Self-Structured Organizing Single-Input CMAC Control for De-icing Robot Manipulator

Thanhquyen Ngo¹², Yaonan Wang¹, Youhui Chen¹, Zan Xiao¹
¹College of Electrical and Information Engineering, Hunan University, Changsha, China
²Faculty of Electrical Engineering, Ho Chi Minh City University of Industry, Ho Chi Minh, Vietnam
E-mail: thanhquyenngo2000@yahoo.com, yaonan@hnu.cn
Received June 1, 2011; revised June 22, 2011; accepted June 29, 2011

Abstract

This paper presents a self-structured organizing single-input control system based on differentiable cerebellar model articulation controller (CMAC) for an n-link robot manipulator to achieve the high-precision position tracking. In the proposed scheme, the single-input CMAC controller is solely used to control the plant, so the input space dimension of CMAC can be simplified and no conventional controller is needed. The structure of single-input CMAC will also be self-organized; that is, the layers of single-input CMAC will grow or prune systematically and their receptive functions can be automatically adjusted. The online tuning laws of single-input CMAC parameters are derived in gradient-descent learning method and the discrete-type Lyapunov function is applied to determine the learning rates of the proposed control system so that the stability of the system can be guaranteed. The simulation results of three-link De-icing robot manipulator are provided to verify the effectiveness of the proposed control methodology.

Keywords: Cerebellar Model Articulation Controller (CMAC), De-icing Robot Manipulator, Gradient-Descent Method, Self-Organizing, Signed Distance

1. Introduction

In general, robotic manipulators have to face various uncertainties in their dynamics, such as friction and external disturbance. It is difficult to establish exactly mathematical model for the design of a model-based control system. In order to deal with this problem, the branches of current control theories are broad including classical control: neural networks (NNs) control [1-3], adaptive fuzzy logic control (FLCs) [4-6] or adaptive fuzzy-neural networks (FNNs) [7-9] etc. They are classified as adaptive intelligent control based on conventional adaptive control techniques where fuzzy systems or neural networks are utilized to approximate a nonlinear function of the dynamical systems. However, many adaptive approaches are rejected as being overly computationally intensive because of the real-time parameter identification and required control design.

Fuzzy logic control (FLCs) has found extensive applications for plants that are complex and ill-defined which is suitable for simple second order plants. However, in case of complex higher order plants, all process states are required as fuzzy input variables to implement state feedback FLCs. All the state variables must be used to represent contents of the rule antecedent. So, it requires a huge number of control rules and much effort to create. To address these issues, single-input Fuzzy Logic controllers (S-FLC) was proposed for the identification and control of complex dynamical systems [10-12]. As a result, the number of fuzzy rules is greatly reduced compared to the case of the conventional FLCs, but its control performance is almost the same as conventional FLCs.

Neural networks (NNs) are a model-free approach, which can approximate a nonlinear function to arbitrary accuracy [1-3]. However, the learning speed of the NNs is slow. To deal with these issues, cerebellar model articulation controller (CMAC) was proposed by Albus in 1975 [13] for the identification and control of complex dynamical systems, due to its advantage of fast learning property, good generalization capability and ease of implementation by hardware [13-15]. The conventional CMACs, regarded as non-fully connected perceptron-like associative memory network with overlapping receptive fields which used constant binary or triangular functions. The disadvantage is that their derivative information is not preserved. For acquiring the derivative information of
input and output variables, Chiang and Lin [16] developed a CMAC network with a differentiable Gaussian receptive-field basis function and provided the convergence analysis for this network. The advantages of using CMAC over neural network in many applications were well documented [17-21]. However, in the above CMAC literatures, the structure of CMAC cannot be obtained automatically. The amount of memory space is difficult to select, which will influence the learning and control schemes. Some self-organizing CMAC neural networks were proposed for structure adaptation [22-25]. In [22] and [23] a data clustering technique is used to reduce the memory size and a structural adaptation technique is developed in order to accommodate new data sets. However, only the structure growing mechanism is introduced and; the pruning mechanism was not discussed. In [24], a self-organizing hierarchical CMAC was introduced. The authors proposed a multilayer hierarchical CMAC model and used Shannon’s entropy measure and golden-section search method to determine the input space quantization. However, their approach is too complicated and lacks online real-time adaptation ability.

In [25] a data clustering technique is used to reduce the memory size and a structural adaptation technique is developed in order to accommodate new data sets. However, only the structure growing mechanism is introduced and; the pruning mechanism was not discussed. In [24], a self-organizing hierarchical CMAC was introduced. The authors proposed a multilayer hierarchical CMAC model and used Shannon’s entropy measure and golden-section search method to determine the input space quantization. However, their approach is too complicated and lacks online real-time adaptation ability.

2. System Description

In general, the dynamic of an $n$-link robot manipulator may be expressed in the Lagrange following form:

$$M(q)\ddot{q}+C(q, \dot{q}) \dot{q} + G(q) + N = \tau$$

where $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ are the joint position, velocity and acceleration vectors, respectively, $M(q) \in \mathbb{R}^{nn}$ denotes the inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{nn}$ expresses the matrix of centripetal and Coriolis forces, $G(q) \in \mathbb{R}^{nn}$ is the gravity vector, $N \in \mathbb{R}^{nn}$ represents the vector of external disturbance $t_i$, friction term $f(\dot{q})$, and un-modeled dynamics, $\tau \in \mathbb{R}^{nn}$ is the torque vectors exerting on joints. In this paper, a new three-link De-icing robot manipulator as shown in Figure 1, is utilized to verify dynamic properties are given in section 4.

The control problem is to force $q_i(t) \in \mathbb{R}^n$, $i=1, 2, \cdots m$ to track a given bounded reference input signal $q_{ref}(t) \in \mathbb{R}^n$. Let $e_i(t)$ be the tracking error vector as follows:

$$e_i = q_{ref} - q_i, \quad i=1, 2, \cdots m$$

and the system tracking error vector is defined as

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix}$$

Figure 1. Architecture of three-link De-icing robot manipulator.
control characteristics due to using the differentiable CMAC in the system. The advantage is that derivative information of input and output variables is preserved in learning process. In addition, the generalization error caused by quantization of input space could be reduced while using the differentiable CMAC.

Based on the standalone CMAC control system, we propose the SOSICM control system as shown in Figure 4, which combines advantages of standalone CMAC and it does not require prior knowledge of a certain amount of memory space. The self-organizing approach demonstrates the properties of generating and pruning the input layers automatically. The developed self-organizing rule of CMAC is clearly and easily used for real-time systems.

### 3. Adaptive SOSICM Control System

#### 3.1. Brief of the S-CMAC

An S-CMAC is proposed and shown in Figure 5. It is composed of an input, association memory, weight and output spaces. The signal propagation and the basic function in each space are expressed as follows:

1. Input space $D_i$; assume that each input state variable $d_{ii}$ can be quantized into $N_{ii}$ discrete states and that the information of a quantized state is regarded as region for each layer $n_{ii}th$. Therefore, there exist $N_{ii}+1$ individual points on the $d_{ii}$ - axis. Figure 6 shows the case of $N_{ii}=10$. Each activated state in each layer becomes a firing element, thus, the weight of each layer can be obtained. The Gaussian basic function for each layer is given as follows:

$$
\Delta k_{ij} = \frac{1}{\sqrt{2\pi \sigma_i^2}} e^{-\frac{(d_{ij} - \mu_i)^2}{2\sigma_i^2}}
$$

where $\mu_i$ is the mean of the Gaussian function in layer $i$ and $\sigma_i$ is the standard deviation. The firing element in layer $i$ is given by

$$
\eta_i = \sum_{j=1}^{N_{ii}} \alpha_{ij} \Delta k_{ij}
$$

where $\alpha_{ij}$ is the weight between layer $i$ and layer $j$. The output of the S-CMAC is given by

$$
R = \sum_{i=1}^{n} \eta_i
$$

where $n$ is the number of layers in the S-CMAC. The learning rule for the S-CMAC is given by

$$
\delta w_{ij} = \eta_i \eta_j
$$

where $\delta w_{ij}$ is the adjustment in the weight between layer $i$ and layer $j$. The learning rate is given by

$$
\eta = \frac{1}{\sqrt{\sum_{i=1}^{n} \eta_i^2}}
$$

where $\eta$ is the learning rate. The adaptive law is given by

$$
\beta_i = \frac{\eta_i}{\sum_{j=1}^{N_{ii}} \eta_j}
$$

where $\beta_i$ is the adaptive weight for the $i$th layer. The self-organizing rule of CMAC is clearly and easily used for real-time systems.

- **Figure 2.** Derivation of a signed distance.
- **Figure 3.** Block diagram of standalone CMAC control system.
- **Figure 4.** Block diagram of proposed SOSICM control system.
chosen too small, the learning performance may be insufficient to achieve a desired performance. Otherwise, if the number of layers is chosen too large, the calculation process is too heavy, so it is not suitable for real-time applications. To deal with this problem, a self-structured organizing S-CMAC is proposed includes structure and parameter learning as shown in Figure 4.

3.2. Self-Structured Organizing S-CMAC

In this section, structural learning is necessary to determine whether to add a new layer in association memory A depends on the firing strength $\phi_{ki} \in R^{\infty}$ of each layer for each incoming data $d_{ui}$. If the firing strength $\phi_{ki} \in R^{\infty}$ of each layer for new input data $d_{ui}$ falls outside the bounds of the threshold, then, SOSICM will generate a new layer. The self-structured organizing S-CMAC can be summarized as follows:

1) Calculate the firing strength $\phi_{ki} \in R^{\infty}$ of each layer for each input data $d_{ui}$ in (6).

\[
\hat{k}_i = \arg \min_{1 \leq k \leq n_{ki}} \phi_{ki} (d_{ui}), \quad k = 1, 2, \ldots, n_{ki}
\]

(8)

If

\[
\phi_{ki} (d_{ui}) < K_{gi}
\]

(9)

Here $K_{gi}$ is a threshold value of adaptation with $0 < K_{gi} \leq 1$. In our case $K_{gi} = 0.1$ and a new layer is generated.

This means that for a new input data, the exciting value of existing basic function is too small. In this case, number of layers increased as follows:

\[
n_{ki} (t+1) = n_{ki} (t) + 1
\]

(10)

where $n_{ki}$ is the number of layers at time $t$. Thus, a new layer will be generated and then the corresponding parameters in the new layer such as the initial mean and variance of Gaussian basic function in association memory space and the weight memory space will be defined as

\[
m_{w_{ki}} = d_{ui}
\]

(11)

\[
\sigma_{w_{ki}} = \sigma_{w_{ki}}
\]

(12)

\[
w_{w_{ki}} = 0
\]

(13)

Another self-structured organizing learning process is considered to determine whether to delete existing layer, which is inappropriate. A Max-Min method is proposed for layer pruning.

Considering the output of SOSICM in (7), the ratio of the $k$th component of output is defined as
\[ MM_{ki} = \frac{v_{ki}}{\tau_{i}}, \quad k = 1, 2, \ldots, n_{ki} \]  

(14)

where \( v_{ki} = \phi_{ki} w_{ki} \). Then, the minimum ratio of the \( k \)th component is defined as follows:

\[ \tilde{k}_i = \arg \min_{k \in 1, k \in n_{ki}} MM_{ki} \]

(15)

If

\[ MM_{\tilde{k}_i} \leq K_{ci} \]

(16)

Here \( K_{ci} \) is a predefined deleting threshold. In our case \( K_{ci} = 0.03 \) and the \( \tilde{k}_i \)th layer will be deleted. This means that for an output data, if the minimum contribution of a layer is less than the deleting threshold, then this layer will be deleted.

### 3.3. On-Line Learning Algorithm

The central part of the learning algorithm for a SOSICM is how to choose the weight memory \( w_{ki} \), mean \( m_{ki} \), variance \( \sigma_{ki} \) of the Gaussian basic function. \( \tilde{k}_i \) are the scaling factors of the error \( e_i \) and the change of error \( \dot{e}_i \), which will significantly affect the performance of SICM. For achieving effective learning, an on-line learning algorithm, which is derived using the supervised gradient descent method, is introduced so that it can in real-time adjust the parameters of SOSICM. The energy function \( E_i \) is defined as

\[ E_i = \frac{1}{2} (q_i - q_i^2) = \frac{1}{2} e_i^2 \]

(17)

According to the energy function (17) and the system structure in Figure 4, the error term to be propagated is given by

\[ \delta_i = -\frac{\partial E_i}{\partial \tau_i} = -\frac{\partial E_i}{\partial q_i} \frac{\partial q_i}{\partial \tau_i} = e_i \frac{\partial \dot{e}_i}{\partial \tau_i} \]

(18)

where \( \frac{\partial \dot{e}_i}{\partial \tau_i} \) represent the sensitivity of the plant with respect to its input. With the energy function \( E_i \), the parameters updating law based on the normalized gradient descent method can be derived as follows

1) The updating law for the \( k \)th weight memory can be derived according to

\[ \Delta w_{ki} = -\beta_{wi} \frac{\partial E_i}{\partial w_{ki}} = -\beta_{wi} \frac{\partial E_i}{\partial q_i} \frac{\partial q_i}{\partial \tau_i} \]

(19)

\[ = a_{ki} \beta_{wi} \delta_i \phi_{ki} (d_i) \]

where \( \beta_{wi} \) is positive learning rate for the output weight memory \( w_{ki} \). The connective weight can be updated according to the following equation:

\[ w_{ki} (t + 1) = w_{ki} (t) + \Delta w_{ki} \]

(20)

2) The mean and variance of the \( k \)th Gaussian basic function can be also updated according to

\[ \Delta m_{ki} = -\beta_{mi} \frac{\partial E_i}{\partial m_{ki}} = -\beta_{mi} \frac{\partial E_i}{\partial q_i} \frac{\partial q_i}{\partial \tau_i} \]

\[ = a_{ki} \beta_{mi} \delta_i \phi_{ki} (d_i) \]

(21)

\[ \Delta \sigma_{ki} = -\beta_{si} \frac{\partial E_i}{\partial \sigma_{ki}} = -\beta_{si} \frac{\partial E_i}{\partial q_i} \frac{\partial q_i}{\partial \tau_i} \]

\[ = a_{ki} \beta_{si} \delta_i \phi_{ki} (d_i) \]

(22)

where \( \beta_{mi} \), \( \beta_{si} \) are positive learning rates for the mean and variance, respectively. The mean and variance can be updated as follows:

\[ m_{ki} (t + 1) = m_{ki} (t) + \Delta m_{ki} \]

(23)

\[ \sigma_{ki} (t + 1) = \sigma_{ki} (t) + \Delta \sigma_{ki} \]

(24)

3) Finally, the updating law for scaling factors can be derived as follows:

\[ \Delta k_{si} = -\beta_{ki} \frac{\partial E_i}{\partial k_{si}} = -\beta_{ki} \frac{\partial E_i}{\partial \tau_i} \frac{\partial \tau_i}{\partial \tau_i} \]

\[ = \beta_{ki} \delta_i \phi_{ki} (d_i) \]

(25)

\[ \sqrt{1 + \lambda_{s1}^2 + \ldots + \lambda_i^2 + \lambda_i^2} \]

where \( \lambda_{si} \) is the learning rate, and it can be updated by the following:

\[ k_{si} (t + 1) = k_{si} (t) + \Delta k_{si} \]

(26)

The plant sensitivity \( \frac{\partial q_i}{\partial \tau_i} \) in (18) can be calculated if the plant model is exactly known. However, the plant model is unknown, so \( \frac{\partial q_i}{\partial \tau_i} \) cannot be obtained in advance. To deal with this problem, in [28], a simple approximation of the error term of the system can be used as follows:

\[ \delta_i \approx \dot{e}_i + e_i \]

(27)

### 3.4. Convergence Analysis

The update laws of Equations (19), (21), (22) and (25) require a proper choice of the learning rates \( \beta_{mi} \), \( \beta_{si} \), \( \beta_{wi} \), and \( \beta_{ki} \) in order to the convergence of the output error is guaranteed; however, this is not easy which depends on each person’s experience. To train the S-CMAC effectively, the variable learning rates which guarantee convergence of the output error are derived in the following.

Defined a discrete-type Lyapunov function can be
given by

\[ V_i(k) = \frac{1}{2} e_i^2(k) \]  

Thus, the change of the Lyapunov due to the training process is obtained as

\[ \Delta V_i(k) = V_i(k+1) - V_i(k) = \frac{1}{2} \left[ e_i^2(k+1) - e_i^2(k) \right] \]  

(29)

where \( e_i(k+1) \) is represented by [28]

\[ e_i(k+1) = e_i(k) + \Delta \xi_i(k) + \frac{\partial e_i(k)}{\partial P} \Delta P_i \]  

(30)

where \( \Delta \xi_i \) represents the in the learning process, \( \Delta P_i \) denotes a change of an adjustable parameters. Using Equation (18), we have

\[ \frac{\partial e_i}{\partial P} = -\delta_p \frac{\partial e_i}{\partial \xi} \frac{\partial \xi}{\partial P} \]

and

\[ \Delta P_i = -\beta_{pi} \frac{\partial e_i}{\partial \xi} \frac{\partial \xi}{\partial P} = \beta_{pi} \frac{\partial e_i}{\partial \xi} \frac{\partial \xi}{\partial P} \]

where \( \beta_{pi} \) is the learning rate for the parameter \( P_i \).

Thus:

\[ \Delta V_i(k) = \Delta \xi_i(k) \left[ e_i(k) + \frac{1}{2} \Delta e_i(k) \right] \]

\[ = - \beta_{pi} \delta_p \frac{\partial \xi}{\partial P} \frac{\partial e_i}{e_i(k)} \left[ e_i(k) - \frac{1}{2} \beta_{pi} \frac{\partial \xi}{\partial P} \frac{\partial e_i}{e_i(k)} \right] \]

\[ = \frac{1}{2} \beta_{pi} \delta_p \frac{\partial \xi}{\partial P} \left[ \frac{\partial e_i}{e_i(k)} \right] \left[ \beta_{pi} \frac{\partial e_i}{e_i(k)} \right] \frac{\partial \xi}{\partial P} \]

\[ = \frac{1}{2} \beta_{pi} \delta_p \frac{\partial \xi}{\partial P} \left[ \frac{\partial e_i}{e_i(k)} \right] \left[ \frac{\partial e_i}{e_i(k)} \right] \frac{\partial \xi}{\partial P} \]

(31)

If the learning rate \( \beta_{pi} \) is selected as:

\[ 0 < \beta_{pi} < 2 \left[ \left( \frac{\partial e_i}{e_i(k)} \right) \left[ \frac{\partial e_i}{e_i(k)} \right] \frac{\partial \xi}{\partial P} \right] \]

(32)

Then \( \Delta V_i(k) \leq 0 \), therefore \( V_i(k+1) \leq V_i(k) \), the Lyapunov stability (system stability) and the convergence of the tracking error could be guaranteed. In addition, the optimal learning rate can be found for achieving faster convergence by taking the differential equation (31) with respect to \( \beta_{pi} \) and equals to zero. Finally, the optimal learning rate can be determined as follows:

\[ \beta_{pi} = \sqrt{ \left( \frac{\partial e_i}{e_i(k)} \right) \left[ \frac{\partial e_i}{e_i(k)} \right] \frac{\partial \xi}{\partial P} } \]

(33)

where \( \frac{\partial \xi}{\partial P} \) for \( P_i = \omega_{li}, m_{li}, \sigma_{li} \) and \( k_{li} \), it can be obtained as:

\[ \frac{\partial \xi_i}{\partial \omega_{li}} = a_{li}\phi_{li} \]

\[ \frac{\partial \xi_i}{\partial m_{li}} = a_{li}w_{li}\phi_{li} \]

\[ \frac{\partial \xi_i}{\partial \sigma_{li}} = a_{li}w_{li}\phi_{li} \]

\[ \frac{\partial \xi_i}{\partial k_{li}} = a_{li}w_{li}\phi_{li} \]

4. Simulation Results

A three-link De-icing robot manipulator as shown in Figure 1 is utilized in this paper to verify the effectiveness of the proposed control scheme. The detailed system parameters of this robot manipulator are given as: link mass \( m_1, m_2, m_3(kg) \), lengths \( l_1, l_2(m) \), angular positions \( q_1,q_2(rad) \) and displacement position \( d_i(m) \).

The parameters for the equation of motion (1) can be represented as follow:

\[ M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \]

\[ M_{11} = 9/4m_1l_1 + m \left( 1/4c_1l_2 + l_1l_2 \right) \]

\[ + m_2 \left( c_1l_2^2 + l_2^2 + 2c_2l_2l_3 \right) \]

\[ M_{22} = 1/4m_1l_2^2 + m_2l_2^2 + 4/3m_2l_2^2 \]

\[ M_{33} = m_3 \]

\[ M_{12} = M_{21} = M_{31} = 0 \]

\[ C(q) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \]

\[ C_{11} = -8m_2l_2c_2s_2q_1 \]

\[ + \left( -1/2m_2s_2c_2l_2^2 + m_2 \left( -2c_2c_3l_2^2 - 2s_2l_1l_2 \right) \right) q_2 \]

\[ C_{21} = -1/2m_2s_2c_2l_2^2 + m \left( -2c_2c_3l_2^2 - 2s_2l_1l_2 \right) q_1 \]

\[ C_{22} = -m_2s_2q_1 \]

\[ C_{23} = -2m_2s_2q_2 \]

\[ C_{32} = -m_2s_2q_2 \]

\[ C_{12} = C_{13} = C_{31} = C_{33} = 0 \]

\[ G(q) = \begin{bmatrix} \left( 1/2c_2c_3l_2 + c_3l_3 \right) m_2g \\ -1/2s_2s_2l_2^2 + c_3l_2^2 \right) m_2g \]

\[ m_2g \]

where \( q \in R^3 \) and the shorthand notations \( c_1 = \cos(q_1), c_2 = \cos(q_2), s_1 = \sin(q_1) \) and
\( s_2 = \sin(q_2) \) are used.

For the convenience of the simulation, the nominal parameters of the robotic system are given as: \( m_1 = 3(\text{kg}) \), \( m_2 = 2(\text{kg}) \), \( m_3 = 2.5(\text{kg}) \), \( l_1 = 0.14(\text{m}) \), \( l_2 = 0.32(\text{m}) \), and \( g = 9.8(\text{m/s}^2) \) and the initial conditions \( q_t(0) = 1 \), \( q_2(0) = 0 \), \( d_2(0) = 0 \), \( \dot{q}_t(0) = 0 \), \( \dot{q}_2(0) = 0 \), \( d_3(0) = 0 \). The desired reference trajectories are \( q_{d1}(t) = \sin(t) \), \( q_{d2}(t) = \cos(t) \), \( d_{d3}(t) = \cos(t) \), respectively.

The most important parameters that affect the control performance of the robotic system are the external disturbance \( t_f \), the friction term \( f(\dot{q}) \), which are injected into the robotic system, and their shapes are expressed as follows:

\[
t_f(t) = \begin{bmatrix} 5\sin(5t) & 5\sin(5t) & 5\sin(5t) \end{bmatrix}^T
\]

(36)

In addition, friction forces are also considered in this simulation and given as

\[
f(\dot{q}) = \begin{bmatrix} 20\dot{q}_1 + 0.8\text{sgn}(\dot{q}_1) & 4\dot{q}_2 & 2\text{sgn}(\dot{q}_2) \end{bmatrix}
\]

\[
4d_3 + 2\text{sgn}(d_3)
\]

(37)

In order to exhibit the superior control performance of the proposed SOSICM control system, the control system standalone CMAC is introduced in Figure 3 and examined in the mean time [28]. They are applied to control three-link De-icing robot manipulator and the same setting of SOSICM and standalone CMAC control system are chosen as follows: The inputs space of S-CMAC are \( d_{s1} \), \( d_{s2} \) and \( d_{s3} \), the mean and variance of Gaussian basic functions are selected to cover the input space \( \left\{ [-1, 1], [-1, 1], [-1, 1] \right\} \); all initial weights are set to zero, i.e., \( w_{ki} = w_{kj} = 0 \), \( k = 1, 2, \cdots, n_k \). The parameter \( \lambda \) in the switching line is one. For recording respective control performance, the mean-square-error of the position-tracking response is defined as:

\[
mse_i = \frac{1}{T} \sum_{j=1}^{T} [q(i) - q_{d}(j)]^2, \; i = 1, 2, 3
\]

(38)

where \( T \) is the total sampling instant, and \( q_i \) and \( q_{di} \) are the elements in the vector \( q_i \) and \( q_{di} \). In this paper, the numerical simulation results carried out by using Matlab software.

**Example 1:** Consider the standalone CMAC control system is shown in Figure 3.

For the standalone CMAC control system, the parameters are chosen such as: \( \beta_{\text{mi}} = 0.02 \), \( \beta_{\text{wi}} = 0.02 \), \( \beta_{\text{bi}} = 0.02 \), \( \beta_{\text{ii}} = 0.02 \), the initial value of Gaussian basic functions and scaling factors are defined as \( m_{\text{bi}} = -1.0 \), \( m_{\text{wi}} = -0.8 \), \( m_{\text{bi}} = -0.6 \), \( m_{\text{ii}} = -0.4 \), \( m_{\text{wi}} = -0.2 \), \( m_{\text{bi}} = 0.0 \), \( m_{\text{ii}} = 0.2 \), \( m_{\text{ii}} = 0.4 \), \( m_{\text{i}} = 0.6 \)

\( m_{\text{wi}} = 0.8 \), \( m_{\text{bi}} = 1.0 \), \( \sigma_{\text{bi}} = 0.15 \), \( k_{\text{bi}} = 0.5 \) and \( k_{\text{ii}} = 0.2 \) for \( k = 1, 2, \cdots, 11 \), \( i = 1, 2, 3 \). The simulation results of joint position and MSE are depicted Figures 7(a)-(f), respectively.
Example 2: Consider the proposed SOSICM control system is shown in Figure 4.

For the proposed SOSICM control system, the parameters are chosen in the following:

\[
\beta_{pi} = \frac{1}{\left| \mathbf{\delta}_{pi} \right|} \left\| \mathbf{\delta}_{pi} \right\| \mathbf{\delta}_{pi} \mathbf{P}_{i} \quad \text{for} \quad P_i = w_{o}, m_{o}, \sigma_{o} \quad \text{and} \quad k_{o},
\]

and the initial values of system parameters are given as \( n_i = 2 \), the inputs of S-CMAC \( d_{i1}, d_{i2} \) and \( d_{i3} \), the mean and variance of Gaussian basic functions are selected to cover the input space \( \left\{ [-1 \ 1][-1 \ 1][-1 \ 1] \right\} \).

The threshold value of \( \gamma_i \) is set as 0.1; \( \gamma_i \) is set as 0.01 for \( i = 1, 2, 3 \). The simulation results of the proposed SOSICM system, the responses of joint position, MSE and layer number are depicted in Figures 8(a)-(f) and (g), (h) and (k) respectively.

According to the simulation results as shown in Figures 7 and 8, the joint-position tracking responses of the proposed SOSICM system can be controlled to more closely follow desired reference trajectories than the standalone CMAC as shown in Figure 7 and Figures 8(a)-(c). Our proposed control system for each joint shows that the MSE in Figures 8(d)-(f) is faster than the MSE in Figures 7(d)-(f) and finally converges to 0.009, 0.015 and 0.019. Meanwhile the MSE of standalone CMAC is 0.032, 0.031 and 0.036 and number of layers of S-CMACs converge to three layers as shown in Figures 8(g), (h) and (k).

5. Conclusions

In this paper, a SOSICM control system is proposed to
control the joint position of a three-link De-icing robot manipulator. In the SOSICM system, dynamical system is completely unknown and auxiliary compensated control is not required in the control process. The online tuning laws of S-CMAC parameters are derived in gradient-descent learning method and the discrete-type Lyapunov function is applied to determine the variable optimal learning rates so that the stability of the system can be guaranteed. This paper has successfully developed the SOSICM control system for a three-link De-icing robot manipulator not only requires low memory with online structure and parameters tuning algorithm, but also the input space can be reduced through the signed distance. The simulation results of the proposed SOSICM system can achieve favorable tracking performance for three-link De-icing robot manipulator.

6. Acknowledgments

The authors would like to thank the reviewers for their valuable comments.

7. References


