JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 20, No. 3, May–June 1997

Missile Control Using Fuzzy Cerebellar Model Arithmetic Computer Neural Networks

Z. Jason Geng*
Genex Technologies, Inc., Rockville, Maryland 20850
and
Claire L. McCullough†
U.S. Army Space and Strategic Defense Command, Huntsville, Alabama 35807-3801

A novel fuzzy neural network control system architecture, called the fuzzy cerebellar model arithmetic computer (fuzzy CMAC) neural network, is proposed. This fuzzy neural network architecture exploits a synergism between the original CMAC neural network and the theory of fuzzy logic controller. The fuzzy CMAC is able to perform arbitrary function approximation and can achieve a higher learning speed and greater accuracy than the original CMAC. It also has the capability of conveniently incorporating human knowledge into the system and processing information based on fuzzy inference rules. We evaluated the use of fuzzy CMAC in the design of advanced missile control systems. The method is demonstrated by missile interception simulations, and promising results are obtained.

I. Introduction

INTELLIGENT control technology plays a major role in modern flight control system design and implementation. The first goal of the intelligent control approach is to make advanced control systems easier to design. The second goal is to make them less vulnerable to uncertainties in system components and the environment and to enable them to handle unmodeled dynamic characteristics. To achieve these goals, control systems should have the capability to gain increasing knowledge of the system through operational experience, without the interference of human operators. A self-learning and adaptive control approach that explicitly handles nonlinearity and uncertainty would result in a significant improvement in many applications. Two very popular approaches for performing real-time control are neural nets and fuzzy logic.

Fuzzy logic controllers (FLCs)[1–3] have several important benefits in that they do not require a complete analytical model of a dynamic system, they provide knowledge-based heuristic controllers for ill-defined and complex systems, and they can be analytically validated. However, FLCs are neither primarily designed for nor well suited to learning. This means that they cannot meet the goals of adaptation to changes in system dynamics or to unmodeled dynamic characteristics, and they cannot gain increased performance through learning.

Neural networks have been evaluated extensively for use in real-time control systems with some success.[3–9] The neural nets based on multiple-layer feedforward structure and backpropagation learning algorithms are, in general, computationally intensive. To learn a single element of new data, all neurons in the network could be affected. A cerebellar model arithmetic computer (CMAC) neural network[4,5] is a type of neural network that has a faster learning rate than conventional neural nets and has a limited amount of computation required at any point in the learning process. CMAC neural nets can be implemented on feasible hardware and can operate in real time. On the other hand, CMAC networks are difficult to design, and their final learning accuracy has been limited by a fundamental quantization of the input space.

As two separate control schemes, both CMAC neural networks and FLCs have strengths and weaknesses for intelligent control systems. However, there has been no research activity reported that bridges the connection between these two approaches. In this paper, an innovative fuzzy neural network control system architecture, called the fuzzy CMAC neural network, is proposed and evaluated for the use in the design of advanced missile control systems. This fuzzy neural network architecture exploits a synergism between the original CMAC neural network developed by Albus[4] and the theory of FLC proposed by Zadeh.[2] The fuzzy CMAC is able to perform arbitrary function approximation and can achieve higher learning speed and greater accuracy than the original CMAC. It also has the capability of conveniently incorporating human knowledge into the system and processing information based on fuzzy inference rules.

Our initial study results, detailed in the following sections, prove that this new fuzzy CMAC has many advantages. The learning rates are faster than the original CMAC neural networks, and the Universality Theorem we have formally proven shows that any function can be learned to any degree of accuracy with enough learning cycles, thus eliminating a weakness of CMACs. We have proven the ease of use and elegance of fuzzy CMACs by implementing a simulated flight control system. The flight control system that results from our work is evaluated using a series of nonlinear simulations driven by mathematical models of HAVE DASH II Bank-to-Turn (BTT) missile to examine the stability, high-angle-of-attack tracking, flight-path angle tracking, and ultimately the target interception maneuvers.

The rest of this paper is organized as follows. Section II briefly discusses the fuzzy CMAC neural network structure. Section III establishes the nonlinear dynamic model of missile motion. The model will be used in our simulation for validation of the control system design. Section IV illustrates a guidance law for a missile–target interception maneuver. Section V presents an inner-loop controller design using a linearized model and eigenstructure assignment. Section VI details the outer-loop controller design based on fuzzy CMAC neural networks. The guidance law and the control laws for the inner and outer loops provide a missile with a comprehensive integrated guidance and control system design. Section VII gives some conclusions of the study.

II. Fuzzy CMAC Neural Networks

A. Background of CMAC Neural Networks

Inspired by the human cerebellum structure and the neuromuscular control mechanism, a unique neural network model CMAC was established two decades ago by Albus.[4] Figure 1 shows schematically a typical CMAC network, where the CMAC is designed to approximate a single-valued nonlinear mapping $\boldsymbol{f} : \boldsymbol{u} \rightarrow \boldsymbol{z}$ from a multidimensional real-valued input space R^n to a multidimensional real-valued output space R^m.

Received Sept. 25, 1995; revision received Aug. 20, 1996; accepted for publication Dec. 13, 1996.

*President, 4950 Cloister Drive.
†Senior General Engineer, Sensor Directorate.

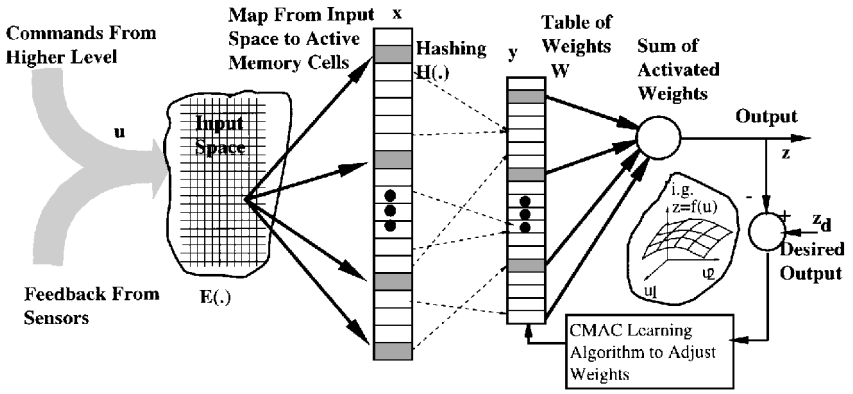


Fig. 1 Diagram of a CMAC neural network.

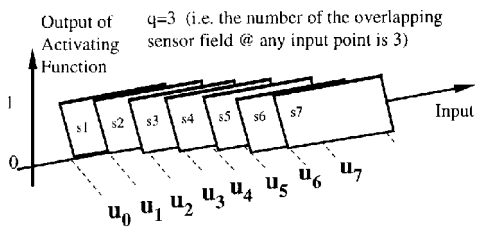


Fig. 2 CMAC binary activating function.

The mapping $\boldsymbol{f}$ is assumed to be a piecewise continuous function, and there is a metric measure defined in the output space (e.g., Euclidean, Hamming, or Manhattan metric). The accuracy of the approximation will be improved through learning based on that metric measure.

Architecturally, overall CMAC mapping $\boldsymbol{f}: \boldsymbol{u} \rightarrow z$ can be split into three separate modules, namely, input encoding mapping $\boldsymbol{E}(.): \boldsymbol{u} \rightarrow \boldsymbol{x}$, hash-code based mapping $\boldsymbol{H}(.): \boldsymbol{x} \rightarrow \boldsymbol{y}$, and a weighted summation function $\boldsymbol{W}: \boldsymbol{y} \rightarrow z$.

The input encoding scheme $\boldsymbol{E}(.): \boldsymbol{u} \rightarrow \boldsymbol{x}$ generates a nonlinear mapping, in a look-up table fashion, from input space $\boldsymbol{u}$ into a finite number of memory locations $\boldsymbol{x}$. In contrast to a conventional look-up table that assigns only one memory location for each possible input state, the CMAC encoding scheme activates q ($q > 1$) memory cells for each input state (the value of q can be selected by the designer). One unique feature of the CMAC, called generalization, guarantees that the q memory cells assigned to an input state will have $q-1$ memory cells overlapping with the memory cells assigned to its neighboring input state, as shown in Fig. 2. The results of the generalization are that the input resolution is increased, the memory requirement is reduced, and the smoothness of function mapping is improved in the sense that similar input states will always generate similar outputs.

To further reduce the physical memory size requirement, a hash mapping $\boldsymbol{H}(.): \boldsymbol{x} \rightarrow \boldsymbol{y}$ is employed from a large memory space $\boldsymbol{x}$ to a smaller physical memory space $\boldsymbol{y}$ where a binary value of the elements in the $\boldsymbol{y}$ vector shows the memory cell activation status (1 for activated and 0 for nonactivated). Each memory cell location in $\boldsymbol{y}$ is associated with a weight element in the weight matrix $\boldsymbol{W}$. For each input state, there will be q activated elements in $\boldsymbol{W}$. Finally, the real valued output vector z of a CMAC is generated by summing the activated weights:

$$z = \boldsymbol{W}^T \boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{H} \boldsymbol{x} = \boldsymbol{W}^T \boldsymbol{H} \boldsymbol{E}(\boldsymbol{u}) \tag{1}$$

Notice that the value of the ith element in z is the sum of the ith raw weight matrix elements whose position coincides with the nonzero elements of the $\boldsymbol{y}$ vector.

Multiple iterations of training are required by a CMAC to determine appropriate values of its weight matrix elements to represent a given nonlinear mapping function $\boldsymbol{f}$ over the region of interest. A supervised learning scheme is used based on observed training data pairs $\{u_i, z_{di}\}$, where i denotes the number of learning iterations and z_{di} is the desired function value with respect to u_i. The learning rule of a CMAC is given as a linear adaptation in multidimensional space,

$$\boldsymbol{W}_{i+1} = \boldsymbol{W}_i + \beta \boldsymbol{y} \frac{(z_{di} - z_i)^T}{\boldsymbol{q}} \tag{2}$$

where W is the CMAC weight matrix, β is the learning rate scaled between 0 and 1, z_i is CMAC's actual output in response to input state u_i, and q is the total number of weights activated by an input.

The CMAC neural network has the following features that are desirable to real-time nonlinear system modeling and control applications: 1) table lookup (able to represent a wide variety of functions, nonlinear, multi-input/multi-output mapping); 2) self-learning (least mean square law for partial weights updating, unique minimum); 3) local generalization (close inputs give similar outputs even when the input is not trained, less learning interference than multilayer perceptron); 4) computationally inexpensive network mapping and learning algorithms (only activated weights need to be calculated and adjusted, in contrast to multilayer perceptron networks where all of the weights need to be calculated for every network mapping and learning cycle); 5) simple, straightforward, and hardware implementation possible (real-time processing); and 6) implementable using parallel processing, scalable to large number input/output networks.

B. Comparison Between CMAC and Fuzzy Controller

The advantages of CMAC over FLC are as follows.

1) There are very efficient learning laws to update the values of weights based on experience and examples.

2) There is a random mapping mechanism to reduce the physical memory requirement for multiple input and high-resolution applications.

3) There exist efficient input encoding schemes for high-dimensional input vectors.

The advantages of FLC over CMAC are as follows.

1) It is possible to interpret the implication of weight values using linguistic labels.

2) The membership functions and the firing strengths contain additional information as to how close the input vector is to each linguistic variable. Therefore, the number of input space partitions may be smaller to achieve the same generalization and output smoothness.

3) The fuzzy rules can take a variety of forms, whereas only numeric values can be associated with CMAC associative memory locations.

4) There are many methods to construct a fuzzy control knowledge base, such as expert's experience and knowledge, modeling of the operator's control actions, modeling of a process, and self-organization.

C. Extension of CMAC to Fuzzy Neural Networks

One drawback of the original CMAC, however, is that the input activating function is binary, meaning that a memory cell is either associated with an input state or totally disconnected from this input state. Because of the binary nature of input encoding, CMAC output function is often quite jumpy and exhibits many discontinuities

even when a continuous and smooth input signal is fed to a CMAC. To enhance the smoothness of the output and reduce the learning error, a large number of memory cells need to be used, which increases the complexity of CMAC implementation and compromises the real-time performance. The output of a CMAC is not differentiable with respect to the input, resulting in complications for using gradient descent learning rules.

Instead of using the binary input activating function, the proposed fuzzy CMAC adopts the concept of fuzzy membership function and uses a family of continuous smooth functions as the input activating functions of CMAC. The weight elements in the fuzzy CMAC neural networks are associated with input state based on the firing strengths (between 0 and 1) of their fuzzy membership functions. The output of the fuzzy CMAC is the sum of activated weight values weighted by the firing strengths of the fuzzy membership functions.

Figure 3 shows a comparison between the CMAC binary input activating function and the fuzzy input activating function in the fuzzy CMAC neural network. As the result of introducing the fuzzy membership functions to CMAC neural networks, not only does the output function of the neural network become much more smooth and the learning error is greatly reduced, but also fewer memory cells are required and fewer learning cycles are needed to achieve the same level of learning accuracy. Furthermore, the learning results of the neural network can now be interpreted using fuzzy linguistic variables. This feature permits us to validate the reasonableness of the neural network learning results. This unique feature also provides a practical channel for knowledge acquisition. A knowledge base can be established based on the learning results of a fuzzy CMAC.

Figure 4 illustrates the architecture of the fuzzy CMAC neural network for a two-inputs/one-output case. The fuzzy CMAC inherits preferred features of arbitrary function approximation, self-learning, and parallel processing from the original CMAC neural network, as well as the capability of acquiring and incorporating human knowledge into a system and the capability of processing information

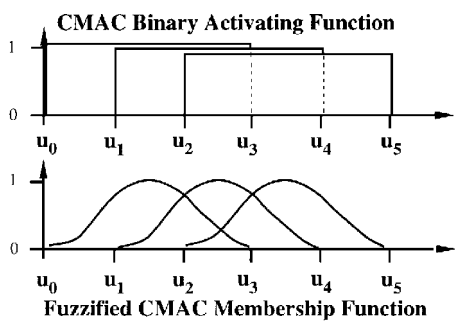


Fig. 3 Activating function comparison between CMAC (binary) and fuzzy CMAC (continuous).

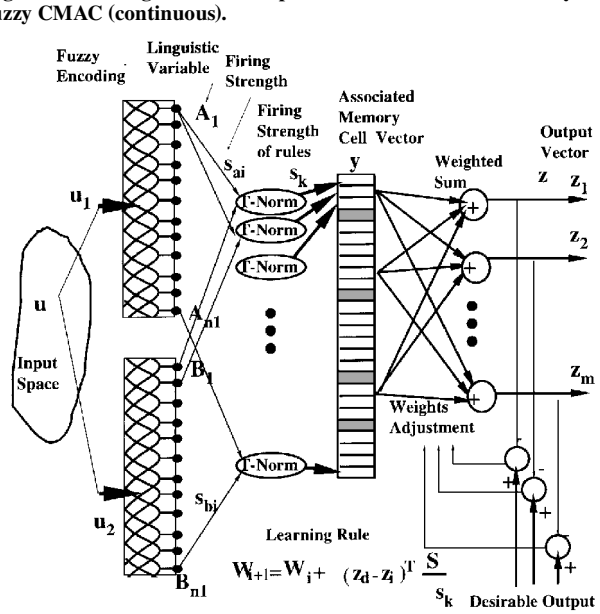


Fig. 4 Architecture of fuzzy CMAC neural networks.

based on fuzzy inference rules from the fuzzy logic. The fuzzy CMAC neural network, as a synergistic combination of the CMAC and the fuzzy logic, is well suited for solving nonlinear system real-time control and identification problems in many application areas.

The fuzzy CMAC controller consists of the following functional modules.

1. Input Encoder

The fuzzy CMAC inherits a desirable feature from the CMAC model in that the receptive field of the sensing element has limited width. This means that there are only a small number of sensing elements to be activated for any sensor reading. Comparing the activating function of CMAC to the linguistic variables in the fuzzy CMAC, one can view the activating function as the membership function of the fuzzy CMAC input variables. Figure 3 shows the comparison. The membership function with limited span leads to a very efficient way to implement the associative memory and learning algorithm, while it guarantees that neighboring locations in an input space share a number of memory locations and produces a desirable generalization between similar input vectors.

As an example, the third-order spline function can be employed as the template of fuzzy membership functions. The spline type of membership functions have the following very desirable properties.

1) Positivity: $\mu_i(u) > 0$ for all $u \in (u_{i-3}, u_i)$.

2) Compact support: $\mu_i(u) = 0$ for all u not belonging to (u_{i-3}, u_i).

3) Normalization:

$$\sum_i \mu_i(u) = 1$$

for any u.

4) Derivatives exist and can be recursively calculated.

It is often necessary to compute derivatives of a fuzzy CMAC mapping, such as when pack-propagating error through a neural network model. The fact that the derivative of a learned mapping can be calculated facilitates many real-time control applications, such as active vibration control.

For an input vector presented to the fuzzy CMAC, only a small number of membership functions will be fired and, therefore, need to be computed. This represents a considerable amount of computation savings when compared to a general feedforward backpropagation neural network, which must calculate in each cycle a fully connected network with all weights activated.

2. Control Knowledge Rule Base

The proposed fuzzy CMAC differs from the Albus[4] CMAC in that the weight values can be interpreted as knowledge rules through linguistic variables defined for a particular application. This feature permits us to validate a learned fuzzy CMAC and guarantees the reasonableness of the learning results. This unique feature also provides a practical channel for knowledge acquisition. A knowledge base can be established based on the learning results of a fuzzy CMAC.

On the other hand, the fuzzy CMAC distinguishes itself from Zadeh's[2] fuzzy controller in that it incorporates a combination of unique input encoding scheme, hash mapping, and powerful and efficient self-learning capability inherited from the CMAC. Unlike a fuzzy controller, the fuzzy CMAC is able to build a self-learning control system starting with an empty knowledge base.

For a fuzzy CMAC application, we construct the initial fuzzy IF-THEN rules in the following form.

Rule k:

$$\text{IF } x^1 \text{ is } F_{k1}^1 \text{ and } x^2 \text{ is } F_{k2}^2 \text{ and } \cdots \text{ and } x^N \text{ is } F_{kN}^N, \text{ THEN } y \text{ is } Y^k$$

where Y^k are fuzzy sets defined in the output space. The linguistic knowledge of human experts can be incorporated into these rules. Based on the initial knowledge base, a self-learning algorithm can be employed to modify the existing rules to improve system performance.

3. Fuzzy CMAC Mapping Function

According to a general rule of fuzzy CMAC, the output of a fuzzy CMAC is given by

$$z(\boldsymbol{u}^l) = \left\{ \left[\sum_{j_N=1}^{m_N} \sum_{j_{N-1}=1}^{m_{N-1}} \cdots \sum_{j_1=1}^{m_1} \prod_{i=1}^{N} \mu_{i,j_i}(u_i)^* w(j_1, j_2, \ldots, j_N) \right] \Big/ \left[\sum_{j_N=1}^{m_N} \sum_{j_{N-1}=1}^{m_{N-1}} \cdots \sum_{j_1=1}^{m_1} \prod_{i=1}^{N} \mu_{i,j_i}(u_i) \right] \right\} \tag{3}$$

It seems from Eq. (3) that the calculation of the numerator, as well as the denominator of the $z(\boldsymbol{u}^l)$, involves a summation of

$$M = \prod_{i=1}^{N} m_i$$

items; each in turn requires N (or $N-1$) multiplications.

By using the salient property of B-spline membership function, the calculation of the $z(\boldsymbol{u}^l)$ becomes much simpler.

We have proven a universal approximation theorem for the fuzzy CMAC neural network. The theorem essentially guarantees that the fuzzy CMAC neural network is able to uniformly approximate any nonlinear continuous function over a specific compact region to any degree of accuracy. The universal approximation theorem provides us with justification for applying the fuzzy CMAC to almost any nonlinear system modeling problems. Details on the universal approximation theorem can be found in Ref. 10.

4. Self-Learning Algorithm of the Fuzzy CMAC

The conventional CMAC self-learning process is designed so that the correction initiated by an output error is evenly distributed among all weights that contribute to the output, regardless of their true proportion of contribution to the output. In the fuzzy CMAC, a more intelligent method is adopted in which the correction of output error will be distributed to the weights in proportion to their contribution. The weight updating algorithm is first expressed as a statement of a fuzzy CMAC learning problem. Given the lth training pair $(\boldsymbol{u}^l, z_d^l)$, where $\boldsymbol{u}^l = [u_1^l, u_2^l, \ldots, u_N^l]^T$ is input vector and z_d^l is desired output, update weights, and knot points associated with fired fuzzy rules, such that the error between z^l and z_d^l is reduced. Notice that for each presentation of input vector, the number of fired weights is 3^N and the number of effected knot points is 4^N.

Then a learning (i.e., weights updating) algorithm for the fuzzy CMAC is employed. The fuzzy CMAC mapping can be expressed in terms of the product of firing strength and weights

$$z = \sum_{p=1}^{M} \boldsymbol{s}_p \boldsymbol{w}_p = \boldsymbol{S}^T \boldsymbol{W} \tag{4}$$

The learning algorithm is targeted to reduce error between desired output z_d and actual output of fuzzy CMAC by adjusting the values of weights:

$$\boldsymbol{E} = \tfrac{1}{2}(z_d - z)^2 = \tfrac{1}{2}\boldsymbol{e}^2 \tag{5}$$

The adjustment of the weights is based on the following rules:

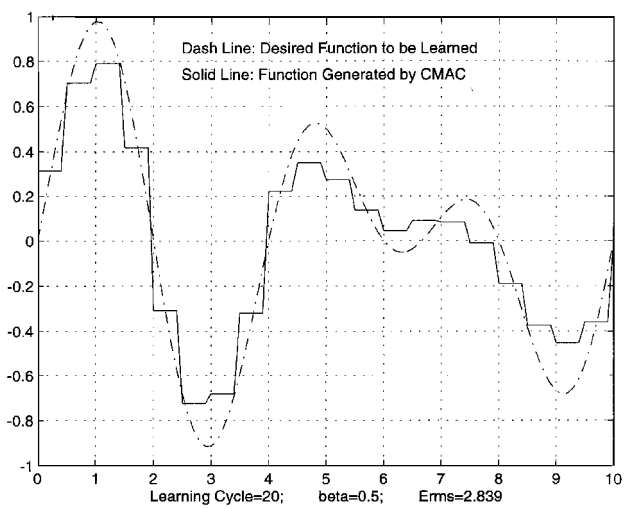
$$\Delta \boldsymbol{w}_p = \beta \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}_p} = \beta \boldsymbol{e} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{w}_p} = -\beta \boldsymbol{e} \frac{\partial z}{\partial \boldsymbol{w}_p} = -\beta \boldsymbol{e}\, \boldsymbol{s}_p, \qquad p = 1, 2, \ldots, M \tag{6}$$

where β is a learning rate. These learning rules are extremely simple and efficient. They differ from the original CMAC proposed by Albus[4] in that the adjustment of a weight is proportional to its contribution (measured by its firing strength) to the output signal, based on which the error is generated. The original CMAC distributes the output error evenly among all of the contributing weights.

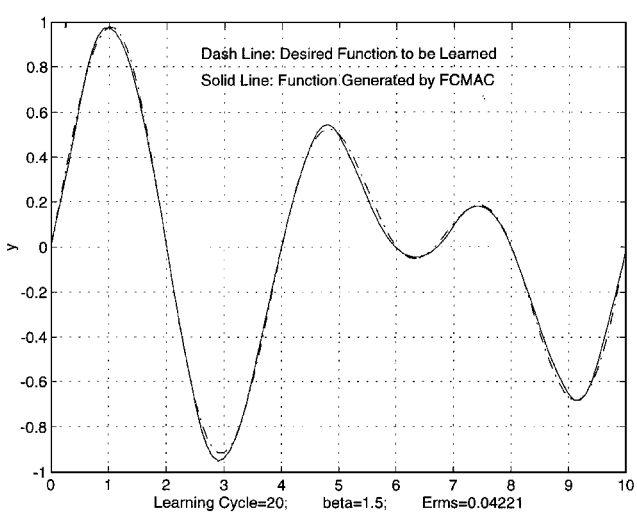
D. Simulation Results Comparing Fuzzy CMAC with Original CMAC

Comparison was also made between the original CMAC proposed by Albus[4] and the fuzzy CMAC for arbitrary function approximation. Figures 5a and 5b show a comparison of one-dimensional arbitrary function approximation capability of the original CMAC and the fuzzy CMAC. The dashed line in both plots represents an arbitrary function to be learned. Because of the binary nature of the CMAC activating function, the learning result of the CMAC is not smooth and the rms of the residual error of the learning process is fairly large (2.839), as shown in Fig. 5a. Figure 5b presents the learning result using the fuzzy CMAC neural network. With the same definition of the knot points, same number of memory units, and the same number of learning iterations, the fuzzy CMAC achieves a much better learning result. The rms value of the residual error is 0.04221 (which represents only 1.4% of the error obtained using the original CMAC).

Figures 6a–6d show another comparison result but in a two-dimensional case. An arbitrary two-dimensional function, as shown in Fig. 6a, is selected as the desired function to learn. Given the same input space partition and the same learning rate, Figs. 6b and 6c show the learning capabilities of a conventional CMAC and a fuzzy CMAC, respectively. Both simulations used 150 learning cycles. The rms values of the residual errors are 79.65 for CMAC

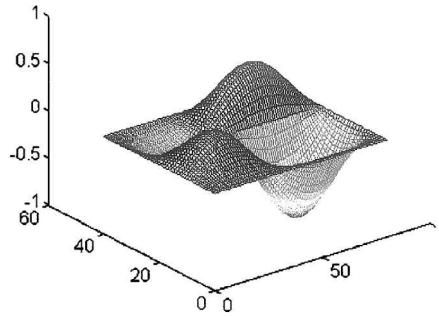


a) CMAC learning one-dimensional function

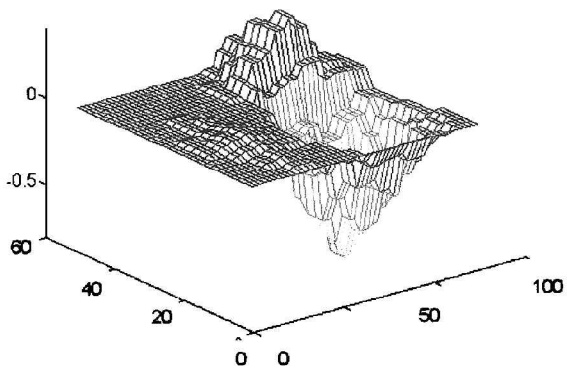


b) Fuzzy CMAC learning function

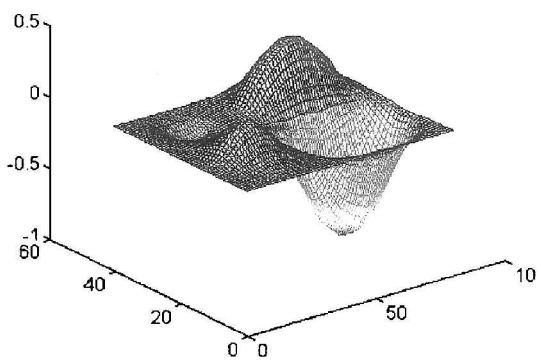
Fig. 5 Learning performance comparison between fuzzy CMAC and CMAC in one-dimensional case: Erms, rms value of residual error; beta, learning rate.



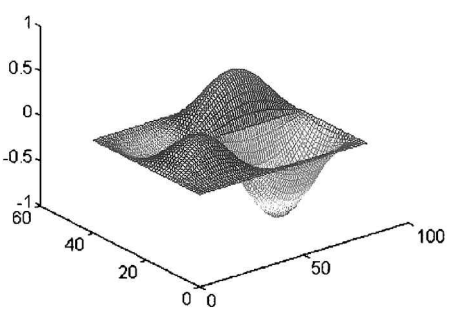
a) Desired two-dimensional function to be learned



b) CMAC learning results: learning cycle = 150, Erms = 79.65



c) Fuzzy CMAC learning results: learning cycle = 150, Erms = 14.83



d) Fuzzy CMAC learning results: learning cycle = 4500, Erms = 0.474

Fig. 6 Learning performance comparison between fuzzy CMAC and CMAC in two-dimensional case: Erms, rms value of residual error.

vs 14.84 for fuzzy CMAC. Fuzzy CMAC is able to obtain a much smoother function. More importantly, the learning accuracy of the original CMAC is restricted by its binary activating function. Additional learning iterations could not reduce the learning error further. On the other hand, the fuzzy CMAC is able to keep improving its learning performance. Figure 6d shows a learning result of the fuzzy CMAC after 4500 learning iterations. The rms value of the residual error has decreased to 0.4739, which is only 0.59% of the error obtained using the original CMAC.

III. Dynamic Models of Missile

For the sake of convenience, we consider a three-dimensional interception problem as two two-dimensional intercepts, one on the projected horizontal plane (X–Y plane) and another on the projected vertical plane (X–Z plane). The simultaneous intercepts on both planes imply the true intercept in the original three-dimensional space. In the following discussion, we will focus on X–Z plane interception and control problem for the HAVE DASH II BTT missile.

The translational motion equation in the vertical plane of a missile can be expressed as

$$\dot{V}_m = (1/m)\{\cos(\theta-\gamma)(F_x - mg\sin\theta + T_x) + \sin(\theta-\gamma)(F_z + mg\cos\theta)\} \tag{7}$$

$$\dot{\gamma} = -(1/mV_m)\{-\sin(\theta-\gamma)(F_x - mg\sin\theta + T_x) + \cos(\theta-\gamma)(F_z + mg\cos\theta)\} \tag{8}$$

Because roll and yaw angles are zero, the missile angular rates in the body frame become

$$\dot{Q} = M/I_{yy} \tag{9}$$

$$\dot{\theta} = Q \tag{10}$$

The forces and moments are

$$F_x = k_F\rho V_m^2 C_x \qquad F_z = k_F\rho V_m^2 C_z \qquad M = k_M\rho V_m^2 C_m \tag{11}$$

where ρ is atmospheric density, k_F is a constant determined by vehicle geometry, and C_x and C_z are aerodynamic coefficients.

The analytic approximates of the aerodynamic coefficients for HAVE BTT can be described as

$$C_x = -0.57 + 0.0083\alpha + 0.004\delta_e$$
$$C_z = C_{z0}(\alpha, M_m) - 0.09\delta_e$$
$$C_{z0}(\alpha) = C_{z1}(\alpha) + C_{z2}(\alpha)M_m$$
$$C_{z1}(\alpha) = -0.0015\alpha^3 + 0.0125\alpha^2 - 0.5052\alpha + 0.0429$$
$$C_{z2}(\alpha) = 0.0006\alpha^3 - 0.0138\alpha^2 + 0.1230\alpha - 0.0191$$
$$C_{m0}(\alpha) = C_{m1}(\alpha) + C_{m2}(\alpha)M_m$$
$$C_{m1}(\alpha) = -0.0055\alpha^3 + 0.2131\alpha^2 - 2.7419\alpha + 0.0381$$
$$C_{m2}(\alpha) = 0.0014\alpha^3 - 0.0623\alpha^2 + 0.8715\alpha - 0.4041$$

The position in an inertial frame is given by

$$\dot{x} = V\cos\theta - W\sin\theta \tag{12}$$

$$\dot{z} = V\sin\theta + W\cos\theta \tag{13}$$

Let

$$\boldsymbol{X} = \begin{bmatrix} V_m \\ \gamma \\ Q \\ \theta \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} \delta_e \\ T_x \end{bmatrix}, \qquad \boldsymbol{y} = \gamma \tag{14}$$

Then the dynamic model can be compactly expressed in a general form as follows:

$$\dot{\boldsymbol{X}} = f(\boldsymbol{X}, \boldsymbol{u}) \tag{15}$$

$$\boldsymbol{y} = h(\boldsymbol{X}) \tag{16}$$

IV. Guidance Law for Missile Target Interception

The interception between an autonomously guided missile and a highly maneuverable flying target is a nonlinear dynamic process with only partially known missile aerodynamic parameters and target state dynamics. In a vertical plane, the missile–target interception can be characterized by two variables, namely, the missile–target range R, and the line-of-sight (LOS) angle σ, both measured relative to any fixed reference frame.

The proportional navigation (PN) scheme is the navigation law used in our design: the rate of the missile flight-path angle $\dot{\gamma}$ is controlled to be proportional to the rate of the LOS angle, i.e.,

$$\dot{\gamma} = K_g \dot{\sigma} \tag{17}$$

where K_g is a positive parameter of gain. Under this guidance law, the missile detects any rotation of the LOS angle and applies a turning rate proportional to this rotation in a direction such as to reduce the LOS rate and thus tend to get back onto an intercepting course.

V. Inner-Loop Autopilot Design

The nonlinear dynamics of a missile can be described by Eqs. (15) and (16), where the missile state variable X represents the velocity, flight-path angle, pitching rate, and pitch angle; $\boldsymbol{u}$ is the control surface setting, which includes throttle position and aileron angle. The function F, which represents the nonlinear dynamic relationship of the missile, is contaminated with aerodynamic, aircraft structural, and other uncertainties.

A combination of a fuzzy CMAC neural network and a robust linear feedback flight control system is developed. The control objective of the autopilot system is to have the missile track the trajectory commands provided by the navigation guidance law. In the case of X–Z planar motion, the desired trajectory is defined by the flight-path angle γ_d. A good gamma controller is indispensable to ensure the interception performance. In its inner control loop, a linear feedback controller is designed based on a linearized model. Though it cannot cope with severe nonlinearities in the full flight envelope, the controller does provide an easy to deal with and stabilized closed-loop system for the outer-loop controller. In the outer loop, a fuzzy CMAC-based nonlinear adaptive controller is structured to account for nonlinearity, unmodeled dynamics, and uncertainties.

To simplify the problem of linear gamma controller design, we assume that gravity effects on the dynamic model can be mostly compensated by an appropriate control signal of δ_e and T_x. In fact, δ_e and T_x can be obtained by solving the simultaneous equations

$$F_x - \text{mg} \sin\theta + T_x = 0 \tag{18}$$

$$F_z + \text{mg} \cos\theta = 0 \tag{19}$$

Therefore, we ignore the gravity effect during the gamma controller design. Equations (18) and (19) also suggest that the speed change of the missile is very slow so that the $\dot{V}_m$ equation needs not be considered during the design. We then have a third-order missile system dynamics,

$$\dot{\gamma} = -\frac{1}{mV_m}\Big\{-\sin(\theta-\gamma)\big[k_F \rho V_m^2 C_x(0, \gamma, V_m, \delta_e) + T_x\big] + \cos(\theta-\gamma) k_F \rho V_m^2 C_z(\theta, \gamma, V_m, \delta_e)\Big\} \tag{20}$$

$$\dot{q} = \frac{k_M \rho V_m^2 [C_{m0}(\theta, \gamma, V_m) + C_{me}(\delta_e)]}{I_{yy}} \tag{21}$$

$$\dot{\theta} = q \tag{22}$$

where the areodynamic coefficient C_m is partitioned into C_{m0} and $C_m e$: $C_m = C_{m0} + C_{me}\,\delta_e$. To apply linear control system design methods, the preceding nonlinear model is linearized at an operating point. Define

$$f_1(\theta, \gamma, V_m, \delta_e, T_x) = -\frac{1}{mV_m}\Big\{-\sin(\theta-\gamma) \times \big[k_F \rho V_m^2 C_x(\theta, \gamma, V_m, \delta_e) + T_x\big] + \cos(\theta-\gamma) k_F \rho V_m^2 C_z(\theta, \gamma, V_m, \delta_e)\Big\} \tag{23}$$

$$f_2(\theta, \gamma, V_m, \delta_e) = \frac{k_M \rho V_m^2 [C_{m0}(\theta, \gamma, V_m) + C_{me}(\delta_e)]}{I_{yy}}$$

$$f_3(\theta) = \theta$$

The equilibrium manifold can be found according to the following conditions:

$$\gamma = \gamma_d \tag{24a}$$

which is desired system output,

$$\theta = \theta_d \tag{24b}$$

which is selected under constraint $\alpha_d = \theta_d - \gamma_d$, $0 < \alpha_d < 25\,\text{deg}$,

$$q = q_d = 0 \tag{24c}$$

$$\delta_{ed} = -\frac{C_{m0}(\theta_d, \gamma_d, V_m)}{C_{me}} \tag{24d}$$

$$T_x = T_{xd} = \text{ctg}(\theta_d - \gamma_d) C_z(\theta_d, \gamma_d, V_m, \delta_{ed}) - C_x(\theta_d, \gamma_d, V_m, \delta_{ed}) \tag{24e}$$

where δ_{ed} is an offset deflection of δ_e at the equilibrium operating point. The linearized system is denoted as

$$\frac{\text{d}}{\text{d}t}\begin{bmatrix}\gamma\\ q\\ \theta\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}\begin{bmatrix}\gamma\\ q\\ \theta\end{bmatrix} + \begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\\ b_{31} & b_{32}\end{bmatrix}\begin{bmatrix}\delta_e\\ T_x\end{bmatrix} \tag{25}$$

The linear fixed gain control command is formed as

$$\boldsymbol{u} = \begin{bmatrix}\delta_{ed}\\ T_{xd}\end{bmatrix} + K[x - x_d] \tag{26}$$

where the state feedback gain matrix $K \in R^{2\times 3}$ is calculated through eigenstructure assignment algorithms[11] with the desired eigenvalues $\{-5 \pm 3j, -10\}$. Selection of these particular values of desired eigenvalues is based on results of extensive simulations.

This linear feedback control constitutes the inner loop of the proposed flight control scheme. This stabilization is essential because otherwise the fuzzy CMAC-based adaptive control may not generate sufficiently rapid correction to prevent the missile dynamics from divergence.

VI. Fuzzy CMAC Learning Controller: Outer Loop

The linear feedback control discussed in the preceding section is a point design. As the flight condition changes, the linear feedback control action has to be changed accordingly. This function is traditionally accounted for by gain scheduling. As the flight envelope expands, the gain scheduling and robust design may not be sufficient to handle system variations.[12–16] This is particularly true for a high-angle-of-attack flight where asymmetric vortex shedding from the leading-edge extension makes the dynamic model extremely uncertain. Compensation for this type of uncertainty can be addressed by the fuzzy CMAC neural network.

Figure 7 shows the block diagram of the fuzzy CMAC-based fuzzy-neural control system for missile–target interception control. In the outer loop, the fuzzy CMAC networks are used to achieve overall robust performance by using a fuzzy knowledge base and CMAC learning capability. The CMAC is operated around the closed-inner-loop dynamic model. For a designated maneuver, the CMAC will update the weights to seek an optimal performance. Because the system has been properly stabilized by the inner-loop controller discussed earlier, the fuzzy CMAC algorithm can be executed efficiently.

The adaptive learning mechanism embedded in the overall system shown in Fig. 7 operates as follows. When the system is first activated, all of the weights in the fuzzy CMAC are set to zero, or the initial weight can be selected according to the fuzzy knowledge base. At each control cycle, the inner-loop feedback control signal u_a is synthesized to stabilize the system. The error between the desired output and actual output is used to trigger the fuzzy CMAC learning algorithm, which adjusts the weights in the fuzzy CMAC networks. Gradually, successive learning will enable the fuzzy CMAC to establish a mapping reflecting the system dynamics. The signal u_b generated from the fuzzy CMAC controller is then used to minimize the tracking error. As more training cycles are executed, the resulting fuzzy CMAC controller may become the primary stability and control augmentation device because functions of feedback control law are also learned by fuzzy CMAC.

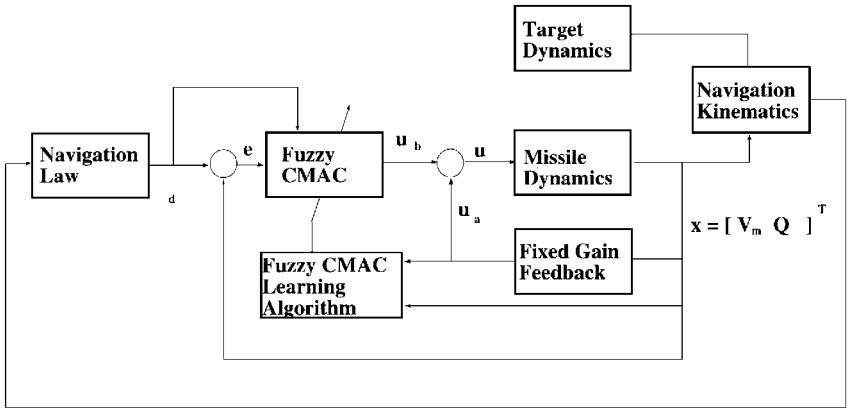


Fig. 7 Fuzzy CMAC-based integrated missile guidance and control system.

Using the fuzzy CMAC in the structure just described yields the following advantages.

1) The two-level control system design hierarchy allows the controller at each level to achieve stabilization and performance enhancement independently. The inner feedback control level guarantees the stability of the flight control system, and the fuzzy CMAC adaptive learning controller enhances the performance and reliability by compensating for the unknown dynamics and uncertainties during system operation. More importantly, the missile dynamics is stabilized before the fuzzy CMAC learning process begins, which makes the neural network easier to adapt and results in high performance. When these two-level control functions are combined and operated on-line, the resulting neural network-based flight control system achieves the maximum synergism.

2) The CMAC-based adaptive learning control structure is well suited for real-time application because the training process is based on real-time input and output data, and the learning algorithm is executed on-line.

3) Because the learning controller requires only qualitative information about the system being controlled, the same fuzzy CMAC control design applies to a large class of controlled systems, and it can be easily adjusted to system changes.

4) Another advantage of the fuzzy CMAC neural network is that it is relatively straightforward to implement in high-speed, highly parallel hardware. In fact, we have implemented the fuzzy CMAC on a custom-designed multiple DSP hardware for a real-time active vibration control application.[10] We can easily achieve a sampling frequency of 2000 Hz.

A. Simulation Results

Computer simulations were conducted to validate the control accuracy and efficiency of the proposed guidance law and control scheme. The nonlinear HAVE DASH II bank-to-turn missile model discussed in Sec. III is used. The missile parameters are selected as $G = 32.174$ ft/s^2, $m = 9.89$ slug, $c = 968$ ft/s, $k_F = 0.1534$ ft^2, $k_M = 0.0959$ ft^3, $\rho = 5.124^* 10^{-4}$ slugs/ft^3, $I_{xx} = 1.1913$ slug ft^2, $I_{yy} = 100.5139$ slug ft^2, $I_{zz} = 100.5749$ slug ft^2, and $T_x = 389$ lb, where c and ρ are computed at the altitude of 40,000 ft and 1 slug $= 32.2$ lb mass. Two sets of simulation results are presented here, tracking performance of the gamma (γ) controller and missile–target interception.

B. Simulation of Tracking Performance of the Gamma Controller

Figure 8 shows a comparison of a simulated missile tracking performance controlled by a fuzzy CMAC self-learning controller and a proportion-integral-derivative (PID) linear feedback controller that can be implemented using analog electronic circuits. A desired tracking trajectory for missile flight-path angle γ is specified in a fairly large range, as shown in the line marked "Ideal signal." Control performance of the fixed gain inner-loop controller is shown by the line marked "fixed gain controller" in Fig. 8, and the control performance of adding the fuzzy CMAC learning controller is shown by the line marked "fuzzy CMAC." The angle is in radians. Equilibrium state was set at $Vm = 2.8$ Mach, $\gamma = 0$, $Q = 0$, and

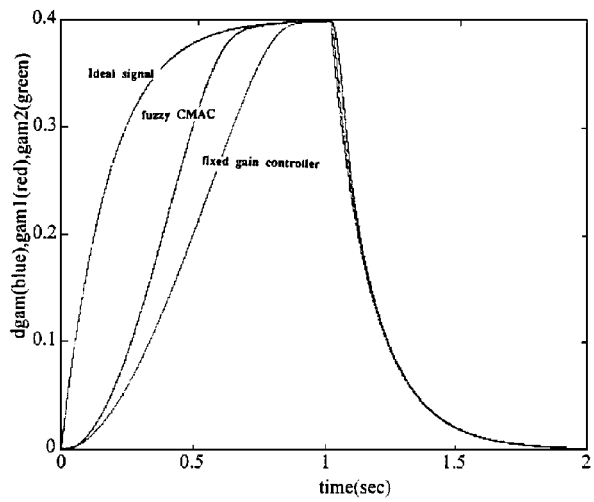
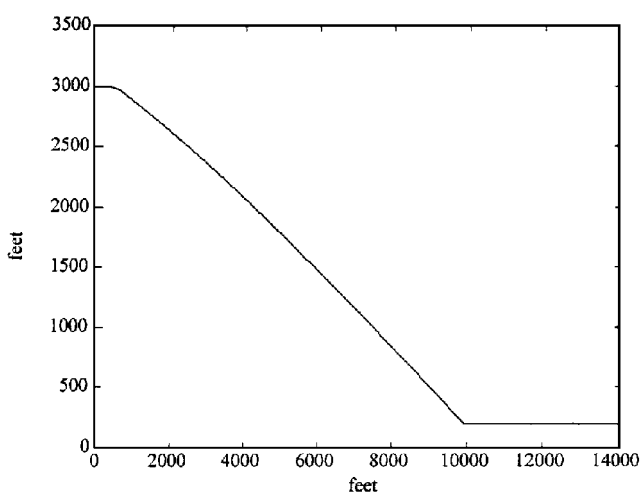


Fig. 8 Comparison of a simulated missile tracking performance controlled by a fuzzy CMAC self-learning controller and a plan linear feedback controller.



Missile interception using FCMAC controller

Fig. 9 Engagement scenario from which we see that the miss distance reduces gradually and eventually hits the target, as the missile was controlled by the guidance low and the fuzzy CMAC.

$\theta = 0$. The digital control cycle was chosen as 0.001, whereas the simulation model update cycle was 0.0001 s.

C. Simulation of Missile–Target Interception

A head-on missile–target interception scenario was also simulated where a fuzzy CMAC-based flight control and guidance system was employed. The missile velocity is assumed to be 1.5 times that of the target. Figure 9 shows the engagement scenario from which we see that the miss distance gradually reduces and eventually hits

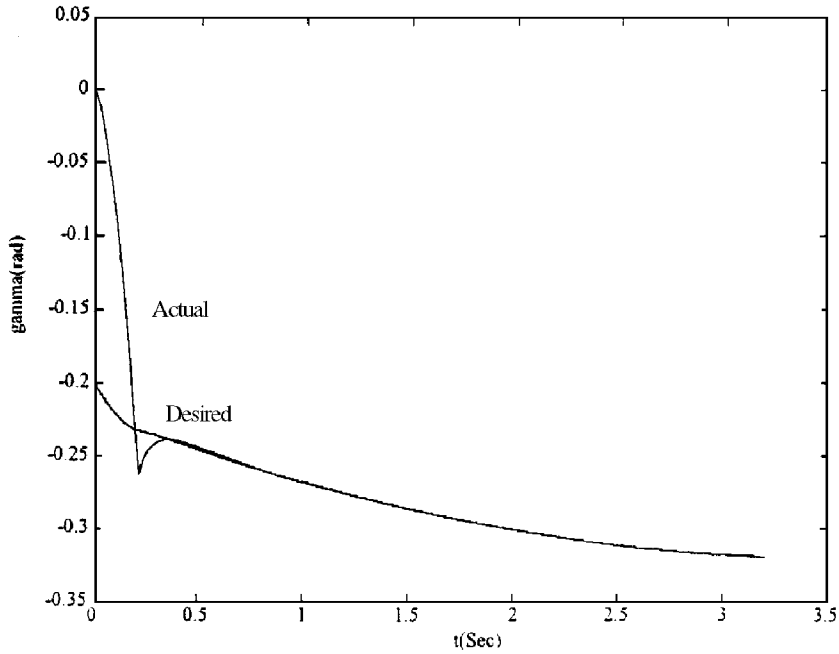


Fig. 10 Tracking performance of the flight-path angle in the interception process: Desired, desired gamma command generated by the guidance law, and Actual, actual gamma angle trajectory.

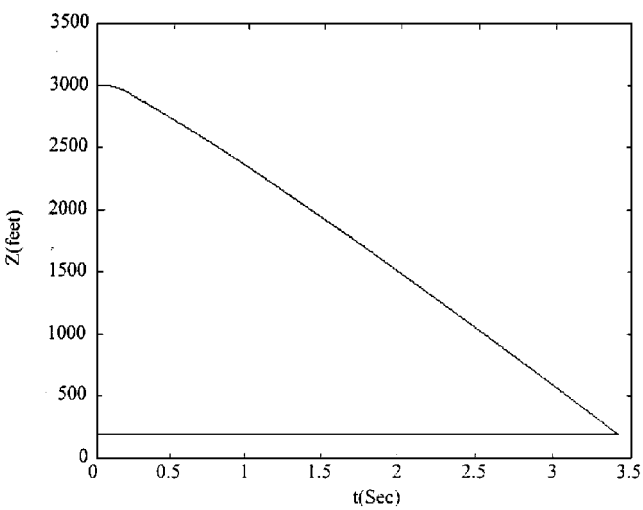


Fig. 11 Time history data of the missile and the target trajectories in *X* and *Z* coordinates.

the target, as the missile was controlled by the guidance law and the fuzzy CMAC. Figure 9 shows the tracking performance of the flight-path angle in the interception process. The line marked "Desired" is the desired gamma command generated by the guidance law, and the line marked "Actual" represents the actual gamma angle trajectory. It can be seen that the missile follows the desired guidance command trajectory very closely after a 0.5-s transition period. Figures 10 and 11 illustrate the time history data of the missile and the target trajectories in X and Z coordinates.

D. Analysis of Simulation Results

Because the linear feedback controller is designed based on the equilibrium point parameters of the missile dynamics, it performed poorly when the gamma angle got larger. Under large variation of flight conditions and missile parameters, the missile system controlled solely by the feedback loop exhibits oscillative behavior, as shown in Fig. 8. A self-learning fuzzy CMAC controller was imposed on the inner-loop controller to enhance the performance of the missile system. The nonlinear relationship among $\mathrm{d}X/\mathrm{d}t$, x, and u can be realized using a fuzzy knowledge base initially and improved through learning of the fuzzy CMAC. Consequently, the inverse mapping from $\mathrm{d}X/\mathrm{d}t$ and X to u can be established such that for a specified maneuver profile, the corresponding control can be exercised. The simulation results show a substantial improvement. Not only does the missile trajectory track the desired one more closely, but also consistency of performance was achieved.

We believe that our two-layer control scheme works primarily because of the synergism of inner- and outer-loop controller design. A well-designed inner loop offers a stabilized and well behaved system in terms of equilibrium point design. This also provides a wider margin for the outer-loop learning controller to trial and error different control parameters without risking the violation of stability and causing oscillative system behavior. These simulation results demonstrate the powerful learning capability of the fuzzy CMAC neural network. The complex inverse dynamics of the BTT missile can be learned in real-time fashion by a fuzzy CMAC. The universal approximation theorem we proved provides a theoretical foundation and justification for us to use a fuzzy CMAC to approximate various type of complex nonlinearity. The learned fuzzy CMAC outer-loop controller superimposes fine turning of the control signal to the missile system so that an optimal output (trajectory) is generated, based on learned inverse dynamic mapping of the system.

Fuzzy CMAC learns faster than the original CMAC neural network. It learns orders of magnitude faster than popular multilayer perceptron back-propagation neural networks. The fast learning capability is another reason why the proposed fuzzy neural network control system is well suited to missile control systems, due to the fast dynamic response of missile systems.

VII. Conclusion

A novel fuzzy CMAC neural network architecture is presented. By synergetically combining the preferred features of the FLC and the CMAC neural network, the fuzzy CMAC shows excellent approximation and self-learning capabilities. The fuzzy CMAC neural network is applied to an integrated guidance and control system design for a BTT missile. A two-layer control system structure is employed. In the inner loop, a feedback control law based on a linearized missile model is applied to stabilize the missile nonlinear dynamics. The outer loop consists of a fuzzy CMAC neural network performing on-line self-learning of the missile dynamics and superimposes a fine turning control signal to enhance the tracking ability. The limitation of the fuzzy CMAC control design method, however, is that it does not ensure closed-loop stability. This stability concern is addressed by applying the inner-loop linear robust control

techniques to the missile dynamics. Both the flight-path angle tracking and missile target interception scenarios were simulated, and the performance was satisfactory.

Acknowledgment

This work is supported by U.S. Army Space and Strategic Defense Command under Contract DASG60-94-C-0050.

References

[1]Sugeno, M., "An Introductory Survey of Fuzzy Control," *Information Science*, Vol. 36, No. 1, 1985, pp. 59–83.

[2]Zadeh, L., "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges," *Journal of Cybernetics*, Vol. 2, No. 2, 1972, pp. 4–34.

[3]Anon. (ed.), *Proceedings of Artificial Neural Networks in Engineering (ANNIE)*, Univ. of Missouri–Rolla, MO, 1995.

[4]Albus, J. A., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller," *Journal of Dynamic Systems, Measurement and Control*, Vol. 63, No. 3, 1975, pp. 220–227.

[5]Miller, W. T., "Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 2, 1987, pp. 329–332.

[6]Narendra, K. S., and Annaswamy, A. M., *Stable Adaptive Systems*, Prentice–Hall, New York, 1989.

[7]Narendra, K. S., and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, 1990, pp. 4–27.

[8]Rumelhart, D., Hinton, G. E., and Williams, R. J., "Learning Internal Representations By Error Propagation," *Parallel Distributed Processing*, Vol.1, MIT Press, Cambridge, MA, 1986, pp. 683–691.

[9]Widrow, B., and Stearns, S. D., *Adaptive Signal Processing*, Prentice–Hall, New York, 1985.

[10]Geng, Z. J., "Machine Tool Active Chatter Control Using Fuzzy CMAC Neural Networks," *2nd S.M. Wu Symposium on Manufacturing Science* (Ann Arbor, MI), American Society of Mechanical Engineers, New York, 1996, p. 134.

[11]Geng, Z., "Eigenstructure Assignment and Its Applications to Design of Flight Control Systems," *Proceedings of the 26th Conference on Decision and Control* (Los Angeles, CA), Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1987, pp. 2037–2041.

[12]Lin, C.-F., *Advanced Control System Design*, Wiley, New York, 1994.

[13]Cloutier, J. R., Evers, J. H., and Feeley, J. J., "Assessment of Air-to-Air Missile Guidance and Control Technology," *IEEE Control System Magazine*, Vol. 97, No. 1, 1987, pp. 27–29.

[14]Irwin, G. W., "Design of Controller for Bank-to-Turn CLOS Guidance Using Optimal Control," *Proceedings of 1986 American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1986, pp. 1143–1148.

[15]Lian, K. Y., Wamg, L. S., and Fu, L. C., "Robust Output Tracking for Nonlinear System with Weakly Non-Minimum Phase," *International Journal of Control*, Vol. 58, No. 2, 1993, pp. 301–316.

[16]Schumacher, D. A., and McClamroch, N. H., "Planar Autopilot Design for the EMRAAT Missile Using Sliding Modes," *Proceedings of the American Control Conference* (Baltimore, MD), Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1994.