*Engng Applic. Artif. Intell.* Vol. 5, No. 6, pp. 527–538, 1992
Printed in Great Britain.

0952-1976/92 $5.00 + 0.00
Pergamon Press Ltd

*Contributed Paper*

# Application of Neural Networks to Temperature Control in Thermal Power Plants*

XIANZHONG CUI
The University of Michigan, Ann Arbor

KANG G. SHIN
The University of Michigan, Ann Arbor

*In a thermal power plant with once-through boilers, it is important to control the temperature at the middle point where water becomes steam. However, there are many problems in the design of such a control system, due to a long system response delay, dead-zone and saturation of the actuator mechanisms, uncertainties in the system model and/or parameters, and process noise. To overcome these problems, an adaptive controller has been designed using neural networks, and tested extensively via simulations.*

*One of the key problems in designing such a controller is to develop an efficient training algorithm. Neural networks are usually trained using the output errors of the network, instead of using the output errors of the controlled plant. However, when a neural network is used to control a plant directly, the output errors of the network are unknown, since the desired control actions are unknown. This paper proposes a simple training algorithm for a class of nonlinear systems, which enables the neural network to be trained with the output errors of the controlled plant. The only a priori knowledge of the controlled plant is the direction of its output response. Due to its simple structure and algorithm, and good performance, the proposed controller has high potential for handling difficult problems in process-control systems.*

Keywords: Neural networks, process control systems, system response delay, dead zone.

## 1. INTRODUCTION

Supercritical once-through boilers are widely used in large-scale thermal power plants. In such a boiler, since water fed into the boiler (feedwater) is over the critical point for water (pressure: 22.129 MPa, and temperature: 374.15 °C), when water is sufficiently heated, it becomes steam continuously without going through the evaporation process. In order to avoid any premature damage of the steam turbine and improve efficiency, it is crucial to precisely control the main steam temperature, reheat steam temperature, and main steam pressure. However, since the dynamics of once-through boilers are much more complicated than those of drum boilers, it is more difficult to design control systems for once-through boilers. Figure 1 shows a conceptual diagram of the steam–water loop in a thermal power plant with once-through boilers. The main steam temperature depends on feedwater flow rate, steam flow rate and fuel flow rate, and is regulated by spray water flow rate. The reheat steam temperature is controlled by gas recirculation flow rate. Water is heated through economizer and water walls. The point at which water becomes steam is called the *middle point*, which changes with different operating conditions. The steam is continuously heated through primary and secondary superheaters and then sent to a steam turbine. If the temperature at the middle point is controlled well, it will be easier to precisely control the temperature at the outlet of the boiler. That is, the control of the middle-point temperature works as a rough adjustment of main steam temperature. This rough adjustment is important to ensure the operational safety of the boiler and the control quality of the main steam temperature, since the main steam temperature regulated by spray water

Correspondence should be sent to: Xianzhong Cui, Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122, U.S.A.

* The work reported in this paper was supported in part by the National Science Foundation under Grant No. DMC-8721492. Any opinions, findings, and recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the NSF. A subset of this paper was presented at the 1991 American Control Conference, June 1991, Boston, MA.

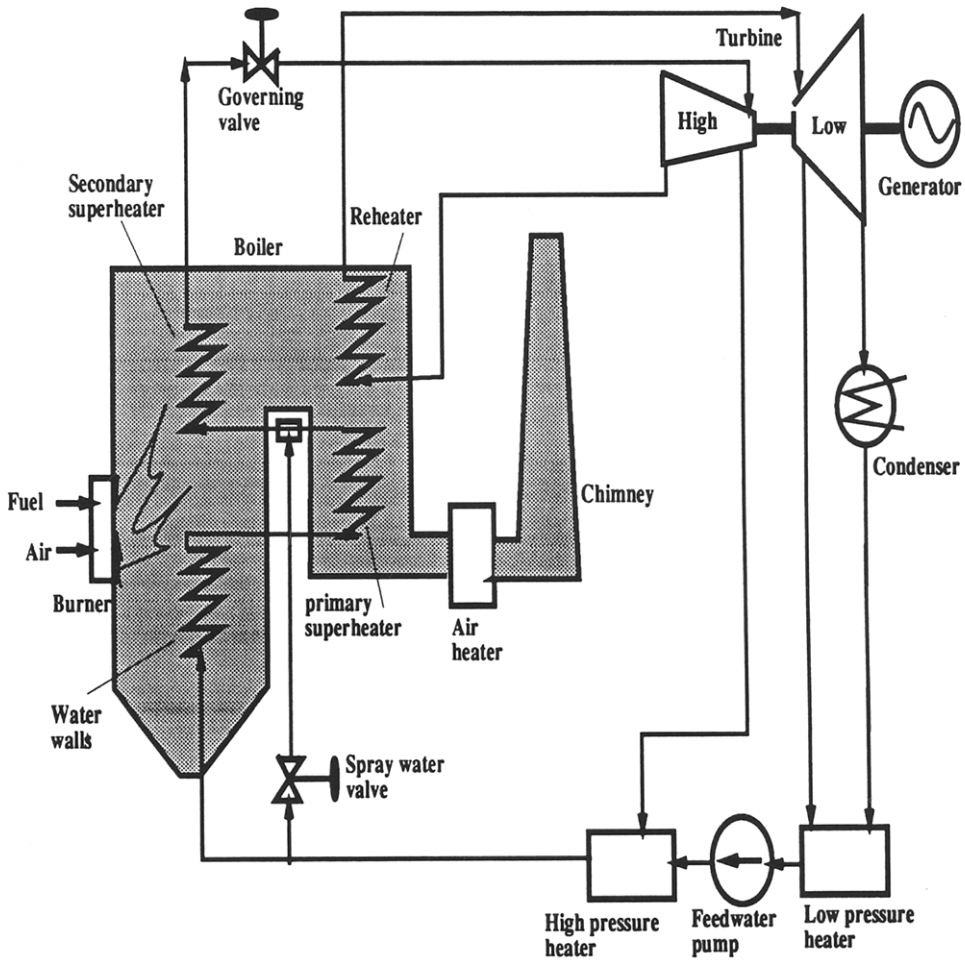


Fig. 1. Conceptual diagram of the steam-water loop of thermal power plants.

has a relatively small regulation range. However, the middle-point temperature is difficult to control using conventional PID-type controllers, mainly for the following reasons:

1. The middle-point temperature is regulated by feedwater flow rate. Unlike the quick response of main steam temperature to spray water, the response of middle-point temperature to feedwater flow has a very long time delay.
2. The middle-point temperature is affected severely by the random disturbances resulting from the variation of feedwater temperature and pressure, heating value of coal, oxygen content of gas, position of fire center, position of the middle point, etc.
3. System dynamics may change due to fouling on the surface of water walls and the gas flow system.
4. The actuator mechanism may suffer the problem of dead-zone and saturation.

The main issues to consider for the design of a middle-point temperature control system are the negative effects of long system response delay, dead zone and saturation of actuator mechanisms, and process and measurement noises. If these effects are not considered, the dynamic property of a controlled plant may not be very complex, even though its detailed structure and/or parameters are usually unknown. However, when such a plant is put in operation, it will be difficult for the control system to achieve high performance due mainly to the negative effects mentioned above. Contemporary industrial process control systems dominantly rely on PID-type controllers, though the hardware to implement control algorithms has been improved significantly in recent years. In addition to the difficulty in achieving high control quality, the fine tuning of the controller's parameters is a tedious task, requiring experts with knowledge both in control theory and process dynamics. The reliability of such a system is also very important for operational security and efficiency. All of these call for the development of new controllers. The goal of this paper is to develop such a new controller using neural networks (NNs). Particularly, the focus is on dealing with the nonlinearity of dead zone and saturation, and the negative effects of long response delays and process noises.

The potential of NNs for control application lies in the following properties: (1) they could be used to approximate any continuous mapping, (2) they achieve this approximation through learning, (3) parallel pro-

cessing and fault tolerance can be easily accomplished with NNs. One of the most popular NN architectures is a multilayer perceptron with the back propagation (BP) algorithm. It is proved that a four-layer (with two hidden layers) perceptron can be used to approximate any continuous function with the desired accuracy.[1] BP has been used successfully for pattern classification, though its original development placed more stress on control applications.[2] A controller is usually connected serially to the controlled plant under consideration. For a multilayer perceptron, the weights of the network need to be updated using the network's output error. For an NN-controller, the NN's output is the control command to the system. However, when the NN is serially connected to a controlled plant, the network's output error is unknown, since the desired control action is unknown. This implies that BP cannot be applied to control problems directly. Thus, one of the key problems in designing an NN-controller is to develop an efficient training algorithm.

Several related schemes have been proposed to design a controller using neural networks. One of them is training an NN to learn the system's inverse, and then the desired system output is achieved using the control input produced by the system's inverse. Certainly, this requires the system to be invertible. References 3, 4 and 5 are such examples. In Ref. 3 the controlled plant was treated as an additional, unmodifiable layer, and the output error of the network was computed from the output error of the system. In Ref. 4, the system's output error was propagated back through the plant using its partial derivatives at an operating point. In Ref. 5, a set of actual system outputs are selected as training data and fed into the NN during its training period. By comparing the output of the NN with the desired system output, the network's output error is computed, which is then used to train the NN. After the NN becomes well-trained, the input of the NN is switched to the desired system output. Then, the NN acts as the inverse of the plant, and its output will drive the system to reach the desired value. However, in practice, even if the system is invertible, the inverse control scheme may not be acceptable. For example, if the system is a non-minimum phase system, then the resulting design is not internally stable. The invertibility of nonlinear systems was discussed in Ref. 6, and a sufficient-input criterion for designing an NN to learn a system's inverse was established. Other examples of NN-controllers are Refs 7 and 8, which used reference models to train the NN. Kraft and Miller designed controllers using a similar structure to a CMAC (cerebellar model articulation controller).[9,10] Narendra and Parthasarathy[11] proposed a scheme of indirect adaptive control using a multilayer perceptron with the BP algorithm. The NN was trained first to attain the same dynamic behavior as the controlled plant. Then a controller was designed by using the NN's output to cancel the nonlinear part of the controlled plant, and by including the same terms of a reference model. Five dominant system architectures with NNs for control applications were summarized in Refs 2 and 12, and the importance and applications of NNs to control and system identification were also addressed there.

However, most of the work mentioned above is in the form of indirect adaptive control or has the problems of complex training methods and system structures. Thus, this paper proposes a simple algorithm based on the BP for a class of nonlinear systems typified by process-control applications. The proposed NN-controller is trained by using the system's output errors directly, with little *a priori* knowledge of the controlled plant.

In Section 2, the control problem using NNs is stated formally, and the basic structure of the proposed NN-controller is analyzed. The training algorithm is developed in Section 3. Section 4 presents a procedure for designing the NN-controller, and addresses the problems related to its implementation. Section 5 summarizes the simulation results. The NN-controller is used to control the middle-point temperature. The controlled system is characterized by the properties of long response delay, nonlinearity of dead zone and saturation, and process noise. The performance of the NN-controller is also compared with a PI controller. Finally, the paper concludes with Section 6.

## 2. PROBLEM STATEMENT AND THE NN-CONTROLLER

A controlled plant can be viewed as a mapping from control input to system output:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t),$$

where $\mathbf{x} \in \mathbf{R}^m$, $\mathbf{y} \in \mathbf{R}^n$, and $\mathbf{u} \in \mathbf{R}^{N_2}$ are the state, system output and input, respectively. The controller of this plant, if it exists, can be represented as a mapping from the system feedback and/or feedforward to control commands:

$$\mathbf{u} = \mathbf{c}(\mathbf{y}, \mathbf{y}_d, t), \tag{1}$$

where $\mathbf{y}_d$ is the desired system output. It is assumed that only the system output is measured.

The objective is to design an NN-controller which will replace a conventional controller. In other words, the NN-controller is cascaded with the controlled plant as shown in Fig. 2, and trained to learn the mapping in Eqn (1). The control input $\mathbf{u}_d(t)$ is required to produce the desired output $\mathbf{y}_d(t)$. The *system-output error* and the *control-input error* are then defined, respectively, by

$$\mathbf{e}_y(t) = \mathbf{y}_d(t) - \mathbf{y}(t), \tag{2}$$

$$\mathbf{e}_u(t) = \mathbf{u}_d(t) - \mathbf{u}(t). \tag{3}$$

The control-input error $\mathbf{e}_u(t)$ is also called the *network-output error*, since $\mathbf{u}(t)$ is the output of the NN-controller. An NN is usually trained by minimizing the network-output error $\mathbf{e}_u(t)$. However, if the NN controller is cascaded in series with the controlled plant, $\mathbf{e}_u(t)$ is not known, since the desired control input $\mathbf{u}_d(t)$ is unknown. So, the immediate problem in designing such an NN-controller is how to train the NN.

One of the well-developed NNs is a multilayer perceptron with BP.[13,14] The basic structure of a three-layer perceptron is shown in Fig. 3. The BP algorithm is based on the gradient algorithm to minimize the network-output error, and is derived from the special structure of the networks. Using the structure in Fig. 3, computation of the NN's output and updating of the NN's weights are summarized in the following five steps:

(1) *Compute the output of the HIDDEN layer*, $X_{1j}$

$$X_{1j}(t)=\frac{1}{1+\exp(-O_{1j}-\theta_{1j})}, \tag{4}$$

$$\text{where } O_{1j}=\sum_{i=1}^{N} W_{ij}X_i(t),$$

$j=1,2,\ldots,N_1$, and $X_i(t)$, $i=1,2,\ldots,N$, are the inputs of the NN.

(2) *Compute the output of the OUTPUT layer*: $X_{2k}$

$$X_{2k}(t)=\frac{1}{1+\exp(-O_{2k}-\theta_{2k})}, \tag{5}$$

$$\text{where } O_{2k}=\sum_{j=1}^{N_1} W_{1jk}X_{1j}(t),$$

$k=1,2,\ldots,N_2$.

(3) *Update the weights from the HIDDEN to the OUTPUT layer*, $W_{1jk}$

$$W_{1jk}(t+\Delta t)=W_{1jk}(t)+\eta_1\delta_{1k}X_{1j}(t), \tag{6}$$

$$\text{where } \delta_{1k}=(X_{2k}^d(t)-X_{2k}(t))X_{2k}(t)(1-X_{2k}(t)), \tag{7}$$

and $X_2^d(t)$ is the desired value of $X_{2k}(t)$.

(4) *Update the weights from the INPUT to the HIDDEN layer*, $W_{ij}$

$$W_{ij}(t+\Delta t)=W_{ij}(t)+\eta\delta_j X_i(t), \tag{8}$$

$$\text{where } \delta_j=\left[\sum_{k=1}^{N_2}\delta_{1k}W_{1jk}\right]X_{1j}(t)(1-X_{1j}(t)). \tag{9}$$

(5) *Update the thresholds*, $\theta_{2k}$, $\theta_{1j}$

$$\theta_{2k}(t+\Delta t)=\theta_{2k}(t)+\eta_{1\theta}\delta_{1k}, \quad \theta_{1j}(t+\Delta t)=\theta_{1j}(t)+\eta_\theta\delta_j, \tag{10}$$

where $\eta$, $\eta_1$, $\eta_\theta$, and $\eta_{1\theta}>0$ are gain factors.

In any control system design, it is desired to specify the system performance in terms of system-output errors, $\mathbf{e}_y(t)=\mathbf{y}_d(t)-\mathbf{y}(t)$, rather than the unknown network-output error $\mathbf{e}_u(t)$. To design such a controller using NNs, the basic principle of multilayer perceptron with BP is adopted because of its ability of universal approximation and its convergent property based on

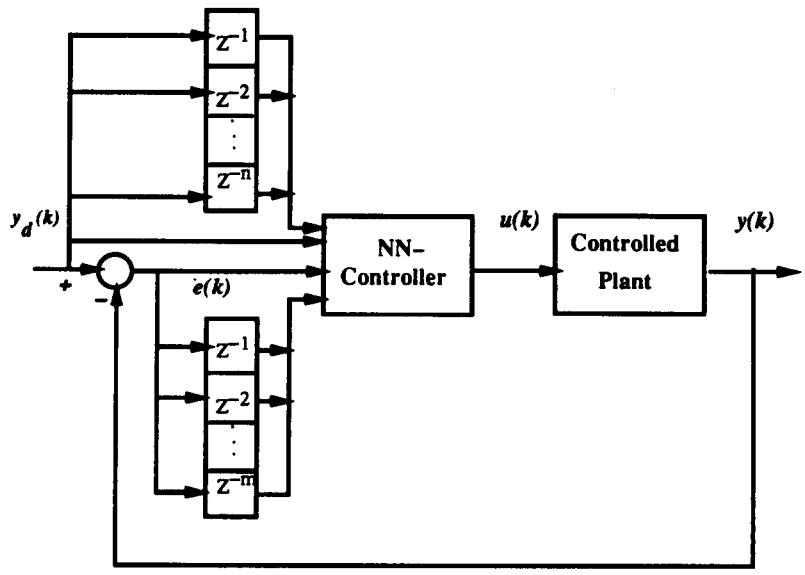


Fig. 2. A control system with an NN-controller.

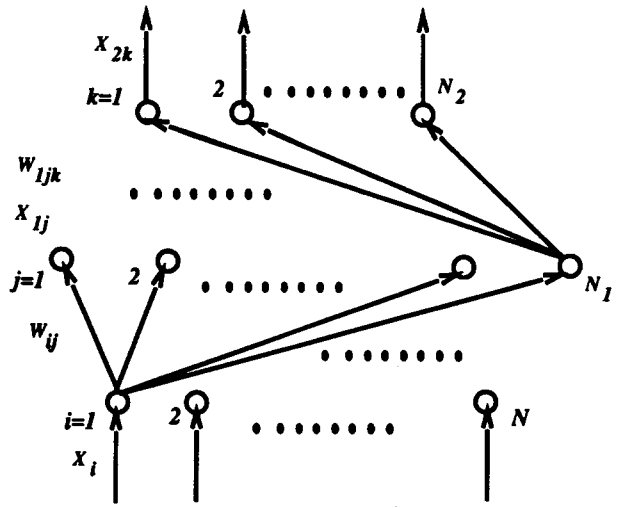


Fig. 3. The basic structure of a multilayer perceptron.

the gradient algorithm.[15] The major obstacle to designing such an NN-controller is to train the NN using system-output errors $\mathbf{e}_y(t)$, rather than the network-output errors $\mathbf{e}_u(t)$. The next section presents a solution to this problem.

## 3. TRAINING AN NN-CONTROLLER WITH SYSTEM-OUTPUT ERRORS

To derive the BP algorithm, the cost function of the network is defined as:

$$E_u(t) = \frac{1}{2}\sum_{k=1}^{N_2} (e_{uk}(t))^2$$

where $e_{uk}(t) = u_{kd}(t) - u_k(t)$ is the network-output error at the $k^{\text{th}}$ node of the OUTPUT layer. The middle-point temperature control system is treated as a single-input, single-output (SISO) system. That is, the input is the feedwater flow rate and the output is the middle-point temperature, and other effects mentioned before are treated as disturbances. Therefore, the cost function of the neural network becomes

$$E_u(t) = \tfrac{1}{2}(e_u(t))^2 = \tfrac{1}{2}(u_d(t) - u(t))^2.$$

As mentioned earlier, $E_u(t)$ is not available since $u_d(t)$ is unknown. Let the system-output error be defined by

$$e_y(t) = y_d(t) - y(t). \tag{11}$$

Then, the cost function in terms of the system-output error is defined as:

$$E_y(t) = \tfrac{1}{2}(e_y(t))^2 = \tfrac{1}{2}(\mathbf{G}(u_d) - \mathbf{G}(u))^2, \tag{12}$$

where the system is represented by $y(t) = \mathbf{G}(u(t))$. Equation (12) is computable from the measurement of the system output. In other words, a function of the network output error is known, though the detailed structure and parameters of the mapping $\mathbf{G}(\cdot)$ may not be known. The NN is to be trained by minimizing the cost function eqn (12).

Using the gradient algorithm, the weights from the HIDDEN to the OUTPUT layer are modified by:

$$W_{1j1}(t + \Delta t) = W_{1j1}(t) + \Delta W_{1j1}, \tag{13}$$

$$\text{where } \Delta W_{1j1} \propto -\frac{\partial E_y(t)}{\partial W_{1j1}(t)}. \tag{14}$$

Note that $u(t) = X_{21}(t)$ in the NN-controller:*

$$\frac{\partial E_y(t)}{\partial W_{1j1}(t)} = -(y_d(t) - y(t))\frac{\partial y(t)}{\partial u(t)}\frac{\partial X_{21}(t)}{\partial O_{21}(t)}\frac{\partial O_{21}(t)}{\partial W_{1j1}(t)}. \tag{15}$$

Because $\dfrac{\partial X_{21}(t)}{\partial O_{21}(t)} = X_{21}(t)\,(1 - X_{21}(t))$,

and $\dfrac{\partial O_{21}(t)}{\partial W_{1j1}(t)} = X_{1j}(t)$, eqn (15) becomes:

$$\frac{\partial E_y(t)}{\partial W_{1j1}(t)} = -(y_d(t) - y(t))\frac{\partial y(t)}{\partial u(t)} X_{21}(t)(1 - X_{21}(t))X_{1j}(t). \tag{16}$$

Substituting eqn (16) into eqn (14) gives:

$$\Delta W_{1j1}(t) = \eta^y_1 \delta^y_{11} X_{1j}(t), \tag{17}$$

$$\text{where } \delta^y_{11} = (y_d(t) - y(t))\frac{\partial y(t)}{\partial u(t)} X_{21}(t)(1 - X_{21}(t)), \tag{18}$$

and $\eta^y_1 > 0$ is a gain factor. The only unknown in eqn (18) is $[\partial y(t)/\partial u(t)]$.

Recall that the network-output error at the OUTPUT layer is defined by:

$$e_u(t) = u_d(t) - u(t). \tag{19}$$

Referring to eqn (18), the system-output error contributed by the control input is defined by

$$e_s(t) = (y_d(t) - y(t))\frac{\partial y(t)}{\partial u(t)}. \tag{20}$$

In the gradient algorithm, the solution converges to a minimum of the cost function if and only if the search is made along the negative direction of the gradient of the

* In fact, $X_{21}(t)$ is the scaled value of $u(t)$. At this stage, it is assumed that $u(t)$ is within the range of (0, 1). The scaling problem will be discussed later.

cost function. BP is based on the gradient algorithm and listed in eqns (6)–(10). Because $u_d(t)-u(t)=X_{21}^d(t)-X_{21}(t)$, eqn (7) becomes:

$$\delta_{11}=e_u(t)X_{21}(t)(1-X_{21}(t)). \tag{21}$$

Substituting eqn (20) into eqn (18) gives:

$$\delta_{11}^y=e_s(t)X_{21}(t)(1-X_{21}(t)). \tag{22}$$

Because both eqns (21) and (22) are derived by applying the gradient algorithm, the necessary and sufficient condition for the convergence of the training algorithm given in eqns (13) and (17) is

$$sign(e_s(t))=sign(e_u(t-d)), \tag{23}$$

where $d$ is the system response delay.

The accurate value of $\left|\frac{\partial y(t)}{\partial u(t)}\right|$ is not important, because the step size can be adjusted by setting $\eta_s\equiv\eta_1^y\left|\frac{\partial y(t)}{\partial u(t)}\right|$. Certainly, this requires $\left|\frac{\partial y(t)}{\partial u(t)}\right|<\infty$, $\forall t$. However, for a general nonlinear system, the sign of $\frac{\partial y(t)}{\partial u(t)}$ at each instant is not available and difficult to estimate on-line. To overcome this problem, a class of systems is characterized by the following two definitions.

**Definition 1**: If the system output monotonically increases (decreases) as the control input of a controlled plant increases, then the system is called *positive-responded (negative-responded)*. Both positive-responded and negative-responded systems are called *monotone-responded*.

**Definition 2**: For an SISO system $y(t)=\mathbf{G}(u(t))$, if the system is positive-responded (negative-responded), then the *system direction* is defined by $D(\mathbf{G})=1$ ($D(\mathbf{G})=-1$).

Definition 1 characterizes a class of systems. For example, a linear system is cascaded with an element of pure response delay, dead zone and saturation. The middle-point temperature control system is a system with monotone response to the control input—feedwater flow rate.

For a positive-responded system, from eqns (19)–(23),

$$sign(u_d(t-d)-u(t-d))=sign(y_d(t)-y(t)). \tag{24}$$

Similarly, for a negative-responded system,

$$sign(u_d(t-d)-u(t-d))=-sign(y_d(t)-y(t)). \tag{25}$$

From eqns (24) and (25), it can be concluded that for an SISO monotone-responded system in order to train the NN-controller in Fig. 3 using system-output error, the condition for convergence given in eqn (23) becomes

$$sign(u_d(t-d)-u(t-d))=sign(y_d(t)-y(t))\ D(\mathbf{G}), \tag{26}$$

assuming that $d$ is somehow known. Therefore, the corresponding algorithm for updating the weights from the HIDDEN to OUTPUT layer is

$$W_{1j1}(t+\Delta t)=W_{1j1}(t)+\eta_1^y\delta_{11}^y X_{1j}(t), \tag{27}$$

where $\delta_{11}^y=(y_d(t)-y(t))D(\mathbf{G})X_{21}(t)(1-X_{21}(t))$.

## 4. DESIGN OF THE NN-CONTROLLER

Figures 2 and 3 show the basic structures of the system and the NN-controller, respectively. The choice of the NN's inputs should reflect the desired and actual status of the controlled system. Therefore, the inputs of the NN-controller are usually the system's desired and actual outputs, and tracking errors:

$$y_d(t), y_d(t-\Delta t), \ldots, y_d(t-m_1\Delta t),$$
$$y(t), y(t-\Delta t), \ldots, y_d(t-m_2\Delta t),$$
$$e_y(t), e(t-\Delta t), \ldots, e_y(t-m_3\Delta t),$$

where $m_1$, $m_2$ and $m_3>0$ are integer constants, and $e_y(t)=y_d(t)-y(t)$. The number of the HIDDEN nodes depends on the controlled plant under consideration, and is selected experimentally.

Based on eqn (27), the formulae for updating the weights from the INPUT to the HIDDEN layer and the thresholds are derived using the same procedure given in Section 3. The computation of the NN-controller is then summarized as follows:

*A. Compute the output of the HIDDEN layer, $X_{1j}(t)$*

$$X_{1j}(t)=\frac{1}{1+\exp(-O_{1j}-\theta_{1j})},$$

$$\text{where } O_{1j}=\sum_{i=1}^{N} W_{ij}X_i(t),\quad j=1,2,\ldots,N_1,$$

and $X_i(t)$, $i=1,2,\ldots,N$, are the inputs of the NN-controller.

*B. Compute the output of OUTPUT layer, $X_{21}(t)$*

$$X_{21}(t)=\frac{1}{1+\exp(-O_{21}-\theta_{21})},$$

$$\text{where } O_{21}=\sum_{k=1}^{N_1} W_{1j1}X_{1j}(t).$$

*C. Update the weights from HIDDEN to OUTPUT layer, $W_{1j1}(t)$*

$$W_{1j1}(t+\Delta t)=W_{1j1}(t)+\eta_1^y\delta_{11}^y X_{1j}(t),$$

$$\text{where } \delta_{11}^y=(y_d(t)-y(t))D(\mathbf{G})X_{21}(t)(1-X_{21}(t)).$$

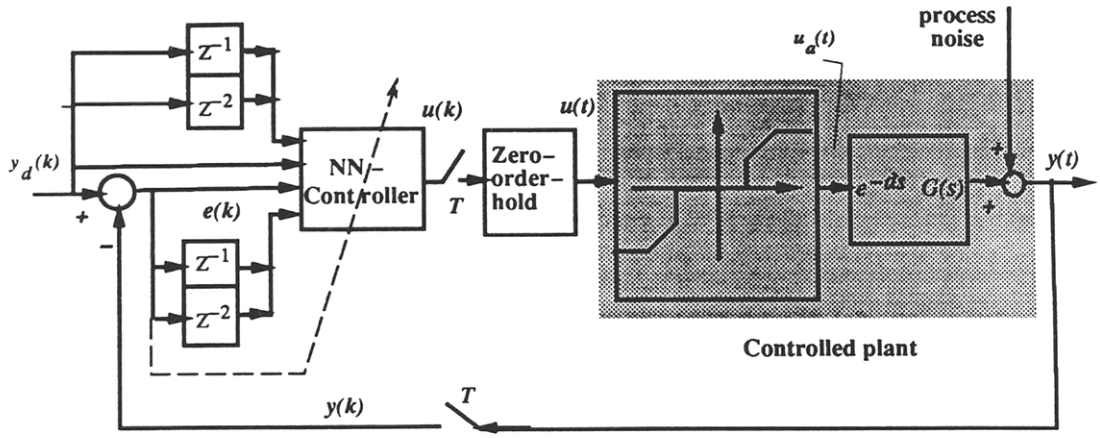


Fig. 4. The structure of the NN-based control system.

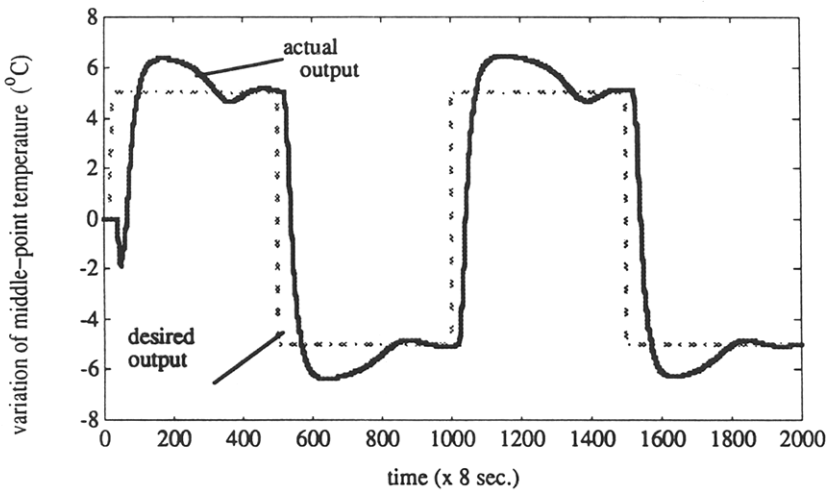


Fig. 5. System response with NN-controller, when *dead_zone* = 5.0, and $R = 0.0$.

*D. Update the weights from INPUT to HIDDEN layer, $W_{ij}(t)$*

$$W_{ij}(t+\Delta t) = W_{ij}(t) + \eta^y \delta_j^y X_i(t),$$
$$\text{where } \delta_j^y = \delta_{11}^y W_{1j1} X_{1j}(t)(1 - X_{1j}(t)).$$

*E. Update the thresholds: $\theta_{21}$ and $\theta_{1j}$*

$$\theta_{21}(t+\Delta t) = \theta_{21}(t) + \eta_{1\theta}^y \delta_{11}^y, \quad \theta_{1j}(t+\Delta t) = \theta_{1j}(t) + \eta_\theta^y \delta_j^y,$$

where $\eta_1^y > 0$, $\eta^y > 0$, $\eta_{1\theta}^y$ and $\eta_\theta^y > 0$ are gain factors.

Another problem in designing an NN-controller is the choice of scaling factors. The sigmoid function in NN computation forces the NN outputs to be within the range of (0, 1), although the control inputs are limited by the range of actuator ($U_{\min}$, $U_{\max}$). Therefore, the NN outputs should coincide with, or be a little narrower than, the range of the actuator's limits. The output of the NN-controller is then computed by $u(t) = X_{21}(t)(U_{\max} - U_{\min}) + U_{\min}$.

Generally, an NN works in the mode of *train-first-then-operate*. In other words, an NN is put into operation only after it is "well-trained". By "well-trained", is meant that the weights of the NN need not be modified any more. However, for a time-varying system, it is meaningless to say that an NN is "well-trained", since the system always changes with time. Thus, not updating the weights for a time-varying system may result in the system going out of control. It is therefore always necessary to update the weights of the NN-controller, though the updating may not be done during every sampling interval.

## 5. SIMULATION RESULTS

The proposed NN-controller is applied to the middle-point temperature system which can be characterized by a linear system cascaded with a nonlinear element as the result of dead zone and actuator limits, and a pure time delay caused by system response delay. The simulated system is a simplified model of the middle-point temperature system of a once-through boiler. The principal specifications of the boiler are as follows:

| | |
|---|---|
| Full-load main steam flow: | 1000 t/h. |
| Full-load main steam pressure: | 170 kg/cm$^2$. |
| Main steam temperature: | 555 °C. |
| Fuel: | heavy oil or crude oil. |

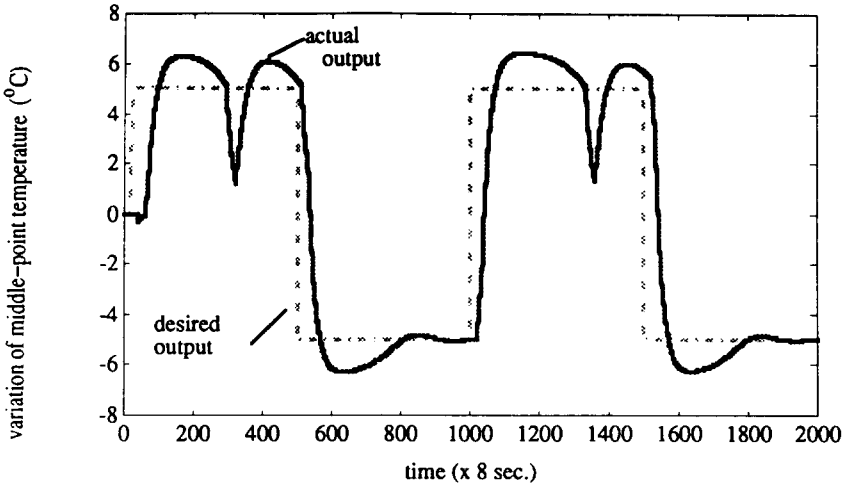


Fig. 6. System response with NN-controller, when *dead_zone* = 7.0, and $R = 0.0$.

Simulations were conducted while emphasizing the ability to overcome the negative effects of dead zone, saturation, long response delay, and process noise. The system is represented by an ARMAX model:

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-d) + \mathbf{C}(z^{-1})\xi(k), \quad (28)$$

where

$$\mathbf{A}(z^{-1}) = 1 - 0.45181z^{-1} - 0.47546z^{-2},$$
$$\mathbf{B}(z^{-1}) = -0.04560z^{-1} - 0.00404z^{-2},$$
$$\mathbf{C}(z^{-1}) = 1 - 0.35740z^{-1} - 0.03392z^{-2},$$
$$d = 18 \text{ sampling intervals.}$$

Here the sampling interval is 8 s, $y(k)$ and $u(k)$ are the variation of middle-point temperature and variation of feedwater flow rate at a discrete time $k$, respectively, and $\xi(k)$ is an uncorrelated random sequence with zero mean and variance $R$ representing the process noise. The NN-controller has no *a priori* knowledge about this model except its response direction.

A nonlinear element of dead zone and saturation is cascaded with the system eqn (28) to model an actuator, which is described by

$$u_a(t) = \begin{cases} U_{\min} & \text{if } |u(t)| \geq dead\_zone \text{ and } U_{\min} > u(t) \\ 0, & \text{if } |u(t)| < dead\_zone \\ u(t), & \text{if } |u(t)| \geq dead\_zone \\ & \text{and } U_{\min} \leq |u(t)| \leq U_{\max} \\ U_{\max}, & \text{if } |u(t)| \geq dead\_zone \text{ and } u(t) > U_{\max}. \end{cases} \quad (29)$$

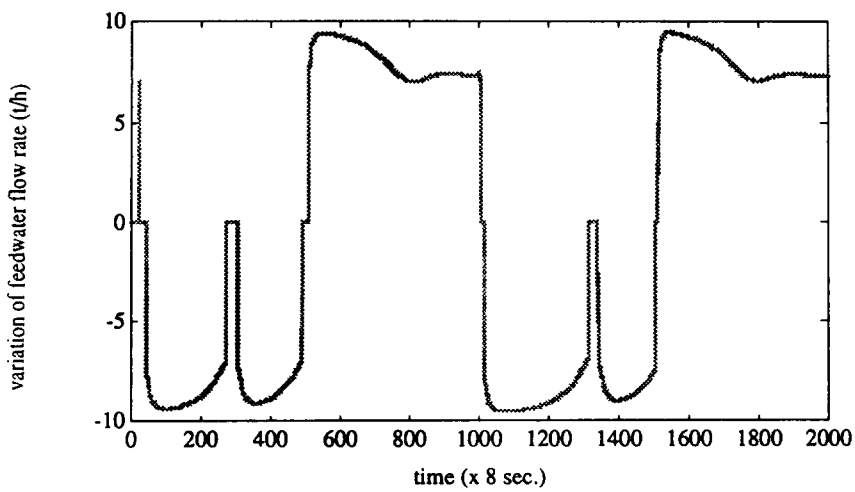


Fig. 7. System control input with NN-controller, when *dead_zone* = 7.0, and $R = 0.0$.

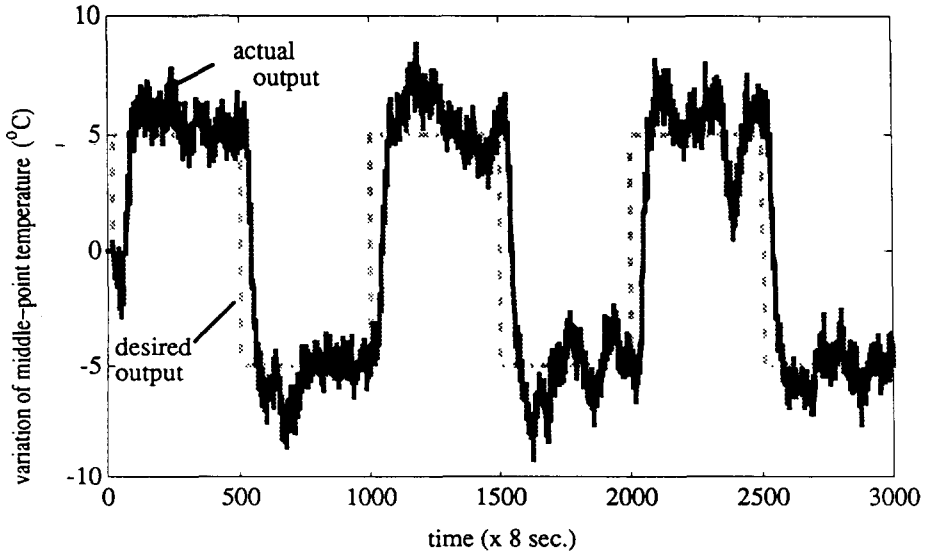


Fig. 8. System response with NN-controller, when *dead_zone* = 5.0, and $R = 0.5$.

The dead zone and saturation are treated as unknown properties of the controlled plant. The NN-controller is required to overcome their negative effects by the NN's learning ability. Actually, since the system response direction will not be changed by adding dead zone and saturation, the NN-controller should work well. Moreover, there is no special consideration for process noise in the design of the NN-controller, as in other deterministic controller designs, though controllers have to be tested for the capability of noise rejection.

To reflect the status of the controlled system, the inputs of the NN-controller are chosen as the desired system outputs and the output errors:

$$y_d(k), \quad y_d(k-1), \quad y_d(k-2),$$
$$y_d(k)-y(k), \quad y_d(k-1)-y(k-1), \quad y_d(k-2)-y(k-2).$$

That is, there are six inputs at the INPUT layer of the NN-controller ($N = 6$). Note that the middle-point temperature system is in fact a high-order system, though it can be approximately modeled by a low-order linear system with a long pure time delay. Certainly, the NN-controller is not used to model this high-order controlled plant but to control it. So, it may not be necessary to use the same time delay of the controlled plant (18 sampling intervals in eqn (28)) as its inputs. The NN-controller was also tested with more delayed inputs. The results are not superior to those presented below. The number of the HIDDEN nodes is selected to be three ($N_1 = 3$). The overall system structure is sketched in Fig. 4.

The main simulation results are summarized as follows.

1. When *dead_zone* = 5.0, $U_{max} = 10.0$, $U_{min} = -10.0$, and no process noise ($R = 0.0$), the results

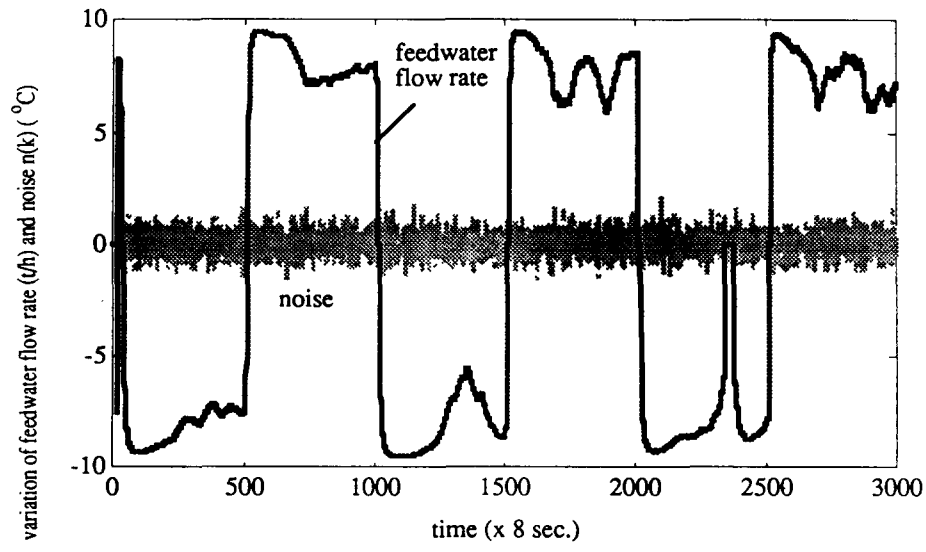


Fig. 9. System control input with NN-controller, when *dead_zone* = 5.0, and $R = 0.5$.

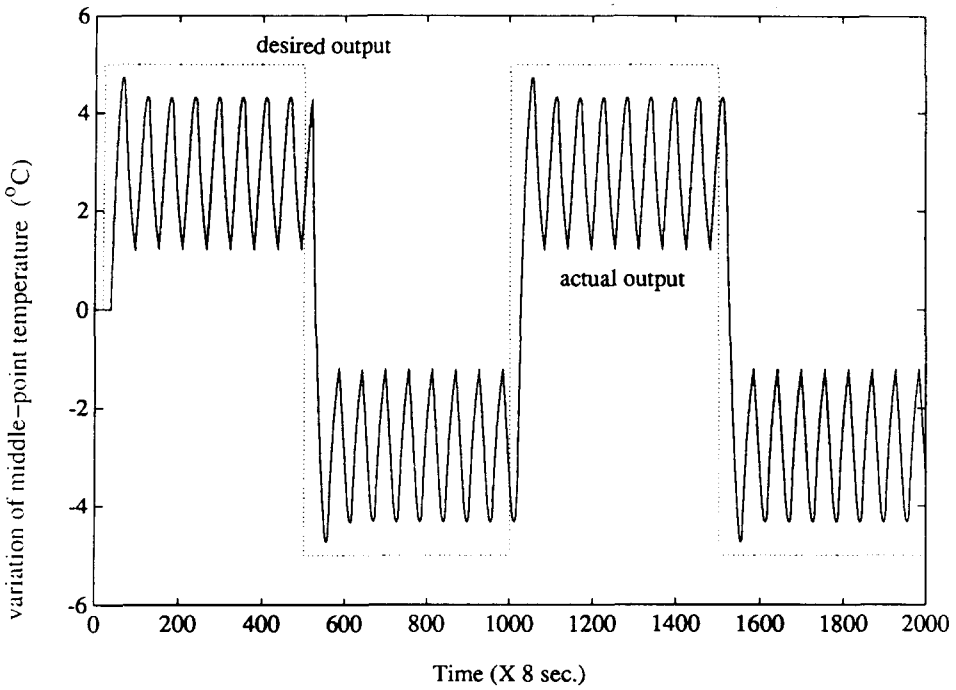


Fig. 10. System response with a PI-controller, when *dead_zone* = 5.0, and $R = 0.0$.

are plotted in Fig. 5. The initial weights of the NN are selected randomly, and the NN's weights converge within 150 sampling intervals.

2. When *dead_zone* = 7.0, $U_{max} = 10.0$, $U_{min} = -10.0$, and $R = 0.0$, Figs 6 and 7 present the system response and the corresponding control input, respectively. Obviously, a large dead zone affects the system performance seriously, but the NN-controller still works well.
3. When *dead_zone* = 5.0, $U_{max} = 10.0$, $U_{min} = -10.0$, and $R = 0.5$ to test the ability of noise rejection, the desired and actual system output responses are plotted in Fig. 8. The corresponding control input and the process noise are shown in Fig. 9, where $n(k) = \xi(k) - 0.35740\xi(k-1) - 0.03392\xi(k-2)$.

To compare the performance of the proposed NN-controller with that of a PID-type controller, a PI

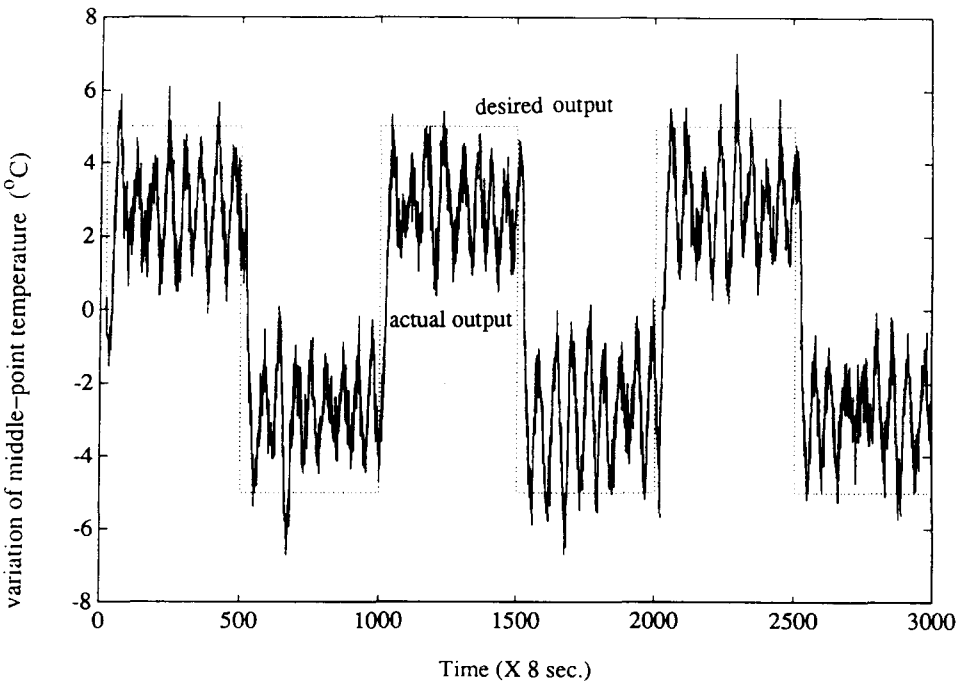


Fig. 11. System response with a PI-controller, when *dead_zone* = 5.0, and $R = 0.5$.

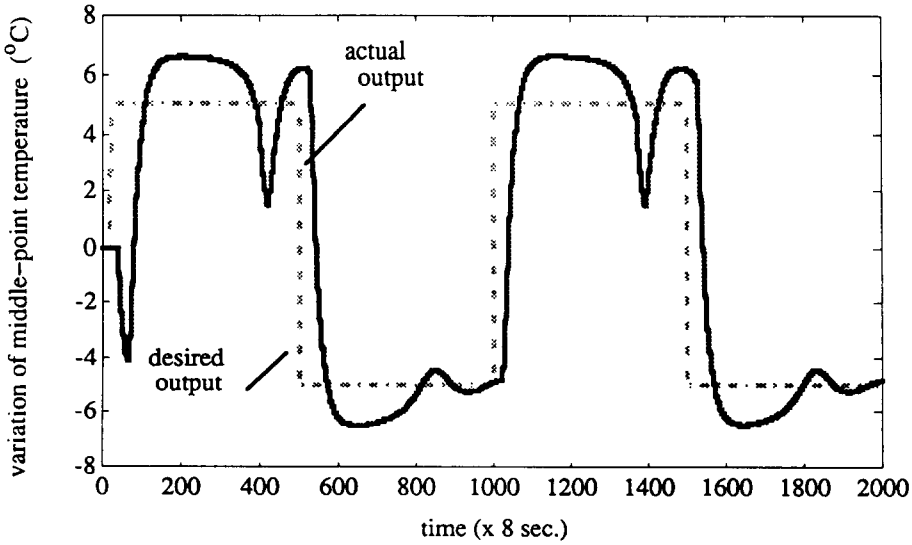


Fig. 12. System response with NN-controller for six hidden nodes.

controller has been designed for the middle-point temperature system. The PI controller is

$$u(k) = -K_p(y_d(k) - y(k)) - K_I \sum_{i=0}^{k} (y_d(i) - y(i)),$$

with $K_p = 2.2$ and $K_I = 0.3$. When $dead\_zone = 5.0$, $U_{max} = 10.0$, $U_{min} = -10.0$ and no process noise ($R = 0.0$), the system response controlled by the PI controller is plotted in Fig. 10. According to this figure, one should decrease $K_p$ in order to reduce the oscillation. However, due to the effects of the dead zone, one cannot make any notable improvement in the system performance. On the other hand, due to the effects of the long time delay, increasing $K_p$ will lead to an unstable response. When $dead\_zone = 5.0$, $U_{max} = 10.0$, $U_{min} = -10.0$ and $R = 0.5$, the result of the PI controller is plotted in Fig. 11. Comparing Fig. 10 with Fig. 5, and Fig. 11 with Fig. 8, it was concluded that the performance of the NN-controller is much better than that of the PI controller. Actually, a PI controller cannot perform well for a system with a long time delay, dead zone, saturation, and process noise.

The above simulation results indicate how well the proposed NN-controller performs. In the NN-controller, the system-output error is computed from the actual measurements. As *a priori* knowledge, the system direction is easily obtained from the physical property of the controlled plant. To test the need for eqn (27), $-D(\mathbf{G})$ is used in the training algorithm, which instantly results in the NN's divergence.

The remaining problem is how to choose the number of the HIDDEN nodes. There is no systematic way to choose the number of the nodes at the HIDDEN layer(s) to approximate a given mapping. Therefore, selection of hidden nodes may depend on experiments. Figure 12 shows the result using $N_1 = 6$, $dead\_zone = 5.0$, $U_{max} = 10.0$, $U_{min} = -10.0$ and $R = 0.0$. Comparing Fig. 12 with Fig. 5, one can see that adding more HIDDEN nodes does not improve the system performance. But adding more nodes will improve the system's reliability.

## 6. CONCLUSION

Focusing on process control systems, a new direct adaptive controller using neural networks has been designed and tested for the middle-point temperature control in a thermal power plant. For such a control system, the negative effects of a long system response delay, nonlinear elements with dead zone and saturation, and process noises, are the main obstacles in designing a high performance controller and fine tuning its parameters. The proposed NN-controller can replace a conventional controller, and is shown to overcome all of the problems mentioned above.

A training algorithm is derived based on BP, enabling the NN to be trained with system-output errors, rather than the network-output errors. In the BP algorithm, it is required to modify the weights by using the network-output error which is not known when a multilayer perceptron is applied directly to the controlled plant. Therefore, the proposed algorithm enhances the NN's ability to handle control applications. The only *a priori* knowledge about the controlled plant is the direction of its response, which is usually easy to determine. The proposed NN-controller has been applied to the middle-point temperature control in a thermal power plant, and extensive simulations conducted, showing very promising results. Good performance, a simple structure and algorithm, and the potential for fault tolerance make the proposed NN-controller attractive for process-control applications.

*Acknowledgement*—The authors are grateful to the anonymous reviewers for their constructive comments on an earlier version of this paper.

## REFERENCES


1. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math. Control, Signals and Syst.* **2**, 303–314 (1989).
2. Werbos P. J. Back propagation and neurocontrol: a review and prospectus. *Proc. of 1989 Int. Joint Conf. on Neural Networks* **1**, I209–I216 (1989).
3. Chen V. C. and Pao Y. H. Learning control with neural networks. *Proc. of 1989 IEEE Conf. on Robotics and Automation*, pp. 1448–1453 (1989).
4. Psaltis D., Sideris A. and Yamamura A. A. A multilayered neural network controller. *IEEE Control System Mag.* pp. 17–21 (1988).
5. Levin E., Gewirtzman R. and Inbar G. F. Neural network architecture for adaptive system modeling and control. *Proc. of 1989 Int. Joint Conf. on Neural Networks* **2**, 311–316 (1989).
6. Gu Y. L. On nonlinear system invertibility and learning approaches by neural networks. *Proc. of 1990 American Control Conference* **3**, 3013–3017 (1990).
7. Guez A. and Selinsky J. A neuromorphic controller with a human teacher. *Proc. of 1988 Int. Conf. on Neural Networks* **2**, II595–II602 (1988).
8. Bar-Kana I. and Guez A. Neuromorphic adaptive control. *Proc. of 1989 IEEE Int. Conf. on Decision and Control* **2**, 1739–1743.
9. Kraft L. G. and Campagna D. P. A comparison of CMAC neural network and traditional adaptive control. *Proc. of 1989 American Control Conference* **1**, 884–889 (1989).
10. Miller W. T., Hewes R. P., Glanz F. H. and Kraft L. G. Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. *IEEE Trans. on Robotics and Automation* **6**, 1–9 (1990).
11. Narendra K. S. and Parthasarathy K. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks* **1**, 4–27 (1990).
12. Werbos P. J. Neural networks for control and system identification. *Proc. of 1989 IEEE Int. Conf. on Decision and Control* **1**, 260–265 (1989).
13. Werbos P. J. Back propagation: past and future. *Proc. of 1988 Int. Conf. on Neural Networks* **1**, I343–I353 (1988).
14. Rumelhart D. E. and McCelland J. L. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, Cambridge, MA (1986).
15. Wang J. and Malakooti B. On training of artificial neural networks. *Proc. of 1989 Int. Joint Conf. on Neural Networks* **2**, 387–393 (1989).