

## Integral Variable Structure Control of Nonlinear System Using CMAC-based Learning Approach

Wei-Song Lin<sup>1</sup> and Chin-Pao Hung<sup>1,2</sup>

<sup>1</sup>Institute of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan, R.O.C

<sup>2</sup>Department of Electrical Engineering  
National Chin-Yi Institute of Technology  
Taichung, Taiwan, R. O. C.

E-mail: cbhong@chinyi.ncit.edu.tw

### Abstract

A CMAC-based controller with a compensating neural network and an update rule is proposed to design the integral variable structure control (IVSC) of nonlinear system. The control scheme comprises a stabilizer controller and a CMAC neural network. Based on the Lyapunov theorem, the stabilizer controller guarantees the global stability of the system. The CMAC neural network performs the equivalent control by a real-time learning algorithm. The proposed control scheme is globally stable in the sense that all signals involved are bounded. The new IVSC control scheme reduced the dependency to system parameters. Simulation results of numerical example demonstrate the effectiveness and robustness of the proposed controller.

Keywords: integral variable structure control, CMAC, nonlinear system, neural network control, learning control, sliding mode, VSC

### 1. Introduction

Dr. Chern first proposed the integral variable structure control (IVSC) to solve the steady state error problem, and to improve the robustness of traditional variable structure control (VSC) in 1991[1]. Then Chern applied this control scheme to robot manipulator [2], brushless DC servo system [3-4][8], DC motor [5], and induction motor [6][9][11], to demonstrate the feasibility of IVSC. The related researchers also proposed a new design method for IVSC [7]. Other applications are such as the engine system [10], the UPS system [13], and the voltage regulator system [12]. Since the IVSC introduced the integral state variable into the controller design, it can solve the steady state error problem of conventional VSC because of the dead zone or the boundary layer [14-15]. However, regardless of either the traditional VSC or the IVSC scheme, the controller design is a parametric scheme that more dependent on the system model or parameters.

In the past year, the authors developed a real time learning scheme to solve the VSC design problem of unknown parameters dc servo system [16]. In [16], we introduced a CMAC [17] neural network into the VSC. The CMAC, in a table look-up fashion, produced a vector output in response to a state vector input. It is like the models of human memory, using local data to perform reflexive processing. Fig. 1 shows a schematic diagram of CMAC network, through a series of mapping every input state  $x$  map to produce an output  $y$  [18]. The mapping processes

satisfy the similar inputs excite similar memory addresses, i.e. if the input states are close in input space will have their corresponding sets of association cells overlap. For example, if  $x_1$  and  $x_2$  are similar (close),  $x_1$  excites the memory addresses  $a_1, a_2, a_3, a_4$ ,  $x_2$  should excite the memory addresses  $a_2, a_3, a_4, a_5$  or  $a_3, a_4, a_5, a_6$ . If two inputs fire up the same memory addresses, we say the similarity of the two inputs is high. Low similarity would active fewer same memory addresses. If the reference input state is a smooth function, every adjacent state can be thought of as similar inputs (with proper quantization level width). Assuming similar input command should have similar control force, the characteristic of local reflexive action makes CMAC attractive to on line application in control system.

In this paper we expand the control scheme to the IVSC of nonlinear system and improve the learning law to demonstrate why the CMAC can work well. In this new IVSC structure, the controller design only needs little model information. Based on the Lyapunov theorem, a stabilizer controller guarantees the system stable running. The e CMAC neural network with learning algorithm approximates and performs the equivalent control depending on the uncertainties and the constraints of the desired integral sliding surface. This is unlike the conventional parametric methods in that equivalent control is obtained from the nominal model assume free uncertainty. With application of this scheme to a nonlinear system, computer simulations demonstrate the success of the proposed method.

### 2. Problem formulation

Consider the n-order nonlinear systems of the form [19]

$$\dot{x}^{(n)} = f(x, \dot{x}, \dots, x^{(n-1)}) + bu, y = x \quad (1)$$

where  $f$  is an unknown continuous function,  $b$  is a positive unknown constant, and  $u \in R$  and  $y \in R$  are the input and output of the system, respectively. We assume that the state vector  $\underline{x} = (x_1, x_2, \dots, x_n)^T = (x, \dot{x}, \dots, x^{(n-1)})^T \in R^n$  is available for measurement. Therefore, the control objective of this paper can be described as follows:

Determine a feedback control  $u(\underline{x}|\underline{w})$  and a learning law for adjusting the vector  $\underline{w} = (w_1, w_2, \dots, w_g)^T \in R^{g \times 1}$ ,  $g$  is the selected memory size, such that the following conditions are met:

1) The closed loop system must be globally stable in the sense that all variables,  $\underline{x}(t), \underline{w}(t)$  and  $u(\underline{x}|\underline{w})$ , must be uniformly bounded; i.e.,  $|\underline{x}(t)| \leq M_x < \infty$ ,  $|\underline{w}(t)| \leq M_w < \infty$  and  $|u(\underline{x}|\underline{w})| \leq M_u < \infty$ ,  $M_x, M_w, M_u$  are design parameters specified

by the designer.

2) The tracking error  $e \equiv y_d - y$ , should be as small as possible under constraints in 1) and  $\underline{y}_d = (y_d, \dot{y}_d, \dots, y_d^{(n-1)})^T$ ,  $\underline{e} = (e, \dot{e}, \dots, e^{(n-1)})^T = (e_1, e_2, \dots, e_n)^T$  and  $\underline{k} = (k_n, \dots, k_1)^T \in R^n$  be such that all roots of the polynomial  $h(s) = s^n + k_1 s^{n-1} + \dots + k_n$  are in the open left-half plane. Here,  $k_i, i=1, \dots, n$  are designed to satisfy the IVSC sliding surface function defined as following [1]

$$K_I Z + \sum_{i=1}^n c_i e_i = 0 \quad (2)$$

and

$$\dot{Z} = y_d - x_1 = e \quad (3)$$

$$c_n = 1, c_{n-i} = k_i, i=1, \dots, n-1, k_n = K_I \quad (4)$$

### 3. The CMAC-based integral variable structure control (CIVSC)

#### 3.1 The structure of proposed controller

As described above, if the exact system model is known

$$K_I \dot{Z} + \sum_{i=1}^n c_i \dot{e}_i = 0 \quad (5)$$

$$K_I (y_d - x_1) + \sum_{i=1}^{n-1} c_i e_{i+1} + (y_d^{(n)} - x^{(n)}) = 0 \quad (6)$$

$$K_I e_1 + \sum_{i=1}^{n-1} c_i e_{i+1} + y_d^{(n)} - f(\underline{x}) - bu = 0 \quad (7)$$

Then the optimal equivalent control  $u^*$  is

$$u^* = \frac{1}{b} \left\{ -f(\underline{x}) + y_d^{(n)} + K_I e_1 + \sum_{i=1}^{n-1} c_i e_{i+1} \right\} \\ = \frac{1}{b} - f(\underline{x}) - y_d^{(n)} - \underline{k}^T \underline{e} \quad (8)$$

where  $\underline{k} = (k_n, \dots, k_1)^T = (K_I, c_1, \dots, c_{n-1})^T$ . Applied (8) to (1) results in

$$\dot{x}^{(n)} = f(\underline{x}) - f(\underline{x}) - y_d^{(n)} - \underline{k}^T \underline{e}$$

i.e.

$$e^{(n)} + k_1 e^{(n-1)} + \dots + k_n e = 0 \quad (9)$$

which implies that  $\lim_{t \rightarrow \infty} e(t) = 0$ , the main objective is

achieved. Since  $f$  and  $b$  are not exactly known, the optimal equivalent control cannot be implemented. Our purpose is to design an IVSC controller using CMAC-based learning approach to approximate the optimal equivalent control  $u^*$ .

Suppose the control  $u(\underline{x} | \underline{w})$  is the summation of a CMAC control  $u_N(\underline{x} | \underline{w})$  and a stabilizer control  $u_s(\underline{x})$ :

$$u(\underline{x} | \underline{w}) = u_N(\underline{x} | \underline{w}) + u_s(\underline{x}) \quad (10)$$

Substituting (10) into (1), we have

$$\dot{x}^{(n)} = f(\underline{x}) - b[u_N(\underline{x} | \underline{w}) + u_s(\underline{x})] \quad (11)$$

Now adding and subtracting  $b(t)u^*$  to (11) and after some

straightforward manipulation we obtain the following error equation governing the closed-loop system:

$$\dot{e}^{(n)} = -\underline{k}^T \underline{e} - b[u^* - u_N(\underline{x} | \underline{w}) - u_s(\underline{x})] \quad (12)$$

or, equivalently,

$$\dot{\underline{e}} = \Lambda_c \underline{e} + \underline{b}_c [u^* - u_N(\underline{x} | \underline{w}) - u_s(\underline{x})] \quad (13)$$

where

$$\Lambda_c = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -k_n & -k_{n-1} & \dots & -k_1 & 0 \end{bmatrix}, \underline{b}_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b \end{bmatrix} \quad (14)$$

Define  $V_e = 0.5 \underline{e}^T P \underline{e}$ , where  $P$  is a symmetric positive definite matrix satisfy the Lyapunov equation

$$\Lambda_c^T P + P \Lambda_c = -Q \quad (15)$$

where  $Q > 0$  is specified by designer.

Using (15) and the error equation (13), we have

$$\dot{V}_e = 0.5 \underline{e}^T P \dot{\underline{e}} + 0.5 \underline{e}^T P \dot{\underline{e}} \quad (16)$$

$$= 0.5 \Lambda_c \underline{e} + \underline{b}_c [u^* - u_N(\underline{x} | \underline{w}) - u_s(\underline{x})]^T P \underline{e} + \frac{1}{2} \underline{e}^T P \Lambda_c \underline{e} + \underline{b}_c [u^* - u_N(\underline{x} | \underline{w}) - u_s(\underline{x})]^T P \underline{e}$$

$$- 0.5 \underline{e}^T Q \underline{e} + \underline{e}^T P \underline{b}_c [u^* - u_N(\underline{x} | \underline{w}) - u_s(\underline{x})] \\ \leq 0.5 \underline{e}^T Q \underline{e} + \left| \underline{e}^T P \underline{b}_c \right| \left[ |u^*| + |u_N(\underline{x} | \underline{w})| + |u_s(\underline{x})| \right] \quad (17)$$

Our task now is to design  $u_s$  such that  $\dot{V}_e \leq 0$ . In order to do so, we need the following assumption:

Assumption 1: We can determine a function  $f^U(\underline{x})$  and a constant  $b_L$  such that  $|f(\underline{x})| \leq f^U(\underline{x})$  and  $0 < b_L \leq b$ .

Therefore, substituting (8) into (17) we construct the  $u_s$  as follows:

$$u_s(\underline{x}) = I_1^* \operatorname{sgn}(\underline{e}^T P \underline{b}_c) \left\{ |u_N| + \frac{1}{b_L} (f^U + |y_d^{(n)}| + |\underline{k}^T \underline{e}|) \right\} \quad (18)$$

where  $I_1^* = 1$ , if  $V_e \geq \bar{V}$  ( $\bar{V}$  is a constant specified by the designer) and  $I_1^* = 0$ , if  $V_e \leq \bar{V}$ .

Considering the  $I_1^* = 1$  case, substituting (18) and (13) into (17) we have

$$\dot{V}_e \leq -0.5 \underline{e}^T Q \underline{e} - \left| \underline{e}^T P \underline{b}_c \right| \left\{ \frac{1}{b} |f| + |y_d^{(n)}| + |\underline{k}^T \underline{e}| \right\} |u_N(\underline{x})| \\ - |u_N(\underline{x})| - \frac{1}{b_L} \left\{ |f^U| + |y_d^{(n)}| + |\underline{k}^T \underline{e}| \right\} - \frac{1}{2} \underline{e}^T Q \underline{e} \leq 0 \quad (19)$$

Therefore, using the supervisory control  $u_s$ , we always have  $V_e \leq \bar{V}$ . Because  $P > 0$ , the boundedness of  $V_e$  implies the boundedness of  $\underline{e}$ , which in turn implies the boundedness of  $\underline{x}$ . From (18)  $u_s$  is nonzero only when the error state is greater than specified value. That is, if the CMAC is well behaved, the  $u_s$  is zero to avoid the control signal being too large.

#### 3.2 Learning process

Next we develop a learning law to adjust the weighting vector  $\underline{w}$ .

Define the optimal weighting vector [19]:

$$\underline{w}^* \equiv \arg \min_{\|\underline{w}\|} M_w [\sup_{\underline{x}} M_x |u_N(\underline{x}|\underline{w}) - u^*|] \quad (20)$$

and the minimum approximation error:

$$\equiv u_N(\underline{x}|\underline{w}^*) - u^* \quad (21)$$

The CMAC output is described as [23]

$$u_N(\underline{x}|\underline{w}) = \underline{\mu}(\underline{x})\underline{w} \quad (22)$$

where  $\underline{\mu}(\underline{x}) \in R^{1 \times s}$  is the mapping function of CMAC network. Note that the  $\underline{\mu}(\underline{x})$  only contains the elements of 0 and 1. The numbers of 1 are equal to the numbers of fired memory cells  $A^*$  and then  $\|\underline{\mu}(\underline{x})\| = \sqrt{A^*}$ .

Then the error equation (13) can be rewritten as

$$\begin{aligned} \dot{\underline{e}} &= \Lambda_c \underline{e} + \underline{b}_c [u_N(\underline{x}|\underline{w}^*) - u_N(\underline{x}|\underline{w})] - \underline{b}_c u_s(\underline{x}) - \underline{b}_c \\ &= \Lambda_c \underline{e} + \underline{b}_c \underline{\mu}(\underline{x}) \underline{w} - \underline{b}_c u_s(\underline{x}) - \underline{b}_c \end{aligned} \quad (23)$$

where  $\underline{w} = \underline{w}^* - \underline{w}$ .

Define the Lyapunov function candidate

$$V = \frac{1}{2} \underline{e}^T P \underline{e} + \frac{\beta A^*}{2\beta} \underline{w}^T \underline{w} \quad (24)$$

where  $\beta$  is a positive constant. Using (21), (24) and (17), we have

$$\dot{V} = -\frac{1}{2} \underline{e}^T Q \underline{e} + \underline{e}^T P \underline{b}_c (\underline{\mu}(\underline{x}) \underline{w} - u_s - \xi) + \frac{\beta A^*}{\beta} \underline{w}^T \underline{w} \quad (25)$$

let  $\underline{p}_n$  be the last column of  $P$ ; then from (14) we have

$$\underline{e}^T P \underline{b}_c = \underline{e}^T \underline{p}_n \underline{b} \quad (26)$$

Substituting (26) into (25), and using the fact  $\underline{\mu}(\underline{x}) \underline{\mu}^T(\underline{x}) = \underline{\mu}^T(\underline{x}) \underline{\mu}^T(\underline{x})$  and (since  $\underline{\mu} \in R^{1 \times s}$ ,  $\underline{\mu}^T \in R^{s \times 1}$ ), we have

$$\dot{V} = \frac{1}{2} \underline{e}^T Q \underline{e} + \frac{\beta}{\beta} \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x}) \underline{w} + \beta \underline{w}^T \underline{w} - \underline{e}^T P \underline{b}_c u_s - \underline{e}^T P \underline{b}_c \xi \quad (27)$$

Since  $\underline{w} = \underline{w}^* - \underline{w}$ , therefore, if we choose the learning law as follows:

$$\dot{\underline{w}} = \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w} \quad (28)$$

then (27) becomes

$$\dot{V} \leq -0.5 \underline{e}^T Q \underline{e} - \underline{e}^T P \underline{b}_c \xi \quad (29)$$

where we use the fact  $\underline{e}^T P \underline{b}_c u_s \geq 0$  (cf.(18)). This is the best we can achieve. Fortunately, assume the memory size is large enough the CMAC can approximate function to arbitrarily accurate [23]. The second term of (29) will tend to zero. A large enough  $Q$  matrix will easily have  $\dot{V} < 0$ .

In order to guarantee the  $\|\underline{w}\| \leq M_w$ , we modify the learning law as

$$\dot{\underline{w}} = \begin{cases} \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w}, & \text{if } \|\underline{w}\| < M_w \text{ or } \|\underline{w}\| = M_w \text{ and } \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x}) \leq 0 \\ \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w}, & \text{if } \|\underline{w}\| = M_w \text{ and } \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x}) > 0 \end{cases} \quad (30)$$

where the operator  $T(\cdot)$  is define as [19]:

$$T\left\{\frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w}\right\} = \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w} \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^* \|\underline{w}\|^2} \quad (31)$$

the following theorem guarantees the properties of the proposed control scheme.

**Theorem 1:** Consider the nonlinear plant (1) with the control (10), where  $u_N(\underline{x}|\underline{w})$  is the CMAC output and  $u_s(\underline{x})$  is given by (18). Let the weighting vector  $\underline{w}$  be updated by the learning law (30) and let the assumption 1 be true. Then, the overall control scheme guarantees the following properties.

$$1. \|\underline{w}\| \leq M_w, \|\underline{x}(t)\| \leq \|\underline{y}_d\| + \left(\frac{2\bar{V}}{\min}\right)^{1/2} \quad (32)$$

and

$$\|\underline{u}(t)\| \leq 2\sqrt{A^*} M_w + \frac{1}{b_L} \left[ f^U + \|\underline{y}_d^{(n)}\| + \|\underline{k}\| \left(\frac{2\bar{V}}{\lambda_{\min}}\right)^{1/2} \right] \quad (33)$$

for all  $t \geq 0$ , where  $\lambda_{\min}$  is the minimum eigenvalue of  $P$ .

$$2. \int_0^t \|\underline{e}(\tau)\|^2 d\tau \leq a + c \int_0^t \|\xi(\tau)\|^2 d\tau, \quad (34)$$

for all  $t \geq 0$ , where  $a$  and  $c$  are constants, and  $\xi$  is defined by (21).

3. If  $\xi$  is squared integrable, i.e.  $\int_0^t \|\xi(\tau)\|^2 d\tau < \infty$ , then

$$\lim_{t \rightarrow \infty} \|\underline{e}(t)\| = 0.$$

Proof:

i) To prove  $\|\underline{w}\| \leq M_w$ , we let  $V_w = 0.5 \underline{w}^T \underline{w}$ . If the first line of (30) is true, we have either  $\|\underline{w}\| \leq M_w$  or

$$V_w = \frac{\beta \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x})}{A^*} \underline{w}^T \underline{w} \leq 0 \text{ when } \|\underline{w}\| = M_w, \text{ it is clear that we}$$

always have  $\|\underline{w}\| \leq M_w$ . If the second line is true, we have  $\|\underline{w}\| = M_w$  and

$$\dot{V}_w = \frac{\beta}{A^*} \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x}) \underline{w}^T \underline{w} - \frac{\beta}{A^*} \underline{e}^T \underline{p}_n \underline{b} \underline{\mu}^T(\underline{x}) \frac{\underline{w}^T \underline{w} \underline{w}^T \underline{w}}{\|\underline{w}\|^2} \underline{w} \leq 0$$

Therefore we always have  $\|\underline{w}\| \leq M_w, \forall t \geq 0$ .

As described above, using the supervisor control  $u_s$ , we always have  $V_e \leq \bar{V}$ . Therefore,

$$\frac{1}{2} \min \|\underline{e}\|^2 \leq \frac{1}{2} \underline{e}^T P \underline{e} \leq \bar{V}, \text{ i.e. } \|\underline{e}\| \leq \left(\frac{2\bar{V}}{\min}\right)^{1/2} \quad (35)$$

Since  $\underline{e} = \underline{y}_d - \underline{x}$ , we have  $\|\underline{x}\| \leq \|\underline{y}_d\| + \|\underline{e}\| \leq \|\underline{y}_d\| + \left(\frac{2\bar{V}}{\min}\right)^{1/2}$ ,

which is (32). Finally, we prove (33). Since  $u_N(\underline{x}|\underline{w}) = \underline{\mu}(\underline{x})\underline{w}$  is only the sum of fired memory cells,

we have  $\|u_N(\underline{x}|\underline{w})\| \leq \|\underline{\mu}(\underline{x})\| \|\underline{w}\| = \sqrt{A^*} \|\underline{w}\| \leq \sqrt{A^*} M_w$ .

Therefore, from (10) and (18), we have (33).

ii) From (27), (30), we have

$$\dot{V} = -\frac{1}{2} \underline{e}^T Q \underline{e} - \underline{e}^T P b_c u_s - \underline{e}^T P b_c \xi - I_1 b \phi^T \frac{e^T p_n w w^T - \tau(x)}{A^* |w|^2} \quad (36)$$

where  $I_i=0(1)$  if the first (second) line of (30) is true. Now we will show that the last term of (36) is nonpositive. If  $I_i=0$ , the conclusion is trivial. If  $I_i=1$ , which means that  $|w| M_w$  and  $e^T p_n w^T - \tau(x) = 0$ . Since  $|w| = M_w |w^*|$ , we have  $- \tau w = (w^* - w)^T w = 0.5 [ |w^*|^2 - |w|^2 - |w - w^*|^2 ] \leq 0$ , therefore the last term of (36) is nonpositive, and we have

$$V \leq \frac{1}{2} \underline{e}^T Q \underline{e} - \underline{e}^T P b_c u_s - \underline{e}^T P b_c \xi \quad (37)$$

As described above,  $\underline{e}^T P b_c u_s \geq 0$ ; therefore, (37) can be further simplified to

$$\begin{aligned} V &\leq \frac{1}{2} \underline{e}^T Q \underline{e} - \underline{e}^T P b_c \xi \\ &\leq -\frac{\lambda_{Q \min} - 1}{2} |\underline{e}|^2 - \frac{1}{2} |\underline{e}|^2 - 2 \underline{e}^T P b_c \xi - |P b_c \xi|^2 - \frac{1}{2} |P b_c \xi|^2 \\ &\leq -\frac{\lambda_{Q \min} - 1}{2} |\underline{e}|^2 - \frac{1}{2} |P b_c \xi|^2 \end{aligned} \quad (38)$$

where  $\lambda_{Q \min}$  is the minimum eigenvalue of  $Q$ . Integrating both side of (38) and selecting that  $\lambda_{Q \min} > 1$  (we can choose such a  $Q$  to satisfy this condition), we have

$$\int_0^t |\underline{e}(s)|^2 ds \leq \frac{2}{\lambda_{Q \min} - 1} |V(0)| + \frac{1}{\lambda_{Q \min} - 1} |P b_c \xi|^2 \int_0^t |\xi(s)|^2 ds \quad (39)$$

Define  $a = \frac{2}{\lambda_{Q \min} - 1} |V(0)| + \sup_{t \geq 0} |V(t)|$  and

$$c = \frac{1}{\lambda_{Q \min} - 1} |P b_c \xi|^2, \text{ then (39) becomes (34).}$$

iii) If  $\underline{e}(t) \in L_2$ , then from (34) we have  $\underline{e} \in L_2$ , since all the variable of the right-hand side of (23) are bounded, we have  $\underline{e} \in L$ , using Barbalat's lemma, we have  $\lim_{t \rightarrow \infty} |\underline{e}(t)| = 0$ . Q.E.D.

**Remark 1:** For real control system, the state  $\underline{x}$  and control force  $u$  are required to be constrained within a certain region. Such as the robot system, the joint positions are constrained within workspace and the control signal should be smaller or equal to the output of the computer interface card. Depending on the practical constrain, the designer specified  $M_x$  and  $M_u$ . Then based on (33) and (34) the sliding surface,  $M_w$ ,  $Q$ , and  $\bar{v}$  can be chosen to satisfy the required performance and let the state  $\underline{x}$  and control force  $u$  be within the constraint sets.

### 3.3 Control algorithm

The structure of the proposed scheme is shown in Fig. 2 and the control algorithm is summarized as follows:

- Step 1. Specify the design parameters  $M_x$ ,  $M_u$  based on practical constraints.
- Step 2. Specify the integral sliding surface function  $\sigma_d$ ,  $M_w$ ,  $Q$  and  $\bar{v}$ , based on (33) and (34). And using Matlab

lyap() and eig() functions to solve the  $P$  matrix,  $\lambda_{\min}$  and  $Q_{\min}$ .

- Step 3. Transform the sliding surface to the desired error states for the next control cycle  $\sigma_d(t) \Rightarrow \underline{e}_d(t)$ .
- Step 4. Send the desired next error states  $\underline{e}_d(t)$  to the CMAC network.
- Step 5. Perform a series of mappings to transform the input values to the CMAC output  $u_N(t)$ .
- Step 6. Calculate the stabilizer control signal  $u_s(t)$ .
- Step 7. The  $u_N(t)$  is assumed to be an optimal equivalent control signal and is added to the output of the  $u_s(t)$  to form the control signal  $u(t)$ .
- Step 8. Send  $u(t)$  to plant to produce the actual output states  $e_o(t)$ .
- Step 9. Calculate the values of the actual error state  $\underline{e}$ .
- Step 10. Update the fired weights using equation (30).
- Step 11. If  $t \geq T$ , stop. ( $T$  is the control cycle.) Otherwise, go to step 3.

Noted that the step 1 to step 3 are off-line processes and others are on line computing. The related parameters of CMAC, quantization levels and  $A^*$ , are also specified off line. They can be adjusted depending on the weighting distribution plot.

### 4. Numerical example

Considering a simplified robot system model described as following [16]:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= 0x_1 - (33 - 5 \sin t)x_2 + (300 - 50 \sin t)u \end{aligned} \quad (40)$$

where  $x_1 = x$  is the end-effector output. Define the error output  $e \equiv y_d - x$ ,  $e_1, \dot{e}_1, e_2, \dot{y}_d$  is the desired end-effector output. Without lose of generality, we let  $y_d = 1$ . We choose the integral sliding surface as following:

$$s = K_I \int e_1 dt + c_1 e_1 + e_2, K_I = 1, c_1 = 10 \quad (41)$$

i.e.  $k = [1 \ 10]$ . In equation (15) the design parameters  $Q$  is given by  $Q = 10I_{2 \times 2}$ . The  $P$  matrix can be obtained easily using the Matlab lyap() function. The associated design parameters of CMAC network are list as follows:

$y_d = 1$ , quantization level 30, memory size 4096, fired memory cells  $A^* = 10, M_x = 5(\text{rad}), M_u = 10(\text{volt}), M_w = 2, \bar{v} = 1, \beta = 0.8, b_L = 250, f^U = 38 \cdot |x_2|, Q_{\min} = 10, \lambda_{\min} = 0.5049$

and  $P = \begin{bmatrix} 51 & -5 \\ -5 & 1 \end{bmatrix}$ . Fig. 3 shows the phase plane output only

the stabilizer controller. Introducing the CMAC network into the system (shown as Fig. 2), the simulation results are shown in Fig. 4. Fig.4 (a) shows the weighting values of the memory cells. Fig.4 (b)(c) are the error output, it is clear that the new CIVSC approach can give an almost accurate servo tracking response to partial known parameters system. Figs. 4(d) shows the waveforms of control functions, and we

see that the CMAC outputs are dominant gradually. Fig. 4(e), the phase plane plot, also shows the output nearly tracks the integral sliding surface.

**Remark 2:** The selection of  $M_x$ ,  $M_u$  and  $M_w$  are dependent on practical constraints.  $M_x$  and  $M_u$  represent the maximum rotated angle of robot arm and the maximum control signal of the D/A card output. It is easy to specify as 5 rad and 10 volt. Using (33) we let the  $M_w=2$ . The control result of Fig. 4(a) shows  $|w|$  1.226 that is smaller than  $M_w$ .

## 5. Discussion

In Fig.4(e) we see that the trajectory vibrated in the neighborhood of hitting point. Before the trajectory hit the sliding surface, the learning is blind. It is easy to cause over tuning and give rise to the chattering. We wish the learning behavior happened in the neighborhood of the desired surface, therefore using multiple segments sliding surface is better than only one sliding surface. But related sliding surfaces design and learning law proof are under studying.

The stabilizer control just to preserves the output states near the sliding surface, the switching gain depends on the parameters bounds should be chosen carefully. Too large switching gain is easy to cause vibration. Fortunately, the stabilizer control is only active when some special conditions exist. In Fig.4(d) we see that  $u_s$  is equal to zero for most time.

The training gain is related to the learning effect. In generally, the value of is between 0 and 1 for supervised learning [18], i.e.  $0 < \dots < 1$ . The memory size of CMAC also affects the control performance. The memory size depends on the desired input signal and the performance requirement. It is hard to make a clear decision but we can obtain some idea from the activity of memory address. For example, the weighting value distribution plot (Fig. 4(a)) shows that not all the addresses excited, in this case the memory size is enough. We can reduce the quantization level or association cells number to decrease the memory size. On the other hand, if all the addresses have been excited, the memory size may be not enough. We need to adjust the quantization level or association cells number to increase the memory size. When the memory size is too large beyond the computer specification, the Hash coding is needed.

## 6. Conclusion

In this paper, a new integral variable structure control scheme is proposed to control the nonlinear system. A CMAC network is used to learn the equivalent control of CIVSC for the partial known system model. A stabilizer controller based on Lyapunov synthesis approach is designed to guarantee the stability of the proposed scheme in the sense that all signals involved are uniformly bounded. This CIVSC scheme only used little model information to design the controller. Using the generalization property of neural network, the CMAC can learn the approximated equivalent control signal on line and yield a robust sliding mode control. It can keep all the advantages of IVSC and alleviate the dependency to system parameters. Applying this scheme to a simplified robot manipulator model, it can

produce an integral variable structure control signal for specified sliding surface. The simulation results demonstrate the success of the proposed scheme.

## 6. Reference

1. T. L. Chern, Y. C. Wu," Design of integral variable structure controller and application to electrohydraulic velocity servosystems", *IEE Proc. D* vol. 138, no. 5, pp. 439-444, 1991.
2. T. L. Chern, Y. C. Wu," Integral variable structure control approach for robot manipulator", *IEE Proc. D* vol. 139, no. 2, pp. 161-166, 1992.
3. T. L. Chern, Y. C. Wu," Design of brushless DC position servo systems using integral variable structure approach", *IEE Proc. B* vol. 140, no. 1, pp. 27-34, 1993.
4. S. K. Chung, J. H. Lee, J. S. Ko, M. J. Youn," Robust speed control of brushless direct-driver motor using integral variable structure control", *IEE Proc. Electr. Power Appl.*, vol. 142, no. 6, pp. 361-370, 1995.
5. T. L. Chern, J. S. Wong," DSP based integral variable structure control for DC motor servo drivers", *IEE Proc. Control Theory Appl.* vol. 142, no. 5, pp. 444-450, 1995.
6. T. L. Chern, C. S. Liu, C. F. Jong, and G. M. Yan, "Discrete integral variable structure model following control for induction motor drivers", *IEE Proceedings-Electric Power Applications*, vol. 143, no. 6, Nov., pp. 467-474, 1996.
7. J. D. Wang, T. L. Lee, Y. T. Juang," New methods to design an integral variable structure controller", *IEEE Trans. Automation Control*, vol. 41, no. 1, pp. 140-143, 1996.
8. T. L. Chern, J. Chang, G. K. Chanf," DSP based integral variable structure model following control for brushless DC motor drivers", *IEEE Trans. Power Electronics*, vol. 12, no. 1, pp. 53-63, 1997.
9. T. L. Chern, J. Chang, " DSP based induction motor drivers using integral variable structure model following control approach", *IEEE Trans. Power Electronics*, vol. 12, no. 1, pp. 53-63, 1997.
10. T. L. Chern, C. W. Chuang, "Design of optimal MIMO DIVSC systems and its application to idle speed control of spark ignition engine", *ASME Journal of Dynamic Systems Measurement and Control*, vol. 119, June, pp.175-182, 1997.
11. T. L. Chern, J. Chang, K. L. Tsai, "IVSC-based Adaptive Speed Estimator and Resistance Identifier for an Induction Motor", *International Journal of Control*, vol. 69, no. 1, pp.31-47, 1998.
12. T. L. Chern, G. K. Chang, "Automatic voltage regulator design by modified discrete integral variable structure model following control", *Automatica*, pp.1575-1582, 1998.
13. T. L. Chern, J. Chang, C. H. Chern, H. T. Su, "Microprocessor-based Modified Discrete Integral Variable Structure Control for UPS," *IEEE Trans. Industrial Electronics*, April, 1999.
14. K. K. D. Young, " Controller design for a manipulator using theory of variable structure systems", *IEEE Trans. Syst., Man, Cybern.* , vol. SMC-8 , no. 2, pp.101-109, 1978.
15. J. J. Slotine, S. S. Sastry, Tracking control of nonlinear systems using sliding surface, with application to robot manipulators , *INT. J. Control*, vol.38, no. 2, pp. 465-492, 1983.
16. W. S. Lin, C. P. Hung," Variable structure control of unknown parameter dc servo systems using CMAC-based learning approach", *Proc. Am. Control Conf.*, 2001.
17. J. S. Albus, "A new approach to manipulator control: the

cerebellar model articulation controller (CMAC)<sup>1</sup>, *Trans. ASME J. Dynam., Syst., Meas., and Contr.*, vol. 97, pp.220-227, 1975

18. D. A. Handeiman, S. H. Lanc, and J. J. Gelfand, "Integrating neural networks and knowledge-based systems for intelligent robotic control" *IEEE Control System Magazine*, pp. 77-86, 1990.
19. L. X. Wang, "Stable adaptive fuzzy control of nonlinear system", *IEEE Trans. on Fuzzy Sytem*, vol. 1,no. 2, pp. 146-155, 1993.
20. Y. F. Wong, A. Sideris, "Learning convergence in the cerebellar model articulation controller", *IEEE Trans. on Neural Network*, vol. 3,no. 1, pp. 115-121, 1992.
21. K. S. Fu, R.C. Gonzalez, C.S.G. Lee," ROBOTICS: control, sensing, vision, and intelligence", McGRAW-Hill, 1987.
22. A. Davari, and Z. Zhang, "Application of the three-segments variable structure systems", *Proc. Am. Control Conf.*, 1, pp. 62-63, 1991.
23. S. Commuri, S. Jagannathan, F. L. Lewis, "CMAC neural network control of robot manipulators", *Journal of Robotic System*, 14(6), pp.465-482, 1997.
24. W. S. Lin, and C. H. Tsai," Neurofuzzy-model-following control of MIMO nonlinear systems", *IEE Proc. Control Theory Appl.* vol. 146, no. 2, pp. 157-164, 1999.

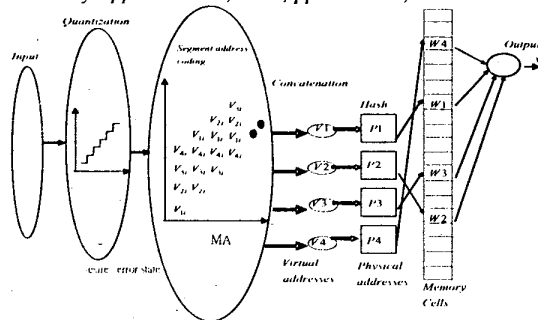


Fig. 1 Functional schematic of CMAC

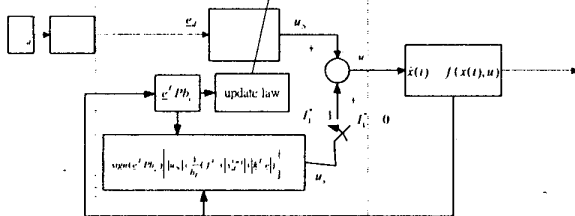


Fig.2 Block diagram of CIVSC

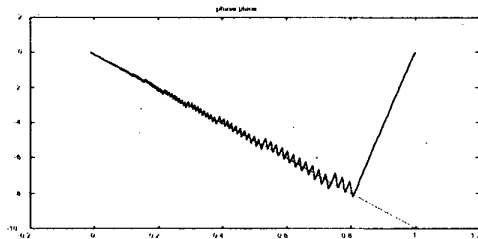


Fig.3 phase plane only the stabilizer control (without the CMAC network)

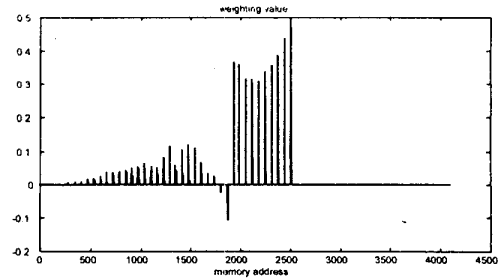


Fig. 4(a) Weight value of memory cells for CMAC network

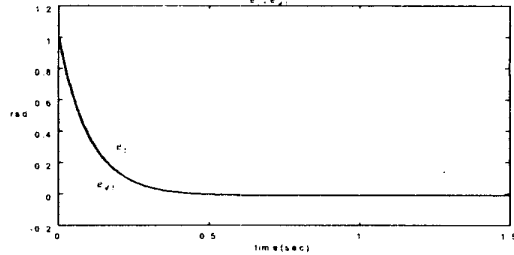


Fig. 4(b) Desired and actual output state  $x_1$

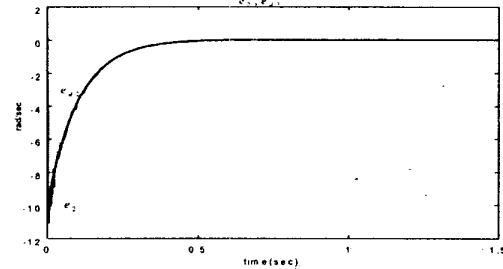


Fig. 4(c) Desired and actual output state  $x_2$

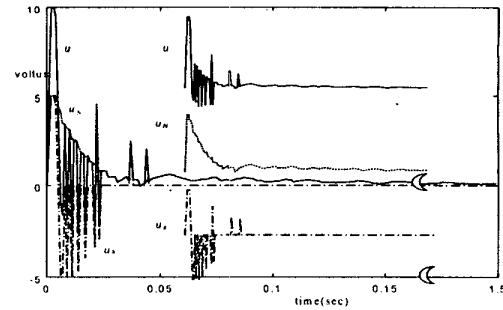


Fig. 4(d) Control signal  $u_N, u_S$  and  $u_y$

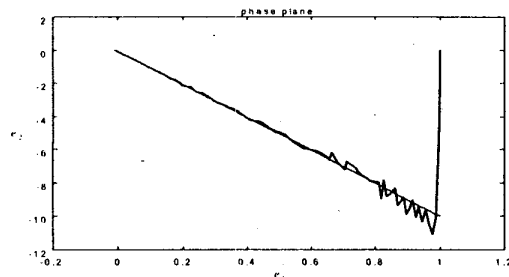


Fig. 4(e) Phase plane plot of desired and actual output states