航空技術學院學報　第二十二卷　第 51-60 頁(民國一一二年)
Journal of Air Force Institute of Technology, Vol. 22, pp. 51-60, 2023

Introduction and Application of Cerebellar Model Articulation Controller Fast Learning Algorithm

Ho-Nien Shou　Kun-Cheng Tsai　Ming-Feng Zhong

Abstract

The Cerebellar Model Articulation Controller (CMAC) is widely used in many fields due to its local generalization performance and fast convergence rate. CMAC is a type of neural network based on local learning, which possesses strong local generalization ability and fast learning and convergence characteristics, making it suitable for real-time control systems. A high-performance real-time control strategy for complex systems has been designed based on a new CMAC. This scheme does not rely on the mathematical model of the system and exhibits strong robustness and versatility. Simulation results demonstrate the feasibility and effectiveness of the proposed scheme.

Keyword: Cerebellar Model Articulation Controller, local learning, real-time control system

1.Introduction

In the process of the development of modern artificial intelligence (Artificial Intelligence, AI), all of them are closely related to the learning process of human cognition, many of which are based on the human learning methods, such as: Neural Network, CMAC, Genetic algorithm (GA), Particle Swarm Optimization(PSO) and Perceptual Learning Algorithms (PLA), etc., are all modern methods of artificial learning, are very similar to the human learning patterns and the laws of nature. In the literature, the Cerebellar Model Articulation Controller (CMAC) is a neural network model proposed by Albus [1], based on the structural biological model of the cerebellum. The development process of cerebellar model was first presented in 1969 Marr[2] the cerebellar theory of the cerebellar cortex, which is a three-layer structure of molecular layer, purkinje cell layer and granular layer, respectively. Can be stored in layers that come from the brain and various sensory organs of the body. According to the Marr model of cerebellum, Albus[1,3,4,5] developed a cerebellar mathematical model resembling cerebellar cortex operation structure as CMAC, a class of neural networks that can store input messages and output messages, and have learning functions, update the weights of memory cells through repetitive learning processes, To achieve the desired output, CMAC is essentially the Artificial neural network memory Lookup Table, which gives the cerebellum model the following advantages:

(i)Fast learning convergence: within the framework of the cerebellum model, learning-related memory cells are programmed to store information, so that the actual output of the cerebellum model and the expected output error feedback can be used to adjust the weight of the various levels of memory, and establish the relationship between input and output, because in each training process only need to change the local memory weight, so the convergence speed of the general neural network class.

(ii)Regional generalization: at least one memory cell

is mapped to each of the two adjacent reference states, that is, at least two memory cells are mapped to each state, and information can be stored in a distributed manner.

(iii)Approximation or recognition can be applied to nonlinear systems: compared to other neural networks, the microcephaly model is simple in construction and easy to learn, and has a large number of nonlinear square root functions.

Based on a detailed analysis of the conventional CMAC, and considering the reliability of weight adjustment of each storage unit, a Credit Assessment-CMAC(CA-CMAC) learning algorithm is proposed. However, the balance between “new knowledge learning” and “old knowledge forgetting” is not further considered in the literature[6].In this paper, Improved Credit Assign-CMAC(ICA-CMAC) learning algorithm[7] is designed. This improved algorithm improves the conventional CMAC from the way of weight adjustment and maintenance, and effectively improves the learning speed and precision of the network. Both the conventional CMAC and ICA-CMAC networks, the forward mapping should be discrete, coding, quantization and hashing mapping, in the input variable is low dimension space, the computational complexity is not high, the impact is not obvious. When the input variable is in the high dimension space, with the increase of network complexity, the result is that the computation quantity is increased, the data collision caused by the hashing mapping reduces the learning speed and precision of the network. In this paper, the concept of nuclear space is introduced, the IK-CMAC (improved KERNEL-CMAC) network model is designed, and the feature space is implicitly defined by the kernel-air question, which can avoid the complex operation in the feature space. The method of using kernel space can improve the network modeling ability and reduce the effect of hashing image without increasing the complexity of the network. The following will be the normal CMAC, CA- CMAC, ICA-CMAC, IK-CMAC derivation, increase learning speed and accuracy. It can be seen from the simulation results that ICA-CMAC has a faster learning speed and higher learning accuracy, especially in the initial stage of network learning its error is much smaller than that of conventional CMAC network, and also smaller than CA-CMAC network; IK-CMAC in the early stage of network learning error is significantly smaller than ICA-CMAC network, the margin of error change is smooth. Due to the above advantages, it has been widely received attention in recent years by researchers in various fields, such as motor control [8], digital signal channel and other fields [9], literature [10] for the heart disease database as an example, the design of a two-dimensional cerebellar model (cerebellar model) Heart disease classification system, Using only some of the data of the database as input, to adjust the weight of the memory cells at all levels of the cerebellar model, in order to find the classification of the disease in the heart database.

The structure of the paper is divided into five sections: the first section is “Introduction”, the background and research significance of CMAC neural network learning algorithm are introduced, the present situation of domestic and international research and the main research contents of this paper. The second section is “CMAC Neural network Learning Algorithm Research”, analysis of conventional CMAC, CA- CMAC, ICA-CMAC, IK-CMAC Learning algorithm: and discuss the comparison, to study its rapidity and accuracy. The third section is the address collision problem after “Hashing Transformation”, in which the address

function is used to produce the symbol of the desired storage unit, it is a concise address method, and it is a succinct addressing way, and there is no data collision. The fourth section is “Simulation result analysis”, aiming at different physical system modes, the system is simulated and analyzed by CMAC neural network learning algorithm. In the last section, “Summing up and looking forward”, this paper sums up the research work of the subject, and puts forward some ideas and ideas for the future research.

2.The structure and principle of CMAC

2.1 Structure of CMAC

The conventional CMAC structure is shown in Fig. 1, in which $\boldsymbol{S}$ represents the p -dimensional input state space, and $\boldsymbol{A}$ is a storage area with n units (also known as an associated space or conceptual memory space). The input vectors of the CMAC network are input to the $\boldsymbol{s}_i = \left(s_{i1}, s_{i2}, \cdots s_{ip}\right)^T$ in the state space $\boldsymbol{S}$ in the p -dimension indicates that the corresponding output vector is represented by $\boldsymbol{y}_i = \boldsymbol{F}\left(s_{i1}, s_{i2}, \cdots s_{ip}\right)^T$, and that a point $\boldsymbol{s}_i$ of the $i = 1, 2, 3$ input space in the graph will activate the n elements of a at the same time ($n = 4$ in Fig. 1). So that it is 1 at the same time and most of the other elements are 0, and the output y of the network is the cumulative sum of the corresponding weights for the 4 active cells in $\boldsymbol{A}$. N is called the generalization parameter, the response network generalization ability size, also can regard it as the signal detection unit's feeling field size. For CMAC, its working process generally includes two aspects: first, the result output calculation and error generation stage, the second is the weight adjustment stage.

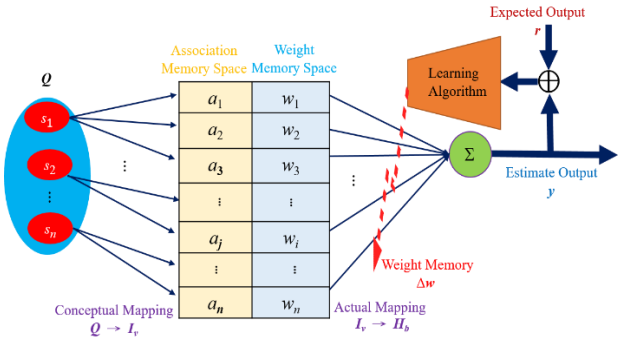

Fig. 1 The conventional CMAC structure

2.2 Principle of CMAC

In general, when applied, the components of an input vector come from different sensors, the value of which is more than the analog, and each element in $\boldsymbol{A}$ takes only 0 or 1 of the two values. In order to map the point of $\boldsymbol{S}$ space to the discrete point of $\boldsymbol{A}$ space, the analog quantity $\boldsymbol{s}_i$ must be quantified so that it becomes the discrete point of the input state space. If each component of the input vector $\boldsymbol{S}$ can be quantified to q grade, the p component can be combined into a possible state q^p of the input state space $\boldsymbol{s}_i, i = 1, 2, \cdots, q^p$. Each state $\boldsymbol{s}_i$ is mapped to a set $\boldsymbol{A}_i$ of the $\boldsymbol{A}$ space store, and the N_L elements of $\boldsymbol{A}_i$ are all 1. From Fig.1, it can be seen that the sample $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$ close in the $\boldsymbol{S}$ space are mapped to $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ in $\boldsymbol{A}$ by the intersection $\boldsymbol{A}_2 \cap \boldsymbol{A}_3$, that is, two of their corresponding four weights are the same, so the addition of the right values and the two computed outputs are also closer to the generalization function. Obviously, for a very distant sample $\boldsymbol{A}_1$ and $\boldsymbol{A}_3$, the $\boldsymbol{A}_1 \cap \boldsymbol{A}_3$ mapped to a is an empty set, and this generalization does not work, so it is a partial generalization. The closer the input space is to the input distance, the closer the elements of the

intersection are to the N_L, and the higher the intersection of the corresponding input samples in A is, the effect of clustering the similar samples. For each state of the S space, there is a unique mapping in the A space. The number of cells in the A storage area should be at least equal to the number of states in the S space, that is, $n \geq p^q$. To quantify each component of the 3 dimension input to 10 levels, $n \geq 1000$. For many real-world systems, p^q tends to be much larger than this number, but because most learning problems do not contain all possible input values, it is not actually necessary to store the p^q weights. A is a virtual memory address that corresponds to each virtual address and a sample point in the input state space. The hash code maps address space A with p^q storage units to a much smaller physical address connection s_i. For each input, only N_L units in A are 1, and the rest are all 0, so A is a sparse matrix. Hash encoding is a common technique for compressing sparse matrices by means of a program that produces random numbers. The address of A is used as a variable for programs that generate random numbers, and the random number is used as the address of A_i. A_i is much smaller than A because the resulting random number is limited to a smaller integer range. Obviously, compression from A to A_i is a random mapping of many pairs of few. In A_i, each sample has N_L random address corresponding to it, the weight value of N_L address is obtained by learning, its accumulation and that is as the output of CMAC. Its expression is Eq.(1)

$$y_i = \sum_{j=1}^{NL} w_j a_j(x), \quad i = 1, 2, \cdots, m \tag{1}$$

In Eq. (5-1), w_j is the weight of the *j* memory cell, and if $a_j(x)$ is active, its value is 1, otherwise 0, and only N_L memory cells have an effect on the output. The memory cells activated by similar inputs overlap to produce similar outputs, and the input not similar to produce similar outputs. Corresponding error expressions such as Eq.(2)

$$\Delta E_i = y_d - \sum_{j=1}^{N_L} w_j a_j(x), \quad i = 1, 2, \cdots, m \tag{2}$$

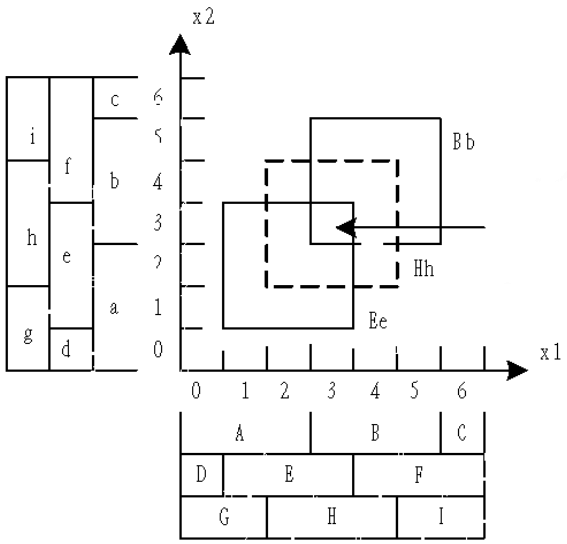

Fig. 2 Two-dimensional CMAC structure

2.3 The Weight Adjustment Stage of CMAC

The result output phase in the CMAC algorithm produces an actual output from the CMAC storage unit. The learning process updates the weights in the CMAC storage unit according to the error size of the expected output and the actual output. In the conventional CMAC algorithm, the error is equally distributed to all the activated storage units. Set s to a state, $w_j(t)$ is the weight that is stored in the j storage unit after the t iteration. The General CMAC Update $w_j(t)$ algorithm is:

$$w_j(t) = w_j(t-1) + \frac{\alpha}{N_L} a_j \left(y_d - \sum_{j=1}^{N_L} w_j(t-1) a_j(x) \right) \quad (3)$$

where y_d is the desired output of state $\boldsymbol{s}$, $\sum_{j=1}^{N_L} w_j(t-1) a_j(x)$ is the actual output of the state $\boldsymbol{s}$, α is the learning constant.

3.Improved learning algorithm for CMAC

In CMAC applications, the real-time requirements are generally higher. For example, online identification of nonlinear dynamic systems requires not only high accuracy, but also fast learning. However, conventional CMAC still requires multiple cycles to achieve a certain convergence accuracy, that is, although the conventional CMAC converges faster than BP network, but as online learning, it is still difficult to meet the requirements of its rapidity.

3.1 A balanced learning CMAC algorithm based on reliability assignment

In the weight learning adjustment of conventional CMAC learning algorithms, the error is equally distributed to each activated memory cell without considering the contribution rate of each activated memory cell to the error, that is, after t learning, the weight reliability of the activated memory cell with different adjustment times is still regarded as identical. The weight updating algorithm completely violates the concept of reliability assignment, so the weights learning algorithm will make the memory unit (whose weight value is high credibility) should be adjusted repeatedly if the weights are not adjusted or should be adjusted less. But the storage unit which contributes more to the error (its weight credibility is low), should make its weight worth to the larger adjustment, but in fact the weight learning adjustment is reduced. In order to achieve the predefined approximation accuracy, the network must study repeatedly, so that the learning efficiency of CMAC is reduced and the learning time is extended. In order to improve the learning speed of CMAC, a CA-CMAC algorithm based on the reliability assignment is proposed in the paper [11] on the basis of analyzing the normal CMAC weight adjustment rules and considering the credibility of the learned knowledge. Literature [12] on this basis, further considering the balance between the new knowledge “learning” and the Old knowledge “forgetting” in the network weight adjustment, a CMAC neural network learning algorithm based on “balanced learning” is proposed.

In Eq.(3), it is important to note that only the weights of those cells that are activated are updated. In the general algorithm above, the error is equally distributed to all cells that are activated, but after $(t-1)$ iteration, the initial cells already contain some of the previously learned knowledge. Not every cell has the same learning history, so these cells should not have the same confidence. Ignoring these differences, all of the active memory cells get the same correction error, and those errors from the unlearned state will corrode the previously learned information, which will, of course, fade over multiple training cycles, which is the basis for the success of many conventional CMAC algorithms. But for the learning of online dynamic systems, the real-time requirement is very high, in some cases only allow one or two cycles to complete the learning task, there is not enough time to eliminate this corrosion. Therefore, the learning results are often unable to meet the requirements of online learning.

3.2 Credit Assessment-CMAC(CA-CMAC)

In order to avoid the "corrosion" effect, the correction error must be distributed according to the reliability of the storage unit. However, in CMAC learning process, there is not a good way to determine a storage unit for the current error of more responsibility. In other words, there is no good way to determine the storage unit weight. The only information available is the number of times the storage unit weights are updated, and document [] assumes that the more times the storage unit learns to update, the more reliable the stored values are. So the number of times the memory unit learns is considered to be its credibility. The higher the credibility, the smaller the weight correction. This Eq.(3) is rewritten as:

$$w_j(t)=w_j(t-1)+\alpha a_j\left(\frac{\left(f(j)+1\right)^{-1}}{\sum_{l=1}^{m}\left(f(l)+1\right)^{-1}}\right)\left(y_d-\sum_{j=1}^{N_L}w_j(t-1)a_j(x)\right) \quad (4)$$

where $f(j)$ is the learning times of j memory cells and m is the number of memory cells activated by a state. Here the idea of weight updating is that the correction error must be inversely proportional to the learning times of the active cells. Here $\left(\left(f(j)+1\right)^{-1}\Big/\sum_{l=1}^{m}\left(f(l)+1\right)^{-1}\right)$ is used instead of Eq.(3) $1/N_L$, which effectively improves the learning performance.

3.3 Improved Credit Assign CMAC (ICA-CMAC)

Based on the above analysis, a concept of "balanced learning" is proposed to design an improved CMAC model ICA-CMAC based on reliability allocation, in which case the (4) is rewritten as:

$$w_j(t)=w_j(t-1)+\alpha a_j\left(\frac{\left(f(j)+1\right)^{-k}}{\sum_{l=1}^{m}\left(f(l)+1\right)^{-k}}\right)\left(y_d-\sum_{j=1}^{N_L}w_j(t-1)a_j(x)\right) \quad (6)$$

where k is a equilibrium learning constant, and it is clear that when k is 0 or 1, it is conventional CMAC and CA-CMAC, respectively[11].

In other words, CMAC and CA-CMAC are a special case of ICA-CMAC. The greater the learning frequency $f(j)$ of the active memory cell, the more knowledge it stores (information previously learned). The greater the equilibrium learning constant k, the less the weight change is for the memory cell with a higher learning number $f(j)$. When k is very large, the weight of the storage unit with a larger $f(j)$ is basically unchanged. At this time, no learning or the number of learning $f(j)$ less active units in the weight correction, will obtain most error correction values. In this case, "memory" or "retention of learned knowledge" predominates in online learning, whereas when k is small, learning times $f(j)$ have less effect on reliability allocation. When $k=0$, the number of learning $f(j)$ affects the distribution of reliability by zero. At this point, the error is evenly distributed to all active storage units. All active storage units have the same reliability allocation, regardless of the size of the learning times $f(j)$. The k is a balanced learning constant, which reflects the extent to which information previously learned and information not learned or

little learned influence the weight adjustment of storage units in the process of network training. Different k will have different learning results. The simulation results show that when k is a certain value, its learning speed is the fastest. This shows that the network "memory" and "forget" to achieve the best balance.

4.Hashing code address function design

In conventional CMAC, Hashing encoding technique is usually used to compress the storage space, but Hashing mapping will cause collision and decrease the approximation performance of CMAC. In [11], the address function is used to produce the symbol of the desired storage unit, which is encoded by a certain rule for all possible memory cells, and is a concise address method, and there is no data collision problem.

Take the three input CMAC system as an example, set m is the series of CMAC, nb is the number of blocks per level. The number of blocks per dimension is $m(nb-1)+1$. In this case, each block contains m states, and only $N=m(nb)^3$ storage cells are mapped to the $(m(nb-1)+1)^3$ state. Considering the state $\boldsymbol{s}$ expressed by (x_1,x_2,x_3), the number of storage units activated by it is m, and the address function of each active storage unit is $s(j)$, $i=1,2,\cdots,m$, then $s(j)=F(x_1,x_2,x_3,j)$, defined:

(i) If $j=1$, then $i=0$, other $i=m-j+1$ (7)

(ii) $$a_p=\text{int}\left(\frac{(x_p+1)}{m}\right),\ p=1,2,3 \tag{8}$$

(iii) $$s(j)=F(x_1,x_2,x_3,j)=\left(\left(\sum_{p=1}^{3}a_p\right)+(j-1)(nb)^2\right)(nb)+1 \tag{9}$$

5.CMAC Simulation Control Problem Description

5.1 Problem 1

In Fig. 3 is a schematic diagram of a three-joint manipulator moving on a plane. If the manipulator's arm L_1, L_2 and L_3 as well as the joint angle θ_1, θ_2 and θ_3 are known, the position of manipulator terminal in rectangular coordinates can be obtained, and the motion equation of manipulator is described as follows:

$$x_1=L_1\cos\theta_1+L_2\cos(\theta_1+\theta_2)+L_3\cos(\theta_1+\theta_2+\theta_3) \tag{10}$$

$$x_2=L_1\sin\theta_1+L_2\sin(\theta_1+\theta_2)+L_3\sin(\theta_1+\theta_2+\theta_3) \tag{11}$$

Setting the manipulator parameter to $L_1=L_2=L_3=0.5\ m$ requires the tracking space trajectory to be a circle on the plane

$$x_{d1}(t)=0.5-0.25\cos\omega t \tag{12}$$

$$x_{d2}(t)=0.25+0.25\sin\omega t \tag{13}$$

It is known from Eq.(12) and (13) that the center of the robotic arm centered at $(0.5,0.25)m$, has a radius of 0.25 m, and $\omega=0.5\pi\ rad/\sec$ .The sampling period is $T_s=0.02$ sec, and it takes 4 seconds to circle a circle and 200 times a circle.

A schematic diagram of a three-joint manipulator moving on a plane, show in Fig. 3. The three-joint manipulator control of CMAC show as Fig. 4, and Fig. 5 show the X-Y plane trajectory of

three-joint manipulator. The Joint angle change of three-joint manipulator trajectory and Mean square root error $E^T E$ of three-joint manipulator trajectory show as in Fig.6 and Fig.7.

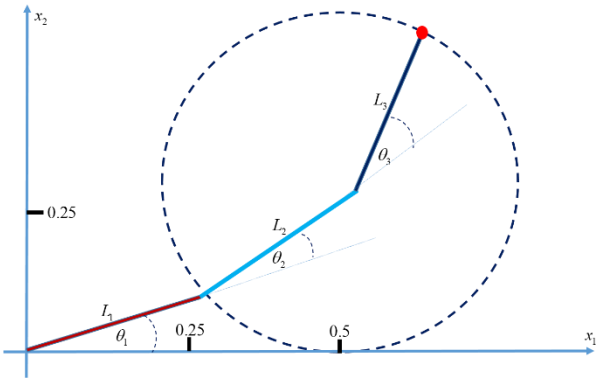

Fig. 3 A schematic diagram of a three-joint manipulator moving on a plane

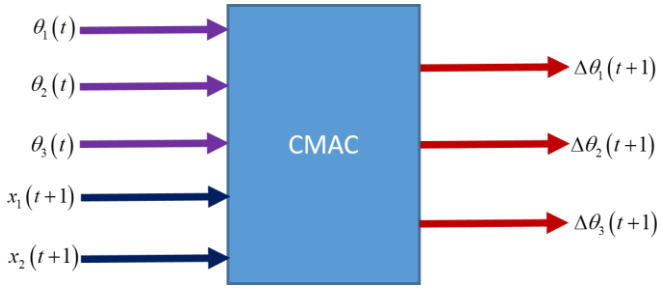

Fig. 4 three-joint manipulator control of CMAC

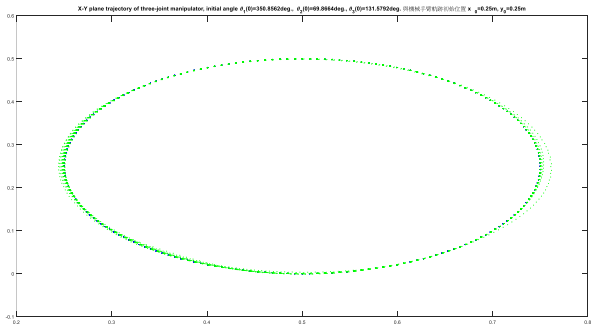

Fig. 5 X-Y plane trajectory of three-joint manipulator

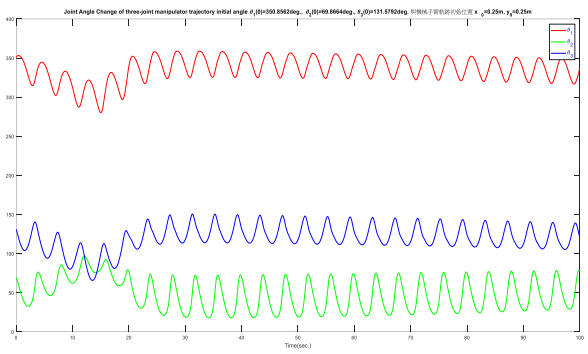

Fig. 6 Joint angle change of three-joint manipulator trajectory

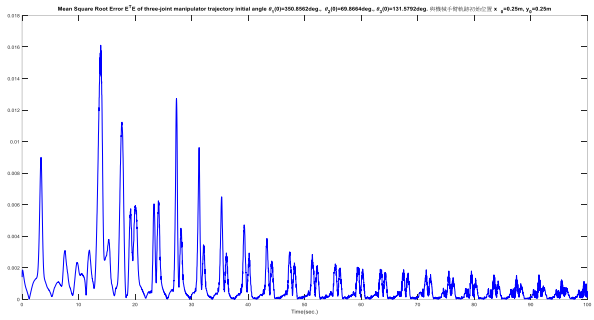

Fig. 7 Mean square root error $E^T E$ of three-joint manipulator trajectory

5.2 Problem 2

Two-dimensional input x_1 and x_2, one dimensional output y CMAC network, approximation $y = \dfrac{\sin(x_1)\sin(x_2)}{x_1 x_2}$,

$x_1,\ x_2 \in [-2\pi, 2\pi]$.

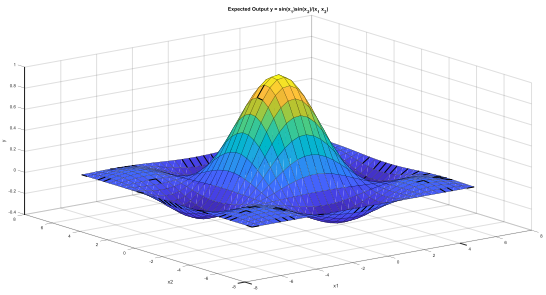

Fig. 8 Expected output $y = \dfrac{\sin(x_1)\sin(x_2)}{x_1 x_2}$

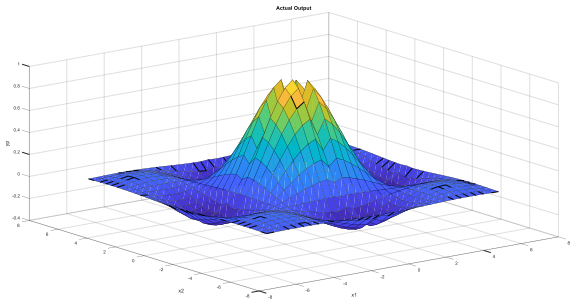

Fig. 9 Actual output simulation by CMAC

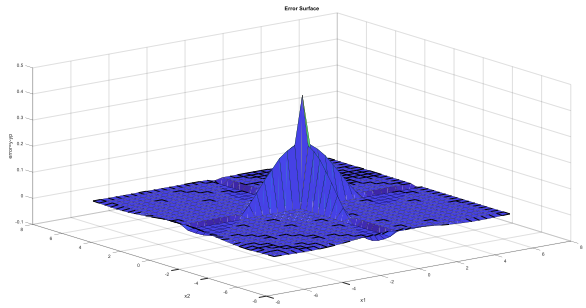

Fig. 10 Error surface

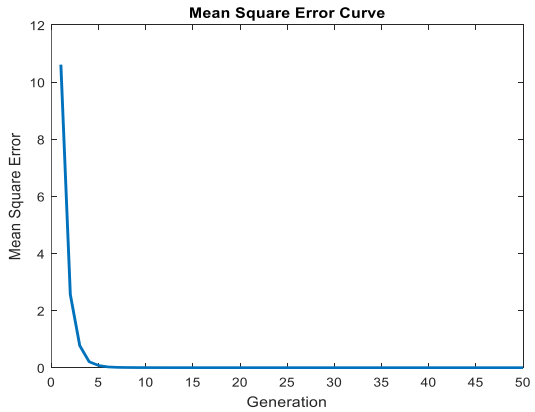

Fig. 11 Mean Square Error Curve by CMAC iteration

6.Conclusion

The basic CMAC network controller is simple and easy to implement. In practical use, using hardware storage to store the training learning weights directly affects the cost of the controller. By using the compact address space algorithm introduced in this paper, the storage scale is reduced by at least one unit compared to the controller that directly uses hash mapping on the virtual address space. The weight storage space is further reduced while maintaining the same control performance, resulting in increased speed and accuracy. One of the drawbacks of CMAC is the increase in high-dimensional input memory units, and there is a lack of theoretical basis for solving this problem. Furthermore, further study is needed to prove the convergence and stability of CMAC.

The base CMAC Network controller is simple and easy to implement. In the actual use of hardware storage to store the training learning weights, storage size directly affect the cost of the controller. Using the compact address space algorithm introduced in this paper, the storage scale is reduced to at least one unit, compared with the controller using hash mapping directly on the virtual address space, the weight storage space is decreased further with the same control performance, and the speed and accuracy are increased. One of the disadvantages of CMAC is the increase of high dimensional input memory unit, the method of solving this problem is lack of theoretical basis. In addition, the proof of the convergence and stability of CMAC needs further study.

7.Reference

[1] Albus, J.S. “Theory of Cerebellar Function,” *Mathematical Biosciences*, vol.10, No. 1, pp. 25–61, 2 Feb. 1971.

[2] Marr DA, “A theory of cerebellar cortex,” *The Journal of Physiology*, vol. 202, no. 2, pp.437–470, June 1969.

[3]Albus J.S., “A new approach to manipulator control: the cerebellar model articulation controller,” *ASME J-Dynamic Systems,*

Measurement, and Control. vol.97,no.3, pp.220-227, 1975.

[4]Albus J.S., "Data storage in cerebellar model articulation controller(CMAC)," *ASME J-Dynamic Systems, Measurement, and Control*, vol.97, no.3, pp.228-233, 1975.

[5] Albus J.S., "Mechanisms of Planning and Problem Solving in the Brain," *In: Mathematical Biosciences*, vol. 45, pp. 247-293, 1979.

[6] Su S.F., Tao T., Hung T. H., " Credit assigned CMAC and its application to online learning robust controllers," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*,vol.33, no.2,pp.202-213, 2003.

[7] Zhu D.Q., Zhang W., "Nonlinear identification algorithm of the improved CMAC based on balanced learning," *Control and Decision*, vol.19, no.12, Dec 2004.

[8] Miller W. T. , Glanz F. H., Kraft L. G., "CMAC: An associative of a bipedal walking alternative to backpropagation," *IEEE Journals & Magazines,* vol. 78,no.10, pp. 1561-1567, 1990.

[9] Pittner S., Kamarthi S. V. , "Feature extraction from wavelet coefficients for pattern recognition tasks," *IEEE Transaction on pattern analysis and machine intelligence* , vol.21, no.1, pp.83-88, 1999.

[10] Lee J.L., Tsai S., "On 2D Cerebellar Model-Based Heart-Disease Classification System," *MC-Transaction on Biotechnology*, vol.4, no.1, pp.30-44, 2012.

[11]Karayiannis N. B. Purushothaman G., "Fuzzy pattern classification using feedforward neural networks with multilevel hidden neurons," *IEEE International Conference on Neural Networks*, vol.3, no.27, pp. 1577-1582, 28 June-2 July 1994.

[12] Wong Y., Sideris A., "Learning convergence in the cerebellar model articulation controller," *IEEE Transactions on Neural Networks*, vol.3, no.1, pp.115-121, 1992.

Albus, J. S., " A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," Journal of Dynamic Systems, Measurement, and Control, 97(3), 220-227. 1975.

Albus, J. S., " A theory of cerebellar function," Mathematical Biosciences, 33(1-2), 247-265, 1976.

Albus, J. S., " Cerebellum-based neural networks for robotics," In Computational Intelligence: Imitating Life (pp. 102-109). IEEE Press., 1991.

Albus, J. S., & Norgaard, M., "A unified cerebellar model for control of robot manipulators and feedback error learning," Neural Networks, 9(8), 1237-1254, 1996.

Huynh Tuan-Tu,Lin Chih-Min,Le Tien-Loc,Cho Hsing-Yueh,Pham Thanh-Thao T.,Le Nguyen-Quoc-Khanh,Chao Fei, "A New Self-Organizing Fuzzy Cerebellar Model Articulation Controller for Uncertain Nonlinear Systems Using Overlapped Gaussian Membership Functions," *IEEE Transactions on Industrial Electronics*, Vol.67, Issue: 11, pp. 9671 - 9682, 2020.

Kumar G., Saha R., Conti M., Thomas R., Devgun T., "Adaptive Intrusion Detection in Edge Computing Using Cerebellar Model Articulation Controller and Spline Fit," *IEEE Transactions on Services Computing*,Vol.16, Issue: 2, pp. 900 – 912, 2023.