비선형 제어 시스템을 이용한 다단동적 신경망 제어기 설계

노용기* 김원중** 조현섭***
*전북인력개발원 일렉트로닉스공과
**한국폴리텍V대학 순천캠퍼스 컴퓨터응용기계과
***청운대학교 디지털방송공학과
*e-mail : ygroh@korcham.net

Design of Multi-Dynamic Neural Network Controller using Nonlinear Control Systems

Yong-Gi Rho* Won-Jung Kim** Hynu-Seob Cho***
* Major of Electronics JBHRDI
** Major of Computer Application Machine KOPO
*** Dept of Digital Broadcasting ENG. Chungwoon UNIV.

Abstract

The intent of this paper is to describe a neural network structure called multi dynamic neural network(MDNN), and examine how it can be used in developing a learning scheme for computing robot inverse kinematic transformations. The architecture and learning algorithm of the proposed dynamic neural network structure, the MDNN, are described. Computer simulations are demonstrate the effectiveness of the proposed learning using the MDNN.

1. Introduction

Incessant researches have been made to identify the linear system with unknown variables and to control its adaptability, bringing forth remarkable outcomes. The conventional technique of adaptability control requires a mathematical model with dynamic system, but the dynamic system is non-linear and the strongly non-linear system is very difficult to control. Actually, it is believed to be impossible to exactly describe the dynamic system model due to nonlinear, uncertainty, time delay, variables of the time-dependent system, structure and etc. The preciseness of a robot controller, especially, depends on its moving ability to the target position on the rectangular coordinates in a specifically given working space. Hence, dynamics-related problems should be solved on real time basis in order to calculate the articulatory angles toward the position of the target control terminal. Such problems may be solved by the arithmetic or geometric method or repetition method [1] . The progress in the neural network area has led us to a new dimension of the robot control. The neural network, due to its advantageous properties of function value and dynamic repetition ability, can be used in learning the coordinates conversion. The neural network becomes able to learn how to combine exercise patterns through its parallel dispersion process. The structure of the neural network discussed in this paper is the result of interaction which is activated among neural sub-groups of excitatory

(positive) and inhibitory (negative) by neural activities with random complexity, that is, MDNN developed on the basis of neural physiology. A learning algorithm is herewith presented for the structure of MDNN and the flexible weight values for neural network. Results of learning method and computer simulations are also examined. [2-5]

2. Structure of Neural Network

2.1 Structure and Mathematic Model of MDNN

The basic function of MDNN with flexible synapse strength is based on dynamic neural unit. [6-8]

(i) Dynamic Neural Unit(DNU)

The memory unit of DNU is composed of forward and backward route synapse weight as shown in Fig. 1. The output of this dynamic structure comprises the components for time-dependent nonlinear activation function. DNU performs two major functions; (i) synaptic operation and (ii) somatic operation. The former corresponds to the adaptability of forward and backward route synapse weight and the latter to that of gain (form) in nonlinear activation function. What constitutes DNU is the forward and backward route delay units weighted by synapse weights a_{ff} and b_{fb}, which reveals the second structure following the nonlinear activation function.

$$v_1(k) = -b_1 v_1(k-1) - b_2 v_2(k-2) + a_0 s(k) + a_1 s(k-1) + a_2 s(k-2) \quad (1)$$

where $s(k) \in R^n$ is neural input vector, $v_1(k) \in R^1$ is output of dynamic structure, $u(k) \in R^1$ is neural output, k is dispersion time indicator, z^{-1} is unit delay indicator. $a_{ff} = [a_0, a_1, a_2]$ and $b_{fb} = [b_1, b_2]$ are defined as follows:

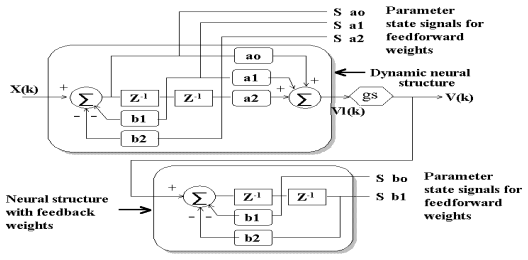

Fig. 1 Structure of DNU

$$\Gamma^T(k, v_1, s) = [v_1(k-1)\ v_2(k-1)]\ s(k)\ s(k-1)\ s(k-1) \quad (2)$$

$$\zeta^T(a_{ff}, b_{fb}) = [-b_1, -b_2, a_0, a_1, a_2] \quad (\Gamma\text{:transpose}) \quad (3)$$

Formula (1) is determined by (2) and (3) as follows;

$$v_1(k) = \Gamma(k, v-1, s)\ \zeta^T(a_{ff}, b_{fb}) \quad (4)$$

Nonlinear value for $v_1(k)$ yields following outputs;

$$u(k) = \Psi[g_s v_1(k) - \Theta] \quad (5)$$

where $\Psi[\cdot]$ is nonlinear activation function, normally called sigmoid function, g_s is somatic gain which controls the tilt of activation function and Θ is threshold igniting the neuron. In order to strengthen the mathematical activities of both excitatory and inhibitory, activation function for $[-1,1]$ should be defined as follows;

$$\Psi[v(k)] = \tan[g_s v_1(k) - \Theta] = \tanh[v(k)] \quad (6)$$

where $v(k) = g(s)\ v_1(k)$이다.

(ii) Multi Dynamic Neural Network(MDNN)

MDNN (Multi Dynamic Neural Network) is composed of two DNU combined with excitatory and inhibitory methods as shown in Fig. 2.

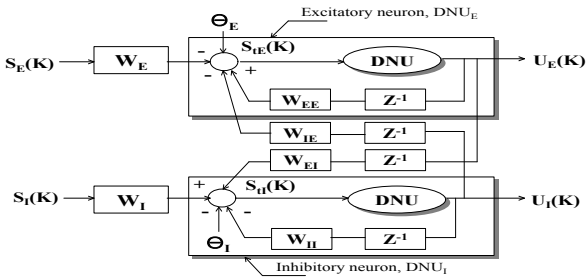

Fig. 2 Structure of MDNN

In this structure, $s_\lambda(k)$ and u_λ mean stimulus (input) and state reaction (output) of neural calculation unit when λ points to excitatory E or inhibitory I. $s_{t\lambda}(k)$ refers to total input of neural unit, while $w_{\lambda\lambda}$ points to interconnection strength of synapse from one neuron to another (as shown

by w_{IE}, w_{EI} in Fig. 2). The functional dynamics excited by DNU, a neural calculation unit, is defined as quadratic function as shown in formula (1). State variables $u_E(k+1)$ and $u_I(k+1)$ generated by the excitatory and inhibitory neural unit of the proposed neural processor in time(k+1) will be modelled as follows;

$$u_E(k+1) = E[u_E(k),\ v_E(k)],\quad \text{and}$$
$$u_I(k+1) = I[u_I(k),\ v_I(k)] \tag{7}$$

where $v_E(k)$ and $v_I(k)$ represent the rate of neuron in the neural unit in which larger input than the internal threshold is accepted, while E and I represent the operation of excitatory and inhibitory. The neuron which receives the input larger than the critical value is given as nonlinear function $v_\lambda(k)$, where the total input accompanied by the inhibitory neural unit will be as follows;

$$s_{uE}(k) = w_E s_E(k) + w_{EE} u_E(k-1) - w_{EE} u_I(k-1) - \Theta_E \tag{8}$$

$$s_{uI}(k) = w_I s_I(k) - w_{II} u_I(k-1) + w_{EI} u_E(k-1) - \Theta_I \tag{9}$$

where w_E and w_I are scaling factor of the excitatory and inhibitory neural unit each, while w_{EE} and w_{II} represent the linking strength of magnetic synapse, w_{IE} and w_{EI} that of mutual neuron synapse, and Θ_E and Θ_I the critical value of inhibitory neuron, respectively. Following formulas show the absolute refractory period (a period during which neuron can't be ignited newly) of excitatory and inhibitory neuron.

$$u_E(k+1) = u_E(k) + (1 - r_E u_E(k))\,\Psi_E[s_{uE}(k)] : \text{ excitatory neuron} \tag{10a}$$

$$u_I(k+1) = u_I(k) + (1 - r_I u_I(k))\,\Psi_I[s_{uI}(k)] : \text{ inhibitory neuron} \tag{10b}$$

According to the formulas (8) and (10), function for an isoclinic curve can be formulated as follows;

$$u_I(k) = \frac{1}{w_{IE}}\left[\left(w_E s_E(k) - \Theta_E + \Psi_E^{-1}\left[\frac{u_E(k)}{(1 - r_E u_E(k))}\right] + w_{EE} u_E(k)\right]\right) \text{for } u_E(k+1) = 0 \tag{11a}$$

$$u_E(k) = \frac{1}{w_{EI}}\left[\left(-w_I s_I(k) - \Theta_I + \Psi_I^{-1}\left[\frac{u_I(k)}{(1 - r_I u_I(k))}\right] + w_{II} u_I(k)\right]\right) \text{for } u_I(k+1) = 0 \tag{11b}$$

Thanks to the mathematical property of sigmoid function, Ψ_E and Ψ_I have the solitary value increasing simply within the range of $[-\infty, \infty]$. u_I, defined by the formulas (11a), is always the simple increment function of u_E. Whereas, u_E, by the (-) sign preceding Ψ_I^{-1} in the formula (11b), becomes the function of gradually decreasing u_{EI}. Such a qualitative difference between the two isoclinic angles is the immediate result of the asymmetry between excitatory and inhibitory. Based on the fact that the functional operation of neuron groups can be simulated by the nonlinear system theory, the response $u(k)$ of MDNN will be the multiples of individual response $u_\lambda(k)$ of excitatory and inhibitory in the neuron sub-group and is given as follows;

$$u(k) = u_E(k) + u_I(k) \tag{12}$$

where the total activities of neuron group refer to the sum of synapse response following excitatory and inhibitory.

3. Development of Learning Algorithm for MDNN

3.1 Learning Algorithm for MDNN Controller

In the learning procedures, the adaptation process of somatic gain is contained to minimize the weight value of forward and backward route as well as error function. By means of the repetition learning technique, the control sequence is transformed to generate the neuron output of $u(k)$ in order to reach the target status $u_d(k)$ at each repetition learning stage. In other words, the components of deviation $e(k)$ and parameter

vector $\Omega(a_{ff}, b_{fb}, g_s\ w_{\lambda\lambda})$ are changing together with each learning sequence k against the random set under the initial condition.

$$u(k) \rightarrow u_d(k) \quad as \quad k \rightarrow \infty \qquad or,$$
$$\lim_{k\rightarrow\infty}[\ u_d(k) - u(k) = e(K)] \rightarrow 0 \tag{13}$$

Solely the information set $\{(e(k-m),\ e(k),\ \Omega(a_{ff}, b_{fb}, g_s,\ w_{\lambda\lambda}(k)\}$ is required to find the solution of $\Omega(a_{ff}, b_{fb}, g_s\ w_{\lambda\lambda})(k+1)$, where m=1, 2, … and defines the size of constant. In line with the increased learning frequencies, information set is only reduced to $\{\Omega^*(a_{ff}, b_{fb}, g_s, w_{\lambda\lambda}(k), e^*(k)\}$, indicating the optimum convergence of DNU parameter and variance. The performance indicator which should be optimized against each parameter vector will be defined as follows, where E is expectation operator;

$$J = E\ \{F\ [e(k\ ;\ \Omega(\ a_{ff},\ b_{fb},\ g_s,\ w_{\lambda\lambda})]\} \tag{14}$$

In the formula (14), the general form of F [e(k; Ω (a_{ff}, b_{fb}, g_s, $w_{\lambda\lambda}$] is the symmetric function of variance, i.e.

$$J = \frac{1}{2} E\ \{[\ e^2(k\ ;\ \Omega(\ a_{ff},\ b_{fb},\ g_s,\ w_{\lambda\lambda})]\} \tag{15}$$

where E is an expectation operator and $e(k)$ is an error sign defined as difference between the target sign $u_d(k)$ and actual sign $u(k)$. Each component of vector $\Omega(a_{ff}, b_{fb}, g_s\ w_{\lambda\lambda})$ is applied in the way J is minimized by steepst-descent algorithm. In the steepst-descent method, parameter vector is arranged to be adjusted in proportion to the negative curve of J, that is;

$\delta\ \Omega(\ a_{ff},\ b_{fb},\ g_s\ w_{\lambda\lambda})(k) \propto (-\nabla J)$ *where*,

$$\nabla J = \frac{\delta J}{\delta\Omega(\ a_{ff},\ b_{fb},\ g_s,\ w_{\lambda\lambda})} \tag{16}$$

Hence, if dia [μ] is an independent adaptation gain matrix, the formula will be as follows;

$$\delta\ \Omega(\ a_{ff},\ b_{fb},\ g_s\ w_{\lambda\lambda}) =$$
$$-dia[\mu]\frac{\delta J}{\delta\Omega(\ a_{ff},\ b_{fb},\ g_s,\ w_{\lambda\lambda})} = -dia[\mu]\nabla J \tag{17}$$

In the above formula, dia [μ] is

$$dia[\mu] = \begin{bmatrix} \mu_{ai} & 0 & 0 & 0 \\ 0 & \mu_{bj} & 0 & 0 \\ 0 & 0 & \mu_{gs} & 0 \\ 0 & 0 & 0 & \mu_{\lambda\lambda'} \end{bmatrix} \tag{18}$$

where μ_{ai}, i=0, 1, 2 μ_{bi}, j=1, 2, μ_{gs} is the independent learning gain of DNU adaptation parameter and $w_{\lambda\lambda}$ represents the learning gain linking the magnetic and mutual neuron synapse. When synapse weight vector of DNU is described by $\emptyset(a_{ff}\ b_{fb})$, the tilt of performance indicator against $\emptyset(a_{ff}\ b_{fb})$ will be determined as following;

$$\frac{\delta J}{\delta\Phi(\ a_{ff},\ b_{fb})} = \frac{1}{2}\mathrm{E}\left[\frac{\delta[\ u_d(k) - u(k)]^2}{\delta\emptyset(\ a_{ff},\ b_{fb})}\right]$$
$$= E\left[e(k)\left\{-\frac{\delta\Psi(v)}{\delta\emptyset(\ a_{ff},\ b_{fb})}\frac{\delta v}{\delta\emptyset(\ a_{ff}\ b_{fb})}\right\}\right]$$
$$= E[\ e(k)\{\ \mathrm{sech}^2[v(k)]P\emptyset(\ a_{ff},\ b_{fb})\}] \tag{19}$$

where

$$P\emptyset(\ a_{ff},\ b_{fb})(k) = \frac{\delta v(k)}{\delta\emptyset(\ a_{ff},\ b_{fb})} = g_s\frac{\delta\ v_1(k)}{\delta\emptyset(\ a_{ff},\ b_{fb})}$$

representing the vector of parameter-status (or sensitivity) signal. [9-10]

$$P\emptyset(\ a_{ff}(k)) = g_s[S(k-i)],\quad i=0,1,2\ P\emptyset\ b_f\ b_i(k)$$
$$= -\ g_s[\ v_1(k-j)],\quad j=1,2 \tag{20}$$

In the similar way, the tilt of performance indicator for somatic gain g_s is determined by the following formula;

$$\frac{\delta J}{\delta\ g_s} = \frac{1}{2}E\left[\frac{\delta[\ u_d(k) - u(k)]^2}{\delta\ g_s}\right]$$
$$= E[-e(k)\{\mathrm{sech}^2[v(k)]v_1(k)\}] \tag{21}$$

The adaptation into the magnetic and mutual neuron synapse linkage can be attained as following;

$$\frac{\delta J}{\delta\ w_{\lambda\lambda'}} = \frac{1}{2}E\left[\frac{\delta[\ u_d(k) - u(k)]^2}{\delta\ w_{\lambda\lambda'}}\right]$$
$$= E[-e(k)\{\mathrm{sech}^2[v(k)]\ g_s\ u_\lambda(k-1)\}]$$
$$= E\left[-e(k)\left\{\frac{\delta\Psi(v)}{\delta v}\frac{\delta v}{d\ w_{\lambda\lambda'}}\right\}\right] \tag{22}$$

From the above formulas, the revised parameter algorithm of MDNN can be described as foll.;

$$a_{ffi}(k+1) = a_{ffi}(k) + \mu\, a_i E[e(k)\ \mathrm{sech}^2[v(k)]P\varnothing a_{ffi}(k)], \quad i=0,1,2 \tag{23a}$$

$$b_{fbi}(k+1) = b_{fbi}(k) + \mu\, b_i E[e(k)\ \mathrm{sech}^2[v(k)]] Pphj\, b_{fbi}(k)], \quad i=1,2 \tag{23b}$$

$$g_s(k+1) = g_s(k) + \mu\, g_s E[e(k)\ \mathrm{sech}^2[v(k)]\ v_1(k)] \tag{23c}$$

$$w_{\lambda\lambda'}(k+1) = w_{\lambda\lambda'}(k) + \mu_{\lambda\lambda'} E[-e(k)\ \mathrm{sech}^2][v(k)] g_{su}\lambda(k+1)] \tag{23d}$$

3.2 Control Technique of Nonlinear Dynamic Systems

In order to optimize the control status of nonlinear plant, MDNN controller should be able to recognize the point of early saturation of neural network and, accordingly, adjust the tilt of nonlinear activation function. It is assumed that the same parameter value is given for comparison of the existing DNU and MDNN controllers. Therefore, the initial weight value of MDNN is fixed at 0.8694, learning rate η at 0.0003, value of motion quantity term at 0.15 and dispersion time at 800. Based on these values, following 3 simulations are performed. [11-16]

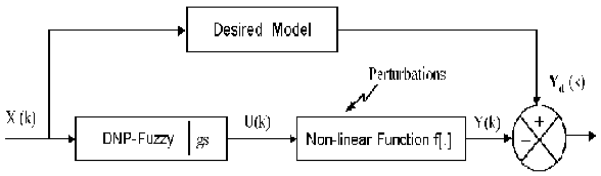

Fig. 3 A control modifier of nonlinear dynamic systems using MDNN algorithm.

3.3 Computer Simulation

Case 1. Unknown Nonlinear Model

Mathematical model of the plant to control is described by the equation with the degree as in the formula (24).

$$y(k) = f[y(k-1), y(k-2), u(k), u(k-1), u(k-2)] \tag{24}$$

where unknown nonlinear function $f[\cdot]$ is same as the formula (25).

$$f[\cdot] = [2 + \cos\{7\pi (y^2(k-1) + y^2(k-2)] + e^{-u(k)}/1 + u^2(k-1) + u^2(k-2) \tag{25}$$

where system input $x(k)$ = $\sin(2\pi k/250)$.

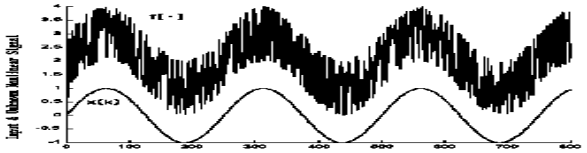

Fig. 4 System input x(k) and property of unknown nonlinear function F[·] of simulation.

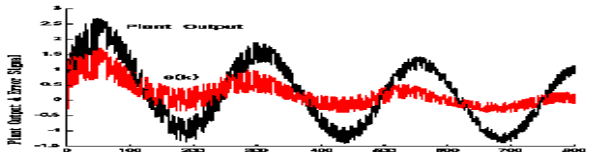

Fig. 5 Plant output and error response e(k) of existing DNU control in the 100th learning for simulation.

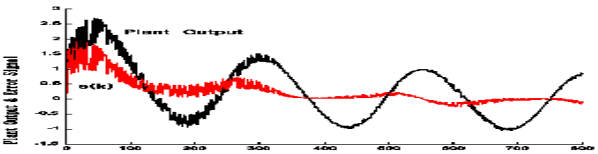

Fig. 6 Plant output and error reponse e(k) of existing MDNN control in the 100th learning for simulation.

Case 2. Plant Control by which unknown nonlinear property changes

Unknown nonlinear function $f[\cdot]$ is changed into 3 nonlinear functions during the control process as shown in the formulars (26), (27) and (28).

$$f[\cdot] = \frac{[\sin\{\pi(y^2(k-2)+0.5)\}] + 0.3\sin(2\pi u(k))}{1+u^2(k-1)+u^2(k-2)}$$

for $0 \le k \le 99$ and $500 \le k < 599$ (26)

$$f[\cdot] = e^{-(y^2(k-1)+y^2(k-2))} + \sqrt{|\{u^2(k)+u^2(k-1)+u^2(k-2)\}|}$$

for $200 \le 299$, $400 \le 499$ and $600 \le k < 800$ (27)

$$f[\cdot] = \frac{[0.5-0.5\cos\ddot{y}(7\pi\ (y^2(k-1)}{4+u^2(k-1)}$$

$$\frac{+\ y^2(k-2))\}]+e^{-u(k)}}{+\ u^2(k-2)}$$

for $100 \le k \le 199$ and $300 \le k < 399$

(28)

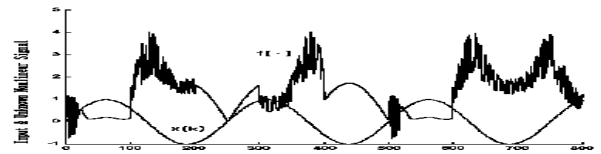

Fig. 7 System input x(k) and property of unknown nonlinear function f[·] of simulation.

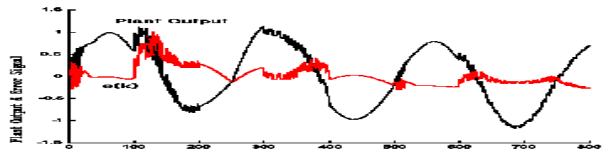

Fig. 8 Plant output and error e(k) of existing DNU control in the 100th learning for simulation.

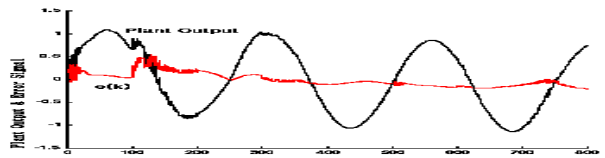

Fig. 9 plant output and error response e(k) of existing MDNN control in the 100th learning for simulation.

Case 3. Plant Control by which unknown nonlinear property changes

Plant and unknown nonlinear function $f[\cdot]$ are same each other as shown in example 1 and input signal $x(k)$ changes as in the formular (29).

$$\begin{aligned}
&x(k) = \cos(2\pi k/250) \quad 0 \le k < 75\\
&x(k) = 0.8 \quad \text{for } 76 \le k < 125\\
&x(k) = 0.6 \quad \text{for } 126 \le k < 175\\
&x(k) = 0.4 \quad \text{for } 176 \le k < 300\\
&x(k) = -0.6 \quad \text{for } 301 \le k < 350\\
&x(k) = 0.5\cos(2\pi k/150) \text{ for } 351 \le k < 600\\
&x(k) = \cos(2\pi k/250) \quad \text{for } k \ge 601
\end{aligned}$$

(29)

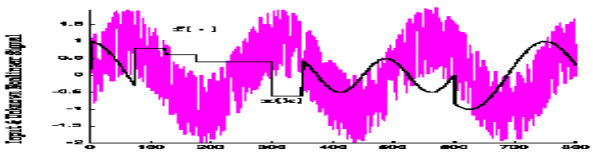

Fig. 10 System input x(k) and property nonlinear function f[·] of simulation.

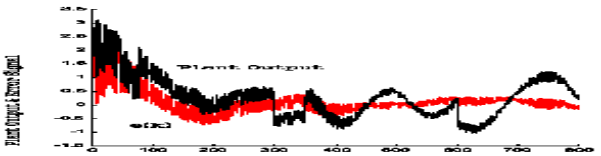

Fig. 11 Plant output and error response e(k) of existing DNU control in the 100th learning for simulation.

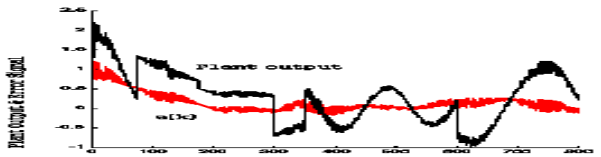

Fig. 12 Plant output and error response e(k) of existing MDNN control 50th learning for simulation.

4. Conclusion

It is found that MDNN controller has improved general convergence speed more than the DNU single controller in terms of dependability, strength, and adaptability in compliance with change of control environment factors such as changed basic input of plant, influence of disturbance, change of system parameter value and etc. In the words, nonlinear dynamic system of the learning algorithm in the single neuron network shows dependability and adaptability staring from the 100th learning, while the system control by neuron network fuzzy logic algorithm proposed in this study enables the dependability and adaptability to occur from the 50th learning onwards. Consequently, the latter demonstrater faster learning convergence and more improved control performance than the former and, by thus, that the plant output is better adapted to the input signal.

[Reference]

[1] K.s Narendra and K. Parthasarthy, "Indnetification and Control of Dynamical System Using Neural Network." IEEE Trans. Neural Network, pp.1-14, 1990.

[2] 조현섭, "A Simple and Effient Technique for Rapid Convergence Speed of The LMS Algorithm", ITC-CSCC'97(Japan,Okinawa),(Vol

1),pp31-34,199.7

[3] Madan M. Gupat, “Fuzzy Logic and Neural Networks”, Proceeding of the Ten International Conference(TAIPEI‘92), Vol 3, pp281-294,July,1992

[4] 조현섭, “A Study on Korean Digit Recognition Using Semi-Dnamic Neural Network With Sequential Feedback Architecture”, ICCT‘94(Shanghai), Vol1, pp1442-1444, 1994.6.8

[5] Robert Hecht-Nielsen, “Neural Computing”, HNC, Inc. and University of California San Diego, 1991

[6] D. Stipanicev, M. De Neyer, and R. Gorez, “Self-tuning Self-organizing Fuzzy Robot Control”, Proc., IFAC Symp. Robot Control SYROCO‘91, Vienna, Sept권., 1991

[7] M. M. Gupta, and D. H. Roa,, “Dynamic Neural Units with Applications to the Control of Unknown Nonlinear System”, Journal of Intelligent and Fuzzy Systems, Vol.1, pp73-92, 1993

[8] K. J. Astrom, “Adaptive Feedback Control”, In Proceedings of the IEEE, Vol.75, pp185-217, 1987

[9] M. D. Peek and P. J. Antsaklis, “Parameter Learning for Performance Adaptation”, IEEE Control Syst. Mag., Vol.10, No.3, 1990

[10] M. M. Gupta and D. H. Rao, “Dynamic Neural Unit in the Control of Linear and Nonlinear System”, In Proceeding of the International Joint Conf. on Neural Networks, pp100-105, June, 1992

[11] M. M. Gupta and D. H. Rao,, “Synaptic and Somatic Adaptations in Dynamic Neural Networks”, In Proceeding of the Second International Conf. on Neural Networks, pp173-177, 1992

[12] S. Chen, S. A Billings, and P. M. Grant, “Nonlinear System Identification using Neural Networks”, Int. J. Control, 51, 1990

[13] M. M. Gupta and G. K. Knopf, “A Multitask Visual Information Processor with a Biologically Motivated Design”, J. Visual Comm. Image Represent.,3, 230, 1992

[14] M. M. Gupta and Rao, "Neural Learning of Robot Inverse Kinematics Transformations", In Neural and Fuzzy System: The Emerging Science of Intelligent Computing, S. Mitra, W. kraske, and M. M. Ghpta, Eds. SPIE Institude Series Editor, Bellingham, Washington, 1995

[15] S. W. Park and B. H. Seo,“Design of Nonlinear System using Dynamic Neural Networks,", proc., KACC, pp60-64, March 1995

[16] L. H, Tsoukalas and R. E. Uhrig, "Fuzzy and Neural Approaces in Engineering", Jone Wiley & Sons, Ins., 1996