

# Design of a minimum variance multiple input–multiple output neuro self-tuning proportional-integral-derivative controller for non-linear dynamic systems

L Z Guo<sup>1</sup>, Q M Zhu<sup>2</sup>, and K Warwick<sup>3\*</sup>

<sup>1</sup>Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK

<sup>2</sup>Faculty of Computing, Engineering, and Mathematical Sciences, University of the West of England, Bristol, UK

<sup>3</sup>Department of Cybernetics, The University of Reading, Reading, UK

*The manuscript was received on 2 December 2003 and was accepted after revision for publication on 14 July 2006.*

DOI: 10.1243/095965118I12503

**Abstract:** In this study a minimum variance neuro self-tuning proportional-integral-derivative (PID) controller is designed for complex multiple input–multiple output (MIMO) dynamic systems. An approximation model is constructed, which consists of two functional blocks. The first block uses a linear submodel to approximate dominant system dynamics around a selected number of operating points. The second block is used as an error agent, implemented by a neural network, to accommodate the inaccuracy possibly introduced by the linear submodel approximation, various complexities/uncertainties, and complicated coupling effects frequently exhibited in non-linear MIMO dynamic systems. With the proposed model structure, controller design of an MIMO plant with  $n$  inputs and  $n$  outputs could be, for example, decomposed into  $n$  independent single input–single output (SISO) subsystem designs. The effectiveness of the controller design procedure is initially verified through simulations of industrial examples.

**Keywords:** MIMO dynamic systems, neuro PID controller, self-tuning control

## 1 INTRODUCTION

The ideas behind predictive and adaptive control have been widely used in process industry applications such as steel rolling and petrochemical. A common requirement is that, for all kinds of control, an accurate model of the plant is required. However, this is not easily satisfied in many practical situations (e.g. see reference [1]). Self-tuning control, as a type of adaptive predictive control, in fact has a similar requirement of an accurate model if good closed-loop control is to be achieved. In order to improve model accuracy and hence the feasibility of controller design, one important development has been the use of artificial neural networks to model and control non-linear dynamical systems more accurately [2–6].

The utilization of neural networks for the control of non-linear systems has, to a great extent, been

due to such a network's capacity to learn from and adapt to coarse environment conditions in mapping complicated input-to-output relationships. A neural network, composed of neurons, or simple processing elements, which are connected together with some functional structures, originally came about from an attempt at modelling the information processing capabilities of nervous systems [7], and hence they can be considered as mathematical representations of biological neural networks. For some time now neural networks, based on a variety of learning and training concepts, have been successfully used to model non-linear phenomena. One reason for this is their natural ability to deal with multivariable relationships. Therefore the introduction of neural networks to a multivariable control system design should be seen as a natural extension of classical approaches and should significantly relieve difficulties encountered when resolving certain mathematical solutions, particularly in decomposing interactive/coupling effects.

Early attempts to apply neural networks to non-linear discrete-time adaptive control systems can be

\* Corresponding author: Department of Cybernetics, University of Reading, Whiteknights, PO Box 225, Reading RG6 6AY, UK. email: kw@cyber.rdg.ac.uk

traced to the work of Narendra and Parthasarathy [3], Chen [8], and Chen and Khalil [9, 10]. These investigations mainly concentrated on the construction of an inverse model of the controlled plant and the design of feedback linearization-like controllers, which required a well-trained neural network as an inverse model. Neural networks have also been used as predictors over a specified horizon for predictive control by Proll and Karim [11] and Liu Kadirkamanthan, and Billings [12]. With this scheme, predictions of system performance obtained from the network are passed to a numerical optimization routine to minimize a specified performance criterion by calculating a suitable control signal. However, the computational complexity of this approach still presents a large problem.

To improve the performance of adaptive control systems and to reduce the computational burden, a different approach to the design of a neural network enhanced generalized minimum variance self-tuning controller for non-linear discrete-time systems was proposed [13]. In this approach unknown non-linear plants were firstly approximated with a linear submodel, the parameters of which were identified by a recursive least squares algorithm; then a neural network was employed to predict the remaining error from the linear submodel. In fact, most design approaches developed from linear control systems can be directly applied to non-linear systems by using the same technique. This type of method has, however, been shown to be particularly useful in realizing adaptive control of non-linear systems [14].

Based on this type of model structure and design principle, Zhu and Warwick [15] proposed a neural network enhanced generalized minimum variance self-tuning proportional-integral-derivative (PID) control algorithm for single input–single output (SISO) non-linear dynamic systems. The necessity of this investigation for PID control was well justified in the studies. Since the introduction of automatic control, PID controllers have been the most widely used tools for industrial automation, as they exhibit many favourable characteristics such as user-friendly manipulation, low-cost maintenance, robust performance to plant dynamic variations, both theoretically and experimentally oriented tunings, and widely available commercial products. The parameters of a PID controller are, for the most part, traditionally tuned by trial and error or the so-called Ziegler–Nichols rules. Now, however, the parameters can be adjusted according to a PID self-tuning design mechanism. With the proposed methodology [15], it becomes a natural procedure to deal with complex dynamics, to link optimal control performance to

PID control implementation, and to implant the control algorithm into existing commercial PID controller products.

It should be noticed that this approach does not introduce further complexity in the method employed to represent non-linear dynamics. Conversely, it actually reduces the overall design effort for non-linear control systems by separating the system description into two submodels. The reason behind using a linear model + a neural network has in fact been well justified in previous studies [13, 15]. On top of this, by using only a neural network model to represent a non-linear dynamic plant, the design will almost surely not be as transparent as the proposed structure in which linear design approaches can be directly applied. Further to this, the structure of a linear model + a neural network has been found to be generally more computationally efficient in comparison with a large-scale neural network in approximating complex non-linear plants [15].

In the study reported here, a design procedure for SISO neuro self-tuning PID controllers [15] is generalized to the multiple input–multiple output (MIMO) case. There are two distinctive advantages of such an approach.

1. By using neural networks as predictors to estimate the errors between a linear submodel and the real plant, the integrated model can be naturally decoupled so that an MIMO plant with  $n$  outputs can be viewed as  $n$  independent SISO subplants. Further, the parameters of the linear submodel can be estimated independently and the  $n$  independent PID controllers for each SISO plant can be designed separately.
2. The neural networks are built with the inherent ability to deal with MIMO dynamics so that it is a natural option to map MIMO relationships without any apparent extra load.

The sections in this paper are organized as follows. A minimum variance MIMO PID controller design is introduced in section 2. A procedure to estimate the linear submodel parameters using an RLS (recursive least squares) algorithm and to predict the remaining error using a BP (back propagation) neural computational algorithm is described in section 3, including an indication of how implementation can be achieved in an MIMO self-tuning framework. Two realistic industrial cases are then simulated to demonstrate the potential efficiency of the proposed design procedure in section 4. Finally, conclusions are drawn in section 5.

## 2 PID CONTROLLER DESIGN

The overall control system structure is shown in Fig. 1, which includes both the controller design and model parameter estimation. The PID controller employed is constructed in two steps. First of all a minimum variance controller is derived based on a combination of a linear submodel of the plant under control and a BP neural network model that approximates the residual error. Then the equivalent parameters of the PID controller are found by means of a direct comparison with the derived minimum variance controller.

### 2.1 MIMO predictive model

In previous studies [13, 15, 16] a concise model structure has been proposed to formulate characteristics of a dynamic plant in terms of a finite set of parameters. Accordingly, mathematical manipulation can make it relatively easy to analyse the behaviour of the plant and a respective controller can be designed to obtain the desirable behaviour. It has to be said that most study results have been obtained based on the control of SISO dynamic plants, largely because of their relative simplicity. However, many existing practical plants are inherently MIMO in nature, and exhibit strong, significant coupling effects.

The analysis and design of MIMO dynamic systems is almost always by no means a simple extension of the SISO case and/or just an additional requirement of extra computational effort. As a result, the decoupling of the MIMO dynamic plant has been a

key topic in the analysis and design of MIMO control systems. In this study, it is shown that a MIMO dynamic plant can be naturally decoupled by using a neural network model. Not only that, but the corresponding controllers can be designed separately as well.

In general, a discrete-time causal MIMO dynamic plant can be described in the following form

$$Y(t + 1) = F(Y(t), U(t), t) \tag{1}$$

where  $Y(t) \in R^n$  and  $U(t) \in R^n$  are the output and input signals of the plant respectively at discrete instants  $t = 1, 2, \dots, n$ , and  $F: R^n \times R^n \times R^+ \rightarrow R^n$  is a complicated smooth non-linear vector function. Note that a linear vector function can be treated as a special simple case of the non-linear vector function.

As was the case with the SISO plant [13], expression (1) may be further expanded into a generalized parameter model

$$\mathbf{A}(z^{-1})Y(t + 1) = \mathbf{B}(z^{-1})U(t) + \mathbf{F}_0(\dots) \tag{2}$$

in which  $\mathbf{A}(z^{-1})$  and  $\mathbf{B}(z^{-1})$  are diagonal polynomial matrices based on the backward shift operator  $z^{-1}$  such that

$$\begin{aligned} \mathbf{A}(z^{-1}) &= \text{diag}(1 + a_1^i z^{-1} + \dots + a_l^i z^{-l}) \\ \mathbf{B}(z^{-1}) &= \text{diag}(b_0 + b_1^i z^{-1} + \dots + b_m^i z^{-m}) \end{aligned} \quad i = 1, \dots, n \tag{3}$$

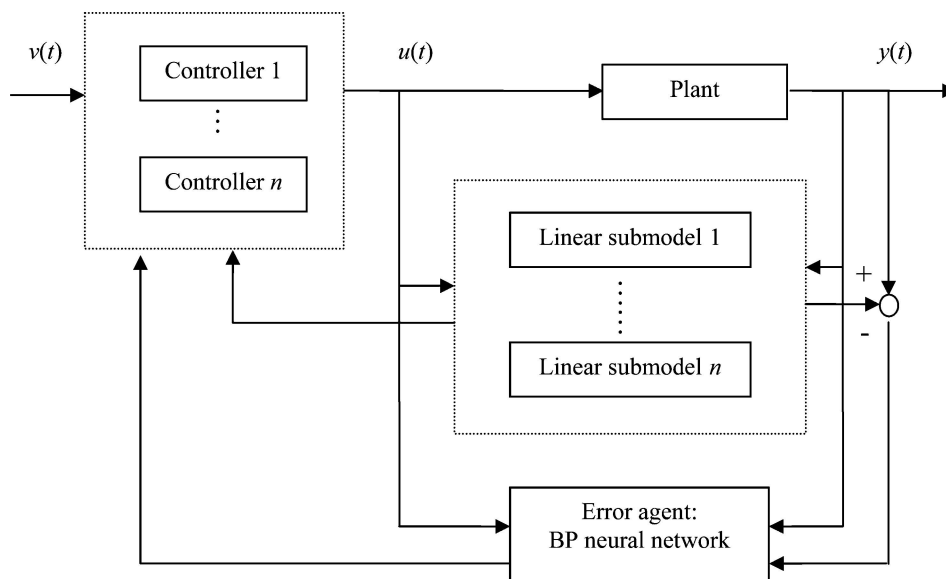


Fig. 1 Overall control system structure

and  $F_0(\cdot, \cdot)$  is a non-linear  $n$ -vector function and is defined here as an error agent.

*Remark 1*

The linear model can be considered as a linear expansion around operating points, which are originated by assuming that no coupling relationships exist. The coupling effects and other non-linear relationships are then accommodated in the error agent  $F_0(\cdot, \cdot)$ .

*Remark 2*

$\mathbf{A}(z^{-1})$  and  $\mathbf{B}(z^{-1})$  are both  $n \times n$  matrices. This implies the same dimension of input and output, so that the interactions are decoupled into one input against one output.

*Remark 3*

Note that the orders of all subplant models can be different, without any loss in generality. In this study a unique order is chosen for all subplant models for the purpose of computational convenience only.

*Remark 4*

In practice, random disturbances and measured noise will be completely independent of plant dynamics, which cannot be predicted precisely even though a neural network is employed. However, an advantage of using a neural network predictor is that the measurable variables related to the disturbance can be considered within the neural network structure. It is good, however, to bear in mind here stochastic control problems within this framework that will be investigated in future studies. In this study it is assumed that there are neither deterministic nor stochastic disturbances in the model, purely for the sake of demonstrating the main idea about how a neural network can play a vitally important role in a self-tuning controller design procedure.

## 2.2 Design of the minimum variance controller

To derive a generalized MIMO minimum variance controller, the performance index can be described as

$$J = \|\mathbf{S}(z^{-1})y(t+1) - \mathbf{R}(z^{-1})v(t) - \mathbf{Q}(z^{-1})u(t) - \mathbf{H}(z^{-1})F_0(\cdot, \cdot)\| \quad (4)$$

where  $v(t)$  is a bounded reference input vector,  $\|\cdot\|$  is any kind of norm defined in an appropriate space, and  $\mathbf{S}(z^{-1})$ ,  $\mathbf{Q}(z^{-1})$ ,  $\mathbf{R}(z^{-1})$ , and  $\mathbf{H}(z^{-1})$  are diagonal

weighting polynomial matrices of  $z^{-1}$ , which are defined as

$$\begin{aligned} \mathbf{S}(z^{-1}) &= \text{diag}(1 + s_1^i z^{-1} + \dots + s_{ns}^i z^{-ns}) \\ \mathbf{Q}(z^{-1}) &= \text{diag}(q_0 + q_1^i z^{-1} + \dots + q_{nq}^i z^{-nq}) \end{aligned} \quad i = 1, \dots, n \quad (5)$$

Now an auxiliary output vector is additionally defined as

$$\phi(t+1) = \mathbf{S}(z^{-1})y(t+1) \quad (6)$$

In a similar fashion to the SISO case, the optimal predictor  $\phi^*(t+1/t)$  for the auxiliary output vector  $\phi(t+1)$  can be determined by

$$\begin{aligned} \phi^*(t+1/t) &= \mathbf{G}(z^{-1})y(t) + \mathbf{C}(z^{-1})\mathbf{B}(z^{-1})u(t) \\ &\quad + \mathbf{C}(z^{-1})F_0(\cdot, \cdot) \end{aligned} \quad (7)$$

in which  $\mathbf{C}(z^{-1})$  and  $\mathbf{G}(z^{-1})$  are diagonal polynomial matrices and are defined as

$$\begin{aligned} \mathbf{C}(z^{-1}) &= \text{diag}(1 + c_1^i z^{-1} + \dots + c_{nc}^i z^{-nc}) \\ \mathbf{G}(z^{-1}) &= \text{diag}(g_0 + g_1^i z^{-1} + \dots + g_{ng}^i z^{-ng}) \end{aligned} \quad i = 1, \dots, n, \quad ng = n-1 \quad (8)$$

which satisfy

$$\mathbf{S}(z^{-1}) = \mathbf{C}(z^{-1})\mathbf{A}(z^{-1}) + z^{-1}\mathbf{G}(z^{-1}) \quad (9)$$

To minimize the performance index (4), the optimal predictor is then found to be

$$\phi^*(t+1/t) = \mathbf{R}(z^{-1})v(t) + \mathbf{Q}(z^{-1})u(t) + \mathbf{H}(z^{-1})F_0(\cdot, \cdot) \quad (10)$$

According to equations (7) and (10), the controller output  $u(t)$  satisfying the minimization requirement can be obtained from

$$\begin{aligned} &[-\mathbf{Q}(z^{-1}) + \mathbf{C}(z^{-1})\mathbf{B}(z^{-1})]u(t) \\ &= \mathbf{R}(z^{-1})v(t) + \mathbf{H}(z^{-1})F_0(\cdot, \cdot) - \mathbf{G}(z^{-1})y(t) \\ &\quad - \mathbf{C}(z^{-1})F_0(\cdot, \cdot) \end{aligned} \quad (11)$$

*Remark 5*

Note that all matrices above are diagonal so that the controllers can actually be calculated independently, which is one of the advantages of using this type of model.

By combining equations (11) and (2), the closed-loop system characteristic equation can be expressed as

$$\begin{aligned} & [\mathbf{B}(z^{-1})\mathbf{S}(z^{-1}) - \mathbf{A}(z^{-1})\mathbf{Q}(z^{-1})]y(t+1) \\ &= \mathbf{B}(\mathbf{R}(z^{-1})v(t) + \mathbf{H}(z^{-1})\mathbf{F}_0(\cdot, \cdot) - \mathbf{C}(z^{-1})\mathbf{F}_0(\cdot, \cdot)) \\ &+ \mathbf{F}(\cdot, \cdot) \end{aligned} \quad (12)$$

where  $\mathbf{F}(\cdot, \cdot) = [-\mathbf{Q}(z^{-1}) + \mathbf{C}(z^{-1})]\mathbf{B}(z^{-1})$ . To produce satisfactory dynamics, a stable polynomial matrix  $\mathbf{T}(z^{-1})$  is chosen to express the desirable closed-loop characteristic equations as

$$\mathbf{T}(z^{-1}) = \text{diag}(t_0^i + t_1^i z^{-1} + \dots + t_m^i z^{-n_i}), \quad i = 1, \dots, n \quad (13)$$

By inspection of the left-hand side of equation (12), it can be seen that the following relationship must be satisfied in order to achieve the desired performance of the closed-loop system, i.e. in order that the closed-loop characteristic equations are given by equation (13)

$$\begin{aligned} & \mathbf{B}(z^{-1})\mathbf{S}(z^{-1}) - \mathbf{A}(z^{-1})\mathbf{Q}(z^{-1}) = \mathbf{T}(z^{-1}) \\ & ns = n - 1, \quad nq = n_y - 1, \quad nt \leq 2n - 1 \end{aligned} \quad (14)$$

As long as  $\mathbf{T}(z^{-1})$  is specified either by a designer or a (sensible) customer,  $\mathbf{S}(z^{-1})$  and  $\mathbf{Q}(z^{-1})$  can be determined from equation (14) in which, at the time of calculation,  $\mathbf{A}(z^{-1})$  and  $\mathbf{B}(z^{-1})$  are all known matrices, or at least parameter estimates have been found and can be employed.

To cancel the static offset,  $\mathbf{R}(z^{-1})$  can be selected, most simply, as

$$\mathbf{R}(z^{-1}) = \text{diag} \left[ \frac{\mathbf{T}^1(1)}{\mathbf{B}^1(1)} \right], \quad i = 1, \dots, n \quad (15)$$

Finally, to eliminate the effect, in the steady state, of the non-linear part,  $\mathbf{H}(z^{-1})$  can be chosen as

$$\mathbf{H}(z^{-1}) = \text{diag} \left[ \frac{\mathbf{Q}^i(1)}{\mathbf{B}^i(1)} \right], \quad i = 1, \dots, n \quad (16)$$

All the design parameters in the polynomial matrices can now be obtained effectively, according to the formulations described in equations (14), (15), (16), and (9).

## 2.3 Implementation of the PID controller

Generally a standard digital PID controller for the SISO plant [17] can be represented by

$$\text{PID}(z^{-1}) = K_p \left[ 1 + \frac{T_s}{T_i(1-z^{-1})} + \frac{T_d(1-z^{-1})}{T_s} \right] \quad (17)$$

where  $T_s$  is the sampling time, and  $K_p$ ,  $T_i$ , and  $T_d$  are the proportional gain, integral time, and derivative time respectively. For a particular pre-selected sampling time the only remaining design stage for the PID controller is to determine the best values of  $K_p$ ,  $T_i$ , and  $T_d$ .

It should be noted that the PID controller design procedure for SISO plants generally may not be directly applicable to MIMO cases because of the effect of coupling between different variables. However, within this proposed control framework, coupling effects are initially disregarded by means of using a non-linear function, which is subsequently approximated by a neural network. An MIMO plant with  $n$  outputs can, in this way, be viewed as  $n$  independent SISO plants so that the PID parameters for each SISO plant can be determined separately.

From the above discussion, to implement the generalized minimum variance controller for a MIMO plant in terms of PID control, a transformation can be made by rearranging equation (11) as

$$\begin{aligned} u_i(t) &= \frac{\mathbf{R}^i v^i(t) + \mathbf{H}^i(z^{-1})\mathbf{F}_0^i(\cdot, \cdot) - \mathbf{G}^i(z^{-1})y^i(t) - \mathbf{C}^i(z^{-1})\mathbf{F}_0^i(\cdot, \cdot)}{-\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1})} \\ &= \frac{\mathbf{G}^i(z^{-1})[v^i(t) - y^i(t)] + (\mathbf{R}^i - \mathbf{G}^i(z^{-1}))v^i(t) + [\mathbf{H}^i - \mathbf{C}^i(z^{-1})]\mathbf{F}_0^i(\cdot, \cdot)}{-\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1})} \\ &= \frac{\mathbf{G}^i(z^{-1})[v^i(t) - y^i(t)]}{\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1})} \\ &+ \frac{[\mathbf{R}^i - \mathbf{G}^i(z^{-1})]v^i(t)}{\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1})} \\ &+ \frac{[\mathbf{H}^i - \mathbf{C}^i(z^{-1})]\mathbf{F}_0^i(\cdot, \cdot)}{\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1})} \\ &= \frac{\mathbf{G}^i(z^{-1})}{1-z^{-1}} \times \frac{1-z^{-1}}{\mathbf{D}^i(z^{-1})} \times e(t) + \frac{[\mathbf{R}^i - \mathbf{G}^i(z^{-1})]v^i(t)}{\mathbf{D}^i(z^{-1})} \\ &+ \frac{(\mathbf{H}^i - \mathbf{C}^i(z^{-1}))\mathbf{F}_0^i(\cdot, \cdot)}{\mathbf{D}^i(z^{-1})}, \quad i = 1, \dots, n \end{aligned} \quad (18)$$

where  $X^i$  means the  $i$ th component of vector  $\mathbf{X}$  or the  $i$ th diagonal element of matrix  $\mathbf{X}$ , and

$$\begin{aligned} \mathbf{D}^i(z^{-1}) &= -\mathbf{Q}^i(z^{-1}) + \mathbf{C}^i(z^{-1})\mathbf{B}^i(z^{-1}) \\ e^i(t) &= v^i(t) - y^i(t), \quad i = 1, \dots, n \end{aligned} \quad (19)$$

Then a PID-based control algorithm can be obtained with a set of standard PID controllers along with a corresponding low-pass filter to the reference input and also a low-pass filter to the non-linear part

$$\begin{aligned} \text{PID}^i(z^{-1}) &= \frac{\mathbf{G}^i(z^{-1})}{1 - z^{-1}} \\ \text{Filter 1}^i(z^{-1}) &= \frac{\mathbf{R}^i - \mathbf{G}^i(z^{-1})}{\mathbf{D}^i(z^{-1})} \\ \text{Filter 2}^i(z^{-1}) &= \frac{\mathbf{H}^i - \mathbf{C}^i(z^{-1})}{\mathbf{D}^i(z^{-1})}, \quad i = 1, \dots, n \end{aligned} \quad (20)$$

### 3 ESTIMATION OF MODEL PARAMETERS

As explained previously, a wide range of non-linear plants can be approximated with a linear submodel, and the inaccuracy of the linear submodel of a particular plant can be compensated by an error agent which will be predicted by a neural network. In order to implement an auto-tuning PID controller, it is necessary to identify the linear submodel on line because in practice the parameters of the linear model are most likely unknown *a priori* and/or may be time varying. Actually the identification of the plant model of equation (2) consists of two parts: firstly a linear model parameter estimation and secondly an error prediction.

#### 3.1 Parameter estimation of the linear submodel using a recursive least squares algorithm

For a linear model parameter estimation a standard recursive least squares (RLS) algorithm can be applied directly [18]. Note that an MIMO RLS algorithm can be used to generate the estimation for all parameters simultaneously or an SISO RLS algorithm can be used to obtain the parameters separately for each SISO subplant because the coupling effect has been removed. Here an MIMO RLS is introduced to estimate the parameters of the linear submodels

$$\mathbf{Y}(t+1) = \mathbf{\Phi}(t)^T \boldsymbol{\theta}(t) + \mathbf{F}_0(t+1) \quad (21)$$

where  $\mathbf{Y}(t+1) = (y^1(t+1), \dots, y^n(t+1))^T$  is the output vector of the plant at the time  $t+1$ ,  $\mathbf{\Phi}(t) \in R^{nm \times n}$

is the regressor matrix, and  $\boldsymbol{\theta}$  is the parameter vector as follows

$$\begin{aligned} \mathbf{\Phi}(t) &= \text{diag}[\phi_1(t), \phi_2(t), \dots, \phi_n(t)] \\ \boldsymbol{\theta}(t) &= [\theta_1(t)^T, \theta_2(t)^T, \dots, \theta_n(t)^T]^T \\ \phi_i(t) &= [y^i(t), \dots, y^i(t-n_y); u^i(t), \dots, u^i(t-n_u)]^T \\ \boldsymbol{\theta}_i(t) &= (a_1^i, \dots, a_{n_y}^i; b_0^i, \dots, b_{n_u}^i)^T \\ & \quad i = 1, \dots, n; n_y + n_u = m \end{aligned} \quad (22)$$

The update procedure is then

$$\begin{aligned} \boldsymbol{\theta}(t) &= \boldsymbol{\theta}(t-1) + \mathbf{P}(t)\mathbf{\Phi}(t)\boldsymbol{\varepsilon}(t) \\ \boldsymbol{\varepsilon}(t) &= \mathbf{Y}(t) - \mathbf{\Phi}(t)^T \boldsymbol{\theta}(t-1) \\ \mathbf{P}(t) &= \frac{1}{\lambda} \{ \mathbf{P}(t-1) - \mathbf{P}(t-1)\mathbf{\Phi}(t) \\ & \quad \times [\lambda \mathbf{I} + \mathbf{\Phi}(t)^T \mathbf{P}(t-1)\mathbf{\Phi}(t)]^{-1} \mathbf{\Phi}(t)^T \mathbf{P}(t-1) \} \end{aligned} \quad (23)$$

where  $0 < \lambda \leq 1$  is a forgetting factor intended to affect the convergence of the parameter estimates and to cope with a slowly time-varying plant.

#### 3.2 Weight update for the neural network using a back propagation algorithm

Up till now it has been shown that a general MIMO non-linear plant can be approximated by an equivalent decoupled MIMO linear model, which inevitably introduces a modelling error because of the potential difference between the behaviours of the linear model and the real plant. As mentioned before, this error may result from complicated factors such as non-linearities, the presence of unmodelled dynamics, uncertainties, disturbances, and all kinds of random noise in the controlled plant. Among these factors non-linearity and unmodelled dynamics may be considered as deterministic factors depending on the input and output history of the controlled plant, which are predictable, whereas in general uncertainties, disturbances and noise are treated as random variables as they are not predictable in advance.

In this study it has been assumed that only predictable factors are present, such that by using the information available at the current time a neural network can be used to play a role to predict the error. As mentioned earlier, investigations into stochastic control requirements will be reported in future studies.

To detect the error agent  $\mathbf{F}_0(\cdot, \cdot)$ , a three-layer back propagation (BP) neural network was structured

with hyperbolic tangent activation functions. It is worth noting here that a neural network is inherently suitable for dealing with MIMO function relationships. With regard to the computational efficiency in real-time control, a fast BP network is proposed in this study consisting of three layers: the input layer with  $n_i$  neurons, the hidden layer with  $n_h$  sigmoidal neurons, and the output layer with  $n$  linear neurons. The objective of the network training is to adjust all variable weights to minimize the error  $E_k$  between the real plant output and the linear submodel output, where  $E_k$  is defined as

$$E_k = \frac{1}{2}[\mathbf{F}_0(\cdot, \cdot)^T \mathbf{F}_0(\cdot, \cdot)] \quad (24)$$

With the parameter estimates previously obtained, equation (21) can be alternatively described as

$$\mathbf{Y}(t+1) = \mathbf{\Phi}(t)^T \hat{\boldsymbol{\theta}}(t) + \mathbf{F}_0(t+1) \quad (25)$$

In each sampling period, the parameters of the linear submodel are estimated by an RLS algorithm. Therefore the output training signal of the neural network for the error agent  $\mathbf{F}_0(\cdot, \cdot)$  can be calculated from

$$\mathbf{F}_0(\cdot, \cdot) = \mathbf{Y}(t+1) - \mathbf{\Phi}(t)^T \hat{\boldsymbol{\theta}}(t) \quad (26)$$

The input training signal vector of the neural network can be chosen from the controller output (i.e. plant input) and plant output regressor signals, for example

$$\mathbf{I}(t+1) = [\mathbf{Y}(t)^T, \dots, \mathbf{Y}(t-n_y)^T; \mathbf{U}(t)^T, \dots, \mathbf{U}(t-n_u)^T]^T \quad (27)$$

The training procedure for a neural network can be described as consisting of three phases, the forward propagation phase, a back propagation phase, and the weight updating phase.

In the forward propagation phase the output of the hidden layer is firstly calculated as

$$\mathbf{O}_j(t) = \frac{1}{1 + \exp(-\text{Net}_j)}, \quad j = 1, \dots, M$$

$$\text{Net} = \mathbf{W}^{(1)} \times \mathbf{I}(t) \quad (28)$$

The output of the overall network is, in the case of a linear activation function

$$\hat{\mathbf{F}}_0(\cdot, \cdot) = \mathbf{W}^{(2)} \times \mathbf{O} \quad (29)$$

where  $\mathbf{W}^{(1)}$  and  $\mathbf{W}^{(2)}$  are weights between the input and hidden layers and between the hidden layer and output layer respectively, and with reference to equations (28),  $\mathbf{O} = [O_1, O_2, \dots, O_M]^T$  is the hidden layer output vector.

In the back propagation phase, so-called 'delta vector' operations are computed as

$$\begin{aligned} \mathbf{A}^{(2)}(t) &= \mathbf{E}(t) \\ \mathbf{A}^{(1)}(t) &= \mathbf{O}(t)(\mathbf{I} - \mathbf{O}(t)) \times \mathbf{W}^{(2)T} \mathbf{A}^{(2)}(t) \end{aligned} \quad (30)$$

The weight updating equations are then

$$\begin{aligned} \mathbf{W}^{(1)}(t+1) &= \mathbf{W}^{(1)}(t) + \alpha[\mathbf{W}^{(1)}(t) - \mathbf{W}^{(1)}(t-1)] \\ &\quad + \eta \mathbf{A}^{(1)}(t)^T \mathbf{I}(t) \\ \mathbf{W}^{(2)}(t+1) &= \mathbf{W}^{(2)}(t) + \alpha[\mathbf{W}^{(2)}(t) - \mathbf{W}^{(2)}(t-1)] \\ &\quad + \eta \mathbf{A}^{(2)}(t)^T \mathbf{O}(t) \end{aligned} \quad (31)$$

where  $0 \leq \alpha < 1$  and  $0 < \eta < 1$  are the momentum constant and learning rate in the computation of the weights respectively.

#### Remark 6

The learning rate is a key factor in the convergence of the algorithm. The smaller the learning rate, the smaller will be changes to the weights in the network from one iteration to the next and the smoother will be the trajectory in weight space. By contrast, an increase in the learning rate will speed up convergence of the weight computations. However, it should be noticed that too large a learning rate value may well cause instability problems in the algorithm. The momentum constant is introduced in order to avoid convergence to a local stationary point.

#### Remark 7

Normally  $n_y$  and  $n_u$  are, strictly speaking, unknown. In most cases it is practically sensible to select small numbers for them, 2 or 3 say.

#### Remark 8

The selection of input training variables is vitally important to the performance of the system. In general, these input training variables can be chosen as historical data generated from input and output signals of the controlled plant. Further, measurable disturbance variables can also be used as input elements of the network to give some insight into the forthcoming disturbances at the current sampling instant. In other words, the input variables of the network practically depend on the underlying physical background, which must necessarily reflect the real characteristics of the process.

*Remark 9*

The reason behind representing  $F_0(\cdot, \cdot)$  by a model/neural network comes from the requirement to predict its value at the next time instant  $t + 1$ , such that a controller can be designed to operate successfully at the same time  $t + 1$ . Parameters of the linear submodel and the neural network are, to this end, recursively updated.

*Remark 10*

The neural network in this study is trained on-line. The basic idea is that, during each sampling period, the neural network is trained through the history data of the system and is then used to predict the likely output error for the next sampling instant. In practice, it has been found that the initial iteration steps for this training are less than 10, and that this is sufficient to provide a good prediction result. The convergence of the overall algorithm comes from the following explanation: note that the neural network used in the algorithm is not only for predicting the modelling error but also for decoupling the overall MIMO system. By using the type of neural networks described here, an MIMO system can be effectively decoupled into several independent SISO subsystems. Therefore the discussion of the convergence of the algorithm about SISO systems can be applied here to each SISO subsystem. A proof of the convergence for SISO systems can be found in reference [13].

### 3.3 Neuro self-tuning PID control algorithm

From the above discussion, an auto-tuning control procedure is employed to update the equivalent model using RLS and to predict the error by using the neural network output. The PID controller parameters can then be tuned recursively. The computations at each time instant  $t$  can be outlined as follows.

1. Sample the plant output  $Y(t)$  and form the regression vector  $\Phi(t-1)$  by means of the plant input  $U(t-1)$  and output  $Y(t-1)$  historical data sequences. Form the neural network input vector  $I(t)$  by means of  $U(t-1)$  and  $Y(t-1)$  data sequences.
2. Predict the next error  $F_0(\cdot, \cdot)$  using the trained neural network from the last time instant,  $t-1$ .
3. Estimate the parameters of  $\mathbf{A}(z^{-1})$  and  $\mathbf{B}(z^{-1})$  within the linear submodel.
4. Calculate the controller parameters from formulations (9), (13) to (16), (19), and (20).
5. Generate the controller output  $U(t)$  from equation (18).

6. Form the next step neural network training input signal vector  $I(t+1)$ .
7. Train the neural network for a pre-selected number of epochs using formulations (28) to (31).
8. Wait for the sampling clock pulse. Then go to step 1.

## 4 CASE STUDIES

Selecting suitable problem domains in which to prove the performance and ultimately success of a novel controller type is a challenge in itself. In particular, when it comes to non-linear systems the variety and range of possibilities are seemingly endless. There are, it must be admitted, control scenarios that would present the controller with a considerable challenge (e.g. see reference [19]) because they exhibit multivariable couplings which are in themselves non-linear in nature. What has been attempted here is to select two different non-linear application domains that are felt to be reasonably representative of a large number of control system problems and yet are areas in which, it transpires, the controller described in this paper operates effectively. To demonstrate the efficiency of the derived neural network enhanced minimum variance PID controller, two industrial non-linear examples have been selected.

### Example 1: control of an induction motor

Voltage equations of induction motors in a synchronously rotating frame can be expressed according to the general theory of electric machines [20] as

$$\begin{bmatrix} v_{qs} \\ v_{ds} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_s + sL_s & \omega_e L_s & sL_m & \omega_e L_m \\ -\omega_e L_s & R_s + sL_s & -\omega_e L_m & sL_m \\ sL_m & \omega_{sl} L_m & R_r + sL_r & \omega_{sl} L_r \\ -\omega_{sl} L_m & sL_m & -\omega_{sl} L_r & R_r + sL_r \end{bmatrix} \times \begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{qr} \\ i_{dr} \end{bmatrix} \quad (32)$$

where respectively  $v_{qs}$  and  $v_{ds}$  are the  $q$  axis and  $d$  axis stator voltages,  $i_{qs}$  and  $i_{ds}$  are the  $q$  axis and  $d$  axis stator currents,  $i_{qr}$  and  $i_{dr}$  are the  $q$  axis and  $d$  axis rotor currents,  $R_s$  and  $R_r$  are the stator and rotor resistances,  $L_s$  and  $L_r$  are the stator and rotor inductances,  $L_m$  is the magnetizing inductance,  $P$  is the number of poles,  $\omega_e$  is the electrical angular speed, and  $\omega_{sl}$  is the slip angular speed. The mechanical

equations of the induction motor can be of the following form

$$T_e = \frac{3P}{4} L_m (i_{qs} i_{dr} - i_{ds} i_{qr}) = J \frac{d\omega_r}{dt} + B\omega_r + T_L \quad (33)$$

where  $T_e$  is the generated torque,  $\omega_r$  is the rotor angular speed,  $T_L$  is the load torque,  $J$  is the inertia constant, and  $B$  is the damping ratio.

Equations (32) and (33) can be rearranged to yield the following state description

$$\begin{bmatrix} \dot{i}_{ds} \\ \dot{i}_{qs} \\ \dot{\lambda}_{dr} \\ \dot{\lambda}_{qr} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{\sigma L_s} - \frac{R_r(1-\sigma)}{\sigma L_r} & \omega_e & \frac{L_m R_r}{\sigma L_s L_r^2} & \frac{P\omega_r L_m}{2\sigma L_s L_r} \\ -\omega_e & -\frac{R_s}{\sigma L_s} - \frac{R_r(1-\sigma)}{\sigma L_r} & -\frac{P\omega_r L_m}{2\sigma L_s L_r} & \frac{L_m R_r}{\sigma L_s L_r^2} \\ \frac{L_m R_r}{L_r} & 0 & -\frac{R_r}{L_r} & \omega_e - \frac{P}{2}\omega_r \\ 0 & \frac{L_m R_r}{L_r} & -\left(\omega_e - \frac{P}{2}\omega_r\right) & -\frac{R_r}{L_r} \end{bmatrix} \begin{bmatrix} i_{ds} \\ i_{qs} \\ \lambda_{dr} \\ \lambda_{qr} \end{bmatrix} + \frac{1}{\sigma L_s} \begin{bmatrix} v_{ds} \\ v_{qs} \\ 0 \\ 0 \end{bmatrix} \quad (34)$$

and

$$\begin{aligned} \dot{\omega}_r &= -\frac{B}{J}\omega_r + \frac{1}{J}(T_e - T_L) = -\frac{B}{J}\omega_r \\ &+ \frac{3P}{4} \frac{L_m}{JL_r} (i_{qs}\lambda_{dr} - i_{ds}\lambda_{qr}) - \frac{1}{J}T_L \end{aligned} \quad (35)$$

where  $\sigma = 1 - L_m^2/(L_s L_r)$ ,  $\lambda_{qr} = L_m i_{qs} + L_r i_{qr}$  is the  $q$  axis rotor flux, and  $\lambda_{dr} = L_m i_{ds} + L_r i_{dr}$  is the  $d$  axis rotor flux.

Let  $\lambda = (\lambda_{qr}^2 + \lambda_{dr}^2)^{1/2}$  represent the rotor flux amplitude. Specify  $\lambda$  and  $\omega_r$  as the plant outputs and  $v_{qs}$  and  $v_{ds}$  as the control inputs to drive the plant outputs. This is a typical case of an MIMO non-linear dynamic system. It should be mentioned that in practice the rotor flux of an induction motor is not easily measured, so it is common practice for a rotor flux observer to be inserted in the feedback loop. In this study, without losing generality in the demonstration of the controller design procedure, it is assumed that the rotor flux is measurable for the purpose of simplicity of the simulations.

In the simulation, the complete model was selected as

$$\begin{aligned} (1 + a_1^1 z^{-1})\lambda(k+1) &= (b_0^1 + b_1^1 z^{-1})v_{ds}(k) + f_\lambda \\ (1 + a_1^2 z^{-1})\omega_r(k+1) &= (b_0^2 + b_1^2 z^{-1})v_{qs}(k) + f_\omega \end{aligned} \quad (36)$$

The forgetting factor for recursively estimating the associated parameters of the linear submodel was selected as 0.98. The error agents  $f_\lambda$  and  $f_\omega$  were obtained by a BP neural network. The neural network for predicting the error was set up with  $n_i=15$ ,  $n_h=1$ , and  $n_o=2$  for the input layer, hidden layer, and output layer respectively. The momentum constant was set to be 0.2, with the learning rate at 0.05. The network was trained twice in each sampling period. The input signal vector to the BP neural network was

chosen as

$$\begin{aligned} \mathbf{x} &= [-1, \lambda(t), \lambda(t-1), \lambda_1(t), \lambda_1(t-1), v_{ds}(t) \\ &v_{ds}(t-1), \lambda(t) - \lambda_1(t), \omega_r(t), \omega_r(t-1) \\ &\omega_{r1}(t), \omega_{r1}(t-1), v_{qs}(t), v_{qs}(t-1), \omega_r(t) - \omega_{r1}(t)] \end{aligned} \quad (37)$$

where  $\lambda_1$  and  $\omega_{r1}$  are the linear model outputs. The desired closed-loop dynamic polynomial matrix  $\mathbf{T}(z^{-1})$  was assigned in order to provide closed-loop stability, as

$$\mathbf{T}(z^{-1}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} z^{-1} \quad (38)$$

The simulation parameters were assigned as

$$\begin{aligned} R_s &= 3.20 \, \Omega, R_r = 2.349 \, \Omega, L_s = 0.1294 \, \text{H}, \\ L_r &= 0.1329 \, \text{H}, L_m = 0.1267 \, \text{H}, J = 0.009 \, \text{kg m}^2 \end{aligned}$$

Simulation results for this example are shown in Figs 2 to 5. Figure 2 indicates the responses of the rotor flux amplitude and rotor angular speed to show that the flux amplitude is actually affected shortly after the rotor angular speed is changed. This is because the system is decoupled using just one neural network, whose input is composed of all available information (see equation (37)) and then when the reference signal of the rotor angular speed changes, it causes a disturbance for the flux amplitude control

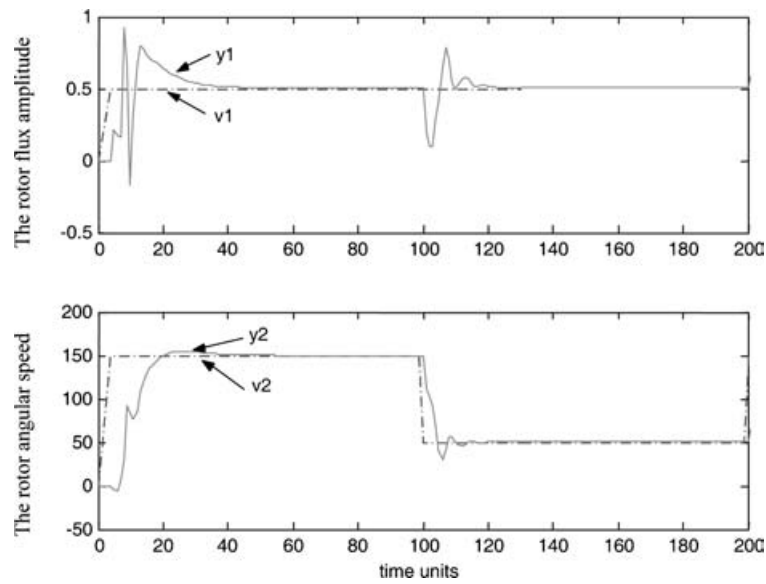


Fig. 2 Example 1: closed-loop response of the control system

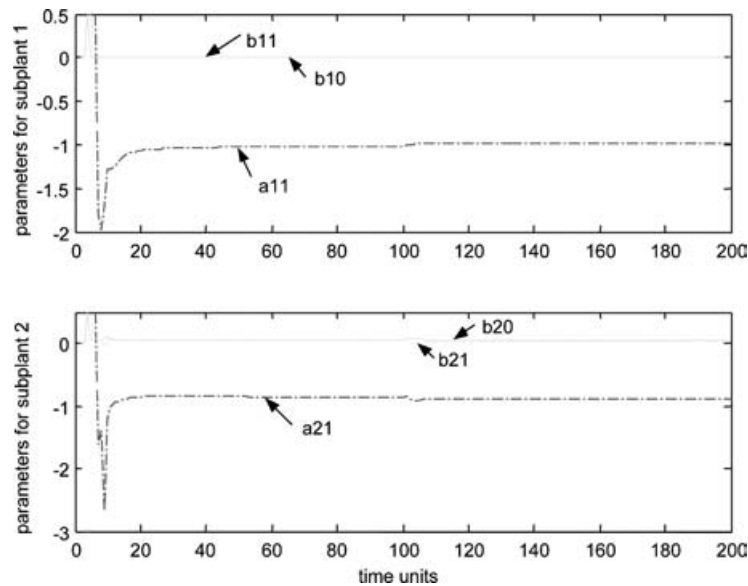
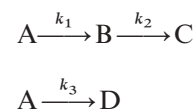


Fig. 3 Example 1: estimated parameters of the linear models by means of RLS

loop. Figure 3 shows the estimated parameters of the linear model using a recursive least squares algorithm, which finally converge to constants. Figure 4 indicates the neural network output versus the actual error between linear model outputs and real outputs. Figure 5 shows the control inputs  $v_{ds}$  and  $v_{qs}$ , where it can be observed that the rotor flux amplitude and rotor speed are decoupled by the proposed control strategy. Overall the simulation results show that the entire closed-loop system is stable with good transient and static performance.

#### Example 2: control of a non-isothermal CSTR

The second case study considered here is the control of a non-isothermal continuously stirred tank reactor (CSTR). The plant kinetics are governed by the van de Vusse reactions



(39)

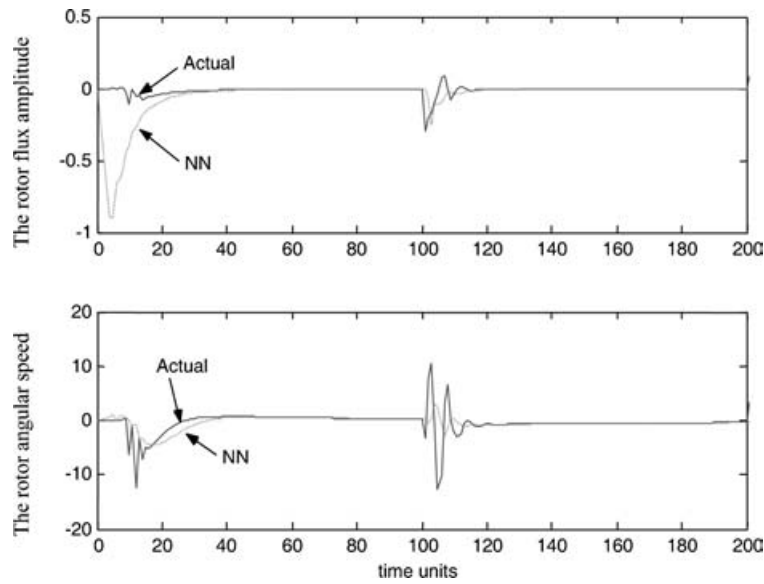


Fig. 4 Example 1: neural network outputs and actual errors

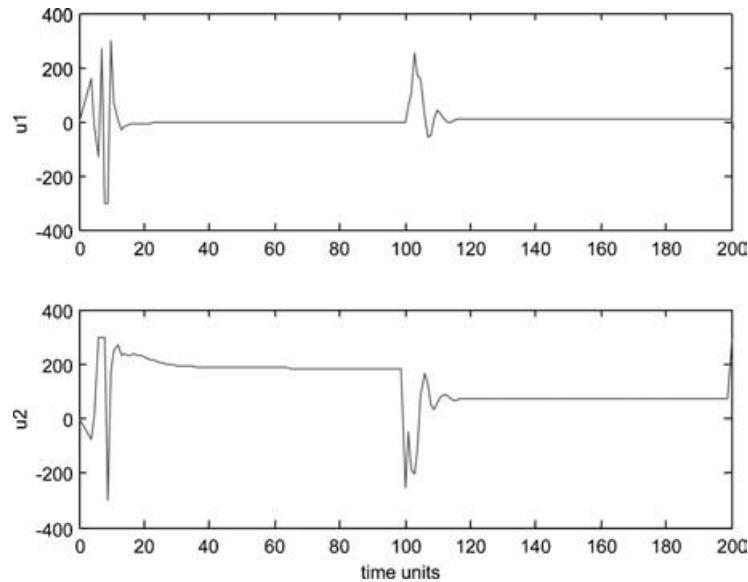


Fig. 5 Example 1: control inputs  $u_1$ ,  $u_2$  are  $v_{ds}$  and  $v_{qs}$

Operation of the system represents the production of cyclopentenol (B) from cyclopentadiene (A). The reaction is catalysed acid, and the side products are dicyclopentadiene (D) and cyclopentanediol (C) [21]. The reaction takes place in a jacketed cooled, perfectly mixed CSTR, where the coolant is introduced by an external heat exchanger. The state-space equations describing the reactor's operation can be derived as [22]

$$\dot{x}_1 = \frac{u_1}{V}(c_{A,0} - x_1) - k_1(T)x_1 - k_3(T)x_1^2$$

$$\dot{x}_2 = -\frac{u_1}{V}x_2 + k_1(T)x_1 - k_2(T)x_2$$

$$\begin{aligned} \dot{x}_3 = & -\frac{1}{\rho C_p} \\ & \times [k_1(T)x_1 \Delta H_1 + k_2(T)x_2 \Delta H_2 + k_3(T)x_1^2 \Delta H_3] \\ & + \frac{u_1}{V}(T_0 - x_3) + \frac{k_w A_w}{\rho C_p}(x_4 - x_3) \\ \dot{x}_4 = & \frac{1}{m_w \rho C_p} [u_2 - k_w A_w(x_4 - x_3)] \end{aligned} \tag{40}$$

where  $x_1, x_2, x_3, x_4$  represent the concentrations of species A and B and the temperatures of the reactor and coolant respectively. The control variables  $u_1$  and

$u_2$  are the reactor flowrate and the external heat exchanger duty. The control objective is to control the outlet concentration of B, i.e.  $y_1 = x_2$ , and the reactor temperature  $y_2 = x_3$  by  $u_1$  and  $u_2$ . The notation and relevant parameters in equations (40) are listed in Table 1.

The reaction rate coefficients  $k_i$ ,  $i = 1, 2, 3$ , are assumed to depend on temperature in an Arrhenius fashion

$$k_i = k_{i,0} \exp(E_i/x_3) \quad (41)$$

**Table 1** Physical parameters for the cyclopentenol reactor

Parameter	Feature
$V = 10$ L	Reactor volume
$T_0 = 403.15$ K	Inlet temperature
$C_p = 30.1$ kJ/kg K	Average heat capacity
$\rho = 0.9342$ kg/L	Average density
$k_w = 4032$ kJ/h m <sup>2</sup> K	Coolant conductivity
$C_{p,w} = 2.0$ kJ/kg K	Coolant heat capacity
$m_w = 5.0$ kg	Coolant mass
$A_w = 0.215$ m <sup>2</sup>	Heat exchange area
$k_{1,0} = 1.287 \times 10^{12}$ 1/h	Arrhenius constant
$k_{2,0} = 1.287 \times 10^{12}$ 1/h	Arrhenius constant
$k_{3,0} = 9.043 \times 10^9$ 1/h	Arrhenius constant
$E_1 = -9758.3$ K	Normalized activation energy
$E_2 = -9758.3$ K	Normalized activation energy
$E_3 = -8560.0$ K	Normalized activation energy
$\Delta H_1 = 4.3$ kJ/mol	Heat of reaction
$\Delta H_2 = -11.0$ kJ/mol	Heat of reaction
$\Delta H_3 = 41.85$ kJ/mol	Heat of reaction
$c_{A,0} = 1.0$ mol/L	Inlet concentration

In the simulation, the decoupled model was selected as

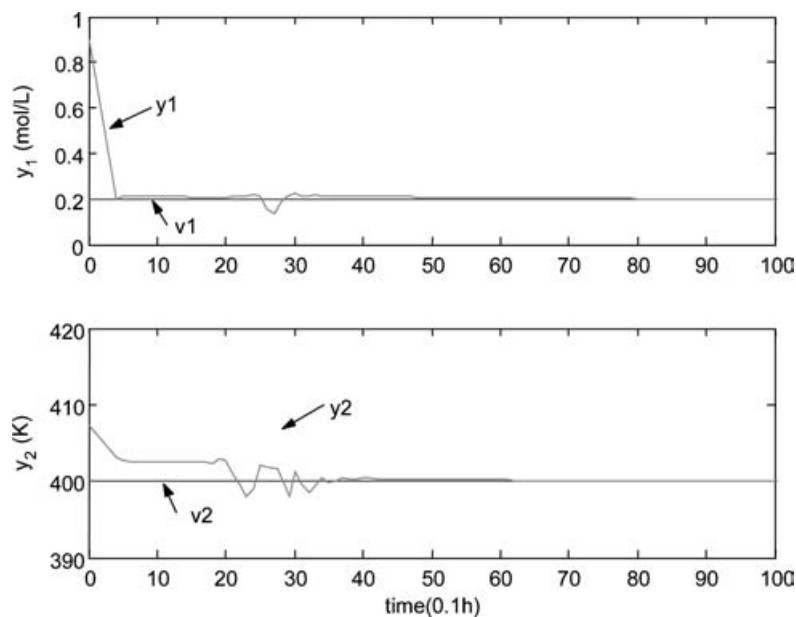
$$\begin{aligned} (1 + a_1^1 z^{-1})y_1(k+1) &= (b_0^1 + b_1^1 z^{-1})u_1(k) + f_1 \\ (1 + a_1^2 z^{-1})y_2(k+1) &= (b_0^2 + b_1^2 z^{-1})u_2(k) + f_2 \end{aligned} \quad (42)$$

where  $f_1$  and  $f_2$  are the neural network compensators. The neural network based decoupling controller was designed as follows.

The forgetting factor for recursively estimating the associated parameters in the linear submodel was selected to be 0.98. The neural network used for predicting the error was set up with  $n_i = 15$ ,  $n_h = 1$ , and  $n = 2$  for the input layer, hidden layer and output layer respectively. The momentum constant was selected to be 0.2, with a learning rate of 0.05. The network was trained twice in each sampling period. The desired closed-loop dynamic polynomial matrix  $\mathbf{T}(z^{-1})$  was, once again, to ensure closed-loop stability, assigned as

$$\mathbf{T}(z^{-1}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} z^{-1} \quad (43)$$

In simulation, the reference values were set to be  $v = [0.2, 400.0]^T$ . The simulation was carried out using a fourth-order Runge-Kutta formula with an initial state  $[1.235, 0.9, 407.3, 402.1]^T$ . Simulation results are shown in Figs 6 and 7. Figure 6 indicates the closed-loop response of the neural network based



**Fig. 6** Example 2: closed-loop response of the control system

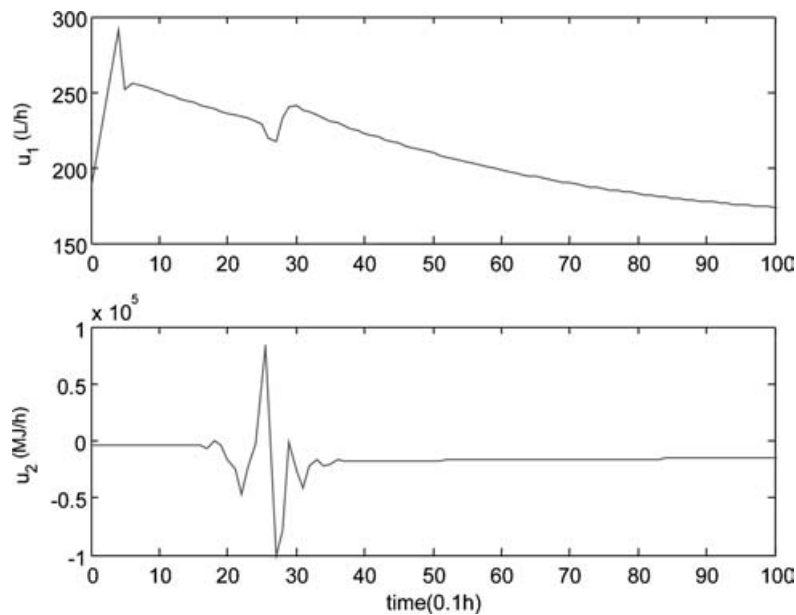


Fig. 7 Example 2: neural network based decoupling controller

decoupling controller, which shows that the entire closed-loop system is stable with a satisfactory performance.

## 5 CONCLUSIONS

The idea of neural network enhanced minimum variance self-tuning PID controller design for SISO plants has been generalized to the MIMO case in this study. By introducing an error agent vector that is approximated on-line by a neural network, non-linear MIMO dynamic plants exhibiting complicated couplings can be readily decoupled so that a MIMO plant with  $n$  outputs can be treated as  $n$  independent SISO plants. Therefore mathematical difficulties in the design procedure and an associated heavy computational cost in control operation can be significantly reduced.

The auto-tuning algorithm and industrial simulations in this study have shown that the neural network approach taken is very promising, and at least worth being tried, as a tool to deal with non-linearities and other complexities. However, it is still far from reaching its maturity in both theory and applications. Topics such as how the structure of a neural network affects the performance of the closed-loop plant, how to analyse the stability of the system, and how to design more robust controllers, etc., still remain open.

Two example application areas were considered in this paper as proof of concept trials in order to show the potential actual application breadth of the

algorithm developed. While these results are felt to be relatively impressive it is clear that further application studies need to be carried out in order to obtain a clearer picture of the true potential for this type of controller. Suffice to say here that from what has been shown, the application potential appears to be considerable.

## ACKNOWLEDGEMENTS

The first two authors are grateful for the support from the grant (GR/M84305) of EPSRC, UK. The authors are also grateful to the anonymous reviewers for their helpful comments and constructive suggestions with regard to the revision of the paper.

## REFERENCES

- 1 Goodhew, I., Hutt, B., and Warwick, K. Control and experimentation of a personal robot tracking system. *Int. J. Modelling, Identification and Control*, 2006, 1(1), 4–12.
- 2 Hunt, K. J., Sbarbaro, D., Zbikowski, R., and Gawthrop, P. J. Neural networks for control systems – a survey. *Automatica*, 1992, 28, 1083–1112.
- 3 Narendra, K. S. and Parthasarathy, K. Identification and control of dynamic systems using neural networks. *IEEE Trans. Neural Networks*, 1990, 1, 4–27.
- 4 Omatu, S., Khalid, M., and Yusof, R. *Neuro-control and its applications* (Eds M. Grimble and M. A. Johnson), Industrial Control Series, 1995 (Springer-Verlag, London).

- 5 **Narendra, K. S.** Neural networks for control: theory and practice. *Proc. IEEE*, 1996, **84**(10), 1385–1406.
- 6 **Zhu, Q. M.** and **Guo, L. Z.** Stable adaptive neuro-control for nonlinear discrete-time system. *IEEE Trans. Neural Networks*, 2004, **15**, 653–662.
- 7 **Rojas, R.** *Neural networks – a systematic introduction*, 1996 (Springer-Verlag, Berlin).
- 8 **Chen, F. C.** Back-propagation neural networks for non-linear self tuning adaptive control. *IEEE Control Systems Mag. (Special Issue on Neural Networks in Control Systems)*, 1990, **10**(3), 44–48.
- 9 **Chen, F. C.** and **Khalil, H. K.** Adaptive control of non-linear systems using neural networks. *Int. J. Control*, 1992, **55**, 1299–1317.
- 10 **Chen, F. C.** and **Khalil, H. K.** Adaptive control of a class of non-linear discrete-time systems using neural networks. *IEEE Trans. – AC*, 1995, **40**(5), 791–801.
- 11 **Proll, T.** and **Karim, M. N.** Model-predictive pH control using real-time NARX approach. *Am. Inst. Chem. Engrs J.*, 1994, **40**, 269–282.
- 12 **Liu, G. P., Kadiramanathan, V.,** and **Billings, S. A.** Predictive control for nonlinear systems using neural networks. *Int. J. Control*, 1998, **71**(6), 1119–1132.
- 13 **Zhu, Q. M., Ma, Z.,** and **Warwick, K.** A neural network enhanced generalised minimum variance self tuning control for non-linear discrete-time systems, *IEE Proc. Control Theory and Applics*, 1999, **146**(4), 319–326.
- 14 **Zhu, Q. M., Warwick, K.,** and **Douce, J. L.** Adaptive general predictive control for non-linear systems. *IEE Proc. Control Theory and Applics*, 1991, **138**(1), 33–40.
- 15 **Zhu, Q. M.** and **Warwick, K.** A neural network enhanced generalized minimum variance self-tuning proportional, integral and derivative control algorithm, for complex dynamic systems. *Proc. Instn Mech. Engrs, Part I: J. Systems and Control Engineering*, 2002, **216**(I3), 265–273.
- 16 **Zhu, Q. M., Hayns, M. R.,** and **Garvey, S. D.** Design of a neural network enhanced auto-tuning PID controller. *Computational intelligence for modelling, control, and automation*, 1999, vol. 54, pp. 69–74 (IOS Press, Oxford).
- 17 **Franklin, G. F., Powell, J. D.,** and **Workman, M. L.** *Digital control of dynamic systems*, 2nd edition, 1990 (Addison-Wesley, Reading, Massachusetts).
- 18 **Astrom, K. J.** and **Wittenmark, B.** *Adaptive control*, 2nd edition, 1995 (Addison-Wesley, Reading, Massachusetts).
- 19 **Marino, R., Tomei, P.,** and **Verrelli, C.** Adaptive control for speed-sensorless induction motors with uncertain load torque and rotor resistance. *Int. J. Adaptive Control and Signal Processing*, 2005, **20**(9), 661–686.
- 20 **Krause, P. C.** *Analysis of electric machinery*, 1986 (McGraw-Hill, New York).
- 21 **Engell, S.** and **Klatt, K. U.** Nonlinear control of a nonminimum phase CSTR. In Proceedings of the 1993 ACC, San Francisco, 1993, pp. 2941–2945.
- 22 **Harris, K. R.** and **Palazoglu, A.** Studies on the analysis of nonlinear processes via functional expansions – III: controller design. *Chem. Engng Sci.*, 1998, **53**(23), 4005–4022.