Cerebellar Model Articulation Controller (CMAC) for Suppression of Structural Vibration

Dong-Hyawn Kim¹; Ju-Won Oh²; and In-Won Lee³

Abstract: Applicability of the cerebellar model articulation controller (CMAC) to structural control is explored, and a training algorithm for the suppression of structural vibration is proposed. The training epochs and controlled responses by both the CMAC and the well-known multilayer neural network are compared in the numerical example. The dynamics of the hydraulic actuator and the time delay of the controller are considered in the simulation. It is concluded that the CMAC is a promising candidate for a structural controller.


CE Database keywords: Neural networks; Vibration control; Simulation; Costs; Sensitivity analysis; Damping.

Introduction

Artificial neural networks (ANNs) have emerged as new, promising tools for many engineering applications. ANNs do not use the mathematical model of a system to obtain solutions, but use an input-output relationship that can be measured from the system. The final solution is obtained by learning the relationship. Once an ANN is trained, it can predict the desired output, or solution, to a given input based on the trained intelligence. The reason why ANNs are attractive to engineers is that an ANN can generalize some trained relationship to other, untrained ones and therefore can solve problems with limited training data.

Structural control has been one of the most frequent application areas of ANNs during the last decade. ANNs can be used to design a controller of a structure with nonlinearity and uncertainty, which has been a challenging task for conventional control approaches. The learning capability of ANNs makes it possible to solve such problems. With high nonlinearity and modeling uncertainty, it is not easy or even possible to design a controller by conventional approaches based on the mathematical modeling of a structure. However, design methodologies based on ANNs do not require a mathematical model, but instead use only the structural responses for the design of the ANN controller, simply called the neurocontroller.

Ghaboussi and Joghataie (1995) and Chen et al. (1995) first introduced the ANN concept for structural vibration control by showing the basic process of neurocontrol with simple application examples. A number of neurocontrol methods have been proposed subsequently (Bani-Hani and Ghaboussi 1998a,b; Nikzad and Ghaboussi 1996). The multilayer neural network (MLNN) is used in all these works; it is the most widely used type of neural network due to the simplicity of both its structure and learning algorithm. However, the training convergence of the MLNN is inherently very slow because of the structure of the MLNN, which updates all the weights simultaneously for a given training pattern. Therefore, weights trained by one training pattern are always interfered with by those of another pattern, which interference inevitably delays the convergence. This kind of learning is the so-called global learning. On the other hand, the training of CMAC (cerebellar model articulation controller) is done very locally; in other words, interference by training patterns is minimized. Two input patterns with long hammering distance cause almost no interference with each other (Albus 1975), with the result that the convergence of CMAC training becomes quite a bit faster than that of MLNN.

Nevertheless, the CMAC has not yet been applied to the structural control, but has been used only for process or motion control, which requires predefined target output (Lane et al. 1992; Patel and Carroll 1995). This study therefore explores the applicability of the CMAC to structural control. Contrary to process or motion control, usually a target output is unknown in a structural control. In addition, the training rule for the CMAC is proposed based on cost function and evaluated sensitivity data. In the numerical simulation, the training epochs and controlled responses by the CMAC and MLNN are compared.

Cerebellar Model Articulation Controller (CMAC)

History

The CMAC was first proposed by J. S. Albus (1975) and has since been modified and improved. A concept of basis function has been proposed by many researchers in order to find derivatives information from the trained CMAC. Spline function was introduced to the basis function of the CMAC (Lane et al. 1992), and then the fuzzy membership function was used as a basis (Nie and Linkens 1993; Patel and Carroll 1995), which is the so-called fuzzified CMAC, or simply FCMAC. Recently, the generalized...
CMAC, or so-called GCMAC, was proposed (Chiang and Lin 1996; Gonzalez-Serrano et al. 1998). The GCMAC generalizes the CMACs proposed in the other studies by introducing the general basis function, which includes all the possible functions. The performance of the CMAC could be expanded by the improvements, but on the contrary, the learning convergence becomes slow. Therefore, the CMAC originally proposed by Albus is used in this study, since it keeps the fast convergence property. The following sections describe the computational steps of the CMAC, such as transforming the input space, hash mapping, output calculation, and the learning algorithm.

Transforming Input Space

This step may not be required for other applications, but it is very useful to transform the input space of the CMAC to a nonnegative one for structural application. This can be easily done by

\[ x_k = x_k - x_{k, \text{min}}; \quad k = 1, 2, \ldots, N \]  

where \( x_{k, \text{min}} \) is the minimum of \( x_k \), and \( N \) is the dimensionality of the input space.

Hash Mapping

Case 1: Single Input Single Output (SISO)

To explain the concept of the CMAC, the case of a single input and single output (SISO) is considered. The variable of the transformed input space is denoted by \( x' \), as in Fig. 1. In the first layer, the input space is parted into five divisions, which division is the so-called hypercube. Now, five weights from \( W_1^1 \) to \( W_5^1 \) are allocated to each hypercube. Then, the second layer can be added by shifting the first layer to \( S_{21} \). The \( S_{ik} \) means the amount of shift in the direction of the \( k \)th degree of freedom (DOF) on the \( l \)th layer. Therefore, \( S_{11} \) necessarily becomes zero because the first layer is the basis. Another layer can be added in this manner. The more layers are used, the finer the output resolution that can be obtained; but the required size of memory increases according to the number of layers. One should be careful in adding the layers so that none of the shifted layers coincide with one another.

In this example, a total of four layers are used. Therefore, the size of the memory is \( 4 \times 5 = 20 \). From this memory structure, the activated weights according to any input can be decided. For example, if the CMAC receives \( x' \) as an input, then the four weights \( (W_1^1, W_2^1, W_3^1, W_4^1) \) are activated on each layer. Only these four weights are used in making output and also in training. If another input, \( x'' \), is received, then another set of weights \( (W_4^1, W_4^2, W_3^3, W_3^4) \) is activated.

Case 2: Multi-Input and Single Output (MISO)

Fig. 2 shows a CMAC memory structure with two inputs and a single output. Each of the input spaces, \( x_1' \) and \( x_2' \), is divided into several blocks whose widths are \( q_1 \) and \( q_2 \), respectively. More layers can be added to enhance the resolution of the output. But only two layers are used to simplify the problem. The second layer is obtained by shifting the first layer to \( S_2 = [S_{21}, S_{22}] \), as in Fig. 2(b). Then one can activate a set of weights for any input based on the structure. For example, an input \( X^* = [x_1^*, x_2^*]' \) invokes \( W_{33}^1 \) and \( W_{34}^2 \) on each layer. These two weights are used in the output calculation and in the training.

The general rule for the subscripts of activated weight on the \( l \)th layer, \( W_{I_1 I_2}^l \), can be derived as follows. On the first layer, the two subscripts can be derived as

\[ I_1 = \text{ceil} \left( \frac{x_1 - S_{11}}{q_1} \right) \]  
\[ I_2 = \text{ceil} \left( \frac{x_2 - S_{12}}{q_2} \right) \]

where the \( \text{ceil}(\cdot) \) operator means the smallest integer greater than the input value. There is no shift in the first layer; hence \( S_{11} \) and \( S_{12} \) are dummy values used only to generalize the formulation. In
the second layer, the values differ slightly from the above equations because shifts occur there in each direction. Considering the shifts, the second layer equations can be derived as

\[ I_1 = \text{ceil} \left( \frac{x_1 - S_{11}}{q_1} \right) + 1 \]  
\[ I_2 = \text{ceil} \left( \frac{x_2 - S_{22}}{q_2} \right) + 1 \]

Each subscript is increased by one because the first index should be assigned to the newly made region by shift. If another layer is added, one can obtain the general equations for the subscripts of invoked weights on the \( l \)th layer as

\[ I_1 = \text{ceil} \left( \frac{x_1 - S_{1l}}{q_1} \right) + \text{ceil} \left( \frac{S_{1l}}{q_1} \right) \]  
\[ I_2 = \text{ceil} \left( \frac{x_2 - S_{2l}}{q_2} \right) + \text{ceil} \left( \frac{S_{2l}}{q_2} \right) \]

In this manner, the hash mapping scheme can be easily expanded to the case of \( N \) inputs and a single output. If the CMAC receives an input \([x_1, x_2, \ldots, x_N]\) and the invoked weights on the \( l \)th layer by the input are denoted by

\[ W_l = W_{l1} \cdot W_{l2} \cdots W_{lN} \]  
then the subscript can be obtained as

\[ I_j = \text{ceil} \left( \frac{x_j - S_{ij}}{q_j} \right) + \text{ceil} \left( \frac{S_{ij}}{q_j} \right) ; \quad j = 1, 2, \ldots, N \]

Usually, the layers are added in order to be equally spaced from one another as follows:

\[ S_{ij} = (l-1) \frac{q_j}{L} \]

where \( L \) is the total number of layers. Therefore, applying Eq. (10) to Eq. (9) gives the index of the invoked weights on the \( l \)th layer as follows:

\[ I_j = \text{ceil} \left( \frac{x_j - l - 1}{L} \right) + \text{ceil} \left( \frac{l - 1}{L} \right) ; \quad j = 1, 2, \ldots, N \]

### Calculating Output and Learning

A CMAC output is obtained by summing all invoked weights as follows:

\[ u_{\text{CMAC}} = \sum_{l=1}^{L} W_l^j \]

The equation means that the equal information on the output is stored in each layer, and hence output is obtained by summing all the invoked weights on all layers.

The learning rule for the CMAC can be derived based on the cost function, such as

\[ J = \frac{1}{2} \sum_{\mu=1}^{P} (t^\mu - u_{\text{CMAC}}^\mu)^2 \]

where \( P \) is the number of training patterns, and \( t \) is the target output. The weight update rule can be found by applying the so-called gradient descent rule to Eq. (13) as follows:

\[ \Delta W_l^j = -\frac{m}{L} \frac{\partial J}{\partial W_l^j} \]

where \( m \) is the learning rate. Here, \( L \) is used to equally distribute the amount of updates to \( L \) invoked weights. Differentiating the cost gives the final learning rule as

\[ W_l^j |_{\text{new}} = W_l^j |_{\text{old}} + \frac{m}{L} \sum_{\mu=1}^{P} (t^\mu - u_{\text{CMAC}}^\mu) \]

### Required Size of Memory

The basic idea of the CMAC is to store information in a table look-up fashion. However, the size of the memory is not considerably increased due to the hash mapping algorithm. If one needs the resolution that produces \( \gamma \) different outputs, and there are \( N \) dimensional spaces with the same resolutions, then the required size of the conventional table look-up (TLU) scheme is

\[ N_w,\text{TLU} = \gamma^N \]

In the hash mapping, however, the size is relatively quite small. If the number of block divisions on each layer is \( \delta_1, \delta_2, \ldots, \delta_n \), respectively, and the number of layers is \( L \), then the memory size of the CMAC is

\[ N_w,\text{CMAC} = L \prod_{i=1}^{N} \delta_i \]

Now assume that the number of divisions in each dimension is the same, as follows:

\[ \delta_1 = \delta_2 = \cdots = \delta_N \]

To achieve the same resolution as a table look-up scheme, the number of layers should be

\[ L = \gamma / \delta \]

Then, the total number of the CMAC size becomes

\[ N_w,\text{CMAC} = (\gamma / \delta)^N \]

Finally, the ratio of the two memory structures is

\[ \frac{N_w,\text{CMAC}}{N_w,\text{TLU}} = (\delta / \gamma)^{N-1} \]

One can easily know from this ratio that the hash mapping becomes more effective as the dimensionality of the input space increases.

### Structural Control Using CMAC

#### Control Algorithm and Training Rule

Fig. 3 shows the block diagram of neurocontrol using the CMAC, which serves as a controller in the diagram. To train the CMAC, a cost function composed of the structural response and the control signal is defined as

\[ J = \sum_{k=0}^{N_f-1} J_k = \sum_{k=0}^{N_f-1} \frac{1}{2} (z_{k+1}^T + Qz_{k+1} + u_k^T R u_k) \]

where \( z(n \times 1) \) and \( u(m \times 1) \) are the state and the control signal; \( Q(n \times n) \) and \( R(m \times m) \) are the weighting matrices; and \( k, N_f, \) and \( T_s \) are the sampling number, the total number of the sampling time, and the sampling interval, respectively. The first term of the
The equation of motion of a linear system can be written as

\[ \dot{z} = Az + Bu \]  (28)

where \( A(n \times n) \) and \( B(n \times m) \) denote the system matrix and location matrix of the controller, respectively. The discretized equation based on the zero-order-hold approximation can be expressed as

\[ z_{k+1} = Gz_k + Hu_k \]  (29)

where \( G(n \times n) \) and \( H(n \times m) \) are expressed in terms of \( A \) and \( B \) as

\[ G = e^{AT}\nu \]  (30)

\[ H = (e^{AT} - I)A^{-1}B \]  (31)

In a linear system, it can easily be found that the matrix \( H \) is only the function of the sampling time and is constant for a fixed sampling time. By differentiating Eq. (29) with respect to the control signal at the \( k \)th step, the following sensitivity equation is obtained:

\[ \frac{\partial z_{k+1}}{\partial u_k} = H \]  (32)

where Eq. (32) means that the matrix \( H \) equals the response sensitivity to the control signal. Therefore, if \( H \) is found in some way, the emulator neural network becomes obsolete, and \( H \) can simply be found as follows. If all the initial states are set to zero, and the unit control signal is applied only to the \( i \)th controller, namely

\[ z_0 = 0 \]  (33)

\[ u_{j,k} = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \quad j = 1, 2, \ldots, m \]  (34)

then Eq. (29) becomes

\[ z_{k+1} = h_i \]  (35)

where \( h_i \) denotes the \( i \)th column of \( H \). By measuring the state at the \( (k+1) \)th step, the response sensitivity to the control signal of the \( i \)th controller can be found. If another unit input signal is applied to the system, the sensitivity vector corresponding to the controller is obtained. Therefore, the full matrix \( H \) can be found. Although this is derived by assuming the linearity of structural
Parameters, the obtained sensitivity matrix can be used for the training of nonlinear structural control. This is well described in Kim and Lee (2001).

Limitations of CMAC

Although the CMAC quickly learns the functional input-output relationship, it should be noticed that it has some limitations. Since the CMAC stores information to a few localized memories, the size of the memory is enormously increased in storing full information. In addition, since any one instance of training data is used to train only a few localized weights, the training data should be scattered as uniformly as possible to have generalization potential. This usually increases the size of the training data, and therefore the memory mapping and generalization algorithm should be developed to extend the applicability of the CMAC.

Structure with Active Mass Damper (AMD) System

The three-story shear building structure in Fig. 4 is used for the following numerical examples (Bani-Hani and Ghaboussi 1998b). The hydraulic actuator is excited by the control signal generated from the neurocontroller. Then the active mass damper (AMD) attached to the actuator applies reaction force to the structure. Since the dynamics of the actuator and the structure are coupled, actuator dynamics should be included in the analysis.

Hydraulic Actuator

The equation of motion of the hydraulic actuator can be divided into two parts, the valve dynamics and the piston equation. First, the valve equation is expressed as

$$\frac{\tau}{g_{1}} + \frac{1}{g_{2}} q = u$$

(36)

where \(g_{1}\) and \(g_{2}\) denote the valve gains; \(\tau\) is the time constant of the valve; and \(q\) and \(u\) are the flow rate of oil and the control signal, respectively. The change of oil flow through the valve induces the piston motion, and the relationship can be modeled by

$$a_r x_r + \frac{c_l}{a_r} f + \frac{V}{2\beta a_r} f = q$$

(37)

where \(a_r\), \(\beta\), \(c_l\), and \(V\) denote the area of the piston, the compressibility coefficient, the leakage coefficient, and the volume of the cylinder, respectively; \(x_r\) is the relative displacement between the roof and the piston; and \(f\) is the force exerted on the structure by the AMD.

Three-Story Shear Building

The equation of motion of the three-story building with an AMD can be expressed as

$$M \ddot{x} + C \dot{x} + K(x, \dot{x}) = Lf - M[1] \ddot{x}_g$$

(38)

where \(M\) and \(C\) are \((4 \times 4)\) mass and damping matrices; \(x\) is the \((4 \times 1)\) relative displacement vector, consisting of three stories and an AMD; \(K(x, \dot{x})\) is the \((4 \times 1)\) restoring-force vector; \(L\) is the \((4 \times 1)\) vector indicating the location of the actuator; \(x_g\) is ground acceleration; and \([1]\) is the direction vector of the ground motion.

Nonlinear Dynamic Model

The nonlinear hysteretic model (Baber and Wen 1981) is used to simulate the motion of the nonlinear structure. This model has been used for the control simulation in many works (Bani-Hani and Ghaboussi 1998b; Kim and Lee 2001). The restoring force of the model is composed of the linear and the nonlinear terms as

$$k_s(x_s, \dot{x}_s) = ak_0 x_s + (1 - \alpha) k_d y$$

(39)

where \(x_s\) denotes the interstory displacement; \(k_0\) and \(\alpha\) are the linear stiffness and its contribution to the restoring force, respectively; and \(d\) and \(y\) are a constant and a variable, respectively, satisfying the following equation:

$$\dot{y} = \frac{1}{d} (\rho \dot{x}_s - \mu \| \dot{x}_s \| y |y|^{-1} y - \sigma \ddot{x}_s |y|)$$

(40)

where \(\rho\), \(\mu\), and \(\sigma\) are the constants that affect the hysteretic behavior.

Numerical Examples

Simulation Parameters

The structural properties are as follows: the story mass is 200 kg; the interstory stiffness is \(2.25 \times 10^5\) N/m; and the damping ratios of the three modes are 0.6, 0.7, and 0.3%, respectively. The parameters of the AMD are designed based on the suggestions of Soong and Dargush (1997) for the optimal tuned mass damper. The mass of the AMD is 18 kg, which corresponds to 3% of the total mass of the structure. The actuator parameters are used as follows (Bani-Hani and Ghaboussi 1998b): \(g_1 = 32.14 \times 10^2\) and \(g_2 = 2.8 \text{ m}^3/\text{s}\); \(\tau = 0.1\); \(a_r = 1.52 \times 10^{-3}\) m²/s; \(V = 4.56 \times 10^{-2}\) m³; \(c_l = 1.0 \times 10^{-11}\); and \(\beta = 2.1 \times 10^7\). For the linear structure \(\alpha = 1.0\) is used, and for the nonlinear structure, \(\alpha = 0.5\). For the nonlinear model \(d = 0.01\); and \(\rho\), \(\mu\), \(\sigma\), and \(p\) are 1.0, 0.5, 0.5, and 5.0, respectively. The sampling time is 0.005 s, and the delay time is assumed to be 0.0005 s. The equation of motion is integrated at
used for the training criterion is responses participates in the cost function. The cost at the step follows:

\[ J_k = z_{3,k+1}^T Q z_{3,k+1} + ru_k^2 \]  

where \( z_{3,k+1} \) and \( u_k \) denote the state of the third floor and control signal, respectively, and the weighting matrix \( Q \) and \( r \) are as follows:

\[
Q = \begin{bmatrix}
\frac{1}{\bar{x}_3} & 0.0 \\
0.0 & \frac{1}{\bar{x}_3}
\end{bmatrix} \quad r = 0.1 \left| \frac{1}{\bar{u}} \right|^{1.0} 
\]

In these equations, \( \bar{x}_3 \) and \( \bar{x}_3 \) are the maximum displacement and velocity of the third floor when the control input signal is off; \( \bar{u} \) is the maximum control input voltage.

The CMAC has two inputs: relative displacement and velocity of the third floor. Each input space has two quantization blocks, and the generalization width is 200. Therefore, the total number of weights, \( N_w \), equals 200!1(2+1) = 1,800.

For the MLNN the same input information and learning rate are used, as in the case of the CMAC. The number of hidden layer nodes is chosen optimally by trial and error. The MLNN is trained by the well-known delta learning rule with momentum constant 0.7. The final epochs and costs for both the CMAC and MLNN cases are compared in Table 3. The training speed of the CMAC is absolutely faster than that of the MLNN, but the costs converge to a slightly larger value. This may cause larger structural vibration when the CMAC is used, but is not serious compared to the MLNN.

### Control Results

The controlled displacements of the third floor by two controllers, CMAC and MLNN, are compared in Fig. 5 during the El Centro earthquake. Although the controlled displacement by CMAC is slightly larger than that of the MLNN, the vibration reduction performance of the CMAC is still satisfactory. Fig. 6 shows the control signal produced by the CMAC, displacement of the AMD, and force exerted on the structure by the AMD during El Centro earthquake, respectively. Fig. 7 shows the control effect under the other two earthquakes that are not considered in the training phase. Though the CMAC is not trained under the two ground motions, the control effect is acceptable.

For the nonlinear case, the interstory relationships of the restoring force to displacement are compared in Fig. 8, which shows that the proposed control scheme is also applicable to nonlinear structural control.

### Conclusions

Structural control using a neural network has been widely investigated in recent years. In almost all cases, the MLNN is used for the controller. One of the drawbacks of the MLNN is the slow convergence of training. To improve the convergence of the controller, the CMAC was first applied to the structural control in this study. A training algorithm for the CMAC as a structural controller was newly proposed. In numerical examples, the structural vibration induced by ground excitation can be effectively controlled by the trained CMAC, and the response reduction capability of CMAC is comparable to that of the MLNN. However, the controlled response by the CMAC is slightly larger than that

---

**Table 2. Response Sensitivities to Control Signal**

<table>
<thead>
<tr>
<th>Sensitivity</th>
<th>Exact</th>
<th>Calculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displacement: m/volt</td>
<td>4.433×10^{-5}</td>
<td>4.436×10^{-5}</td>
</tr>
<tr>
<td>(ratio)</td>
<td>(1.00)</td>
<td>(1.001)</td>
</tr>
<tr>
<td>Velocity: m/s/volt</td>
<td>2.443×10^{-2}</td>
<td>2.441×10^{-2}</td>
</tr>
<tr>
<td>(ratio)</td>
<td>(1.00)</td>
<td>(0.999)</td>
</tr>
</tbody>
</table>

---

**Table 3. Final Epochs and Costs after Training**

<table>
<thead>
<tr>
<th></th>
<th>Linear Case</th>
<th>Nonlinear Case (α = 0.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Epoch</td>
<td>Cost</td>
</tr>
<tr>
<td>MLNN</td>
<td>412</td>
<td>1.77×10^{-2}</td>
</tr>
<tr>
<td>(1.00)</td>
<td>(1.00)</td>
<td>(1.00)</td>
</tr>
<tr>
<td>CMAC</td>
<td>65</td>
<td>1.94×10^{-2}</td>
</tr>
<tr>
<td>(0.15)</td>
<td>(1.09)</td>
<td>(0.08)</td>
</tr>
</tbody>
</table>

---

Fig. 5. Controlled responses by MLNN and CMAC (under El Centro earthquake)
Fig. 6. Control signal, AMD displacement, and control force under El Centro earthquake

Fig. 7. Displacement under untrained earthquakes: (a) Northridge earthquake; (b) Kern County earthquake

Fig. 8. Restoring force versus displacement under Kern County earthquake (first floor)
by the MLNN. This is mainly caused by the fact that the memory storing capability of the CMAC is not yet enough for structural control. This should be improved before it is put into practice.

References


