A NEW ON-LINE PID TUNING METHOD USING NEURAL NETWORKS

A.B. Rad, T.W. Bui , Yan Li and Y.K. Wong

Department of Electrical Engineering
The Hong Kong Polytechnic University
Hunghom, Hong Kong

Abstract: This paper presents an on-line PID tuning control method, based on the parameters of a first-order plus dead-time (FOPDT) model, which are obtained by using Neural Networks (NN). The outputs of the neural networks are the three parameters of the FOPDT model. By combining this algorithm with a conventional PID controller, an adaptive controller is obtained which requires very little a priori knowledge about the plant under control. The simplicity and feasibility of the scheme for real-time control provide a new approach for implementing neural network applications for a variety of on-line industrial control problems. Simulation results demonstrate the feasibility and adaptive property of the proposed scheme. *Copyright © 2000 IFAC*

Keywords: Adaptive Control, PID, Neural Networks, Back-Propagation algorithm

1. INTRODUCTION

The ubiquitous Proportional-Integral-Derivative (PID) controller is regarded as the most widely used controller due to its simple structure, easy implementation and robust performance. However, the dynamics of many systems exhibit complex characteristics such as non-linearity, time-varying parameters and time-delay which impose different controller parameters each time the system operating conditions are varied and/or disturbances are added to the system. This often leads to a poor control performance if one uses a conventional fixed-parameters PID controller to handle the mentioned problems. A possible way to circumvent the control problem in these situations is to employ some form of adaptive control. Several PID auto-tuners and self-tuners are reported in the literature whereby the parameters of a PID controller are adjusted on-line (Astrom, *et al.*, 1993; Smith and Corripio, 1997). In the last few years, the applications of the Artificial Neural Networks (ANN) in system identification and control has been reported (Narendra and Parthsarathy, 1990; Hunt *et al.*, 1992). In particular, the design of both *direct* and *indirect* adaptive controllers based on neural networks has gained momentum recently. Model based approaches in the context of *Internal model control (IMC)* or *Model predictive control (MPC)* have been studied extensively by several researchers mainly in the form of indirect adaptive control (Tanomaru and Omatu, 1992; Mills *et al.*, 1994). In this control strategy, a sub-network – sometimes called emulator- is trained either off-line or on-line before the control phase. Another neural network is then designed based on the inverse of the model. However, some of these approaches can be regarded as facilitators and not true adaptive controllers since the identification is carried out off-line. In the direct adaptive control strategies (Noriega and Wang, 1998; Cui and Shin, 1993), the error used for training the neural networks is the output error of the controlled plant rather than the output error of the network. In this strategy, the plant is viewed as an additional but not modifiable layer of the neural network. The main problem with this scheme is that the exact calculation of the partial derivatives of a performance index with respect to the output of the neural controller is generally required which in turn is expedient of the knowledge of the system Jacobian, i.e. dy/du where y is the plant output and u is the controller output.

The approximating method of high-order systems with a first-order plus dead time using neural networks is described in the next section. An on-line PID tuning control method is derived in Section 3. Simulations of time responses for different systems are given in Section 4. Finally, the paper is concluded in Section 5.

2. LOWER-ORDER APPROXIMATION OF SYSTEMS WITH NEURAL NETWORKS

The discussion will be restricted to a single-input / single-output (SISO) models in this paper. A high order system can be approximated by a first order dead time model in form of:

$$\frac{Y(s)}{U(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_m}{s^n + a_1 s^{n-1} + \ldots + a_n} e^{-\theta s} \approx \frac{K \cdot e^{-\tau \cdot s}}{Ts+1} \qquad (1)$$

Here, K is gain, T is the dominant time constant, τ is the apparent dead-time and $m \leq n$. $Y(s)$ and $U(s)$ are the Laplace transformed output and input signals respectively. The parameters K, T and τ can be determined from various methods (Ziegler and Nichols, 1942; Smith, 1972; Tsang and Rad, 1995). Smith's (1972) method is a popular approach whereby the parameters are determined from the process reaction curve by injecting a step signal to the system. Although, this method is convenient and accurate, however, it should be carried out off-line and it is very sensitive to noise. Furthermore, in case of time varying parameters, this process should be repeated.

The construction of the on-line approximating approach is shown in Fig. 1. As shown in this figure, the control signal u(t) is applied to the high-order system, the neural network, and the FOPDT model generator at the same time. The outputs of neural network are the three parameters, namely, the gain K, the time constant T, the dead-time τ of the approximated FOPDT model of the high-order system. These three parameters are sent to the first-order plus dead time model generator to get an output of the model. The error between output of the plant and the output of the model is used to train the weights of the neural networks. The training process tends to force the output of the FOPDT model generator to approximate the output of the system. Thus, the inputs of the FOPDT model generator are the approximating parameters of the first-order model for the high-order system. The output of the FOPDT model is expected to match the output of the high-order system after the neural network converges.

The transfer function of the FOPDT model generator is rewritten below:

$$\frac{Y_m(s)}{U(s)} = \frac{K \cdot e^{-\tau \cdot s}}{Ts+1} \qquad (2)$$

According to the convolution theorem, we get:

$$y_m(t) = \frac{K}{T} \cdot \int_{0^-}^{t^-} u(x-\tau) \cdot \exp(-(t-x)/T)\, dx \qquad (3)$$

Where, $u(x)$ is the input of the FOPDT model generator. The model output $y_m(t)$ depends on the parameters K, T and τ at each time instant.

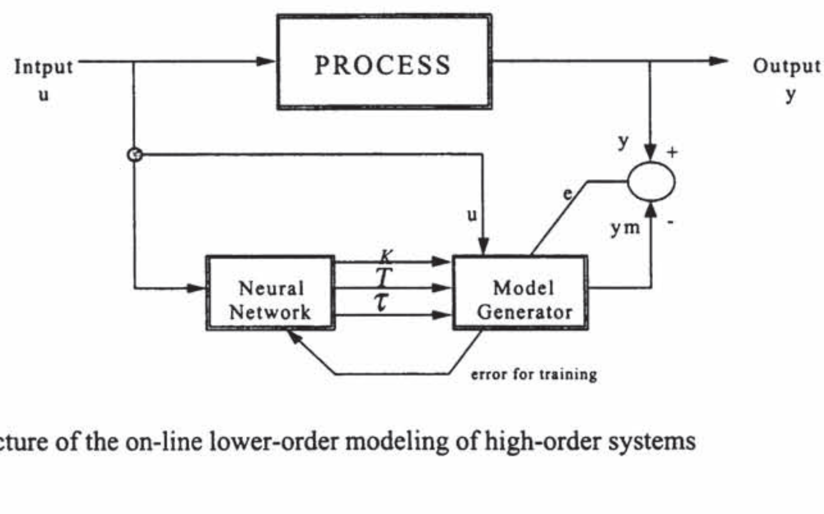

Figure 1: The structure of the on-line lower-order modeling of high-order systems

The neural network architecture used for plant modeling is a three-layer feed-forward network with *1* node in the input layer, *8* nodes in the hidden layer, and *3* nodes in the output layer. The basic structure, having one hidden layer with sigmoid function, has been shown to be powerful enough to produce an arbitrary mapping among variables. Thus, a three layers network is usually used for control applications. The activation function used here is the standard sigmoidal function with range between 0 and 1. To train the above neural network, a direct learning strategy is employed for on-line training. As the desired outputs of the neural network are unknown, the FOPDT model generator is considered as an additional but not modifiable layer of the neural network. Back-Propagation (BP) algorithm is used to update the weights of the neural network. The algorithm consists of two passes, forward pass and backward pass. The calculation of the forward pass and updating the connection weights from the input layer to hidden layer are the same as those in the standard BP algorithm. To update the connection weights from the hidden layer to the output layer, the momentum technique (Battiti, 1989) is employed. The weight adjustment in each iteration is derived below. The error function E is defined as:

$$E = \frac{1}{2}\sum_{t=1}^{r}(y - y_m)^2 \tag{4}$$

Where r is the number of input/output pairs available for training the network, y and y_m are the output of the plant and the output of the FOPDT model at any time instant t.

Within each time interval from t to $t+1$, the BP algorithm is used to update the connection weights, according to the following relationship:

$$W_{ij}(t+1) = W_{ij}(t) - \eta \cdot \frac{\partial E}{\partial W_{ij}(t)} + \alpha \cdot \Delta W_{ij}(t) \tag{5}$$

Here, η is the learning rate; α is the momentum factor; Δw_{ij} is the amount of the previous weight change. Using the chain rule, one has

$$\frac{\partial E}{\partial W_{ij}(t)} = \frac{\partial E}{\partial y_m(t)} \cdot \frac{\partial y_m(t)}{\partial Y_n(t)} \cdot \frac{\partial Y_n(t)}{\partial W_{ij}(t)}$$

$$= -(y(t) - y_m(t)) \cdot \frac{\partial y_m(t)}{\partial Y_n(t)} \cdot X_i(t) \cdot (1 - X_i(t)) \cdot X_j(t) \tag{6}$$

Here X_i is the output of the i^{th} node of the output layer; X_j is the input vector of the nodes of the j^{th} output layer. $Y_n(t)$ is $3x1$ input vector of the FOPDT model (the output of vector of the neural network)

$$\frac{\partial y_m(t)}{\partial Y_n(t)} = \frac{\partial y_m(t)}{\partial K} \cdot \frac{\partial y_m(t)}{\partial T} \cdot \frac{\partial y_m(t)}{\partial \tau} \tag{7}$$

It is straightforward to show that:

$$\frac{\partial y_m(t)}{\partial K} = \ell^{-1}\left[\frac{e^{-\tau \cdot s}}{T \cdot s + 1} u(s)\right] \tag{8}$$

$$\frac{\partial y_m(t)}{\partial T} = \ell^{-1}\left[\frac{-sKe^{-\tau \cdot s}}{(Ts+1)^2} u(s)\right] \tag{9}$$

$$\frac{\partial y_m(t)}{\partial \tau} = \ell^{-1}\left[\frac{-sKe^{-\tau s}}{Ts+1} u(s)\right] \tag{10}$$

where ℓ^{-1} stands for inverse Laplace transform.

3. ON-LINE PID TUNING CONTROL METHOD USING NEURAL NETWORK

The control structure for the on-line PID tuning is shown in Fig. 2. There are two parts in the control structure of the on-line PID tuning method. The first part, which was described in the previous section, is

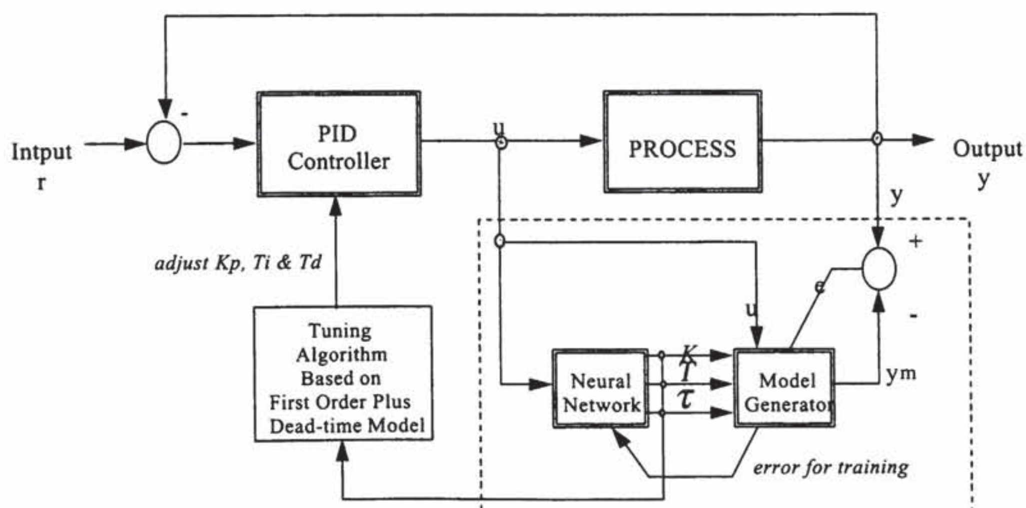

Figure 2: Overall structure of the auto-tuning PID controller

the approximation of high-order systems with FOPDT using neural networks, and the second part is the design of the PID controller. Normally, the parameters of a PID controller can be obtained after the corresponding parameters of a FOPDT model of the high-order system are known. There are many tuning methods based on the parameters of FOPDT model, such as, Ziegler and Nichols (ZN) (1942) ultimate cycle tuning formulae, Minimum Error Integral Tuning Formulas, Minimum IAE, 5% Overshoot tuning formulas (Smith and Corripio, 1997), and Refined Ziegler-Nichols tuning formulae (Hang, *et al.*, 1991). Although ZN ultimate cycle tuning algorithm is not the best tuning method, it is the most widely known PID tuning algorithm. In principle, any other tuning method can be integrated with the proposed identification algorithm; however, to demonstrate the merits of this approach, we have used the ZN method. The parameters of the PID controller suggested by ZN algorithm is:

$$K_p = 0.6K_u \qquad T_I = 0.5T_u \qquad T_d = 0.125T_u \tag{11}$$

Here, K_p, T_i, T_d, K_u and T_u are the proportional gain, integral time constant, derivative time constant, the ultimate gain and the ultimate period respectively, which are calculated from the FOPDT model of the high-order plant (Smith and Corripio, 1997).

The output of the PID controller is in the form of

$$\begin{aligned} u(t) &= K_p\left(e(t) + \frac{1}{T_i}\int e(t)dt - T_d \cdot \frac{dy_f}{dt}\right) \\ e(t) &= r(t) - y(t) \\ y_f(s) &= \frac{1}{1 + T_d \cdot s/10}\, y(s) \end{aligned} \tag{12}$$

where $u(t)$, $y(t)$, $r(t)$ are the controller output, process output, and set-point, respectively.

4. CONTROL PERFORMANCE

To show the adaptive behavior of the algorithm, let us consider two processes as:

Process I $\qquad \dfrac{Y(s)}{U(s)} = \dfrac{1.5\, e^{-2.5s}}{(1+s)^2}$

Process II $\qquad \dfrac{Y(s)}{U(s)} = \dfrac{1 - 1.4s}{(1+s)^3}$

As it is noted, the second process is a non-minimum phase system. The set-point was chosen to be a square-wave with an amplitude *0.5* and a period of *40s*. In order to get a more realistic environment, a Gaussian noise with mean zero and variance of 0.001 is injected at the output of the system. We employed a fourth-order Runge-Kutta numerical integration algorithm for all time responses and the integration interval was selected to be 0.01s. The NN also used the same time interval for updating its parameters. The simulation proceeded as follows: the PID controller was initialized with $K_p = 1$, $T_i = 1000$, $T_d = 0.0$. The architecture of the NN was (1,8,3) and a bias term of 0.5 was added to all the hidden nodes. The number of hidden layers was determined by a trial and error procedure. The connection weights were randomly initialized. The learning rate and the momentum parameter were set at 0.8 and 0.1 respectively. The updating of the PID started at time $t = 40s$ based on the estimated parameters $\hat{K}, \hat{T}\, and\, \hat{\tau}$ by the FOPDT neural model. K_u and T_u were then calculated according to these estimated parameters (Smith and Corripio, 1997). The corresponding PID parameters K_p, T_i and T_d were updated from ZN ultimate cycle method (Eq. 11). The control law like other self-tuning methods is based on *certainty equivalent* (Astrom, *et al.*, 1993), i.e., the controller uses the estimated parameters and does not check the validity of these estimates. However, as the time goes on, the estimated parameters of the approximating FOPDT neural model for the corresponding systems tend to converge to a final value. Therefore, the performance of the controller improves with time. In order to demonstrate the adaptive property of the algorithm, at $t = 195s$, the system was changed to process 2 and again at $t = 320s$, it was switched back to process 1. Furthermore, it should be noted that the gain in system 1 and 2 is different (1.5 and 1). It is known that some adaptive controllers can not cope with change in steady-state gain of the controlled system. However, as it is seen in Figure 3, the proposed method can successfully track the system change. Figure 3 shows the overall performance of the proposed algorithm. In this figure, the set-point and the output, the controller signal -output of the PID controller- and the estimated parameters of gain, apparent time delay and the dominant time constant are shown in top, middle and bottom curves respectively. These simulation studies demonstrate the adaptive property of the proposed algorithm. In all these system changes, the neural networks converged and the estimated parameters of the FOPDT also converged to their steady-state values. The closed-loop stability was also maintained since the underlying controller was a PID structure. This controller is known to provide stable and robust control under various conditions. The parameters from all other methods except the proposed one were obtained off-line, in an open-loop excitation with unit step and were noise-free. The values quoted for the proposed algorithm is based on the last measurement before each system change and not the average value. Nevertheless, these measurements match the original system very well especially with the non-minimum phase system.

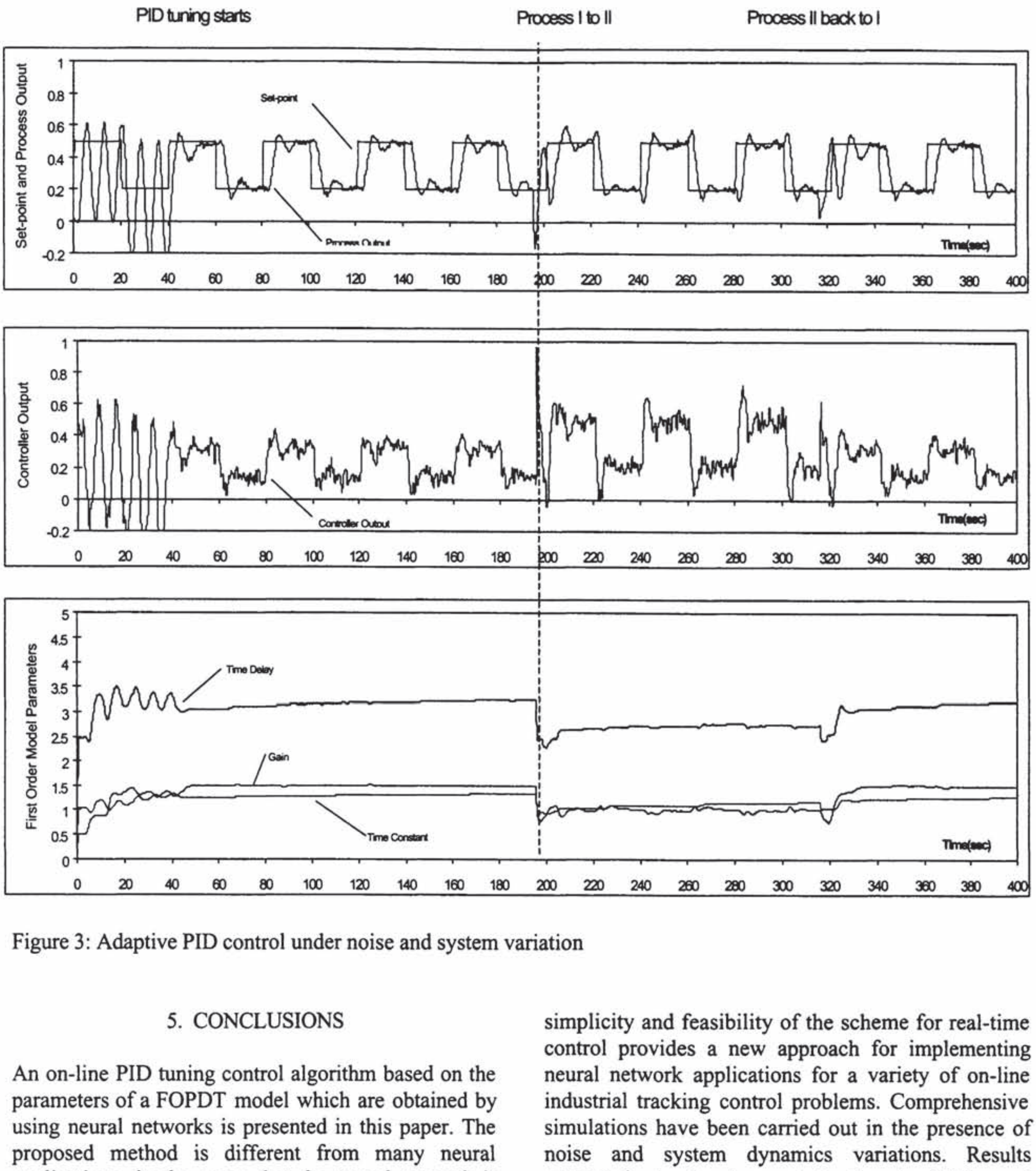

Figure 3: Adaptive PID control under noise and system variation

5. CONCLUSIONS

An on-line PID tuning control algorithm based on the parameters of a FOPDT model which are obtained by using neural networks is presented in this paper. The proposed method is different from many neural applications, in the sense that the neural network is implemented as an approximator rather than a neural identifier or controller. The outputs of the neural network are the three parameters of the FOPDT model for the plant to be controlled. Combining with a conventional ZN PID controller (or any other similar tuning algorithm), an on-line adaptive control using neural networks is proposed. The method can be integrated with any other tuning algorithm, which utilizes the parameters of a FOPDT model. The simplicity and feasibility of the scheme for real-time control provides a new approach for implementing neural network applications for a variety of on-line industrial tracking control problems. Comprehensive simulations have been carried out in the presence of noise and system dynamics variations. Results presented clearly demonstrate the feasibility and adaptive property of the proposed method.

6. ACKNOWLEDGMENTS

The authors gratefully appreciate the support of the Hong Kong Polytechnic University through grants G-V067 and G-YW06.

REFERENCES

Astrom, K. J., T. Hagglund, C. C. Hang and W. K. Ho (1993). Automatic tuning and Adaptation for PID controllers - A survey. *Control Engineering Practice*, **1**, 699-714.

Battiti, R (1989). Accelerating Back-propagation Learning, Two Optimization Methods. *Complex system*, **3**, 331-342.

Cui, X. and K.G. Shin (1993). Direct control and coordination using neural networks. *IEEE Transactions on Systems, man, and Cybernetics*, **23**, 686-697.

Hang, C. C., K. J. Astrom and W. K. Ho (1991). Refinements of the Ziegler-Nichols tuning formula. *Proceedings of IEE – Part D*, **138**, 111-118.

Hunt, K J, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop (1992). Neural networks for control systems - a survey. *Automatica*, **28**, 1083-1112.

Mills P.M., A.Y. Zomaya, and M.O. Tade (1994). Adaptive model-based control using neural networks. *Int. J. Control*, **60**, 1163-1192.

Narendra K. S. and K. Parthsarathy (1990). Identification and control of dynamic systems using neural networks. *IEEE Transactions on neural networks*, **1**, 4-27.

Noriega J.R. and H. Wang (1998). A direct adaptive neural network control for unknown systems and its application. *IEEE Transactions on Neural Networks*, **9**, 27-34.

Smith C. A. and A. B. Corripio (1997). *Principles and Practice of Automatic Process Control 2nd Ed.* John Wiley & Sons, New York.

Smith, C. L. (1972). *Digital Computer Process Control.* International Textbook Company, U.S.A.

Tanomaru J. and S. Omatu (1992). Process control by on-line trained neural controllers. *IEEE Transactions on Industrial Electronics*, **39**, 511-521.

Tsang, K. M. and Rad, A. B. (1995). A new approach to auto-tuning of PID controllers. *International Journal of Systems Science*, **26** 639-658.

Ziegler J. G. and Nichols N. B. (1942). Optimum settings for automatic controllers. *Trans. ASME* 759-768.