Application of recurrent neural networks in batch reactors Part I. NARMA modelling of the dynamic behaviour of the heat transfer fluid temperature

I.M. Galván [a], J.M. Zaldívar [b,*]

[a] *Carlos III University of Madrid, Department of Computer Science, Butarque 15, 28911, Leganés, Madrid, Spain*
[b] *European Commission, Joint Research Center, Institute for Systems, Informatics and Safety; Systems Modelling and Assessment Unit, TP250, 21020, Ispra, VA, Italy*

Abstract

This paper is focused on the development of nonlinear models, using artificial neural networks, able to provide appropriate predictions when acting as process simulators. The dynamic behaviour of the heat transfer fluid temperature in a jacketed chemical reactor has been selected as a case study. Different structures of NARMA (Non-linear ARMA) models have been studied. The experimental results have allowed to carry out a comparison between the different neural approaches and a first-principles model. The best neural results are obtained using a parallel model structure based on a recurrent neural network architecture, which guarantees better dynamic approximations than currently employed neural models. The results suggest that parallel models built up with recurrent networks can be seen as an alternative to phenomenological models for simulating the dynamic behaviour of the heating/cooling circuits which change from batch installation to installation. © 1997 Elsevier Science S.A.

Keywords: Neural networks; Batch reactors; Mathematical modelling; Systems identification

1. Introduction

The rapid development of the chemical industry in the past decades has increased the complexity of chemical plants, the diversity of products and the number of processes. This has produced a parallel increase of batch reactors, which due to their versatility allow the production of special chemicals with very good yields in small amounts when compared to those of continuous processes and permit a rapid change from one process to other with minor modifications in the installation.

However, 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. Furthermore, the fact that in a batch cycle there is no steady state implies that the operator must perform continuous corrections to control the process.

It has been argued that due to the small production levels, time constraints and the enormous variety of processes, the understanding of the reactor dynamic behaviour is usually not economically justified. However, we believe that there are a considerable number of advantages, the most important being the optimization of such processes.

Mathematical modelling and dynamic simulation applied to these processes have proven to be useful tools to obtain optimum yields [1] and to carry out thermal hazard assessment [2,3]. However, this type of applications has been performed mainly for specific reactions in which the thermo-kinetic model was well-known. In the last years, due to the fast development of digital computers, on-line identification applications have appeared [4–6]. These applications are based on the fact that the possibilities to optimize and to control this type of reactors are limited, because only few state variables or parameters can be measured on-line and amongst these, only a few may be incorporated in the control or/and decision chains. However, if the state

* Corresponding author. E-mail: jose.zaldivar-comenges@jrc.it

variables and fundamental parameters can be inferred, these new data can be used to perform predictive calculations about the future behaviour of the reactor, to detect in advance dangerous situations, to improve the control algorithms, and to assist the plant operators in their decisions about the correct measures to be adopted under certain situations. Therefore, the application of mathematical modelling, estimation techniques and numerical simulation is of great interest in this field [7].

In some cases, phenomenological or first-principle models, which are generated according to the physical laws governing the dynamic evolution of the system, are often not available and in many cases they are time consuming and extremely expensive to build up. Hence, the development of efficient methods for simulating nonlinear dynamic systems is of great value.

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 [8]. Furthermore, the heating/cooling circuits will change from installation to installation and hence a new mathematical model has to be developed. On the other hand, once the thermokinetic parameters of the chemical reaction have been found, the modelling of the internal reactor dynamics may be carried out in a more standard way. There are an extensive number of works in the literature dealing with mass transfer and heat transfer scaling-up procedures [9].

The present study is concerned with the generation of nonlinear models capable of acting, in an efficient fashion, as process simulators. That is, given the initial state of the system $\mathbf{y}(0)$ and the input signal $\mathbf{u}(k)$, the model should be able to predict the outputs of the process over a large number of sampling times. Situations where models have to act as process simulator are often presented in control applications. The use of neural models for temperature control of the heat transfer fluid temperature will be covered in the second part of this work.

NARMA (Non-linear ARMA) models provide a unified representation for a wide class of nonlinear systems [10]. In a NARMA description the system is modelled in terms of a nonlinear 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 approximation functions which can be used to that end. These include polynomials, trigonometric series, orthogonal functions, splines, and artificial neural networks (ANNs) [11,12].

The application of ANNs to nonlinear dynamic process modelling problem has been dominated by the static multilayer neural networks [13–15]. In these structures the processing of input patterns does not depend upon the order of presentation during the training. Moreover, the creation of cycles or loops among neurons is forbidden. Thus, the representation and processing of temporal information is not an intrinsic capability of these architectures, even if it is always possible to use static structures to encode temporal information. The approach consists in forming tapped delay line representations of applied inputs and measured outputs. Thus, they can be used to generate NARMA models which are normally referred to as *series-parallel models* [16–19]. However, these models cannot act as process simulators because a discrete sequence of delayed measured output of the process is required.

Narendra and Pathasarthy [20] 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 normally 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 feedforward network, the approximation of the dynamic behaviour of the process provided by this parallel structure could not be suitable.

An alternative to built up neural nonlinear simulators is the use of recurrent neural networks (RNN). RNN are characterised by the presence of feedback connections between the neurons of the same layer, including the originating node itself, and/or connections to nodes of preceding layers. It has been shown that the capabilities, for temporal representation of recurrent networks, are considerably grater than those of purely static networks [21–25]. Furthermore, models built up with recurrent structures are capable of acting as process simulators because any discrete sequence of delayed input and output process must be considered to represent temporal information. However, RNN are significantly more complex than feedforward 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 algorithms to guarantee the convergence of the network weights.

The neural network architecture studied in this paper to build up nonlinear models able of acting as process simulators, is a particular architecture of RNN, which is characterised by its low degree of complexity, i.e. equivalent to the complexity degree of the multilayer feedforward network.

The paper is organized as follows. The experimental set-up of the pilot plant reactor, the control configuration as well as the first principles model are presented in the second section. The RNN architecture developed in this work as well as the dynamic backpropagation learning algorithm is described in the third section. The fourth section deals with the construction of neural NARMA models. A series–parallel model identification and two different parallel models structures are presented. In the fifth section the parallel structures are tested and compared with the experimental data. In addition, their performances are compared with the phenomenological model. The results suggest that parallel models built up with recurrent networks can be seen as an alternative to phenomenological models.

2. Experimental configuration and first principles model

To test the neural NARMA models, the modelling of a real process which describes the dynamic behaviour of the heat transfer fluid temperature circulating in the jacket of a batch reactor has been selected as a case study. In this section, some issues concerning to the reactor and the heating/cooling circuits are presented. Furthermore, to compare the neural simulation results a phenomenological model has been developed and their parameters characterized.

2.1. Reactor and heating/cooling circuits description

The FIRES (Facility for Investigation Runaway Events Safely) reactor [26,27] 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 overall experimental set-up is presented in Fig. 1 and the main specifications for the reactor, jacket and stirrer are listed in Table 1.

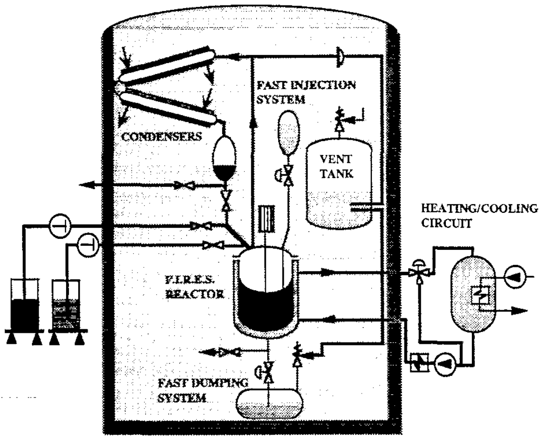

Fig. 1. Schematic layout of the FIRES facility.

Operation in various thermal modes, i.e. isothermal, isoperibolic or adiabatic, is achieved by appropriate control of the temperature of the heat transfer fluid.

The cooling/heating systems, see Fig. 2, consist of two loops in which a 50/50 wt% glycol–water mixture circulates at high speed, i.e. 12 m^3 h^{-1}. The main circuit is connected to the reactor jacket and contains a heat source up to 40 kW of power. The secondary circuit provides cold fluid to the main loop and has 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°C and it can be used for providing full cooling in emergency situations through a bypass valve.

2.2. Description of the control configuration

The control configuration is summarised in Fig. 2. When the process is carried out in isothermal conditions, the reactor temperature is maintained at its desired value, Tr^{sp}, by adjusting the temperature set-point for the glycol-water mixture recirculating through the reactor jacket, Te^{sp}. This is accomplished by the master controller. The temperature of the heat transfer fluid

Table 1
Main specifications of the reactor, jacket and stirrer

Reactor		Jacket heat exchange		Stirrer	
Volume:	100 l	Max. exchange area:	0.9 m^2	Type:	Turbine
Diameter:	0.5 m	Max. pressure:	16 bar	Diameter:	0.23 m
Height:	0.5 m	Max. temperature:	525 K	Max. Speed:	300 rpm
Body material:	EH 21 HI			Baffles:	4
Thickness:	12.0 mm				
Lining:	Nucerite(3115), 1.5 mm				
Max. T:	525 K				
Max. P:	16 bar				

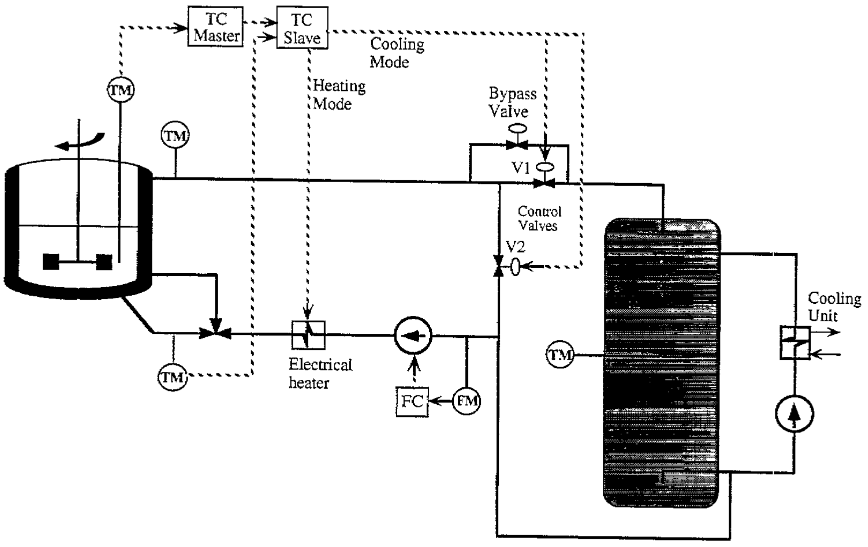

Fig. 2. Schematic layout of the heating/cooling circuits and cascade temperature controllers.

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 $-20°C$ enters to 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. 3a. 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. 3b as a function of the control action. As can be seen the control action is nonlinear in both cases.

2.3. Experimental test cases

Three different experimental data sets, summarised in Table 2, have been used to estimate the unknown parameters of the developed models (phenomenological and neural) and to test their ability to track the dynamic behaviour of the heat transfer fluid temperature. These data sets have been obtained by manipulating directly the control signal **u** and, hence, without the intervention of the control system. The temperatures of the reactor (Tr), inlet and outlet jacket (Te), ambient (Ta) and coolant reservoir (Tc), at three different points, have been measured and recorded. Furthermore, the flow rates of the heat transfer fluid have also been measured using a coriolis flow meter. The sampling period time was 10 s.

The first experiment, *Data* 1, was carried out at ambient temperature and empty reactor, i.e. no heat transfer between the reactor and jacket. This experiment has been used to adjust the unknown parameters in the neural and phenomenological models.

However, in the development of the first principle model additional information was required and, hence, characterization experiments [28] have been also used to estimate different values, e.g. power introduced by the pumps in the circuits, controller characteristics, heat

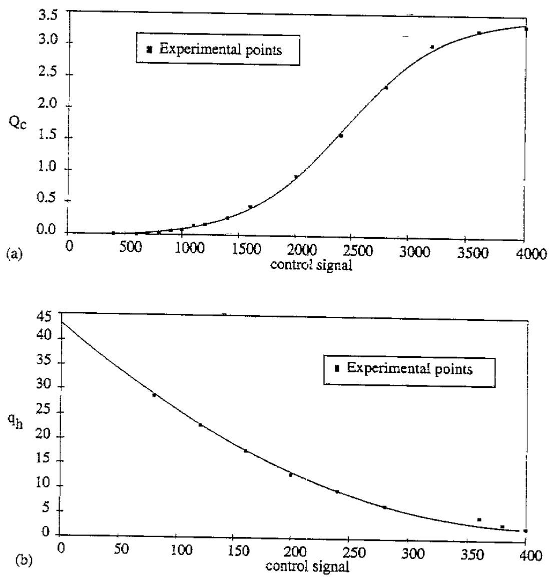

Fig. 3. Correlation between the control signal, **u**, and (a) Q_c; (b) q_h.

Table 2
Experimental data sets

	Reactor	Stirrer (rpm)	Te (initial) (°C)
*Data*1	Empty	0	20
*Data*2	80 l. water	200	20
*Data*3	80 l. water	300	60

losses, etc. Furthermore, the overall heat transfer coefficient, UA, was measured using heat flow calorimetry [29].

The second and third experiments, *Data*2 and *Data*3, were performed with 80 l of water inside the reactor, i.e. including heat transfer effects between reactor and jacket, and at different temperatures, i.e. different heat loses, and stirring speeds, to study the predictive capabilities in the different modelling approaches.

2.4. First principles model

To develop a phenomenological model, the heating/cooling circuits, see Fig. 2, were divided in different thermally homogeneous subsystems and separate energy balances were applied. The division was effectuated according to the characteristics of the system and after different tests. The subsystems identified are the following: the main loop Te, the insulation of the main loop Tp, and the cold reservoir Tc. 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, Cp, of the heat transfer fluid (50/50 wt% 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 [30].

   Energy balances in the subsystems allowed the time-profile of temperature to be written as follows:
6. Energy balance in the main loop:

$$\frac{d\mathrm{Te}}{dt}=\frac{1}{\Gamma_e}[q_{\mathrm{rxn}}+q_{\mathrm{h}}+q_{\mathrm{p}}-Q_{\mathrm{c}}\cdot \mathrm{Cp_e}(\mathrm{Te}-\mathrm{Tc}_0)] + \frac{\mathrm{Tp}-T}{\tau_{e_0}} \tag{1}$$

7. Energy balance in the isolation of the main loop:

$$\frac{d\mathrm{Tp}}{dt}=\frac{\mathrm{Te}-\mathrm{Tp}}{\tau_{e_2}}-\frac{q_{\mathrm{L}}}{\tau_k} \tag{2}$$

8. Energy balance in the cooling loop:

$$\frac{d\mathrm{Tc}_i}{dt}=\frac{6}{\Gamma_c}[Q_{\mathrm{c}}\cdot \mathrm{Cp_c}(\mathrm{Te}-\mathrm{Tc_i})+Q_k\cdot \mathrm{Cp_c}(\mathrm{Tk}-\mathrm{Tc}_i)] \tag{3}$$

$$\frac{d\mathrm{Tc}_1}{dt}=\frac{6}{\Gamma_c}Q_{\mathrm{T}}\cdot \mathrm{Cp_c}(\mathrm{Tc}_i-\mathrm{Tc}_1) \tag{4}$$

$$\frac{d\mathrm{Tc}_2}{dt}=\frac{6}{\Gamma_c}Q_{\mathrm{T}}\cdot \mathrm{Cp_c}(\mathrm{Tc}_1-\mathrm{Tc}_2) \tag{5}$$

$$\frac{d\mathrm{Tc}_3}{dt}=\frac{6}{\Gamma_c}Q_{\mathrm{T}}\cdot \mathrm{Cp_c}(\mathrm{Tc}_2-\mathrm{Tc}_3) \tag{6}$$

$$\frac{d\mathrm{Tc}_4}{dt}=\frac{6}{\Gamma_c}Q_{\mathrm{T}}\cdot \mathrm{Cp_c}(\mathrm{Tc}_3-\mathrm{Tc}_4) \tag{7}$$

$$\frac{d\mathrm{Tc}_0}{dt}=\frac{6}{\Gamma_c}Q_{\mathrm{T}}\cdot \mathrm{Cp_c}(\mathrm{Tc}_4-\mathrm{Tc}_0) \tag{8}$$

9. Energy balance in the reactor:

$$\frac{d\mathrm{Tr}}{dt}=\frac{1}{\Gamma_r}\cdot q_{\mathrm{rxn}} \tag{9}$$

2.4.1. Characterisation of model parameters

In order to characterise the model parameters, different experiments were performed [8]. The experiments have shown the following results:

1. *Power introduced by the recirculation pump (q_p):* The following correlation is used:

$$q_{\mathrm{p}}(\mathrm{kW})=1.38\exp\left(\frac{352.6}{\mathrm{Te}}\right) \tag{10}$$

2. *Flow circulating in the main loop (Q_e):* The flow is measured by a mass flow meter using coriolis force. Fig. 4 shows a typical operation. The main fluctuations are produced when opening the control valves that regulate the flow distribution between the main loop and the cold reservoir. 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^{-1}.

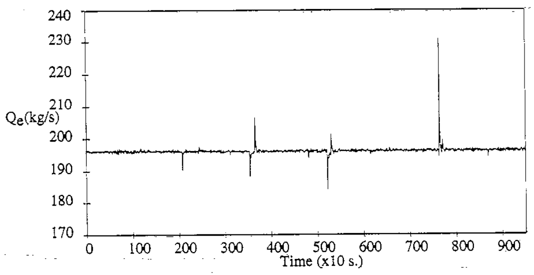

Fig. 4. Measured mass flow in the main loop during *Data*1.

Table 3
Values of physical model parameters

Parameter	Value
τ_{e1}	1156 (s)
τ_{e2}	2504 (s)
τ_k	500 (s)
Γ_e	605.35 + 1.28Te (kJ K^{-1})
Γ_c	9191 (kJ K^{-1})

3. *Control actions (Q_c and q_h):* They are calculated according to the control system output as shown in Fig. 4.
4. *Flow circulating in the cryostat (Q_k):* A value 0.9 kg s^{-1} was measured by a coriolis type flow meter. Q_T is obtained by the adding of Q_c and Q_k.
5. *Heat losses in the main loop (q_L):* Using data from heating/cooling experiments the following correlation was obtained:

$$q_L(\mathrm{kW}) = (0.46) - 9.56 \cdot 10^{-4}\mathrm{Te}) \cdot (\mathrm{Te} - \mathrm{Ta}) \quad (11)$$

6. *Thermal capacities (Γ) and time constants (τ):* Firstly, they were approximated according to the total mass of the considered subsystem. Afterwards, they were optimised using the experimental data set *Data*1 (empty reactor, $q_{rxn} = 0.0$). The values obtained are shown in Table 3. For the simulation of the experiments *Data*2 and *Data*3, $q_{rxn} = \mathrm{UA(Tr - Te)}$.

2.5. Comparison between experimental and simulated results

The experimental and simulated temperature–time profiles for the jacket and cooling circuits (Te and Tc$_2$) for the experiment *Data*1 are shown in Fig. 5a whereas in Fig. 5b and c the experimental and simulated reactor and jacket temperatures, for experiments *Data*2 and *Data*3, are presented. Notice that the 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, as can be seen in Fig. 4, 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.

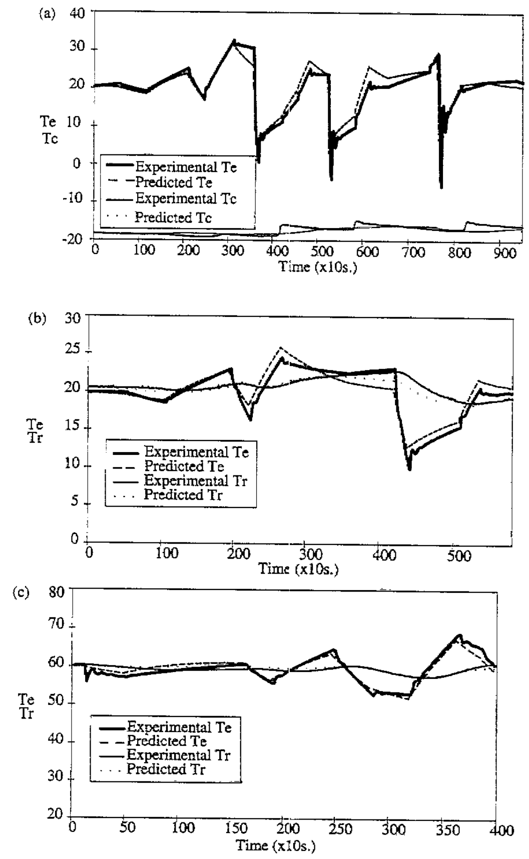

Fig. 5. Te, Tr, Tc time-profiles provided by first principle model. (a) *Data*1 (b) *Data*2 (c) *Data*3.

3. Recurrent neural network

3.1. Recurrent neural architecture

The RNN is constructed by starting from a multilayer feedforward neural network and by adding feedback connections from the output neuron to the input layer, as it is shown in Fig. 6. The neurons in the RNN are divided in the input, hidden and output layers. The input layer is formed by two groups of neurons. The first group acts as the input to the network receiving the input patterns from the external world, $I(k) = (I_1(k), \ldots, I_{n1}(k))$. The second group is formed by the context neurons which memorise the output of the network associated to patterns previously presented.

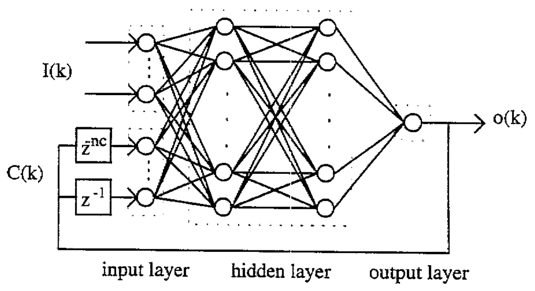

Fig. 6. Recurrent neural network architecture.

Introducing the vector to $C(k) = (C_1(k), ..., C_{n_c}(k))$ indicate the activation of context neurons, each component is calculated as,

$$C_i(k) = z^{-i}(o(k)), \qquad i = 1, ..., n_c \tag{12}$$

where $o(k)$ is the network answer to the kth input pattern and z^{-i} is an operator defined as: given $x(1)$, $x(2)$,..., $x(n)$,... a discrete sequence, the operator z^{-i} delays by i terms that sequence, this is, $z^{-i}(x(n) = x(n-i)$. Hence, when the kth input pattern is presented to the network each context node receives the output of the network associated to $k-1$th, $k-2$th,..., $k-n_c$th input patterns previously presented. Notice that when the variable k is referred to as the discrete time, the context neurons receive the output of the network delayed by $1, 2,..., n_c$ unit time steps.

Consider a RNN composed of L layers, with each layer containing n_l neurons, for $l = 1, ..., L$. Referring $s_j^l(k)$ to as the activation of the jth node located at the lth layer of the RNN for the kth input pattern, the equations describing the activation of the RNN can be expressed as follows:

$$s_j^1(k) = I_j(k), \qquad j = 1, .., n_1 \tag{13}$$

$$s_{j+n_1}^1(k) = \begin{cases} t(k) & k \le n_c \\ C_j(k) & k \ge n_c + 1 \end{cases},$$

$$j = 1, ..., n_l, \; l = 1, ...,L \tag{14}$$

with $t(k)$ the target output.

$$s_j^l(k) = \sigma\left(\sum_{i=1}^{n_l - 1} w_{ji}^l \cdot s_j^{l-1}(k) + \mu_j^l\right),$$

$$j = 1, ..., n_1 \; l = 1, ..., L \tag{15}$$

$$o(k) = s_1^L(k) \tag{16}$$

where

1. $I(k) = (I_1(k), ..., I_{n_1}(k))$ is the kth input pattern.
2. $W^l = (w_{ji}^l)$ is the matrix weight associated to the connections between the nodes located at $l-1$th layer and lth layer; $\Pi^l = \mu_j^l$ is the vector bias associated to the nodes of the lth layer. Both weights and bias are the adjustable parameters of the network.
3. $\sigma()$ is the activation function which is often chosen from smooth sigmoidal functions.

The RNN determines a correspondence from $\Re^{n_1}$ to $\Re$, denoted by $\tilde{F}_r(\cdot, C(k), W)$:

$$o(k) = \tilde{F}_r(I(k), C(k), W) \tag{17}$$

3.2. Learning algorithm: dynamic backpropagation

The learning procedure involves the determination of a vector W^* which optimises the following performance function $E(W)$:

$$E(W) = \frac{1}{2N_p} \cdot \sum_{k=1}^{N_p} (t(k) - o(k))^2 \tag{18}$$

where N_p is the number of input–output patterns. The weights are adjusted along the negative gradient direction of local errors as:

$$w(k) = w(k-1) + \alpha \cdot (t(k)) - o(k)) \cdot \frac{\partial o(k)}{\partial w},$$

$$\forall w, \forall k = 1, ..., N_p \tag{19}$$

Since recurrent connections appear in the RNN, the traditional backpropagation algorithm [13]—also referred in this study to as static backpropagation algorithm—cannot be directly used to calculate the term $\partial o(k)/\partial w$ in Eq. (19). The RNN output depends on earlier network activations and the total derivative concept must be applied[1]. The learning algorithm proposed in this work to carry out the training of the RNN is a reworked version of dynamic backpropagation algorithms developed in [20].

Considering that the vector $C(k)$ is composed by RNN output delayed, Eq. (12), and applying the total derivative concept in Eq. (17), it follows that:

$$\frac{\partial o(k)}{\partial w} = \frac{\partial \tilde{F}_r(I(k), C(k), W)}{\partial w} + \sum_{i=1}^{n_c} \frac{\partial \tilde{F}_r(I(k), C(k), W)}{\partial C_i(k)} \cdot \frac{\partial C_i(k)}{\partial w} \tag{20}$$

where, $\partial \tilde{F}_r(I(k), C(k), W)/\partial w$, $\partial \tilde{F}_r(I(k), C(k), W)/\partial C_i(k)$ are the partial derivatives of the RNN output with respect to the parameter w and to the context neurone $C_i(k)$, respectively.

Taking account that $C_i(k) = o(k\text{-}i)$, it follows:

$$\frac{\partial C_i(k)}{\partial w} = \frac{\partial o(k-i)}{\partial w} \tag{21}$$

Denoting $x(k) = \partial o(k)/\partial w$, the variation of the kth pattern RNN output can be obtained as the output at time k of a dynamic process governed by the following equation:

$$x(k) = \frac{\partial \tilde{F}_r(I(k), C(k), W)}{\partial w} + \sum_{i=1}^{n_{cr}} \frac{\partial \tilde{F}_r(I(k), C(k), W)}{\partial C_i(k)} \cdot x(k-i) \tag{22}$$

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

Both terms $\partial \tilde{F}_r(I(k), C(k), W)/\partial w$ and $\partial \tilde{F}_r(I(k), C(k), W)/\partial C_i(k)$ have to be calculated for each kth input pattern. Since the internal structure of the RNN is a feedforward network, those terms can be determined using the static backpropagation [31].

Therefore, at each time step, the output of the dynamic system given by Eq. (22) is calculated and the

[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 $\partial f/\partial x \cdot \partial x/\partial w + \partial f/\partial w$.

weights of the RNN are adjusted according to the following rule:

$$w^{(k)} = w^{(k-1)} + \alpha \cdot (t(k) - o(k)) \cdot x(k),$$
$$\forall w, \forall k = 1, ..., N_p \qquad (23)$$

3.2.1. Remarks about the learning rule given by Eq. (23)

It is worth pointing out that the computational effort required by the dynamic learning rule, see Eq. (22), could become time-consuming in practical applications, if the number of context neurons is high. In that case, it is possible to approximate the variations of the RNN output, $\partial o(k)/\partial w$, by the term $\partial \tilde{F}_r/\partial w$, see Eq. (20), obtaining the learning rule:

$$w^{(k)} = w^{(k-1)} + \alpha \cdot (t(k) - o(k)) \cdot \frac{\partial \tilde{F}_r}{\partial w},$$
$$\forall w, \forall k = 1, ..., N_p \qquad (24)$$

If the term $\partial \tilde{F}_r/\partial w$ in the Eq. (20) is dominant with respect to $\Sigma_{i=1}^{n} \partial \tilde{F}_r/\partial C_i(k) \cdot \partial C_i(k)/\partial w$, the learning rule given by Eq. (24) may perform quite satisfactorily.

3.2.2. Accelerating the algorithm convergence

Generally the learning of RNNs presents some problems concerning the algorithm convergence and the number of learning cycles to reach some minimal point. Due to the feedback connections, the training procedure may be difficult and arduous when the initial weights are set to random values. 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 feedforward network and the RNN used in this work, it is possible to obtain easily initial parameters for the last structure. The procedure is the following: since the learning training is carried out in supervised form, the target output for each input pattern, $t(k)$, is always available and, hence, its delayed values $t(k-1), t(k-2), ..., t(k-n_c)$. The weights obtained after the training of the multilayer feedforward network with input vector $(I_i(k), ..., I_{n_1}(k), t(k-1), ..., t(k-n_c))$ and the same number of layer and neurons in each layer as the RNN, can be used to initialise the recurrent structure.

4. Neural NARMA models design methodology

4.1. Structures of neural narma models

Consider a discrete nonlinear dynamic system governed by the following NARMA model:

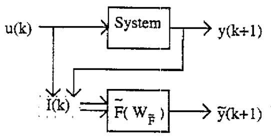

Fig. 7. Series-parallel model structure.

$$y(k+1) = F(y(k), ..., y(k-n_y), u(k-d)$$
$$\times, ..., u(k-d-n_u) \qquad (25)$$

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

Introducing the vector $I(k) = (y(k), ..., y(k-n_y), u(k-d), ..., u(k-d-n_u))$ as the kth network input pattern, the multilayer feedforward network can be used to approximate the map F obtaining the *series–parallel model*, see Fig. 7:

$$\tilde{y}(k+1) = \tilde{F}(y(k), ..., y(k-n_y), u(k-d)$$
$$\times, ..., u(k-d-n_u), W_{\tilde{F}}) \qquad (26)$$

When the model has to simulate the dynamic of the process, the sequence of measured values $y(k), ..., y(k-n_y)$ is not available and the series–parallel models identification given by Eq. (26) cannot be used. To solve this problem, Narendra and Pathasarthy [16] had proposed a different structure of neural model, referred to as *parallel model*, see Fig. 8a. It consists in replacing the measured values in $y(k), ..., y(k-n_y)$ Eq. (26) by the network predicted values $\tilde{y}(k), ..., \tilde{y}(k-n_y)$. Denoting $\tilde{y}p(k)$ as the output of this parallel model, it can be written as,

$$\tilde{y}_p(k+1) = \tilde{F}(\tilde{y}_p(k), ..., \tilde{y}_p(k-n_y), u(k-d)$$
$$\times, ..., u(k-d-n_u), W_{\tilde{F}}) \qquad (27)$$

The parameters of this neural models are fixed to the series–parallel model set of weights, $W_{\tilde{F}}$, Eq. (26).

On the other hand, the RNN described in Section 3 permits the generation of parallel models differing from models given by the Eq. (27), see Fig. 8b. Considering the vector $I(k) = (u(k-d), ..., u(k-n_u))$ as the input vector to the RNN and $n_y + 1$ context neurons, it is possible to built up the following parallel model:

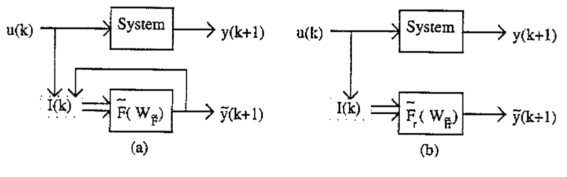

Fig. 8. Parallel models structure.

$$\begin{aligned}\tilde{y}_r(k+1) &= \tilde{F}_r(I(k), C(k), W_{\tilde{F}_r}) \\ &= \tilde{F}_r(u(k-d), ..., u(k-d-n_u), C_i(k) \\ &\quad \times ,..., C_{n_y}(k), W_{\tilde{F}_r})\end{aligned} \quad (28)$$

where $C_i(k) = \tilde{y}_r(k+1-i)$, $i = 1, ..., n_y + 1$.

4.2. Analysis of parallel models identification

To have models capable of simulating the dynamic behaviour of the process, parallel model structures have to be used. In the previous section two different parallel models have been presented, Eqs. (27) and (28). At this point, the immediate question that arises concerns the choice of the approach to be used.

Structures of both parallel models, Eqs. (27) and (28), are identical because the context neurons memorise the network output delayed by $1, 2, ..., n_y$ time steps. However, there exists an important difference between them: the way to determine the set of parameters. The weights of the parallel model given by Eq. (27) are fixed to the set $W_{\tilde{F}}$ obtained after the training of the series–parallel model. Thus, they have been updated using the training data set $\{I(k) = (y(k), ..., y(k-n_y), u(k-d), ..., u(k-d-n_u)), y(k+1)\}$, $\forall k = d, ..., N-1$. However, the set $W_{\tilde{F}_r}$ captures the map $\{I(k) = (u(k-d), ..., u(k-d-n_u)), y(k+1)\}$, $\forall k = d, ..., N-1$. This fact produces different behaviours of models when they are used to simulate the dynamic of the process.

In this section it will be shown that to guarantee the best process simulator the second parallel model, Eq. (28), should be used.

Let us assume for simplicity that $n_y = 0$ and $n_u = 0$. The models can be re-written as:

$$\tilde{y}(k+1) = \tilde{F}(y(k), u(k-d), W_{\tilde{F}}) \quad (29)$$

$$\tilde{y}_p(k+1) = \tilde{F}(\tilde{y}_p(k), u(k-d), W_{\tilde{F}}) \quad (30)$$

with $\tilde{y}_p(d) = y(d)$.

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

$$\tilde{y}(k+1) = y(k+1) + \epsilon_{k+1}, \qquad k = d, ..., N-1 \quad (31)$$

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

Definition: Given $I(k) = (y(k), u(k-d))$ the kth input pattern, $\tilde{y}(k+1)$ the answer of the multilayer feedforward neural network with parameters $W_{\tilde{F}}$ and ϵ a real number, the answer of that network for the input pattern $I_\epsilon(k) = (y(k) + \epsilon, u(k-d))$ is written as $\tilde{y}(k+1) + \delta(\epsilon)$ where $\delta(\epsilon)$ is a real number evaluating the capability of the neural network to approximate the perturbed input pattern $I_\epsilon(k)$.

Tacking account this definition and Eq. (31), the parallel model outputs given by Eq. (30) can be expressed as:

$$\tilde{y}_p(d+1) = \tilde{y}(d+1) = y(d+1) + \epsilon_{d+1} \quad (32)$$

$$\begin{aligned}\tilde{y}_p(k+1) &= \tilde{y}(k+1) \\ &\quad + \delta[\epsilon_k + \delta(\epsilon_{k-1} + \delta(\epsilon_{k-2} + ... + \delta(\epsilon_{d+1})))] \\ &= y(k+1) + \epsilon_{k+1} \\ &\quad + \delta[\epsilon_k + \delta(\epsilon_{k-1} + \delta(\epsilon_{k-2} + ... + \delta(\epsilon_{d+1})))] \\ k &= d+1, ..., N-1\end{aligned} \quad (33)$$

where $\delta(e_{d+1}), \delta(e_{d+2} + \delta(e_{d+1})), ..., \delta[e_k + \delta(e_{k-1} + \delta_{k-2} + ... + \delta(e_{d+1})...))]$ are real numbers measuring the capability of multilayer feedforward neural network with parameters to respond to perturbations of the input patterns $I(d+1), ..., I(k)$, respectively. The validity of Eq. (33) can be proved by induction.

From Eqs. (32) and (33) two different cases are deduced:

1. If ϵ_k are zero or closed to zero $\forall k = d+1, ..., N$ then $\delta(\epsilon_{d+1}), \delta(\epsilon_{d+2} + \delta(\epsilon_{d+1}))$ $\times , ..., \delta[\epsilon_N + \delta(\epsilon_{N-2} + ... + \delta(\epsilon_{d+1})))]$ will be close to zero because multilayer feedforward networks, generally, filter the noise. Thus, the approximations $\tilde{y}_p(k+1)$ $\forall k = d, ..., N-1$ could be considered suitable predictions of the dynamic system.
2. If there exists a natural number n such that the local error ϵ_n is distant to zero, then the real number $\delta[\epsilon_n + \delta(\epsilon_{n-1} + \delta(\epsilon_{n-2} + ... + \delta(\epsilon_{d+1})))]$ is also distant to zero because multilayer neural networks can not correctly approximate input patterns different from patterns used during the training procedure. Hence, $\tilde{y}_p(n+1)$ is not an appropriate approximation of the measured value $y(n+1)$, see Eq. (34). 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 $\tilde{y}_p(n+i)$ such that $\tilde{y}_p(n+i) \approx y(n+i)$ for $i = 2, ..., N-1$.

Hence, errors occur in the series–parallel model, Eq. (26) produce input patterns to the parallel model given by Eq. (27) differing from the training data set. Thus the capability of the parallel model to simulate the dynamic process behaviour can be compromised. for some pattern, the capability of the parallel model given by Eq. (27) to simulate the dynamic process behaviour can be compromised.

If the aim is to build-up models able to act as process simulators their parameters must be adjusted using the input patterns $(u(k-d), ..., u(k-d-n_u))$. This implies that the parallel model must be built up using the RNN model given by Eq. (28).

5. Comparative modelling of the heat transfer fluid temperature in a jacketed batch reactor using neural NARMA models

To reproduce the dynamic behaviour of the heat transfer fluid temperature in the jacket, a model of heating/cooling circuits is necessary. The evolution over the time of the heat transfer fluid temperature in the jacket, Te, is given by the output of a nonlinear dynamic process. From the point of view of input–output representation, it can be considered as a MISO (multiple input–single output) system whose input variables are: reactor temperature, Tr, cooling temperature, Tc, ambient temperature, Ta, 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 compare with the phenomenological model, the quadratic prediction errors, being $y(k+1)$ the heat transfer fluid temperature at discrete time $k+1$ and $\tilde{y}_p(k+1)$ the predicted temperature produced by the physical model, for each experimental data set are the following: *Data*1: $E^p = 4.99$, *Data*2: $E^p = 1.08$, *Data*3: $E^p = 0.42$.

5.1. Modelling phase

To determine the structure of NARMA models representing the dynamic behaviour of the heat transfer fluid temperature, the following considerations were made. Firstly, the lags in ambient and cooling temperature were 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. A 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 $0 \leq n_u \leq 20$, $1 \leq n_e \leq 2$, $0 \leq n_r \leq 1$. The delay of the process was estimated around $3 \leq d \leq 6$. 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 (model1):

$$\begin{aligned}\mathrm{Te}(k+1) = {}& \mathrm{Te}(k) \\ & + F(\mathrm{Te}(k-1), \mathrm{Te}(k-2), \mathrm{Tr}(k), \mathrm{Tc}(k) \\ & \times, \mathrm{Ta}(k), u(k-3), \ldots, u(k-23)) \end{aligned} \qquad (34)$$

The model given by Eq. (34) has a large number of variables. To simplify this model it was believed convenient to remove the least influential ones. The method used to eliminate some variables in Eq. (34) consists on training multilayer feedforward networks and using the information from their weights. Given an input node I_i to the network and being w_{ij}^1 $j = 1, \ldots, n_2$ the weights corresponding that input once the training has been finished, the value $m_i = 1/n_2 \cdot \Sigma_{j=1}^{n_2} |w_{ij}^1|$ is evaluated. It was observed during this study that m_i values can be used to evaluate the importance of input variables on $\Delta\mathrm{Te}(k+1)$. Hence, the input variables with lower m_i were removed.

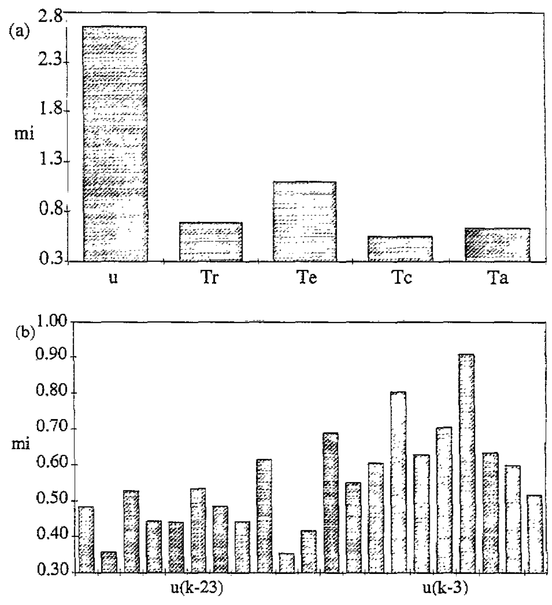

Fig. 9. m_i values associated to (a) u, Tr, Te, Tc, Ta (b) sequence $u(k-23), \ldots, u(k-3)$.

The procedure was divided in two parts. Firstly, a multilayer feedforward network with inputs u, Tr, Te, Tc, Ta was trained to determine their importance. Fig. 9a shows the m_i values associated to each input. As can be seen, u and Te are the most important input variables. Secondly, a multilayer feedforward network with inputs $u(k-23), \ldots, u(k-3)$ was also trained. The obtained m_i values are shown in Fig. 9b. According to that information, the following simplified NARMA model (model2) was chosen:

$$\begin{aligned}\mathrm{Te}(k+1) = {}& \mathrm{Te}(k) \\ & + F(\mathrm{Te}(k-1) - \mathrm{Tr}(k-1), \mathrm{Te}(k-2) \\ & - \mathrm{Tr}(k-2), u(k-5), \ldots, u(k-12) \\ & \times, u(k-15), u(k-18) \\ & \times, u(k-21, u(k-23)) \end{aligned} \qquad (35)$$

It should be notice that the use of Te–Tr instead of Te and Tr will guarantee good generalisation properties of neural models as it was pointed out in [32].

Once the identification phase of models is realised, it will be possible to observe that models given by Eqs. (34) and (35) can provide similar representations of the dynamic process. Thus, the use of the information provided by the m_i values to simplify the model given by Eq. (34), will be verified.

5.2. Identification phase

Once the lags in input variables have been determined, the functions F appearing in Eqs. (34) and (35) are approximated by neural networks. To verify the results presented in Section 3.2, firstly those functions are approximated using the multilayer feedforward neural network and the capability of respective parallel models, Eq. (27), to simulate the dynamic behaviour of the process is evaluated. Secondly, the RNN is used to generate parallel models.

5.2.1. Approximating F by multilayer feedforward networks

The series–parallel models identification adopt the following expression:

$$\tilde{\mathrm{Te}}(k+1) = \mathrm{Te}(k) + \tilde{F}(\mathrm{Te}(k-1), \mathrm{Te}(k-2), \mathrm{Tr}(k), \mathrm{Tr}(k-1) \times, \mathrm{Tc}(k), \mathrm{Ta}(k), u(k-3), ..., u(k-23) \times, W_{\tilde{F}}) \quad (36)$$

$$\tilde{\mathrm{Te}}(k+1) = \mathrm{Te}(k) + \tilde{F}(\mathrm{Te}(k-1) - \mathrm{Tr}(k-1), \mathrm{Te}(k-2) - \mathrm{Tr}(k-2), u(k-5), ..., u(k-12) \times, u(k-15), u(k-18), u(k-21) \times, u(k-23), W_{\tilde{F}}) \quad (37)$$

They have been trained over the training *Data*1 set using the static backpropagation algorithm and learning rule varying from 0.1 down to 0.001. After 6000 or 7000 learning cycles the convergence of learning procedure is reached. The identification errors as well as the architecture of multilayer feedforward networks and the number of adjustable parameters, are presented in Table 4.

The respective parallel structures can be written as:

Table 4
Approximating F with multilayer feedforward networks: *Data*1

	Architecture	Number of parameters	$E^{\mathrm{I}}(W_{\tilde{F}})$	$E^{\mathrm{P}}(W_{\tilde{F}})$
Model1	27–28–1	812	$1.9 \cdot 10^{-2}$	359.84
Model2	14–20–1	320	$1.5 \cdot 10^{-2}$	9.81

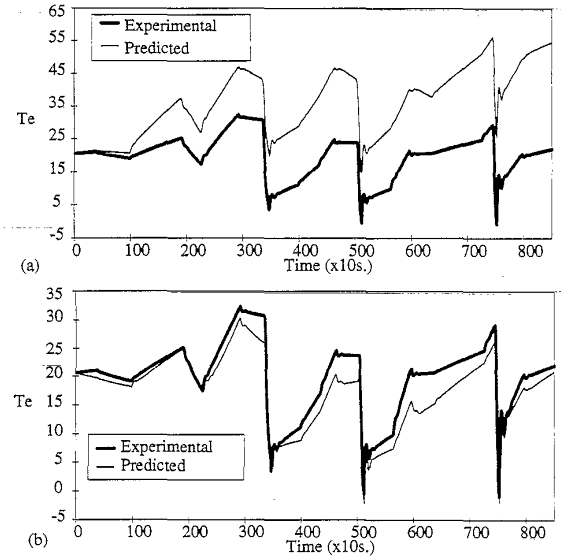

Fig. 10. Te time-profile provided by parallel model of parameters $W_{\tilde{F}}$ for *Data*1. (a) 1st model (b) 2nd model.

$$\tilde{\mathrm{Te}}(k+1) = \tilde{\mathrm{Te}}(k) + \tilde{F}(\tilde{\mathrm{Te}}(k-1), \tilde{\mathrm{Te}}(k-2), \mathrm{Tr}(k), \mathrm{Tr}(k-1) \times, \mathrm{Tc}(k), \mathrm{Ta}(k), u(u-3), ..., u(k-23), W_{\tilde{F}} \times \quad (38)$$

$$\tilde{\mathrm{Te}}(k+1) = \tilde{\mathrm{Te}}(k) + \tilde{F}(\tilde{\mathrm{Te}}(K-1) - tr(k-1), \tilde{\mathrm{Te}}(k-2) - \mathrm{Tr}(k-2), u(k-5),..., u(k-12) \times, u(k-15), u(k-18), u(k-21) \times, u(k-23), W_{\tilde{F}} \quad (39)$$

where $W_{\tilde{F}}$ denote the sets of weights previously obtained for the different models, Eqs. (36) and (37), respectively. The prediction errors obtained by the parallel models, Eqs. (38) and (39) are presented in Table 4. Figs. 10 and 11 shows the dynamic behaviour of the heat transfer fluid temperature predicted by each model when they are acting as process simulator. As can be observed, the first model does not provide an appropriate approximation of the dynamic process, while the performance of the second model could be considered adequate.

5.2.2. Approximation F by the RNN

Introducing the input vectors $I(k) = (\mathrm{Tr}(k), \mathrm{Tr}(k-1), \mathrm{Tc}(k), \mathrm{Ta}(k), u(k-3), ..., u(k-23))$ and $I(k) = (u(k-5), ..., u(k-12), u(k-18), u(u-21), u(k-23))$ respectively, and considering two context neurons, the RNN can be used to approximate the functions F appearing in Eqs. (34) and (35):

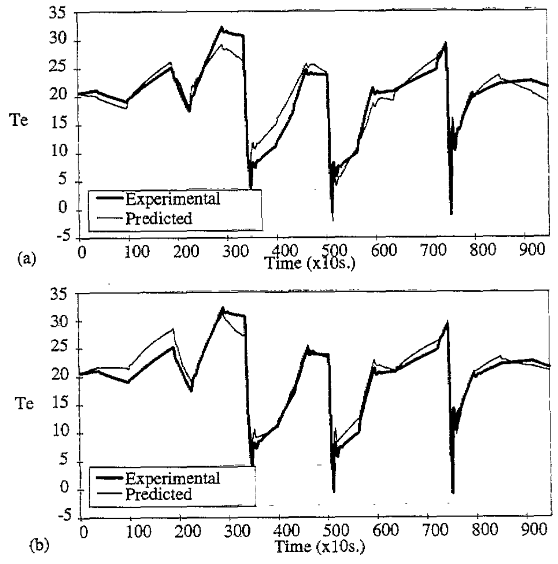

Fig. 11. Te time-profile provided by parallel model of parameters $W_{\tilde{F}}$ for *Data*1. (a) 1st model (b) 2nd model.

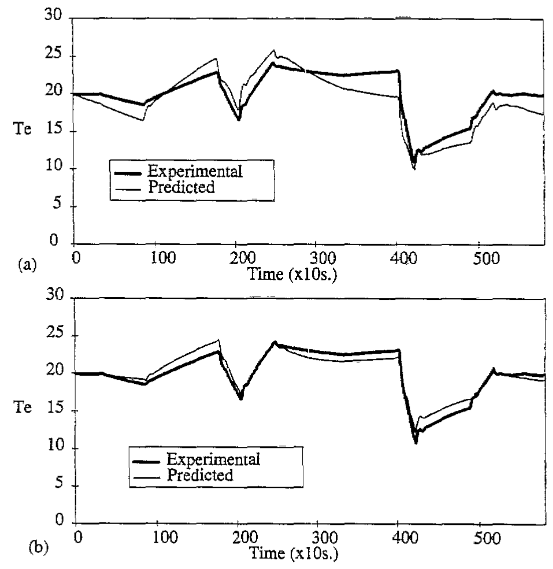

Fig. 12. Te time-profile provided by parallel model of parameters $W_{\tilde{F}}$ for *Data*2. (a) 1st model (b) 2nd model.

$$\begin{aligned}\tilde{\mathrm{Te}}_{\mathrm{r}}(k+1) = {} & \tilde{\mathrm{Te}}_{\mathrm{r}}(k) \\ & + \tilde{F}_{\mathrm{r}}(\mathrm{Tr}(k), \mathrm{Tr}(k-1), \mathrm{Tc}(k), \mathrm{Ta}(k) \\ & \times, u(k-3), \ldots, u(k-23), R_1(k), R_2(k) \\ & \times, W_{\tilde{F}_{\mathrm{r}}} \end{aligned} \tag{40}$$

$$\begin{aligned}\tilde{\mathrm{Te}}_{\mathrm{r}}(k+1) = {} & \tilde{\mathrm{Te}}_{\mathrm{r}}(k) \\ & + \tilde{F}_{\mathrm{r}}(u(k-5), \ldots, u(k-12), u(k-15) \\ & \times, u(k-18), u(k-21), u(k-23), R_1(k) \\ & \times, R_2(k), W_{\tilde{F}_{\mathrm{r}}}) \end{aligned} \tag{41}$$

where $R_1(k) = \tilde{\mathrm{Te}}_{\mathrm{r}}(k-1)$, $R_2(k) = \tilde{\mathrm{Te}}_{\mathrm{r}}(k-2)$ in Eq. (40) and $R_1(k) = \tilde{\mathrm{Te}}_{\mathrm{r}}(k-1) - \mathrm{Tr}(k-1)$, $R_2(k) = \tilde{\mathrm{Te}}_{\mathrm{r}}(k-2) - \mathrm{Tr}(k-2)$ in the second model, Eq. (41).

The new parallel structures have been initialised with the parameters obtained after 2000 learning cycles over the respective series–parallel models. Subsequently, the training of the RNNs is carried out according to the learning rule given by Eq. (23) and learning rate fixed to 0.00001. After 1000 learning cycles the convergence is reached. The prediction errors are shown in Table 5. In contrast to the results shown in Fig. 11, the parallel models built up with the RNN provide appropriate predictions of the heat transfer fluid temperature, see Fig. 12. On the other hand, both model1 and model2, Eqs. (40) and (41) provide similar predictions. However, due to its simplified structure Eq. (41) is more convenient for practical applications. The results also validate the method used in this work for model simplification.

Table 5
Approximating F with the RNN: *Data*1

	Number of parameters	$E^{\mathrm{p}}(W_{\tilde{F}_{\mathrm{r}}})$
Model1	812	3.25
Model2	320	2.48

To test the generalisation (extrapolation) properties of the parallel models built up with the RNN, the prediction error was evaluated for the data sets *Data*2 and *Data*3. These errors are shown in Table 6 and the simulated temperature–time profiles of the heat transfer fluid temperature can be seen in Figs. 12 and 13, respectively. As can be observed in Fig. 13, the generalisation properties of the first model are quite poor in comparison with the second model. This is due to the fact that context neurons in the second model memorise the differences Te–Tr instead of Te and Tr as in the first model.

6. Discussion and conclusions

As we have mentioned, physical models are difficult and time consuming to build up. Moreover, in some

Table 6
Generalisation capability of the parallel model built up with the RNN

	$E^{\mathrm{p}}(W_{\tilde{F}_{\mathrm{r}}})$: *Data*2	$E^{\mathrm{p}}(W_{\tilde{F}_{\mathrm{r}}})$: *Data*3
Model1	2.86	685.4
Model2	0.77	20.3

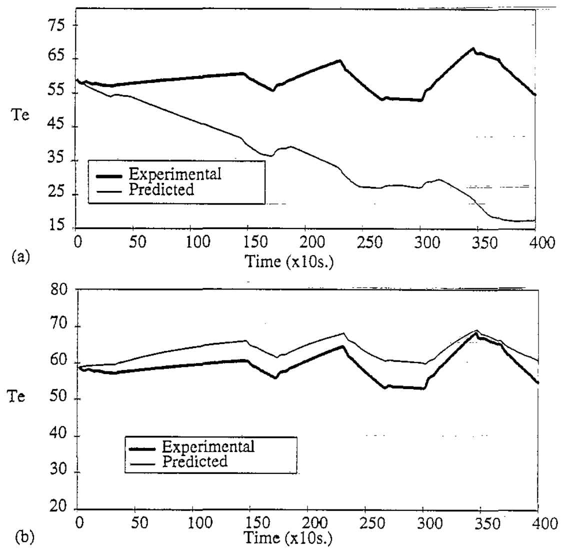

Fig. 13. Te time-profile provided by parallel model of parameters W_F for *Data*3. (a) 1st model (b) 2nd model.

cases, they may be not suitable in real-time applications because the integration time may be higher than the sampling time when a high number of differential equations are requested to explain the dynamic behaviour of the process. On the other hand, models capable of simulating the nonlinear 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, for predicting the dynamic behaviour of some process, to first principle (phenomenological) models if they are built up in an appropriate way. If 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.

Furthermore, to develop a simulator of the process, the parameters or weights determining the parallel model have to be identified, i.e. the prediction error has to be minimised. This implies the use of the RNN to approximate the functional determining the NARMA model and, hence, the best simulator of dynamic process belonging to the class of NARMA model is guaranteed. As can be seen in Fig. 5a and Fig. 11, process representations arisen from the parallel model built up with the RNN are similar to representations obtained by first principle models, whereas predictions provided by the parallel model whose parameters has 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 property, principally, when the identification phase of models has to be carried out in real time.

The generalisation capacities of models built up with the RNN have also been evaluated. The temperature–time profile prediction for *Data*2 provided by neural models, Fig. 12, is satisfactory and similar to the prediction obtained by the phenomenological model, see Fig. 5b. However, the predictions are not so acceptable for *Data*3, see Fig. 5c and 13. In this case, the range of the heat transfer fluid temperature ($\approx 50-70°C$) differs from the temperature range used to train the neural model ($\approx 10-30°C$). The consideration of temperature differences as input to the second neural model, Eq. (35), is an artefact to improve the extrapolation capacities of neural networks. However, to obtain suitable extrapolation properties with neural models, the training data set may to cover a wide range of output process variable.

Acknowledgements

The authors are most grateful to Dr Pedro Pimenta for his valuable comments and suggestions. I.M.G. gratefully acknowledges the research grant from the TMR programme of the European Commission.

References

[1] G.D. Cawthon, K.S. Knaebel, Optimization of semibatch polymerization reactions, Comput. Chem. Eng. 13 (1989) 63–72.
[2] M.H. Gordon, G.J. O'Brien, C.J. Hensler, K. Marcali, Mathematical modelling in thermal hazards evaluation, Plant Operat. Prog. 1 (1982) 27–32.
[3] J.M. Zaldívar, M.A. Alós, E. Molga, H. Hernández, K.R. Westerterp, The effect of phase inversion during semibatch aromatic nitrations, Chem. Eng. Process. 34 (1995) 529–542.
[4] D. Bonvin, P. de Vallière, D.W.T. Rippin, Application of estimation techniques to batch reactors I. Modelling thermal effects, Comput. Chem. Eng. 13 (1989) 1–9.
[5] P. de Vallière, D. Bonvin, Application of estimation techniques to batch reactors II. Experimental studies in state and parameter estimation, Comput. Chem. Eng. 13 (1989) 11–20.
[6] R. King, E.D. Gilles, Multiple filter methods for detection of hazardous states in an industrial plant, AIChE J. 36 (1990) 1697–1706.
[7] J. Villermaux, C. Georgakis, Current problems concerning batch reactions, Int. Chem. Eng. 31 (1991) 434–440.
[8] J.M. Zaldívar, H. Hernández, C. Barcons, Development of a mathematical model and a simulator for the analysis and optimisation of batch reactors: Experimental model characterisation using a reaction calorimeter, Thermochim. Acta 289 (1996) 267–302.
[9] M. Zlokarnik, Dimensional Analysis and Scale-up in Chemical Engineering, Springer-Verlag, Berlin, 1991.
[10] I.J. Leontaritis, S.A. Billings, Input–ouput parametric models for nonlinear systems. Part I: Deterministic nonlinear systems, Int. J. Control 41 (1985) 303–328.

[11] G. Cybenko, Approximation by superposition of a sigmoidal function, Math. Control Signals Syst. 2 (1989) 303–314.

[12] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal aproximators, Neural Networks 2 (1989) 359–366.

[13] D. Rumelhart, G. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation, in Parallel Distributed Processing, MIT Press, Cambridge, 1986.

[14] J. Moody, C.J. Darken, Learning with localized receptive fields. in: Touret-zky, Hinton and Sejnowski (Eds.), Proceedings of the 1988 Connectionist Models Summer School, Morgan-Kaufmann, 1988.

[15] J.J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 281–294.

[16] K.S. Narendra, K. Pathasarthy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1990) 4–27.

[17] N. Bhat, T.J. McAvoy, Use of neural nets for dynamic modelling and control of chemical process systems, Comput. Chem. Eng. 14 (1990) 573–583.

[18] S. Chen, S.A. Billings, Neural networks for nonlinear dynamic systems modelling and identification, Int. J. Control 56 (1992) 319–346.

[19] A.U. Levin, K.S. Narendra, Identification using feedforward networks, Neural Comput. 7 (1995) 349–357.

[20] K.S. Narendra, K. Parthasarathy, Gradient methods for the optimization of dynamical systems containing neural networks, IEEE Trans. Neural Networks 2 (1991) 252–262.

[21] M.M. Polycarpou, P.A. Ioannou, Identification and control of nonlinear systems using neural network models: Design and stability analysis. Department of Electronic Engineering Systems, University of South California, Los Angeles, USA. Technical Report 91-09-01, September 1991.

[22] Y. Yonhg, M. Nikolaou, Dynamic process modeling with recurrent neural networks, AIChE J. 39 (1993) 1654–1667.

[23] B. Srinivasan, U.R. Prasad, N.J. Rao, Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models, IEEE Trans. Neural Networks 5 (1994) 213–228.

[24] A.G. Parlos, K.T. Chong, A.F. Atiya, Application of recurrent multilayer perceptron in modeling complex process dynamics, IEEE Trans. Neural Networks 5 (1994) 255–266.

[25] E.B. Kosmatopoulos, M.M. Polycarpou, M.A. Chistodoulou, P.A. Ioannou, High-order neural network structures for identification of dynamical systems, IEEE Trans. Neural Networks 6 (1995) 422–431.

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

[27] J.M. Zaldívar, H. Hernández, H. Nieman, E. Molga, C. Bassani, The FIRES project: experimental study of thermal runaway due to agitation problems during toluene nitration, J. Loss Prev. Process Ind. 6 (1993) 319–326.

[28] C. Barcons, Equipment Characterization, in: A. Benuzzi, J.M. Zaldívar (Eds.), Safety of Chemical Batch Reactors and Storage Tanks, Kluwer, Dordrecht, 1991, pp. 99–124.

[29] J.R. Bourne, M. Buerli, W. Regenass, Heat transfer and power measurements in stirred tanks using heat flow calorimetry, Chem. Eng. Sci. 36 (1981) 347–354.

[30] H. Hernández, J.M. Zaldívar, C. Barcons, Development of a mathematical model and a numerical nimulator for the analysis and optimization of batch reactors, Comput. Chem. Eng. 17S (1993) 45–50.

[31] M.I. Jordan, Generic Constraints on underspecified target trajectories, IJCNN Proceedings, IEE, New York, June, 1989.

[32] J.M. Zaldívar, F. Panetsos, H. Hernández, Control of batch reactors using neural networks, Chem. Eng. Process. 31 (1992) 173–180.