Temperature Control of a Water Bath Process: A Comparative Study Between Neuro-control, a Self-tuning Adaptive control, a Generalised Predictive Control, and a Conventional Feedback Control Approach

by

Marzuki Khalid, Rubiyah Yusof, and Sigeru Omatu

Dept. of Information Science and Intelligent Systems,
Faculty of Engineering,
University of Tokushima,
Tokushima 770, Japan.
Tel & Fax: 0886-264739

ABSTRACT

Currently, neural networks are being used to solve problems related to control. One way to determine the reliability of the neuro-control technique is to test it on a variety of realistic problems, and to compare directly with existing traditional control techniques, to see whether it works well and where it needs further refinement. In this article, we compare the neuro-control approach to a self-tuning adaptive control approach, a generalised predictive control approach, and a conventional feedback control approach on a real-time process control system. The neuro-control scheme consists of a backpropagation through time utility where two neural networks are trained, one as an emulator, and the other as a controller. The four systems are compared conceptually and through experimental studies on the same single-input single-output water bath temperature control process. Comparisons, where applicable, are made with respect to methodology, system tracking performance, speed of adaptation, disturbance rejection, effect of long time-delay, and noise rejection. The results show that the neural network controller performs very well and offers encouraging advantages in many aspects over the other three controllers.

1. INTRODUCTION

Neuro-control has begun to create a new horizon in the areas of system control since adaptive control. The capability of artificial neural networks to learn and solve nonlinear control problems, where traditional control approaches have failed, has promised some hope and interest among the control community. Although recently, much emphasis has been given to solve nonlinear control problems, among which are included in [1]-[4], it is rather unprogressive and wasteful if the simple yet powerful neuro-control technique is not applied to solve a wide variety of existing control problems. The self learning ability of artificial neural networks has a significant advantage over many traditional classical, modern, and adaptive control methods where many control problems can now be easily solved with less precise advanced knowledge of the plant. The trend now is to test the neuro-control techniques on a variety of realistic problems, and to compare with existing traditional control techniques, to see whether they work well and where they need further refinement.

A recent study has been done by Kraft and Campagna [5] where a CMAC neural network control approach is compared to two traditional adaptive techniques in controlling a low order plant through simulation. In this paper, we compare the performance of a multilayered backpropagation neural network controller to a self-tuning adaptive controller, a generalised predictive controller, and a conventional feedback controller on a real-time water bath temperature control system. The neuro-control scheme is implemented

using the concept of backpropagation through time [1], [2], [6]-[8] where two neural networks are trained, one as an emulator, and the other as a controller. A novel feature of this scheme as compared to other types of neuro-control schemes is that both the emulator and controller may be further trained in an online way which gives greater robustness to the system. The self-tuning adaptive control system is implemented using the algorithm by Clarke and Gawthrop [9], [10]. An extension of the self-tuning control algorithm which exhibits greater robustness is the generalised predictive control of Clarke, Mohtadi, and Tuffs [11],[12]. The performance of this long range predictive controller is also compared on the same single-input single-output water bath process. As PID control is by far still the most widely used method in industrial applications, it is a worthwhile effort to compare the performance and methodology of this conventional method to the neuro-control approach.

The idea of this paper is to investigate how well each system performs on the same experimental setup. We then make a direct comparison between these four systems both conceptually and through detailed real-time experiments. Comparisons (where applicable) are made with respect to methodology, system tracking performance, speed of adaptation, disturbance rejection, effect of long time-delay, and noise rejection. By comparing the results and methodology of the four approaches, it is hope that future research efforts can extract the best characteristics from each of these different classes of systems for the key to intelligent and efficient control. This paper has been organised as follows: Section II presents a brief description of the water bath temperature control system and a derivation of its mathematical model; Sections III, IV, V and VI give an overview of the neuro-control approach, the selftuning adaptive control approach, the generalised predictive control approach, and the PID control approach, respectively. Results of the experimental studies are compared in Section VII.

2. DESCRIPTION OF THE TEMPERATURE CONTROL SYSTEM

A. Experimental Setup

The temperature control system consists of a Yamato Science Inc. laboratory water bath temperature control process (BT-15 model). A schematic diagram of the experimental setup is shown as in Fig. 1. The system can be divided into five main components: (i) the water bath, (ii) the sensor module, (iii) the PIO interface board, (iv) the microcomputer, and (v) the actuator. A brief description of the five components will be given briefly as follows:

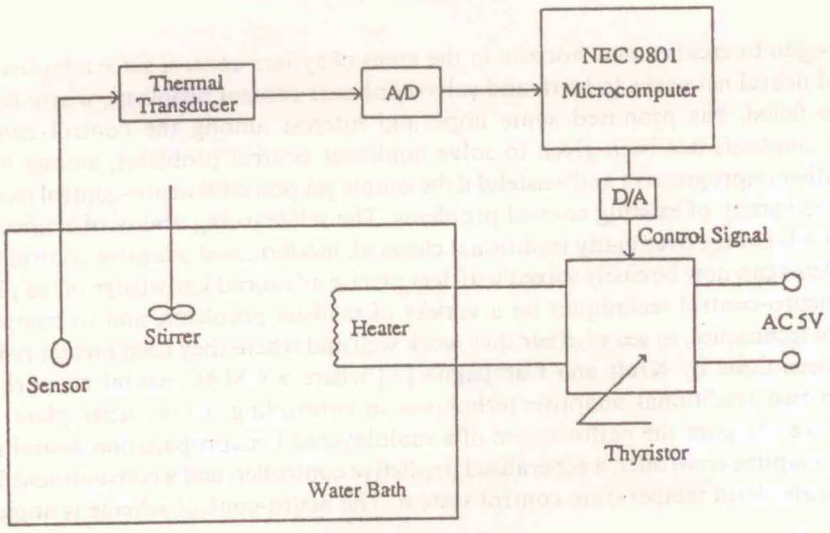

Fig. 1: A block diagram of the water bath temperature control system

(i) The water bath

The capacity of the water bath is 8 litres and its dimension is 250 × 290 × 100 (mm^3). The water bath is heated by a 600W base heater which is connected to a thyristor (Sl6G12S-12) circuit. To ensure even temperature distribution, a stirrer which can rotate at 120 rpm is used.

(ii) The sensor module

The sensor module has been developed using diodes (1S1588) and high gain amplifiers (A741). It consists of a two step amplification circuit and is able to transform the measured temperature from 0°C to 100°C into corresponding voltages of 0V to 10V with a resolution of 0.24°C.

(iii) The PIO interface board

The interfacing circuit consists of an A/D converter, a D/A converter and a programmable peripheral interface device (PD8255A). An external clock is designed to operate the A/D and D/A converter. The clock circuit is designed using crystal oscillator and JK flip flops.

(iv) The microcomputer

The microcomputer used in this experiment is the NEC PC 9801F having an Intel's 8086 16-bit CPU with a 10 MHz clock speed. A simple control routine is written using Microsoft-C to provide the control input to the actuator through the D/A and also to measure the output temperature.

(v) The actuator

A thyristor (S 16G12S-12) is used as an actuator for the heater and is switched on and off according to the following constraints:

if $u(kT) \leq 0.0$, then $V_i = 0.0$ volt, heater off
if $u(kT) \geq 5.0$, then $V_i = 5.0$ volt, heater on and
if $0.0\ u < (kT) < 5.0$, then $V_i = u(kT)$ volt, heater on

where u(kT) is the output of the neural controller, T denotes the sampling time, k is the sampling number (k = 0, 1, 2, 3 ...), and V_i is the input voltage to the actuator.

B. Mathematical model

The self-learning ability of the neural network offers the advantage of no a priori knowledge regarding the mathematical model of the plant (as we will explain later). However, in applying the self tuning and generalised predictive control algorithms, a prior knowledge of the process's model is essential. In a number of self tuning algorithm, such as the self tuning conroller of Clarke and Gawthrop, a few parameters are to be choosen by the user on trial and error basis. A wrong choice of the user defined parameters may result in a poor performance. In order to avoid this, it is necessary to have a model which adequately describes the behaviour of the process. To obtain the model of the process, two important procedures are required to be carried out; the first is the derivation of the model structure through the frst principles of physics and the second is identifying the model parameters using parameter estimation schemes.

In this section, the two procedures carried out on the water bath are described. The mathematical model of the water bath system can be obtained by balancing the heat equation as follows:

$$\left(h - \frac{q_o - q_c}{R}\right) dt = C dq_o \tag{1}$$

where q_o is the temperature of the water in the bath (°C)
q_c is the circumambient temperature (°C),
C is the thermal capacity (kcal/°C),
h is the power supplied by the heater (kcal),
and R is the thermal resistance (°C/kcal).

Taking the Laplace transform of (1), we have

$$sq_o(s) = \frac{-1}{RC} q_o(s) + \frac{1}{C}\left[H(s) + \frac{1}{R} q_c(s)\right] \tag{2}$$

If we let $a = 1/RC$ and $b = 1/C$, we can regard $q_o(s)$ as the output $Y(s)$, and $H(s) + aq_c/b$ as the input $U(s)$, since the circumambient temperature is known. Therefore, the equation is now

$$sY(s) = -aY(s) + bU(s) \tag{3}$$

Rewriting the above equation in time domain, we obtain

$$\frac{dy(t)}{dt} = -ay(t) + bu(t)$$

$$u(t) = h(t) + \frac{ap_c}{b} \tag{4}$$

The discrete time equivalent model is required in order to control the process using a microcomputer. This can be obtained by using a zero order hold and taking the z transform of the model as follows:

$$y(z) = (1 - z^{-1})\, 3\left[£^{-1}\frac{G(s)}{s}\right] u(z) \tag{5}$$

where

$$G(s) = \frac{b}{s + a}$$

Here, $£^{-1}$ denotes inverse Laplace transform and 3 denotes the z transform. Thus, the pulse transfer function $G(z^{-1})$ becomes

$$G(z^{-1}) = (1 - z^{-1})\left[\frac{b}{a}\left\{\frac{1}{1 - z^{-1}} - \frac{1}{1 - e^{-aTs}z^{-1}}\right\}\right]$$

$$= \frac{z^{-1}(1 - e^{-aTs})}{1 - e^{-aTs}z^{-1}}$$

$$= \frac{bz^{-1}}{1 + az^{-1}} = \frac{z^{-1}B(z^{-1})}{A(z^{-1})} \tag{6}$$

where

$$\varepsilon = e^{-aTs},\ b = \frac{b(1 - e^{-aTs})}{a},\ A(z^{-1}) = 1 + az^{-1}, \text{ and } B(z^{-1}) = b$$

and T_s is the sampling time. The water bath can now be modelled in the following form

$$y(k) + a_1\, y(k - 1) = b_0\, u(k - 1) + \xi(k) \tag{7}$$

where $\xi(k)$ is an uncorrelated sequence of random variables. Estimates of the parameters a_1 and b_0 can be obtained by performing recursive least squares method on the water bath. The control input used is of type maximum length pseudo random binary signals (PRBS).

3. THE NEURO-CONTROL APPROACH

Recently, various neuro-control learning schemes [13] have been proposed where the neural network may be trained to perform as a controller by learning the plant's inverse model or as an emulator by identifying the forward model. Of the numerous neural network paradigms, the backpropagation algorithm is the most widely used as it provides a simple learning and update procedure. In implementing our neuro-control system, we use the concept of backpropagation through time [1], [2], [6]-[8] as shown in Fig. 2. In this scheme, two neural networks are used, where one is trained to learn the plant's forward dynamics and the other trained to learn the plant's inverse dynamics. The neural network which is trained to learn the plant's inverse dynamics is used as a controller. The forward model neural network or the emulator is used to get the equivalent error at the output of the controller by backpropagating the performance error, i.e., the error between the desired output and the actual plant output. A novel feature of this scheme is that both the emulator and the controller may be continuously trained online. The algorithm of the backpropagation neural network has been extensively discussed in many literatures which include [14]-[16]. However, the concept of the backpropagation through time which uses an emulator or a forward model has not been widely implemented. Thus, in this section, we describe the algorithm and the approach we used in implementing the system.

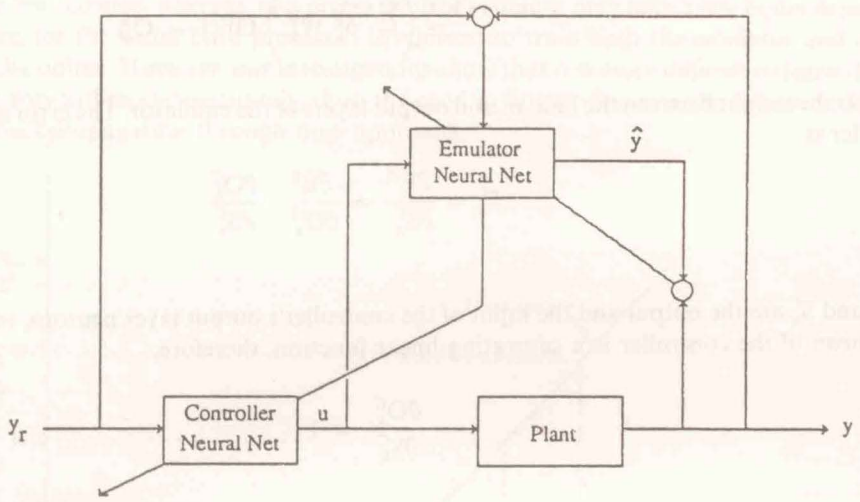

Fig. 2: A neuro-control scheme with backpropagation through time utility

A. The Backpropagation Through Time Algorithm

The forward propagation step of this approach involves only one network each time and thus, is similar to the algorithm as described in [14]-[16]. However, the backward propagation step actually involves two networks which may be considered as one and the equations of the error signals for the controller may be derived in the following way.

We begin by obtaining the error equation at the output of the emulator which is the error between desired output and the actual plant output

$$E^E = \tfrac{1}{2}[y_r - y]^2 \tag{8}$$

where y_r is the desired output and y is the actual plant output. As the output neuron of the emulator is a linear function, the error signal between the hidden and output layers of the emulator is as follows:

$$\delta_k^E = y_r - y \tag{9}$$

where the superscript E denotes the emulator and the subscript k denotes the output layer. The error signal between the hidden and input layers of the emulator is:

$$\delta_j^E = -\frac{\partial E^E}{\partial S_j^E} = \frac{\partial E^E}{\partial O_j^E} \cdot \frac{\partial O_j^E}{\partial S_j^E} \tag{10}$$

where O_j^E and S_j^E are the output and the input of the of the emulator's hidden layer neurons, respectively. By using chain rule, the equation becomes

$$\delta_j^E = \frac{\partial E^E}{\partial S_k^E} \cdot \frac{\partial S_k^E}{\partial O_j^E} \cdot \frac{\partial O_j^E}{\partial S_j^E}$$

$$= \left(\sum_j \delta_k^E W_{kj}^E \right) O_j^E (1 - O_j^E) \tag{11}$$

where W_{kj}^E is the weight between the hidden and output layers of the emulator. The error signal at the output of the conroller is

$$\delta_k^C = \frac{\partial E^E}{\partial S_k^C} = \frac{\partial E^E}{\partial O_k^C} \cdot \frac{\partial O_k^C}{\partial S_k^C} \tag{12}$$

where O_k^C and S_k^C are the output and the input of the controller's output layer neurons, respectively. Since the output neuron of the controller is a saturating linear function, therefore,

$$\frac{\partial O_k^C}{\partial S_k^C} = 1$$

Thus,

$$\delta_k^C = \frac{\partial E^E}{\partial O_k^C} = \frac{\partial E^E}{\partial S_k^C} \cdot \frac{\partial S_k^C}{\partial O_k^C}$$

$$= \sum_j \delta_j^E W_k^{Ej} \tag{13}$$

and for the hidden and input layers of the controller, the error signal is:

$$\delta_j^C = \left(\sum_j \delta_k^C W_j^{Ck} \right) O_j^C (1 - O_j^C) \tag{14}$$

where O_j^C and S_j^C are the output and the input of the of controller's hidden layer neurons, respectively. To improve the performance of the emulator neural network online, the following error equation should be used

$$E = \tfrac{1}{2}[y - \hat{y}]^2 \tag{15}$$

where y and $\hat{y}$ are the outputs of the actual plant and emulator, respectively.

B. Experimental Methodology

A set of corresponding input-output patterns of the water bath process plant must first be obtained in order to train the emulator as well as the controller. We operated the plant without any conventional controller within the working range of the control input which was obtained by inspecting the actuator hardware. Thus, we injected a ramp signal from 0 volt up to 5 volts which are the lower and upper limits of the actuator input, respectively, with an increment of 0.55 volt per sample. Seven sets of corresponding input-output data were then selected as the training patterns as shown in Fig. 3. The emulator neural network was then rained in an of fline way to learn the forward plant model by using the plant output data as the target patterns, and the plant input together with some delayed plant outputs as the input patterns. Fig. 4 shows the performance of the emulator in tracking the actual plant output after offline training.

The controller was also initially rained offline to learn the plant's inverse model by using a similar architecture where the plant input data were used as the target patterns and the present plant output together with some delayed outputs were used as the input patterns. Unlike robotic systems, the water bath is a slow varying process control system where it is rather impossible to rain both the emulator and controller neural networks entirely online from initial random weights using the backpropagation through time approach. This is due to the fact that the backpropagation through time approach is a "goal-directed" learning method where the desired trajectory cycle is repeated over thousands of cycles (see [2], [61]. One trajectory cycle in a robotic system is a matter of a few seconds, whereas, in a prcess control system it may take a few hours depending on applications. Therefore, for the water bath process it is relevant to train both the emulator and controller neural networks initially offline. However, our investigations show that it is more difficult to learn the inverse model rather than the forward model accurately, thus, the need to further fine rain the inverse model neural network though the backpropagation through time approach.

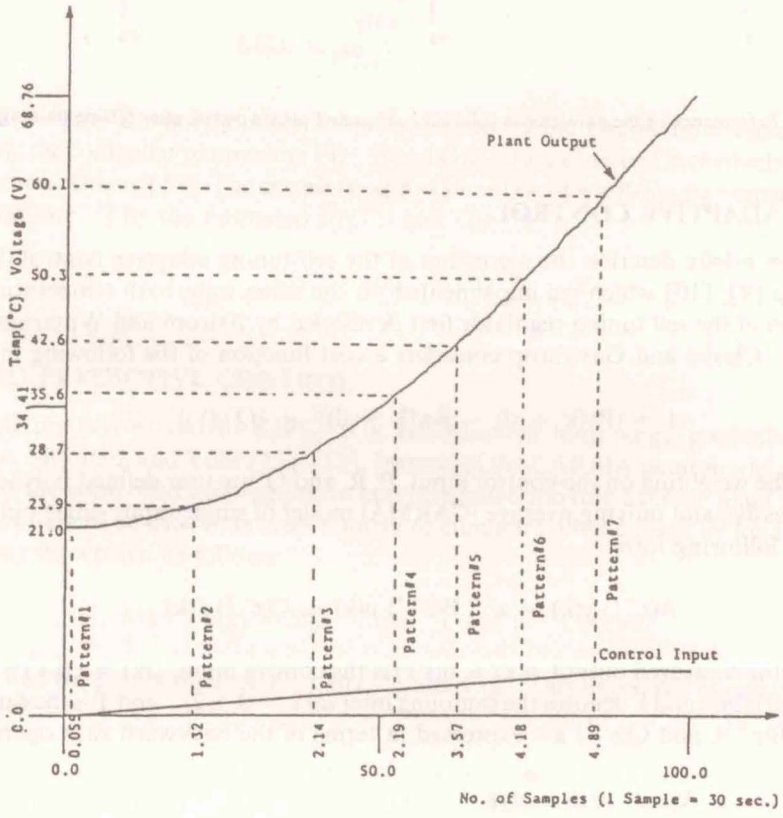

Fig. 3: Training patterns taken from a ramp input of the water bath process

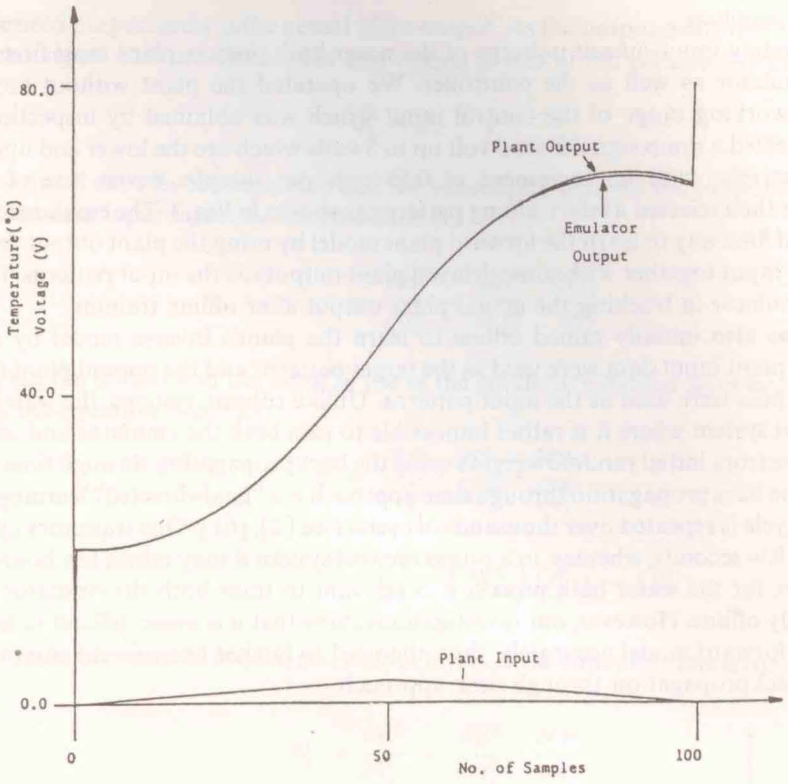

Fig. 4: Peformance of the emulator in following the actual plant's output after offline training

4. SELF-TUNING ADAPTIVE CONTROL

In this section, we briefly describe the algorithm of the self-tuning adaptive control (STC) method of Clarke and Gawthrop [9], [10] which we implemented on the same waterbath temperature process. This method is an extension of the self tuning regulator first developed by Astrom and Witternmark [17]. In the derivation of the STC, Clarke and Gawthrop considers a cost function of the following form

$$J = (Py(k + d) - Rw(k + d))^2 + (Q'u(k))^2 \tag{16}$$

which includes Q′ as the weighting on the control input. P, R, and Q′ are user defined polynomials. Consider a controlled autoregressive and moving average (CARMA) model of single input single output system with a time delay d in the following form

$$A(z^{-1})\, y(k) = z^{-d}\, B(z^{-1})\, u(k) + C(z^{-1})\, \xi(k) \tag{17}$$

where $y(k) = y(kT)$ is the measured output, $u(k) = u(kT)$ is the control input, $\xi(k) = \xi(kT)$ is an uncorrelated sequence of random variables and k denotes the sampling interval $k = 0, 1, 2, \ldots$ and T is the sampling time. The polynomials $A(z^{-1})$, $B(z^{-1})$, and $C(z^{-1})$ are expressed in terms of the backward shift operator z^{-1}

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots + a_{n_a} z^{-n_a} \tag{18}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots + b_{n_b} z^{-n_b} \tag{19}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots\ldots + c_{n_c} z^{-n_c} \tag{20}$$

The roots of $C(z^{-1})$ are assumed to be strictly within the unit circle, $b_0 = 0$, n_a, n_b, n_c are the degrees of $A(z^{-1})$, $B(z^{-1})$, and $C(z^{-1})$, respectively. The control objective of a self-tuning controller is to minimise the variance of the auxilliary output $\phi(k)$ given by

$$\phi(k) = P(z^{-1})\, y(k) + Q(z^{-1})\, u(k - d) - R(z^{-1})\, w(k - d) \tag{21}$$

where w(k) is the set point, and $Q = q_o'Q'/b_o$, and R are ransfer functions in z^{-1} which are important design parameters. Here, only the predictive model based of $\phi\, y(k)$ which is the component of $\phi(k)$ which depends on y(k) is considered and given as follows:

$$\phi\, y(k) = P(z^{-1})\, y(k) \tag{22}$$

where $P(z^{-1}) = P_n(z^{-1})/P_d(z^{-1})$ and $P(1) = 1$. $P(z^{-1})$ provides phase advance such that certain control design procedures using (31) gives desirable closed loop behaviour in terms of y(t). The optimal d step ahead prediction is given by

$$\phi y^*(k + d) = F(z^{-1})\, y_f(k) + G(z^{-1})\, u(k) \tag{23}$$

where

$$G(z^{-1}) = E(z^{-1})\, B(z^{-1}),$$

$$y_f(k) = \frac{y(k)}{Pd(z^{-1})}$$

Here, $F(z^{-1})$ and $E(z^{-1})$ are the polynomials uniquely defined by the Diophantine equation. To make the algorithm self tuning, the controller parameters $F(z^{-1})$ and $G(z^{-1})$ are obtained recursively using the recursive least squares estimation scheme [17]. The certainty equivalence law of a self-tuning controller is obtained by replacing $F(z^{-1})$ and $G(z^{-1})$ by the estimated $\hat{F}(z^{-1})$ and $\hat{G}(z^{-1})$

$$u(k) = \frac{R(z^{-1})\, y(k) - \hat{F}(z^{-1})\, y_f(k)}{\hat{G}(z^{-1}) + \hat{Q}(z^{-1})} \tag{24}$$

5. GENERALISED PREDICTIVE CONTROL

The generalised predictive control (GPC) is a self-adaptive long-range predictive control scheme developed by Clarke, Mohtadi, and Tuffs [11], [12]. Instead of the CARMA plant model as used in the STC approach, GPC uses the controlled autoregressive and integrated moving average plant model (CARIMA) which is more appropriate to be used in process control to eliminate offset caused by load disturbances. The CARIMA model may be written as follows:

$$A(z^{-1})\, y(k) = B(z^{-1})\, u(k^{-1}) + C(z^{-1})\, \xi(k)/\Delta \tag{25}$$

where $\Delta = 1 - z^{-1}$ and the polynomials $A(z^{-1})$, $B(z^{-1})$, and $C(z^{-1})$ are expressed as equations (18), (19) and (20), respectively.

GPC is based on the concept of long range prediction which is based on an assumed model of the process and on an assumed scenario for the future control signals. A sequence of control signals are produced but only the first one is applied to the process at the present sampling instant. This process is repeated at each sampling instant. The updated control action strategy is known as receding horizon strategy and this is used to achieve

the control objective of the predictive law i.e to drive the future output y(k + j) close to the reference point w(k + j) which has been prespecified. The j-step ahead predicted output can be written as follows:

$$\hat{y}(k + j/k) = Gj(z^{-1})\,\Delta u(k + j - 1) + Fj(z^{-1})\,y(k) \tag{26}$$

where $Gj(z^{-1}) = Ej(z^{-1})B(z^{-1})$, E_j and F_j are obtained by solving the Diophantine equation (see [18]. The predicted output can be decomposed into two components, i.e, f(k + j) which are known at the instant k and G Δu(k + j − 1) which are components due to the control input. Therefore,

$$Y = GU + F \tag{27}$$

where

$$Y = [y(k + 1)\ y(k + 2)\ \ldots\ldots\ y(k + N)]^T,$$

$$U = (\Delta u(k)\ \Delta u(k + 1)\ \ldots\ldots\ \Delta u(k + N - 1)]^T,$$

$$F = [f(k + 1)\ f(k + 2)\ \ldots\ldots\ f(k + n)]^T,$$

and G is a lower triangular matrix of dimension N × N and its element are the plant's step response. The cost function of the form below is adopted in the GPC algorithm

$$J(N1, N2) = E\left\{ \sum_{j=N1}^{N2} [y(k + j) - w(k + j)]^2 + \sum_{j=1}^{N2} \lambda(j)\,[(\Delta u(k + j - 1)^2 \right\} \tag{28}$$

where N1 is the minimum costing horizon, N2 is the maximum costing horizon and λ is the control weighting sequence. A distinct feature of the GPC algorithm is that the choices of the design parameters N1, N2, and λ are flexible and does not affect the stability of the controlled system. However, reduction in computation time will result if appropriate values of N1 and N2 are chosen. The projected control increments are obtained by partial differentiating the cost function with respect to u(t) as in the following form

$$\Delta u(k) = (G^T G + \lambda I)^{-1}\, G^T (W - f) \tag{29}$$

Since only the first control signal is required, then

$$u(k) = u(k - 1) + \bar{g}^T (W - f) \tag{30}$$

where W is the vector for the prespecified set points and $\bar{g}^T$ is the first element of $(G^T G + \lambda I)^{-1}\, G^T$.

The ability of GPC to produce stable control of non-minimum phase system is due to the assumption made about future control action in which after an interval of NU > N2, projected control increments are assumed to be equal to zero,.i.e.,

$$\Delta u(k + j - 1) = 0 \qquad j > NU \tag{31}$$

where NU is the control horizon. Hence, the cost function in (28) will have a costing on Δu(k) from j = 1 to j = NU. For a simple system as the water bath process, a good control can be achieved with NU = 1 which reduced the computation considerably.

To make the algorithm self tuning the parameters of the plant are required to be estimated recursively by means of any parameter estimation schemes, where in this case, we used the recursive least squares estimates. The CARIMA plant model can be written as

$$\Delta y(k) = B(z^{-1})\,\Delta u(k-1) + z(1 - A(z^{-1}))\,\Delta y(k) + \xi(k) \tag{32}$$

The data vector and the parameter vector of this model are respectively as follows

$$\varphi^T = [\Delta u(k-1)\,\Delta u(k-2) \ldots\ldots \Delta y(k-1)\,\Delta y(k-2) \ldots\ldots] \tag{33}$$

$$\theta^T = [b_0 b_1 \ldots\ldots - a_1 a_2 \ldots\ldots] \tag{34}$$

The estimates of A and B can be obtained using the least squares estimation principle as discussed.

6. THE CONVENTIONAL FEEDBACK CONTROL APPROACH

In implementing the conventional feedback control approach on the temperature control system, we employ the popular velocity form discrete-time PID controller [19]. Due to its simple structure and easily comprehensible principle, the PID controller has been widely used in many industrial control applications. As such, it is a worthwhile effort to compare its method and performance to that of the neuro and adaptive control approaches. A velocity form of the discrete PID control scheme can be written as follows:

$$\begin{aligned}\Delta u(k) &= K_c[e(k) - e(k-1)] + \frac{T_s}{2T_i}[e(k) + e(k-1)] + \frac{T_d}{T_s}[e(k) - 2e(k-1) + e(k-2)] \\ &= K_p[e(k) - e(k-1) + K_i e(k) + K_d[e(k) - 2e(k-1) + e(k-2)]\end{aligned} \tag{35}$$

where $K_p = K_c - \frac{K_i}{2}$, $K_i = \frac{K_c T_s}{T_i}$, and $K_d = \frac{K_c T_d}{T_s}$

K_p is the proportional gain, K_c is the controller gain or proportional band (PB), K_i is the integral gain, K_d is the derivative gain, T_i is the integral or reset time, and T_d is the derivative or rate time. Letting $e(k) = w(k) - y_f(k)$, we have

$$\begin{aligned}\Delta u(k) = K_p[w(k) - w(k-1) - y_f(k) + y_f(k-1)] + K_i[w(k) - y_f(k) + \\ K_d[w(k) - y_f(k) - 2w(k-1) + 2y_f(k-1) + w(k-2) - y_f(k-2)]\end{aligned} \tag{36}$$

A sudden large change in the set point would result in the proportional and derivative control action producing a large change in the controller output. To suppress this phenomena, the set point $y_r(k)$ is assumed to be constant for a while until the next step change takes place. Hence, we have

$$w(k) = w(k-1) = w(k-2)$$

Equation (36) is now modified to give

$$\Delta u(k) = K_c \frac{T_s}{T_i} w(k) - K_c\left[1 + \frac{T_s}{T_i} + \frac{T_d}{T_i}\right] y_f(k) + K_c\left[1 + 2\frac{T_d}{T_i}\right] y_f(k-1) - K_c \frac{T_d}{T_s} y_f(k-2) \tag{37}$$

As the water bath process is a first order plant, only a two-term controller, namely, the proportional-plus-integral, is used.

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

We carried out experiments on the water bath process as described in Section II using the four control approaches. The controlled variable of this system is the temperature of the water in the 8 litre bath and the manipulated variable is the voltage signal to the actuator. For each case, experiments were conducted over 100 samples with a sampling time of 30 seconds resulting in a 50 minutes experimental duration. Three sets of experiments were conducted for each of the four systems being tested. The approach and experimental results of each system are compared and tabulated and their performance are graded qualitatively.

In the first set of experiments, the tracking performance of the three controllers with respect to set point changes are studied. The set points given in these experiments are as follows:

$$\begin{aligned} y_r(kT) &= 35.0^\circ C \qquad 0 \leq kT \leq 25 \\ y_r(kT) &= 50.0^\circ C \qquad 25 \leq kT \leq 65 \\ y_r(kT) &= 65.0^\circ C \qquad 65 \leq kT \leq 100 \end{aligned}$$

where $y_r(kT)$ is the desired output, T is the sampling time, and k is a set of integers, k = 0, 1, 2, 3, etc.. Figure 5a shows the response of the neuro-controller where the plant output is able to track the desired output very well. The performance of the STC, the GPC, and the PI controller are shown in Figure 5b, 5c, and 5d, respectively. In the case of the STC, the design polynomials P, Q, and R were selected to be 1, 0, 1, respectively. Recursive least squares estimation with no forgetting factor and a covariance matrix of diag {100 I} were used. In the case of the GPC, the design parameters were selected to be as follows: NU = 1, N1 = 1, N2 = 3, and $\lambda = 0$. For the PI controller, the tuning parameters were obtained by using the tuning method of Takahashi [20], where the controller gain, K_c, and reset-time, T_i, were 2.0 and 150 seconds, respectively. It can be observed that the set

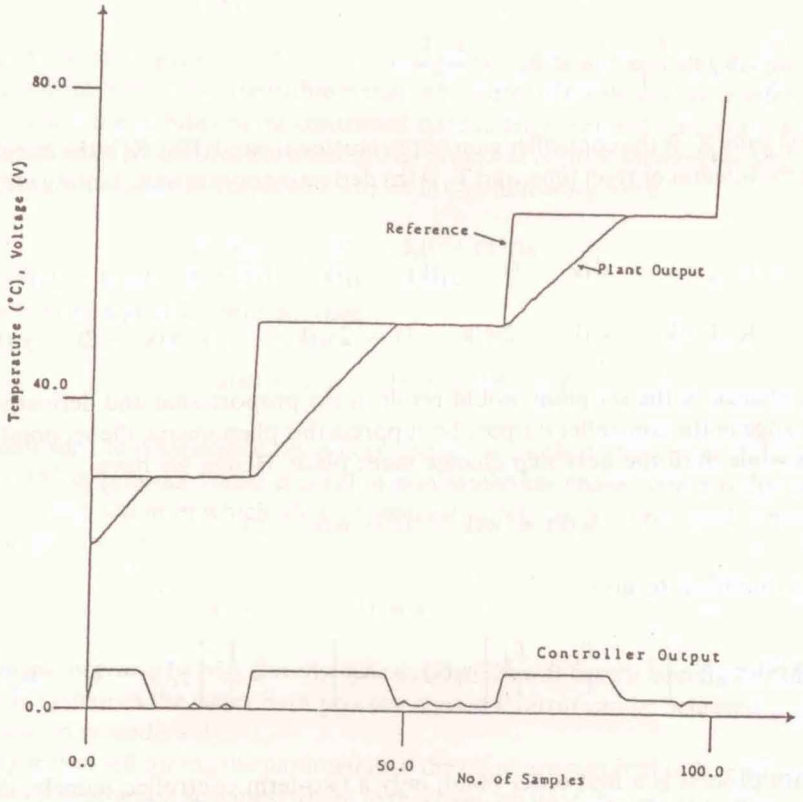

Fig. 5a: Tracking performance of the neuro-controller

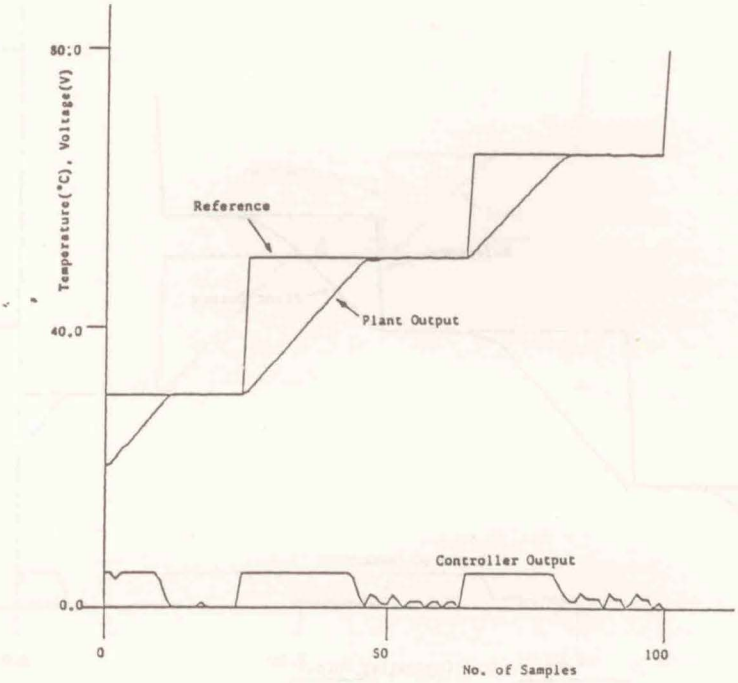

Fig. 5b: Tracking performance of the self-tuning adaptive controller

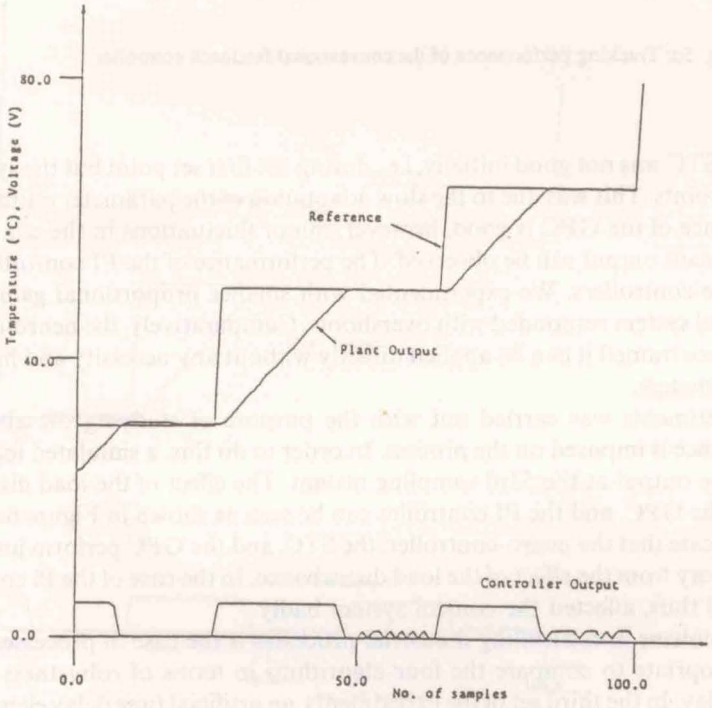

Fig. 5c: Tracking performance of the generalised predictive controller

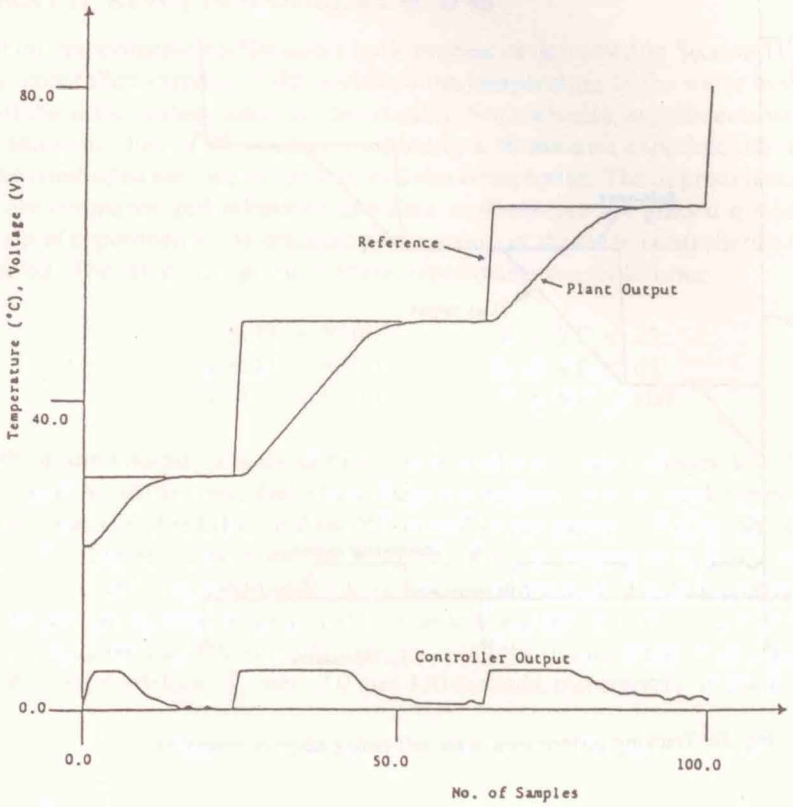

Fig. 5d: Tracking performance of the conventional feedback controller

point tracking ability of the STC was not good initially, i.e., during the first set point but the system tracks well for the second and third set points. This was due to the slow adaptation of the parameter estimation during the initial stages. The performance of the GPC is good, however, minor fluctuations in the control input which result in fluctuations at the plant output can be observed. The performance of the PI controller is rather slow compared to the other three controllers. We experimented with smaller proportional gains to get a faster response but the conventional system responded with overshoots. Comparatively, the neuro-controller shows the best performance and once trained it can be applied directly without any necessity and hassle of selecting any design and tuning parameters.

The second set of experiments was carried out with the purpose of studying the ability of the four controllers if a load disturbance is imposed on the process. In order to do this, a simulated load distubance of value 3.0°C was added to the output at the 53rd sampling instant. The effect of the load disturbance on the neuro-controller, the STC, the GPC, and the PI controller can be seen as shown in Figure 6a, 6b, 6c, and 6d, respectively. The results indicate that the neuro-controller, the STC, and the GPC perform just as good which show a very fast rate of recovery from the effect of the load disturbance. In the case of the Pl controller, the rate of recovery is very slow and thus, affected the control system badly.

One of the common problems in controlling industrial processes is the case of processes with long time delays. Therefore, it is appropriate to compare the four algorithms in terms of robustness in dealing with a plant having a long time delay. In the third set of the experiments, an artificial time delay element of 3 samples is introduced in the control loop. In these experiments only two set points are given: $y_r(kT) = 40.0°C$ for $0 < kT < 50$ and $y_r(kT = 60.0°C$ for $50 < kT < 100$. For this experiment, we re-rained both the emulator and

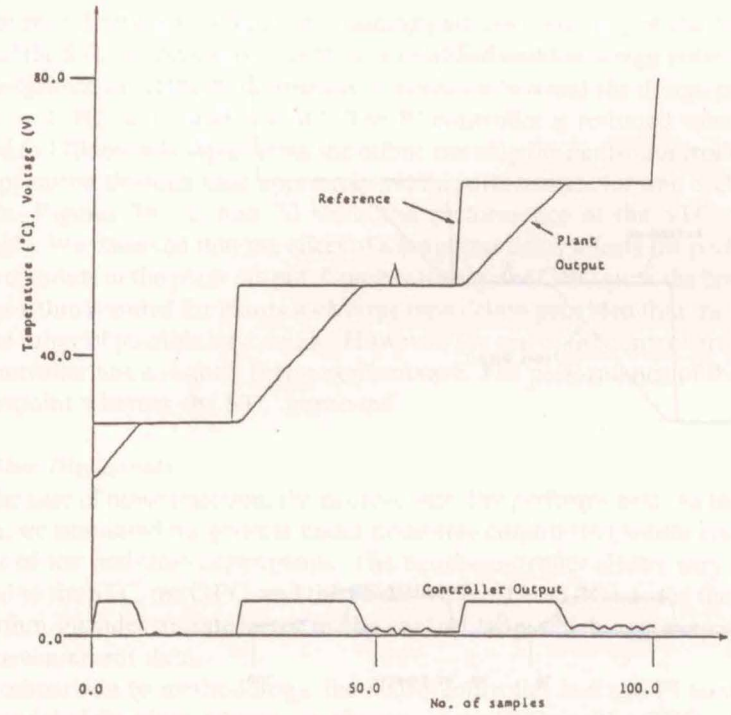

Fig. 6a: Performance of the neuro-controller with a load disturbance at the 53rd sample

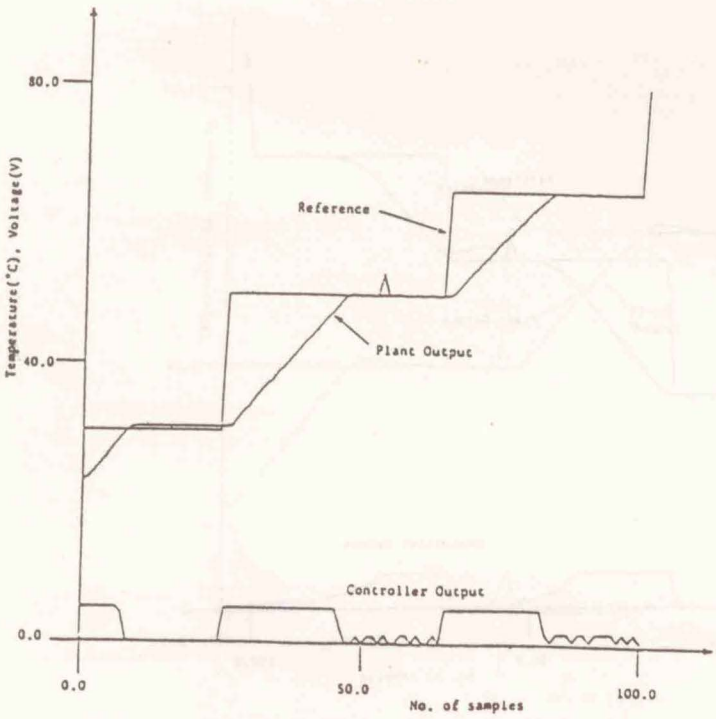

Fig. 6b: Performance of the self-tuning controller with a load disturbance at the 53rd sample

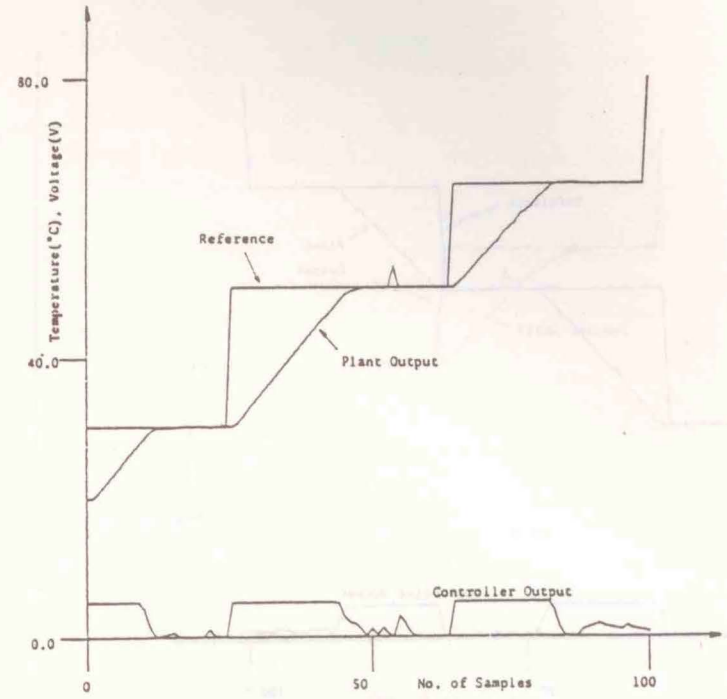

Fig. 6c: Performance of the generalised predictive controller with a load disturbance at the 53rd sample

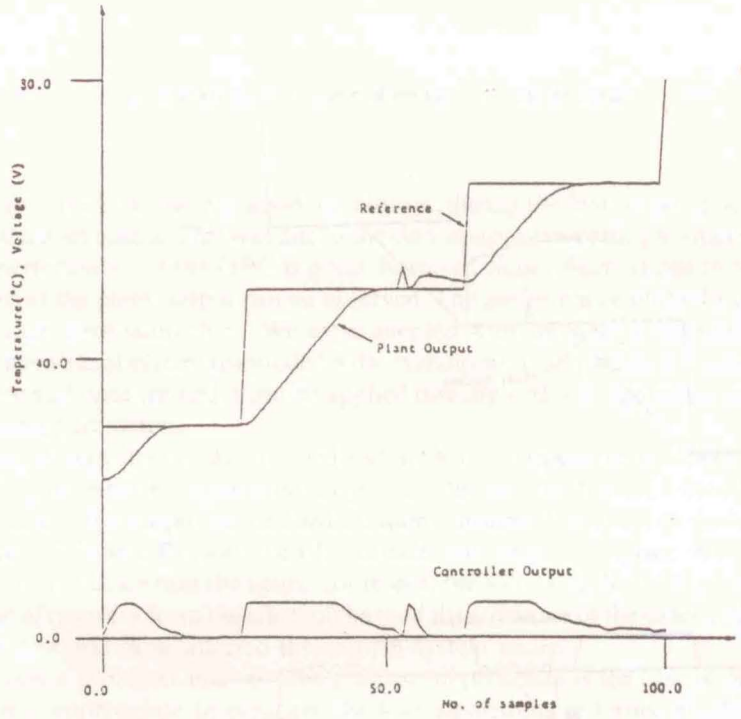

Fig. 6d: Performance of the conventional feedback controller with a load disturbance at the 53rd sample

controller neural networks offline with training patterns consisting of the delayed plant's input-output data. In the case of the STC, its predictive equation is modified and the design polynomial Q is increased to 0.3. For the GPC, the estimation of the B parameters is increased to 4 and the design parameters changed as follows: NU = 2, N1 = 1, N2 = 10, and $\lambda = 0.5$. The PI controller is re-tuned where K_c is increased to 5.0. and T_i is decreased to 120 seconds. Apart from the offline training the neuro-controller is further rained online using the backpropagation through time approach and its performance for one cycle of online training is shown as in Figure 7a. Figures 7b, 7c, and 7d show the performance of the STC, the GPC, and the PI controller, respectively. We observed that the effect of a long time delay affects the performance of all the four controllers giving overshoots in the plant output. Comparatively, the GPC gives the best peformance under this condition as the algorithm is suited for plants with large time delays provided that the number of estimated B parameters covers the range of possible time delays. However, the rest of the controllers did not perform too badly with the neuro-controller has a slightly better performance. The performance of the PI controller deteriorated at the higher setpoint whereas the STC improved.

A. Further Discussions

In the case of noise rejection, the neuro-controller performs best. As the mathematical model of the plant is known, we simulated the process under noise-free conditions (results not shown) and compared the results with that of the real-time experiments. The neuro-controller shows very little fluctuations at its output as compared to the STC, the GPC, and the PI controller. The GPC has the the worst noise rejection capability as its algorithm includes an integrator in the control loop which aggravates the noise effect on the input and output measurement data.

In comparison to methodology, the neuro-controller and the PI controller do not require any mathematical model of the plant, whereas, in the case of the STC and the GPC, a mathematical model is required in prior, which is rather difficult to be derived accurately even for a simple plant as we have shown in Section II.B. In the neuro-control approach, only a set of input-output data of the plant is required initially in order to train

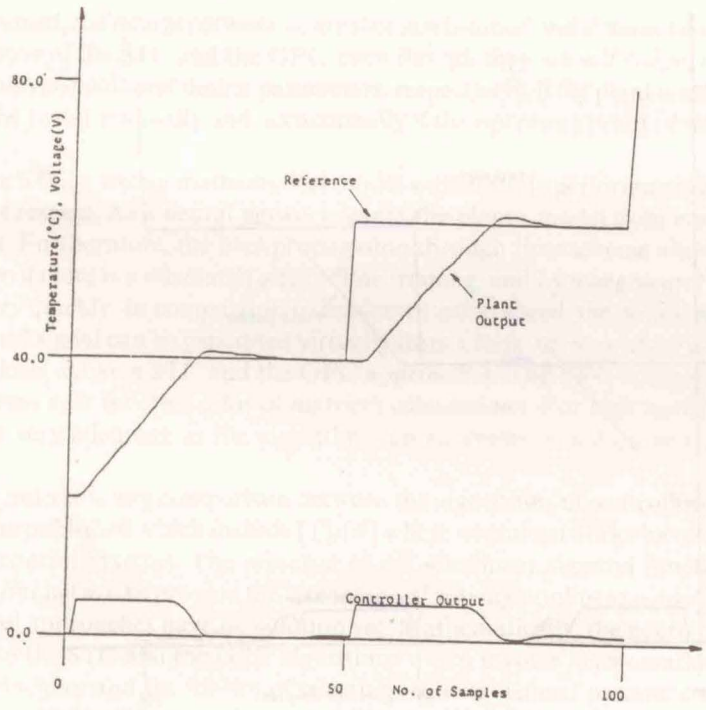

Fig. 7a: Effect of long time-delay on the neuro-control system

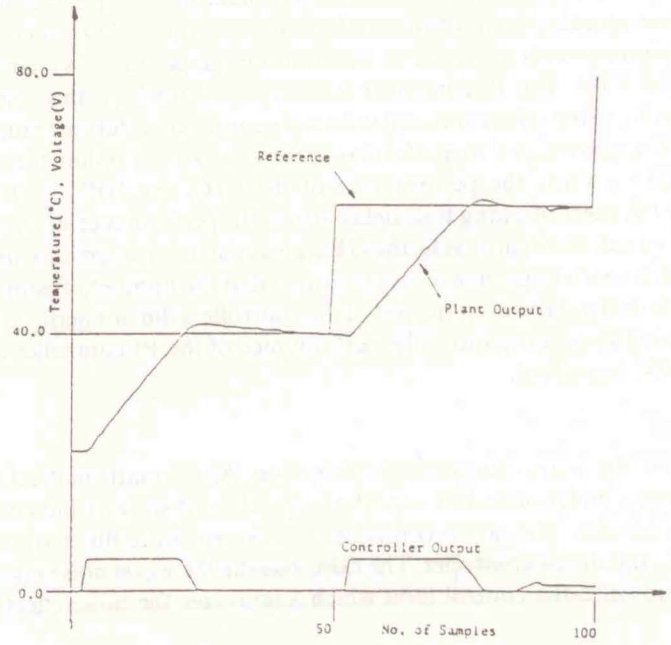

Fig. 7b: Effect of long time-delay on the self-tuning adaptive control system

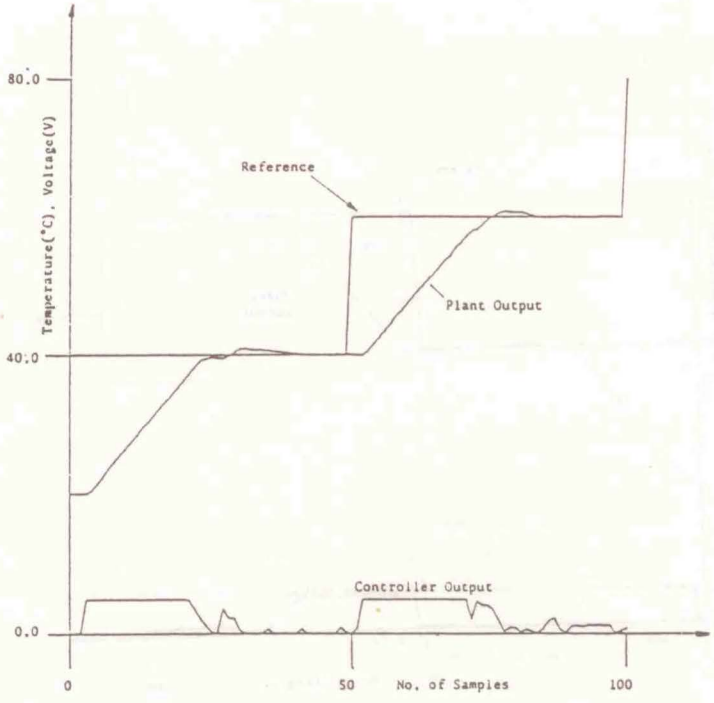

Fig. 7c: Effect of long time-delay on the generalised predictive control system

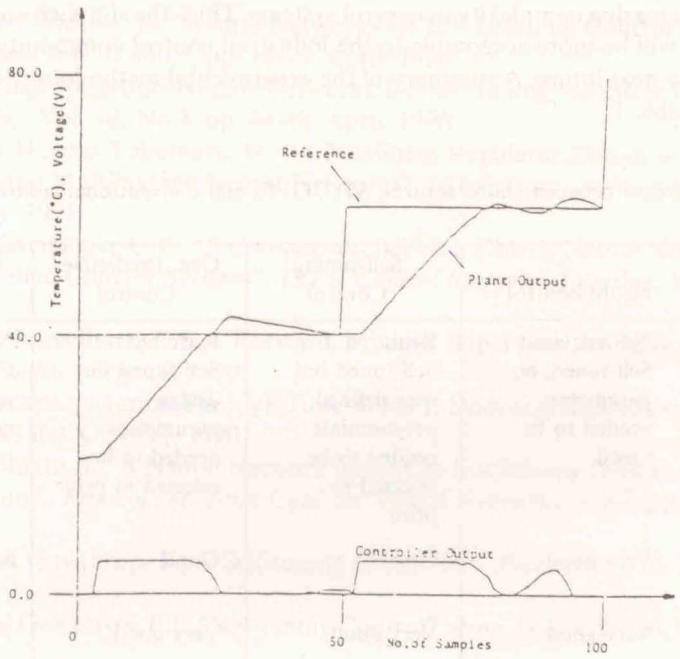

Fig. 7d: Effect of long time-delay on the conventional feedback conuol system

the network. Once trained, the neural network controller is self-tuned and does not consist any requirement for tuning at all. In the case of the STC and the GPC, even though they are self-tuned, there is some difficulty in selecting the design polynomials and design parameters, respectively, if the plant is complex. The conventional Pl controller has to be tuned manually and occassionally if the operating point of the process changes due to disturbances.

In the case of the STC, a wrong mathematical model will affect its performance badly, however, the GPC is more robust in this respect. As a neural network learns the plant's model from input-output data, it is also robust in this respect. Furthermore, the backpropagation through time scheme allows the neural network to be trained online even if there is a mismatch after offline training, and by using neural hardwares, convergence could be achieved very quickly. In comparison to implementation speed, the neural network approach is very fast because the control signal can be generated virtually from a look-up procedure rather than multiplication operations as in the least squares STC and the GPC approaches. The CPC computation time is the slowest among the four systems as it involves a lot of matrices calculations. For high speed applications the neural network approach is very adequate as the algorithm can be implemented on neural hardwares in massive parallel operations.

Although we do not show any comparison between the algorithms in controlling a nonlinear plant, there has been many reports published which include [1]-[4] where neural networks have been successfully applied to highly nonlinear control systems. The presence of the semilinear sigmoid functions in the multilayered backpropagation neural networks provide the advantage of solving nonlinear control problems where to this end traditional control approaches have no solution yet. Mathematically, the neuro control algorithm is very simple as compared to the STC and the GPC algorithms which involve heavy mathematics. The difficulty of understanding the principles and the hassles of selecting the user-defined parameters have been a setback to a variety of adaptive control techniques to be accepted by the industry as compared to the simple conventional feedback control technique which has gained wide popularity. However, such conventional techniques are not

adequate to meet with increasing complexity in control systems. Thus, the simplicity and the reliability of the neuro-control approach will be more acceptable to the industrial control community over these traditional control approaches in the near future. A summary of the experimental methodology and results of the three algorithms is given in Table 1.

Table 1 A comparison between neuro-control, STC, GPC, and conventional control

CRITERIA	Neuro-control	Self-tuning Control	Gen. Predictive Control	Conventional Control
Plant's model Tuning of controller	Not required Self-tuned, no parameters needed to be tuned	Required Self-tuned but user defined polynomials needed to be selected in prior	Required Self-tuned but design parameters needed to be selected in prior	Not required Controller's parameters needed to be tuned
Tracking performance	Very good	Good	Good	Average
Disturbance rejection	Very good	Very good	Very good	Poor
Effect of long time delay	Average (Neural network re-trained)	Average (Predictor eqn and parameters changed)	Good (Parameters changed)	Below average (Controllerr parameters re-tuned)
Noise rejection	Very good	Average	Poor	Average
Convergence speed	Slow, neural network needed to be trained	Fast	Fast	Not applicable
Mismatch Robustness	Good	Poor	Good	Not applicable
Computation Speed	Very fast	Average	Slow	Fast

8. CONCLUSIONS

We have compared the performance of the neural network control approach to that of a self tuning adaptive control approach, a generalised predictive control approach, and a conventional feedback control approach on a real-time control system. Our comparisons show that the neuro-controller performs very well and offers encouraging advantages in many aspects compared to the other three controllers. With cheaper availability of neural hardwares, we have reasons to believe that the neuro-control technique will be the key to intelligent and efficient control in the near future.

9. REFERENCES

[1] Narendra, K.S. and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks". *IEEE Trans. on Neural Networks*, Vol. 1, No.1 pp4-27, March, 1990.

[2] Nguyen, D.H. and Widrow, B., "Neural Networks for Self Learning Control Systems", *IEEE Control Systems Magazine*, Vol. 10, No.3, pp. 18-23, April, 1990.

[3] Chen, F.C., "Backpropagation Neural Networks for Self-tuning Adaptive Control", *IEEE Control Systems Magazine*, Vol. 10, No.3, pp. 44-48, April, 1990.

[4] Iiguni, Y., Sakai, H., and Tokumaru, H., "A Nonlinear Regulator Design in the Presence of System Uncertainties Using Multilayered Neural Networks", *IEEE Trans. on Neural Networks*, Vol.2, No.4, pp. 410-417, July, 1991.

[5] Kraft, L.G. and Campagna, D.P., "A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems", *IEEE Control Systems Magazine*, Vol. 10, No.3, pp. 36-43, April, 1990.

[6] Jordan, M. and Rumelhart, D.E., "Forward Models: Supervised learning with a distal teacher", (submitted to *Cognitive Science*) 1990.

[7] Werbos, P.J., "Backpropagation Through Time: What It Does and How toDo It", *Proc. IEEE*, Vol.78, No. 10, pp. 1550-1560, October, 1990.

[8] Khalid, M. and Omatu, S., "A Neural Network Based Control Scheme With an Adaptive Neural Model Reference Structure", *Proc. of Int. Joint Conf. on Neural Networks*, Vol.3, pp. 2128-2133, November, 1991.

[9] Clarke, D.W., and Gawthrop, P.J., "Self-tuning Controller", *Proc. IEE*, Vol. 122, No. 9, pp. 929-934, Sept. 1975.

[10] Clarke, D.W., and Gawthrop, P.J., "Self-tuning Control", *Proc. IEE*, Vol. 126, No. 6, pp. 633-639, June, 1979.

[11] Clarke, D.W., Mohtadi, C., and Tuffs, P.S., "Generalised Predictive Control-Part I. The Basic Algorithm*", *Automatica*, Vol.23, No.2, pp. 137-148, 1987.

[12] Clarke, D.W., Mohtadi, C., and Tuffs, P.S., "Generalised Predictive Control-Part II. Extensions and Interpretations*", *Automatica*, Vol. 23, No.2, pp. 149-160, 1987.

[13] W.T. Milier III, Sutton, R.S., and Werbos, P.J., editor, *Neural Networks for Control*, MIT Press, 1990.

[14] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol.I*, MIT Ress, pp. 318-362, 1986.

[15] Dayhoff, J., *Neural Network Architectures-An Introduction*, Van Nostrand Reinhoid, 1990.

[16] Pao, Y.H., *An Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, 1989.

[17] Astrom, K.J., and Wittenmark, B., *Adaptive Control*, Addison Wesley, 1989.

[18] Yusof. R. and Omatu, S., "Application of Generalised Predictive Control to a Temperature Control Process", *Proc. IECON* '91, Vol. 3, pp. 1935-1940, Oct., 1991.

[19] Ogata, K., Discrete-Time Control Systems, Prentice Hall, 1987.

[20] Takahashi, Y. and Chan C.S, 1971, Parametereinstellungbei Linearen DDC Algorithmen. *Regelungstechnik u. Prozess-Datenverarbeitung*, Vol. 19,237