chemical reactors

I.M. Galván[a,*], P. Isasi[a], J.M. Zaldívar[b]

[a] Department of Computer Science, Carlos III University of Madrid, Avda. de la Universidad, 30, 28911 Leganés, Madrid, Spain
[b] European Commission, Joint Research Center, Institute for Systems, Informatics and Safety, TP 250, 21020 Ispra (VA), Italy

Received 1 August 1999; received in revised form 1 June 2000; accepted 1 September 2000

Abstract

This paper is focused on the development of non linear neural models able to provide appropriate predictions when acting as process simulators. Parallel identification models can be used for this purpose. However, in this work it is shown that since the parameters of parallel identification models are estimated using multilayer feed forward networks, the approximation of dynamic systems could be not suitable. The solution proposed in this work consists of building up parallel models using a particular recurrent neural network. This network allows to identify the parameter sets of the parallel model in order to generate process simulators. Hence, it is possible to guarantee better dynamic predictions. The dynamic behaviour of the heat transfer fluid temperature in a jacketed chemical reactor has been selected as a case study. The results suggest that parallel models based on the recurrent neural network proposed in this work can be seen as an alternative to phenomenological models for simulating the dynamic behaviour of the heating/cooling circuits.

Keywords: Neural networks; Modelling; NARMA models; Process simulators; Chemical reactors

1. Introduction

The present study is concerned with the generation of neural non linear models capable of acting, in an efficient fashion, as process simulators. That is, given the initial state of the system $y(0)$ and the input signal $u(k)$, the model should be able to predict the outputs of the process, $y(k)$, over a large number of sampling times. This kind of situations, where models have to act as process simulator, are often presented in control applications.

Non linear auto regressive moving average (NARMA) models provide a unified representation for a wide class of non linear systems (Leontaritis and Billings, 1985). In a NARMA description the system is modelled in terms of a non linear functional expansion of lagged inputs and outputs. That functional can be very complex and its explicit form is usually unknown. However, the development of mathematical analysis has led to the discovery of important classes of approxima tion functions which can be used to that end. These include polynomials, trigonometric series, orthogonal functions, splines, etc. Other additional family of functions which has been evolved are artificial neural networks (ANN). Different authors (Cybenko, 1989; Hornik et al., 1989) have shown independently that multilayer neural networks, with as few as one hidden layer and with an arbitrarily large number of neurones in the hidden layer are capable of approximating any non linear continuous function.

The application of ANN to non linear dynamic process modelling problem has been dominated by the static multilayer neural networks, that is multilayer perceptrons (Rumelhart et al., 1986) and radial basis neural networks (Moody and Darken, 1998, 1989). In these structures the processing of input patterns does not depend upon the order of presentation during the training or recall. Moreover, connections between neurones are subjected to strong restrictions avoiding the creation of cycles or loops among neurones. Thus, the representation and processing of temporal informa tion is not an intrinsic capability of these architectures. However, it is always possible to use static structures to encode temporal information. The approach consists in forming tapped delay line representations of applied

*Corresponding author. Fax: +34-91-624-9129.
E-mail address: igalvan@inf.uc3m.es (I.M. Galván).

inputs and measured outputs. Thus, they can be used to generate NARMA models which are referred in this work as *series–parallel models*. This approach has been used in numerous applications (Narendra and Parthasarathy, 1990; Bhat and McAvoy, 1990; Chen and Billings, 1992; Levin and Narendra, 1995; Choi et al., 1996; Suykens and Bersini, 1996; Levin and Narendra, 1996; Narendra and Mukhopadhyay, 1997; Lu and Basan, 1998; Noriega and Wang, 1998). However, these models cannot act as process simulators because a discrete sequence of delayed measured output of the process is required.

Narendra and Parthasarthy (1990) have proposed a different structure of neural NARMA model which consists in replacing the delayed measured output of process in the series–parallel model by the model predicted values at earlier time steps when a process simulator is required. They are referred to as *parallel models* and they have the capability of simulating the dynamic system. However, in this paper, it is shown that since its parameters have been identified using a multilayer feed-forward network, the approximation of the dynamic behaviour of the process provided by this parallel structure could not be suitable.

An alternative to build up neural non-linear simulators is the use of recurrent neural networks (RNN), where feedback connections are allowed. RNN were introduced by Hopfield (1982) and they are characterised by the presence of feedback connections between the neurones of the same layer, including the originating node itself, and/or connections to nodes of preceding layers. The capabilities, for temporal representation of recurrent networks, have been shown to be considerably greater than those of purely static networks. Examples of RNN models can be found in Polycarpou and Ioannou (1991); Yonhg and Nikolaou (1993); Srinivasan et al. (1994); Parlos et al. (1994); Kosmatopoulos et al. (1995) and Stage and Sendhoff (1997). Furthermore, models built up with recurrent structures are capable of acting as process simulator because any discrete sequence of delayed input and output process must be considered to represent temporal information. However, RNNs are generally significantly more complex than feed-forward neural networks and presently there is less experience with their operation. This complexity is due, principally, to the increase in the adjustable parameters and to the problems found in generating efficient learning algorithm to guarantee the convergence of the network weights.

The neural network architecture studied in this work to build up non-linear models capable of acting as process simulators, parallel neural NARMA (PNNARMA), is a particular architecture of RNN. The PNNARMA consists of adding feedback connections to a multilayer feed-forward neural network from the output to the input layer. The recurrent connections do not have any associated parameter, which implies that the degree of complexity is equivalent to the complexity of the multilayer feed-forward networks. The training of the PNNARMA is carried out using a learning algorithm based on the dynamic backpropagation algorithms (Narendra and Parthasarathy, 1991; Wan and Beaufays, 1996; Cohen et al., 1997). A learning rule based on these algorithms is inferred for the proposed architecture and some ways to accelerate the convergence of the learning method are also presented.

The neural model generated by the PNNARMA architecture appertains to the class of parallel models. It predicts the output of the process using only a discrete sequence of delayed input variables. In contrast to the parallel model proposed in Narendra and Parthasarathy (1990), in this case, it is possible to guarantee appropriate approximations to the dynamic behaviour when the model is acting as process simulator because the parameters have been trained with this purpose.

In order to validate the capability of PNNARMA model proposed in this work, the modelling of a real process which describes the dynamic behaviour of the heat transfer fluid temperature circulating in a jacket of a chemical bath reactor has been selected as a study case. Batch processes are usually very complex, with reaction systems that normally are not entirely known, they have also strong non-linear dynamics and their parameters are varying with time.

Phenomenological or first-principle models, which are generated according to the physical laws governing the dynamic evolution of the system, can be used to model the system. In same cases that models are often not available at all and in many cases they are time consuming and extremely expensive to build up. The mathematical modelling of a batch reactor is based on the formulation of the mass and heat balances that leads to a set of algebraic–differential equations which, when solved, produce the temperature and concentration profiles as a function of time. One of the non-trivial problems when simulating isothermal batch operations is the modelling of the heating/cooling circuits as well as their controllers which will influence the dynamic behaviour of the reactor temperature as well as the safety of the process (Zaldívar et al., 1996). Furthermore, the heating/cooling circuits will change from installation to installation and hence a new mathematical model has to be developed. Hence, the development of efficient methods for simulating non-linear dynamic systems is of great value and interest in this field.

The paper is organised as follows. The series–parallel models and parallel models identification are reviewed in Section 2. In Section 2, the PNNARMA architecture is also presented and the dynamic backpropagation learning algorithm for the adjustment of the PNNARMA weights is developed. This section also

includes the description of the parallel model proposed in this work and a comparative study of the different parallel model structures when the goal is to guarantee the best dynamic prediction of the process. The description of the pilot plant reactor and the heating/cooling circuits are carried out in Section 3. In this section, the phenomenological model based on the energy balances of the heating/cooling circuits and different structures of NARMA models are presented. In Section 4, the capability of parallel models based on feed-forward neural network and on the PNNARMA network to simulate the heat transfer fluid temperature is presented. In addition, the PNNARMA model performances are compared with the phenomenological model. The results suggest that PNNARMA neural models provide the best approximations of the heat transfer fluid temperature and they can be seen as an alternative to phenomenological models. Finally, in Section 5, the conclusions drawn from this study are presented.

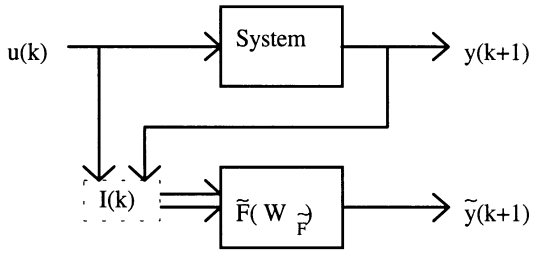

Fig. 1. Series parallel neural model structure.

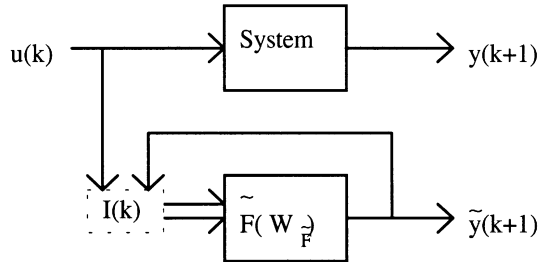

Fig. 2. Parallel neural model structure.

2. Neural network model: PNNARMA

This section deals with the parallel neural model proposed in this paper. Firstly, the classical architecture of NARMA models based on neural networks are described. Secondly, the architecture of the PNNARMA neural model is introduced. For this new architecture, a learning rule is proposed and the parallel model based on the PNNARMA neural network is described more in detail. Finally, a comparative analysis between this new approach and the classical parallel models has been done to emphasise the advantages of the PNNARMA model.

2.1. NARMA models based on neural networks

Consider a discrete non-linear dynamic system governed by the following NARMA model:

$$y(k+1) = F(u(k-d), \ldots, u(k-d-n_u), y(k), \ldots, y(k-n_y)), \quad (1)$$

where $y(\cdot)$ and $u(\cdot)$ are discrete sequences for the system output and input, respectively, d is the delay of the process and F is some non-linear unknown map.

Introducing the vector $I(k) = (u(k-d), \ldots, u(k-d-n_u), y(k), \ldots, y(k-n_y))$ as the kth network input pattern, the multilayer feed-forward neural networks can be used to approximate the map F obtaining the *series–parallel model* (Narendra and Parthasarathy, 1991), see Fig. 1:

$$y(k+1) = F(u(k-d), \ldots, u(k-d-n_u), y(k), \ldots, y(k-n_y), W_F), \quad (2)$$

where $y(k+1)$ is the output of the multilayer feed-forward network associated with the input pattern $I(k)$ and W_F is the set of parameters. This parameter set is obtained using the traditional backpropagation algorithm (Rumelhart et al., 1986) also referred to in this study as static backpropagation algorithm in order to minimise the following function:

$$E^I = \frac{1}{2N} \sum_{k}^{N} \sum_{d}^{1} (y(k+1) - y(k+1))^2, \quad (3)$$

where N is the number of patterns. This function is called in this work as *identification error*.

When the model has to simulate the dynamic of the process, the sequence of measured values $y(k), \ldots, y(k-n_y)$ is not available and the series–parallel models identification given by Eq. (2) cannot be used. To solve this problem, Narendra and Parthasarthy (1991) had proposed a different structure of neural model, referred to as *parallel model* (see Fig. 2). It consists in replacing the measured values $y(k), \ldots, y(k-n_y)$ in Eq. (2) by the network-predicted values $y(k), \ldots, y(k-n_y)$. Denoting $y_p(k)$ as the output of this parallel model, it can be written as

$$y_p(k+1) = F(u(k-d), \ldots, u(k-d-n_u), y_p(k), \ldots, y_p(k-n_y), W_F). \quad (4)$$

The parameters of the neural model given by Eq. (4) are fixed to the parameters of the series–parallel model, W_F, Eq. (2). Hence, when the parallel model is used to simulate the dynamic behaviour of the system, the training of the series–parallel model has been previously carried out.

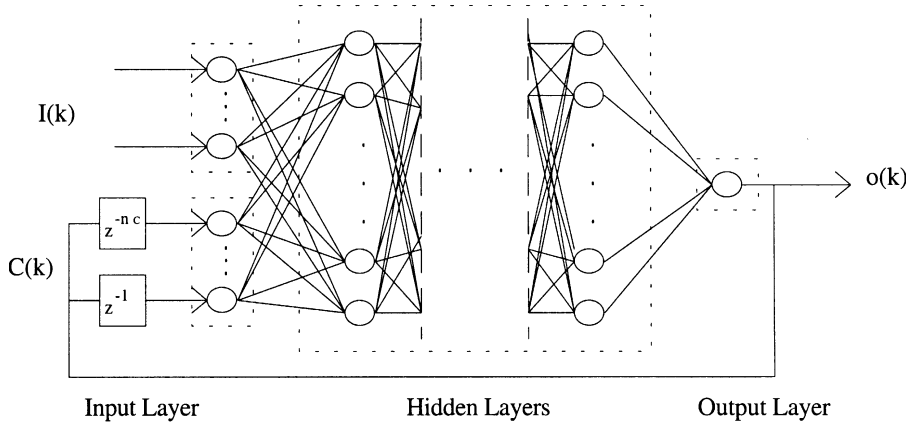

Fig. 3. Recurrent neural network architecture.

2.2. PNNARMA architecture

The PNNARMA is constructed by starting from a multilayer feed-forward neural network and by adding feedback connections from the output neurone to the input layer, as is shown in Fig. 3.

The neurones in the PNNARMA are divided into the input, hidden and output layers, as usual. The input layer is formed by two groups of neurones. The first group acts as the input to the network receiving the input patterns from the external world, $I(k) = (I_1(k), \ldots, I_{n_1}(k))$. The second group is formed by the context neurones which memorise the output of the network associated with patterns previously presented. Introducing the vector $C(k) = (C_1(k), \ldots, C_{n_c}(k))$ to indicate the activation of context neurones, each component is calculated as

$$C_i(k) = z^{\ i}(o(k)), \quad i = 1, \ldots, n_c, \tag{5}$$

where $o(k)$ is the network answer to the kth input pattern, $I(k)$, and $z^{\ i}$ is an operator defined as: given $x(1), x(2), \ldots, x(n), \ldots$ a discrete sequence, the operator $z^{\ i}$ delays by i terms that sequence, that is, $z^{\ i}(x(n)) = x(n-i)$. Hence, when the kth input pattern is presented to the network, the context neurones receive the output of the network associated with $(k-1)$th, $(k-2), \ldots, (k-n_c)$ input patterns previously presented.

The activation of the remaining neurones in the network (hidden and output neurones) follow the same equations that the neurones activation in the multilayer feed-forward network, this is, the sigmoidal function applied to the weighted sum of the neurone activation's in the previous layer.

The PNNARMA determines a correspondence from $\Re^{n_1}$ to $\Re$, denoted by $F_r(\cdot, C(k), W)$:

$$o(k) = F_r(I(k), C(k), W). \tag{6}$$

2.3. Computation of the learning rule

As usual, the learning procedure is based on stochastic gradient methods and the weights are adjusted along the negative gradient direction of local errors as

$$w^{(k)} = w^{(k\ \ 1)} + \alpha(t(k) - o(k))\frac{\partial o(k)}{\partial w} \quad \forall w, \tag{7}$$

where $o(k)$ and $t(k)$ are the output of the PNNARMA and desired output, respectively.

Since recurrent connections appear in the PNNARMA network, the static backpropagation algorithm cannot be directly used to calculate the term $\partial o(k)/\partial w$ in Eq. (7). The PNNARMA output depends on the earlier network activations and the total derivative concept must be applied.[1] The learning algorithm used in this work to carry out the training of the PNNARMA network is a reworked version of dynamic backpropagation algorithms developed in Narendra and Parthasarathy (1991). In the next, the learning rule is inferred.

Considering that the vector $C(k)$ is composed by the PNNARMA outputs delayed, Eq. (5), and applying the total derivative concept in Eq. (6), it follows that

$$\frac{\partial o(k)}{\partial w} = \frac{\partial F_r(I(k), C(k), W)}{\partial w} + \sum_{i\ \ 1}^{n_c} \frac{\partial F_r(I(k), C(k), W)}{\partial C_i(k)} \frac{\partial C_i(k)}{\partial w}. \tag{8}$$

Tacking account that $C_i(k) = o(k-i)$, it follows that

$$\frac{\partial C_i(k)}{\partial w} = \frac{\partial o(k-i)}{\partial w}. \tag{9}$$

Denoting $x(k) = \partial o(k)/\partial w$, the variation of the kth output of the PNNARMA network can be obtained as

[1] Given a function $f(x(w), w)$, where the first variable x depends also on the parameter w, the total derivative or total variation of the function f respect to the parameter w is $\partial f/\partial x \ \ \partial x/\partial w + \partial f/\partial w$.

the output at time k of a dynamic process governed by the following equation:

$$x(k) = \frac{\partial F_r(I(k), C(k), W)}{\partial w} + \sum_{i\ 1}^{n_{cr}} \frac{\partial F_r(I(k), C(k), W)}{\partial C_i(k)} \cdot x(k-i) \qquad (10)$$

with initial conditions $x(0) = x(1) = \cdots = x(n_c) = 0$.

Both terms

$\partial F_r(I(k), C(k), W)/\partial w$

and

$\partial F_r(I(k), C(k), W)/\partial C_i(k)$

have to be calculated for each kth input pattern. Since the internal structure of the PNNARMA is a feed-forward network, those terms can be determined using the static backpropagation (Jordan, 1989).

Therefore, at each time step, the output of the dynamic system given by Eq. (10) is calculated and the weights of the PNNARMA are adjusted according to the following rule:

$$w^{(k)} = w^{(k\ 1)} + \alpha(t(k) - o(k))x(k) \quad \forall w. \qquad (11)$$

Remark (about the learning rule given by Eq. (11)). It is worth pointing out that for each pattern, it will be necessary to apply $n_c + 1$ times the static backpropagation to calculate the partial derivatives

$$\left(\frac{\partial F_r}{\partial w}, \frac{\partial F_r}{\partial C_1(k)}, \ldots, \frac{\partial F_r}{\partial C_{n_c}(k)}\right).$$

Hence, the computational effort could become not recommendable in practical applications, principally, when the number of context neurones is high. Since this could be a substantial problem, it is interesting to use approximate methods which may require considerably less effort. It is possible, for instance, to approximate the variations of the PNNARMA outputs, $\partial o(k)/\partial w$ by the term $\partial F_r/\partial w$; see Eq. (8), obtaining the learning rule given the static backpropagation algorithm:

$$w^{(k)} = w^{(k\ 1)} + \alpha(t(k) - o(k))\frac{\partial F_r}{\partial w}, \ \forall w. \qquad (12)$$

If the term $\partial F_r/\partial w$ in the Eq. (8) is dominant with respect to

$$\sum_{i\ 1}^{n_c} \frac{\partial F_r}{\partial C_i(k)} \frac{\partial C_i(k)}{\partial w},$$

the learning rule given by Eq. (12) may perform quite satisfactorily. However, it must be stressed that $\partial F_r/\partial w$ is an approximation of the precise gradient. Hence, the validity of the rule given by Eq. (12) depends on the approximation goodness, that is, how much dominant is the term $\partial F_r/\partial w$ in Eq. (8).

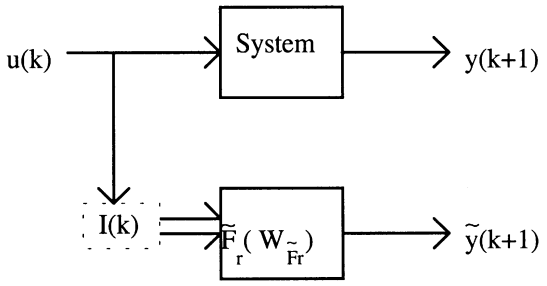

Fig. 4. Parallel neural model based on PNNARMA.

In the simulations carried out in this work, it has been observed that the advantages of the use of precise gradient to train the PNNARMA network, Eq. (11), instead of gradient approximation, Eq. (12), are normally traduced in reaching a minimum in a smaller number of learning iterations.

2.4. NARMA model based on PNNARMA network

As it has been mentioned in Section 2.1, the series–parallel models, Eq. (2), cannot simulate the dynamics of the process because the sequence of measured values $y(k), \ldots, y(k-n_y)$ is not available. Only the parallel model given by Eq. (4) can act as process simulator. The PNNARMA network previously described allows the generation of models capable of simulating the dynamic of the process. Considering the vector $I(k) = (u(k-d), \ldots, u(k-n_u))$ as the input vector to the PNNARMA network and $n_y + 1$ context neurones, it is possible to built up the following parallel model (Fig. 4):

$$\begin{aligned} y_r(k+1) &= F_r(I(k), C(k), W_{F_r}) \\ &= F_r(u(k-d), \ldots, u(k-d-n_u), \\ &C_i(k), \ldots, C_{n_y}(k), W_{F_r}), \end{aligned} \qquad (13)$$

where $y_r(k+1)$ is the output of PNNARMA network; $C_i(k) = y_r(k+1-i)$, $i = 1, \ldots, n_y + 1$, and W_{F_r} is the set of weights of the PNNARMA.

The structure of this model is identical to the structure of parallel model given by Eq. (4) because the context neurones memorise the outputs of network delayed by $1, 2, \ldots, n_y$ time steps. However, there exists an important difference between them: the way to determine the sets of parameters. The weights of the parallel model given by Eq. (4) are fixed to the set W_F obtained after the training of the series–parallel model. Thus, they are updated using the data set $\{I(k) = (u(k-d),$

$\ldots, u(k-d-n_u), y(k), \ldots, y(k-n_y),), y(k+1)\}$. However, the set W_{F_r} captures the map $\{I(k) = (u(k-d), \ldots, u(k-d-n_u)), \ y(k+1)\}, \ \forall k = d, \ldots, N-1$, because they are estimated using the PNNARMA network. This fact produces different behaviours of models when they are used to simulate the dynamic of the process, as it will be shown in the next subsection.

The training of the new parallel structure is carried out using the dynamic backpropagation algorithm described in Section 2.3. Generally, the learning of RNNs presents some problems concerning the algorithm convergence and the number of learning cycles to reach some minimal point. One of the main influential factor is the initialisation of network weights. Due to the feedback connections, the training procedure may be difficult and arduous when the initial weights are set to random values. When the learning of the parallel model given by Eq. (13) is realised, the initial weights also have an important influence on the convergence of the algorithm. To accelerate the convergence, it is convenient, to start from weights which are set to values with some information about the map that will be approximated. Due to the equivalence between the multilayer feed-forward network and the PNNARMA network, the parameters of the series–parallel model given by Eq. (2), W_F, can be used to initialise the parallel model given by Eq. (13).

2.5. PNNARMA versus classical parallel models

To have models capable of simulating the dynamic behaviour of the process, parallel model structures have to be used. Two different parallel models have previously been presented, Eqs. (4) and (13). At this point, the immediate question that arises concerns the choice of the approach to be used. In the next, it will be shown that to guarantee the best process simulator the parallel model based on the PNNARMA network, Eq. (13), should be used.

Let us assume for simplicity that $n_y = 0\, y\, n_u = 0$. The models given by Eqs. (2) and (4) can be re-written as

$$y(k+1) = F(y(k), u(k-d), W_F), \tag{14}$$

$$y_p(k+1) = F(y_p(k), u(k-d), W_F) \tag{15}$$

with $y_p(d) = y(d)$.

The approximated outputs by the series–parallel model identification can be expressed as

$$y(k+1) = y(k+1) + \varepsilon_{k+1}, \quad k = d, ..., N-1, \tag{16}$$

where ε_{k+1} is a real number indicating the local error associated with the input pattern $I(k) = (y(k), u(k-d))$.

Definition. Given $I(k) = (y(k), u(k-d))$ the kth input pattern, $y(k+1)$ the answer of the multilayer feed-forward neural network with parameters W_F and ε a real number, the answer of that network for the input pattern $I_\varepsilon(k) = (y(k) + \varepsilon, u(k-d))$ is written as $y(k+1) + \delta(\varepsilon)$, where $\delta(\varepsilon)$ is a real number evaluating the capability of the neural network to approximate the perturbed input pattern $I_\varepsilon(k)$.

Taking into account this definition and Eq. (16), the parallel model outputs given by Eq. (15) can be expressed as

$$y_p(d+1) = y(d+1) = y(d+1) + \varepsilon_{d+1}, \tag{17}$$

$$\begin{aligned} y_p(k+1) = {} & y(k+1) + \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} \\ & + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1}) \cdots))] \end{aligned} \tag{18}$$

$$\begin{aligned} k \ &= d+1, \ldots, N-1, \\ &= y(k+1) + \varepsilon_{k+1} + \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} \\ &\quad + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1}) \cdots))], \end{aligned}$$

where $\delta(\varepsilon_{d+1}), \ \delta(\varepsilon_{d+2} + \delta(\varepsilon_{d+1})), \ \ldots, \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1})...))]$, are real numbers measuring the capability of multilayer feed-forward neural network with parameters W_F to respond to perturbations of the input patterns $I(d+1), \ldots, I(k)$, respectively.

Proof. The equality in Eq. (17) is immediate because $y_p(d) = y(d)$.

For $k = d+1$ the equality is right,

$$y(d+2) \ = F(y(d+1), u(1), W_F) = y(d+2) + \varepsilon_{d+2},$$

$$\begin{aligned} y_p(d+2) \ &= F(y_p(d+1), u(1), W_F) \\ &= F(y(d+1) + \varepsilon_{d+1}, u(1), W_F) \\ &= y(d+2) + \delta(\varepsilon_{d+1}) \\ &= y(d+2) + \varepsilon_{d+2} + \delta(\varepsilon_{d+1}). \end{aligned}$$

Assuming the equality for k,

$$y(k+1) = F(y(k), u(k-d), W_F) = y(k) + \varepsilon_k,$$

$$\begin{aligned} y_p(k+1) = {} & F(y_p(k), u(k-d), W_F) \\ = {} & F(y(k) + \varepsilon_k + \delta(\varepsilon_{k\ 1} + \delta(\varepsilon_{k\ 2} + \cdots \\ & + \delta(\varepsilon_{d+1})...)), u(k-d), W_F) \\ = {} & y(k+1) + \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} \\ & + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1})...))] \\ = {} & y(k+1) + \varepsilon_{k+1} + \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} \\ & + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1})...))], \end{aligned}$$

i.e., it is right for $k+1$. Hence, for induction, it is concluded that the expression given in Eq. (18) is valid for $k = d+1, \ldots, N-1$.

In consequence, the performance of the parallel model given by Eq. (15) depends on the capability of multilayer feed-forward neural networks to approximate perturbations of input patterns, this depends on the values of $\delta(\varepsilon)$ (see Eqs. (17) and (18)). Taking into account the definition of $\delta(\cdot)$, it concludes that if $\varepsilon \approx 0$, $\delta(\varepsilon)$ will be also close to zero because multilayer feed-forward networks filter the noise in its inputs. If ε is distant from zero, the values of $\delta(\varepsilon)$ will increase because multilayer neural networks cannot correctly approximate input patterns different from patterns used to the training procedure. Therefore, two different cases may be distinguished:

- If ε_k in Eq. (18) are zero or close to zero for $k = d+1, \ldots, N-1$, then $\delta(\varepsilon_{d+1})$, $\delta(\varepsilon_{d+2} + \delta(\varepsilon_{d+1}))$, $\ldots, \delta[\varepsilon_N + \delta(\varepsilon_{N\ 1} + \delta(\varepsilon_{N\ 2} + \cdots + \delta(\varepsilon_{d+1})\ \ldots))]$ will be close to zero. Thus, the approximations $y_p(k+\ 1) \forall k = d, \ldots, N-1$ could be considered suitable predictions of the dynamic system.
- If there exists a natural number n such that the local error ε_n in Eq. (18) is distant to zero, then the real number $\delta[\varepsilon_n + \delta(\varepsilon_{n\ 1} + \delta(\varepsilon_{n\ 2} + \cdots + \delta(\varepsilon_{d+1}) \ldots))]$ is also distant to zero. Hence, $y_p(n+1)$ is not an appropriate approximation of the measured value $y(n+1)$, (see Eq. (18)). Moreover, that error is propagated into the next approximations and the network will have to filter more and more higher errors. In this case, it is not possible to expect predictions $y_p(n+i)$ such that $y_p(n+i) \approx y(n+i)$ for $i = 2, \ldots, N-1$. Hence, if errors occur in the series–parallel model for some pattern, the capability of the parallel model given by Eq. (15) to simulate the dynamic process behaviour can be compromised.

From the previous discussion, it can be concluded that the predictive capability of the parallel model given by Eq. (4) could be destroyed since the input vectors to the model, $(u(k-d), \ldots, u(k-d-n_u), y_p(k), \ldots, y_p(k-n_y))$, were not used during the training procedure. To guarantee an adequate approximation, its parameters should be estimated using those input patterns, i.e., minimising the following performance function, also called *prediction error*:

$$E^P = \frac{1}{2N} \sum_{k\ d}^{N\ 1} (y(k+1) - y_p(k+1))^2, \tag{19}$$

where $y_p(k+1)$ is the output of the parallel model given by Eq. (13).

The parameters W_F are adjusted along the negative gradient direction of the identification error (Eq. (3)). However, this does not imply that W_F point minimises also the error prediction, Eq. (19), since the surfaces described by identification and prediction errors are different. For $n_y = 0\, y\, n_u = 0$ and taking into account Eqs. (17) and (18), it follows that

$$\begin{aligned} 2NE^P = & \sum_{k\ d}^{N\ 1} (y(k+1) - y(k+1))^2 \\ & + \sum_{k\ d+1}^{N\ 1} (\delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} + \delta(\varepsilon_{k\ 2} \\ & + \cdots + \delta(\varepsilon_{d+1}) \cdots))])^2 \\ & - 2 \sum_{k\ d+1}^{N\ 1} (y(k+1) - y(k+1)) \delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} \\ & + \delta(\varepsilon_{k\ 2} + \cdots + \delta(\varepsilon_{d+1}) \cdots))]. \end{aligned}$$

Hence, from Eqs. (3) and (16),

$$\begin{aligned} E^P = E^I + & \frac{1}{2N} \sum_{k\ d+1}^{N\ 1} (\delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} + \delta(\varepsilon_{k\ 2} \\ & + \cdots + \delta(\varepsilon_{d+1}) \cdots))])^2 \\ & - \frac{1}{N} \sum_{k\ d+1}^{N\ 1} \varepsilon_{k+1} (\delta[\varepsilon_k + \delta(\varepsilon_{k\ 1} + \delta(\varepsilon_{k\ 2} \\ & + \cdots + \delta(\varepsilon_{d+1}) \cdots))])^2. \end{aligned}$$

As can be observed, the difference between both surfaces is given by errors ε_k and by the capability of the multilayer feed-forward networks to respond to patterns that are different from the training data set. Therefore, if the aim is to build up models able to act as process simulators so that they provide admissible approximations, their parameters must be adjusted along the negative direction of prediction error E^P. Due to the structure of the PNNARMA network, the performance function used to identify the parallel model proposed in this work coincides with the prediction error and the parameter set W_{F_r} in Eq. (13) is determined to minimise the prediction error. Thus, the parallel model given by Eq. (13) will provide better dynamic approximations than the parallel model given by Eq. (4).

3. Modelling of the heat transfer fluid temperature in a chemical batch reactor

The final goal of this work is to model a real process which describes the dynamic behaviour of the heat transfer fluid temperature circulating in the jacket of a chemical batch reactor. Next, a brief description of the reactor and heating/cooling circuits is presented. Then, a phenomenological model describing the dynamic behaviour of the heating/cooling circuits is included. Finally, different structures of NARMA models capable of simulating the temperature in the jacket reactor are presented.

3.1. Reactor description

The facility for investigation runaway events safely (FIRES) reactor (Hernández and Zaldívar, 1990; Zaldívar et al., 1993) is a 100 l stainless-steel, glass-lined pilot reactor which is equipped with a standard cooling/heating jacket and is provided with condensers, to allow the study of reactions under reflux conditions.

The installation has a distributed control system, organised in such a way that at the head of the hierarchy a workstation supervises and interfaces with the operator. This supervising workstation is connected to a control unit which manages the data acquisition and plant control sub-systems. The complete operation of the facility is menu-supported and experiments can be run manually, by means of a control panel, under direct control through the operator console, or in fully automatic mode following a "recipe" introduced beforehand by the operator. During the experiments, values of variables, status of difference devices, e.g. pumps, valves, etc. and control loops are displayed on screen and recorded on magnetic disk. This recorded data allow subsequent evaluation of parameters such as heat of reaction, heat transfer coefficient, heat capacity, etc. In parallel, there is a second workstation from which it is possible to read the acquired data, to send orders to the control system, to simulate on-line the experiment and to carry out on-line calculations.

The cooling/heating systems (Fig. 5), consist of two loops in which a 50/50 wt% glycol–water mixture circulates at high speed, i.e. 12 m^3/h. The main circuit is connected to the reactor jacket and contains a heat source: 20 m of the metal tube are heated by the application of a direct current, up to 40 kW of power. The secondary circuits provide cold fluid to the main loop and have a large capacity vessel, 2 m^3, connected to a refrigeration unit of 20 kW. The coolant is stored at a temperature between -20 and $-25^{\circ}C$ and it can be used for providing full cooling in emergency situations through a bypass valve.

The control configuration is summarised in Fig. 5. When the process is carried out in isothermal conditions, the reactor temperature is maintained at its desired value, T_r^{sp}, by adjusting the temperature set-point for the glycol–water mixture recirculating through the reactor jacket, T_e^{sp}. This is accomplished by the master controller. The temperature of the heat transfer fluid which circulates through the reactor jacket is controlled using a slave controller. When the slave controller output is between 11 and 100% (equivalent to digital signal between 450 and 4095), the controller is in cooling mode. In this mode, the controller output is used to open the control valve V1, while in parallel the valve V2 is closed by the same percentage. When V1 is open, the coolant at $-20^{\circ}C$ enters the main loop cooling the glycol–water mixture recirculating through the jacket. The correlation between the flow of the cold fluid, Q_c, and the signal send by the slave controller is shown in Fig. 6a. For slave controller output between 0 and 10% the controller is in heating mode. The power generated, q_h, measured during characterisation experiments is shown in Fig. 6b as a function of the control action. As can be seen, the control action is non-linear in both cases.

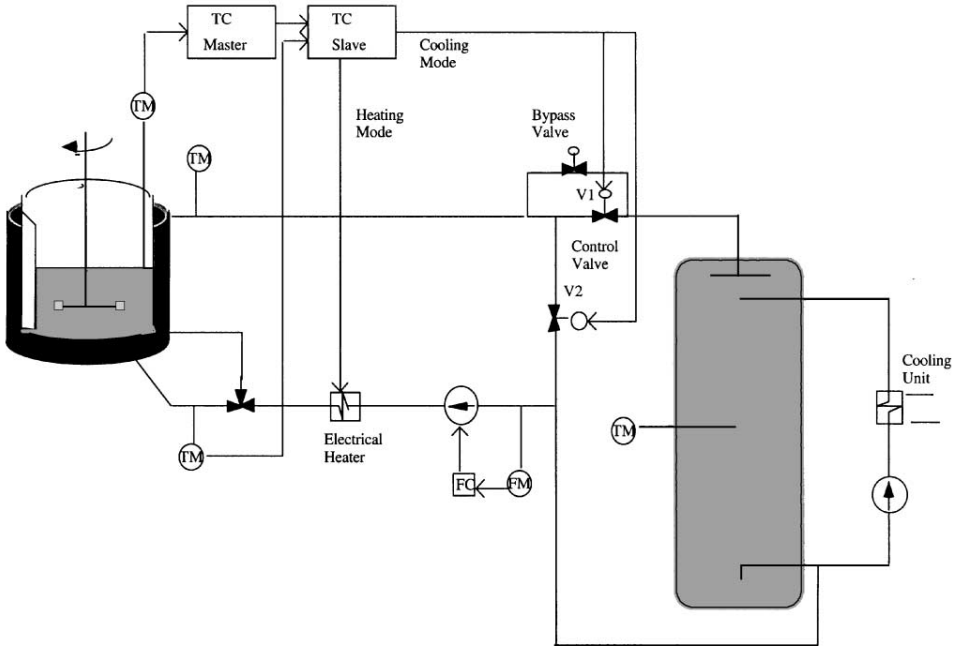

Fig. 5. Schematic layout of the heating/cooling circuits.

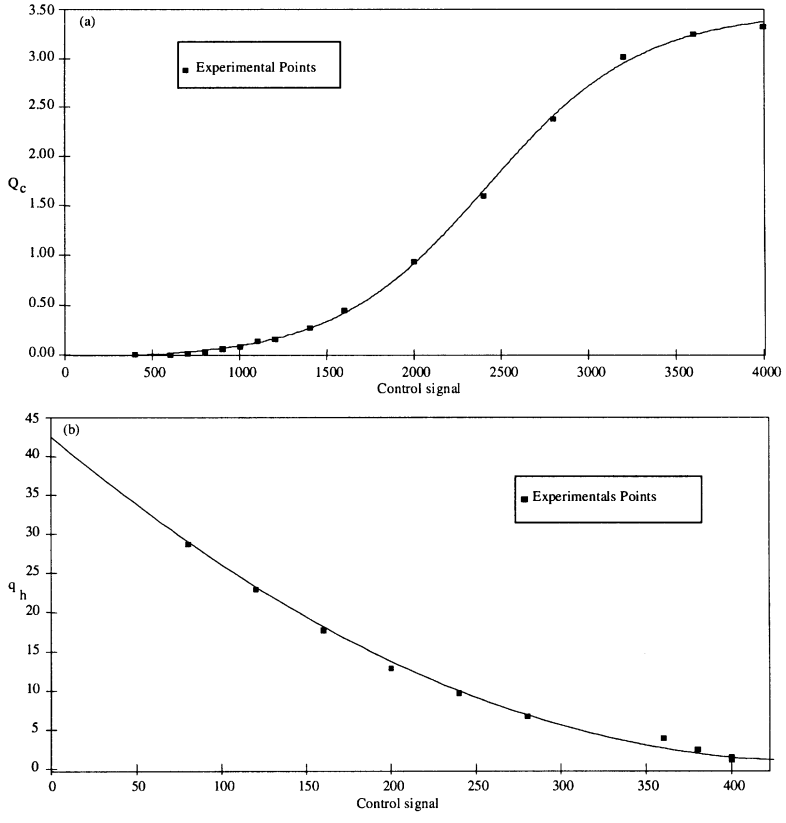

Fig. 6. Correlation between the control signal and (a) Q_c; (b) q_h.

3.2. First-principle model

To develop a phenomenological model, the heating/cooling circuits, (see Fig. 5), were divided in different thermally homogeneous subsystems and separate energy balances were applied (Zaldívar et al., 1996). The subsystems identified are the following: the main loop T_e, the insulation of the main loop T_p, and the cold reservoir T_c. During previous experiments inhomogeneities in the temperatures of the 2 m^3 cold reservoir were observed. For this reason, it was divided into six equal thermal capacity parts.

The followings considerations and simplifications have been taken into account:

1. The electrical heating and the opening/closing of the control valves are zero-time constant processes.
2. The heat losses are modelled with constant transfer coefficients.
3. The specific heat capacity, C_p, of the heat transfer fluid (50/50 8% glycol–water mixture) is calculated as a function of the temperature.
4. No secondary heat effects in the reactor have been considered, e.g. no heat losses, no power introduced by agitation, etc.
5. No heat accumulation in the reactor wall has been taken into account (Hernández et al., 1993).

Energy balances in the subsystems are written as follows:

- Energy balance in the main loop:

$$\frac{dT_e}{dt} = \frac{1}{\Gamma_e}\left[q_{r\times n} + q_h + q_p - Q_c\, Cp_e(T_e - T_{c_0})\right] + \frac{T_p - T}{\tau_{e_1}}.$$

- Energy balance in the isolation of the main loop:

$$\frac{dT_p}{dt} = \frac{T_e - T_p}{\tau_{e_2}} - \frac{q_L}{\tau_k}.$$

- Energy balance in the cooling loop:

$$\frac{dT_{c_i}}{dt} = \frac{6}{\Gamma_c}[Q_c Cp_e(T_e - T_{c_i}) + Q_k Cp_c(T_k - T_{c_i})],$$

$$\frac{dT_{c_i}}{dt} = \frac{6}{\Gamma_c} Q_T C_{p_c}(T_{c_i} - T_{c_1}),$$

$$\frac{dT_{c_2}}{dt} = \frac{6}{\Gamma_c} Q_T Cp_c(T_{c_1} - T_{c_2}),$$

$$\frac{dT_{c_3}}{dt} = \frac{6}{\Gamma_c} Q_T Cp_c (T_{c_2} - T_{c_3}),$$

$$\frac{dT_{c_4}}{dt} = \frac{6}{\Gamma_c} Q_T Cp_c (T_{c_3} - T_{c_4}),$$

$$\frac{dT_{c_0}}{dt} = \frac{6}{\Gamma_c} Q_T Cp_c (T_{c_4} - T_{c_0}).$$

- Energy balance in the reactor:

$$\frac{dT_r}{dt} = \frac{1}{\Gamma_r} q_{r\times n}.$$

The parameters of the model have to be characterised experimentally. The experiments realised to characterise the parameters are described in Zaldıvar et al. (1996). The conclusions derived from these experiments are as follows:

(a) *Power introduced by the recirculation pump* (q_p):

$$q_p(\text{kW}) = 1.38 \exp\left(\frac{352.6}{T_e}\right).$$

(b) *Flow circulating in the main loop* (Q_e): The flow is measured by a mass flow meter using coriolis force. The physical model presented in this section is built up assuming the heat transfer fluid mass flow has constant value equal to 197 kg/s. This assumption prevents model from fluctuating when the control valves are opened or closed, as it will be seen subsequently.

(c) *Control actions* ($Q_c\, y\, q_h$): They are calculated according to the control system output as shown in Fig. 6.

(d) *Flow circulating in the cryostat* (Q_k): A value 0.9 kg/s was measured by a coriolis type flow meter. Q_T is obtained by the addtion of Q_c and Q_k.

(e) *Heat losses in the main loop* (q_L): Using data from heating/cooling experiments the following correlation was obtained:

$$q_L(\text{kW}) = (0.46 - 9.56\times 10^{-4} T_e)(T_e - T_a).$$

(f) *Thermal capacities* (Γ) *and time constants* (τ): Firstly, they have been approximated according to the total mass of considered subsystem. Afterwards, they were optimised using a experimental data set (empty reactor, $q_{r\times n} = 0.0$). The values obtained are shown in Table 1.

Table 1
Values of physical model parameters

Parameter	Value
τ_{e1}	1156 (s)
τ_{e2}	2504 (s)
τ_k	500 (s)
Γ_e	$605.35 + 128 T_e$ (kJ/K)
Γ_c	9191 (kJ/K)

3.3. Different NARMA models for the heat transfer fluid temperature

From the point of view of a dynamic process, the evolution over the time of the heat transfer fluid temperature in the jacket, T_e, is given by the output of a non-linear dynamic process, whose input variables are: reactor temperature, T_r, cooling temperature, T_c, ambient temperature, T_a, and the signal control u. The only manipulated input is the signal control and the rest of input variables can be seen as the output from other dynamic processes or environmental variables.

To determine the structure of NARMA models representing the dynamic behaviour of the heat transfer fluid temperature, the following considerations are made. Firstly, the lags in ambient and cooling temperature are considered equal to zero because both temperatures tend not to vary radically during the experiments. Moreover, the experiments have shown that the dynamic process governing the heat transfer fluid temperature is an integrative process. That means the jacket temperature at time $k+1$ depends on the jacket temperature at time k by a constant value practically equal to one. The first approximation of values n_u, n_e, n_r representing the length of discrete sequences for input signal, heat transfer and reactor temperatures, respectively was provided by the empirical knowledge about the process, resulting in $0 \le n_u \le 20$, $1 \le n_e \le 2$, $0 \le n_r \le 1$. The delay of the process was estimated around $3 \le d \le 6$. Thus, considering $d = 3$, $n_u = 20$, $n_e = 2$, $n_r = 1$ the following NARMA model may explain the dynamic behaviour of the process, which is called in this work as *Structure* 1:

$$T_e(k+1) = T_e(k) + F(u(k-3), \ldots, u(k-23), T_r(k), T_r(k-1), T_c(k), T_a(k), T_e(k-1), T_e(k-2)). \quad (20)$$

The model given by Eq. (20) has a large number of variables. It is convenient to simplify this model and remove the least influential variables. In Galván (1998) it is a proposed method in order to get a simplified model. According to this study, the following NARMA

model, named *Structure* 2 is chosen:

$$T_e(k+1) = T_e(k) + F(u(k-5), \ldots, u(k-12),$$
$$u(k-15), u(k-18), u(k-21), u(k-23),$$
$$T_e(k-1) - T_r(k-1), T_e(k-2) - T_r(k-2)).$$

As it is observed, the number of input variables is reduced from 27 to 14. It should be noticed that $T_e \quad T_r$ is used instead of T_e and T_r in order to guarantee good generalisation properties of neural models as it was pointed out in Zaldívar et al. (1992).

4. Experimental results

In this section, the experimental results obtained with parallels neural models studied in this work are presented. The functions F appearing in Eqs. (20) and (21) are approximated by neural networks. In order to verify the results presented in Section 2.5, firstly those functions are approximated using the multilayer feed-forward neural network and the capability of respective parallel models to simulate the dynamic behaviour of the process is evaluated. Secondly, the PNNARMA network is used to generate the parallel models. Finally, the proposed model is compared against the physical model presented in Section 3.2.

4.1. Experimental test cases

To build models describing the behaviour of the heat transfer fluid temperature, two different data sets, summarised in Table 2, have been obtained. That data sets are used to estimate the unknown parameters of models and to test their ability to track the dynamic behaviour of the heat transfer fluid temperature. These data sets have been obtained by direct manipulation of the control signal u from the second workstation and, hence, without the intervention of the control system. The temperatures of the reactor T_r, inlet and outlet jacket T_e, ambient T_a and coolant reservoir T_c, at three different points, have been measured and recorded. The sampling period time was 10 s.

The first set, *Data* 1, has been carried out at ambient temperature and empty reactor, i.e. no heat transfer between the reactor and jacket. The second, *Data* 2, has been performed with 80 l of water inside the reactor, i.e. heat transfer effects between reactor and jacket.

Table 2
Experimental data sets

	Reactor	Stirrer (rpm)	T_e(initial) (°C)
Data 1	Empty	0	20
Data 2	80 l water	200	20

4.2. Approximating F by multilayer feed-forward networks

Structure 1 Eq. (20) has been approximated by a multilayer feed-forward network with 27, 28 and 1 units in the input, hidden and output layers, respectively. For *Structure* 2 Eq. (21), the architecture is composed of 14 input units, 20 hidden units and 1 output units. Both feed-forward neural networks have been trained over the training set (*Data* 1) using the static backpropagation algorithm and learning rate varying from 0.1 down to 0.001. In this model the convergence is reached after 6000–7000 learning cycles.

Once the parameters of the series–parallel models have been estimated, they are used to build up the classical parallel models. Eq. (4) and the prediction errors have been evaluated to measure the capability of both structures *Structure* 1 and 2 to act as process simulator (see Table 3). Fig. 7 shows the dynamic behaviour of the heat transfer fluid temperature predicted by each structure. As can be observed, when the *Structure* 1 is approximated by a multilayer feed-forward network the parallel model does not provide appropriate approximations of the dynamic process, because errors at same instants are propagated to the rest of the prediction. The performance of the *Structure* 2 when is approximated by a multilayer feed-forward network could be considered adequate.

4.3. Approximating F by the PNNARMA network

Using 25 external inputs, 2 context neurones and 28 hidden neurones, the functional F of *Structure* 1 Eq. (20) has been approximated using the PNNARMA network. In order to approximate *Structure* 2 Eq. (21) by the PNNARMA network 12 external input, 2 context neurones and 20 hidden layers have been used. Since the internal structure of PNNARMA networks is identical to the respective series–parallel models, the parameters of PNNARMA networks have been initialised with parameters obtained after 2000 learning cycles over the respective series–parallel models, that is, training the multilayer feed-forward networks. Subsequently, the training of the PNNARMA networks is carried out according to the learning rule proposed in this work, Eq. (11), and learning rate fixed to 0.00001. After

Table 3
Prediction errors of different parallel models

	Structure 1	*Structure* 2
Approximating F with multilayer feed forward network	359.84	9.81
Approximating F with PNNARMA network	3.25	2.48

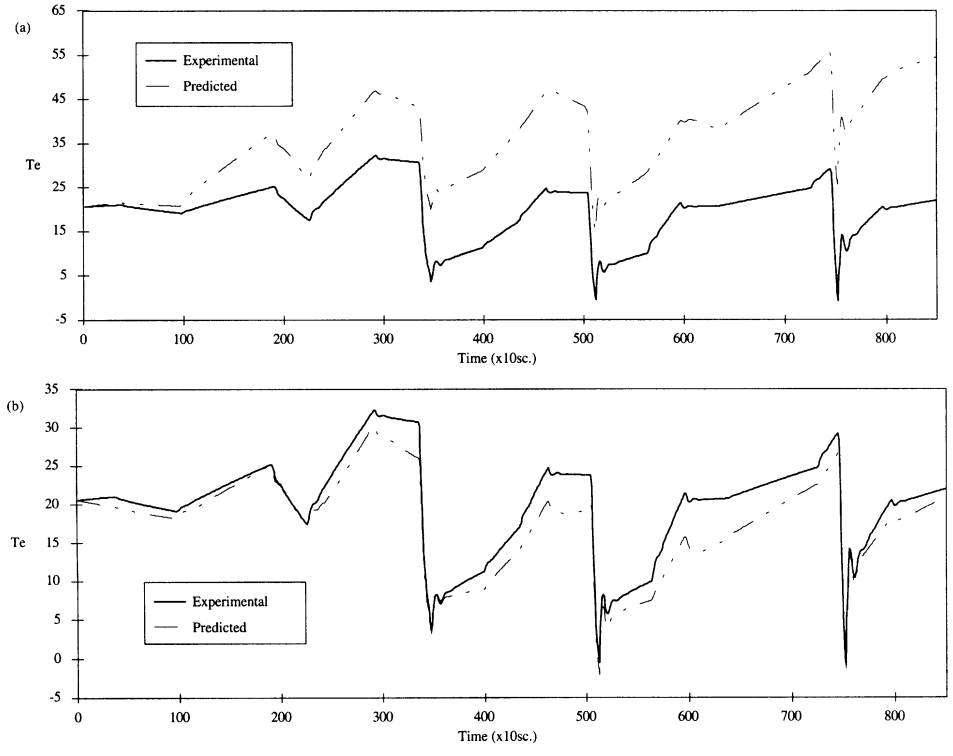

Fig. 7. Heat transfer fluid temperature time profile provided by the parallel model based on feed forward neural network. (a) *Structure* 1 (b) *Structure* 2.

1000 learning cycles, the convergence is reached. The prediction errors, Eq. (19), after the training are shown in Table 3.

As it is possible to observe in Table 3, the errors obtained by the PNNARMA models are smaller than the errors committed by the parallel models build up with the feed-forward neural network. For the *Structure* 1 the superiority of the recurrent network is clearly appreciated. When the *Structure* 2 is used to approximate the heat transfer fluid, the classical parallel model provide an appropriate error, but even in that cases the parallel model proposed in this work has improved the prediction error.

The evolution of the heat transfer fluid temperature provided by the PNNARMA models is show in Fig. 8. In contrast to the results shown in Fig. 7, the parallel models based on the PNNARMA network provide appropriate predictions of the heat transfer fluid temperature. The errors committed at same instants by the classical parallel models (see Fig. 7a) are not propagated when the PNNARMA models are used, because they have been trained to act as process simulator.

4.4. PNNARMA models versus first-principle model

The PNNARMA models obtained in the previous subsection has been compared against the phenomenological or first-principle model describing the dynamic behaviour of the heating/cooling circuits in the reactor (Section 3.2). In order to compare the different models, the prediction errors, Eq. (19), has been evaluated for the phenomenological model. In this case, $y(k+1)$ is the heat transfer fluid temperature at discrete time $k+1$ and $y_p(k+1)$ is the predicted temperature after integrating the physical model using the Runge–Kutta method. The error for the experimental data set (*Data* 1), which has been used to estimate the thermal capacities (Γ) and time constants (τ) in the model, is shown in Table 4. In this table, the prediction errors obtained by the parallel models proposed in this work over the training data (*Data* 1) are also shown. As it is possible to observe, the PNNARMA model can be seen as an alternative to first-principle model because they have similar predictive capabilities.

The simulated temperature–time profile for the jacket during the experiment *Data* 1 is shown in Fig. 9. In contrast to the PNNARMA model (see Fig. 8), it notice that the physical model cannot explain the magnitude of the temperature fluctuations in the heat transfer fluid when the control valve is opened to introduce cold fluid in the main loop; this is due to the fact that the flow circulating in the main loop (Q_e), which is considered constant for the simulation, experiments fluctuations when the control valve to the cooling circuit is opened (Galván, 1998).

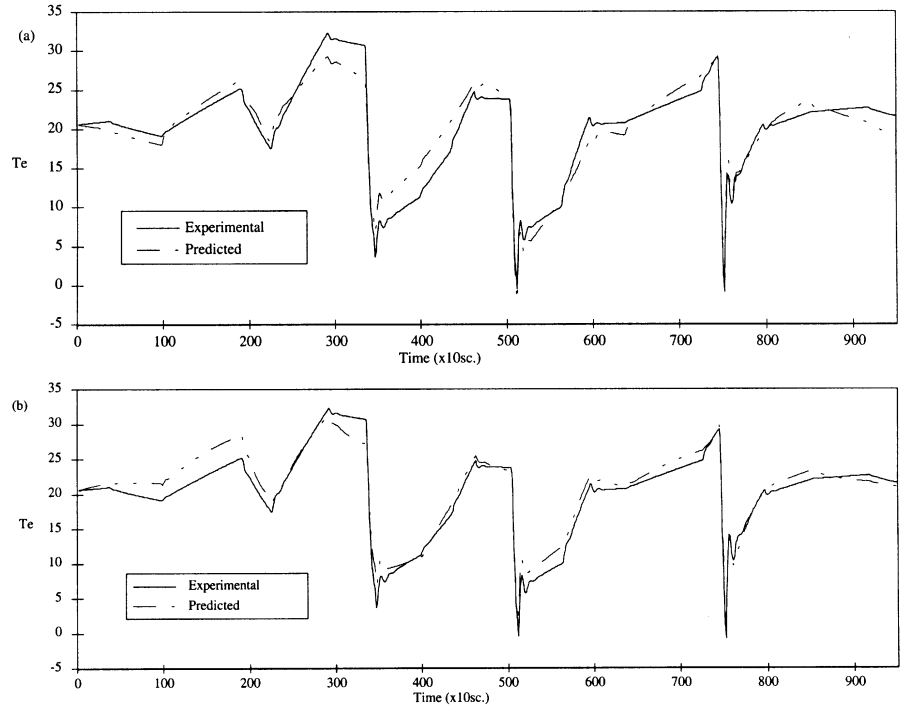

Fig. 8. Heat transfer fluid temperature time profile provided by parallel model based on PNNARMA network (a) *Structure* 1 (b) *Structure* 2.

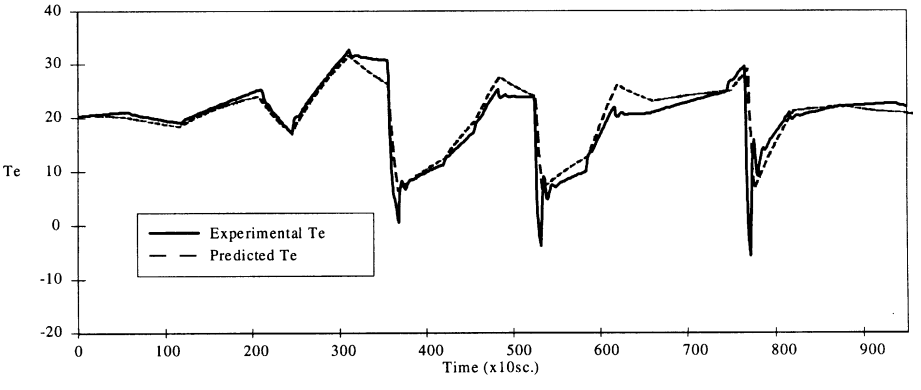

Fig. 9. Heat transfer fluid temperature time profile provided by first principle model over *Data* 1.

Table 4
Prediction errors: PNNARMA model versus first principle model

	Training data: *Data* 1	Test data: *Data* 2
PNNARMA model: structure 1	3.25	2.86
PNNARMA model: structure 2	2.48	0.77
First principle model	4.99	1.08

The generalisation (extrapolation) properties of models have been also measured. Now, the prediction errors for the different models have been evaluated over the data set (*Data* 2) and the results are shown in Table 4. As can be observed in Table 4, the generalisation properties of the *Structure* 1 are poorer than of the *Structure* 2. This is due to the fact that context neurones in the second structure memorise the differences T_e T_r instead to T_e and T_r as in the first structure, which allow to obtain better generalisation properties (Zaldıvar et al., 1992). Compared with the first-principle model, the performance is very similar. The jacket temperature profiles over the test data (*Data* 2) for the different models are shown in Fig. 10. Also, the temperature fluctuations in the heat transfer fluid cannot be predicted by the physical model, whereas the PNNARMA model can do it.

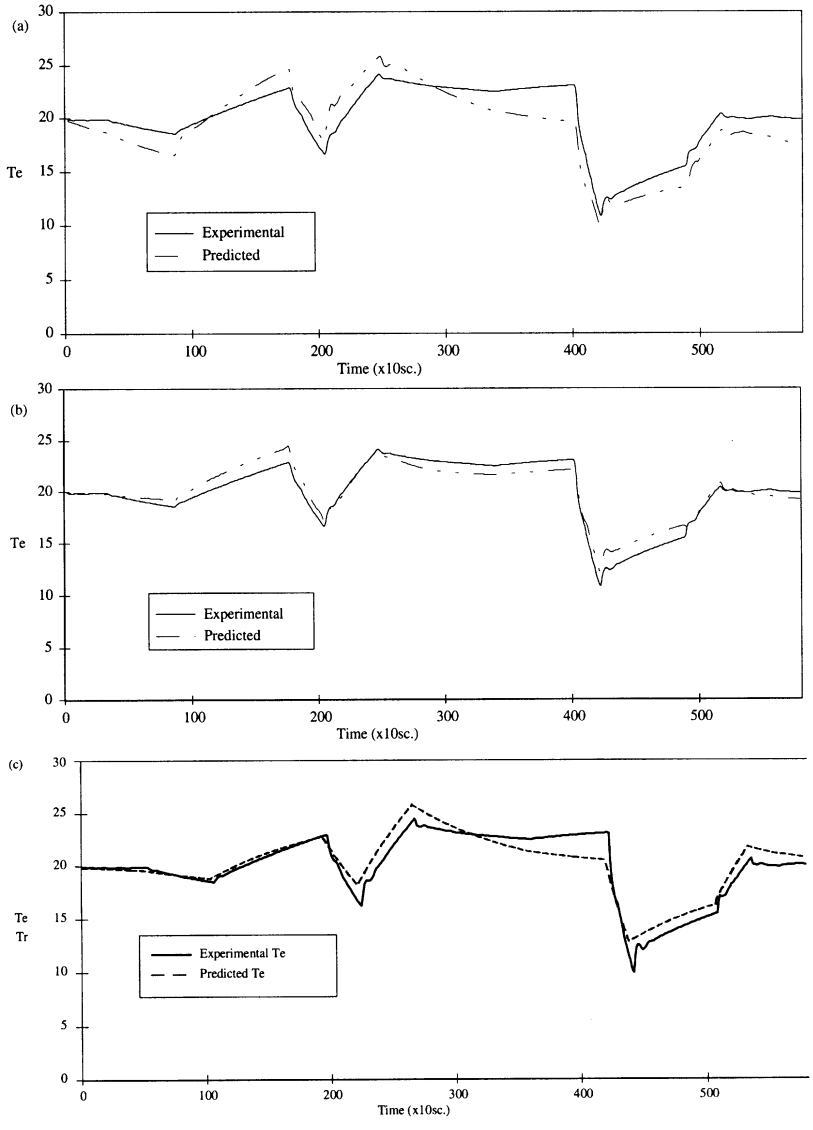

Fig. 10. Heat transfer fluid temperature time profile over the test data *Data* 2. (a) PNNARMA model: *Structure* 1 (b) PNNARMA model: *Structure* 2 (c) first principle model.

5. Conclusions

Phenomenological or first-principle models are extremely expensive to build up because the parameters characterisation of the model involve the realisation of experiments. Moreover, in some cases, they may not be suitable in real-time applications because the integration time may be higher than the sampling time when a large number of differential equations are requested to explain the dynamic behaviour of the process. On the other hand, models capable of simulating the non-linear behaviour of real processes are essential for numerous control applications.

The results presented in this work show that neural NARMA models can be used as suitable alternatives to first-principle (phenomenological) models if they are built up in an appropriate way. When the objective is to generate process simulators, series–parallel models cannot be used because they require tapped delay line of measured process output to approximate the current output. In this case, parallel models are the adequate solution, since no information about the past measured process output is required.

In Section 2.5, it is shown that the performance of the parallel model given by Eq. (4) depends on the capability of multilayer feed-forward neural networks to approximate perturbations of input patterns. Hence, if errors occur in the series–parallel model for some pattern, the capability of the parallel model given by Eq. (4) to simulate the dynamic process behaviour can be compromised (see Table 3 and Fig. 7). In order to guarantee an adequate approximation, the parameters

or weights determining the parallel model cannot remain fixed, they have to be estimated to minimise the prediction error. This implies that the PNNARMA network has to be used to approximate the functional determining the NARMA model, Eq. (13). Thus, the best simulator of dynamic process belonging to the class of NARMA model is guaranteed (see Table 3 and Fig. 8).

Furthermore, process representations arisen from the PNNARMA model (Figs. 8 and 10) are similar to representations obtained by first-principle models (Figs. 9 and 10c), whereas predictions provided by the parallel model whose parameters have been identified to minimise the identification error are not adequate. Moreover, the parallel structure proposed in this work reach the minimal point of the prediction error in a smaller number of learning iterations, which is an interesting propriety, principally, when the identification of models has to be carried out in real time.

When NARMA models are used for approximation purposes, a crucial aspect in the design is the appropriate selection of the input variables. This can be observed in the extremely different results achieved in this work when feed-forward neural network are used as a NARMA model to approximate the heat transfer fluid temperature in the jacket of the reactor (Table 3). With *Structure* 1 the error is 359.84, and it is reduced to 9.81 when the differences between heat transfer fluid temperature and the reactor temperature have been included and some intermediate periods of the control signal have been removed (*Structure* 2).

Additionally, the selection of the input variables is a task that requires a lot of expert knowledge of the problem, and it is difficult to automate. PNNARMA model is much less sensitive to the input selected. The errors achieved in the transfer fluid temperature predictions are, in this case, similar in both structures (3.25 for *Structure* 1 and 2.48 for *Structure* 2, Table 3). It is appreciable how with PNNARMA modelling the selection of appropriate inputs is not a crucial step in the approximation process, without losing efficiency in the results. Even considering simple inputs (*Structure* 1), the results with PNNARMA are better than feed-forward network using more elaborated input data.

References

Bhat, N., McAvoy, T.J., 1990. Use of neural nets for dynamic modelling and control of chemical process systems. Computers and Chemical Engineering 14, 573 583.

Chen, S., Billings, S.A., 1992. Neural networks for non linear dynamic systems modelling and identification. International Journal of Control 56, 319 346.

Choi, J.Y, Vanlandingham, H.F., Bingulac, S., 1996. A constructive approach for non linear system identification using multilayer perceptrons. IEEE Transactions on Systems, Man and Cybernetics 26 (2), 307 312.

Cohen, B., Saad, D., Maram, E., 1997. Efficient training of recurrent neural network with time delays. Neural Networks 10 (1), 51 59.

Cybenko, G., 1989. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals, and System 2, 303 314.

Galván, I.M., 1998. Nuevos modelos de neuronas aritificiales para simulación y control de sistemas dinámicos. Ph.D. Thesis. Facultad de Informática, Universidad Politéctica de Madrid, Spain.

Hernández, H., Zaldívar, J.M., 1990. The JRC FIRES project for investigations on runaway reactions. Proceedings of Heat Transfer and Major Technological Hazards, Eurotherm Seminar no. 14, pp. 13 23.

Hernández, H., Zaldívar, J.M., Barcons, C., 1993. Development of a mathematical model and a numerical simulator for the analysis and optimization of batch reactors. Computers and Chemical Engineer ing 17S, 45 50.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences U.S.A. 79, 2554 2558.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feed forward networks are universal approximators. Neural Networks 2, 359 366.

Jordan, M.I., 1989. Generic constraints on underspecified target trajectories. IJCNN Proceedings, IEE, New York, June.

Kosmatopoulos, E.B., Polycarpou, M.M., Chistodoulou Ioannou, P.A., 1995. High order neural network structures for identification of dynamical systems. IEEE Transactions on Neural Networks 6, 422 431.

Leontaritis, I.J., Billings, S.A., 1985. Input output parametric models for non linear systems. Part I: deterministic non linear systems. International Journal of Control 41, 303 328.

Levin, A., Narendra, K.S., 1995. Identification using feed forward networks. Neural Computation 7, 349 357.

Levin, A., Narendra, K., 1996. Control of non linear dynamical systems using neural networks. Part II: observability, identification and control. IEEE Transactions on Neural Networks 7 (1), 30 42.

Lu, S., Basan, T., 1998. Robust non linear system identification using neural networks models. IEEE Transactions on Neural Networks 9 (3), 407 429.

Moody, J., Darken, C.J., 1989. Fast learning in networks of locally tuned processing units. Neural Computation 1, 281 294.

Moody, J., Darken, C.J., 1998. Learning with localized receptive fields. In: Touretzky, Hinton, Sejnowski (Eds.), Proceedings of the 1998 Connectionist Models Summer School. Morgan Kaufmann Publish ers, Los Altos, CA.

Narendra, K.S., Mukhopadhyay, S., 1997. Adaptive control using neural networks and approximative models. IEEE Transactions on Neural Networks 8 (3), 475 485.

Narendra, K.S., Parthasarathy, K., 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1, 4 27.

Narendra, K.S., Parthasarathy, K., 1991. Gradient methods for the optimization of dynamical systems containing neural networks. IEEE Transactions on Neural Networks 2, 252 262.

Noriega, J.R., Wang, H., 1998. A direct adaptive neural network control for unknown non linear system and its application. IEEE Transactions on Neural Networks 9 (1), 27 34.

Parlos, A.G., Chong, K.T., Atiya, A.F., 1994. Application of recurrent multilayer perceptron in modelling complex process dynamics. IEEE Transactions on Neural Networks 5, 255 266.

Polycarpou, M.M., Ioannou, P.A., 1991. Identification and control of non linear systems using neural network models: design and stability analysis. Technical Report 91 09 01, September.

Rumelhart, D., Hinton, G., Williams, R.J., 1986. Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (Eds.), Parallel Distributed Processing. MIT Press, Cambridge.

Srinivasan, B., Prasad, U.R., Rao, N.J., 1994. Back propagation through adjoints for the identification of non-linear dynamic systems using recurrent neural models. IEEE Trans. on Neural Networks 5, 213–228.

Stage, P., Sendhoff, B., 1997. An extended Elman net for modeling time series. International Conference on Artificial Neural Networks.

Suykens, J.A.K., Bersini, H., 1996. Neural control theory: an overview. Journal A 37 (3), 4–10.

Wan, E.A., Beaufays, F., 1996. Diagrammatic derivation of gradient algorithms for neural networks. Neural Computation 8 (1), 182–201.

Yonhg, Y., Nikolaou, M., 1993. Dynamic process modelling with recurrent neural networks. A.I.Ch.E. Journal 39, 1654–1667.

Zaldívar, J.M., Hernández, H., Barcon, C., 1996. Development of a mathematical model and a simulator for the analysis and optimisation of batch reactors: experimental model characterisation using a reaction calorimeter. Thermochimica Acta 289, 267–302.

Zaldívar, J.M., Hernández, H., Nieman, H., Molga, E., Bassani, C., 1993. The FIRES project: experimental study of thermal runaway due to agitation problems during toluene nitration. Journal of Loss Prevention Process in Industry 6, 319–326.

Zaldívar, J.M., Panetsos, F., Hernández, H., 1992. Control of batch reactors using neural networks. Chemical Engineering Process 31, 173–180.