Control of nonlinear systems with a linear state-feedback controller and a modified neural network tuned by genetic algorithm

H.K. Lam, S.H. Ling, H.H.C. Iu, C.W. Yeung, and F.H.F.Leung

***Abstract*— This paper presents the control of nonlinear systems with a neural network. In the proposed neural network, the neuron has two activation functions and exhibits a node-to-node relationship in the hidden layer. By using a genetic algorithm with arithmetic crossover and non-uniform mutation, the parameters of the proposed neural network can be tuned. Application examples are given to illustrate the merits of the proposed neural network.**

I. Introduction

Neural network has been proved to be a universal approximator [1, 3-4]. A 3-layer fully-connected feed-forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy in a compact domain. Neural networks are widely applied in areas such as prediction [3, 5], system modeling and control [1, 9, 14]. Owing to its particular structure, a neural network is good in learning [2] using some learning algorithms such as Genetic Algorithm (GA) [2] and back propagation [1, 4]. In general, the processing of a traditional feed-forward neural network is done in a layer-by-layer manner. In this paper, by introducing a node-to-node relationship [25] in the hidden layer of the neural network, a better performance can be obtained.

GA is a directed random search technique. It is widely applied in optimization problems [1-2] where the number of parameters is large and the analytical solutions are difficult to obtain. GA can help find the optimal solution over a domain globally [1-2]. It has been applied in different areas such as fuzzy control [6-7, 14], neural control, path planning [8], prediction [3, 5], modeling and classification [9] etc.

Control of nonlinear systems is a challenging task. To tackle such class of systems, a lot of algorithms have been proposed. Sliding mode control [11] and switching controller [20] have been proposed. Due to the switching property of the control signal, an unwanted chattering effect will be seen at the system outputs. Fuzzy controller [15-19, 21-22], which is good in handling ill-defined plants, have been used widely. To obtain the parameters of the fuzzy controller [24], GA has been employed for parameter tuning. Neural networks with parameters tuned by GA can also be found to perform the control task [12-13]. Apart from the above nonlinear controllers, linear control techniques [11] can also be employed to obtain a linear controller for nonlinear systems.

H.K. Lam is with the Division of Engineering, The King's College London, Strand, London, WC2R 2LS, United Kingdom (e-mail: hak-keung.lam@kcl.ac.uk).

S.H. Ling and H.H.C Iu are with School of Electrical, Electronic and Computer Engg, University of Western Australia, WA6009, Australia (e-mail: steve@ee.uwa.edu.au and herbert@ee.uwa.edu.au).

C.W. Yeung and F.H.F. Leung are with the Centre for Signal Processing, Dept. of Electronic and Information Engg., The Hong Kong Polytechnic University, Hung Ham, Hong Kong (e-mail: encwy@ eie.polyu.edu.hk and enfrank@inet.polyu.edu.hk).

For instance, a linear state feedback controller can be designed based on the linearized model of the nonlinear plant. This method is simple and the complexity of the controller is comparatively low. However, the operating range of the controller is small.

In this paper, a neural network combined with a linear state feedback controller is proposed to control a nonlinear system. The linear state feedback controller is obtained based on the linearized model. A modified neural network [25] is proposed. Two different activation functions are used in the neuron and a node-by-node relationship is proposed in the hidden layer. GA is employed to tune the parameters of the neural network by optimizing the system performance. By combining the proposed GA-tuned neural network with the linear state feedback controller, their individual advantages can also be combined. As the linear state feedback controller can provide good performance around the operating point, with the proposed neural network, the operating range of the resultant controller can be extended. Application examples on an XOR problem and stabilizing an inverted pendulum will be given to illustrate the merits of the proposed algorithm.

II. Control Algorithm for the Nonlinear Systems

In this paper, a nonlinear system of the following form is considered.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))\mathbf{u}(t) \tag{1}$$

where $\mathbf{x}(t) \in \Re^{n\times 1}$ is the system state vector, $\mathbf{f}(\mathbf{x}(t)) \in \Re^{n\times n}$ and $\mathbf{g}(\mathbf{x}(t)) \in \Re^{n\times m}$ are the known or uncertain system and input matrices respectively, $\mathbf{u}(t) \in \Re^{m\times 1}$ is the input vector. A controller of the following form is proposed for the nonlinear system of (1).

$$\mathbf{u}(t) = \mathbf{u}_l(t) + \mathbf{u}_n(t) \tag{2}$$

where $\mathbf{u}_l(t) \in \Re^{m\times 1}$ and $\mathbf{u}_n(t) \in \Re^{m\times 1}$ denote the control signals of the linear state feedback controller and neural network respectively. The linear state feedback controller is defined as follows.

$$\mathbf{u}_l(t) = \mathbf{G}\mathbf{x}(t) \tag{3}$$

where $\mathbf{G} \in \Re^{m\times n}$ denotes the constant feedback gain matrix which is designed based on the linearized model of (1). When the parameters of the nonlinear plant are uncertain, their nominal values are used to obtain the linearized model of (1). The control signal $\mathbf{u}_n(t)$ contributed by the proposed neural network will be discussed in the next section. A block diagram of the closed-loop system is shown in Fig. 1.

III. Modified Neural Network

In this section, a modified neuron model of a neural network is presented. The outputs of this proposed neural

1-4244-1340-0/07/$25.00 ©2007 IEEE

network will be used to control the nonlinear plant of (1). Fig. 2 shows the proposed neuron. It has two activation functions to govern the input-output relationship of the neuron. The two activation functions are called static activation functions (SAF) and dynamic activation functions (DAF). For the SAF, the parameters are fixed and its output depends on the input of the neuron. For the DAF, the parameters of the activation function depend on the outputs of other neurons and its SAF. With this proposed neuron, the connection of the proposed neural network is shown in Fig. 3, which is a three-layer neural network. A node-to-node relationship is introduced in the hidden layer. It should be noted that the parameters of the SAF are constant. The parameters of the DAF are obtained from other two neuron outputs, which are variable.

A. The neuron model

The proposed neuron model consists of two activation functions (SAF and DAF). We consider the SAF first. Let v_{ij} be the synaptic connection weight from the i-th input component x_i to the j-th neuron. The output κ_j of the j-th neuron's SAF is defined as,

$$\kappa_j = net_s^j\left(\sum_{i=1}^{n_{in}} x_i v_{ij}\right),\ i = 1, 2, \ldots, n_{in}, j = 1, 2, \ldots, n_h \tag{4}$$

where n_{in} denotes the number of input; n_h denotes the number of hidden nodes; and $net_s^j(\cdot)$ is a static activation function. The activation function is defined as,

$$net_s^j\left(\sum_{i=1}^{n_{in}} z_i v_{ij}\right) = \begin{cases} e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j\,2}}} - 1 & \text{if } \sum_{i=1}^{n_{in}} z_i v_{ij} \le m_s^j \\ 1 - e^{\frac{-\left(\sum_{i=1}^{n_{in}} z_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j\,2}}} & \text{otherwise} \end{cases},\ j = 1, 2, \ldots, n_h \tag{5}$$

where m_s^j and σ_s^j are the static mean and static standard deviation for the j-th SAF respectively. The parameters (m_s^j and σ_s^j) are fixed after the training processing. Thus, the activation function is static. The output of the SAF depends on the inputs of the neuron only. By using the proposed activation function in (5), the output value is ranged from −1 to 1. It should be noted that $net(f) \to 1$ as $f \to \infty$ and $net(f) \to -1$ as $f \to -\infty$.

Considering the DAF, the neuron output z_j of the j-th neuron is defined as,

$$z_j = net_d^j(\kappa_j, m_d^j, \sigma_d^j),\ j = 1, 2, \ldots, n_h \tag{6}$$

where $net_d^j(\cdot)$ is the activation function of the DAF. The activation function is defined as follows,

$$net_d^j(\kappa_j, m_d^j, \sigma_d^j) = \begin{cases} e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j\,2}}} - 1 & \text{if } \kappa_j \le m_d^j \\ 1 - e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j\,2}}} & \text{otherwise} \end{cases},\ j = 1, 2, \ldots, n_h \tag{7}$$

where

$$m_d^j = p_{j+1,j} \times \kappa_{j+1} \tag{8}$$

$$\sigma_d^j = p_{j-1,j} \times \kappa_{j-1} \tag{9}$$

m_d^j and σ_d^j are dynamic mean and dynamic standard deviation for the j-th DAF. κ_{j-1} and κ_{j+1} represent the SAF's output of the j−1-th and j+1-th neurons respectively. $p_{j+1,j}$ denotes the weight of the link between the j+1-th node and the j-th node, and $p_{j-1,j}$ denotes the weight of the link between the j−1-th node and the j-th node. It should be noted from Fig.1 that if $j = 1$, $p_{j-1,j}$ is equal to $p_{n_h,1}$ and if $j = n_h$, $p_{j+1,j}$ is equal to p_{1,n_h}.

Unlike that of the SAF, the activation function of the DAF is dynamic as the parameters of its activation function depend on the outputs of the j−1-th and j+1-th neurons. Referring to (4) to (7), the input-output relationship of the proposed neuron is as follows,

$$z_j = net_d^j\left(net_s^j\left(\sum_{i=1}^{n_{in}} x_i v_{ij}\right), m_d^j, \sigma_d^j\right),\ j = 1, 2, \ldots, n_h \tag{10}$$

B. Connection of the proposed neural network

The proposed multi-input multi-output (MIMO) neural network shown in Fig. 3 is presented in this section. It has three layers with n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer. In the hidden layer, the neuron model presented in the previous section is employed. A node-to-node relationship is introduced in the hidden layer. The output value of the hidden node depends on the neighboring nodes and input nodes. In the output layer, a static activation function is employed. Considering an input-output pair (**x**, **y**), the output of the j-th node of the hidden layer is given by

$$z_j = net_d^j\left(net_s^j\left(\sum_{i=1}^{n_{in}} x_i v_{ij}\right)\right),\ j = 1, 2, \ldots, n_h \tag{11}$$

where v_{ij}, $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots n_h$, denotes the weight of the link between the i-th input and the j-th hidden nodes. The output of the proposed neural network is defined as,

$$y_l = net_o^l\left(\sum_{l=1}^{n_{out}} z_j w_{jl}\right),\ l = 1, 2, \ldots, n_{out} \tag{12}$$

$$= net_o^l\left(\sum_{l=1}^{n_{out}} net_d^j\left(net_s^j\left(\sum_{i=1}^{n_{in}} x_i v_{ij}\right)\right) w_{jl}\right) \tag{13}$$

where w_{jl}, $j = 1, 2, \ldots, n_h$; $l = 1,2, \ldots, n_{out}$, denotes the weight of the link between the j-th hidden and the l-th output nodes; $net_o^l(\cdot)$ denotes the activation function of the output neuron. The activation function of the output node is defined as follows,

$$net_o^l\left(\sum_{j=1}^{n_h} z_j w_{jl}\right) = \begin{cases} e^{\frac{-\left(\sum_{j=1}^{n_h} z_j w_{jl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} - 1 & \text{if } \sum_{j=1}^{n_h} z_j w_{jl} \le m_o^l \\ 1 - e^{\frac{-\left(\sum_{j=1}^{n_h} z_j w_{jl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} & \text{otherwise} \end{cases},\ l = 1,2, \ldots, n_{out} \tag{14}$$

where m_o^l and σ_o^l are the mean and the standard deviation of the output node activation function respectively. The parameters of the proposed modified neural network can be trained by GA. The details about the training processing will be given in the next section. Referring to (2), the control signal contributed by $\mathbf{u}_n(t)$ is defined as,

$$\mathbf{u}_n(t) = w\mathbf{Y}\mathbf{x}(t) \tag{15}$$

where,

$$\mathbf{Y} \in \Re^{m\times n} = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ y_{n+1} & y_{n+2} & \cdots & y_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ y_{(m-1)n+1} & y_{(m-1)n+2} & & y_{m\times n} \end{bmatrix} \tag{16}$$

is a dynamic gain matrix and w is a constant scalar, both are to be tuned. As the outputs of the proposed neural network lie between −1 and 1, the use of w is to scale the range of the network's outputs. It can be seen from (16) that the number of outputs of the proposed neural network is $n_{out} = m\times n$.

IV. Turning of Parameters of the Proposed Neural Network

From (3) and (15), the proposed controller of (2) is as follows.

$$\mathbf{u}(t) = (\mathbf{G} + w\mathbf{Y})\mathbf{x}(t) \tag{17}$$

From (1) and (17), the following closed-loop system is obtained.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))(\mathbf{G} + w\mathbf{Y})\mathbf{x}(t) \tag{18}$$

In (18), the dynamics of the closed-loop is defined. What follows is to determine the parameters of the proposed neural network using GA such that the performance of the closed-loop system of (18) is optimal subject to a defined performance index. The fitness function (performance index) is defined as follows,

$$fitness = \int \mathbf{x}(t)^{\mathrm{T}}\mathbf{W}_x\mathbf{x}(t) + \mathbf{u}(t)^{\mathrm{T}}\mathbf{R}_u\mathbf{u}(t)\mathrm{d}t \tag{19}$$

where $\mathbf{W}_x \in \Re^{n\times n}$ and $\mathbf{R}_u \in \Re^{m\times m}$ are constant semi-positive or positive definite matrices. This fitness function is the performance index used in conventional optimal control [26]. The optimization problem formulated here will be handled by GA. The parameters to be tuned are $[v_{ij}\ \ m_s^j\ \ \sigma_s^j\ \ p_{j+1,j}\ \ p_{j-1,j}\ \ w_{jl}\ \ m_o^l\ \ \sigma_o^l\ \ w]$ which will be used as the chromosome for the GA. The objective is to maximize the fitness value of (19) using GA by searching the values of $[v_{ij}\ \ m_s^j\ \ \sigma_s^j\ \ p_{j+1,j}\ \ p_{j-1,j}\ \ w_{jl}\ \ m_o^l\ \ \sigma_o^l\ \ w]$ for all i, j, l.

The procedure to obtain the proposed controller of (2) using GA can be summarized into the following steps.

1) Obtain the mathematical model of the nonlinear system and its linearized model. If the parameters of the nonlinear system are uncertain, nominal values are used for the linearized model.
2) Obtain the linear state feedback controller for the nonlinear system based on its linearized model.
3) Determine the number of inputs, number of nodes and the inputs (e.g. some of the system states) of the proposed neural network.
4) Obtain the parameters of the neural network and w using GA to optimize the system performance of the closed-loop system.

V. Application Examples

Two application examples will be given in this section, namely, an XOR problem and the stabilization of inverted pendulum.

A. XOR Problem

An XOR problem will be presented to illustrate the merits of the proposed neural network. It is a three-input XOR function which is not linearly separable:

$$\begin{aligned}&(-1,-1,-1)\to -1, (-1,-1,+1)\to +1, (-1,+1,-1)\to +1,\\ &(-1,+1,+1)\to -1, (+1,-1,-1)\to +1, (+1,-1,+1)\to -1,\\ &(+1,+1,-1)\to -1, (+1,+1,+1)\to +1\end{aligned} \tag{20}$$

The three inputs of the proposed neural network are defined as $x_i(t)$, $i = 1, 2, 3$ and $y(t)$ is the network output. The number of hidden nodes (n_h) is set at 3. Referring to (13), the proposed neural network used for the three-inputs XOR classification problem is governed by,

$$y_1(t) = net_o^1\Big(\sum_{j=1}^{3} net_d^j\big(net_s^j\big(\sum_{i=1}^{3} x_i v_{ij}\big)\big)w_{j1}\Big) \tag{21}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{22}$$

$$err = \frac{\sum_{t=1}^{8}\left|y_1^d(t) - y_1(t)\right|}{8} \tag{23}$$

GA with arithmetic crossover and non-uniform mutation [2] is employed to tune the parameters of the proposed neural network of (21). The objective is to maximize the fitness function of (22). The best fitness value is 1 and the worst one is 0. The chromosomes used for the GA are $[v_{ij}\ \ m_s^j\ \ \sigma_s^j\ \ p_{j+1,j}\ \ p_{j-1,j}\ \ w_{j1}\ \ m_o^1\ \ \sigma_o^1]$. The initial values of the parameters of the neural network are randomly generated. The lower and upper bounds for all genes except σ_s^j and σ_o^1 are −1 and 1 respectively. The lower and upper bounds for σ_s^j and σ_o^1 are 0 and 2 respectively. For comparison purpose, a traditional 3-layer feed-forward neural network (3-input-single-output) [1, 4] trained by the same GA is also used to solve the three-input XOR classification problem. The number of hidden nodes of the traditional neural network is 5. (By using 3 hidden nodes for the proposed neural network and 5 hidden nodes for the traditional neural network, the number of parameters to be tuned is the same, which is 26.) The lower and upper bounds for all parameters are −1 and 1 respectively. The number of iterations to train the neural network is 1000, the control parameters of GA with arithmetic crossover and non-uniform mutation, namely the shape parameter, the probabilities of crossover and mutation, are chosen to be 2, 0.7 and 0.2 respectively, and the population size is 20 for both approaches. The simulation results of the proposed and traditional neural networks are tabulated in Table I and shown in Fig. 4. It can be seen from Table I and Fig. 4 that the performance of the proposed neural network is better than that of the traditional one.

B. Inverted Pendulum

An application example of balancing a cart-pole inverted pendulum [20, 23] as shown in Fig. 5 will be given. The design procedure mentioned in the previous section is followed.

1) The dynamic equation of the cart-pole type inverted pendulum system is given by,

$$\ddot{\theta}(t)=\frac{g\sin(\theta(t))-aml\dot{\theta}(t)^2\sin(2\theta(t))/2-a\cos(\theta(t))u(t)}{4l/3-aml\cos^2(\theta(t))} \quad (24)$$

where θ is the angular displacement of the pendulum, g = 9.8m/s^2 is the acceleration due to gravity, $m\in[2\ \ 3]$kg is the mass of the pendulum, $a = 1/(m + M)$, M = 8kg is the mass of the cart, $2l$ = 1m is the length of the pendulum, and u is the force applied to the cart. The objective of this application example is to design a fuzzy controller to close the feedback loop of (24) such that θ = 0 at steady state. (24) can be rewritten in the form of (1) and is shown as follows,

$$\dot{\mathbf{x}}(t)=\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix}=\begin{bmatrix}\dot{\theta}(t)\\ \ddot{\theta}(t)\end{bmatrix}$$
$$=\begin{bmatrix}x_2(t)\\ \dfrac{g\sin(x_1(t))-amlx_2(t)^2\sin(2x_1(t))/2}{4l/3-aml\cos^2(x_1(t))}\end{bmatrix} \quad (25)$$
$$+\begin{bmatrix}0\\ \dfrac{-a\cos(x_1(t))}{4l/3-aml\cos^2(x_1(t))}\end{bmatrix}u(t)$$

2) The linearized model of (25) is as follows,

$$\dot{\bar{\mathbf{x}}}(t)=\begin{bmatrix}\dot{\bar{x}}_1(t)\\ \dot{\bar{x}}_2(t)\end{bmatrix}=\overline{\mathbf{A}}\mathbf{x}(t)+\overline{\mathbf{B}}u(t) \quad (26)$$

where,

$$\overline{\mathbf{A}}=\begin{bmatrix}0 & 1\\ \dfrac{M+m}{Ml}g & 0\end{bmatrix} \quad (27)$$

$$\overline{\mathbf{B}}=\begin{bmatrix}0\\ -\dfrac{1}{Ml}\end{bmatrix} \quad (28)$$

Based on (26), the feedback gain is determined as $\mathbf{G}=[144\ \ 16]$ such that the eigavalues of $\overline{\mathbf{A}}+\overline{\mathbf{B}}\mathbf{G}$ are all located at –4 by considering the nominal value of m = 3kg.

3) The system states $x_1(t)$ and $x_2(t)$ are employed as the inputs of the neural network. From (17), the proposed controller is as follows.

$$u(t)=(\mathbf{G}+w\mathbf{Y})\mathbf{x}(t) \quad (29)$$

4) The GA with arithmetic crossover and non-uniform mutation [2] is employed to tune the values of $[v_{ij}\ \ m_s^j\ \ \sigma_s^j\ \ p_{j+1,j}\ \ p_{j-1,j}\ \ w_{jl}\ \ m_o^1\ \ \sigma_o^1\ \ w]$ of the proposed neural network such that the fitness value of (19) is minimized. $\mathbf{W}_x=\begin{bmatrix}10 & 0\\ 0 & 1\end{bmatrix}$ and $\mathbf{R}_u$ = 5 are chosen arbitrarily. The lower and upper bounds for all genes except σ_s^j and σ_o^1 are –1 and 1 respectively. The lower and upper bounds for σ_s^j and σ_o^1 are 0.5 and 5 respectively. The lower and upper bounds for w are chosen to be 0 and 5000 respectively. The control parameters of the GA with arithmetic crossover and non-uniform mutation, namely the shape parameter, the probabilities of crossover and mutation, are chosen to be 1, 0.8 and 0.1 respectively. The number of iteration for training is 2000. Referring to (19), the simulation period for the inverted pendulum is 1 second. The initial values of the chromosomes (parameters of the neural networks) for training are generated randomly. The fitness values before and after the training under different number of hidden nodes are tabulated in Table II. It can be seen that the best fitness value is 6327.8452 when the number of hidden nodes is 3 (total number of network parameters is 29). It should be noted that Table II shows the best training results among 10 times of training.

The responses of the system states and the control signal are shown in Fig. 6 to Fig. 7 respectively for m = 2kg with the initial system states to be $\mathbf{x}(0)=\begin{bmatrix}\dfrac{1\pi}{45} & 0\end{bmatrix}^{\mathrm{T}}$, $\mathbf{x}(0)=\begin{bmatrix}\dfrac{3\pi}{12} & 0\end{bmatrix}^{\mathrm{T}}$ and $\mathbf{x}(0)=\begin{bmatrix}\dfrac{187\pi}{450} & 0\end{bmatrix}^{\mathrm{T}}$. To test the robustness property of the proposed controller, the value of m is changed to 3kg. The responses of the system states are shown in Fig. 8 and Fig. 9 respectively for m = 3kg under the same conditions. The solid lines show the responses with the proposed control algorithm while the dotted lines show the responses with the linear state feedback controller alone. Under the same initial conditions, the linear state feedback controller cannot stabilize the system. It can be seen from the figures that the proposed controller provides a better performance.

For comparison purpose, traditional three layer fully-connected feed-forward neural networks [1] with different number of hidden nodes are employed to replace the proposed neural network. The logarithmic sigmoid function [1] is employed as the transfer function of the hidden and output nodes. The control parameters of the GA with arithmetic crossover and non-uniform mutation, namely the shape parameter, the probabilities of crossover and mutation, are chosen to be 2, 0.65 and 0.15 respectively for training the traditional neural network. Other conditions are the same as those of the proposed neural network. Table III tabulates the fitness values of the closed-loop system with the traditional neural networks before and after the GA training. They are the best training results among 10 trainings. It can be seen that the best fitness value is 6582.2685 when the number of hidden nodes is 5 (total number of network parameters is 28).

Moreover, the nonlinear plant cannot be stabilized by the traditional neural network under 23 network parameters (3 hidden nodes) while the proposed neural network can when the number of network parameters is more or less the same (21 network parameters, 2 hidden nodes). It can be concluded that the proposed neural network is superior in terms of fitness values and learning ability.

VI. CONCLUSION

The control of nonlinear systems with a linear state feedback controller combined with a neural network has been presented. A modified neural network exhibiting node-to-node relationships among neurons with two transfer functions has been proposed. The genetic algorithm with arithmetic crossover and non-uniform mutation has been employed to tune the parameters of the proposed controller. Application examples on an XOR problem and stabilizing an inverted pendulum have been given.

ACKNOWLEGEMENT

The work described in this paper was partially supported by King's Collage London, U.K., The University of Western Australia, Australia, and a grant from the Hong Kong Polytechnic University (Project No. G-YE92).

REFERENCES

[1] D.T. Pham and D. Karaboga, *Intelligent optimization techniques, genetic algorithms, tabu search, simulated annealing and neural networks*. Springer, 2000.

[2] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, (2n ed.) Springer-Verlag, 1994.

[3] I. Drezga and S. Rahman, "Short-term load forecasting with local ANN predictors," *IEEE Trans. Power System*, vol. 14, no. 3, pp. 844-850, Aug. 1999.

[4] M. Brown and C. Harris, *Neuralfuzzy Adaptive Modeling and Control*. Prentice Hall, 1994.

[5] M. Li, K. Mechrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural network," in *Proc. 5th IEEE International Symposium on Intelligent Control, 1990*, pp.524-528.

[6] Y.S. Zhou and L.Y. Lai, "Optimal design for fuzzy controllers by genetic algorithms," *IEEE Trans., Industry Applications*, vol. 36, no. 1, pp. 93-97, Jan.-Feb. 2000.

[7] K. Belarbi and F. Titel, "Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach," *IEEE Trans., Fuzzy Systems*, vol. 8, no. 4, pp. 398-405, Aug. 2000.

[8] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms," *Electronics Letters*, vol. 36, no. 4, pp. 374-376, Feb. 2000.

[9] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," *IEEE. Trans, Fuzzy Systems*, vol. 8, no. 5, pp. 509–522, Oct. 2000.

[10] A.E. Bryson and Y.C. Ho, *Applied Optimal Control*. Blaisdell, 1969.

[11] J.J.E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[12] S. Park, L.J. Park, and C.H.H. Park, "A neuro-genetic controller for nonminimum phase systems," *IEEE Trans Neural Networks*, vol. 6, no. 5, pp. 1297-1300, 1995.

[13] C.T. Lin and C.P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 30, no. 2, pp. 276-289, 2000.

[14] A. Wu and P.K.S. Tam, "A fuzzy neural network based on fuzzy hierarchy error approach," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 6, pp. 808-816, 2000.

[15] F.H.F. Leung, H.K. Lam, and P.K.S. Tam, "Design of fuzzy controllers for uncertain nonlinear systems using stability and robustness analyses," *System and Control Letters*, vol. 35, no.4, pp. 237-243, 1998.

[16] H.K. Lam and F.H.F. Leung, "Synchronization of uncertain chaotic systems based on the fuzzy-model-based approach," *International Journal of Bifurcation and Chaos*, vol. 16, no. 5, pp. 1435-1444, 2006.

[17] H.K. Lam, F.H.F. Leung, and P.K.S. Tam, "Stable and robust fuzzy control for uncertain nonlinear systems," *IEEE Trans. Syst., Man, and Cybern., Part A: Systems and Human*, vol. 30, no. 6, pp. 825-840, November 2000.

[18] H.K. Lam and F.H.F. Leung, "Fuzzy rule-based combination of linear and switching state-feedback controllers," *Fuzzy Sets and Systems*, vol. 156, no. 2, pp.153-184, Dec. 2005.

[19] H.K. Lam, F.H.F. Leung and P.K.S. Tam, "Nonlinear state feedback controller for nonlinear systems: Stability analysis and design based on fuzzy plant model," *IEEE Trans. Fuzzy Systems*, vol. 9, no. 4, pp. 657-661, August, 2001.

[20] H.K. Lam, F.H.F. Leung, and Y.S. Lee, "Design of a switching controller for nonlinear systems with unknown parameters based on a fuzzy logic approach," *IEEE Trans. Syst., Man and Cybern, Part B: Cybernetics*, vol. 34, no. 2, pp. 1068-1074, April 2004.

[21] H.K. Lam and F.H.F. Leung, "Fuzzy controller with stability and performance rules for nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, no. 2, pp. 147-163, Jan. 2007.

[22] H.K. Lam, F.H.F. Leung, and P.K.S. Tam, "Design of stable and robust fuzzy controllers for uncertain multivariable nonlinear systems," *Stability Issues in Fuzzy Control*, J. Aracil (ed.) Springer, pp. 125-164, 2000.

[23] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, "Optimal and stable fuzzy controllers for uncertain nonlinear systems based on an improved genetic algorithm," *IEEE Trans. Industrial Electronics*, vol. 51, no. 1, pp. 172-182, Feb. 2004.

[24] H.K. Lam, F.H.F. Leung and P.K.S. Tam, "Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm," *IEEE Trans. Syst., Man and Cybern, Part B: Cybernetics*, vol. 33, no. 2, pp. 250-257, April 2003.

[25] S.H. Ling, F.H.F Leung and H.K. Lam, "An improved genetic algorithm based fuzzy-tuned neural network," *International Journal of Neural Systems*, vol. 15, no. 6, pp. 457-474, Dec. 2005.

[26] B.D.O. Anderson and J. B. Moore, *Optimal control: Linear quadratic methods*. Prentice Hall, 1990.

[27] Y. Lecun, "A learning procedure for asymmetric network," *Cognitiva*, pp.599-604, 1985.

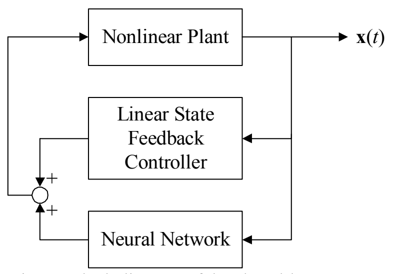

Fig. 1. Block diagram of the closed-loop system.

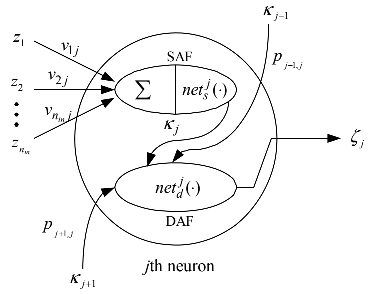

Fig. 2. Model of the proposed neuron.

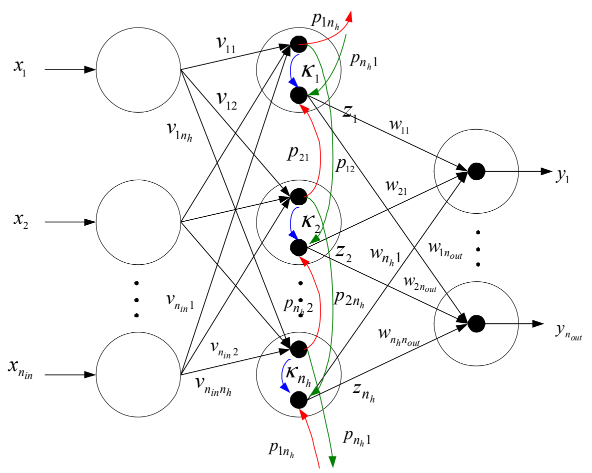

Fig. 3. Connection of the modified neural network.

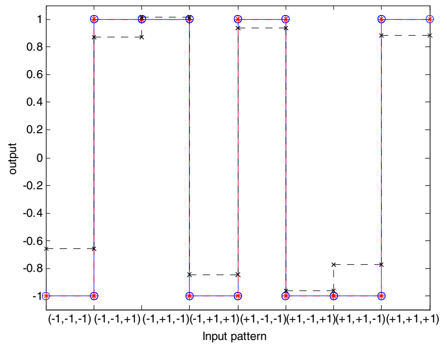

Fig. 4. Output pattern obtained by the proposed neural network (dotted line with '+' mark) and the traditional neural network (dotted line with '×' mark), comparing with the desired output (solid line with 'O' mark) for the XOR problem.

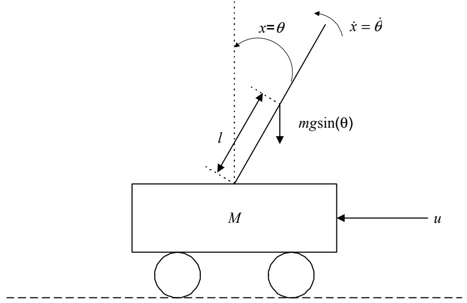

Fig. 5. Cart-pole type inverted pendulum system.

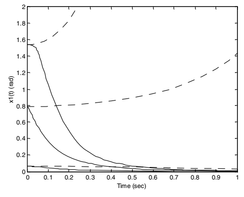

Fig. 6(a). $x_1(t)$

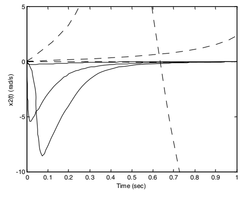

Fig. 6(b). $x_2(t)$

Fig. 6. Responses of $x_1(t)$ with the proposed controller (solid lines) and the linear state feedback controller (dotted lines) for $m = 2$kg.

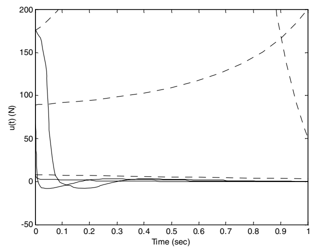

Fig. 7. Control signal of $u(t)$ with the proposed controller (solid lines) and the linear state feedback controller (dotted lines) for $m = 2$kg.

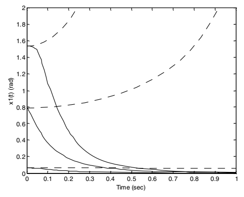

Fig. 8(a). $x_1(t)$

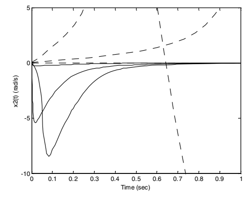

Fig. 8(b). $x_2(t)$

Fig. 8. Responses of $x_1(t)$ with the proposed controller (solid lines) and the linear state feedback controller (dotted lines) for $m = 3$kg.

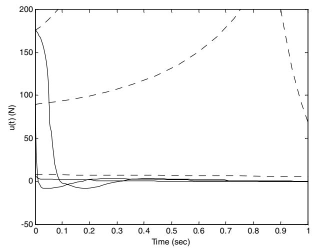

Fig. 9. Control signal of $u(t)$ with the proposed controller (solid lines) and the linear state feedback controller (dotted lines) for $m = 3$kg.

	Fitness Value	*err*
Proposed Neural Network	0.999988	0.002533
Traditional Neural Network	0.880901	0.135201

Table I. Simulation results of the proposed neural network and the traditional neural network for the 3-input XOR classification problem after 1000 times of iteration.

Number of hidden nodes	Number of network parameters	Fitness value before GA training	Fitness value after GA training
2	21	1330783.8098	7091.3400
3	29	1342497.5964	6327.8452

Table II. The best training results among 10 times of training of the proposed neural networks for the inverted pendulum system.

Number of hidden nodes	Number of network parameters	Fitness value before GA training	Fitness value after GA training
3	18	1335085.7452	160178.0768
4	23	1339570.3772	6676.2255
5	28	1327040.0567	6582.2685
6	33	1332190.9745	7395.1501
7	38	1351645.4928	7172.3486

Table III. The best training results among 10 times of training of the traditional neural networks for the inverted pendulum system.