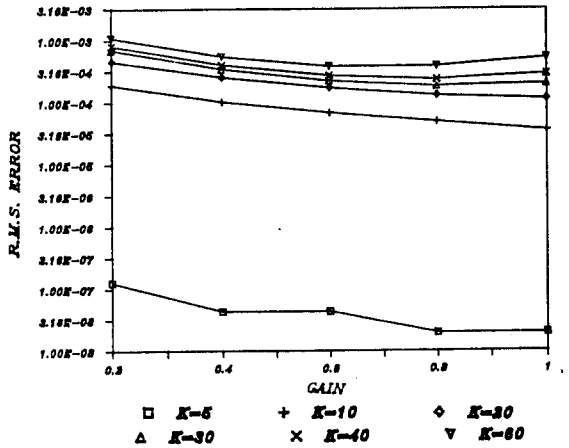
Selected learning gain:

0.2～1.0 by 0.2

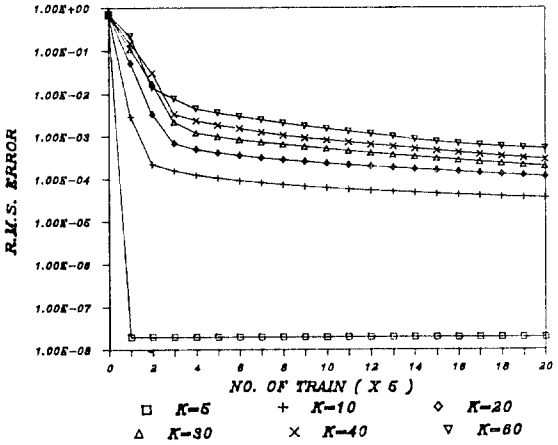
Number of training : 0～6000

At the first glance, Fig.3a seems to show a chaotic behavior of trends with resepect to G and K. The REC learning has the similar behavior as SEC learning except the nonaccumulating property caused by its random selction of input patterns.

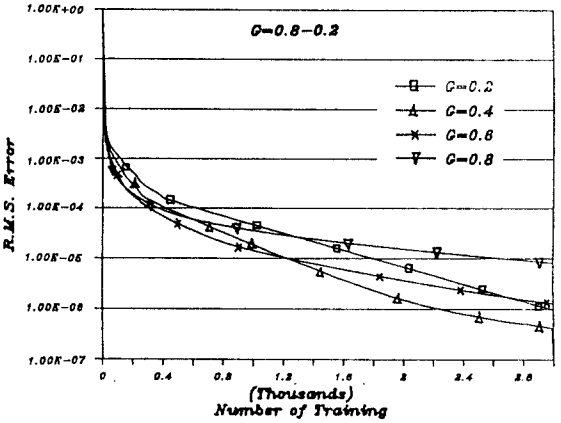
With a relatively small K, the interference is not serious as gain increases at learning number of 6000. The distributing effect is



(a) RMS error vs G with various K (learning epoch=100)



(b) RMS error vs learning epoch with various K (G=1.0)



(c) RMS error vs learning epoch with various G (K=40)

Fig.2 On-line type SEC learning for $P=Sin(X)$.

rather good at high gains. However, this effect is interfered by large values of K when the shape of function has varying curvature over K. Trends of the performance from K=10 to K=60 can be explained by the amount of the interaction between the distributing and the interference with various gains at this specific learned point.

Although it is not shown clearly the number of training exactly in fig.3b, all sampled node inputs are selected at least once at training number of 403 with K=5. It is due to the characteristics of REC learning which randomly selects inputs among the sampled nodes.

2.2 Functional Shape

Equivalent learning period

Learning algorithms:

On-line sequential error correction

Random error correction

Selected Learning gain:

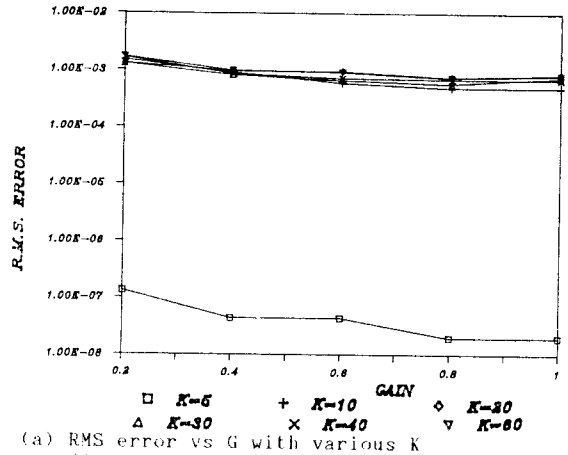
0.2～0.8 by 0.2

Interval of sampled node inputs: 15

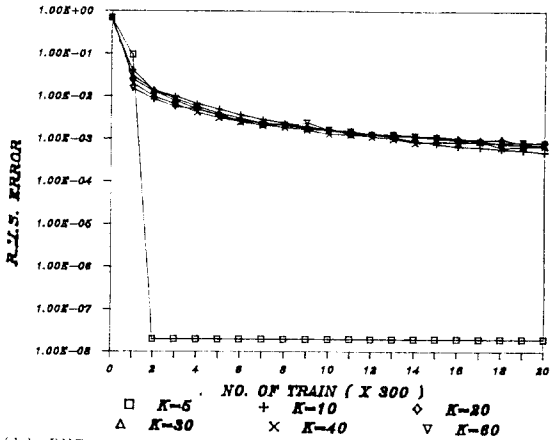
Selected size of quantizing block:

20, 30, 40, 60, 80, 120

CMAC offset: 1



(a) RMS error vs G with various K (learning numer=6000)



(b) RMS error vs train number with various K (G=1.0)

Fig.3 REC learning for $P=Sin(X)$.

(1) Function 1

Desired function to be learned(fig.4a):

$$P = Cos(X)Cos(y)$$

Input range: $X,Y \in [0,180]$ (deg)

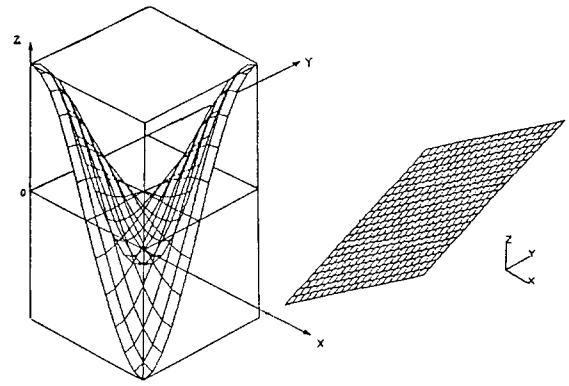
◇ On-line SEC:

Trends of learned rms errors were investigated up to learning epoch of 200. Figure 5a and 5b show the rms error versus learning gain G for various K values at learning epoch of 200 and the rms error versus learning epoch for various K values at G of 0.8 respectively. Since the value of the sampled node interval is 15, with K=20 the learned performance is excellent.

The rms error does not vary significantly as G varies or learning epoch increases with relatively large K. This is due to the shape of the function to be learned. In other words, the function to be trained has a rather steep variation of values over the distributed region defined by K. As K increases, the value of G does not improve the system performance as it is desired. The difference between K=20 and K=30 will be reduced if the sampled interval is defined less than 15, for example 5.

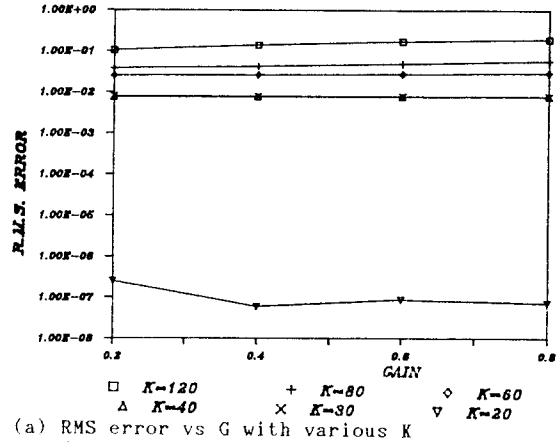
◇ REC:

Trends of learned rms errors were investigated up to learning number of 70000 which results into the equivalent learning period of on-line SEC. As shown in fig.6,

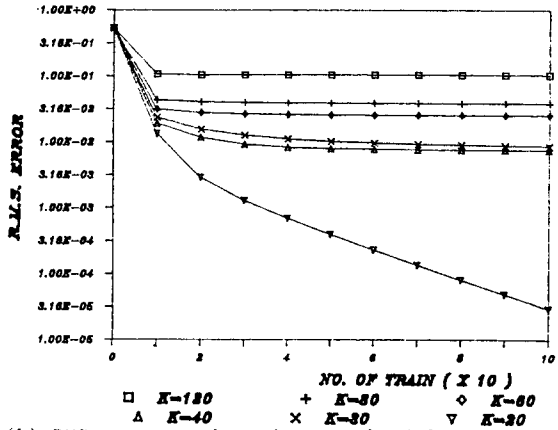


(a) $P = Cos(X)Cos(Y)$ (b) $P = 10X - 2Y$

Fig.4 Desired functions to be trained.



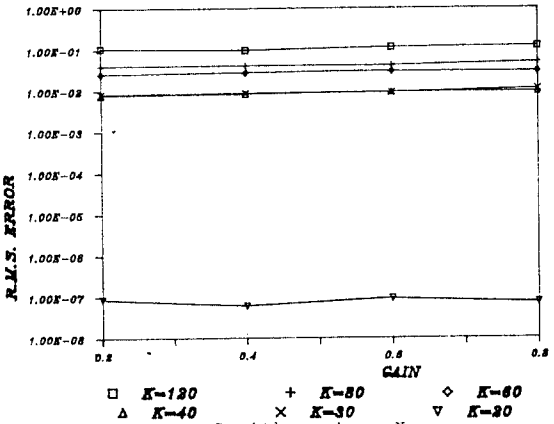
(a) RMS error vs G with various K (learning epoch=200)



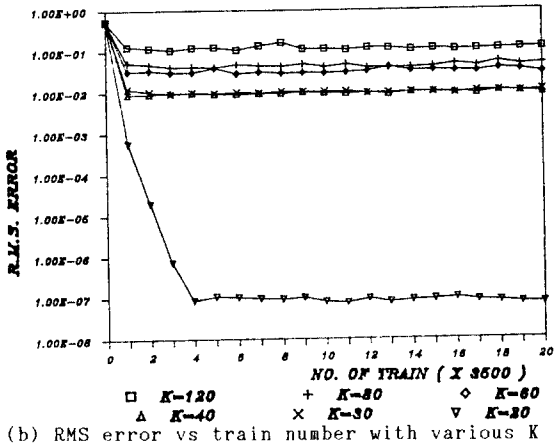
(b) RMS error vs learning epoch with various K (G=0.2)

Fig.5 On-line SEC learning for $P=Cos(X)Cos(Y)$.

except the oscillating behavior of learned rms error with an increase of learning number, the REC learning shows a similar learned performance and pattern of rms error versus G and K.



(a) RMS error vs G with various K (learning numer=70000)



(b) RMS error vs train number with various K (G=0.8)

Fig.6 REC learning for $P=Cos(X)Cos(Y)$.

(2) Function 2

Desired function to be learned(fig.4b):

$$P = 10\,X - 2\,Y$$

Input range: $X,Y \in [-150,150]$

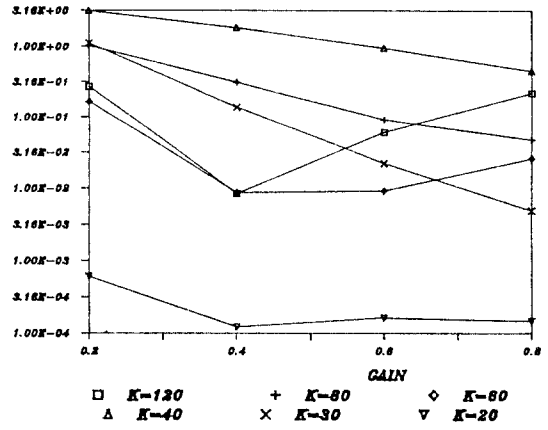
◇ On-line SEC:

Although the overall shape of the desired function is flat, with large value of K the effect of the interference caused by the sequentially accumulated learning errors increases when learning gain is high.

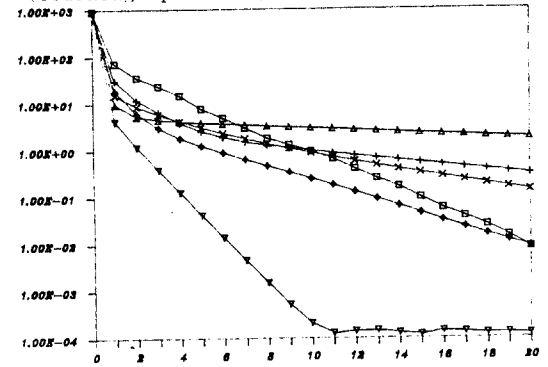
With small values of K such as 30 and 40, larger gain shows better performance because of the small distributing effect with little interference as shown in fig.7a, at the learning epoch of 200. As K increases, the combined effect of the distributing and interference makes the learned system be oscillatory with respect to gain.

It is shown that logarithmically straight variation occurs at K=80 with respect to G. However, the learned performance is quite poor compared with K=60. It is expected as learning epoch increases the oscillating behavior will occur.

Similarly to the case of function type 1, at K=20 since the interval of the sampled node inputs is 15, the performance is rather greater then other K values. Fig.7b shows trends of rms error versus learning epoch at G=0.4 with various K values.



(a) RMS error vs G with various K (learning epoch=200)



(b) RMS error vs learning epoch with various K (G=0.4)

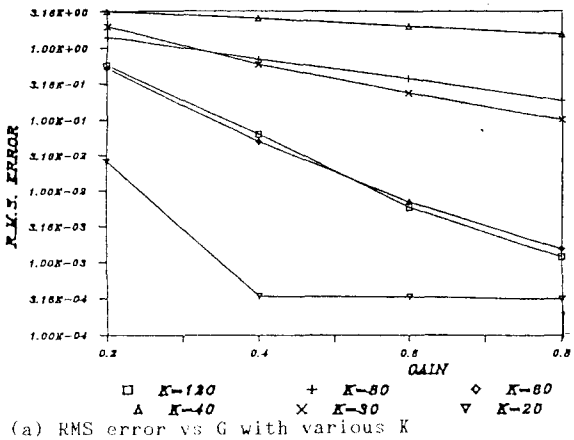
Fig.7 On-line SEC learning for $P=10X-2Y$.

◇ REC:

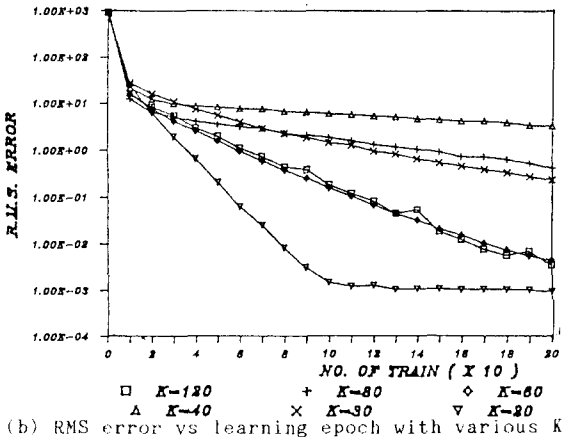
Since input nodes are selected randomly and also desired function is a rather flat shape, the correction amount distributed by the K is not accumulated that much compared to the on-line SEC even large value of G. Fig.8a shows logarithmically linear trend of the performance improval as gain increases. Although it is not shown here, it is expected the effect of the gain gets smaller as number of learning increases. Other trends can be analyzed similarly as on-line SEC. Fig.8b shows trends of rms error versus learning epoch at G=0.8 with various K values.

3. DESIGN GUIDE

The input offset of the CMAC is related to the continuity property. With a fixed input space, as the offset of the quantized block becomes more precise, the mapped non-dimensional CMAC input space gets bigger. As the CMAC input space gets bigger, the size of the quantizing block should be properly increased to enlarge the distributing effect of the error correction and to reduce the required system memory. Given sampled node inputs, as the offset between the quantized blocks gets preciser, the CMAC can generate more distinct linear interpolated results at the untrained intermediate nodes.



(a) RMS error vs G with various K (learning number=70000)



(b) RMS error vs learning epoch with various K (G=0.8)

Fig.8 On-line SEC learning for $P=10X-2Y$.

The size of the quantizing block, K plays a key role in the storage and retrieval of the learned data in a distribute manner. In fact, the CMAC learns unmodeled system behavior by slicing desired fuction, which is usually nonlinear, into many precise linear segments. The size of K should be determined considering the slope of the function to be trained. Generally when the slope of the function is steep over the mapped input space, the value of K should be decreased and vice versa. With an unproper size of K, learning performance of the CMAC can not be improved by increasing training number or by varying learning gain.

With N number of different inputs, ideally the system can do its best with N number of memory. The reduction of the memory size, however, is one of the important merits in applying neuro nets while maintaining certain fault-tolerance. Learning gain has a function of the moderate adjustment toward the minimum of the LMS(Least Mean Square) error.

The way of learning is also quite critial to its performance. The designer should decide which learning should fit to their application best among REC, SEC.

REC learning is good for handling a quite large input space and off-line generating of the desired system behavior such as solving robot inverse kinematics or dynamics and calibration etc. While the SEC learning is good for the type of applications that learning is required with on-line error measurement while a robot is in motion or after one trial of task motion such as robot trajectory control.

The CMAC controller can be implemented as a reference function generator or an adaptive control function generator. Considering human's motor behavior, many sub-CMAC controllers can be connected hierarchically according to the level of the object to be controlled. The selective on-line learning can be implemented depending on whether situation of the task environment is normal or abnormal.

When applying the CMAC to learn the unmodeled system behavior, following simple design step is suggested from the analyzed characteristics of the CMAC network.

① set offset a little preciser than the anticipated.
② specify sampled node inputs.
③ set up regular CMAC input variable space using the proposed quantizing scheme[10].
④ size of K is selected 1/3~2/5 of CMAC input variable range.
⑤ learning gain is selected as 0.4 to 0.8 for the regular REC and SEC training.
⑥ number of learning epoch for SEC is determined via on-line checking of the system improvement at every epoch. In the case of REC, checking of the system improvement is suggested at every 10 times number of the sampled node inputs.
⑦ If the learned performance is not good, reduce K by half and increase G by about 0.05~0.1 and go to step ④ and repeat.
⑧ stop

4. CONCLUSION

The interrelation of control parameters specially for the quantizing value K and learning gain G was investigated by analyzing trained results of the various model functions with basic learning algorithms. A general step following design guide was set up from the characteristic of the CMAC analyzed theoretically and experimentally.

The CMAC system controller can be extended to contol the integrated system behavior employing several sub-CMACs of different function generator and controller connected each other hierarchically in a closed loop. Research and application of this concept for the task of the sensor integrated robot system control is widely open.

The development of the nonlinear CMAC meta connected network for a function generator and a decision controller is undergone by authors to overcome the limitations posed by the linearity of the CMAC network.

6. ACKNOWLEDGMENT

The authors gratefully acknowledge the Korea Institute of Machinery and Metals' Basic and Advanced Research Assistance Program.

8. REFERENCES

[1] DARPA Neural Network Study, AFCEA Int. Press, 1987, pp.123-133, 445-450.

[2] Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1 and 2, D.E. Rumelhart and J.L. McClelland(Eds), Cambridge, MA:MIT Press, 1987.

[3] Albus, J.S.,'A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller(CMAC),' Journal of Dynamic Systems, Measurement, and Control, Tran. of the ASME, Vol.97, No.3, Sept. 1975, pp.220-227

[4] Albus, J.S.,'Data Storage in the Cerebella Model Articulation Controller(CMAC) ,'Journal of Dynamic Syatems, Measurement, and Control, Tran. of the ASME, Vol. 97, No.3, Sept. 1975, pp.228-233.

[5] Albus, J.S.,'Mechanisms of Planning and Problem Solving in The Brain,' Mathematical Biosciences, 45, 1979, pp.247-291.

[6] Camana, P.C.,'A Study of Physiologically Motivated Mathematical Models for Human Postural Control,' Ph.D. Thesis, Dept. of Electrical Engineering, Ohio State University, 1977.

[7] Miller III, W.T.,'Sensor-Based Control of Robotic Manipulators Using A General Learning Algorithm,' IEEE Journal of Robotics and Automation, Vol. RA-3,No.2,April 1987, pp.62-75.

[8] Miller III, W.T., F.H. Glantz, and L.G. Kraft, 'Application of A General Learning Algorithm to The Control of Robotic Manipulators,' Int. Journal of Robotics Research Vol.6, No.2, Summer 1987, pp.84-98.

[9]Manglevkedakar, Sunit,'An Adaptive Hierarchical Model for Computer Vision,' MSME Thesis, Dept. of Mechanical Eng. Louisiana State University, 1986.

[10]Hwang, H. and D.Y. Choi,'On Learning of CMAC for Manipulator Control,' to be published at Korean Automatic Control Conf. Int. session Seoul, Korea, Oct. 1989.