Published in *Engineering Letters*, **13**, (2), pp. 148–157, 2006.

GA-based Neural Fuzzy Control of Flexible-link Manipulators

M. N. H. Siddique, and M. O. Tokhi

Abstract—**The limitations of conventional model-based control mechanisms for flexible manipulator systems have stimulated the development of intelligent control mechanisms incorporating fuzzy logic and neural networks. Problems have been encountered in applying the traditional PD-, PI-, and PID-type fuzzy controllers to flexible-link manipulators. A PD-PI-type fuzzy controller has been developed where the membership functions are adjusted by tuning the scaling factors using a neural network. Such a network needs a sufficient number of neurons in the hidden layer to approximate the nonlinearity of the system. A simple realisable network is desirable and hence a single neuron network with a nonlinear activation function is used. It has been demonstrated that the sigmoidal function and its shape can represent the nonlinearity of the system. A genetic algorithm is used to learn the weights, biases and shape of the sigmoidal function of the neural network.**

***Index Terms*—Fuzzy control, Flexible-link manipulators, Genetic algorithms, Neuro-fuzzy control.**

I. INTRODUCTION

Due to elastic properties of flexible manipulators, the development of a mathematical description and subsequent model-based control of the system is a complicated task. This is made difficult by the presence of a large (infinite) number of modes of vibration in the system. The modes become significant in two ways: firstly, because the oscillations themselves prolong the settling time and secondly, because attempts to actively control some modes result in instability of the other modes. This non-linear behaviour of the structure at high speeds, firstly, degrades end-point accuracy and secondly complicates controller development. Furthermore, the performance of such a control system depends mainly on the parameters during operation. These limitations of conventional model-based control for flexible manipulator systems have stimulated the development of intelligent control mechanisms incorporating adaptive control, neural networks (NNs) and fuzzy logic. Thus, an investigation into the development of an intelligent control mechanism using fuzzy logic and neural networks is intended in this research work.

Although fuzzy logic controllers (FLCs) exhibit superior applicability to the traditional PID controllers and are highly robust, PI-like and PD-like FLCs possess mainly the same characteristics as traditional PI and PD controllers, respectively. The PI-like fuzzy controller has good performance at the steady state, but yields penalised rise time and settling time. On the other hand, PD-like control can reliably predict and correct large overshoots, but the derivative control will affect the steady-state error of a system only if the steady-state error varies with time. If the steady-state error of a system is constant with respect to time, the time derivative of error will be zero, and derivative control will have no effect on the steady-state error [1], [2]. In order to meet the design criteria of zero steady state error, minimum overshoot and fast rise time, a further option is to develop a PID-type FLC which ensures fast rise time, smaller overshoot and settling time from PD part and minimum steady state error from PI part of the PID controller. The generic fuzzy PID controller is a four-dimensional (three input - one output) fuzzy system with a huge rule-base, which increases exponentially with the number of inputs and number of fuzzy sets. Processing of such a rule-base is time consuming and demands large memory space. To overcome the problems of PD-, PI-, and PID-type controllers described earlier, a PD-PI-type FLC is developed for a flexible-link manipulator where a PD-type FLC is used first and after reaching the set point the controller is switched from PD-type to PI-type [3],[4]. Thus, a shorter rise time and smaller overshoot is guaranteed with the use of PD-type controller and shorter settling time and minimised steady-state error is guaranteed with the use of the PI-type controller. The membership functions (MFs) for error, change of error and sum of error were chosen heuristically, especially for change of error and sum of error these were heuristically defined within the same universe of discourse and a single rule-base was used.

Heuristically chosen membership functions do not reflect the actual data distribution in the input and output spaces. In general, the designer chooses the shape of membership functions and the respective parameters are required to be adjusted by using learning algorithms. Furthermore, the performance of Mamdani-type fuzzy controller mainly depends on the If-Then rules, membership functions and tuning of both [5]. Unfortunately, there are no formal methods to construct the rule-base or define the membership functions for

Manuscript received February 15, 2006.

M. N. H. Siddique is with the School of Computing and Intelligent Systems, University of Ulster at Magee, Londonderry BT48 7JL, NI, UK. (Phone: +44(0)28-71375340; fax: +44(0)28-71375470; e-mail: nh.siddique@ulster.ac.uk).

M. O. Tokhi, is with the Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, England, UK (e-mail: o.tokhi@sheffield.ac.uk).

Mamdani-type fuzzy controllers. Efforts have been made to automate the construction of rule-bases and define the MFs in various ways using NNs and genetic algorithms (GAs) [6]-[10]. In most of the cases, the rule-base is fixed and the parameters of the MFs are adjusted. In many cases, the same result can be obtained by tuning the scaling factors or adjusting the MFs. Adjustment of MFs requires learning of several parameters and hence scaling factor tuning is a much simpler task than adjustment of parameters [11]. A multilayer NN with sufficient number of neurons in the hidden layer can approximate non-linearities such as this but a possible difficulty is that it will consume most of the processing time in calculating values of the scaling factors, and this will make the real-time application difficult. A simple realisable network is desirable, which can be employed in the PD-PI FLC. Considering this as a design criterion, the self-learning task of a multilayer perceptron could be simply replaced by a single neuron with a non-linear activation function.

As experience persuades to believe that single neuron network with non-linear activation function shows better performance than that with a linear activation function. A criterion is required for selection of an optimal NN to represent the non-linearity of the system. However, many parameters of such non-linear activation functions, such as the optimum shape of a sigmoid function, are determined by trial and error. In this paper, a GA-based technique is used to optimise the shape of the activation function, weights and bias of the network.

II. Flexible-link Manipulator

The experimental rig constituting the flexible manipulator system consists of two main parts: a flexible arm and measuring devices. The flexible arm contains a flexible link driven by a printed armature motor at the hub. The measuring devices are shaft encoder, tachometer, accelerometer and strain gauges along the length of the arm. The shaft encoder, tachometer and accelerometer are essentially utilised in this work. The schematic diagram of the flexible-link manipulator is shown in Figure 1. The flexible arm consists of an aluminium-type beam, shown in Figure 2. The outputs of the sensors as well as a voltage proportional to the current applied to the motor are fed to a computer through a signal conditioning circuit and an anti-aliasing filter for analysis and calculation of the control signal. Physical parameters of the flexible arm are given in Table 1.

Due to the elastic properties of the flexible manipulators, the development of a dynamic mathematical description is a complicated task. This non-linear behaviour of the structure firstly degrades end-point accuracy and secondly complicates controller development. Furthermore, the performance of such control depends mainly on the parameters during operation. These limitations of conventional model-based control mechanisms for flexible manipulator systems have stimulated the development of a GA-based neuro-fuzzy controller.

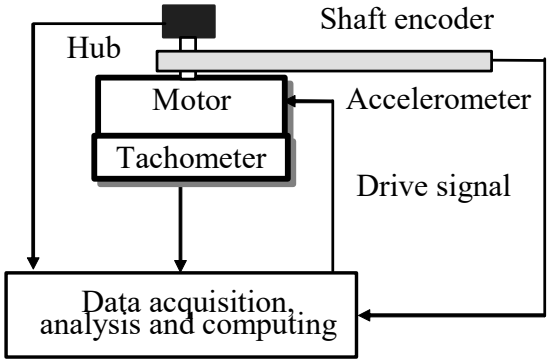


Figure 1: Schematic diagram of the manipulator.

Figure 2: Experimental flexible-link manipulator.

Table 1: Physical parameters of the flexible manipulator.

Parameter	Value
Length	960.0 mm
Width	19.008 mm
Thickness	3.2004 mm
Mass density/ unit volume	2710 kgm^{-3}

III. Fuzzy Logic Controller for Flexible-link Manipulator

The conventional approach to FLC design is to generate a fuzzy rule set based upon the system states such as error, change of error or sum of error, thus producing a two-input single-output PD-, or PI-type or three-input single-output PID-type control rule base. PI-type FLCs are most common and practically followed by the PD-type FLCs. Generally, good performance is achieved with PD-type fuzzy controller during the transient state, i.e., the PD-type fuzzy controller will result in a rapid response. However, at the steady state, elements of error and change of error are possibly too small and the control signal,

through fuzzy inference, becomes zero. The zero control signals will cause steady-state error or oscillations at the steady state [24, 25].

A PD-like FLC can be developed by using an error and change of error model as

$$u = k_p e + k_d \Delta e \tag{1}$$

where k_p and k_d are the proportional and the differential gain coefficients and e is the error, Δe is the change of error and u is the control input. In this type of FLC, it is assumed that no mathematical model for the flexible-link is available except two states, namely, the hub angle error and change of error. Only hub angle θ is measured from the system and the error and change of error are derived from θ. The hub angle error and change of error are defined as:

$$e(k) = \theta_d - \theta(k) \tag{2}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{3}$$

where θ_d is the desired hub angle, e is the error and Δe is the change in angle error.

Triangular MFs are chosen for error e, change of error Δe and torque input u. The membership functions for hub angle error, change of hub angle error, and torque input are shown in Figures 3(a)-3(c). The universe of discourse for the hub angle error, change in hub angle error are chosen as [–36, +36] degree, and [–25, +25]. The universe of discourse of the output, i.e., input torque is chosen as [–3, +3] volts. To construct a rule base, the hub angle error, change of angle error and torque input are partitioned into five primary fuzzy sets as:

Hub angle error E = *{NB, NS, ZO, PS, PB}*
Change of angle error C = {NB, NS, ZO, PS, PB}
Torque U = {NB, NS, ZO, PS, PB}

where E, C and U are the universes of discourse for hub angle error, change of hub angle error, and torque input respectively. The *n*th rule of the rule base for the FLC, with error and change of error as inputs, is as:

R_n : IF (e is E_i) and (Δe is C_j) THEN (u is U_k)

where R_n, $n = 1,2,\cdots,N_{\max}$ is the *n*th fuzzy rule, E_i, C_j, and U_k, for $i, j, k = 1,2,\cdots,5$ are the primary fuzzy sets.

The performance of PI-type FLC, on the other hand, is known to be quite satisfactory for linear first-order systems [22, 23]. But, as with conventional PI-controllers, the performance of PI-type FLCs for higher order systems, and for systems with integrating elements or large dead time, and also for non-linear systems may be very poor due to large overshoot and excessive oscillation. It is well known that the PI-type FLC exhibits good performance at the steady state like the traditional PI-type controllers. That is, the PI-like FLC reduces steady-state error, but yields penalized rise time and settling time [24]. The PI-type controllers give inevitable overshoot when attempted to reduce the rise time, especially when a system of order higher than one is under consideration [22]. These undesirable characteristics of fuzzy PI controllers are caused by integral operation of the controller, even though the integrator is introduced to overcome the problem of steady state error.

A conventional PI-controller is described as:

$$u = k_P \cdot e + k_I \cdot \int e dt \tag{4}$$

where k_P and k_I are the proportional and the integral gain coefficients. Taking the derivative with respect to time of equation (4) yields

$$\dot{u} = k_P \dot{e} + k_I e \tag{5}$$

This can equivalently be written as:

$$\Delta u = k_P \Delta e + k_I e \tag{6}$$

The PI-like FLC rule-base, accordingly, consists of rules of the form:

If e is E_i and Δe is CE_j Then Δu is CU_k

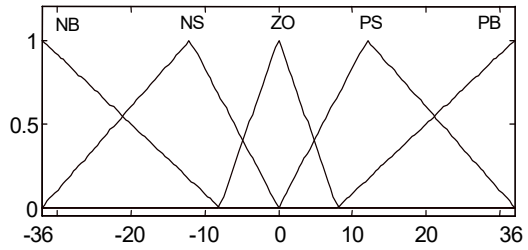
This type of controller is called an incremental PI-like FLC. The inputs are the same as a PD-like FLC with error and change of error except the control input is incremented at each time. Actually, the rules of fuzzy controller are designed with phase plane in mind, in which the fuzzy controllers drive a system into the so-called sliding mode. The tracking boundaries in the phase plane, however, are related not with incremental control input but with control input itself, which is calculated as

$$u(k) = \Delta u(k) + u(k-1) \tag{7}$$

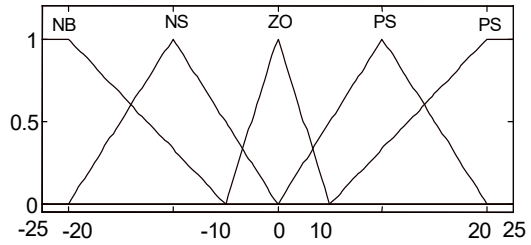
To select the maximum variation of the incremental control input Δu giving satisfactory rise time and maximum overshoot is not so easy as in the case where the control input itself is to be determined [22]. One natural approach to overcome such difficult situation is to adopt the rate of change of error. Such a controller may be called as PID fuzzy controller. The problems associated with implementing a fuzzy PID-type controller will be discussed later in this section. Rather an absolute PI-type controller is computationally viable. In an absolute PI-type FLC, error and sum of error are used as inputs and it is expressed as

$$u = k_P e + k_I \sum e \tag{8}$$

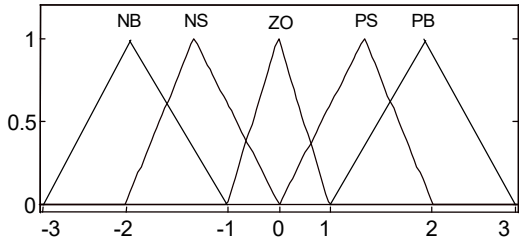
where $\sum e$ is the sum of error. In this type, the hub angle is measured from the system and the sum of hub angle error is derived from the hub angle error. Triangular MFs are chosen for error e, sum of error Σe and torque input u. The MF for sum of hub angle error is shown in Figure 3(d). The universes of discourse for sum of hub-angle error is chosen as [–150, +150] degree. The MFs for error and torque inputs are defined in Figure 3(a) and 3(c).



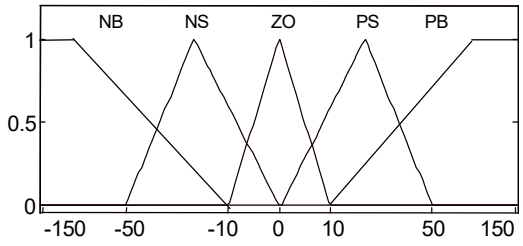
(a) Hub angle error.



(b) Change of hub-angle error.



(c) Torque input.



(d) Sum of hub-angle error.

Figure 3: Membership functions for inputs and output.

To construct a rule base, the hub angle error, sum of hub angle error and torque input are partitioned into five primary fuzzy sets as:

Hub angle error E = {NB, NS, ZO, PS, PB}
Sum of hub-angle S = {NB, NS, ZO, PS, PB}
Torque U = {NB, NS, ZO, PS, PB}

where E, S and U are the universes of discourse for hub-angle error, sum of hub-angle error and torque input respectively. The nth rule of the rule base for this PI-type FLC is as:

$$R_n : \text{IF } (e \text{ is } E_i) \text{ and } (\text{ s is } S_j) \text{ THEN } (\text{ } u \text{ is } U_k)$$

where R_n, $n = 1,2,\cdots,N_{\max}$ is the nth fuzzy rule, E_i, S_j, and U_k, for $i, j, k = 1,2,\cdots,5$ are the primary fuzzy sets.

A practical problem of implementing a fuzzy controller with an integral term is the difficulty of deciding on the number of time units to go back in calculating the sum in equation (8). Even the literature on conventional control theory tends to be somewhat vague on this point, and many texts use an indefinite integral type of notation when representing the integral term, though obviously it is not to be taken literally. Experience with the system suggested using 10 time units to indicate recent tendencies in the error, and experimentation demonstrated that this works very well. It was also convenient to work with an average rather than a sum so that the base value can be easily compared with the current error. Thus, the $\sum e$ base value is calculated as

$$\sum e(k) = \sum_{i=k-9}^{k} e(i) \tag{9}$$

Generally, PD-type two-term fuzzy controllers usually cannot eliminate steady state error and PI-type two-term fuzzy controller can eliminate steady state error but it has slower response due to the integral control variable. In order to meet the design criteria of fast rise time, minimum overshoot, shorter settling time and zero steady state error, a further option is to develop a PID-type FLC which enables fast rise time, smaller overshoot and settling time from PD part and minimum steady state error from PI part of the PID controller. The generic fuzzy PID controller is a four-dimensional (three input-one output) fuzzy system. The basic idea of a PID controller is to choose the control law by considering the error e, change of error Δe and integral of error or sum of error $\sum e$, and thus giving the controller as

$$u_{PID} = k_P \cdot e + k_D \cdot \Delta e + k_I \cdot \sum e \tag{10}$$

The fuzzy control rule corresponding to the PID-controller has the form

$$R_{PID}^{(n)}: if\ e\ is E_i\ and\ \Delta e\ is CE_j\ and \sum e\ is SE_k\ then\ u\ is U_l$$

Theoretically, the number of rules to cover all possible input variations for a three-term fuzzy controller is $n_1 \times n_2 \times n_3$, where n_1, n_2, and n_3 are the number of linguistic labels of the three input variables. If $n_1 = n_2 = n_3 = n$, then the number of rules $R = n \times n \times n = n^3$. For example, if $n = 5$, then the total number of rules will be $R = 125$. In practical applications the design and implementation of such a large rule base is a tedious task, and it will take a substantial amount of memory space and reasoning time. Because of a long reasoning time the response of such a generic PID-type FLC will be too slow and hence not suitable when a fast response is desired e.g. for a flexible-link manipulator.

A variety of approaches have been made to overcome the problems of PID controllers in [26]. Kwok et al. have considered a novel means of decomposing a PID controller into a fuzzy PD controller in parallel with various types of fuzzy gains, fuzzy integrators, fuzzy PI controller and deterministic integral control [27, 28, 29]. A fuzzy PD controller in parallel with a fuzzy PI controller will still require $R = n \times n + n \times n = 2n^2$, i.e., 50 rules in case of $n = 5$ linguistic labels. The first set of rule-base used for PD-type and the second set of rule-base used for PI-type FLC.

A further reduction of rule base is possible if the controller is switched from PD- to absolute PI-type after a certain period of time. In that case only one set of rules, $5 \times 5 = 25$ rules for each type of controller, will be executed at a time and thus the executed rules in a controller rule base will be reduced to only 25 rules for 5 linguistic labels in each input variable. Having been impressed with this idea, a switching type FLC is developed for the flexible-link manipulator where a PD-type FLC is executed first and then switched to a PI-type FLC [4]. The block diagram of this switching PD-PI-type controller is shown in Figure 4. The state variables used in PD-PI-type FLC are the same as in equations (1) and (8) and k_P, k_D, k_I, and k_C are the proportional, differential, integral and controller gain coefficients (or scaling factors) respectively.

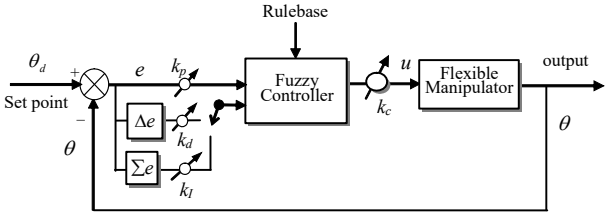


Figure 4: Block diagram of a PD-PI-type FLC system.

The data obtained by experimentation on the PD-PI FLC can be split into two separate data sets which represent change of error during PD control, i.e., before switching point and sum of error during PI control, i.e., for the rest of the time. The change of error before switching point and sum of error after switching point is plotted over time in Figure 5. As can be seen in Figure 5 the range of change of error and sum of error are within such suitable interval that they can be brought within a common universe of discourse. In FLC design, the actual values of the inputs do not matter, rather the MF for each linguistic variable to be defined is important. Therefore, the aim is to unify the MFs for change of error and sum of error so that a further simplification of the rule-base can be achieved in designing an FLC. Now the initial universes of discourse for change error and sum of error are chosen within the same interval [–25, +25]. This enables the FLC to use a single rule-base for the both parts of the controller.

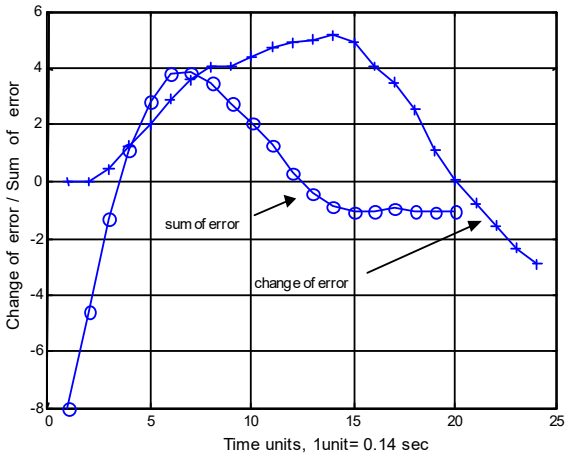


Figure 5: Change and sum of error within a common universe of discourse.

Table 2: Rule base for PD-PI-type FLC.

Error	Change/Sum of error				
	NB	NS	ZO	PS	PB
NB	PB	PB	PB	PS	ZO
NS	PB	PS	ZO	ZO	NS
ZO	PS	ZO	ZO	ZO	NS
PS	PS	ZO	ZO	NS	NB
PB	ZO	NS	NB	NB	NB

A common rule-base, shown in Table 2, was designed and used for both the PD- and PI-type controllers. This actually demands the MFs of the fuzzy sets for change of error and sum of error to be re-adjusted as they were forced to deviate and were merged within a common universe of discourse. Tuning of the MFs becomes more important if a merging procedure is used to reduce the number of fuzzy rules. One possible way is learning of the parameters of the fuzzy sets. In many cases, tuning the scaling factors or adjusting the membership functions can lead to the same result. Adjustment of membership functions requires learning of several parameters and hence scaling factor tuning is a much simpler task than adjusting the parameters [11].

IV. INTEGRATION OF FUZZY LOGIC, NEURAL NETWORKS AND GENETIC ALGORITHMS

The MFs, which were shifted from their original universe of discourse by merging procedure, are now re-adjusted by tuning the scaling factor k_d and k_I using a neural network. There is no need to re-adjust the MF for error, since it is the same in both types of controller. For simplicity the scaling factor k_p is not tuned and hence eliminated from equation (1) and (8). Dividing both sides of equations (1) and (8) by k_p yields

$$k_c^{'} \cdot u = e + k_d^{'} \cdot \Delta e \tag{11}$$

$$k_c^{'} \cdot u = e + k_I^{'} \cdot \sum e \tag{12}$$

where $k_d' = \frac{k_d}{k_p}$, $k_I' = \frac{k_i}{k_p}$ and $k_c' = \frac{k_c}{k_p}$ are the new differential, integral and controller gain coefficients of the PD-PI-type controllers. This can be done by determining suitable parameters, or by approximating the MFs with an NN. Modern neuro-fuzzy systems are often represented as multi-layer feedforward NNs [12]. The ANFIS model for example implements a Sugeno-type fuzzy system in a network structure, and applies a mixture of backpropagation algorithms and least squares procedure to train the system [13]. The problem associated with these types of neuro-fuzzy models is that they sometimes are not as easy to interpret for Mamdani-type fuzzy systems [14].

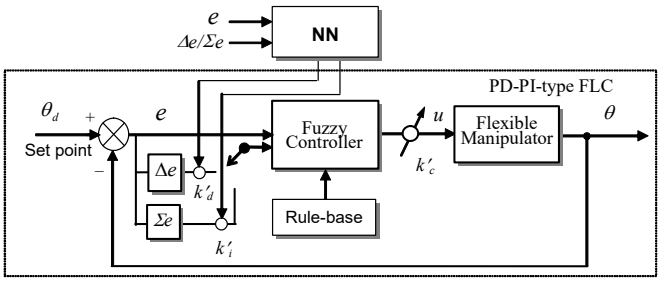


Figure 6: Learning scaling factors using NN.

Simulation results in [15] with tuned membership functions show a marginal improvement in transient response of a second-order linear process, where tuning has resulted in asymmetric membership functions (triangular) with unequal base for e. To be more specific, the width of membership functions increased around $e = 0$. Such membership functions contradict the usual practice where the membership functions take narrow width and become more crowded near the origin to provide increased sensitivity at steady state [16],[17]. Thus, the purposed MFs tuning scheme cannot guarantee improved performance under load disturbance, which is a very important criterion for performance evaluation of a control system. Moreover, a training scheme such as backpropagation algorithm is bounded by its input-output data set though it is minimising the objective function during training and does not guarantee any improved performance of the controller. Furthermore, the use of multilayer perceptron could simply exhaust the system by calculating exponential terms in the network, causing very slow response of the system. Hence scaling factor tuning is a much simpler task than adjustment of parameters. A single neuron network with non-linear activation function will be used to tune the scaling factors k_d' and k_I' and k_c' will be chosen by heuristic rule. A block diagram of the PD-PI FLC with modified scaling factors and neural network learning scheme is shown in Figure 6.

A. SIGMOID FUNCTION SHAPE LEARNING

Nonlinearity can be represented with suffient number of hidden layers with fixed activation function. However, many parameters such as the optimum shape of the sigmoid function are determined by trial and error in most of the cases. This limits advanced application of neural networks. There have been few studies on the optimum shape of the sigmoid function. Yamada and Yabuta proposed an auto-tuning method for the sigmoid function shape in order to apply it to a servo control system. Their method is based on the steepest descent method and confirmed the characteristics and practicality of the method with simulation results [18].

The usual sigmoid function *f(x)* is defined as

$$f(x) = \frac{1 - e^{-ax}}{1 + e^{-ax}} \tag{13}$$

where x is the network output and a defines the shape of the activation function. The shape of sigmoid function is shown in Figure 7 for different values of a.

This type of activation function is characterised by its gain (slope) and seriously affects the control characteristics. If this gain tuning is used in control applications, the plant output may become unstable in certain cases. When the usual sigmoid function is used only in the hidden layer, sigmoid function shape tuning is the same as weight tuning. A mathematical proof is given in [18]. Therefore, sigmoid function shape tuning in a single neuron network can contribute more in improving performance of the controller.

B. GA-BASED TRAINING OF NEURAL NETWORK

Interest in training neural networks using genetic algorithms has been growing rapidly in recent years [19]-[22]. The interest in this study is to explore possible benefits arising from the interactions between neural networks and evolutionary search procedures. One of the most popular training algorithms for feed forward NNs is backpropagation (BP). BP is a gradient descent search algorithm, which is based on minimization of the total mean squared error between actual output and a desired output. This error is used to guide BP's search in the weight space. However, the BP algorithm suffers from a

number of problems. It is very often trapped in local minima and is very inefficient in searching for global minimum of the search space. BP's speed and robustness are sensitive to several parameters of the algorithm and the best parameters to use appear to vary from problem to problem [19]. Shape of the sigmoid function in BP learning is chosen mostly heuristically or by trial and error. There are several basic arguments suggesting that applying GAs to NN weight optimisation is advantageous. GAs have the potential to produce a global minimum of the weight space and thereby avoid local minima. It is also an advantage to apply GAs to problems where gradient information is either not available or costly to obtain or there is non-differentiable node transfer function involved. In this specific issue the parameter of the sigmoid function shape can be easily included in the learning process.

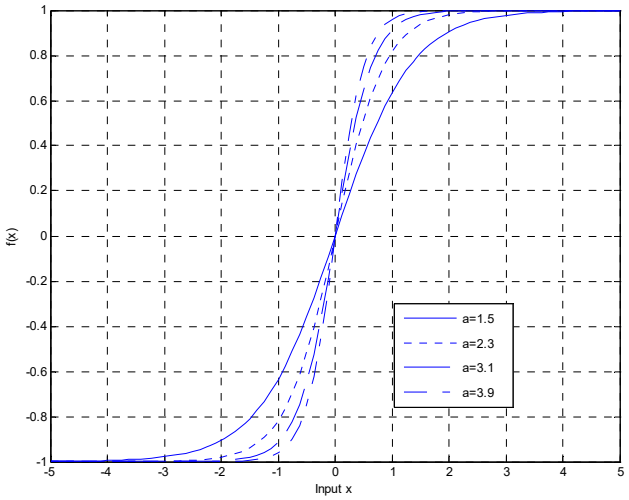


Figure 7: Shape of sigmoid function for different values of a.

A block diagram of the GA-based neuro-fuzzy control system is shown in Figure 8, which incorporates a single neuron network. The activation function is defined in equation (13) and the parameter a defines the shape of the sigmoidal function. The use of different shapes of sigmoidal function can lead to different weights and biases during learning with the backpropagation algorithm, which is experienced in a previous investigations. That is, the shape of sigmoidal functions should be fixed during execution of the backpropagation algorithm.

A mechanism is sought to learn the weights, bias and the parameter a of the network. Two approaches present themselves instantly for this purpose: firstly, backpropagation algorithm learning of weights and bias and trial and error method for parameter a and secondly, genetic algorithm based learning of the weights, biases and the parameter a simultaneously. It seemed somewhat tedious and slow because of the computation involved in updating the weights and bias for each parameter a, which prolonged the computation in each learning epoch in BP. This study aims to investigate the possible benefit of learning the shape of sigmoid function together with the weights and bias, which will reduce the computing time greatly and can exploit the non-linearity involved in the system. Genetic algorithm can best serve such a learning objective. In this section, the weights (w_1, w_2), bias b and parameter a of the neural network, shown in Figure 9, are learnt by genetic algorithm. The chromosome representation is straightforward and shown in Figure 10.

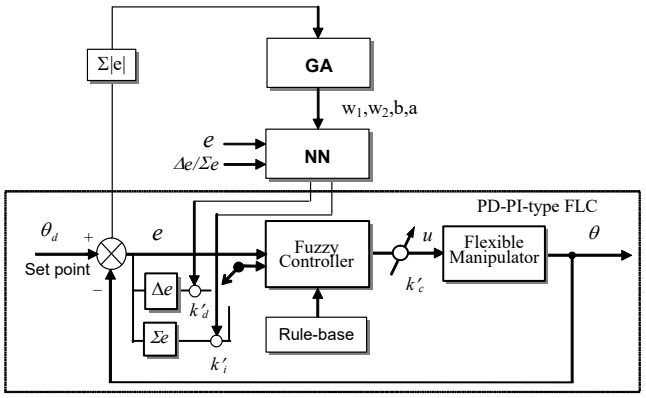


Figure 8: Block diagram of the GA-based neuro-fuzzy control system.

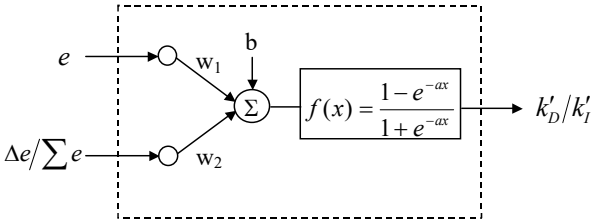


Figure 9: Single neuron network with non-linear activation function.

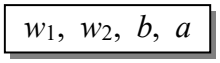


Figure 10: String representation of chromosome of the neural network.

The objective function is defined as

$$J = \sum_{k=1}^{N} |e(k)| \qquad (14)$$

where $e(k)$ is hub-angle error and N is some reasonable number of time units by which the system can be assumed to have settled close to steady state. The evaluation of the objective function is performed by applying the controller on the experimental manipulator.

Experience from the experiments in a previous research says that the values of the weights-bias and sigmoid function shape parameter are within the ranges $[-0.5, +0.22]$ and

$[2.0, 2.6]$ respectively. Considering these results a population of 10 chromosomes is initialised within the ranges of values.

Elitists single point crossover operation is used. Elitism is an optional characteristic of genetic algorithm. When used, it makes sure that the fittest chromosome of a population is passed on to next generation unchanged. In this investigation an extended form of elitism is used where best m ($m = 8$ in this study) chromosomes are retained from N chromosomes, N is the population size ($N = 10$ in this study). In other words, the worst two chromosomes are replaced by two offsprings created by crossing two best chromosomes in the population.

The crossover operation can suffer from two well-known problems: firstly, crossover operation, when applying genetic algorithms to neural networks, can result in a competing conventions problem. Competing conventions prevent standard crossover operation to produce useful offsprings. Also the number of competing conventions grows exponentially with respect to number of hidden neurons. Secondly, crossover operation may not produce new chromosomes for a small size of population in higher generations. Mutation operation can thus strike a balance to these problems encountered by crossover. Montana and Davis used three different types of mutation operators [20] to overcome such problems. In this study, a mutation operation with a higher mutation rate is applied to genetic algorithm based learning of the neural network. A randomly chosen value from the offspring is mutated with a mutation rate of 0.5. This mutation rate will ensure changes of at least two values in the offspring chromosome.

V. Experimental Results

A population of 10 chromosomes is initialised within the range of $[-0.5, +0.5]$ and $[0, 4]$ for weights and bias and for the parameter a respectively. The practical constraints of applying the GA-based Neuro-Fuzzy controller to the flexible-link manipulator involved how to evaluate the objective function. The easiest way is to operate the Neuro-Fuzzy controller repeatedly and evaluate its performance (evaluation function) by calculating the absolute sum of error. The population is tested up to the 13th generation. Figure 11 shows the system responses of the best 4 individuals in 1st generation. In earlier generations, some of the chromosomes needed a longer time to settle, and the chromosomes were required to evaluate for 250 iterations in the program loop. This has caused some ties of the fitness values. To help resolve the ties, only 50 iterations were evaluated in the later generations. Figure 12 shows the system response with the best individual in generations 5, 7 and 9. Figure 13 shows the system response with the best individual in generations 11, 12 and 13. This shows significant improvement of the performance in respect of rise time, maximum overshoot and settling time. Figure 14 shows fitness convergence of GA over generations. The weights, bias and the parameter a after learning were found to be $w_1 = -0.029$, $w_2 = 0.01$, $b = 0.23$ and $a = 2.18$.

The performance of a three-neuron network with linear activation function and with non-linear activation function was also verified. The performance of the three-neuron network is shown in Figure 15. The network was trained using the backpropagation algorithm. The performance degradation of the system is obvious and possibly caused by the excessive calculation of weights and biases updates required in backpropagation algorithms.

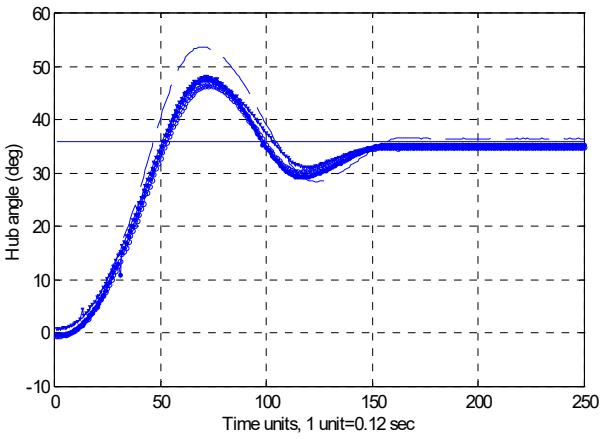


Figure 11: System response using best 4 individuals in generation 1.

VI. Conclusion

Experimentations showed improved performance of the system response using the proposed controller over the PD-, PI-, and PD-PI-type FLC. It has also been demonstrated that the sigmoidal function and its shape can represent the nonlinearity of the system. There are several algorithms like backpropagation that learn the weights and biases of a neural network but very few algorithms that learn the shape of the sigmoidal function. Genetic algorithm can be used to learn the weights, biases and shape of the sigmoidal function of the neural network simultaneously.

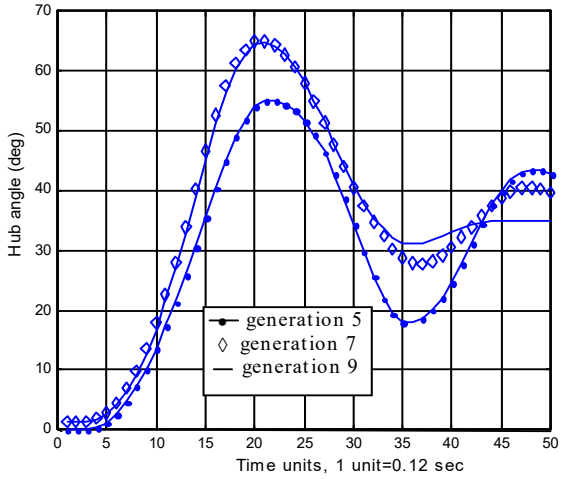


Figure 12: System response of best individuals in generations 5, 7 and 9.

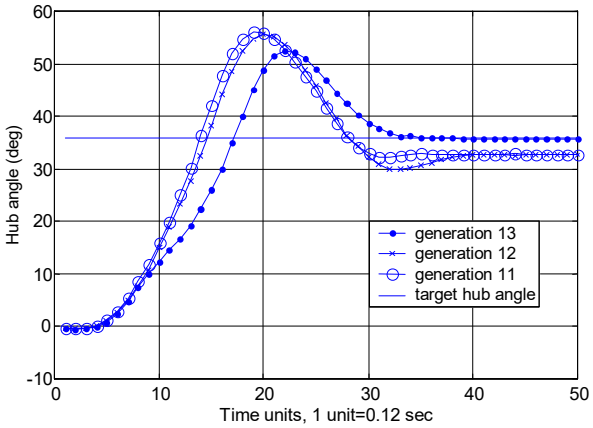


Figure 13: System response with best individuals in generations 11, 12 and 13.

Table 3: Comparison of response parameters (1 Time unit = 0.12 sec)

Generations	Rise time (Time units)	Max overshoot (deg)	Settling time (Time units)	Steady state error (deg)
11	13	56.13	35	3.24
12	14	55.55	39	3.34
13	17	52.51	34	0.33

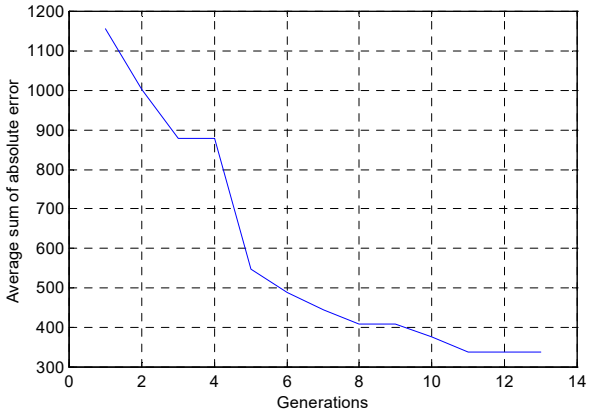


Figure 14: Convergence of the fitness.

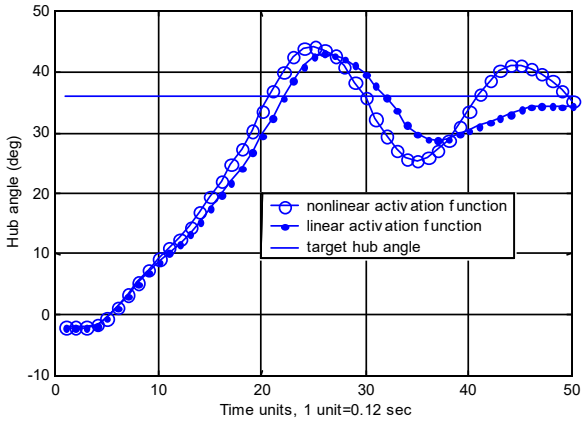


Figure 15: System response using Neuro-fuzzy controller with 3-neuron network

REFERENCES

[1]. Chao, C.-T. and Teng, C.-C. "A PD-like self-tuning fuzzy controller without steady-state error", *Fuzzy Sets and Systems,* Vol. 87, 1997, pp. 141-154.

[2]. Chung, H.-Y., Chen, B.-C. and Lin, J.-J. "A PI-type fuzzy control with self-tuning Scaling Factors", *Fuzzy Sets and Systems,* Vol. 93, 1998, pp. 23-28.

[3]. Siddique, M.N.H. and Tokhi, M.O. "PD-PI Fuzzy Logic Control of Flexible-link Manipulators", *Proceedings of 7th UK Workshop on Fuzzy Systems*, Sheffield, 26-27 October, 2000, pp. 36-41.

[4]. Siddique, M.N.H. "Intelligent Control of Flexible-link Manipulators", *PhD Thesis*, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, England, 2000, UK.

[5]. Nauck, D and Kruse, R "A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error

backpropagation", *Proceeding of IEEE International Conference on Neural Networks*, 1993, pp. 1022-1027.

[6]. Lin, .C.-T. and C.S.G. Lee "Neural-Network-based fuzzy logic control and decision system", *IEEE Transaction on Computer*, vol. 40, 1991, pp. 1320-1336.

[7]. Lin, .C.-T. and Lee, C.S.G. "Real-time supervised structure-parameter learning for fuzzy neural network", *Proceeding of IEEE International Conference on Fuzzy Systems*, 1992, pp. 1283-1290.

[8]. Lin, .C.-T. and Lee, C.S.G. "Reinforced structure-parameter learning for neural-network-based fuzzy logic control systems", *Proceeding of IEEE International Conference on Fuzzy Systems*, 1993, pp. 88-93.

[9]. Lin, C.-T. and Lee, C.S.G. "A neural fuzzy control system with structure and parameter learning", Fuzzy *Sets and Systems*, vol. 70, 1995, pp. 183-212.

[10]. Buckley, J.J. and Hayashi, Y. "Neural networks for fuzzy systems", *Fuzzy Sets and Systems*, Vol. 71, 1995, pp.265-276.

[11]. Chen, M. and Linkens, D.A. "A hybrid neuro-fuzzy controller", *Fuzzy Sets and Systems*, Vol. 99, 1998, pp. 27-36.

[12]. Jang, J.-S. R. "ANFIS: Adaptive-Network-based fuzzy inference systems", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, 1993, pp. 665-685.

[13]. Nauck, D., Klawonn, F. and Kruse, R. "Foundations of Neuro-Fuzzy Systems", John Wiley and Sons, Chichester, NY, Weinheim, Brisbane, Singapore, Toronto, 1997.

[14]. Zheng, L., "A practical guide to tune of proportional and integral (PI) like fuzzy controller", *Proceedings of 1st IEEE International Conference on Fuzzy Systems*, San Diego, CA, March, 1992, pp. 633-641.

[15]. Drainkov, D.; Hellendorn, H. and Reinfrank, M. "An Introduction to Fuzzy Control", New York, Springer-Verlag.

[16]. Harris, C.J.; Moore, C.G. and Brown, M. "Intelligent Control – Aspects of Fuzzy Logic and Neural Nets", Singapore, World Scientific, 1993.

[17]. Yamada, T. and Yabuta, T. "Neural Network Controller using Auto-tuning Method for Nonlinear Functions"; *IEEE Trans. on Neural Networks;* vol.3, No.4, 1992, pp. 595-601.

[18]. Caudell, Thomas P. and Dolan, Charles P. "Parametric Connectivity: Training of Constrained Networks using Genetic Algorithms", *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA '89)*, George Mason University, June 4-7, 1989, pp. 370-374.

[19]. Montana, D. J. and Davis, L. "Training Feedforward Neural Network using Genetic Algorithms", *Proceedings of 11th International Joint Conference on Artificial Intelligence*, San Mateo, CA, Morgan Kaufmann, 1989, pp. 762-767.

[20]. Whiteley, D.; Starkweather, T. and Bogart, C. "Genetic Algorithms and neural Networks: Optimizing Connections and Connectivity", Parallel Computing, vol. 14, 1990, pp. 347-361.

[21]. Yam, Jim Y. F. and Chow, Tommy W.S. "Extended Least Squares Based Algorithm for Training Feedforward Networks", *IEEE Trans. on Neural Networks*, Vol. 8, No. 3, 1997, pp. 806-810.

[22]. Lee, J. "On Methods for Improving Performance of PI-type Fuzzy Logic Controllers"; *IEEE Trans. on Fuzzy Systems;* vol.1, No.1, 1993, pp. 298-301.

[23]. Mudi, R.K. and Pal, N.R. "A robust self-tuning scheme for PI- and PD-type fuzzy controllers", *IEEE Trans. on Fuzzy Systems*, vol. 7(1), 1999, pp. 2-16.

[24]. Chao, C.-T. and Teng, C.-C. " PD-like self-tuning fuzzy controller without steady-state error", *Fuzzy Sets and Systems,* Vol. 87, 1997, pp. 141-154.

[25]. Chung, H.-Y., Chen, B.-C. and Lin, J.-J. "A PI-type fuzzy control with self-tuning Scaling Factors", *Fuzzy Sets and Systems,* vol. 93, 1998, pp. 23-28.

[25]. Tzafestas, S. and Papanikolopoulos, N.P., "Incremental fuzzy expert PID control", *IEEE Transaction on Industrial Electronics*, vol. 37, 1990, pp. 365-371.

[26]. Brehm, T., "Hybrid fuzzy logic PID controller", *Proceedings of 3rd IEEE Conference on Fuzzy Systems*, Vol.3, 1994, pp. 1682-1687.

[27]. Kwok, D.P., Tam, D., Li, C.K., and Wang, P., "Linguistic PID controllers", *Proceeedings of 11th IFAC World Congress*, Tallin, USSR, 1990, pp. 192-197.

[28]. Kwok, D.P., Tam, D., and Li, C.K., "Analysis and design of fuzzy PID control systems", *Proceedings of IEE Control '91 Conference*, Heriot Watt University, Edinburg, 1991, pp. 955-960.

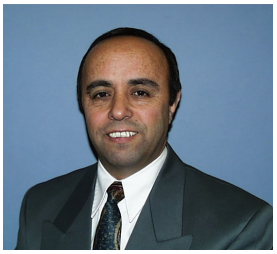
[29]. Harris, C.J., Moore, C.G., and Brown, M., "Intelligent control: Aspects of fuzzy logic and neural nets", *World Scientific* (World scientific series vol. 6), Singapore, NJ, London, Honkong, 1993.

MNH Siddique graduated from Dresden University of Technology, Germany in Cybernetics and Automation Engineering in 1989. He obtained M. Sc. Eng. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in 1995. He received his PhD in intelligent control from the Department of Automatic Control nd Systems Engineering, University of Sheffield, England in 2003.

He has been a Lecturer in School of Computing and Intelligent Systems, University of Ulster at Magee, UK since 2001. Prior to that he was with Computer Science and Engineering Discipline, Khulna University, Bangladesh since 1991.

Dr Siddique's research interests relate to intelligent systems, computational intelligence, perception-based system modelling, evolutionary robotics and

neuro-fuzzy-evolutionary hybrid techniques. Applications of the research include control systems, robotics, pattern recognition, vision systems, and signal processing. Dr Siddique has published some 50 journal/refereed conference papers, book chapter and book. He has served as committee members and chairs of a number of national and international conferences. He is on the executive committee of the IEEE SMC UK-RI Chapter.



M. Osman Tokhi obtained his BSc (Electrical Engineering) from Kabul University (Afghanistan) in 1978 and PhD from Heriot-Watt University (UK) in 1988. He has worked as lecturer and Senior Lecturer in Kabul University, Glasgow College of Technology (UK) and the University of Sheffield (UK) and as sound engineer in industry. He is currently employed as Reader in the Department of Automatic Control and Systems Engineering, the University of Sheffield (UK). His main research interests include active control of noise and vibration, adaptive/intelligent and soft computing techniques for modelling and control of dynamic systems, high-performance computing for real-time signal processing and control, and biomedical applications of robotics and control. He has over 350 publications in print in these areas including textbooks, journal and conference papers