ปรับปรุงตัวควบคุมแบบฟัซซีลอจิกชนิดไม่ปรับตัวด้วยนิวรอลเนตเวอร์คแบบป้อนไปข้างหน้า: ตัวควบคุมแบบนิวรอล-ฟัซซีแบบป้อนไปข้างหน้า

Improve Static Fuzzy Logic Controllers with Feedforward Neural Networks: Neural-Fuzzy Feedforword Controller

PINIT NGAMSOM

Dept. of Mechanical and Industrial Engineering
Rangsit University
Lakhog, Patumtani 12000
THAILAND

นิวรอลเนตเวอร์คแบบป้อนไปข้างหน้าถูกนำมาใช้ร่วมกับตัวควบคุมแบบฟัซซีลอจิกชนิดไม่ปรับตัวเพื่อใช้ในการแก้ปัญหาต่างๆจากความไม่เป็นเชิงเส้นและความไม่แน่นอนในระบบควบคุม ความสามารถในการเรียนรู้และการปรับตัวโดยไม่ต้องมีการดูแลของนิวรอลเนตเวอร์คแบบป้อนไปข้างหน้าทำให้การเลือกตัวแปรทางฟัซซีที่ให้การตอบสนองที่น่าพอใจเป็นไปได้ง่ายขึ้น ตัวควบคุมที่เสนอนี้ถูกนำไปทดสอบกับระบบของ เฟือง-มอเตอร์-ดาน ที่มีการเปลี่ยนแปลงของภาระและความไม่แน่นอน การเลียนแบบระบบด้วยคอมพิวเตอร์แสดงให้เห็นว่าแนวทางการควบคุมที่ไม่ต้องอาศัยแบบจำลองทางคณิตศาสตร์นี้มีประโยชน์อย่างยิ่งในการจัดการเกี่ยวกับความไม่เป็นเชิงเส้นและความไม่แน่นอนที่พบได้ในระบบทางฟิสิกส์โดยมาก

The feedforward neural network is incorporated with a static fuzzy logic controller to solve problems of nonlinearity and uncertainty in control systems. The unsupervised-learning and adaptation ability of the feedforward neural network considerably eases the task of selecting fuzzy parameters that give satisfactory system responses. The proposed controller is tested with the system of gear-motor-pendulum subjected to loading variation and uncertainty. The computer simulations show that this model-free approach is of great value in dealing with nonlinearity and uncertainty found in most physical systems.

1. Introduction

Conventional and modern control techniques such as PD, PID or state-space design require that the designers must be able to express system information and control schemes mathematically. However, these mathematical models are difficult, or even impossible in some cases, to be reasonably obtained if the systems of interest are very complex. In addition to that, even though the models can be found, these model-based techniques are not powerful enough to handle nonlinear-uncertain systems efficiently over wide ranges of operation as evident in control systems literatures [6, 11]. For this kind of systems, intelligent control techniques such as neural networks and/or fuzzy logic show very satisfactory performances [10, 12]. This paper presents a possible way to combine advantages over the above analytical schemes of neural networks and those of fuzzy logic together.

A static fuzzy logic controller (i.e., fuzzy logic controller without adaptation algorithms) cannot deal with systems of high degree of uncertainty and nonlinearity. In fact, one can consider it as a way to perform the required *static*-nonlinear mappings from input domains to control actions by using knowledges from experts in the field of interest. The structure of fuzzy logic control allows local control of the shape of these mapping in an intuitive manner.

However, establishing an appropriate set of fuzzy rules and the corresponding membership functions is the task of trial and error. Normally, if the system is of high degree of nonlinearity in *loading variations*, finding the knowledge base (i.e., fuzzy rule base and membership functions) that gives satisfactory system responses over a *wide* range of reference signals is a very difficult task. Without questions, if SFLCs (Static Fuzzy Logic Controllers) cannot efficiently handle effects of nonlinearity mentioned above, they certainly cannot efficiently handle those of uncertainty as well.

One solution to the problem of nonlinearity in loading variations and of uncertainty is the approach of leaning and/or adaptation. There exists some fuzzy adaptation schemes that have shown very successful results [12]. In this paper, however, the approach of neural networks has been chosen to perform the above task.

As a learning approach, neural networks cannot be used to control a system unless they have been trained. The training algorithms such as α-LMS, μ-LMS or backpropagation use error signals to adjust the weights in networks. This means that errors are supposed to occur in the first place in order to conduct the adaptation. Once errors occur and the networks are trained, we hope that these networks can effectively control our system under the situations in which these errors occur, if this situation is to be encountered again. In fact, adaptation of the networks to deal with new situations may deteriorate the knowledge that has been previously stored unless the capacities of these networks are large enough.

This fact is unacceptable when we deal with systems of high degree of uncertainty such as manipulator arms [2]. According to this reference, if the loading of the arm is changed, a considerable amount of error occurs before the neural-network based controller can fully adapt. At this point, the ability to incorporate human knowledge into control schemes of fuzzy logic comes to the forefront. That is, our own intuitive knowledge can be used to deal with this loading variation via fuzzy logic principles while the adaptation is being conducted.

The principles of FFNN (FeedForward Neural Network) and static fuzzy logic are combined to construct a neural-fuzzy feedforward controller. The control configuration has two degrees of freedom. Therefore, the problem of conflicts in control specifications (e.g., small rise time and small overshoot) can be managed easier [6].

2. Control Considerations and the Proposed Model

In this section, we consider the cases in which nonadaptive control schemes such as static fuzzy logic do not give satisfactory performances. We begin by considering a nonlinear system subjected to loading variation and draw some intuitive facts. Then, we evaluate the limitation and difficulty of using static fuzzy logic control to handle this kind of systems. After that, the mathematical model of the system will be derived for the purpose of simulations.

Consider the system of gear-motor-pendulum depicted in the following figure:

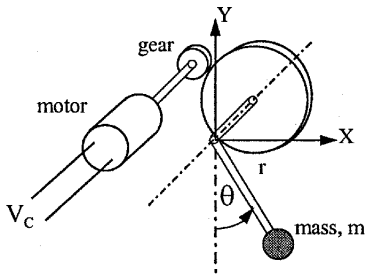

Fig. 1 The system used to test the proposed controller

According to the above figure, our objective is to control the angle θ between the negative Y axis and the pendulum over the range of $[0,2\pi]$. For this system, the torque required to rest the pendulum at any value of θ depends on, not only error in the controlled variable and its derivatives, but also the controlled variable θ itself.

From the static fuzzy logic control viewpoint, although one can adjust the mapping from input domain(s) to output domain(s) by adjusting fuzzy rule base and the corresponding membership functions, intrinsic nonlinearity in these systems tremendously increases the difficulty of the task. In addition to that, for the case in which more than two input signals are necessary, the static fuzzy logic approach consumes a considerable amount of memory and computation time. To overcome the problem of multidimensional mapping, the approach of learning and/or adaptation is considered as a solution.

In this research, a FFNN takes care of learning and adjusting feedforward control efforts (i.e., feedforward controller) while a SFLC is used as the main controller. The overall structure of the proposed controller will be shown in section 5.

To find the differential equations governing the system, we begin with derivation of the equation of motion of the pendulum, in accordance with the following assumptions:

- the gear train has no backlash,
- the pendulum has no mass,
- the shaft, the pendulum and the gear train are rigid.

Consider the schematic diagram of the following mass-pendulum subsystem.

We select θ as the only one generalized coordinate necessary to describe configurations of this subsystem. The position vector of the mass m in the Cartesian coordinate system can be expressed in vector-matrix form as:

$$\mathbf{r} = \begin{bmatrix} r\sin\theta \\ -r\cos\theta \end{bmatrix} \quad (1)$$

Differentiating the above expression twice gives

$$\ddot{\mathbf{r}} = \begin{bmatrix} \ddot{r} - r\dot{\theta}^2 \\ r\ddot{\theta} + 2\dot{r}\dot{\theta} \end{bmatrix} \quad (2)$$

Consequently, the equation of motion in θ is:

$$mr^2\ddot{\theta} = -mgr\sin\theta + T \quad (3)$$

in which $T \equiv$ torque applied to the pendulum

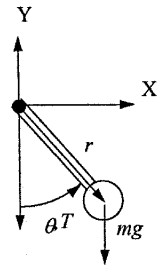

Fig. 2 Mass-pendulum subsystem

Next, consider the schematic diagram of the gear-motor subsystem provided below:

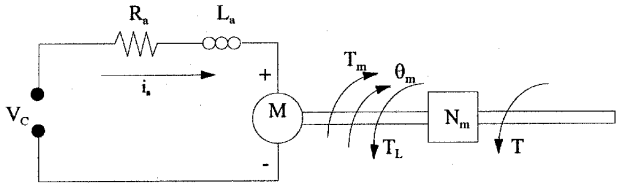

Fig. 3 Schematic diagram of the gear-motor subsystem

At the armature, we have the following equation:

$$\frac{di_a(t)}{dt} = \frac{V_C(t)}{L_a} - \frac{R_a i_a(t)}{L_a} - \frac{V_b(t)}{L_a} \quad (4)$$

in which $V_C(t) \equiv$ control voltage (Volt)

$i_a(t) \equiv$ armature current (Ampare)

$V_b(t) \equiv$ back emf (Volt)

The back emf relates to the angular velocity of the motor as:

$$V_b(t) = K_b \frac{d\theta_m}{dt} = K_b Q_m \quad (5)$$

in which $K_b \equiv$ back-emf constant

$Q_m \equiv$ angular velocity of the motor

The equation relating the motor torque to the armature current is:

$$T_m(t) = K_i i_a(t) \quad (6)$$

in which $K_i \equiv$ torque constant ≈ back-emf constant

Substituting $V_b(t)$ in Equation (5) and $i_a(t)$ in Equation (6) into Equation (4) gives the following equation:

$$\frac{dT_m(t)}{dt} = \frac{K_i V_C(t)}{L_a} - \frac{R_a T_m(t)}{L_a} - \frac{K_i^2 Q_m}{L_a} \quad (7)$$

The equation of motion of the armature is given by

$$\ddot{\theta}_m = \frac{T_m}{J_m} - \frac{T_L}{J_m} - \frac{B_m Q_m}{J_m} = \frac{dQ_m}{dt} \quad (8)$$

in which $J_m \equiv$ equivalent moment of inertia of the armature-gear train-shaft set

$B_m \equiv$ viscous damping coefficient

$T_L \equiv$ load torque

Substituting $\theta = N_m\theta_m$ and $T = T_L/N_m$ in Equation (3) gives:

$$mr^2 N_m \dot{Q}_m = -mgr\sin(N_m\theta_m) + \frac{T_L}{N_m}$$

in which $N_m \equiv$ gear ratio

Rearranging gives:

$$T_L = mr^2 N_m^2 \dot{Q}_m + mgrN_m \sin(N_m\theta_m) \quad (9)$$

Substitute T_L in Equation (9) into Equation (8) and manipulate algebraic expressions, we obtain:

$$\frac{dQ_m}{dt} = \frac{T_m - mgrN_m\sin(N_m\theta_m) - B_m Q_m}{J_m + mr^2 N_m^2} \quad (10)$$

Intuitively, we also have the state equation of:

$$\frac{d\theta_m}{dt} = Q_m \quad (11)$$

Equation (7), (10), and (11) comprise the system of three state equations (linear and nonlinear) in three state variables which is necessary and sufficient to describe the states of the system. Note that a positive armature voltage tends to rotate the pendulum in CCW or positive direction.

3. Structure of Static Fuzzy Logic Controllers

To present the mechanism of static fuzzy logic control, the components of this kind of controller are need to be discussed. In general, a SFLC is composed of a fuzzifier, a fuzzy rule base, a fuzzy inference engine, and a defuzzifier. These components are connected as shown in the following figure. We will discuss the functions of each the components in the following subsections.

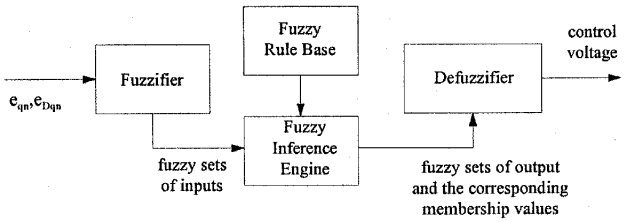

Fig. 4 Structure of adaptive fuzzy logic controller

3.1 Fuzzifier As do all control schemes, fuzzy logic based control needs inputs. But in fuzzy logic schemes, these inputs will be expressed by multivalue logic via set descriptors such as big, small, and so on. Thus, we need to transform input signals in the real world which are numerically expressed into a form which is compatible with this kind of logic. The outputs of a

fuzzifier are linguistic values or membership values of the numerical inputs in all fuzzy sets in the universe of discourse.

In order to build a fuzzifier, the designer first defines fuzzy sets and their shapes. These fuzzy sets are quantities that a human can use in the processes of reasoning and decision-making. Normally, they are quantities with magnitude and direction such as negative big or positive small. This is because humans use these quantities in the kinds of processes mentioned above. Each multivalue quantity is called a fuzzy set and the degree of being in this set is called membership value. A numerical input has its own membership values in all of the fuzzy sets in the universe of discourse. In other words, the mapping of an input from a numerical domain to a fuzzy domain is not a one-to-one mapping. The set that contains all defined fuzzy sets is called the universe of discourse.

After defining inputs, we define fuzzy sets on which these inputs will be mapped. The mechanism of fuzzification will be discussed in accordance with the following figure in which fuzzy sets A and B are assumed:

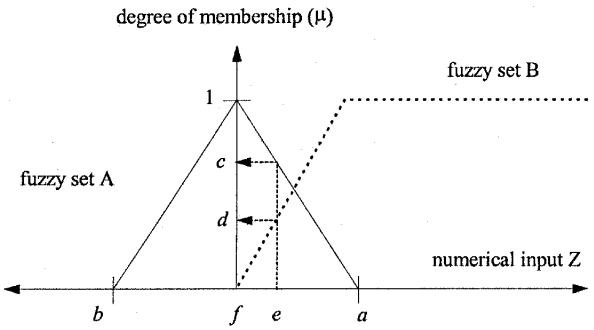

Fig. 5 Membership functions of A and B

The above figure shows that fuzzy set A is defined with a triangular shape and its membership value is zero everywhere except from b to a. The membership value of fuzzy set B is not equal to zero for $Z \in [0,\alpha]$. This reflects the fact that Z fully belongs to set A at Z=0 and the degree of being in set A will decrease as Z moves away from 0. If Z moves away from 0 to the right, the degree of being in set B increases while the degree of being in set A decreases. For every fuzzy set used in this research, its degree of membership varies from zero to one.

The kind of fuzzifier that has been chosen is the singleton fuzzifier. For a crisp point e in Z, a singleton fuzzifier emits unity as degree of membership of e in a fuzzy set A if $e \in [b,a]$. That is, the membership value of e is always unity if e is in the range in which the membership function of A is greater than zero. Otherwise, the degree of membership of e in A is zero. In fact, the numerical degree of membership and the corresponding fuzzy set are the outputs from fuzzifier.

In this research, we choose positive big, positive medium, positive small, zero, negative small, negative medium, and negative big as our fuzzy sets. The names of these sets are same for the inputs and the output but their shapes are different.

3.2 Fuzzy Rule Base The fuzzy rule base is an area of memory in which the knowledge of how to control a system is stored. The fuzzy rule base, position, and shape of the membership functions are deeply interrelated. In fact, it is the shapes and positions of membership functions that define the meaning of the corresponding fuzzy sets in the rule base. Intuitively, two SFLCs with the same fuzzy rule base may or may not behave in the same manner, depending on the shapes and positions of their membership functions.

In this research, the error in the pendulum angle and the rate of change of the angular velocity of the pendulum, denoted by e_θ and $D\dot{\theta}$ respectively, are the two inputs connected to our SFLC. The rule base used in this research is composed of 49 fuzzy rules. For convenience, we use numbers to represent the fuzzy sets as:

1 ≡ negative big 2 ≡ negative medium
3 ≡ negative small 4 ≡ zero
5 ≡ positive small 6 ≡ positive medium
7 ≡ positive big

3.3 Fuzzy Inference Engine A fuzzy inference engine performs the mapping from fuzzy sets of input vectors and their corresponding membership values to fuzzy sets of output vectors and their degrees of membership. The mapping will be implemented by using fuzzy logic implications. There are many fuzzy logic implications that can be used in this part of a fuzzy logic controller. The min-operation rule and the product-operation rule are the two most popular. For the purpose of representation, we denote the membership values of e_θ and $D\dot{\theta}$ by μ_θ and $\mu_{D\dot{\theta}}$ respectively. We also assume that the implementation is done on the fuzzy rule of the following form:

IF e_θ IS F_{in1} AND $D\dot{\theta}$ IS F_{in2} THEN OUTPUT IS F_{out}

in which, F_{in1} ≡ the fuzzy set for error in pendulum angle,
F_{in2} ≡ the fuzzy set for change in angular velocity of the pendulum,
F_{out} ≡ the fuzzy set for output (i.e., control voltage).

Now that we have defined all of the necessary notation, the mentioned two fuzzy operations can separately be described in the following paragraphs.

Using the min-operation rule, the membership value μ of F_{out}, denoted by μ_{out}, can be determined as:

$$\mu_{out} = \min\{\mu_\theta(e_\theta), \mu_{D\dot{\theta}}(D\dot{\theta})\} \qquad (12)$$

If we use the product-operation rule, μ_{out} is given by:

$$\mu_{out} = \mu_{\theta}(e_{\theta}) * \mu_{D\dot{\theta}}(D\dot{\theta}) \quad (13)$$

According to the above two equations, one can obviously see that since $\mu \in [0,1]$, the product-operation rule has more sensitivity to changes in input vectors. Due to this fact, the product-operation rule has been selected so that the controller can rapidly sense small changes in the input vector. Consequently, the resulting controller can generate the appropriate control action to move the system to the desired state before the magnitude of the error increases.

3.4 Defuzzifier A defuzzifier transforms fuzzy quantities to be control actions in the real world. In other words, it maps membership values of the output vector from all fuzzy IF-THEN rules onto a crisp output vector. There are many mappings that can be applied, but we will use the center average mapping.

In order to express the output of the defuzzifier in a mathematical form, we define the center of a fuzzy set as the crisp value of a linguistic variable (a linguistic variable can be expressed either numerically or verbally) that corresponds to the point at which the membership function of the fuzzy set of interest is a maximum. If one considers Fig. 5, one will find that the centers of fuzzy sets A and B are f and a respectively.

Using the center average defuzzifier, the output of our fuzzy logic controller can be determined from:

$$V_{C-FUZZY} = \frac{\sum_{i=1}^{M} \bar{v}_i (\mu_{\theta}(e_{\theta}) * \mu_{D\dot{\theta}}(D\dot{\theta}))_i}{\sum_{i=1}^{M} ((\mu_{\theta}(e_{\theta}) * \mu_{D\dot{\theta}}(D\dot{\theta}))_i} \quad (14)$$

in which, $\bar{v}_i \equiv$ center of membership functions for fuzzy sets of the control voltage from the i^{th} rule

$V_{C-FUZZY} \equiv$ output voltage from defuzzifier,

$M \equiv$ number of fuzzy IF-THEN rules in fuzzy rule base = 49.

4. FFNN and Learning Algorithms

Although there are several kinds of neural networks used nowadays, this research incorporates the network of feedforward . One of the obvious reasons is that this kind of neural networks has shown satisfactory performances in the field of control. In addition to that, the backpropagation learning algorithm can be used to train the above mentioned networks. The backpropagation algorithm is derived from very familiar laws in differential calculus, the chain rule of differentiation and the principles of partial differentiation. Thus, the learning mechanism of the network is rigorous and generally, is easy to be understood by most engineers. This fact enhances the possibility to develop better algorithms in the future.

4.1 Neuron In FFNN, layers of neurons are arranged such that every neuron in any two consecutive layers is fully interconnected by forward paths. The following figure shows the structure of a neuron.

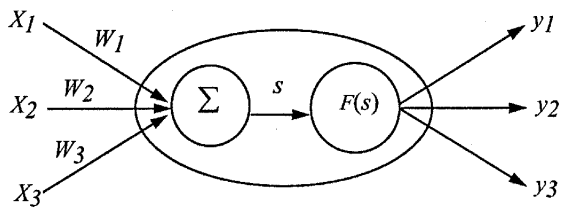

Fig. 6 A neuron

The ellipse in Fig. 6 represents a neuron. Inputs and outputs paths of the neuron are represented by arrows. There is no restriction about the dimensions of input and output vectors. In this research, however, the dimension of inputs of our FFNN is same as that of SFLC. Each line of inputs X_n has the corresponding weight W_n. If we denote the output by y_n, the relation between the input vector, the weight vector, and the output is given by:

$$y_n = F(s) = F(\sum_{i=1}^{n} (X_i W_i)) \quad (15)$$

in which $F \equiv$ activation function.

The choice of F depends on the range of the required output. Normally, F takes one of the following:

1) $F = \dfrac{a}{1+e^{-s}}$

2) $F = a\tanh(s)$

3) $F = as$

in which $a \equiv$ constant to be chosen.

As a matter of fact, there are some discontinuous functions that have widely been used (e.g., $F = \mathrm{sgn}(s)$). We do not use this kind of functions because our training algorithm is based on the principle of differentiation which requires the continuity of functions. The choice of 1) or 2) makes the magnitude of output signals controllable.

4.2 Structure of Feedforward Neural Network As the name implies, this kind of neuron-connected networks has only feedforward paths. A FFNN can be built by connecting neurons which are represented by circles in the following manner.

The network is composed of layers of neurons. The layer exposed to the inputs is called the input layer. The output layer is the one that emits outputs of the networks and the layers between the input layer and the output layer are called hidden layers. There is no restriction about the number of neurons in each layer and the number of hidden layers.

In the input layer, experts in the field usually let the outputs of a neuron equal to its input (i.e., $F(s)=1$). As one shall see in the

derivation of the training algorithm, this tradition is not necessary although it is what most people do. The output signal of every neuron in hidden layers and the output layer can be computed as in Equation (15).

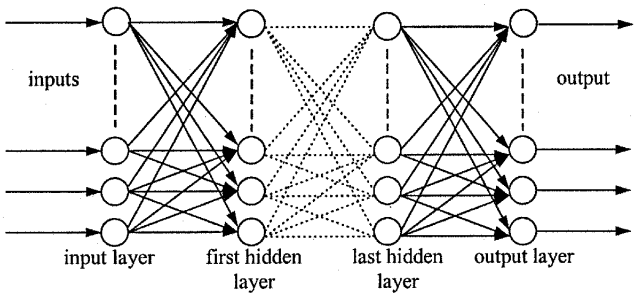

Fig. 7 Structure of feedforward neural network

4.3 Backpropagation Algorithm The beauty of the backpropagation algorithm is that it can clearly be derived from the chain rule of differentiation. Therefore, the mechanism of the algorithm can be realized in every step.

In the algorithm, the error signal of a neuron is fed from its output back to the body of the neuron itself. Normally, this feedback error signal is the square of the actual error. The following is the commonly used relation between the feedback error signal and the actual error:

$$E = (d-y)^2 = err^2 \tag{16}$$

in which $d \equiv$ the desired output

$y \equiv$ actual output

$E \equiv$ feedback error signal

$err \equiv$ actual error

The purpose of the learning is to adjust the weights in the network such that E is minimized. Therefore, we need to know the rate of changes of E with respect to the changes of all of the weights in the network so that we can determine the direction of weight changes that makes E smaller. Thus, in the differentiation point of view, we need to find $\frac{\partial E}{\partial W_i^{\cdots}}$. If we apply the chain rule of differentiation, we may write:

$$\frac{\partial E}{\partial W_i^{H_L O}} = \frac{\partial E}{\partial y^O}\frac{\partial y^O}{\partial s^O}\frac{\partial s^O}{\partial W_i^{H_L O}} \tag{17}$$

in which $y^O \equiv$ output of the neuron in the output layer

$s^O \equiv$ input of the activation function of the neuron in the output layer

$W_i^{H_L O} \equiv$ the weight between the i^{th} neuron in the last hidden layer and the output layer

According to Equation (17), one can differentiate Equation (16) as:

$$\frac{\partial E}{\partial W_i^{H_L O}} = -2E\frac{\partial F^O(s^O)}{\partial s^O}\frac{\partial(\mathbf{W}^{\mathbf{H_L O^T}}\mathbf{y}^{\mathbf{H_L}})}{W_i^{H_L O}}$$

or

$$\frac{\partial E}{\partial W_i^{H_L O}} = -2E\frac{\partial F^O(s^O)}{\partial s^O}y_i^{H_L} \tag{18}$$

in which $\mathbf{y}^{\mathbf{H_L}} \equiv$ output vector of neuron in the last hidden layer

$\mathbf{W}^{\mathbf{H_L O^T}} \equiv$ the weight vector that corresponds to $\mathbf{y}^{\mathbf{H_L}}$

Equation (18) gives the rate of change of E with respect to each of all of the weights between the output layer and the last hidden layer.

To find the rate of change of E with respect to the weights between the last hidden layer and the layer before it, we use the chain rule in the same fashion as the previous.

$$\frac{\partial E}{\partial W_{ji}^{H_{L-1}H_L}} = \frac{\partial E}{\partial y^O}\frac{\partial y^O}{\partial s^O}\frac{\partial s^O}{\partial y_i^{H_L}}\frac{\partial y_i^{H_L}}{\partial s_i^{H_L}}\frac{\partial s_i^{H_L}}{\partial W_{ji}^{H_{L-1}H_L}} \tag{19}$$

in which $W_{ji}^{H_{L-1}H_L} \equiv$ the weight between the j^{th} neuron in the hidden layer L-1 and the i^{th} neuron in the last hidden layer (i.e., layer L)

$y_i^{H_L} \equiv$ output of the i^{th} neuron in the hidden layer

$s_i^{H_L} \equiv$ input of the activation function of the i^{th} neuron in the last hidden layer

According to Equation (19), we can write:

$$\frac{\partial E}{\partial W_{ji}^{H_{L-1}H_L}} = -2E\frac{\partial F^O(s^O)}{\partial s^O}W_i^{H_L O}\frac{\partial F^{H_L}(s_i^{H_L})}{\partial s_i^{H_L}}y_{ji}^{H_{L-1}H_L} \tag{20}$$

At the time of writing this section, simulations of the proposed control system have shown that the network with only one hidden layer is capable of giving satisfactory system responses. Therefore, further derivations will not be discussed here. Nevertheless, one can derive the rate of change of E with respect to any weights between any pair of neurons in any two consecutive layers in the manner shown above.

For the case in which the network has one hidden layer and the activation function of all neurons in the input layer is $F(s)=s$, Equation (20) becomes:

$$\frac{\partial E}{\partial W_{ji}^{IN-H_L}} = -2E\frac{\partial F^O(s^O)}{\partial s^O}W_i^{H_L O}\frac{\partial F^{H_L}(s_i^{H_L})}{\partial s_i^{H_L}}X_j$$

in which $X_j \equiv$ input of the j^{th} neuron in the input layer

$W_{ji}^{IN-H_L} \equiv$ weight between the j^{th} neuron in the input layer and the i^{th} neuron in the hidden layer

The following expressions are the results of differentiating $F(s)$ with respect to s:

1) If $F = \frac{a}{1+e^{-s}}$, then $\frac{\partial F}{\partial s} = aF(s)(1-F(s))$

2) If $F = a\tanh(s)$, then $\frac{\partial F}{\partial s} = a(1-(F(s))^2)$

3) If $F = as$, then $\frac{\partial F}{\partial s} = a$

4.4 Learning Law We apply the delta rule of the following form:

$$W_i^{k+1} = W_i^k - \mu \frac{\partial E}{\partial W_i^k} \qquad (21)$$

in which $\mu \equiv$ step size of change in W_i from the present state to the next state

According to the principle of steepest descent, an inappropriate size of change in W_i (i.e., μ) may lead to problems. If μ is too small, the number of steps that the above law takes to reach the local minimum of E will be considerable and thus the learning rate is slow. Inversely, if μ is too large, the above algorithm will not be able to find the local minimum of E. Note that the choices of initial values of W_i lead to different final values of E.

It is obvious that the algorithm may or may not give the value of W_i that corresponds to the globally minimum of E over domain of W_i. This depends on the initial value of W_i as discussed. Intuitively, since the algorithm minimizes E^2, one can say that the *magnitude* of the actual error is minimized. It is possible that one will vary μ from step to step to increase the rate of convergence by considering the magnitude of change in W_i of the previous steps. For more information about this subject, see prediction and extrapolation techniques in books dedicated to numerical computation.

5. Structure of Neural-Fuzzy Feedforward Controller

This section illustrates the structure of the proposed controller and some ideas that stand behind the design. We begin with the structure of the controller. After that, the choice of error signal fed to the learning algorithm will be discussed.

It is known that using one-degree-of-freedom control schemes (i.e., there is only one controller in the system) leads to the relatively limited performance criteria (e.g., a system with small rise time usually has an excessive amount of overshoot). Because of the above fact, the configuration proposed is of feedforward with two degrees of freedom. Fig. 8 depicted the overall structure of the controller.

Only control action from SFLC is not enough to force the system such that the system responses are satisfactory. This control action, however, can be easily managed by defining fuzzy rule base and the corresponding membership functions. These definitions are, although imprecise, admissible because they are derived form known facts. The fact that fuzzy parameters can be loosely defined to some degree makes defining fuzzy parameters a lot easier. Under this configuration, SFLC-with no adaptation, emits the main control action while FFNN learns and adapts itself for the appropriate feedforward control action.

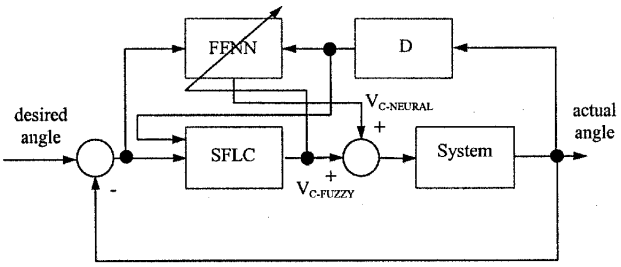

Fig. 8 The overall control structure

As one can see from Fig. 8, the output of the fuzzy controller (i.e., main control voltage) is used as the error signal fed back to FFNN. The backpropagation algorithm will adapt the network such that the output of SFLC is minimized. Therefore, one can say that FFNN is trained to minimize the error signal of θ as well.

It is important to look back to the time the backpropagation algorithm has been derived. At that time, the error was defined to be equal to the difference between the desired output and the actual output of the network. This implies that the dimension of the error and the output signal of FFNN must be the same.

6. Simulation Results

In this part, we investigate and compare the responses of the gear-motor-pendulum system for the cases in which the SFLC and FFNN are on the line to those when FFNN is removed. In other words, we focus on the effects of the adaptation and learning ability of FFNN to the system responses.

The following simulations are based on the fourth-order Runge-Kutta Method with time step of 0.000025 second.

6.1 Fuzzy Parameters in SFLC The fuzzy rule base will be shown in accordance with the notations given in section 3.2 as below:

Rule Base		$D\dot{\theta}$						
		1	2	3	4	5	6	7
e_θ	1	4	3	2	2	1	1	1
	2	6	4	3	2	2	1	1
	3	6	6	4	3	2	1	1
	4	7	6	5	4	3	2	1
	5	7	7	6	5	4	3	3
	6	7	7	6	6	5	4	3
	7	7	7	7	6	6	5	4

Table 1 Fuzzy rule base

The membership functions of the fuzzy sets of each of the two inputs in the fuzzy rule base are defined in accordance with the following figures and tables.

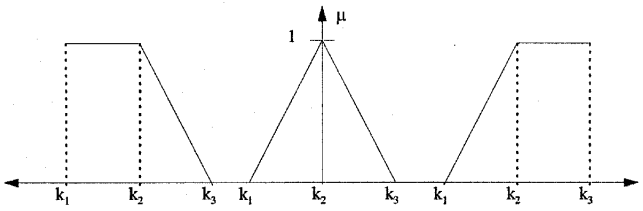

Fig. 9 Input membership functions

Fuzzy Set	1	2	3	4	5	6	7
k_1(rad)	-α	-0.3	-0.1	-0.05	0.0	0.05	0.1
k_2(rad)	-0.3	-0.1	-0.05	0.0	0.05	0.1	0.3
k_3(rad)	-0.1	-0.05	0.0	0.05	0.1	0.3	α

Table 2 Data of membership functions of e_θ

Fuzzy Set	1	2	3	4	5	6	7
k_1(rad)	-α	-0.6	-0.4	-0.2	0.0	0.2	0.4
k_2(rad)	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6
k_3(rad)	-0.4	-0.2	0.0	0.2	0.4	0.6	α

Table 3 Data of membership functions of $D\dot{\theta}$

The centers of membership functions of the fuzzy sets of the control voltage are shown in the following table.

Fuzzy Sets	1	2	3	4	5	6	7
Centers of Membership Functions (Volts)	-12	-9.5	-7	0	7	9.5	12

Table 4 Centers of membership functions of fuzzy sets in output

6.2 Information about FFNN The proposed FFNN has one hidden layer with 29 neurons, two input nodes and one output node. The initial values of all of the weights in the networks are zeroes. The step size of changes in weights (μ) is equal to 0.005 and the activation function is $F(s) = 8\tanh(s)$.

6.3 Physical Properties of the Gear-Motor-Pendulum System The physical properties of the system are realistically taken[1]. They are shown in the following table.

m (kg)	0.6
J_m (kg.m^2)	0.005
B_m (N.m.s)	0.00009
g (m/s^2)	9.81
L_a (Henry)	0.00001
K_i (N.m/Ampare)	0.088
R_a(Ohm)	1
N_m	0.1
r (m)	0.3

Table 5 Physical properties of the gear-motor-pendulum system

6.4 Ability of the Proposed Controller to Handle effects of Nonlinearity The objective of the following simulation is to investigate and compare the performance of the proposed controller to that of SFLC over wide range of step command signal. Initially, the system is launched from the initial condition of θ = 0.5 radian. Then, the command signal is changed during $t \in$ [0,20] second in accordance with the following table.

Time (second)	0	5	9	13
Command (radian)	1	0.7	1.1	0.45

Table 6 Changes in command signal

The result of simulation when only SFLC is in the line is shown below:

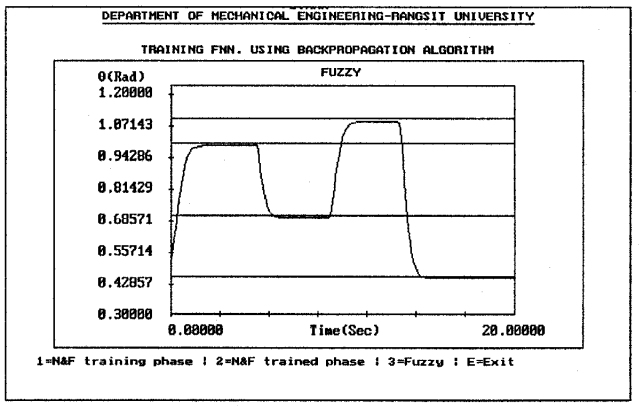

Fig. 10 Response of the system with only SFLC

The response of the system with both SFLC and FFNN is as below:

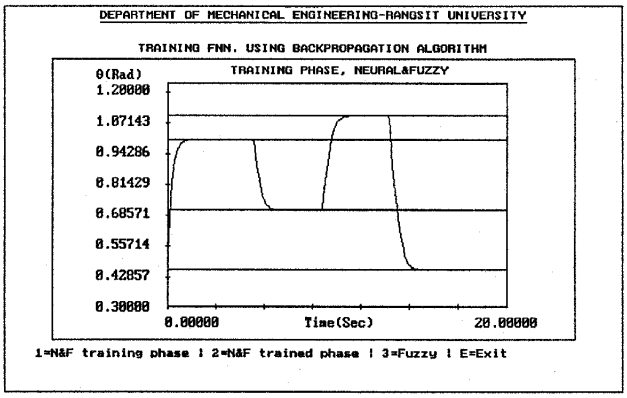

Fig. 11 Response of system with both SFLC and FFNN

According to the above two simulations, FFNN can help SFLC to force the system such that the steady-state error becomes zero over wide range of nonlinear-loading variation. It is obvious that the unsupervised-learning ability of the proposed controller performs well without necessity to adjust fuzzy parameters to some degree.

[1] Properties of motor are taken from data sheets of motor model S9M4H, manufactured by PMI Motors, U.S.A..

6.5 Ability of the Proposed Controller to Handle Effects of Uncertainty To test the ability of the proposed controller in dealing with uncertainty, we consider the situation in which the motor constant K_i and the mass m are changed from 0.088 N.m/Ampare and 0.6 kg to 0.03 N.m/Ampare and 0.8 kg respectively. We do not adjust any control parameters to take into account the above imposed uncertainty so that we can see how well the proposed controller internally adapts and performs in this kind of situations.

The simulation of the system with only SFLC in the line is as shown below:

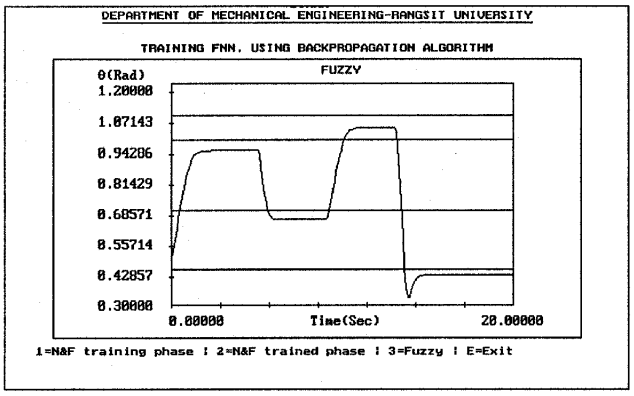

Fig. 12 Response of the system subjected to uncertainty with only SFLC

The simulation of the neural-fuzzy feedforward control system is as shown below.

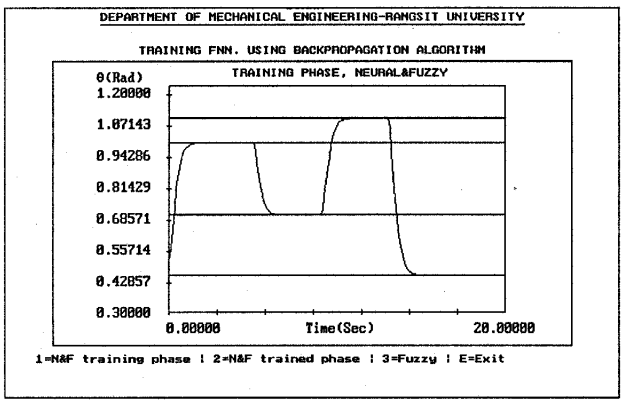

Fig. 13 Response of the proposed control system subjected to uncertainty

In the first simulation of section 6.5, one can see that the response of the system imposed with uncertainty has more steady-state error and overshoot than those without uncertainty. However, according to Fig. 13, considerable effects of uncertainty to the system response can be satisfactorily handled by the proposed neural-fuzzy feedforword controller.

7. Conclusions

In this paper, the natures of analytical control techniques are considered in the first place. It was pointed out that these well-known techniques, although mathematically derivable, give unsatisfactory results when applied to nonlinear-uncertain systems.

The principles of Static Fuzzy Logic Controller (SFLC) are presented. In these principle, the designers cannot avoid difficulties in choosing fuzzy parameters such that the system responses are satisfactory. Because of this fact, the principle of FeedForward Neural Network (FFNN) is proposed with the associated backpropagation algorithm and the delta learning rule. The FFNN is arranged to work with SFLC in the form of feedforward controller having two degrees of freedom. This hybrid controller combines advantages of knowledge-based scheme of SFLC to that of learning/adaptation schemes of FFNN.

The proposed neural-fuzzy feedforward controller is tested without human supervision with a gear-motor-pendulum system subjected to nonlinearity in loading variation and uncertainty. The simulations show that the proposed controller is not only capable of handling effects of the above kind of nonlinearity but it is powerful enough to deal with that of uncertainty as well.

Reference

1 Burden, R. L., and Faires, J. D., 1993, "*Numerical Analysis*," Fifth Edition, Boston, MA: PWS Publishing.

2 Ciliz, M. K., and Isik, C., 1990, "Trajectory Following Control of Robotic Manipulators Using Neural Networks, " *Proc. IEEE Int. Conf. Intelligent Control Systems*, pp. 536-540.

3 Haug, E. J., 1992, "*Intermediate Dynamics*," Englewood Cliffs, NJ: Prentice Hall.

4 Hisao Ishibuchi, Ryosuke Fujioka, and Hideo Tanaka, 1993, "Neural Networks That Learns from Fuzzy If-Then Rules," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp.85-97.

5 Kuschewski, J. G., Hui, S., and Zak, S. H., 1993, "Application of Feedforward Neural Networks to Dynamical System Identification and Control," *IEEE Trans on Control Systems and Technology*, Vol. 1, no. 1, pp. 37-49.

6 Kuo, B. C., 1995, "*Automatic Control Systems*," Seventh Edition, Englewood Cliffs, NJ: Prentice Hall.

7 Li, Y. F., and Lau, C. C., 1989, "Development of Fuzzy Algorithms for Servo Systems," *IEEE Control System Magazine*, Vol. 9, number 3, pp. 65-71.

8 Michael Chester, 1993, "*Neural Networks a Tutorial*," Englewood Cliffs, NJ: Prentice Hall.

9 Nguyen, D. H., and Widrow, B., 1990, "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, Vol. 10, No. 3, pp.18-23.

10 Page, G. F., Gomm, J. B., andWilliams, D., 1993, "*Application of Neural Networks to Modelling and Control*," London: Chapman & Hall.

11 Raven, F. H., 1987, "*Automatic Control Engineering*," Fourth Edition, Singapore: McGraw-Hill Inc..

12 Wang, L. X., 1994, "*Adaptive Fuzzy Systems and Control Design and Stability Analysis*," Englewood Cliffs, NJ: Prentice Hall.

13 Widrow, B., and Lehr, M. A., 1990, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. of IEEE*, Vol. 78, No. 9, pp.1415-1442.