Fuzzy Gaussian Potential Neural Networks Using a Functional Reasoning

Mohammad Teshnehlab* and Keigo Watanabe**

*Graduate School of Science and Engineering
Saga University, 1 Honjo-machi, Saga 840, Japan
**Department of Mechanical Engineering
Saga University, 1 Honjo-machi, Saga 840, Japan

Abstract – This paper presents the principal design of a fuzzy gaussian potential neural network (FGPNN) to achieve high capability to learn expert control rules of the fuzzy controller. In this construction, each membership function consists of a gaussian potential function (GPF) which causes the utilization of a reduced number of labels, and eventually the complexity of structural design becomes simple, specially for large scale inputs. This in turn reduces the learning trials, to improve the learning speed. Thus, the time of the training process, which is based on the back-propagation method, is shortened. The construction of an FGPNN is carried out with the minimum number of GPF, based on the number of input patterns, to learn the mean vectors and shapes of the individual GPFs that basically depend on the desired trajectory. Finally, we provide a simulation to evaluate the proposed method for a multi input-output, two-link manipulator.

1. Introduction

The main idea of fuzzy sets is due to Zadeh [1]-[3]. The key idea is to develop a framework to deal with imprecision. Instead of using the ordinary concept of set inclusion, Zadeh introduced a function that expressed the degree of belonging to a given set as a function taking values in the range 0 to 1. In fuzzy logic the operation corresponds to the minimum or maximum values of the membership function. Recently, fuzzy control using the fuzzy reasoning has been estabilished as one of the control technologies. As an example, any control designer can realize the fuzzy controller with minimum information of the control object, compared with the conventional model-based controls, and it can be applied to any plant, irrespective of linearity or nonlinearity of the plant.

One of the problems with fuzzy controller is the need of more trials and errors compared to other designs of optimal controllers. Another problem is to scale the input data for the fuzzy support set and to scale the inferred consequent. Another problem is to pick up the tuning of the form of membership functions and to select the control rules precisely. The determination of the number of rules from the use of the number of fuzzy labels assigned to each input is a sophisticated procedure, and in this way an effective fuzzy controller is difficult to design, because the number of control rules exponentially increases with the increase in the number of input data.

To overcome the control rule problem, self-organizing fuzzy controllers have been proposed by different authors. In this approach, to evaluate the control

performance, control rules are modified by using the fuzzy reasoning. Specially, a new idea which is a combination of neural network (NN) theory and the fuzzy reasoning sounds more attractive. Such fuzzy neural networks (FNNs) [4]-[8] are applied to design controllers. In some sense the FNN automatically tunes the fuzzy rules and membership functions by using the learning function of neurons. Most of the structures of NN developed to date have focused on the update of learning parameters for a fixed NN structure.

However, if an NN has flexible function to achieve high capability in learning algorithm, then such an NN may becomes optimal network, in the sense of minimizing the control or modelling performance error.

This article presents a fuzzy gaussian potential neural network (FGPNN) to minimize the number of labels in the antecedent part of a fuzzy reasoning, which turns out to decrease the number of control rules. Thus, the proposed method becomes more effective for multi input-output systems to reduce the complexity of the architecture and hence the learning process will be improved.

This paper is organized as follows. Section 2 explains the functional reasoning. Section 3 describes the construction of membership functions in gaussian types. Section 4 explains the construction of fuzzy neural network and also describes the design of conclusion part. Section 5 describes the learning mechnisms of FGPNN and introduces the adaptation method of input scalers. Section 6 gives the dynamic equations of two-link manipulator. Section 7 gives the simulation example to evaluate the proposed FGPNN and finally the conclusion follows in Section 8.

2. Funtional Reasoning Theory

The conventional functional reasoning or its modified version has been applied to many problems. For n input variables $(x_1, .., x_n)$ and p output variables $(u_1, .., u_p)$ as the consequent, any i-th control rule can be given by

$$\begin{aligned} \mathrm{R}_i : \text{If } & x_1 = A_{i1} \text{ and } \cdots \text{ and } x_n = A_{in} \\ & \text{then } u_1 = f_{i1}(x_1, ..., x_n) \text{ and } \cdots \\ & \text{and } u_p = f_{ip}(x_1, ..., x_n) \end{aligned} \tag{1}$$

where R_i is i-th control rule; A_{ij} is the fuzzy set in the antecedent associated with the j-th input variable at the i-th fuzzy rule, $f_{ik}(x_1, .., x_n)$ is the function associated with the k-th variable in the conclusion at the i-th control rule. By applying n confidences, $\mu_{Ai1}(x_1), .., \mu_{Ain}(x_n)$, the confidence in the antecedent h_i is defined by

$$h_i = \mu_{Ai1}(x_1) \cdot \mu_{Ai2}(x_2), ..., \cdot \mu_{Ain}(x_n) \tag{2}$$

where "." denotes the algebraic product. Therefore, the k-th output consequent can either be derived as the weighted mean of $f_{ik}(.)$ with respect to weight h_i:

$$u_k^* = \frac{\sum_{i=1}^{r} h_i f_{ik}}{\sum_{i=1}^{r} h_i}, \quad k = 1, ..., p \tag{3}$$

or the weighted sum for rule's output

$$u_k^* = \sum_{i=1}^{r} h_i f_{ik} \tag{4}$$

where r in both equations (3) and (4) is the number of fuzzy rules; if the number of membership functions in the antecedent is l, then in general $r = l^n$.

3. Construction of Membership Function

3.1 The Gaussian Type Membership Function

Horikawa et al. [4] proposed the FSNN (fuzzy sigmoid neural network) for a case to provide a pseudo-trapezodial membership function, in which we must combine two sigmoid unit functions with the ranges of $[0, 1]$ and of $[-1, 0]$. The difficulty is arised on construction method even for single input output system and small number of labelling. Watanabe et al. [7] proposed FGNN (fuzzy gaussian neural network) which is much simpler than FSNN of Horikawa et al. [4]. Also, Ichihashi [8] used similar gaussian membership function, and proposed some block hierarchical fuzzy models to reduce the number of parameters when the number of input becomes very large.

3.2 Construction of Gaussian Potential Membership Function

The gaussian potential function (GPF) used in this study consists of the squared norm of difference of input and mean value vectors . In this method, no matter when the number of inputs increases the number of fuzzy rules does not increse, but in the case of Watanabe et al. [7] the number of fuzzy rules grows with the increase of the number of inputs, thus, the structures of Watanabe et al. [7] and Horikawa et al. [4] become very complicate for the case of multi input-output systems. The proposed method with GPF turns out to become more flexible than the traditional studies by other researchers. Now, let the input data vector $\boldsymbol{x} \triangleq [x_1, ..., x_n]^T$ be decomposed into some subdata vectors, e.g., $\boldsymbol{x}_1 \triangleq [x_1, ..., x_m]$ and $\boldsymbol{x}_2 \triangleq [x_{m+1}, ..., x_n]^T$. The gaussian potential function for the vector data $\boldsymbol{x}_1$ can be witten by

$$F(\boldsymbol{x}_1) = \exp\{-\frac{1}{2}\|\boldsymbol{x}_1 - \bar{\boldsymbol{x}}_1\|^2_{\Sigma_1^{-1}}\} \tag{5}$$

where $\bar{\boldsymbol{x}}_1$ is the mean value vector of $\boldsymbol{x}_1$ and Σ_1 is the associated covariance matrix. Figure 1 shows the construction of the membership function using a neural network. In this figure, the variable with the curled bracket denotes a signal passing through the neural network, w_s is the scaler for the scalar input variable x_i, and the connection weights w_c and w_d denote the center value and the reciprocal value of the standard deviation for a gaussian function on the standardized support set.

In addition, the unit with symbol -1 generates the output of -1, the unit with the symbol Σ outputs the summation of the inputs, and the input-output relation at the unit with the symbol f is defined by

$$f(x) = x^2 \tag{6}$$

Furthermore, the unit with symbols Σ and s outputs the signal through the function s represented as

$$s(x) = e^{\ln(0.5).x} \tag{7}$$

by summing the input signals to the unit. Consequently, by using such a neural network, a GPF may be generated such as

$$s(x_1, x_2) = e^{\ln(0.5)\Sigma_{i=1}^{2}[w_{si}x_i - w_{ci}]^2 w_{di}^2} \tag{8}$$

where a case with $w_{s1} = w_{s2} = 1, w_{c1} = \bar{x}_1, w_{c2} = \bar{x}_2, w_{d1} = \frac{1}{\sigma_1}$ and $w_{d2} = \frac{1}{\sigma_2}$ is shown in Fig. 1.

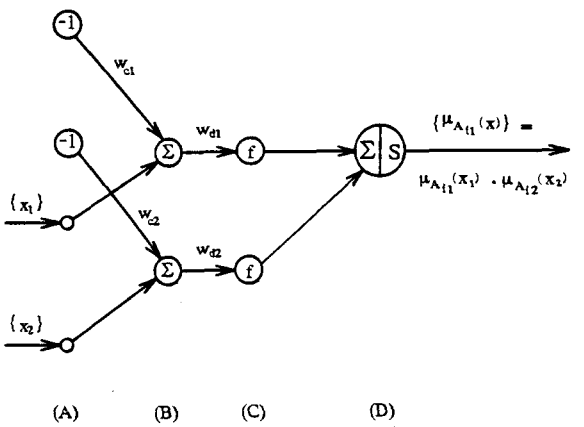

Fig. 1 Fuzzy gaussian potential membership function realized by a neural network

Using this representation gives the i-th gaussian potential membership function for the data x_1 such as

$$\mu_{Ai1}(x_1) = \exp\{-\frac{1}{2}\|x_1 - \bar{x}_1^i\|^2_{\Sigma_1^{i-1}}\} \tag{9}$$

where $\bar{x}_1^i$ denotes the mean value vector of gaussian potential membership function at the i-th fuzzy rule and Σ_1^i is the associated covariance matrix. Note here that when $x_1 = \{x_1, x_2\}$ equation (9) can be rewritten by

$$\mu_{Ai1}(x_1) = \mu_{Ai1}(x_1)\mu_{Ai2}(x_2) \tag{10}$$

if scalar data x_1 and x_2 are each independent, where

$$\mu_{Ai1}(x_1) = \exp\left[-\frac{1}{2}\frac{(x_1 - \bar{x}_1^i)^2}{\sigma_{i1}^2}\right] \tag{11}$$

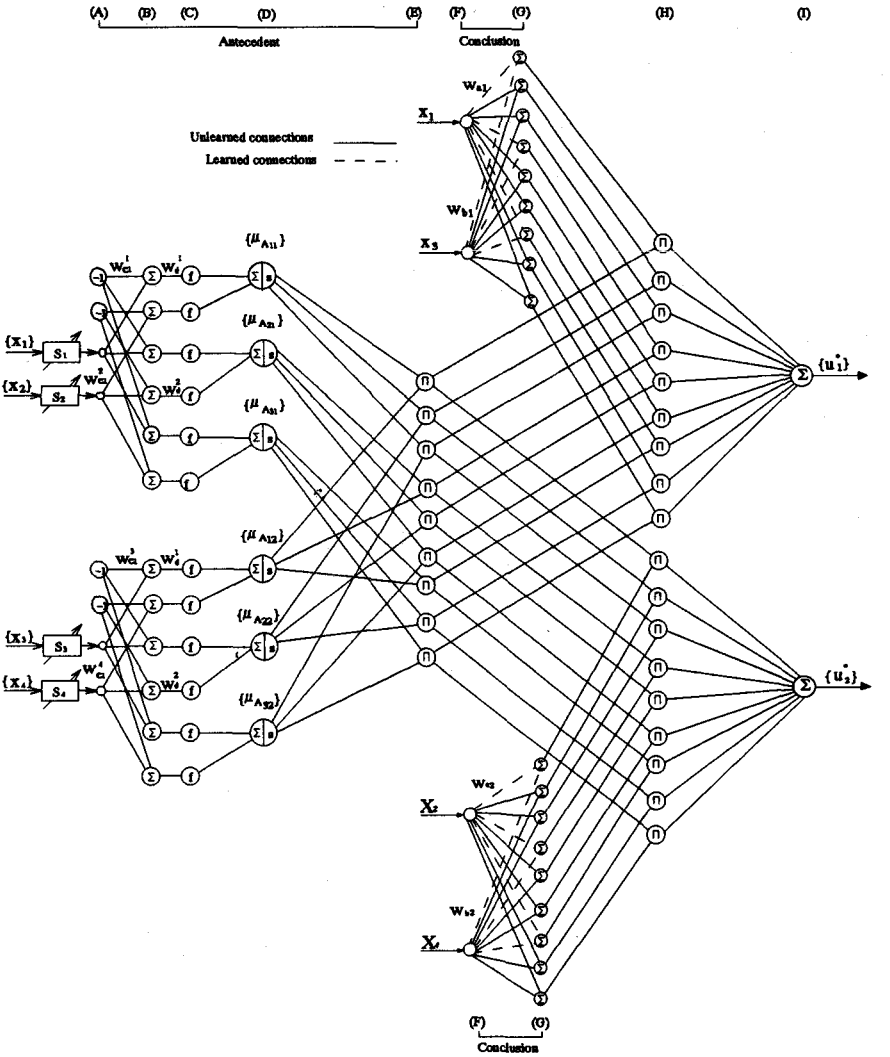

Fig. 2 Fuzzy gaussian potential neural networks based on functional reasoning

$$\mu_{Ai2}(x_2) = \exp\left[-\frac{1}{2}\frac{(x_2 - \bar{x}_2^i)^2}{\sigma_{i2}^2}\right] \tag{12}$$

and

$$\Sigma_1^i = \mathrm{diag}(\sigma_{i1}^2, \sigma_{i2}^2). \tag{13}$$

Thus, in this approach the number of fuzzy rules does not increase even if the number of inputs becomes large, because the fuzzy labeling is made on the vector data. On the other hand, in the traditional approaches [4], [7] the number of fuzzy rules exponentially grows as the number of inputs increases, because the fuzzy labeling is made on the each scalar data. Hence, the proposed method gives a significant simplification in the FNN structure.

4. Construction of Fuzzy Gaussian Potential Neural Networks

Figure 2 illustrates the design example of an FGPNN for a case of four inputs $(x_1, ..., x_4)$ and two outputs (u_1^*, u_2^*) with three labels in the antecedent part. According to the number of labels, the number of the identifiable control rules is $r = 3^2$ if two inputs vector data are defined as $\boldsymbol{x}_1 = (x_1, x_2)$ and $\boldsymbol{x}_2 = (x_3, x_4)$. It should be noted that the number of identifiable control rules is $r = 3^4 = 81$ if the fuzzy labeling is made on each input data $x_1, ..., x_4$. Layers A$\sim$E correspond to the antecedent part, and layers F and G correspond to the conclusion part. The fuzzy consequent is obtained at layers H and I. At the layer C, the function f of unit is represented by (6). At the layer D, the function s of unit is written by (7). Note here that the mean values $\bar{x}_1^i, .., \bar{x}_4^i$ on the data $(x_1, ..., x_4)$ are realized as connection weights between layers A and B, and the reciprocal values $1/\sigma_{i1}, ..., 1/\sigma_{i4}$ on the deviations $(\sigma_{i1}, ..., \sigma_{i4})$ are also realized as connection weights between layers B and C.

Now, consider a case of linear function with subvector data $\boldsymbol{x}_1 = (x_1, x_2)$ and $\boldsymbol{x}_2 = (x_3, x_4)$ in (1). As usual applications, we suppose the tracking control problem under the condition of $x_1 = e_1$, $x_2 = e_2$, $x_3 = \dot{e}_1$ and $x_4 = \dot{e}_2$, where e_1 and e_2 denote the tracking errors of plant output 1 and 2, respectively, and $\dot{e}_1$ and $\dot{e}_2$ denote their rates of errors. If we use PD-type controllers as the conclusion functions f_{ij}, $j = 1$ and 2, it follows that

$$f_{i1} = [K_{pi}^1 \ \vdots \ 0 \ \vdots \ K_{di}^1 \ \vdots \ 0] \begin{bmatrix} e_1 \\ e_2 \\ \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} \tag{14}$$

$$f_{i2} = [0 \ \vdots \ K_{pi}^2 \ \vdots \ 0 \ \vdots \ K_{di}^2] \begin{bmatrix} e_1 \\ e_2 \\ \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} \tag{15}$$

for any i-th rule. Here, gains K_{pi}^1, K_{di}^1, K_{pi}^2 and K_{di}^2 can be realized as connection weight parameters in the conclusion part of an FNN to be learned.

5. Learning Mechanisms

The interesting point is to construct an FGPNN that gives a desired level of performence, even though the network structure does not have enough units. We describe the so-called back propagation algorithm, widely used algorithm for training the multilayered hierarchical neural networks. Here we consider the multilayered hierarchical neural network consisting of M layers. Let us denote the input-output relation of any unit by $f(\cdot)$, the input to the j-th unit at the k-th layer by i_j^k, and the corresponding output from its unit by o_j^k. The weight that connects the j-th unit at the k-th layer and the l-th unit at the$(k+1)$-th layer is written by $w_{j,l}^{k,k+1}$. In addition, Case A denotes a case when the input to the k-th layer is output through the function $f(\cdot)$ and the input to the $(k+1)$-th layer is calculated by the summation (i.e., Σ) operation. Similarly, Case B denotes a case when the input to the k-th layer is output through the function $f(\cdot)$ and the input to the $(k+1)$-th layer is calculated by the product (i.e., Π) operation. Note here that the output layer is assumed to consist of Case A. Under these preparations, we have the following input-output relation of a unit:

$$i_l^{k+1} = \sum_j w_{jl}^{k,k+1} o_j^k, \quad o_l^{k+1} = f(i_l^{k+1}) \tag{16}$$

for Case A and

$$i_l^{k+1} = \prod_j w_{jl}^{k,k+1} o_j^k, \quad o_l^{k+1} = f(i_l^{k+1}) \tag{17}$$

for Case B.

Now, provided a set of the input-output pattern on a plant, let us denote the teaching signal to the i-th unit of the output layer by t_i and the output from the neural network to which the input pattern is fed by o_i^M.

5.1 Generalized Learning

For the generalized learning architecture, we consider the following cost performance:

$$J = \frac{1}{2}\sum_{i=1}^{L}(t_i - o_i^M)^2 \tag{18}$$

so that the weights $w_{ij}^{k,k+1}$ minimize the above J. Here, L is the unit number of the output layer. Following a gradient descent algorithm, the increment of $w_{ij}^{k,k+1}$ denoted by $\Delta w_{ij}^{k,k+1}$ becomes

$$\Delta w_{ij}^{k,k+1} = -\eta \frac{\partial J}{\partial w_{ij}^{k,k+1}} \tag{19}$$

where $\eta > 0$ is a learning rate given by a small positive constant.

At the output layer M, since

$$\frac{\partial J}{\partial w_{ij}^{M-1,M}} = \frac{\partial J}{\partial i_j^M}\frac{\partial i_j^M}{\partial w_{ij}^{M-1,M}} = \frac{\partial J}{\partial i_j^M} o_i^{M-1} \tag{20}$$

if defining

$$\delta_j^M = -\frac{\partial J}{\partial i_j^M} \tag{21}$$

gives

$$\delta_j^M = -\frac{\partial J}{\partial o_j^M}\frac{\partial o_j^M}{\partial i_j^M} = (t_j - o_j^M) f'(i_j^M) \tag{22}$$

where $f'(i_j^M) = df(i_j^M)/di_j^M$. Therefore, the increment of the weight at the output layer is obtained by

$$\Delta w_{ij}^{M-1,M} = \eta \delta_j^M o_i^{M-1} \tag{23}$$

On the other hand, at any two intermediate layers k and $k+1$, we have

$$\frac{\partial J}{\partial w_{ij}^{k-1,k}} = \frac{\partial J}{\partial i_j^k}\frac{\partial i_j^k}{\partial w_{ij}^{k-1,k}} = \frac{\partial J}{\partial i_j^k} o_i^{k-1} \tag{24}$$

Defining

$$\delta_j^k = -\frac{\partial J}{\partial i_j^k} = -\frac{\partial J}{\partial o_j^k}\frac{\partial o_j^k}{\partial i_j^k} = -\frac{\partial J}{\partial o_j^k} f'(i_j^k) \tag{25}$$

yields

$$\frac{\partial J}{\partial o_j^k} = \sum_l \frac{\partial J}{\partial i_l^{k+1}}\frac{\partial i_l^{k+1}}{\partial o_j^k} = -\sum_l \delta_l^{k+1} w_{jl}^{k,k+1} \tag{26}$$

for Case A, and

$$\frac{\partial J}{\partial o_j^k} = \sum_l \frac{\partial J}{\partial i_l^{k+1}}\frac{\partial i_l^{k+1}}{\partial o_j^k} = -\sum_l \delta_l^{k+1} w_{jl}^{k,k+1} \left(\prod_{i \neq j} w_{il}^{k,k+1} o_i^k\right) \tag{27}$$

for Case B. Therefore, it is found that

$$\delta_j^k = f'(i_j^k) \sum_l \delta_l^{k+1} w_{jl}^{k,k+1} \text{ for case A} \tag{28}$$

$$\delta_j^k = f'(i_j^k) \sum_l \delta_l^{k+1} w_{jl}^{k,k+1} \left(\prod_{i \neq j} w_{il}^{k,k+1} o_i^k\right) \text{ for case B} \tag{29}$$

In addition, we have the increment of the weight at the intermediate layer such as:

$$\Delta w_{ij}^{k-1,k} = \eta \delta_j^k o_i^{k-1} \text{ for case A} \tag{30}$$

$$\Delta w_{ij}^{k-1,k} = \eta \delta_j^k o_i^{k-1} \left(\prod_{l \neq i} w_{lj}^{k-1,k} o_l^{k-1}\right) \text{ for case B} \tag{31}$$

Note here that, for the feedback error learning, if the j-th feedback input is represented by u_{fj}, then δ_j^M in (22) is replaced by

$$\delta_j^M = u_{fj} f'(i_j^M) \tag{32}$$

5.2 Specialized Learning

Furthermore, for specialized learning, we consider the following cost performance:

$$J = \frac{1}{2}\sum_{i=1}^{m}(y_{di} - y_i)^2 \tag{33}$$

to obtain the weights $w_{ij}^{k,k+1}$ that minimize J. Here, m denotes the number of the plant outputs, y_{di} the i-th desired reference and y_i the i-th output of the plant. Then, a further deforming of (21) gives the delta quantity at the output layer for a case of multi input-output with coupling input, described by

$$\begin{aligned}
\delta_j^M &= -\frac{\partial J}{\partial i_j^M} \\
&= -\frac{\partial J}{\partial o_j^M}\frac{\partial o_j^M}{\partial i_j^M} \\
&= \sum_{i=1}^{m} -\frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial o_j^M} f'(i_j^M) \\
&= f'(i_j^M)\sum_{i=1}^{m}(y_{di} - y_i)\frac{\partial y_i}{\partial u_j} \quad ,
\end{aligned} \tag{34}$$

where the $\partial y_i / \partial u_j$ is defined as

$$\frac{dy_i}{du_j} = \frac{\partial y_i}{\partial u_j} + \sum_{l \neq j}\frac{\partial u_i}{\partial u_l}\frac{du_l}{du_j} \tag{35}$$

$$\frac{\Delta y_i}{\Delta u_j} \simeq \frac{\partial y_i}{\partial u_j} + \sum_{l \neq j}\frac{\partial y_i}{\partial u_l}\frac{\Delta u_l}{\Delta u_j} \tag{36}$$

also u_j denotes the j-th input to the plant and the $f'(i_j^M)$ is equal one. Finally, the update equation of learning parameters are described by

$$w_{ij}^{k-k,k}(t+1) = w_{ij}^{k-1,k}(t) + \eta\delta_j^k o_i^{k-1} + \alpha\Delta w_{ij}^{k-1,k}(t) \tag{37}$$

for adjusting the shape of gaussian potential function in the antecedent and connection weights in the conclusion part, where α denotes the momentum coefficient with the range in $0 \leq \alpha < 1$

Table 1 Actual values of manipulator parameters

	Link one	Link two
Mass m_i[kg]	5.0	5.0
Inertia I_i[kg m^2]	0.104	0.104
Length l_i[m]	0.5	0.5
Length l_{gi}[m]	0.25	0.25

5.3 Adaptation of Input Scalers

Generally, the constant input scalers can be easily determined from the maximum value of the data obtained through the control experiments. But most of the times the input data fail to transform into the predetermined support set, because of the addition of the fuzzy compensator to the control system. To overcome this problem, we use a simple adaptive method for scaling the input data $x_1, ..., x_n$, as described by

$$w_{s,new} = \begin{cases} \dfrac{0.9 \times L}{|x_i|} & \text{if} \quad |w_{s,old} \times x_i| > L \\ w_{s,old} & \text{otherwise} \end{cases} \tag{38}$$

which means that if the scaled input data are scaled out from the support set $[-L, L]$, then the input data are rescaled to fall into the 90% range of the support set.

6. Manipulator Model

There has recently been a considerable interest in developing efficient control algorithms for robot manipulators. The complexity of the control problem for manipulators arises mainly from that of manipulator dynamics itself. The dynamics of articulated mechanisms in general, and of robot manipulators in particular, involve strong coupling effects between joints as well as centrifugal and Coriolis forces. The equations of motion for 2-DOF planar manipulator can be written in the compact form as

$$M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) = \tau \tag{39}$$

where $\tau \in R^2$ is a vector of joint torques supplied by the actuators, $\tau = [\tau_1 \quad \tau_2]^T$ and $\theta \in R^2$ is a vector of joint positions with $\theta = [\theta_1 \quad \theta_2]^T$, and the $M(\theta) \in R^{2\times 2}$ is called manipulator inertia mass matrix, whose elements are obtained by

$$M_{11} = I_1 + I_2 + m_1 l_{g1}^2 + m_2[l_{n1}^2 + l_{g2}^2 + 2l_{n1}l_{g2}\cos(\theta_2)] \tag{40}$$

$$M_{12} = M_{21} = I_2 + m_2[l_{g2}^2 + l_{n1}l_{g2}\cos(\theta_2)] \tag{41}$$

$$M_{22} = I_2 + m_2 l_{g2}^2 \tag{42}$$

Here I_j is the moment of inertia for the j-th link, l_{gj} is the distance from the joint j to the center of gravity of the j-th link. The actual values for the parameters of the manipulator are given in Table 1.

The vector $V(\theta, \dot{\theta}) \in R^2$ represents forces arising from the Coriolis force and centrifugal force which are expressed as:

$$V_1 = -m_2 l_{n1} l_{g2} \sin(\theta_2)(2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \tag{43}$$

$$V_2 = m_2 l_{n1} l_{g2} \sin(\theta_2)\dot{\theta}_1^2 \tag{44}$$

7. Simulation Example

We use the 4th-order Runge-Kutta-Gill method to simulate the plant dynamics state. It is assumed that the control sampling period is $T = 10$ [ms], the step width of the integration is 0.4 [ms], the initial states are $\theta_1 = 0.5236$ [rad] and $\theta_2 = 0.4363$ [rad], and $\dot{\theta}_1 = \dot{\theta}_2 = 0$. Also, the desired trajectories of the manipulator are assumed to be known as time functions of joint positions, velocities, and accelerations, that is, θ_d, $\dot{\theta}_d$, and $\ddot{\theta}_d$ are expressed as

$$\theta_{d1} = 0.5\cos(\pi t) \quad \theta_{d2} = 0.5\sin(\pi t) + 1.0 \tag{45}$$

$$\dot{\theta}_{d1} = -0.5\pi\sin(\pi t) \quad \dot{\theta}_{d2} = 0.5\pi\cos(\pi t) \tag{46}$$

$$\ddot{\theta}_{d1} = -0.5\pi^2\cos(\pi t) \quad \ddot{\theta}_{d2} = -0.5\pi^2\sin(\pi t) \tag{47}$$

The learning algorithms of both connection weights in the antecedent and conclusion parts of FGPNN are implemented. In this study five lables are applied to each input subvector as indicated in Fig. 3. The center position values, w_c, are $-4, -2, 0, 2, 4$, and the reciprocal values of standard deviation w_d are all 1, in accordance with all of the labels on the support set $[-4, 4]$. The initial values of the connection weights (i.e., PD gain parameters) in the conclusion parts are given in Table 2.

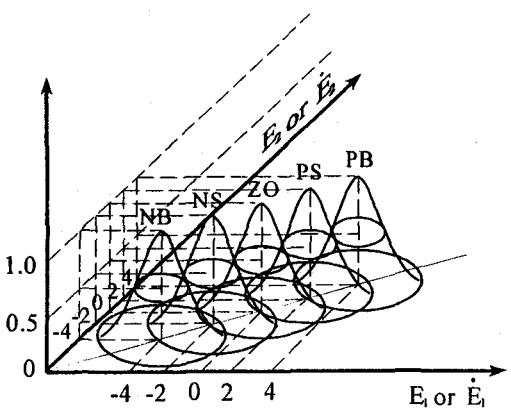

Fig. 3 The gaussian potential membership function with 5 labels on the support set [-4, 4]

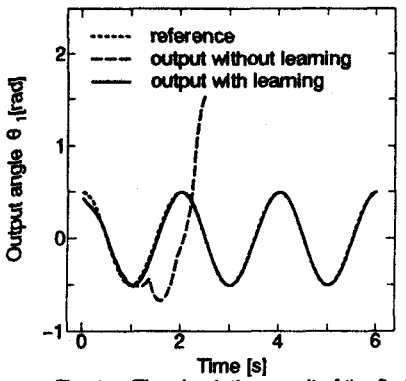

Fig. 4–a The simulation result of the first link using proposed FGPNN controller

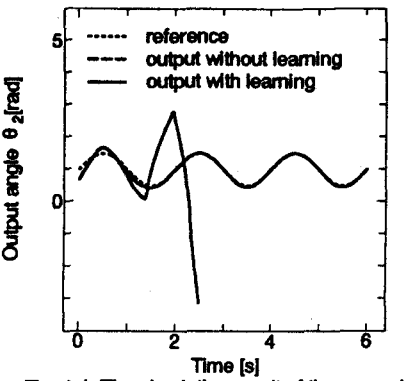

Fig. 4–b The simulation result of the second link using proposed FGPNN controller

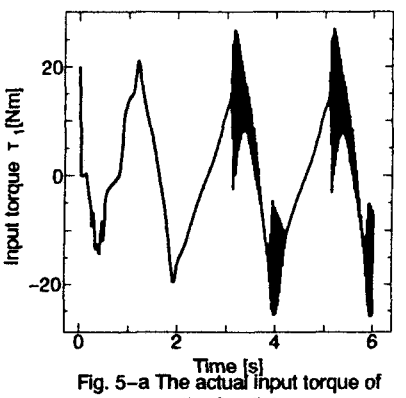

Fig. 5–a The actual input torque of the first link

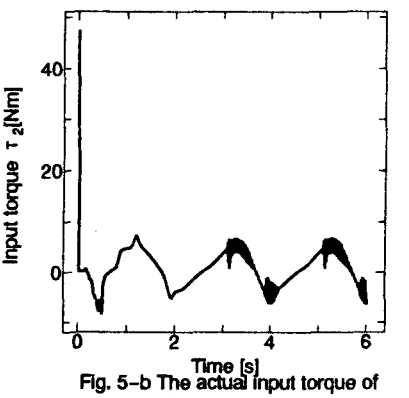

Fig. 5–b The actual input torque of the second link

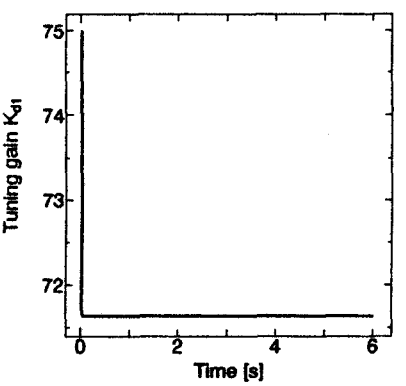

Fig. 6–b Tuning d–gain of the first link

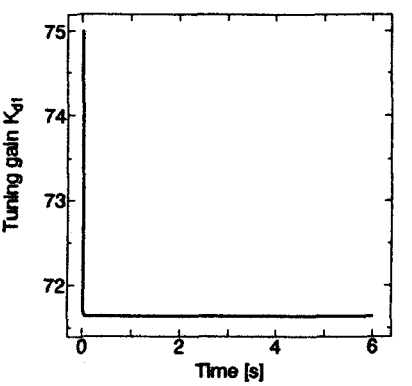

Fig. 6–b Tuning d–gain of the first link

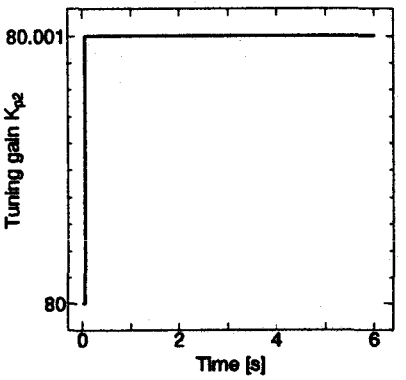

Fig. 6–c Tuning p–gain of the second link

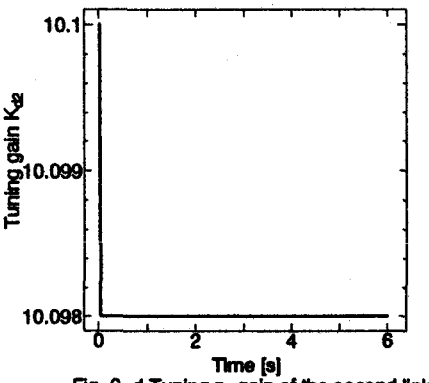

Fig. 6–d Tuning p–gain of the second link

The control results are shown in Figs 4-a and 4-b. These figures indicate that the proposed FGPNN controller shows high ability for the multi/input-output system and the plant outputs try to follow the desired trajectories which can be counted for the high quality of performance in this method. Also, Figs. 5-a and 5-b show the actual control inputs of two-link manipulator. Figures 6-a to 6-d show the time history of PD learning gains in the conclusion parts. In this study, those connection weights are indicated in Fig. 2, used in learning process and the results are replaced by other connection weights which are not learned. Also, to investigate the effectiveness of learning parameters in the antecedent and conclusion parts, the simulations with learning parameters and without learning parameters are compared. Totally, from simulation results of Tables 3-a and 3-b, and figures 6-a to 6-b, it is found that the control was mainly improved by using the learning of the conclusions, because the PD gains in conclusion parts especially for the first link gave a wide range of variations, but the parameters in the antecedent parts made slight changes.

8. Conclusion

In this paper, an FGPNN has been proposed especially for designing a learning fuzzy controller. The proposed method can be utilized for large scale systems to simplify the strcuture of fuzzy-neural network configuration by using the concept of GPFs. The input and its mean were considered to be a vector form, thus, the membership functions formed a potential form. The learning method was based on modifying the back-propagation algorithms for the purpose of on-line learning (the so-called specialized learning method). In this approach, both the connection weights in the antecedent and conclusion parts of FGPNN could be learned. The method was applied to construct a controller for robotic manipulators. The simulation example was also given to demonstrate the validity of the proposed method. It was shown that the suggested FGPNN controller gave a very good error minimization and a quick convergence to the desired trajectories for multi input-output systems.

REFERENCES

[1] L. A. Zadeh,"Fuzzy sets," *Information and Control*, Vol. 8, pp. 330-353, 1965.

[2] L. A. Zadeh,"Fuzzy algorithms," *Information and Control*, Vol. 12, pp. 94-102, 1968.

[3] L. A. Zadeh,"Common sense knowledge representation based on fuzzy logic ," *IEEE Computer*, Vol. 10, pp. 61-65, 1983.

[4] S. Horikawa, T. Furuhashi, and Y. Uchikawa,"On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. on Neural Networks*, Vol.3, No.5, pp.801-806, 1992.

[5] J.-S. R. Jang and C.-T. Sun,"Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," *IEEE Trans. on Neural Networks*. Vol.4, No. 1, pp. 156-158, 1993.

[6] J.-S. R. Jang,"Fuzzy modeling using generalized neural networks and Kalman filter algorithm," *Proceedings of Ninth National Conference on Artificial Intelligence*, pp. 762-767, 1991.

[7] K. Watanabe, J. Tang, M. Nakamura, S. Koga, and T. Fukuda,"A Mobile Robot Control using Fuzzy-Gaussian Neural Networks," *Proceedings of IROS' 93*, Yokohama, Japan, 26-30 July 1993, Vol.2, pp.919-925.

[8] H. Ichihashi,"Learning in Hierarchical Fuzzy Models by Conjugate Gradient Method using Back-propagation Errors,"*Procs. of 1st FAN Symposium*, Oct., 25 and 26, Osaka, pp.235-240, 1991.