PERGAMON

Control Engineering Practice 8 (2000) 3–12

www.elsevier.com/locate/conengprac

Adaptive RBFNN versus conventional self-tuning: comparison of two parametric model approaches for non-linear control

C. Pereira[a,b], J. Henriques[a], A. Dourado[a,*]

[a]*Department of Informatics Engineering, CISUC – Centro de Informática e Sistemas da Universidade de Coimbra, Polo II, Pinhal de Marrocos, 3030 Coimbra, Portugal*

[b]*ISEC – Instituto Superior de Engenharia de Coimbra, 3030 Coimbra, Portugal*

Received 8 May 1998; accepted 21 July 1999

Abstract

In this work a practical study evaluates two parametric modelling approaches — linear and non-linear (neural) — for automatic adaptive control. The neural adaptive control is based on a developed hybrid learning technique using an adaptive (on-line) learning rate for a Gaussian radial basis function neural network. The linear approach is used for a self-tuning pole-placement controller. A selective forgetting factor method is applied to both control schemes: in the neural case to estimate on-line the second-layer weights and in the linear case to estimate the parameters of the linear process model. These two techniques are applied to a laboratory-scaled bench plant with the possibility of dynamic changes and different types of disturbances. Experimental results show the superior performance of the neural approach particularly when there are dynamic changes in the process. © *2000 Elsevier Science Ltd. All rights reserved.*

Keywords: Adaptive control; Pole-placement; Non-linear control; Neural networks

1. Introduction

Self-tuning adaptive control based on linear parametric models is a well-established methodology for linear or linearized processes, especially for the regulation problem (Aström & Wittenmark, 1995; Landau, Lozano & M'Saad, 1998). In the case of non-linear processes, the linear parametric approach for modelling can be unsatisfactory because the linear model may not be able to represent appropriately the process in the entire region spanned by the reference. This may degrade the control performance particularly in dynamics change situations. The parameter adaptation of the linear model may solve this drawback to a certain extent, if there are no significant changes in the dynamics of the process (time-delay variations, order variations). Non-linear parametric models are expected to further improve the control system if they have general mapping capabilities, if they are computationally acceptable and possible to adjust them on-line. The radial basis function neural networks (RBFNN) have these properties, and recent progress in their structure building algorithms strengthens their potential for real-time control (Langari, 1997; Yingwei & Sundararajan, 1998). However, there is still a lack of experimental proof of the advantages of this methodology. Most of the works present simulations for relatively simple case studies.

In this paper an experimental comparison is made between two adaptive controllers and a fixed term controller: self-tuning pole-placement (STPP) controller, RBFNN-based controller and a PID controller. For adapting the parameters (of the linear model in the first case and of the output layer weights in the second case) both use the recursive least-squares method with a directional forgetting factor strategy. A hybrid learning technique is developed for the RBFNN inverse model of the process with an adaptive learning rate for the on-line estimation of the centres of the activation functions of the neurons.

Both algorithms were implemented on a real-bench process (laboratory scaled) with a normal microcomputer. Experiments were carried out to compare the controllers with respect to the following situations: set-point tracking, disturbance rejection, process dynamic

* Corresponding author. Tel.: +351-39-790000; fax: +351-39-701266.

E-mail address: dourado@dei.uc.pt (A. Dourado)

PII: S0967-0661(99)00130-6

changes, sensor failure and time variance. To make comparisons, experiments were carried out so that each controller was subjected (as far as was possible) to the same environmental conditions, disturbances and plant variations.

The remainder of this paper is organised as follows. The STPP control strategy is described in Section 2. Section 3 presents and discusses the RBFNN control structure. In Section 4 the experimental results are provided and analysed, and finally in Section 5 some conclusions are presented.

2. The self-tuning pole-placement controller

An indirect adaptive self-tuning scheme is used. The controller parameters are evaluated after a pole-placement design.

2.1. The linear model

The process to be controlled is assumed to be a single-input–single output (SISO) linear system, described by an ARX model (1)

$$\mathbf{A}(q^{-1})y(k) = q^{-d}\mathbf{B}(q^{-1})u(k) + \zeta(k), \tag{1}$$

where $\mathbf{A}(q^{-1})$ and $\mathbf{B}(q^{-1})$ are polynomials (2,3) in the backward shift operator (q^{-1}), assumed to be, in this study, of second order.

$$\mathbf{A}(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2}, \tag{2}$$

$$\mathbf{B}(q^{-1}) = b_1 q^{-1} + b_2 q^{-2}, \tag{3}$$

where k denotes the sampling instant, d is the time delay and $\zeta(k)$ represents white noise.

2.2. Pole-placement controller

A well-known general linear controller can be described by (4) (Aström & Wittenmark, 1995) (Fig. 1 shows this type of control structure):

$$\mathbf{F}(q^{-1})u(k) = \mathbf{H}(q^{-1})r(k) - \mathbf{G}(q^{-1})y(k). \tag{4}$$

The polynomials **F**, **G** and **H** have the following form (5)–(7):

$$\mathbf{F}(q^{-1}) = 1 + f_1 q^{-1} + f_2 q^{-2}, \tag{5}$$

$$\mathbf{G}(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2}, \tag{6}$$

$$\mathbf{H}(q^{-1}) = h_0 + h_1 q^{-1} + h_2 q^{-2}. \tag{7}$$

The closed-loop transfer function is given by (8)

$$y(k) = \frac{q^{-d}\,\mathbf{BH}}{\mathbf{AF} + q^{-d}\,\mathbf{BG}}\, r(k) + \frac{\mathbf{F}}{\mathbf{AF} + q^{-d}\,\mathbf{BG}}\, \zeta(k), \tag{8}$$

$\mathbf{F}(q^{-1})$ and $\mathbf{G}(q^{-1})$ are determined so that the closed-loop poles (roots of $\mathbf{AF} + q^{-d}\mathbf{BG}$) are placed in desired locations, defined by a polynomial $A_m(q^{-1})$. In order to have a null steady-state error, integral action is assumed, **F** will contain the factor $(1 - q^{-1})$ and therefore $\mathbf{F}(q^{-1}) = 0$ for $q = 1$. It is possible to define a polynomial $A_0(q^{-1})$ that specifies the dynamics resulting from disturbances, without influencing the dynamics from reference input. A Diophantine equation is then defined as

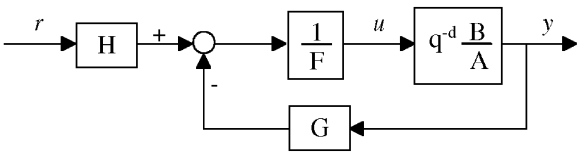

Fig. 1. Linear controller structure.

$$\mathbf{AF} + q^{-d}\,\mathbf{BG} = \mathbf{A}_0\mathbf{A}_m. \tag{9}$$

The polynomial **H** is defined as $\mathbf{H} = \kappa\mathbf{A}_0$, where κ is a constant determined from the steady-state error condition. For more details see Henriques and Dourado (1995).

$$\kappa = \frac{\mathbf{A}_m(1)}{\mathbf{B}(1)}. \tag{10}$$

2.3. Selective forgetting method for least-squares recursive estimation

The process model parameters $\theta^{\mathrm{T}} = [a_1 a_2 b_1 b_2]$ are estimated on line by using a selective forgetting method in order to minimise the square error between the output of the actual plant and the output of the second-order linear model.

A well-known identification method to estimate the parameters $\theta \in \Re^n$, considering n parameters is the RLS-recursive least squares — given by formulae (11) (Ljung, 1987),

$$\varepsilon(k) = y_d(k) - \varphi^{\mathrm{T}}(k)\hat{\theta}(k-1), \tag{11a}$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \mathbf{P}(k)\varphi(k)\varepsilon(k), \tag{11b}$$

$$\mathbf{P}(k) = \frac{1}{\lambda}\left[\mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\varphi(k)\varphi(k)^{\mathrm{T}}\mathbf{P}(k-1)}{\lambda + \varphi(k)^{\mathrm{T}}\mathbf{P}(k-1)\varphi(k)}\right], \tag{11c}$$

where $\varepsilon(k)$ is the estimation error, $y_d(k)$ is the plant output, $\lambda \in \Re^+$ is a forgetting factor, $\mathbf{P}(k) \in \Re^{n\times n}$ is the covariance matrix and $\varphi(k) \in \Re^n$ a regression vector, function of the available information: $\varphi(k)^{\mathrm{T}} = [-y(k-1), y(k-2), u(k-1), u(k-2)]$.

This method solves the problem of estimating varying parameters. However if the sequence of regression vectors $\varphi(k)$ is not sufficiently exciting, the eigenvalues of $\mathbf{P}(k)$ will grow without limit. The resulting problems are discussed for example in Parkum, Poulsen and Holst (1992) and in Bittanti, Bolzern and Campi (1990). In order to solve this problem Parkum et al.

(1992) proposed the following two steps procedure to implement (11):

Step 1: *Measurement update: Eqs*. (11a) *and* (11b).

Step 2: *Update of covariance matrix*

$$\mathbf{P}(k+1) = \Im\{\mathbf{P}(k)\} \geq \mathbf{P}(k). \tag{12}$$

The only difference between the methods lies in the choice of the function $\Im\{\ \}$, the time update for the covariance matrix. It is possible to guarantee that the covariance matrix is bounded from the above and from the below (13):

$$\alpha_{\min}\mathbf{I}_n \leq \mathbf{P}(k) \leq \alpha_{\max}\mathbf{I}_n \tag{13}$$

where $\mathbf{I}_n$ is the identity matrix (n is the number of parameters to be estimated) and $\alpha_{\min}, \alpha_{\max}$ are positive constants verifying (14).

$$0 < \alpha_{\min} < \alpha_{\max} < \infty. \tag{14}$$

The lower boundary allows the identification method to track time-varying parameters, while the upper boundary prevents the blow up of the covariance matrix. The selective forgetting method is a directional one. It is not uniform in the parameter space and it forgets more in those directions with more and better information. The update of the covariance matrix is given by (15)

$$\mathbf{P}(k+1) = \sum_{i=1}^{n} \frac{\alpha_i(k)}{\lambda_i(k)} v_i(k) v_i(k)^{\mathrm{T}}, \tag{15}$$

where $\alpha_i \in \Re$ and $v_i \in \Re$ are the eigenvalues and the corresponding unitary eigenvectors of $\mathbf{P}(k)$. In the conventional forgetting factor method all the λ_i are equal.

In this method the chosen forgetting factor can be a function of the amount of information received in the direction v_i as an increasing function of α_i. The forgetting factor $\lambda_i(k)$ associated with the direction $v_i(k)$, can then be evaluated by

$$\lambda_i(k) = \begin{cases} 1 & \text{if } \alpha_i(k) > \alpha_{\max}, \\ \alpha_i(k)\left[\alpha_{\min} + \alpha_i(k)\dfrac{\alpha_{\max} - \alpha_{\min}}{\alpha_{\max}}\right]^{-1} & \text{if } \alpha_i(k) \leq \alpha_{\max} \end{cases}. \tag{16}$$

3. RBFNN controller

3.1. The RBFNN structure and training

In most generic sense, a RBFNN is any network that has radial symmetric activation functions. The output of a hidden neuron is a function of the distance between an input vector and the centre of the function. Fig. 2 shows the structure of this type of network. Given an input vector of dimension p, $\mathbf{x} \in \Re^p$, the output of the network is described by (17)

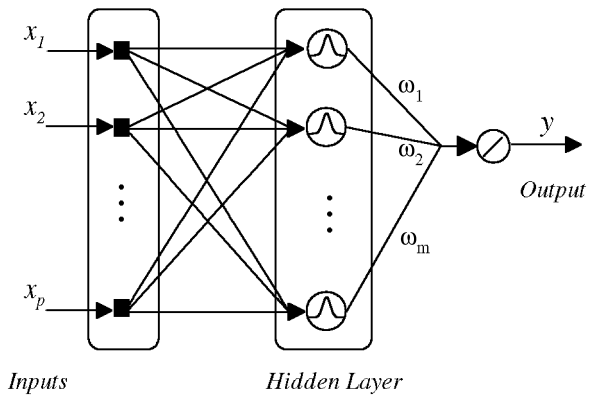

Fig. 2. RBFNN structure.

$$y = \sum_{i=1}^{m} \omega_i \varphi_i(\|\mathbf{x} - \mathbf{c}_i\|) \tag{17}$$

in which ω_i ($i = 1, 2, \ldots, m$) are the network weights, $\|.\|$ denotes de Euclidean norm, $\mathbf{c}_i \in \Re^p$ are the basis function centres and φ: $\Re^p \to \Re$ is the radial basis function. This function can be selected in one of the several ways. In this work the Gaussian exponential function was used (18), where σ_i defines the width of the receptive field

$$\varphi_i : \Re^p \to \Re,\ \varphi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{\sigma_i^2}\right). \tag{18}$$

In the on-line learning procedure proposed in this paper, all the parameters of the NN are identified in two steps. The first step applies the recursive *K-means clustering* in a modified version of the technique proposed by Moody and Darken (1989), Pereira, Henriques, Ribeiro and Dourado (1996), which is essentially a competitive learning algorithm of Kohonen type. In this work a monitoring procedure is introduced to update the learning rate. In the second step the previously described selective forgetting factor is applied to the estimation of the weights of the output layer.

3.2. Calculation of the centres and widths of the Gaussian functions with adaptive learning rate

Since during each period of time the network input exists only in some regions of the input space, it is reasonable to allocate the centres of the Gaussian functions in those regions, using a clustering algorithm, like for example the *K-Means Clustering* (Moody & Darken, 1989). However, the clustering should be adaptive, in order to follow the region in the input space that is being spanned each time. For that purpose consider the following simple but efficient adaptation technique:

Step 1: The initial centres $\mathbf{c}_i$ ($i = 1, 2, \ldots, m$) are chosen randomly.

Step 2: Read the next input vector, $\mathbf{x}(k)$.

Step 3: Modify only the closest centre $\mathbf{c}_i$ according to (19).

$$\mathbf{c}_i(k) = \mathbf{c}_i(k-1) + \beta(k)\,[\mathbf{x}(k) - \mathbf{c}_i(k-1)]. \tag{19}$$

Step 4: Compute the sum of squared error (SSE) for the last N sampling periods:

$$SSE = \sum_{i=1}^{N} [y_p(k-i) - y_d(k-i)]^2. \tag{20}$$

Step 5: If the *SSE* is greater than ε_c then (21):

$$\beta(k+1) = \beta_c \tag{21}$$

otherwise (22):

$$\beta(k+1) = \gamma\beta(k) \tag{22}$$

where $\beta(k)$ represents the learning rate, $0 < \beta(k) < 1$, $y_p(k)$ is the process output, $y_d(k)$ is the desired response, and β_c is an initial constant value for the learning rate. In case the modelling error over a period of time (sliding window of N samples) is greater than a selected fixed threshold ε_c, the adaptive learning rate remains constant, otherwise it decreases to zero, in which γ represents the decay constant, $0 < \gamma < 1$. The widths of the Gaussian functions are determined by the *P-nearest-neighbour* heuristic method (Moody & Darken, 1989):

$$\sigma_l = \sqrt{\frac{1}{P}\sum_{i=1}^{P} \|\mathbf{c}_l - \mathbf{c}_i\|^2}, \tag{23}$$

where $\mathbf{c}_i$ ($i = 1, 2, \ldots, P$) are the P-nearest neighbours of the centre $\mathbf{c}_l$. In this work, $P = 2$ is used.

3.3. Learning the weights by the selective forgetting algorithm

Due to the linearity of the error function with respect to the weights, the weight matrix can be solved so that the error is minimised, in terms of the recursive least-squares algorithm. The conditions assuring that the network inputs provide persistency of excitation, with fixed centres, are given in Gorinevsky (1995). However if the input is sufficiently exciting, the elements of the covariance matrix converge to zero, loosing the tracking capability if the system is time variant. As seen in Section 2 a selective forgetting algorithm allows this problem to be solved and guarantees that the parameter covariance matrix is bounded from above and below.

Another disadvantage of the usual methods is that the forgetting is uniform in space, leading to problems when only some directions of the parameter space are excited. This is the case with the radial basis function networks due to the local representation property of the Gaussian functions. One input will only significantly activate a very restricted number of neurons. The selective forgetting algorithm applied in RBF learning guarantees that the forgetting is proportional to the amount of information received in each direction.

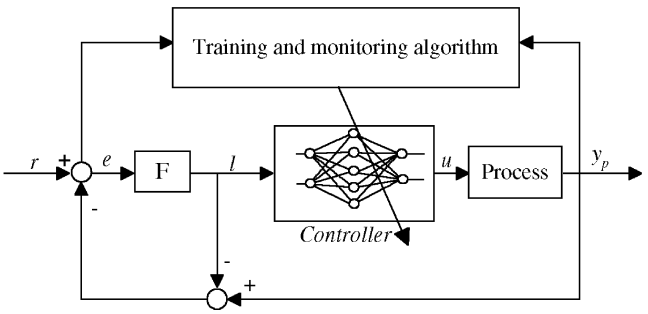

Fig. 3. Structure for real-time adaptive RBFNN control.

The learning procedure described above is used to train the available controller in the form of a RBFNN. The control structure proposed is shown in Fig. 3, with monitoring and adaptive training. This is a simplified adaptive internal model control (IMC) scheme in adaptive form. The RBFNN is used to model the inverse of the plant. The training signal is the difference between the reference and the process output. The feedback signal is a measure of the accuracy of the inverse model. If it is perfect, then l equals y_p and the control is in open loop and in steady state, l equals r. In the presence of process changes or disturbances, y_p no longer equals the reference, a correcting feedback signal $y_p - l$ is generated and at the same time the RBFNN learns on-line the new situation. The output follows quickly the reference. The first-order filter F is applied to smoothen the error signal for reference changes and introduces robustness into the control system.

This method can even be applied to singular systems, which do not satisfy the invertibility conditions over the whole operational space, as it searches for local models. It learns the process dynamics in order to annul the steady-state error, retaining its parameters. Hence, the attained model is local to the addressed operating point.

4. Experimental results

4.1. The experimental bench

The experimental bench is composed of two laboratory processes, the process trainer PT326 and the Process Control Simulator PCS327 (Feedback, 1984) (Fig. 4). Air is forced to circulate by a fan blower through a tube and heated at the inlet. There is an energised electric resistance inside the tube, and due to the Joule effect, heat is released by the resistance and transmitted, by convection, to the circulating air, resulting in heated air. This is a non-linear system with a pure time delay. The pure time delay depends on the position of the temperature sensor

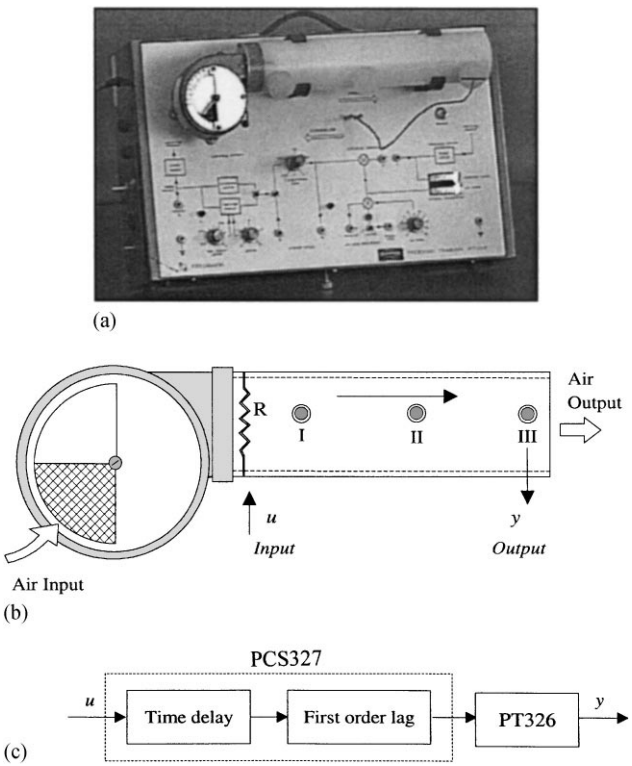

Fig. 4. (a) The experimental bench PT326; (b) PT326 schematics diagram; (c) A schematic diagram of the PT326 in series with the PCS327.

element (positions I, II, and III, see Fig. 4(b)) and the damper position (Ω). The system input, $u(k)$, is the voltage applied to the power electronic circuit feeding the heating resistance, and the output, $y_p(k)$, is the outlet air temperature, expressed by a voltage, between -10 and 10 V, issued from the transducer and conditioning electronics.

The PCS327 is an analogue electronic apparatus allowing the introduction of additional poles of different values and additional time-delays as illustrated in Fig. 4(c).

Four groups of experiments were carried out for each controller: set-point tracking, load disturbances, system dynamics change with sensor failure and time variance. These experiments were conducted using a PC with A/D and D/A converters, the algorithms being implemented in C code. The vertical scales in the figures represent the output air temperature in °C. Its relation to the voltage from the sensor/transducer is 1V-22°C, 3V-32°C, 5V-38°C. As a result of the construction of the apparatus there is a non-linearity in this relation.

Each experiment was of 120 s duration. Initial conditions were: damper position $\Omega = 40^\circ$, detector probe in III position. The sampling time was 200 ms. In these simulations, the following values were chosen using a trial-and-error procedure:

RBFNN: $m = 5$, $\alpha_{\max} = 0.25$, $\alpha_{\min} = 0.01$,

$\mathbf{P}(0) = 0.25\mathbf{I}_5$, $\varepsilon_c = 1$, $\beta_c = 0.6$, $N = 20$, $\gamma = 0.9$.

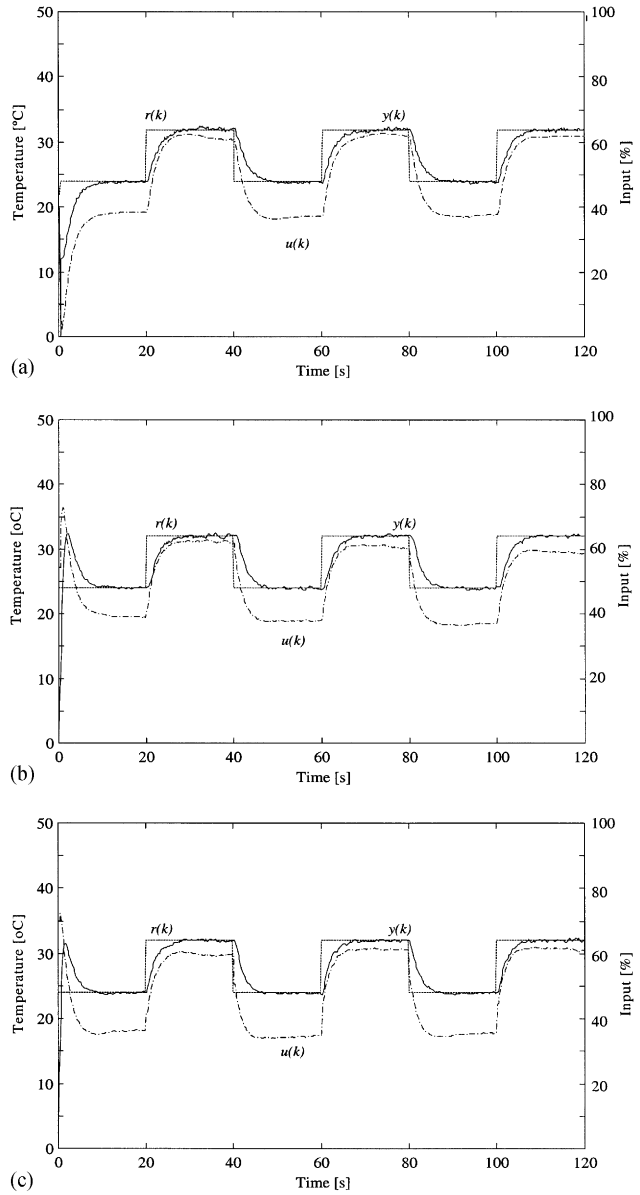

Fig. 5. (a) Set-point tracking — RBFNN; (b) Set-point tracking — STPP; (c) Set-point tracking — fixed term PID.

STPP: $\mathbf{A}_m = (1 - 0.7q^{-1})$, $\mathbf{A}_0 = (1 - 0.5q^{-1})$,

$\alpha_{\min} = 0.01$, $\alpha_{\max} = 0.2$, $\mathbf{P}(0) = 0.1\mathbf{I}_4$.

In the PID controller, the Ziegler–Nichols method was applied, resulting in the following control action: $u(k+1) = u(k) + 0.12e(k) - 0.51e(k-1) + 0.13e(k-2)$.

The NARX (24) model, in the terminology of Chen and Billings (1989), can represent the non-linear system:

$$y_p(k) = f[\, y_p(k-1), y_p(k-2), \ldots, y_p(k-n_a), u(k-1), u(k-2), \ldots, u(k-n_b), e(k)]. \quad (24)$$

In this case $n_a = 2$ and $n_b = 2$ were considered.

In order to assess the performance of the three controllers several indices were considered, namely, sum of square error $sse = \sum_{k=0}^{N} e(k)^2$, sum of absolute error $sae = \sum_{k=0}^{N} |e(k)|$, sum of instant square error $sise = \sum_{k=0}^{N} ke(k)^2$, sum of square control $ssc = \sum_{k=0}^{N} u(k)^2$ and finally the sum of square change of control $sscc = \sum_{k=0}^{N} (u(k+1) - u(k))^2$. With respect to the output error the first one will punish major output tracking errors and the sum of instant square error will penalise most steady-state errors. The last two indices evaluate the actuation effort and smoothness.

4.2. Set-point tracking

In this first set of experiments the tracking performances of the controllers with respect to set-point changes (step input) was studied. Fig. 5(a) and (b) show the behaviour of the three controllers. Both the RBF and STPP controllers started without any a priori parameter estimation. Note that despite being put into control without a priori knowledge of the process (the Gaussian function centres were initialised at random and the widths and the output layer weights were simply initialised at zero), the RBFNN initial performance is quite reasonable. In a similar way the STPP control performed as well as the RBF controller also without any a priori identification. In the remaining experiments, the adaptive controllers also start without any particular knowledge about the process. In this first experiment, the performance of the PID was comparable to the others, and even slightly better (see the performance indices table) since the tuning method was based on the given set points.

4.3. Variation in the damper position (Ω)

The second simulation run addressed the disturbance rejection capabilities of the controllers. Variations were considered in the flow rate of input air: $\Omega = 60^\circ$ at instant 20 s and $\Omega = 20^\circ$ at instant 40 s. As can be seen, both the adaptive controllers perform well in maintaining the output at their set points. Fig. 6(a)–(c), show the behaviour of process for a regulation situation. The PID controller exhibits poor performance in these situations.

4.4. Changes in system dynamics and sensor failure

Now consider an experiment with the purpose of studying the effect of a dynamics change. A detector probe shift from positions III to II occurs suddenly at 20 s. Being a manual operation, there is a sensor failure during that period. After 20 s, the sensor is repositioned to position III. The behaviour of the controllers under this influence are shown in Fig. 7(a)–(c). The RBFNN and

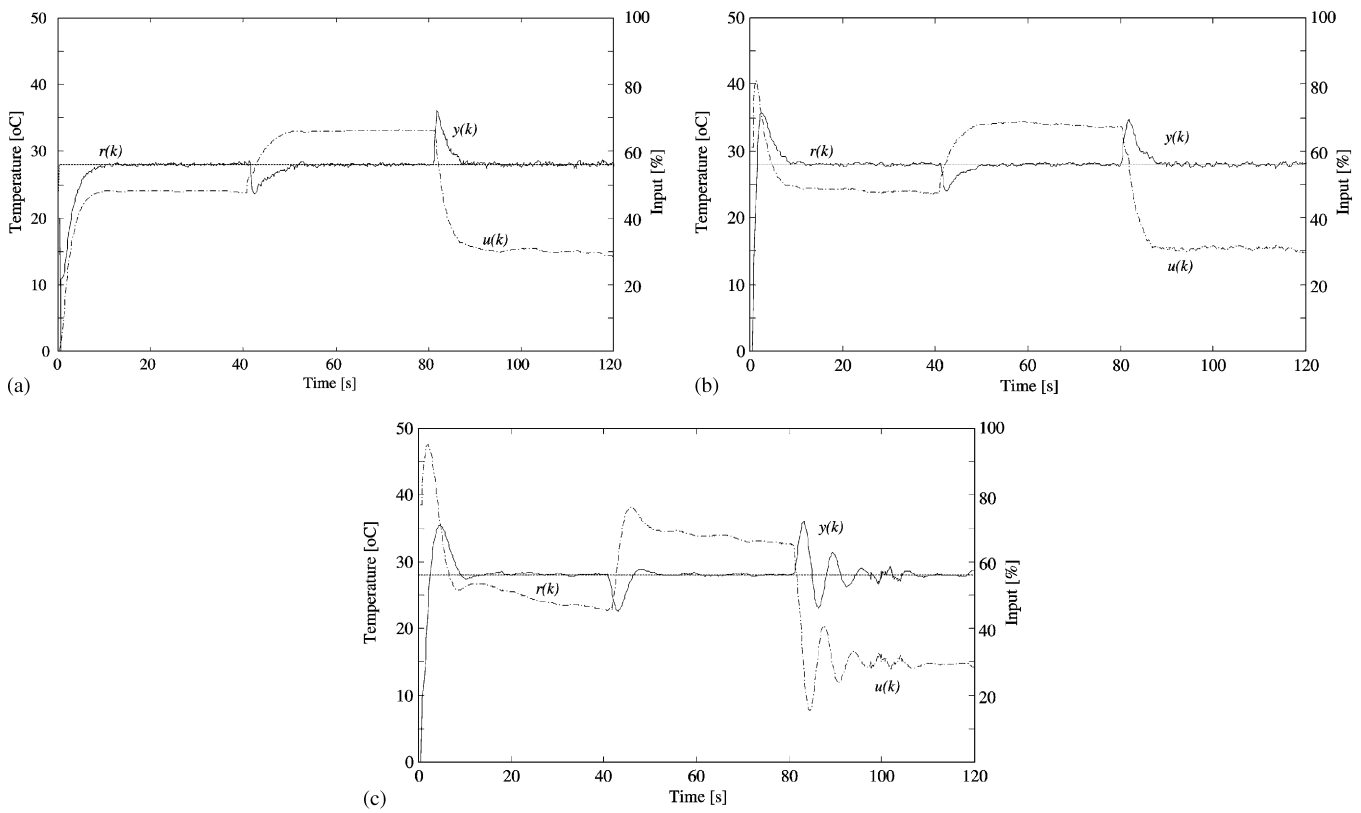

Fig. 6. (a) Regulation–variation in input air — RBFNN; (b) Regulation–variation in input air — STPP; (c) Regulation–variation in input air — PID.

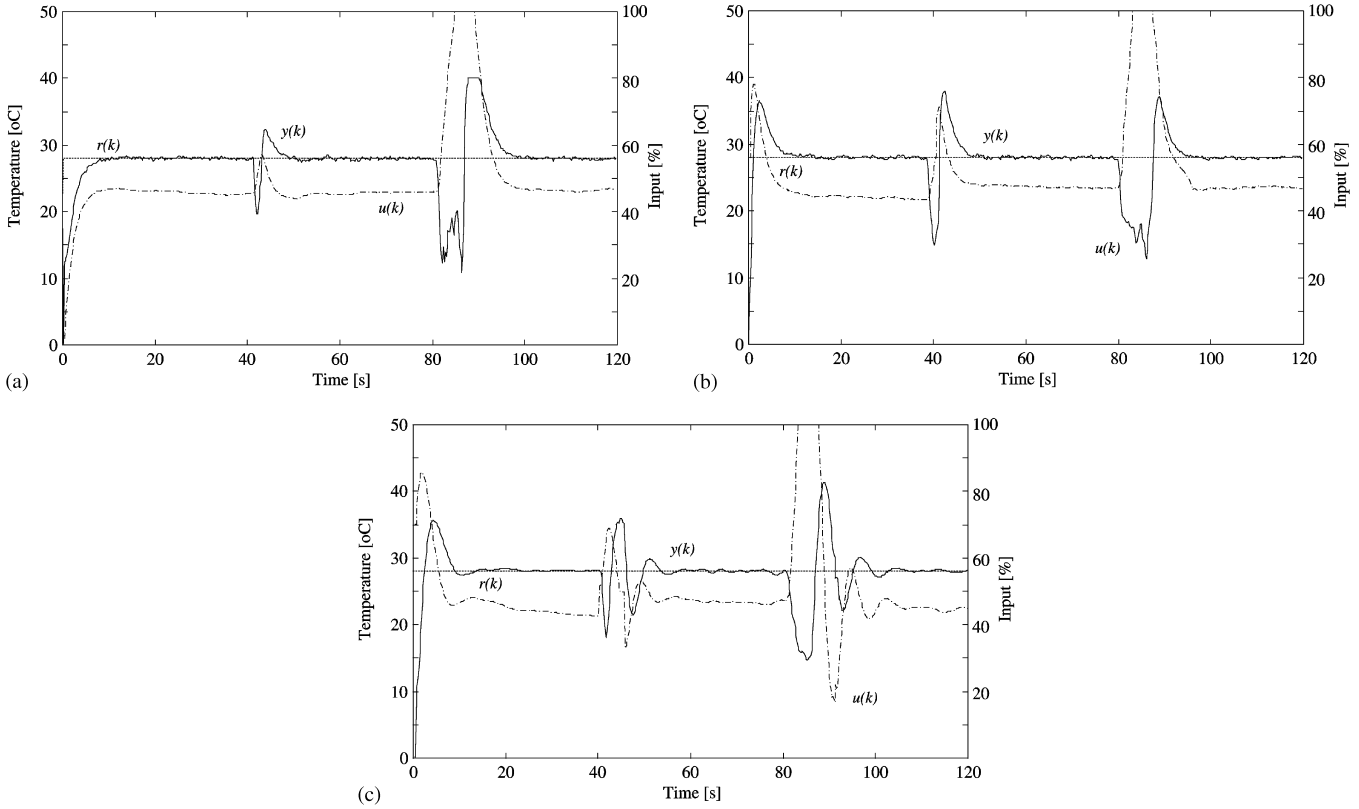

Fig. 7. (a) Regulation–sensor fault — RBFNN; (b) Regulation–sensor fault — STPP; (c) Regulation–sensor fault — PID.

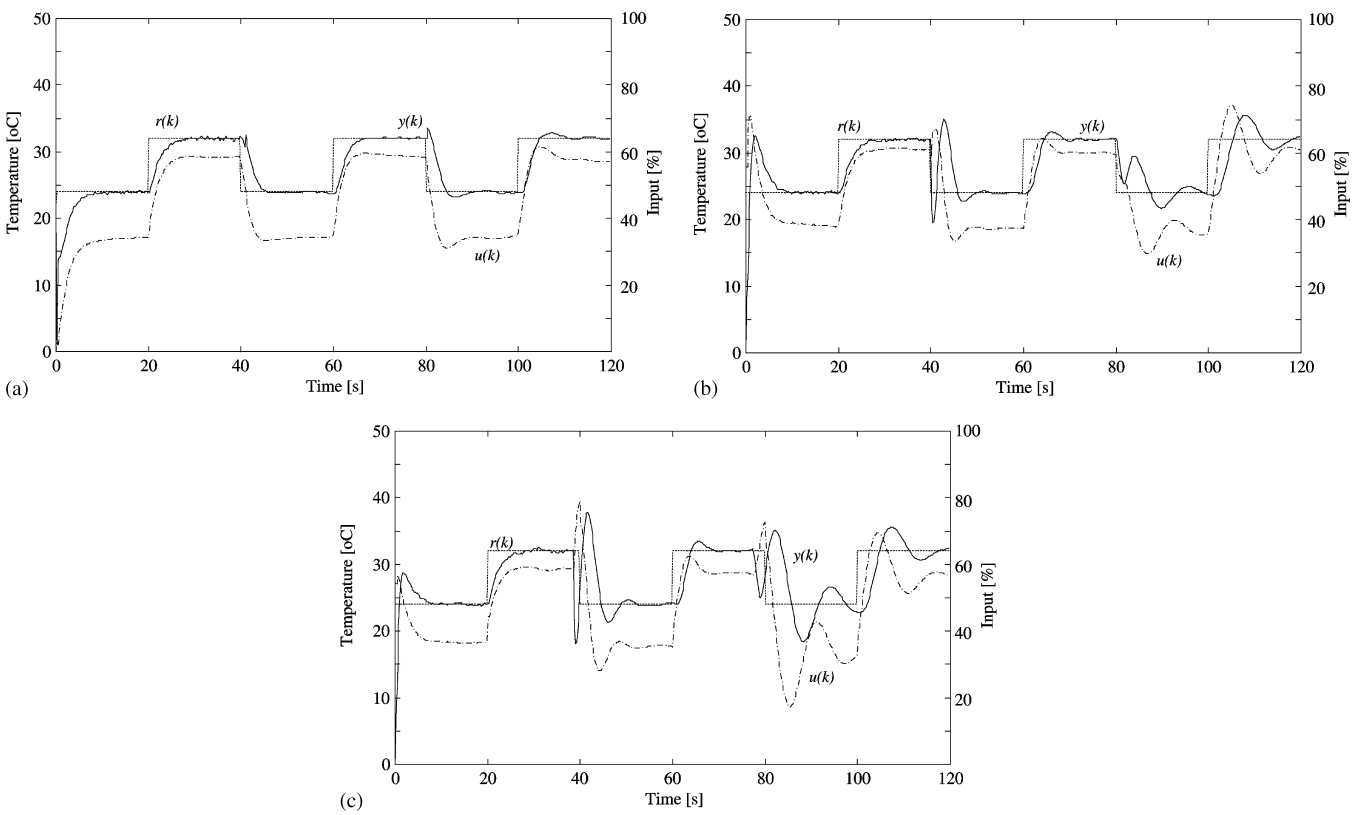

Fig. 8. (a) Additional poles and delays — RBFNN; (b) Additional poles and delays — STPP; (c) Additional poles and delays — PID.

STPP performed best under this condition. The PID performed worst during the second sensor failure (approximately 5 s) as its output oscillated adversely.

4.5. Introduction of additional poles and time delays

Another set of experiments used the PT327 for the introduction of delays and poles in the process dynamics. Fig. 8(a)–(c) show the controllers' behaviour with respect to the introduction of a delay and a pole. A pole of 1 s and a delay of 1 s were introduced at 40 s and 80 s, respectively. The STPP and PID controllers had some difficulty in controlling the process after the introduction of the pole, and were unable to control the process when the pole and delay were simultaneously present. The RBFNN shows greater robustness in these situations as illustrated in the figures. It can be stated that the neural network behaves significantly better in situations of process order changes.

4.6. Time variance

In order to assess how well the controllers handle time variance in this last experiment, the calculated actuation signal $u_q(k)$ is multiplied by a time variable gain: $u(k) = \mu u_q(k)$, where $u(k)$ is the actual actuation signal, and μ is the variable gain, increasing linearly from 1 to 3. Once again the RBFNN controller performs quite well compared to the other applied techniques as can be seen in Fig. 9. The fixed PID controller shows the highest performance degradation for large values of μ.

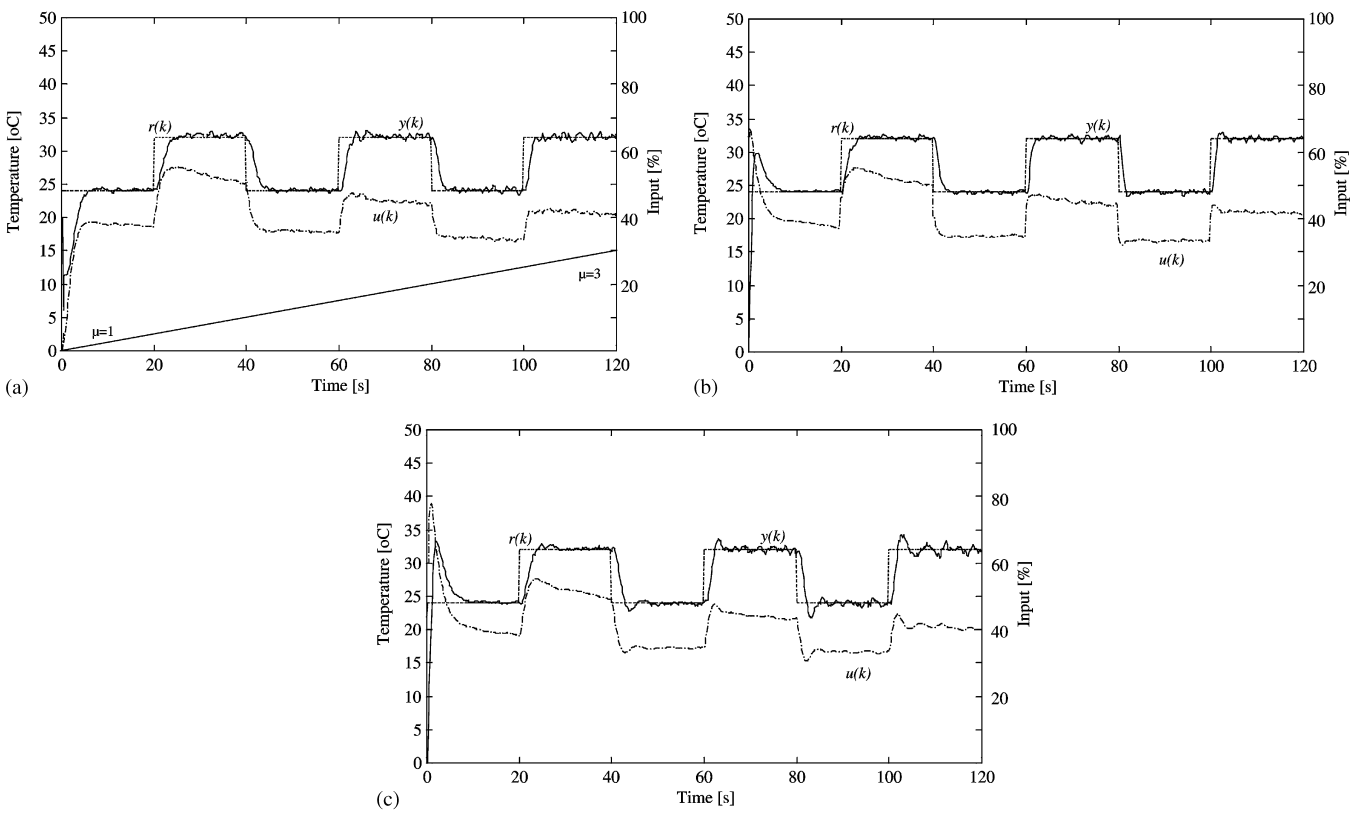

Fig. 9. (a) Time variance — RBFNN; (b) Time variance — STPP; (c) Time variance — PID.

Table 1
Summary of the experimental results

Criteria	RBFNN	STPP	PID
Computational complexity	Medium (heavy, if using a lot of neurons)	Medium	Light
Tracking performance	Good	Good	Best (if tuned correctly)
Effect of sensor failure	Good	Good	Worst
Effect of dynamics change	Superior	Regular	Poor
Time variance	Good	Good	Worst

Table 2
Calculated performance criteria (calculated value/best performance)

RBFNN	SSE	SAE	SISE	SSC	SSCC
Set-point tracking	1.24	1.62	1.09	1	1
Variation in input air	1.46	1.07	1	1	1
Sensor fault	1.13	1	1.30	1	1
Additional poles and delays	1	1	1	1	1
Time variance	1	1	1	1	1
STPP	SSE	SAE	SISE	SSC	SSCC
Set-point tracking	1.40	1.23	1.30	1.04	2.93
Variation in input air	1	1	1.01	1.11	2.37
Sensor fault	1	1.00	1	1.09	2.43
Additional poles and delays	1.52	1.50	1.62	1.11	2.59
Time variance	1.56	1.39	1.43	1.05	1.70
PID	SSE	SAE	SISE	SSC	SSCC
Set-point tracking	1	1	1	1.03	2.18
Variation in input air	2.70	1.93	5.54	1.15	2.81
Sensor fault	2.29	1.50	2.82	1.03	3.81
Additional poles and delays	2.23	1.85	2.71	1.09	4.23
Time variance	1.65	1.38	1.37	1.05	2.04

5. Conclusions

The results obtained by the RBFNN controller are good. In trivial situations it shows comparable performance with the STPP and tuned-fixed PID controllers (see Table 1). The fixed parameter controller applied exhibits very good performance if tuned accordingly. In some experiments the ST performance is better with respect to the error but worse with respect to the control effort. In Table 2 each cell contains the quotient between the calculated criteria and the best-performance criteria for the same set of experiments. The neural controller produces a smoother control. Considering both criteria of error and control effort the RBFNN seems to be better in all situations. From the point of view of robustness the RBFNN controller performs best when there are more serious changes in dynamics. Considering the performance and development costs the RBFNN is preferable. For a more complex, high order non-linear system these advantages will probably be more evident than in this case study. These facts are understandable if one thinks about the RBFNN controller as a self-tuning non-linear controller, as in fact it is. The self-tuning property lies in the recursive on-line learning of the NN, and to control a non-linear process, particularly in the tracking problem, a non-linear self-tuning controller is preferable to a linear ST one.

The authors hope that the present work will be an encouragement for industrial engineers to implement neural adaptive controllers in real factories.

Acknowledgements

The authors thank the anonymous reviewers who contributed to the improvement of this paper. This work was partially financed by JNICT/PRAXIS XXI Program.

References

Aström, K., & Wittenmark, B. (1995). *Adaptive control*. Reading, MA: Addison Wesley Publishing Company.

Bittanti, S., Bolzern, P., & Campi, M. (1990). Recursive least-squares identification algorithms with incomplete excitation: Convergence analysis and application to adaptive control. *IEEE TAC*, *35*, 1371–1373.

Chen, S., & Billings, S. A. (1989). Representation of non-linear systems: The NARMAX model. *International Journal of Control*, *49*, 1013–1032.

Feedback (1984). *Process control simulator PCS 327*. Process Trainer PT326, Feedback Instruments Limited, Park Road, Crowborough, Sussex, England.

Gorinevsky (1995). On the persistency of excitation in radial basis function network identification of nonlinear systems. *IEEE Transactions on Neural Networks*, *6*, 1237–1244.

Henriques, J., & Dourado, A. (1995). Decoupling and directional forgetting factor adaptive controller of industrial evaporators. *Proceedings of ECC95*, Rome, Italy (pp. 1149–1154).

Landau, I. D., Lozano, R., & M'Saad, M. (1998). *Adaptive control*. London: Springer.

Langari, R. (1997). Radial basis function networks regression weights, and the expectation–maximisation algorithm. *IEEE Transactions on Systems Man and Cybernetics — Part A: Systems and Humans*, *27*, 613–623.

Ljung, L. (1987). *System identification — Theory for the user*. Englewood Cliffs, NJ: Prentice-Hall.

Moody, J., & Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*, 281–294.

Parkum, J., Poulsen, N., & Holst, J. (1992). Recursive forgetting algorithms. *International Journal of Control*, *55*, 109–128.

Pereira, C., Henriques, J., Ribeiro, B., & Dourado, A. (1996). Real time adaptive training of RBFNN using a selective forgetting algorithm. *Proceedings of the International Conference on Engineering Applications of Neural Networks*, London (pp. 431–434).

Yingwei, L., & Sundararajan, N. (1998). Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. *IEEE Transactions on Neural Networks*, *9*, 308–318.