A Direct Adaptive Neural-Network Control for Unknown Nonlinear Systems and Its Application

Jose R. Noriega and Hong Wang, *Member, IEEE*

***Abstract*— In this paper a direct adaptive neural-network control strategy for unknown nonlinear systems is presented. The system considered is described by an unknown NARMA model and a feedforward neural network is used to learn the system. Taking the neural network as a neuro model of the system, control signals are directly obtained by minimizing either the instant difference or the cumulative differences between a setpoint and the output of the neuro model. Since the training algorithm guarantees that the output of the neuro model approaches that of the actual system, it is shown that the control signals obtained can also make the real system output close to the setpoint. An application to a flow-rate control system is included to demonstrate the applicability of the proposed method and desired results are obtained.**

***Index Terms*—Fault-tolerant control, flow-rate systems, multilayer perceptrons, neural networks, nonlinear systems, optimization, stability.**

I. Introduction

RESEARCH on neural-network-based control systems has received a considerable attention over the past several years. This is simply because

1) neural networks have been shown to be able to approximate any nonlinear function defined on a compact set to a prespecified accuracy [1]–[3];
2) most control systems exhibit certain types of unknown nonlinearities.

As a result, many methods have been developed [13]–[16] and successfully applied to some real industrial processes which include chemical processes, manufacturing systems, robotics, aeronautical engines, automatically guided vehicles and papermaking systems, etc. The widely used structures of neural-network-based control systems are similar to those employed in adaptive control, where a neural network is used to estimate the unknown nonlinear system and the controller is then formulated using the estimation results. Therefore the major difference is that the system to be controlled is *nonlinear*. The estimation uses the measured input and output from the system and can be achieved via various types of neural networks, such as multilayer perceptron (MLP) networks, radial basis functions (RBF's) and B-splines networks. However, the formulation of the control signal is not an easy task as it has to be determined from the neural-network model, which approximates the system and is nonlinear with respect to its input arguments [4]–[6]. Indeed, solutions from a set of nonlinear equations are unavoidable in most cases. Therefore,

Manuscript received July 6, 1995; revised August 11, 1997. J. R. Noriega was supported by CONACYT (México).

The authors are with the Department of Paper Science, UMIST, Manchester M60 1QD, U.K.

Publisher Item Identifier S 1045-9227(98)00384-1.

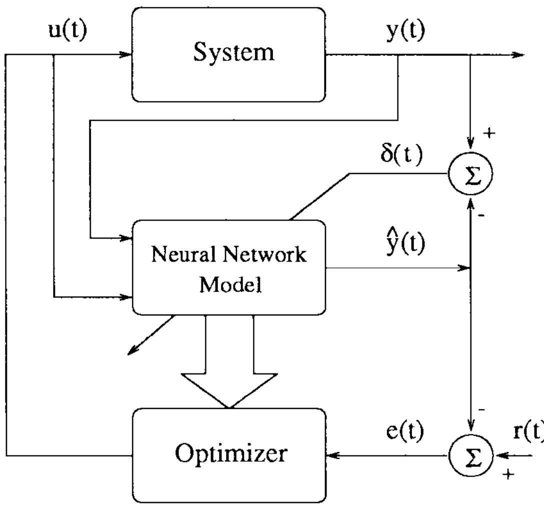


Fig. 1. Neural-network controller.

it is necessary to develop a new approach which simplifies the formulation of the control signal.

II. Algorithm Description

Assume that the unknown nonlinear system to be considered is expressed by

$$y(t+1) = f(y(t), y(t-1), \cdots, y(t-n), u(t) \\ u(t-1), \cdots, u(t-m)) \quad (1)$$

where $y(t)$ is the scalar output of the system, $u(t)$ is the scalar input to the system, $f(\cdots)$ is the unknown nonlinear function to be estimated by a neural network, and n and m are the *known* structure orders of the system. The purpose of our control algorithm is to select a control signal $u(t)$, such that the output of the system $y(t)$ is made as close as possible to a prespecified setpoint $r(t)$.

Fig. 1 shows the overall structure of the closed-loop control system which consists of

1) the system (1);
2) a feedforward neural network which estimates $f(\cdots)$;
3) a controller realized by an optimizer.

A two-layer neural network is used to learn the system and the standard backpropagation algorithm [7] and [8] is employed to train the weights. The activation functions are hyperbolic tangent for the first layer and linear for the second layer. Since the input to the neural network is

$$p = [y(t), y(t-1), \cdots, y(t-n), u(t) \\ u(t-1), \cdots, u(t-m)] \quad (2)$$

the neuro model for the unknown system (1) can be expressed as

$$\hat{y}(t+1) = \hat{f}(y(t), y(t-1), \cdots, y(t-n), u(t) \\ u(t-1), \cdots, u(t-m)) \quad (3)$$

where $\hat{y}(t+1)$ is the output of the neural network and $\hat{f}$ is the estimate of f. Since the backpropagation training algorithm guarantees that

$$[y(t+1) - \hat{y}(t+1)]^2 = \min. \quad (4)$$

$\hat{y}(t+1)$ is also referred to as a predicted output of the system (1). As a result, the control signal can be selected such that $\hat{y}(t+1)$ is made as close as possible to $r(t)$. For this purpose, we define an objective function J as follows:

$$J = \tfrac{1}{2}e^2(t+1) \quad (5)$$

where

$$e(t+1) = r(t+1) - \hat{y}(t+1). \quad (6)$$

The control signal $u(t)$ should therefore be selected to minimize J. Using the neural-network structure, (3) can be rewritten to give

$$\hat{y}(t+1) = w_2[\tanh(w_1 p + b_1)] + b_2 \quad (7)$$

where w_1, w_2, b_1, and b_2 are the weights and biases matrices of the neural network [7]. To minimize J, the $u(t)$ is recursively calculated via using a simple gradient descent rule

$$u(t+1) = u(t) - \eta \frac{\partial J}{\partial u(t)} \quad (8)$$

where $\eta > 0$ is a learning rate. It can be seen that the controller relies on the approximation made by the neural network. Therefore it is necessary that $\hat{y}(t+1)$ approaches the real system output $y(t+1)$ asymptotically. This can be achieved by keeping the neural-network training online. Differentiating (5) with respect to $u(t)$, it can be obtained that

$$\frac{\partial J}{\partial u(t)} = -e(t+1)\frac{\partial \hat{y}(t+1)}{\partial u(t)} \quad (9)$$

where $\partial \hat{y}(t+1)/\partial u(t)$ is known as the gradient of the neural-network model (or sensitivity derivatives) with respect to $u(t)$. Substituting (9) into (8), we have

$$u(t+1) = u(t) + \eta e(t+1)\frac{\partial \hat{y}(t+1)}{\partial u(t)}. \quad (10)$$

The gradient can then be analytically evaluated by using the known neural-network structure (7) as follows:

$$\frac{\partial \hat{y}(t+1)}{\partial u(t)} = w_2[\operatorname{sech}^2(w_1 p + b_1)]w_1 \frac{dp}{du} \quad (11)$$

where

$$\frac{dp}{du} = [0, 0, \cdots, 0, 1, 0, \cdots, 0]' \quad (12)$$

is the derivative of the input vector p respect to $u(t)$. Finally, (10) becomes

$$u(t+1) = u(t) + \eta e(t+1) w_2[\operatorname{sech}^2(w_1 p + b_1)]w_1 \frac{dp}{du}. \quad (13)$$

Equation (13) can now be used in a computer program for real-time control. To summarize, we have

1) produce $\hat{y}(t+1)$ using (7);
2) find $e(t+1)$ using (6);
3) update the weights using backpropagation algorithm [7] and [8];
4) compute new control signal from (13);
5) feed $u(t+1)$ to the system;
6) go to Step 1).

III. An Extended Algorithm

The algorithm described in Section I can be improved by using the technique in generalized predictive control theory [9], which considers not only the design of the instant value of the control signal but also its future values. As a result, future values of setpoint and the system output are needed to formulate the control signal. Since the neural-network model (3) represents the plant to be controlled asymptotically, it can be used to predict future values of the system output. For this purpose, let T be a prespecified positive integer and denote

$$R_{t,T} = [r(t+1), r(t+2), \cdots, r(t+T)]' \quad (14)$$

as the future values of the setpoint and

$$\hat{Y}_{t,T} = [\hat{y}(t+1), \hat{y}(t+2), \cdots, \hat{y}(t+T)]' \quad (15)$$

as the predicted output of the system using the neural-network model (7), then the following error vector:

$$E_{t,T} = [e(t+1), e(t+2), \cdots, e(t+T)]' \quad (16)$$

can be obtained where

$$e(t+i) = r(t+i) - \hat{y}(t+i). \quad (17)$$

Defining the control signals to be determined as

$$U_{t,T} = [u(t+1), u(t+2), \cdots, u(t+T)]' \quad (18)$$

and assuming the following objective function:

$$J_1 = \tfrac{1}{2}[E_{t,T}^T E_{t,T}] \quad (19)$$

then our purpose is to find $U_{t,T}$ such that J_1 is minimized. Using the gradient decent rule, it can be obtained that

$$U_{t,T}^{k+1} = U_{t,T}^k - \eta \frac{\partial J_1}{\partial U_{t,T}^k} \quad (20)$$

where

$$\frac{\partial J_1}{\partial U_{t,T}^k} = E_{t,T}\frac{\partial \hat{Y}_{t,T}}{\partial U_{t,T}^k} \quad (21)$$

and

$$\frac{\partial \hat{Y}_{t,T}}{\partial U_{t,T}^k} = \begin{bmatrix} \frac{\partial \hat{y}(t+1)}{\partial u(t)} & 0 & \cdots & 0 \\ \frac{\partial \hat{y}(t+2)}{\partial u(t)} & \frac{\partial \hat{y}(t+2)}{\partial u(t+1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}(t+T)}{\partial u(t)} & \frac{\partial \hat{y}(t+T)}{\partial u(t+1)} & \cdots & \frac{\partial \hat{y}(t+T)}{\partial u(t+T-1)} \end{bmatrix}'. \quad (22)$$

It can be seen that each element in the above matrix can be found by differentiating (3) with respect to each element in (18). As a result, it can be obtained that

$$\frac{\partial \hat{y}(t+n)}{\partial u(t+m-1)} = \frac{\partial \hat{f}(p)}{\partial u(t+m-1)} + \sum_{i=m}^{n-1} \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+i)} \left[\frac{\partial \hat{y}(t+i)}{\partial u(t+m-1)} \right] \quad \text{for } n = 1, 2, \cdots, T \quad m = 1, 2, \cdots, T. \tag{23}$$

Equation (22) is the well-known Jacobian matrix which must be calculated using (23) every time a new control signal has to be determined. This could result in a large computational load for a big T. Therefore a recursive method for calculating the Jacobian matrix is developed in the following so that the algorithm can be applied to the real-time systems with fast responses. This is illustrated by computing the elements in the Jacobian matrix for $T = 3$

$m = 1, n = 1$:

$$\frac{\partial \hat{y}(t+1)}{\partial u(t)} = \frac{\partial \hat{f}(p)}{\partial u(t)} \tag{24}$$

$m = 2, n = 1$:

$$\frac{\partial \hat{y}(t+1)}{\partial u(t+1)} = 0 \tag{25}$$

$m = 3, n = 1$:

$$\frac{\partial \hat{y}(t+1)}{\partial u(t+2)} = 0 \tag{26}$$

$m = 1, n = 2$:

$$\frac{\partial \hat{y}(t+2)}{\partial u(t)} = \frac{\partial \hat{f}(p)}{\partial u(t)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+1)} \left[\frac{\partial \hat{y}(t+1)}{\partial u(t)} \right] \tag{27}$$

$m = 2, n = 2$:

$$\frac{\partial \hat{y}(t+2)}{\partial u(t+1)} = \frac{\partial \hat{f}(p)}{\partial u(t+1)} \tag{28}$$

$m = 3, n = 2$:

$$\frac{\partial \hat{y}(t+2)}{\partial u(t+2)} = 0 \tag{29}$$

$m = 1, n = 3$:

$$\frac{\partial \hat{y}(t+3)}{\partial u(t)} = \frac{\partial \hat{f}(p)}{\partial u(t)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+1)} \left[\frac{\partial \hat{y}(t+1)}{\partial u(t)} \right] + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+2)} \left[\frac{\partial \hat{y}(t+2)}{\partial u(t)} \right] \tag{30}$$

$m = 2, n = 3$:

$$\frac{\partial \hat{y}(t+3)}{\partial u(t+1)} = \frac{\partial \hat{f}(p)}{\partial u(t+1)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+2)} \left[\frac{\partial \hat{y}(t+2)}{\partial u(t+1)} \right] \tag{31}$$

$m = 3, n = 3$:

$$\frac{\partial \hat{y}(t+3)}{\partial u(t+2)} = \frac{\partial \hat{f}(p)}{\partial u(t+2)}. \tag{32}$$

From (24)–(32) it can be seen that (24) is a part of (27) which is another term in (30). Also, (28) can be used in (31). Rearranging (24), (27), (28), (30), and (31), it can be obtained that

$$\frac{\partial \hat{y}(t+2)}{\partial u(t)} = \frac{\partial \hat{y}(t+1)}{\partial u(t)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+1)} \left[\frac{\partial \hat{y}(t+1)}{\partial u(t)} \right] \tag{33}$$

$$\frac{\partial \hat{y}(t+3)}{\partial u(t)} = \frac{\partial \hat{y}(t+2)}{\partial u(t)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+2)} \left[\frac{\partial \hat{y}(t+2)}{\partial u(t)} \right] \tag{34}$$

$$\frac{\partial \hat{y}(t+3)}{\partial u(t+1)} = \frac{\partial \hat{y}(t+2)}{\partial u(t+1)} + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+2)} \left[\frac{\partial \hat{y}(t+2)}{\partial u(t+1)} \right]. \tag{35}$$

As a result, the recursive form of (23) for the computation of the elements of the system Jacobian matrix is given by

$$\frac{\partial \hat{y}(t+n)}{\partial u(t+m-1)} = \frac{\partial \hat{y}(t+n-1)}{\partial u(t+m-1)} \left[1 + \frac{\partial \hat{f}(p)}{\partial \hat{y}(t+n-1)} \right]. \tag{36}$$

From (36) it can be seen that in order to find all the elements in the Jacobian matrix it is only necessary to calculate the diagonal elements and the partial derivatives of the $\hat{f}(\cdots)$ with respect to the previous predicted output. Therefore less calculation is required to formulate the Jacobian matrix. The two derivative terms in (36) are given by

$$\frac{\partial \hat{y}(t+n-1)}{\partial u(t+m-1)} = w_2[\text{sech}^2(w_1 p + b_1)] w_1 \frac{dp}{du} \tag{37}$$

$$\frac{\partial \hat{f}(p)}{\partial \hat{y}(t+n-1)} = w_2[\text{sech}^2(w_1 p + b_1)] w_1 \frac{dp}{d\hat{y}} \tag{38}$$

where

$$\frac{dp}{d\hat{y}} = [1, 0, \cdots, 0, 0, 0, \cdots, 0]'. \tag{39}$$

Equations (37) and (38) can now be used in a computer program to calculate the Jacobian matrix and the whole algorithm is summarized as follows.

1) Select a T.
2) Find new predicted output $\hat{y}(t+1)$ using (7).
3) Calculate $\partial \hat{y}(t+n-1)/\partial u(t+m-1)$ and $\partial \hat{f}(p)/\partial \hat{y}(t+n-1)$ via (37) and (38).
4) Update vector p using new $\hat{y}(t+1)$ calculated in Step 2) and $u(t+n-1)$ from the vector of future control signals (18).
5) Use (36) and the results obtained in Step 3) to calculate the off-diagonal elements of the Jacobian.
6) Use (20) form a new vector of future control signals.
7) Apply $u(t+1)$ found in Step 6) to close the control loop.
8) Return to Step 1).

IV. Application to a Real System

A. The System Description

The algorithm described here has been tested on a Bytronic Process Control Unit [10] which has a fluid flow process. Both

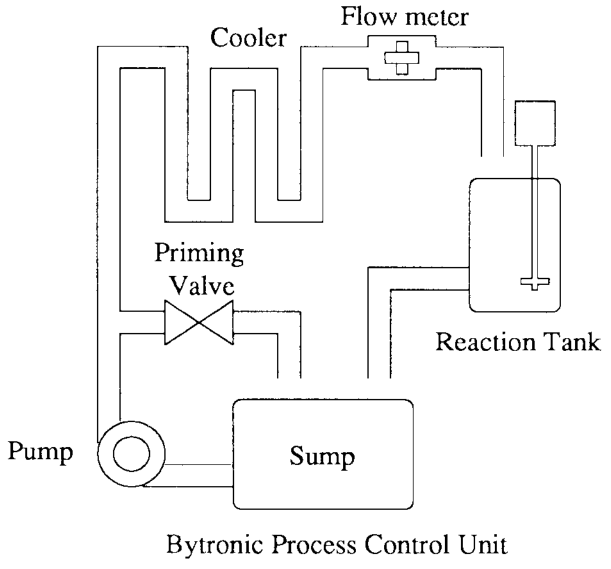


Fig. 2. The bytronic process control unit.

flow rate and the temperature of the fluid can be controlled and are considered in this work. Fig. 2, shows a simplified diagram of the flow-rate control system.

It can be seen from this figure that the system consists of a sump, a pump, flowmeter, a process tank, and a cooler. The sump works just as a water reservoir for the unit. The pump consists of a 12-V dc motor and has a maximum pumping capacity of 2.0 l/min. The flowmeter is of the impeller type. Its full scale output is approximately 570 Hz or 2.0 l/min. The process tank consists of a cylindric metallic container and an electric heater. The cooler is basically a radiator and uses an electric fan to cool down the water. The unit is connected to a PC 386 computer through a computer control module (not shown) which contains an 8-b analog to digital converter for the flowmeter and tank temperature readings and an 8-b digital to analog converter for the pump control. The electric heater is controlled from the computer control module via a pulse modulated control circuit whose duty cycle is determined by the control algorithm.

B. Neural-Network Implementation

Fig. 3 shows the neural-network architecture for the control system. A two-layer neural network with ten hidden neurons and one output neuron is used. The backpropagation algorithm is employed to train the weights.

The weights are therefore updated as follows:

$$w_{ij}(k+1) = w_{ij}(k) + \alpha \frac{\partial E}{\partial w_{ij}(k)} \tag{40}$$

where α is a prespecified learning rate. The performance function E for training the neural network is defined as

$$E = \tfrac{1}{2} \sum_{l=1}^{q} \sum_{r=1}^{s} (y_r^l - \hat{y}_r^l)^2 \tag{41}$$

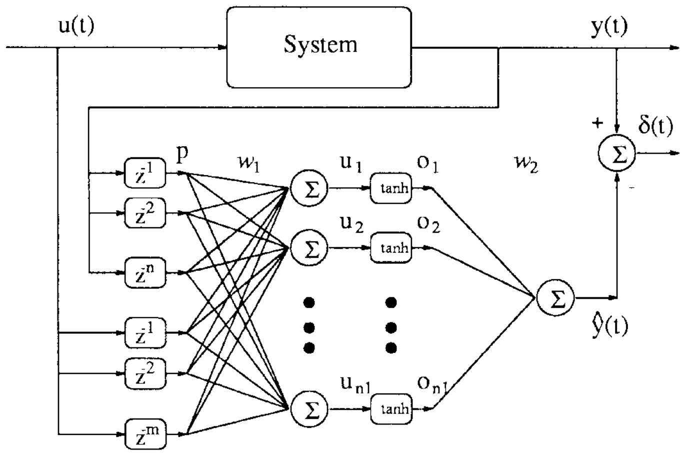


Fig. 3. The neural-network structure.

where q is the number of training samples and s is the number of the neurons in the output layer. The final weights and biases for the neural-network model of the flow process of bytronic system are shown in the Appendix.

C. Experimental Results

In order to demonstrate the effectiveness of basic algorithm two experimental cases are described. Case 1 consist of a flow-rate control in which three different tests are considered and they are listed as follow.

1) The system is subjected to some step setpoint changes.
2) The controller is realized with different learning rates.
3) The system is under an external disturbance.

For these three cases the learning rate of the neural network is fixed at $\alpha = 0.001$. Case 2 consist of a temperature control for the process tank and the following two situations are considered.

1) The system is subjected to some step setpoint changes.
2) The system is under an external disturbance.

In both cases the tank is filled up with tap water, which is then warmed up by the electric heater to a desired temperature. The neural-network learning rate is $\alpha = 0.01$.

1) Flow Control: In the first case, the learning rate of the controller was set to $\eta = 0.015$ and three different set points, $r_1 = 1.4$, $r_2 = 1.2$, and $r_3 = 1.0$ l/min, were applied to the closed-loop system. The sample time was approximately 0.1 s and the controlled system responses are shown in Fig. 4.

In the second test the set point is kept constant at $r = 1.2$ l/min. The learning rate for the controller was changed between three different values, $\eta = 0.015$, $\eta = 0.020$, and $\eta = 0.05$. The rest of the parameters are the same as before and the responses are described in Fig. 5.

It can be seen that any increase of the controllers' learning rate directly leads to an increase in response speed, which could, on the other hand, deteriorate the stability of the closed-loop system.

In the third test, an external disturbance was applied by suddenly opening the priming valve (see Fig. 2) from one position to another position. The learning rate of the controller

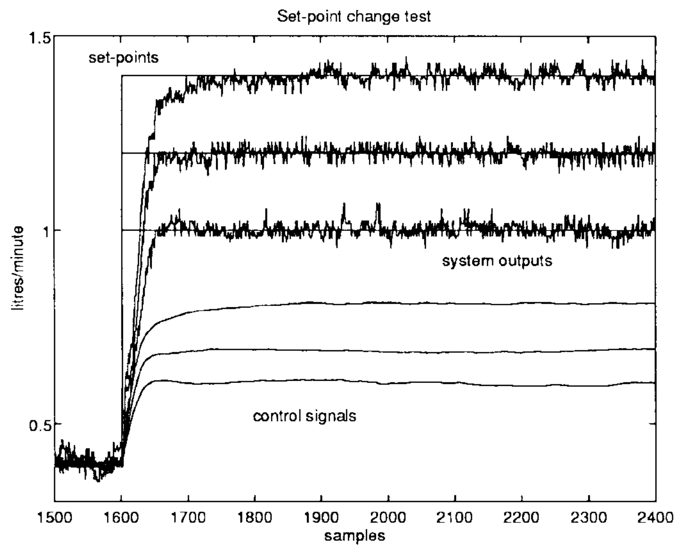


Fig. 4. Test 1. Change of set point.

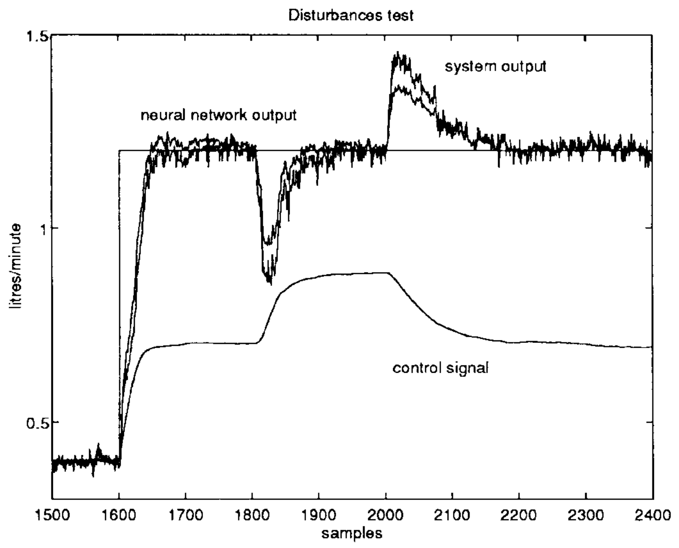


Fig. 6. Test 3. Response to disturbances.

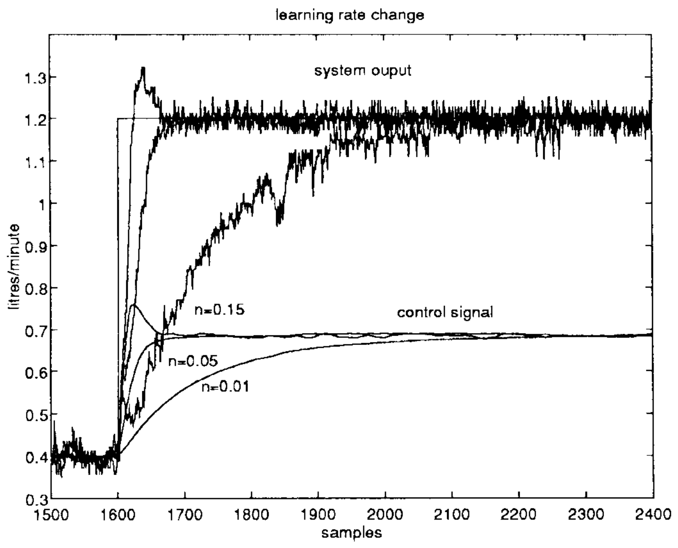


Fig. 5. Test 2. Learning rate changes.

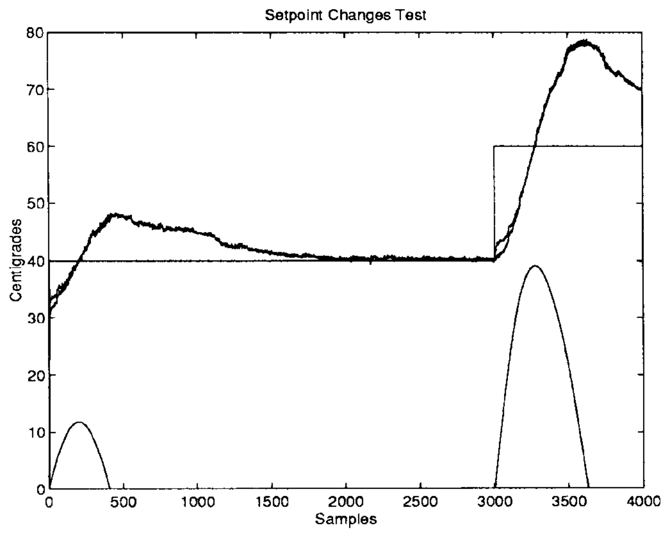


Fig. 7. Test 1: Setpoint changes.

is set to $\eta = 0.015$ and the set point is $r = 1.2$ l/min. The results are shown in Fig. 6. It can be seen that desired disturbance rejection is achieved.

2) Temperature Control: In the first test the learning rate of the controller is set at $\eta = 0.05$ and the learning rate for the neural network is $\alpha = 0.01$ with sampling period being set to 3 s. Two set point changes have been applied to the controlled system. The first and second set points are 40 and 60°C, respectively. Fig. 7 shows the testing results.

Since the system is nonsymmetric, the control action cannot take negative values. As a result, open-loop behavior will be obtained after an overshoot is produced.

For the second test, the tank temperature is set to 40°C and a disturbance is created by pouring 500 ml of cold tap water into the tank. This causes the temperature to go below the set point and a correct control action can therefore be enforced. Fig. 8 shows the results of the disturbance test.

Since the system is nonsymmetric, it is obvious that a disturbance which increases the tank temperature cannot be compensated, although it can be observed.

V. Stability Issues

The control algorithm described in this paper relies on the convergent training of the neural-network model and the updating of the control signals. This leads to the analysis of stability for the closed-loop system. Over the past few years, much effort has been made in this respect and typical analysis tools for deterministic known nonlinear systems have been developed based upon the well-known Lyapunov methods [11], [12]. More recently, several adaptive control strategies for nonlinear systems have been developed [13]–[16] where closed-loop stability has been analyzed and guaranteed. However, most of these approaches only considered a class of unknown nonlinear systems (i.e., affine models [11]). As

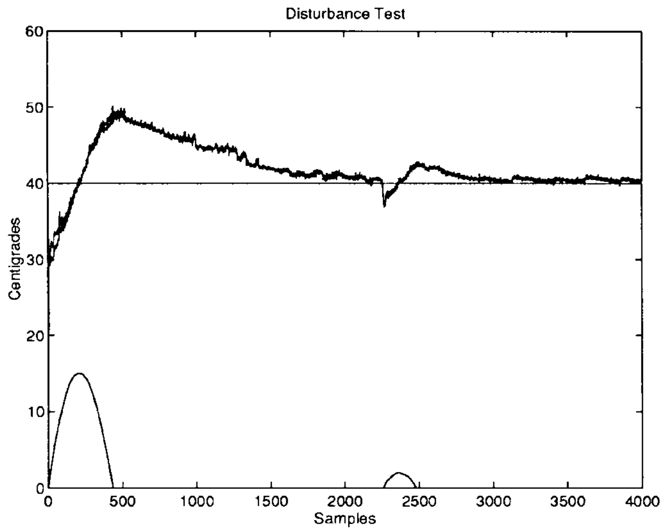


Fig. 8. Test 2: Disturbance test.

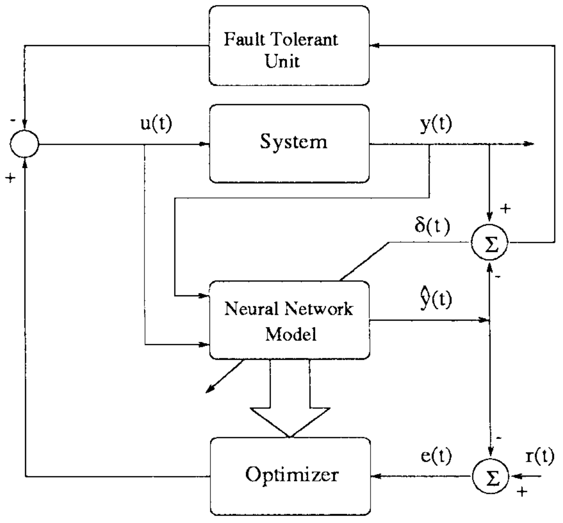


Fig. 9. A modified neural-network control system.

such, difficulty remains in the development of a systematic approach for the stability analysis for the general unknown nonlinear system given by (1). In this section, we will briefly discuss a possibility for the stability analysis of the closed-loop control system shown in Fig. 1.

It can be seen, from the control structure in Fig. 1, that the closed-loop system will have a local stability property if

1) the neural-network model converges to the actual system;
2) the control updating rule (8) or (20) stabilizes the neural-network model.

The first condition can be guaranteed via the well-known universal approximation property of neural networks [7]. This means that there is a Lyapunov function $V_1(\theta, \delta)$ such that

$$\Delta V_1(\theta, \delta) = V_1[\theta(t), \delta(t)] - V_1[\theta(t-1), \delta(t-1)] < 0 \quad (42)$$

where $\theta(t) = [\hat{w}_1(t), \hat{w}_2(t), \hat{b}_1(t), \hat{b}_2(t)]$. While the second condition means that the control algorithm (8) or (20) should stabilize the nonlinear system represented by the neural-network model (3). Using the standard first-order approximation around the trajectory

$$\phi = [y(t-1), \cdots, y(t-n), u(t-1), \cdots, u(t-m)] \quad (43)$$

the neural-network model (3) can be eventually expressed as

$$\Delta\hat{y}(t) = f_1(\phi) + g_1(\phi)\Delta u(t) + \text{h.o.t.} \quad (44)$$

where "h.o.t." represents all the possible high-order terms. This enables us to use the method developed in [11]–[17] to analyze the local stability property when the control algorithm (8) or (20) is applied to control (3). This will lead to another Lyapunov function $V_2[u(t), \hat{y}, \phi]$ satisfying

$$\Delta V_2[u(t), \hat{y}, \phi] < 0. \quad (45)$$

As a result, we can construct the following Lyapunov function:

$$V[\theta, \delta, u(t), \hat{y}, \phi] = V_1(\theta, \delta) + V_2[u(t), \hat{y}, \phi] \quad (46)$$

for the closed-loop system. From (42)–(45), it can be seen that

$$\Delta V[\theta, \delta, u(t), \hat{y}, \phi] < 0 \quad (47)$$

holds.

Since the convergence of the neural-network training can only reach a guaranteed local minima. The above route for stability analysis will be very likely to only produce conditions for a local stability property for the closed-loop system. Further investigation is therefore necessary to establish results on the global asymptotically stability for the closed-loop system. This may lead to the modification of the proposed control algorithm (8) or (20). For example, an alternative approach would be to use the so called "fault-tolerant control structure" shown in Fig. 9. In this case, large parameter (weights) changes are regarded as faults [18] to the system and an additional loop is used to compensate the closed-loop system. The fault-tolerant unit provide an extra input to the system when changes (i.e., such as those caused by large variations of the system's input and output) occur. Indeed, this is the recent development made by our group where a unified scheme for both linear and nonlinear systems has been obtained.

VI. Conclusions

A neural-network-based direct adaptive control strategy is presented in this paper for general unknown nonlinear systems, where a simplified formulation of the control signals is obtained using the combination of a feedforward neural network and an optimization scheme. The neural network is used online to estimate the system and the backpropagation training algorithm is applied to train the weights. Taking the resulting neural-network estimation as a known nonlinear dynamic model for the system, control signals can be directly obtained using the well-established gradient descent rule. An improved algorithm is also included where both instant and future values of control signals are derived. Real applications to the flow rate and temperature control systems have successfully demonstrated the use of the proposed method. Moreover, the stability issues of the proposed control scheme is discussed

$$w_1 = \begin{bmatrix} 0.10755 & 0.10530 & 0.29778 & 0.02061 & 0.24607 & 0.23131 \\ 0.31623 & 0.01526 & 0.16081 & 0.20608 & 0.55607 & 0.01294 \\ 0.31623 & -0.00064 & 0.07041 & 0.33608 & 0.71270 & 0.02362 \\ 0.22066 & 0.31154 & 0.68054 & 0.10207 & 0.92378 & 0.13385 \\ 0.13024 & 0.02151 & 0.40482 & 0.41307 & 0.13378 & 0.41959 \\ 0.07022 & 0.04765 & 0.30681 & 0.22307 & 0.05778 & 0.22624 \\ 0.50621 & 0.25754 & 0.51681 & 0.10689 & 0.36770 & 0.00619 \\ 0.11210 & 0.60754 & 0.14365 & 0.11713 & 0.15770 & 0.21137 \\ 0.52210 & 0.09513 & 0.45578 & 0.42713 & 0.51409 & 0.02200 \\ 0.38450 & 0.28778 & -0.03439 & 0.39596 & 0.32131 & -0.01801 \end{bmatrix} \tag{48}$$

and a possible route for the analysis of the stability of the closed-loop system is given. However, as difficulty remains in the stability analysis for general nonlinear control systems, further investigation is needed in this respect.

Appendix

The weight and bias matrices for the neural-network model of the flow control process are shown in (48)–(51). {Equation (48) is shown at the top of the page.}

$$w_2 = \begin{bmatrix} 0.10896 \\ 0.25686 \\ -0.14044 \\ -0.17614 \\ 0.49180 \\ 0.24099 \\ -0.03533 \\ -0.16433 \\ 0.20741 \\ 0.20316 \end{bmatrix}^T \tag{49}$$

$$b_1 = \begin{bmatrix} 0.37608 \\ 0.20069 \\ 0.38329 \\ 0.29463 \\ -0.14429 \\ 0.00860 \\ 0.51543 \\ 0.54052 \\ 0.01964 \\ 0.11411 \end{bmatrix} \tag{50}$$

$$b_2 = [0.0417625]. \tag{51}$$

Acknowledgment

The authors gratefully acknowledge computer support from the Control Systems Centre (University of Manchester Institute of Science and Technology, U.K.).

References

[1] M. Saerens and A. Soquet, "A neural controller," in *Proc. 1st Int. Conf. Artificial Neural Networks*, London, Oct. 1989, pp. 211–215.

[2] M. I. Jordan and R. A. Jacobs, "Learning to control an unstable system with forward modeling," *Advances in Neural Inform. Processing Syst.*, vol. 2, pp. 324–331, 1990.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[4] W. H. Schiffmann and H. W. Geffers, "Adaptive control of dynamic systems by backpropagation networks," *Neural Networks*, vol. 6, pp. 517–524, 1993.

[5] Y. Tan and R. D. Keyser, "Neural-network-based adaptive predictive control," in *Advances in MBPC*, 1993, pp. 77–88.

[6] D. Hammestrom, "Neural networks at work," *IEEE Spectrum*, vol. 30, pp. 26–32, 1993.

[7] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, pp. 1415–1441, 1990.

[8] F. M. Silva and L. B. Almeida, "Speeding up backpropagation," *Advanced Neural Computers*, pp. 151–158, 1990.

[9] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control: Parts I and II, the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–160, 1987.

[10] *Documentation for the Bytronic Process Control Unit (IBM version)*, Bytronics plc.

[11] A. Isidori, *Nonlinear Control Systems*, 2nd ed. New York: Springer-Verlag, 1989.

[12] A Isidori and C. I. Byrnes, "Output regulations of nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 131–140, 1990.

[13] F. C. Chen and H. K. Khalil, "Adaptive control of nonlinear systems using neural networks," *Int. J. Contr.*, vol. 55, pp. 1299–1317, 1992.

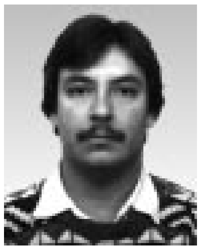
[14] F. C. Chen and C. C. Liu, "Adaptively controlling nonlinear continuous-time systems using multilayer neural networks," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 1306–1310, 1994.

[15] S. Jagannathan and F. L. Lewis, "Multilayer discrete-time neural net controller with guaranteed performance," *IEEE Trans. Automat. Contr.*, vol. 7, pp. 107–130, 1996.

[16] M. M. Polycarpou and P. A. Ioannou, "Modeling, identification, and stable adaptive control of continuous time nonlinear dynamical systems using neural networks," in *Proc. Amer. Contr. Conf.*, 1992, pp. 36–40.

[17] H. Wang, G. P. Liu, C. J. Harris, and M. Brown, *Advanced Adaptive Control*. Oxford: Pergamon, 1995.

[18] H. Wang and S. Daley, "Actuator fault diagnosis: An adaptive observer-based technique," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 1073–1078, 1996.



Jose R. Noriega was born in Hermosillo, Mexico, in 1965. He received the B.Sc. degree in industrial engineering and electronics from the Instituto Tecnologico de Hermosillo in 1989. In 1993, he received the Diploma degree in technical sciences from the University of Manchester Institute of Science and Technology (UMIST) in Manchester, U.K. He is currently completing the Ph.D. degree on advanced control techniques in UMIST under the supervision of Dr. H. Wang.

From 1989 to 1992, he worked in the University of Sonora as an Academic Technician involved in the design and development of electronic instruments. His current research interests includes the development of advanced adaptive control, fault detection and diagnosis for dynamic systems, and fault-tolerant control using neural networks, machine learning, system identification, and modeling.

Hong Wang (M'95) was born in Beijing, China, in 1960. He received the B.Sc., M.Eng., and Ph.D. degrees in electrical and control engineering from Huainan University of Mining Science and Huazhong University of Science and Technology in 1982, 1984, and 1987, respectively.

From 1988 to 1992, he was a Postdoctoral Research Fellow with Salford, Brunel, and Southampton Universities, U.K. In 1992, he joined the University of Manchester Institute of Science and Technology (UMIST), U.K., as a Lecturer in process control. At present, he is a Senior Lecturer in the Paper Science Department leading an active control group. Since 1986, he has published 58 technical papers in various international journals and conferences. He is the leading author of two books: on *Advanced Adaptive Control* (New York: Pergamon, 1995) and *Advanced Process Control for Paper and Board Making* (London: PIRA, 1997). His main research interests include advanced adaptive control, fault detection and diagnosis, and fault-tolerant control for general dynamic systems.