CMAC NEURAL NETWORK: MODELING, SIMULATION, AND A COMPARATIVE STUDY OF LEARNING ALGORITHMS

Magdy Abdelhameed
Ain Shams University
Faculty of Engineering
Abdo Pasha, Abbasia,
Cairo, Egypt
E-mail: magdyma@gmx.net

Ahmed Kassem
Amro Shafik
Benha University
High Institute of Technology
Benha, Qaluobiya, Egypt
E-mail: a_kassem_2010@hotmail.com
E-mail: amrbhit@gmail.com

KEYWORDS
Neural networks, CMAC, Learning algorithms.

ABSTRACT
Cerebellar Model Articulation Controller Neural Networks (CMAC NN) is one of the intelligent systems used for modeling, identification, classification, and controlling of nonlinear systems. In this paper, the mathematical model of CMAC is presented. CMAC is implemented using Simulink environment and its parameters are tuned to get the best CMAC control action. Three different learning algorithms are tested, using a constant learning rate, a variable learning rate, and learning by the control action of the conjugate conventional controller. The effect of varying CMAC parameters is studied and discussed. The simulation results showed that the learning algorithm based on constant learning rate gives the best performance.

INTRODUCTION
The Cerebellar Model Articulation Controller (CMAC) is a type of neural networks based on a model of the mammalian cerebellum. It is also known as the Cerebellar Model Arithmetic Computer. It is a type of associative memory. (Albus 1975) introduced the theory of cerebellar function. He stated that CMAC is a memory management technique which causes similar inputs to tend to generalize so as to produce similar outputs, and dissimilar inputs result in outputs which are independent. (Albus 1975) clarified that CMAC computes control functions by referring to a table rather than by solution of analytic equations or by conventional analog servo techniques.

CMAC has two basic features if it is compared to neural networks (NN), it has a very fast learning capability, and it requires minimal a prior knowledge of the system. CMAC also has good generalization capability and information storing ability. CMAC is more suitable for real time implementation, since it does not contain time consuming sigmoid activation functions. But besides these attractive features it has a serious drawback; its memory complexity may be very large. In multidimensional case this may be so large that practically it cannot be implemented.

(Horvath and Gati 2009) presented a solution for the memory complexity problem of CMAC through using Kernel CMAC (KCMAC). KCMAC reduce the memory complexity of CMAC networks without deteriorating its performance. The proposed version exploits the benefits of kernel representation and the complexity reduction effect of hash-coding, while smoothing regularization helps to reduce the performance degradation.

Several works were done for engineering and industrial applications such as (Abdelhameed et al. 2002) who introduced an adaptive learning algorithm for CMAC, in order to solve the instability problem which is occurred with the conventional CMAC after a long period of real time runs, to improve the tracking control of a piezoelectric actuated tool post. (Tsai and Yeh 2009) developed a CMAC NN speed estimator for speed-sensorless induction motor drives. They used the gradient-type learning technique to provide a real-time adaptive estimation of the motor speed. (Yu et al. 2009) built a hybrid controller includes a proportional-derivative compensator (PD) and a fuzzy CMAC for position tracking and anti-swing of an overhead crane. They discussed the stability of the system using a Lyapunov method and an input-to-state stability technique and proved that the controller is robustly stable with bounded uncertainties. (Ding et al. 2007) introduced a dynamic compensation method based on CMAC neural network for an accelerometer to avoid the noises, the drift error and the disturbances of the system. The data of the accelerometer dynamic response was measured and used for creating the dynamic compensation model which its parameters were trained by CMAC NN. (Li and Xu 2009) designed another hybrid controller in which CMAC feedforward and PID feedback control was employed for XY parallel micropositioning stage. CMAC neural network
with adjustable learning rate was employed into the PID control to compensate the hysteresis arising from piezoelectric actuator used for micropositioning.

Several works were done for medical applications of CMAC. (Bucak and Baki 2010) used CMAC ANN to develop an expert diagnosis system has a three-stage CMAC ANN architecture for classifying the healthy liver (liver data with no liver disease), hepatitis, and cirrhosis through the interpretation of the data consisted of the liver enzymes. The classification had a 100% success rate. (Wen et al. 2009) proposed a self-organizing cerebellar model articulation controller (SOCMAC) for the real-time classification of ECG complexes. The well-trained classifier used to classify the testing set and gives a classification accuracy of 98.21% which is comparable to the existing results. (Teddy 2008) presented a kernel density-based CMAC (KCMAC) model for clinical decision support. This model was differ than the classical CMAC by employing a multi-resolution organization scheme of its computing cells and adopting the Takagi–Sugeno–Kang (TSK) fuzzy model to define its network computation. KCMAC was applied to two case studies, breast cancer diagnosis and the modeling of the human glucose metabolic cycle and the experimental results were encouraging.

(Ortiz et al. 2007) presented another form of CMAC structure called Recurrent Fuzzy CMAC (RF-CMAC) for nonlinear system modeling and identification. A new simple training algorithm with time-varying learning rates was introduced to assure the algorithm stability. But it needs further works on structure training and adaptive control, and also the real-time implementation needs to be tested.

Modeling and identification of nonlinear systems are two important engineering disciplines which are covered by CMAC also. (Lin and Lee 2009) developed a Parametric Fuzzy CMAC (P-FCMAC) by using a self-constructing learning algorithm which was consisted of the self-clustering method (SCM) and the back-propagation algorithm which was used to tune the adjustable parameters in place of Modified Genetic Algorithm (MGA). The performance and applicability of the proposed model were proven through three examples: prediction of the chaotic time series, approximation of a Sugeno’s nonlinear function, and identification of a nonlinear system. (Yu 2008) proposed a Hierarchical Fuzzy CMAC (HF-CMAC) to model nonlinear systems with input-output and state-space forms. Two stable and effective training algorithms were developed. The new algorithms with time-varying learning rates were proven to be stable with input-to-state stability.

From the survey on the literatures relating to the work with CMAC NN, it is found that CMAC is a competitive intelligent controller used in modeling, identification, classification, compensation and for nonlinear control purposes for different computational, industrial, and medical applications.

PROBLEM STATEMENT

The objective of this work is to introduce a simple base on how to build CMAC, how each parameter variation affects the CMAC performance, and how the use of different learning algorithms can affect the CMAC performance.

MATHEMATICAL MODEL OF CMAC

The mathematical model of CMAC can be summarized in those following equations. The CMAC control signal $U_{CMAC}$ at any instant $k$ is equal to the sum of weights of the activated cells like the following equation.

$$U_{CMAC}(k) = \sum_{i=1}^{n} w_i(k)x_i(k)$$  (1)

where $w_i(k)$ is the weight or the content of the memory cell, and $x_i(k)$ is considered to be a switch which has two values for any memory element i, 1 for activated cell and 0 for deactivated cell. Therefore, the allocating vector $s_i(k)$ can be defined as an AND function or an equal statement. It says if the switched summation of $aw_i(k)$, the $i^{th}$ added value at any instant k, and $ml(k)$, the first active memory element location at instant k, equal to the $i^{th}$ memory location, then output 1, else output 0. If $x_i(k)$ is equal to 0, then the memory content of that cell will not be exist in the CMAC control signal, i.e. the cell is deactivated. Thus, the value of $x_i(k)$ can be found using the following equations

$$x_i(k) = [s_i(k)g_i(k)] \text{ AND (i)}$$  (2)

$$g_i(k) = av_i(k) + ml(k)$$  (3)

The $i^{th}$ added value can be determined using the following formula, with knowing that the initial value of the allocating pointer is equal to 0, and $P$ is the memory size.

$$av_i(k) = \sum_{i=1}^{P} x_{(i-1)}(k)$$  (4)

$$x_a(k) = 0$$  (5)

The switch $s_i(k)$ defined by another AND function between two inequalities. The first inequality between $g_i(k)$ and $ml(k)$ which represents the generalized memory location, and the second is between the location value $i$ and the first active memory location $ml(k)$. Thus, the switch value is determined by the following relation.

$$s_i(k) = [g_i(k) \leq ml(g)] \text{ AND } [ml(g) \leq i]$$  (6)

$$ml(g) = ml(k) + n_g - 1$$  (7)

where $n_g$ is the generalization size. The representative diagram of CMAC structure is shown in Figure 1.

\[\text{Diagram of CMAC structure is shown in Figure 1.}\]
CMAC LEARNING ALGORITHMS

Learning With Constant Learning Rate

The contents of the memory cells, i.e. the weights, are changed continuously and updated in order to minimize the error value. The error in our experiments $e$ is equal to the difference in values of the reference or desired piston rod displacement $x_d$ and the actual one $x_a$ at any instant $k$. A gradient-type learning rule is used to update the weights. Theoretically the learning process will be stopped when the error is equal to zero which is experimentally too hard to be achieved even for long running times. Therefore, the error should be minimized by changing the values of the learning rate $\beta$ which is used to change the learning speed, and the scale which is used to size the CMAC control signal. Thus, the learning equation is found to be in the form of

$$w_i(k+1) = w_i(k) + \beta \times \text{scale} \times e(k)$$

(8)

$$e(k) = x_d(k) - x_a(k)$$

(9)

Learning With Adaptive Learning Rate

If the learning rate changes continuously with a defined slope and repeat itself, it may affect the CMAC performance according to (Abdelhameed et al. 2002). The best value of the initial learning rate $\beta_i$ is found to be the unity from previous works, and any values above the unity makes the running $e_{\text{RMS}}$ diverges over time. Thus, the variable $\beta$ structure will change beta from 0 to 1. The variable learning rate equation can be defined from

$$\beta(k) = \beta_i \left(1 - \text{Cm}(k) \sum_{j=0}^{k} |e(j)|\right)$$

(10)

where $C$ is the learning decaying rate parameter, and $m(k)$ is the learning sensitivity function which is a nonlinear switching learning function defined by

$$m(k) = \begin{cases} 0, & \text{if } (|e(k)|_{\text{mean}} \geq \varepsilon) \text{ AND } (\beta(k-1) = 0), \\ 1, & \text{otherwise} \end{cases}$$

(11)

where $\varepsilon$ is the permissible absolute mean error. The learning function is the same as Equation (8), but by using the learning rate value which is given by Equation (10).

Learning With PV Control Action

Another method of learning can be achieved by learning with the conjugate controller control signal. (Hsu et al. 2009) introduced an Adaptive CMAC Neural Control (ACNC) system with a PI-learning algorithm. Therefore, the third learning algorithm can be achieved by updating the weights by a scaled value of Proportional-Velocity (PV) control action $U_{PV}$. This method introduced to let $U_{CMAC}$ to be the dominant signal after learning or to minimize $U_{PV}$ action overtime to let the intelligent control action of $U_{CMAC}$ to be able to overcome the nonlinearities of the system. Thus, the learning algorithm can be written as

$$w_i(k+1) = w_i(k) + \beta \times \text{scale} \times U_{PV}(k)$$

(12)

The PV control action can be defined at any instant $k$ by

$$U_{PV}(k) = K_p(x_d(k) - x_a(k)) - K_v \left(\frac{dx_a(k)}{dt}\right)$$

(13)

The tuning process for $K_p$ and $K_v$ is done by using Parameter Estimation Toolbox which is integrated with MATLAB/SIMULINK environment. The final values are obtained after 32 iterations using pattern search algorithm for optimization, and the values are found to be $K_p=10$ and $K_v=-0.1$.

SYSTEM CONFIGURATION

The CMAC performance is measured while it is connected in parallel form with a conventional servo controller of a Proportional-Velocity (PV) type. The system which is the
subject under control is an electrohydraulic system which is equipped to give a linear simple harmonic motion through using a hydraulic cylinder. The overall structure of the system can be simplified by the block diagram representation as shown in Figure 2.

**PERFORMANCE INDEX**

The root mean square (abbreviated RMS or rms), is a statistical measure of the magnitude of a varying quantity. It is especially useful when the variations are positive and negative, e.g., sinusoidal functions. It can be calculated for a series of discrete values or for a continuously varying function. The RMS value for a continuous function \( x(t) \) defined over the interval \( T_1 \leq t \leq T_2 \) is given by

\[
x_{\text{RMS}} = \sqrt{\frac{1}{T_2 - T_1} \int_{T_1}^{T_2} [x(t)]^2 \, dt}
\]

The running or accumulative \( x_{\text{RMS}} \) defined above will be used for the error function which is deduced from the difference between the desired and actual motion trajectories in order to compare the results. All \( x_{\text{RMS}} \) values in this paper will be in centimeters of the cylinder rod displacement.

**STUDY OF CMAC PARAMETERS**

**Constant Learning Rate Algorithm**

There are four basic parameters will be discussed in this point, the generalization size \( n_g \), the memory size \( P \), the learning rate \( \beta \), and the scale for constant learning rate.

**Studying the Generalization Size Variation**

The generalization means that each input vector to CMAC is mapped into a number of memory locations instead of only one memory location. As the generalization size increases, the memory locations are overlapped which gives a bad performance of CMAC. Before the presence of overlapping, a chattering phenomenon appears in the mapping function output. Figures 3 and 4 present how the generalization size affects the CMAC performance.

![Figure 3: Effect of generalization size variation on CMAC performance](image)

**Studying the Memory Size Variation**

From the logic thinking that as the memory size increases the efficiency of the controller increases, it is required to observe the variation of the memory size with CMAC efficiency. The linear mapping function used is in the form of

\[
ml = \text{Gain} \times \text{input vector} + \text{offset value}
\]

where \( ml \) is the memory location. The offset value doesn’t affect the performance as all the memory locations are above the zero level, i.e. the memory location exists; consequently the gain of the linear function is the factor under consideration. Figures 5 and 6 present how the mapping function gain affects the CMAC performance.
From the results, CMAC performance is constant with a low efficiency for memory locations less than 200, we can called it the critical size. After that, the performance starts to increase until the memory size exceeds the designed size, i.e. 260 memory elements are built for the CMAC under study by using MATLAB/SIMULINK R2007b. For this reason a gain of 45 which gives 270 cells gives the worst condition because the CMAC is oversized.

**Studying the Learning Rate Variation**

The learning rate is the basic parameter in the learning algorithm. The learning rate reflects the speed of changing the weights. A good selection of the learning rate enhances the CMAC performance by adjusting the learning speed. Low learning rates make CMAC needs more time to give a good performance, and high values make CMAC behaves randomly. Figures 7 and 8 present how the learning rate affects the CMAC performance.

**Studying the Scale Variation (for constant learning)**

The scale is used sometimes to size the CMAC control signal to be the effective signal if it added to another conventional controller. The scale and the learning rate $\beta$ are multiplied with error to update the weights in the learning algorithm. The multiplication has a commutative property, therefore if we fixed the other parameters for their best values, especially the learning rate, the scale values will be the same of those in Figure 7.

**Variable Learning rate Algorithm**

The variable (can be called adjustable or adaptive) learning algorithm was firstly presented by (Abdelhameed et al. 2002) to solve the instability problem of the constant learning algorithm for long operating times, therefore it is preferable to work in that case. Herein, it is required to test it for short times. For studying the adaptive learning rate algorithm, there are four different parameters under study, the initial value of the learning rate $\beta_i$, the decaying rate parameter C, the scale, and the permissible absolute mean error $\epsilon$. The last parameter $\epsilon$ is used for making the limiting value that after it the learning process will be stopped. The initial learning rate $\beta_i$ is found to let its value equal to unity. The generalization size $n_g=31$ gives the best performance with a little change in performance among the nearest values.

**Studying the Variable Learning Slope Variation**

Decaying factor C controls the slope of the learning rate curve, i.e. the frequency of $\beta$ pattern, as shown in Figure 9. The smallest values of C give an approximation to the behavior of the constant $\beta$ learning algorithm, and the highest values increases the frequency of $\beta$ pattern which results in good learning capability. The worst condition of C is to choose a middle value. Figure 9 clarify how the change in C values changes the $\beta$ pattern. The change of $\epsilon_{RMS}$ with respect to C variation is shown in Figure 10, and for three different C values over time in Figure 11.
Studying the Scale Variation (for Variable Learning)

The effect of scale variation can be determined by keeping the other values constant and studying the $e_{\text{RMS}}$ performance index for various scales. Decaying parameter $C=50$ is taken for this test. The results are shown in Figures 12 and 13. As the scale parameter increases the error increases with a great amount and make the RMS error value diverges over time, it is done because the scale directly affects the learning function; refer to Equation (8).

Learning by PV Control Action

In this method of learning, there are two parameters only are found to be effective. The generalization size $n_g$ and the scale of the CMAC control action $U_{CMAC}$. The other parameters are kept constant with their best values as found by fine tuning in the learning by the constant learning rate. The generalization size of $n_g$=36 is found to be the best value with a little change of the performance among the nearest values. The scale of 0.04 gives the best performance. The change of $e_{\text{RMS}}$ values with scale is shown in Figures 14 and 15. For scale values above 0.05, the $e_{\text{RMS}}$ values are greatly diverges.

SUMMARY OF RESULTS

The results of the work done in this paper are summarized and tabulated as below. Table 1 includes the final and tuned parameters for the three learning methods.
The memory size, learning rate, and the mapping function gain are found to be constant for all methods. The generalization size and $U_{CMAC}$ scale must be tuned well for each algorithm to give the best performance through decreasing the $e_{RMS}$ values. Table 2 shows the final values of $e_{RMS}$ for the three discussed learning algorithms with their best tuned values. Figure 16 clarify how the use of CMAC enhances the performance of the tested system even with using the PV type servo controller with its optimal gains, and how the use of different learning algorithms affects the CMAC performance. The variable learning algorithm gives a 50% decrease in the CMAC performance rather than using the other algorithms. The learning by a constant learning rate whether using the error value or $U_{PV}$ value for learning gives a good enhancement of about 3.5 times than using PV controller alone which gives an $e_{RMS}$ value of 0.2162 cm.

**CONCLUSIONS**

The mathematical model of Cerebellar Model Articulation Controller (CMAC) is presented. The model parameters are discussed and studied. Three different learning algorithms are distinguished, learning by a constant learning rate, variable learning rate, and learning by a Proportional Velocity (PV) control signal $U_{PV}$ instead of learning by error value. It is found that:

1. Learning by a constant rate gives the best enhancement of the performance.
2. A unity learning rate $\beta$ is the ideal value, which is stated by the previous works.
3. As the memory size increases, the performance increases but after a critical size.
4. Learning by a variable constant rate reaches the steady state value faster than the other methods but gives the worst value for the shortest times.
5. Fine tuning of the generalization size ($n_g$) and CMAC control signal ($U_{CMAC}$) scale is too important to be done because they highly affect the CMAC performance.

Future research can be focused on developing different learning algorithms to shorten the learning time and increasing the performance of CMAC. Also, some work is required to compare between CMAC and different nonlinear controllers or with traditional ANNs. Finally, if an adaptive mean can be introduced in CMAC to change its parameters online, it will make a great challenge.

**REFERENCES**


BIBLIOGRAPHIE

MAGDY ABDELHAMEED studied the mechanical engineering in Ain Shams University, Faculty of Engineering. He was awarded the B.Sc. degree in 1983, M.Sc. degree in 1989, and Ph.D. degree in 1994 from the same institution. He was awarded the Associate Professor in 2000, and full Professor of Automatic Control an Mechatronics in 2005. His main fields of interests are Control Systems Design, Automation and Robotics, Mechatronics, Control of Discrete Event Manufacturing Systems, Intelligent Control, Servo Motion Control Systems, Electro Hydraulic Control, Precision Motion Control, Nano Positioning Devices, Programmable Logic Controllers (PLC), Pneumatic and Hydraulic Control, Fault Detection and Diagnosis.

AHMED KASSEM studied the mechanical engineering in the Shoubra Faculty of Engineering, Zagazig University, Benha Branch. He was awarded the B.Sc. degree in mechanical engineering, production major in 1977, and M.Sc. degree in mechanical engineering in 1985 from the same institution. He awarded the Ph.D. degree in mechanical engineering in 1994 from the West Berlin University, Germany in manufacturing technology. His main fields of interests in research are manufacturing technology, casting and forming processes, non-conventional machining.

AMRO SHAFIK was born 1983 in Benha, Qaluobiya, Egypt. He studied mechanical engineering in the department of mechanical engineering technology, High Institute of Technology, Benha University. He had the Diploma in Mechanical Engineering technology, Manufacturing Major in 2002, and the Bachelor of Science (B.Sc.) degree in the same institute in 2004. He was awarded the Master of Science (M.Sc.) degree in 2010 in the field of Automation and Mechatronics. His main research interest is to develop control systems by using intelligent controllers and expert systems to enhance the performance of mechanical systems.