Application of Cerebellar Model Articulation Controller in Nonlinear System Tracking Control

Chien-Wen Lan[1] Jyh-Haw Wang[1] Ho-Nien Shou[2]

[1] The Graduate Institute of Electronic Engineering, Cheng Shiu University, Taiwan
[2] Air Force Institute of Technology, Taiwan

ABSTRACT

This paper presents an online learning control algorithm based on CMAC network to realize the tracking control problem of nonlinear system. Compared with the traditional two-dimensional input extended to multi-input multi-output nonlinear system, a new address space mapping method is introduced in the cerebellum controller to avoid collision of address space and improve the stability of the microcomputer image controller.

***Keywords*:** Cerebellar Model Articulation Controller, multi-input multi-output nonlinear system, address space collision, robustness

1. Introduction

In 1969, Marr first presented the biological model theory of cerebellar cortex [1], which was composed of the molecular layer, the purkinje cell layer and the granular layer, three-tier architecture, can store neural messages from the brain and various sensory organs of the body in layers. In 1975, based on the Marr model of the cerebellum, Albus developed a set of cerebellar mathematical models resembling cerebellar cortex operation[2], which can store the input information and output the message, and have a class neural network with learning function to update the weight of memory cells through repetitive learning process, To achieve the desired output, the cerebellum model is essentially an artificial neural network architecture that belongs to the memory "Lookup Table", and this feature will give the cerebellum model the following advantages: (i) Learning to converge faster: In the structure of the cerebellar model, the memory cell that is programmed to store the information is related to the learning, so that the input and output relationship can be adjusted by using the error feedback from the actual output of the cerebellum model and the expected output. Because

in each training process only needs to change the local memory weight, so the convergence speed is more rapid than the general kind of neural network. (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 that can be applied to nonlinear systems: compared to other neural networks, the cerebellar model is simple and easy to operate, and can quickly learn many nonlinear functions, and must converge to the least square root error [3]. (iv) It is easy to implement the cerebellar model algorithm on hardware. Due to these advantages, researchers in various fields have received much attention in recent years, such as applications in motor control [5], two-legged robot control [6], Industrial PID control [7] and adaptive control [8] and other fields.

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. CMAC Neural network Learning Algorithm Research

A CMAC neural network is a learning structure that mimics the human cerebellum. Essentially, it is a table lookup technique that uses a linear structure to represent complex nonlinear functions. Its network topology is shown in Fig. 1. The general CMAC Neural Network is divided into four-layer mappings.

(1) $\boldsymbol{S}\rightarrow\boldsymbol{M}$ Mapping: the mapping is also called the quantization process, and the variable s_i in the continuous input space $\boldsymbol{S}$ is quantified and mapped to the variable $[s_i]$ in the discrete input space $\boldsymbol{M}$. where s is the s-dimensional vector, each dimension is quantized, and the p-dimensional discrete variable $[s_i]$ is obtained. The smaller the quantization spacing, the larger the quantization space and the higher the precision of the sample distinction. (2) $\boldsymbol{M}\rightarrow\boldsymbol{A_c}$ mapping: This mapping is also called concept mapping. In the structure of the general CMAC Neural Network, the step function is used as the activation base function. The quantized discrete variable $[s_i]$ activates the C-unit of the virtual storage space $\boldsymbol{A_c}$, that is, $\boldsymbol{M}([s_i])=1$, $i = 1,2,\ldots,C$, the activated cell stores the value 1, and the inactive cell stores the value 0. The principle of mapping is: in the input space adjacent to the two sample vectors, quantization is also adjacent, in c a part of the overlapping units, the closer the distance, the more overlap. Conversely, the sample vectors away from the input space do not overlap in $\boldsymbol{A_c}$, this is the local generalization ability of CMAC neural network, in which C is called generalization factor, which determines the size of network output area, so it has direct influence on the generalization performance of CMAC neural network. The distance of the input sample is judged by the Hamming distance. (3) $\boldsymbol{A_c}\rightarrow\boldsymbol{A_p}$ mapping: This mapping is also called physical mapping. If the quantization space is very large, the virtual storage space is also very large. Assuming that each component in the p-dimensional input space is of magnitude N, then virtual memory a requires at least N^p units. In the problem of general control domain, p takes 10 and n takes 200, so that the size of $\boldsymbol{A_c}$ space is $200^{10}>10^{23}$, so the mass storage is often not achievable. For most learning problems, you do not include all the values of the input space, so consider using virtual memory $\boldsymbol{A_c}$ to map to a much smaller physical, achievable memory $\boldsymbol{A_p}$, via

Hash coding. (4) $A_p \rightarrow Y$ output: The physical storage space A_p is activated by the weight of C units by linear addition, the output Y

$$Y = \sum_{i=1}^{C} M([s_i])w_i \quad (1)$$

The weights updating algorithm of the general CMAC Neural network adopts the LMS algorithm, and the weight modification formula is

$$\Delta w_i = w_i(t+1) - w_i(t) = \frac{\eta(R-Y)}{C} \quad (2)$$

where η is the learning rate, R is the target output value, Y is the network output, and C is the generalization factor.

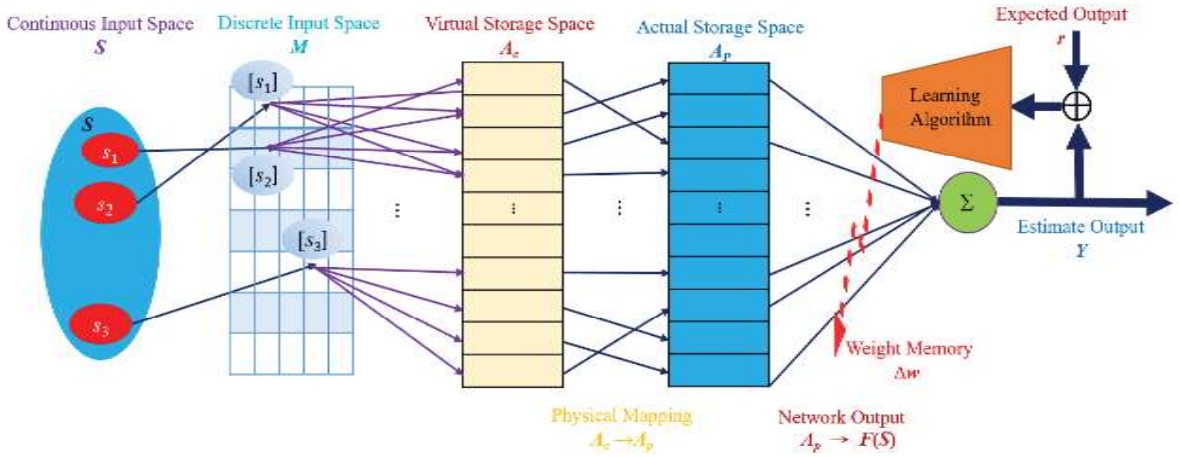

Fig. 1 The structure of CMAC neural network

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

2.2 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 [9] on the basis of analyzing the normal CMAC weight adjustment rules and considering the credibility of the learned knowledge. Literature [10] 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.

2.3 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 [9] 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:

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

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

$$\left(\left(f(j)+1\right)^{-1} \Big/ \sum_{l=1}^{m}\left(f(l)+1\right)^{-1} \right) \quad (5)$$

is used instead of Eq.(3) $1/N_L$, which effectively improves the learning performance.

2.4 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:

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

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[9].

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.

3. 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 [9], 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 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) $a_p=\text{int}\left(\frac{(x_p+1)}{m}\right),\ p=1,2,3$ (8)

(iii) $s(j)=F(x_1,x_2,x_3,j)$

$$=\left(\left(\sum_{p=1}^{3}a_p\right)+(j-1)(nb)^2\right)(nb)+1 \quad (9)$$

4. CMAC Simulation Control Problem Description

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:

$$x_1=L_1\cos\theta_1+L_2\cos(\theta_1+\theta_2)+L_3\cos(\theta_1+\theta_2+\theta_3) \quad (10)$$

$$x_2=L_1\sin\theta_1+L_2\sin(\theta_1+\theta_2)+L_3\sin(\theta_1+\theta_2+\theta_3) \quad (11)$$

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

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

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

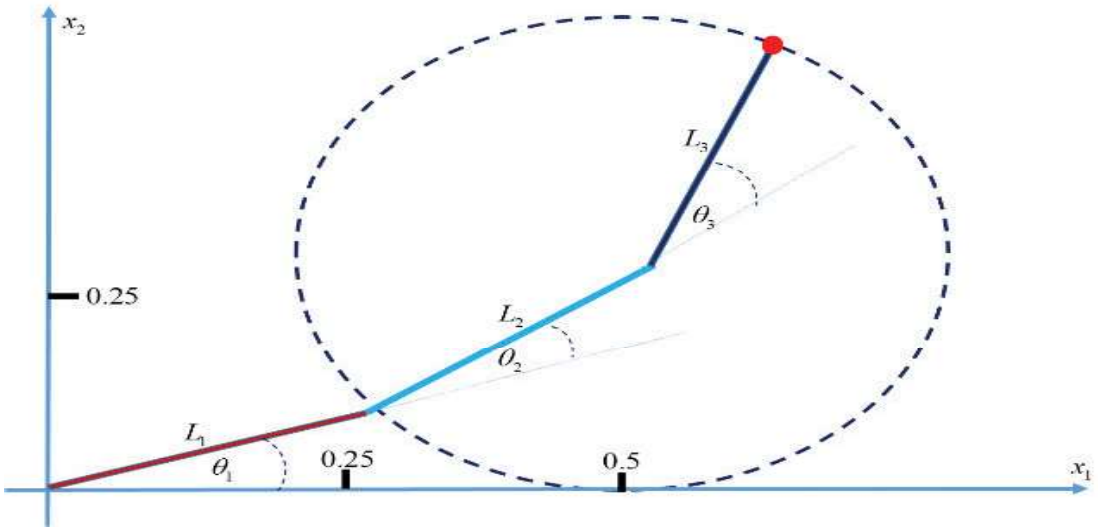

Fig. 3 schematic diagram of a three-joint robotic arm moving on a plane

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 initial angle of the mechanical arm is $[\theta_{10}, \theta_{20}, \theta_{30}]^T = [352, 69.9, 131.6]^T\ deg$.

The sampling period is $T_s = 0.02\sec$, and it takes 4 seconds to circle a circle and 200 times a circle. Simulation using (i) Compact space Address image algorithm controller (ii) Compact space address image plus hashing image algorithm controller.

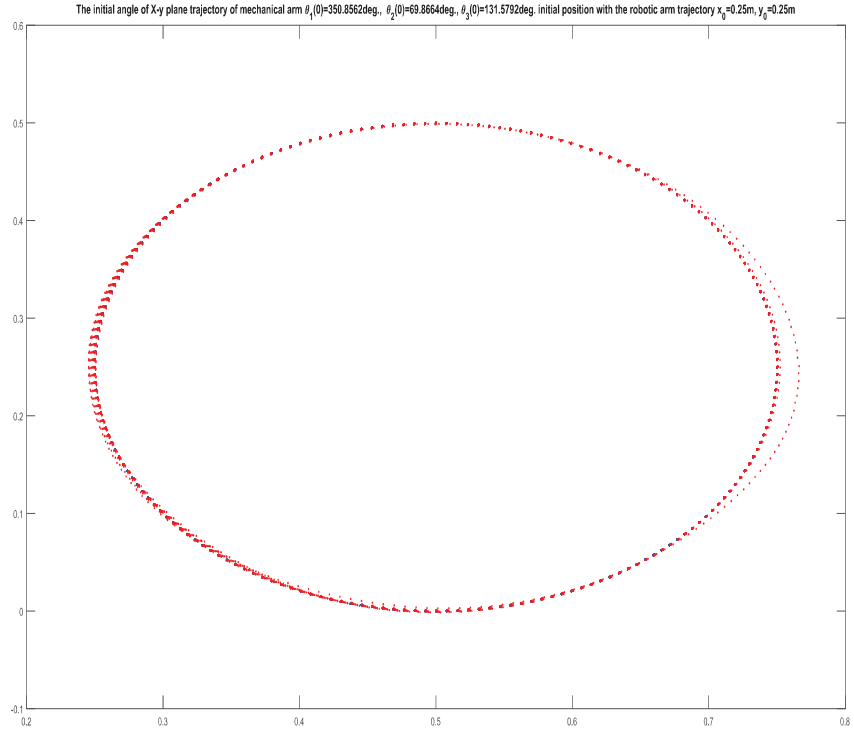

Fig. 4(a)(i) Time response of x-y plane trajectory of robotic arm

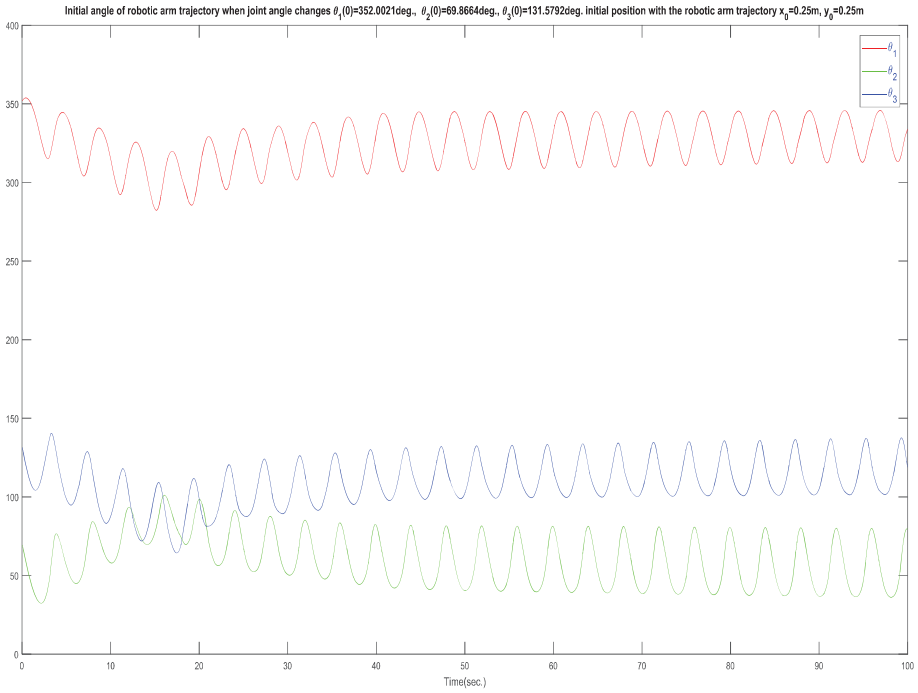

Fig. 4(b)(i) Time response of robotic arm joint Angle

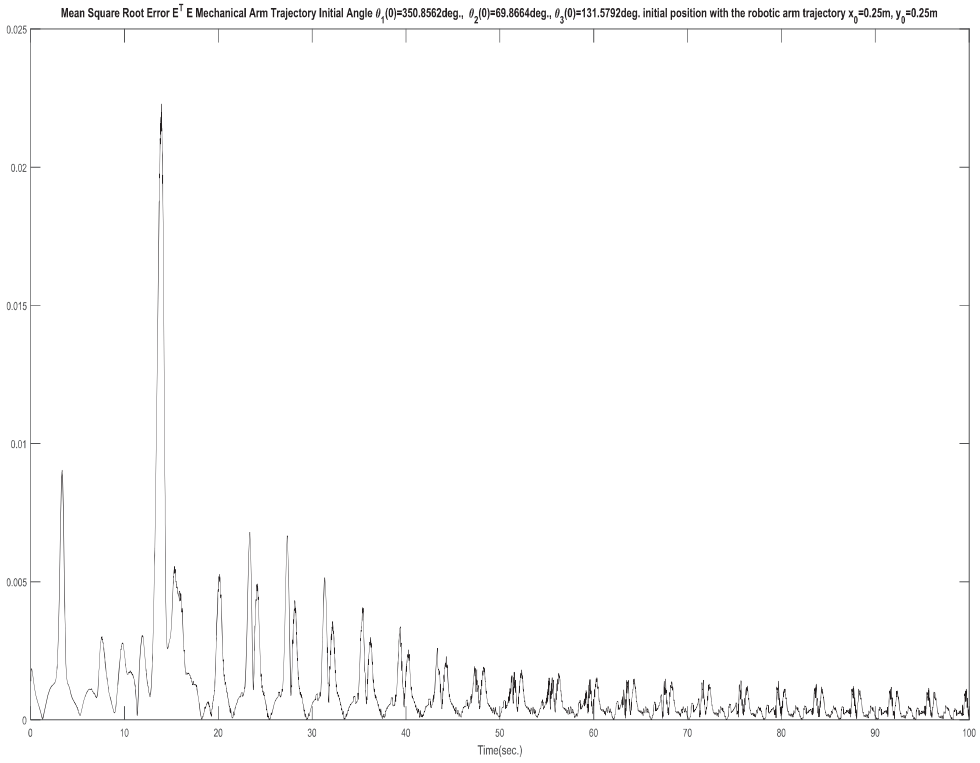

Fig. 4(c)(i) Time response of E^TE in robotic arm trajectory

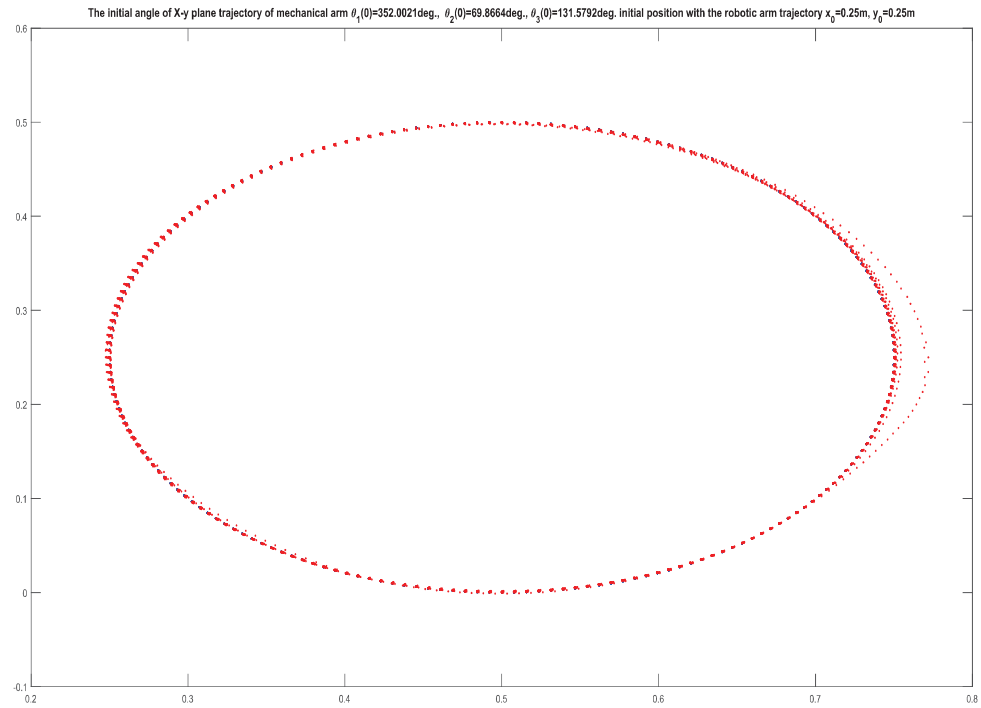

Fig. 5(a)(ii) Time response of x-y plane trajectory of robotic arm

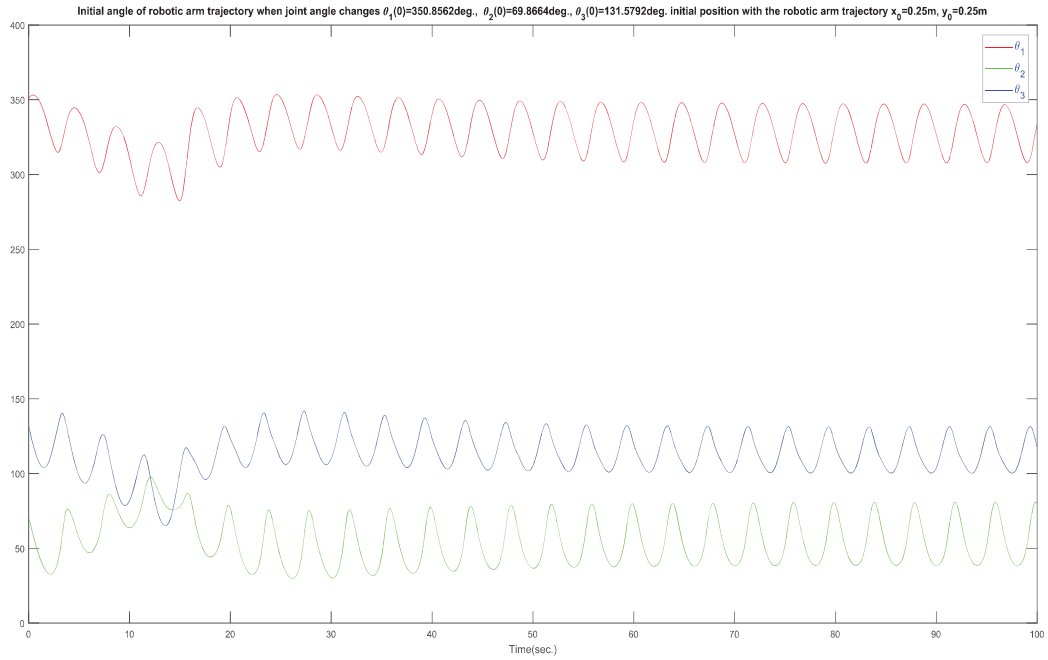

Fig. 5(b)(ii) Time response of robotic arm joint angle

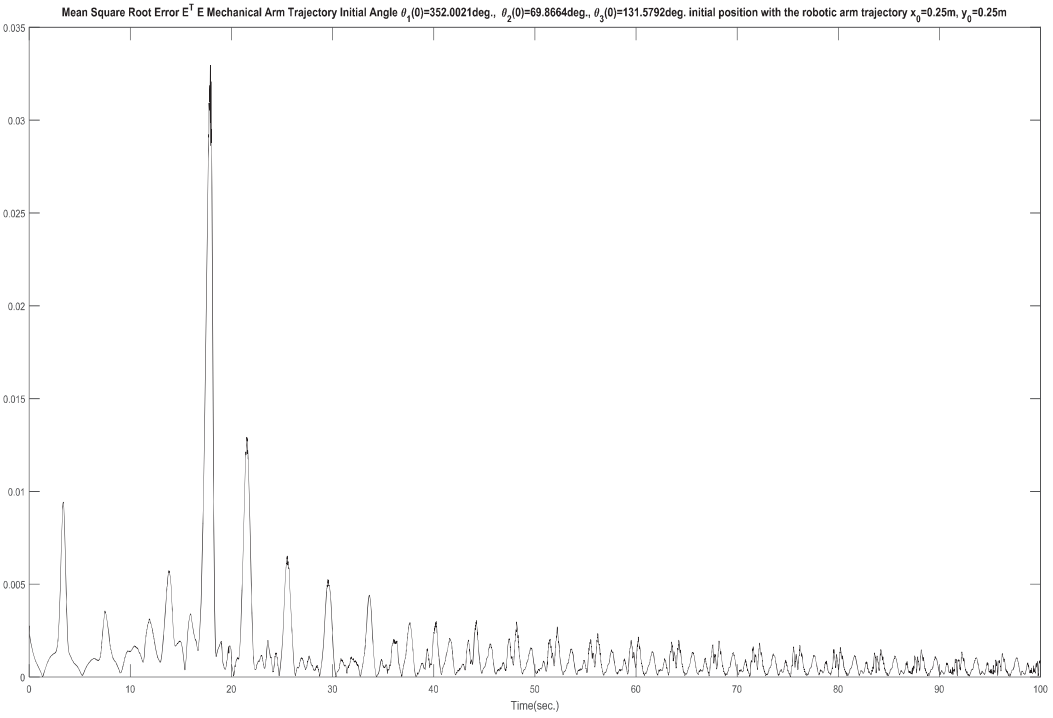

Fig. 5(c)(ii) Time response of E^TE in robotic arm trajectory

5. Conclusion

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.

References

[1] Marr D., “A theory of cerebellar cortex, ” The Journal of physiology, Wiley Online Library, 1969.

[2] Albus, J.S., “ Data Storage in the Cerebellar Model Articulation Controller (CMAC),” *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 97, pp.228–233 , 1975.

[3] Albus JS: A new approach to manipulator control: the cerebellar model articulation controller. ASME J-Dynamic Systems, Measurement, and Control, vol. 97, pp. 220-227, 1975.

[4] Cotter N.E., Kim H., “A pulse Neural Network Capable of Universal Approximation,” *IEEE Transactions on Neural Networks*, vol.3, pp.642-647, 1992.

[5] Li Z., “CMAC Neural Networks Based Combining Control for BLDC Motor,” *IEEE International Workshop on Intelligent Systems and Applications*, pp.1-4, Wuhan, China. May 23-24, 2009.

[6] He B., Lu Q. and Wang Z., “Coupling effect analysis between the central nervous system and the CPG network with proprioception,” *Robotica*,vol.33, no.6, pp. 1281-1294, July 2015.

[7] Jia C. Y., Shan X. Y., Cui Y. C., “Modeling and Simulation of Hydraulic Roll Bending System Based on CMAC Neural Network and PID Coupling Control Strategy,” International Journal of Iron and Steel Research, vol.20, no.10, pp.17-22, 2013.

[8] Meng D. Y., Tao G. L., Li A. M., “Adaptive robust control of pneumatic cylinders using fast switching on/off solenoid valves,” Journal of Mechanical Engineering, , vol.50, no.10, pp.180-188, 2015.

[9]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.

[10]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.

小腦神經控制器在非線性系統跟蹤控制中的應用

藍健文[1]王志浩[1]壽鶴年[2]

[1]正修科技大學 電子工程系

[2]空軍航空技術學院 航空通訊電子系

摘要

提出一種基於小腦神經網路的線上學習控制演算法，實現非線性系統的跟蹤控制問題。本文將小腦神經控制器用於多輸入多輸出非線性系統，相較於傳統將二維輸入擴展到多輸入多輸出系統，並在小腦神經控制器神經網路中引入一種新的位址空間映射方法，避免了位址空間碰撞問題，從而提高了位址映射的精度。電腦模擬結果表明，小腦神經控制器具有較快的學習收斂速度和良好的強健性。

關鍵詞：小腦神經控制器，多輸入多輸出非線性系統，位址空間碰撞，強健性