Application of recurrent neural networks in batch reactors Part II: Nonlinear inverse and predictive control 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-Leganès, 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

Although nonlinear inverse and predictive control techniques based on artificial neural networks have been extensively applied to nonlinear systems, their use in real time applications is generally limited. In this paper neural inverse and predictive control systems have been applied to the real-time control of the heat transfer fluid temperature in a pilot chemical reactor. The training of the inverse control system is carried out using both generalised and specialised learning. This allows the preparation of weights of the controller acting in real-time and appropriate performances of inverse neural controller can be achieved. The predictive control system makes use of a neural network to calculate the control action. Thus, the problems related to the high computational effort involved in nonlinear model-predictive control systems are reduced. The performance of the neural controllers is compared against the self-tuning PID controller currently installed in the plant. The results show that neural-based controllers improve the performance of the real plant.

Keywords: Batch reactors; Heat transfer; Neural networks

1. Introduction

Control theory has been developed in great detail for systems defined by linear operators. However, the nonlinear nature of most dynamic systems in the real world has demanded the incorporation of nonlinear control techniques to improve the controller performance. In recent years there has been a growth beyond all expectation in the development of nonlinear control systems [1–5]. In this context, since the control of batch reactors remains an open problem [6], it has been used as a case study in an innumerable number of publications. Artificial neural networks (NN) have played an important role in this development. Their properties, such as their adaptive and nonlinear nature, and their learning and approximation capabilities have contributed to make NN a useful tool in dealing with the problem of controlling nonlinear dynamic systems. As a consequence, numerous control schemes using NN have been proposed during the last years, see Åström and McAvoy [7] for an overview.

Although nonlinear inverse and predictive control techniques based on NN have been extensively applied to nonlinear systems, their use in real time applications is generally limited. Furthermore, the majority of published experimental results concerning process control and, specifically, batch reactors control have used NN for nonlinear process modelling in a control scheme [8–10]. Recently, Dirion et al. [11] have designed a neural-network controller, and used it for real-time control of an experimental pilot-plant reactor. They created a neural-network learning database using results from an advanced control algorithm (Generalized Predictive Controller) and from plant operators (manual control). Their results show that the trained neural controller was able to mimic decisions made by a human operator.

This paper is focused on the study and development of inverse and predictive nonlinear control strategies

* Corresponding author. Tel.: +39 332 789202; fax: +39 332 785815; e-mail: jose.zaldivar-comenges@jrc.it

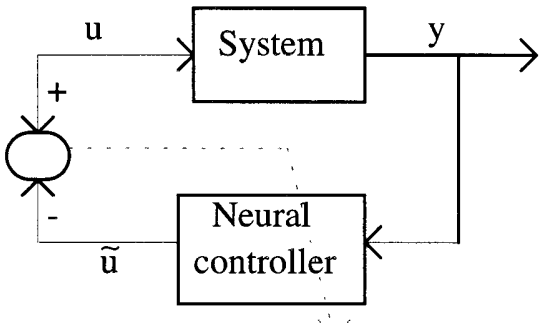

Fig. 1. Schematic representation of the generalized method.

applied to the control of the heat transfer fluid temperature in a 100 l pilot plant. Both inverse and predictive nonlinear control using artificial neural networks have been already studied by different authors. However, they still present some disadvantages concerning real-time applications.

In the inverse neural control systems, the learning of the controller can be accomplished using two different methods, also referred to—in this work—as generalised and specialised learning. In the generalised learning method [12,13], the neural network is trained to learn the total inverse dynamics of the plant. In contrast to the situation when a modelling problem is formulated, the input to the network is the output of the dynamic system and the target output for the network is the system input, see Fig. 1. After the learning procedure is completed, the weights of network are fixed and the NN can be used for controlling, as is shown in Fig. 2. That control scheme is an off-line method because the controller is not operational during the learning stage. Hence, the success of the total inverse neural controller acting in real-time depends on two factors. First, the number of output–input pairs available to train the network. To guarantee an efficient control, the inverse dynamic of the process must be appropriately represented in the training data set, which normally requires a considerable amount of output–input pairs. Secondly, since no objective control (y^{sp}) has been defined during the training of the NN, the success of the controller depends also on the generalisation capacity of the NN, i.e. on the ability of the network to respond correctly to inputs that were not specially used during the training procedure. On the other hand, other problems of the generalised method arise when the inverse dynamic of the system is not well

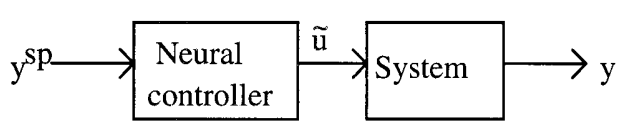

Fig. 2. Inverse neural controller acting in real time.

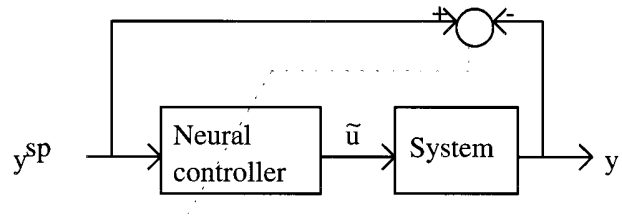

Fig. 3. Schematic representation of the specialised method.

defined, that is, when different plant inputs may produce the same output. In this case, the NN, as any approximation method, will tend to average over the various targets, producing a mapping that is not necessarily the inverse function of the system.

In the specialised learning method, the learning of the controller is carried out using the difference between current and desired output of the system, see Fig. 3, and the patterns for the learning derive directly from the system's evolution. This method appears to be more accurate than generalised learning since it approximates the inverse of the plant in the region of interest instead of in the entire range of the manipulated or input variable. There are numerous versions and applications of this method [8,14–18].

The specialised method solves problems encountered during the generalised learning because it is an on-line method. However, the initialisation of the neural controller weights have an important repercussion on the control quality. If the controller weights are set-up to random values, unstable or even 'no system control' situations may be produced, which is not admissible in most of the real-time applications. Therefore, to obtain an efficient control of the dynamic system in real time, when the specialised method is applied, it is necessary to prepare the parameters of inverse neural controller.

The other control schema studied in this work is the model-predictive control strategy, whose initial idea was due to Martín [19]. In this control schema the manipulated variable or control signal profile is determined optimising some performance function on a time interval extending from the current time to the current time plus a prediction horizon which is defined previously [20]. Predictive control schemes have reached considerable interest during the past decade and they have received wide acceptance in industry [21]. These

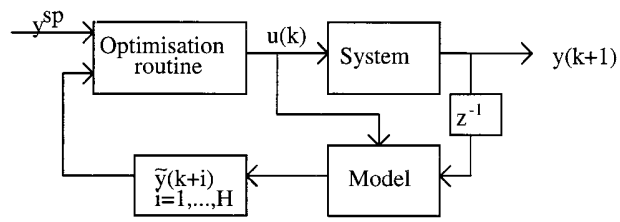

Fig. 4. Nonlinear predictive control system.

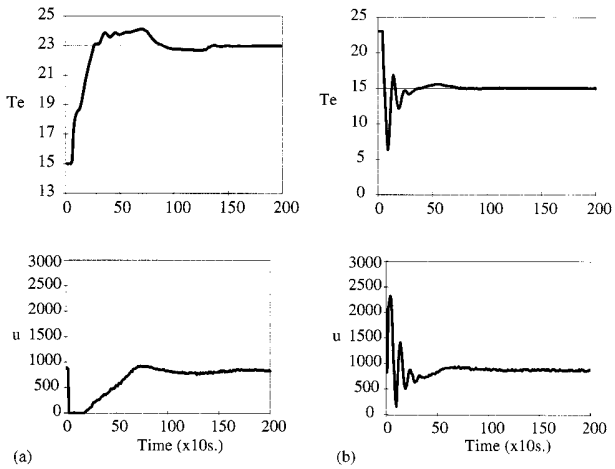

Fig. 5. PID controller performance. Constants: $K = 13\%$, $T_i = 1'41''yT_d = 0$. (a) $T_e^{sp} = 23°C$, (b) $T_e^{sp} = 15°C$.

control strategies had been mathematically developed for linear dynamic systems [22,23] in which linear ARMA models are used to predict the future behaviour of the system. However, since most of real world dynamic systems are highly nonlinear, the predictive control based on linear models may result in very poor controller performance. Therefore, predictive control techniques have recently been extended to nonlinear systems [20,24,25] in which nonlinear models are used to describe the nonlinearities and complexities of the system. NN have also taken part in the development of nonlinear model-predictive control systems [26,27], in which models built up with neural structures have been used to predict the dynamic behaviour of the system along the prediction horizon.

When the predictive control strategy is based on nonlinear models, independently of the nature of models, the prediction equations cannot be solved explicitly, as in the linear case, and an iterative solution of the performance function evaluating the future behaviour of the system is required. At every sampling time, the current manipulated variable is calculated using an optimisation procedure, see Fig. 4, which determines the optimal profile control actions that minimise the difference between the desired and model outputs over the prediction horizon. Hence, the use of nonlinear models allow the development of predictive control strategies for nonlinear dynamic systems, but it also implies that at every sampling time a nonlinear optimisation problem must be solved. This is a potential disadvantage of those control strategies because the solution of optimisation problems are, normally, computationally laborious. Most nonlinear optimisation algorithms use some form of search technique to scan the feasible space of the performance function until the extremal point is located [28]. This search is generally guided by calculations of the performance function and/or its derivatives and it implies a large computational effort because several iterations have to be carried out to reach its extremal point. The computational time requirements may be an obstacle for nonlinear model-predictive control systems in real time applications. As a consequence, it would be worthwhile developing nonlinear predictive control strategies that require less computational effort. In this paper the emphasis is placed on the development of inverse and predictive nonlinear control systems feasible for real time applications and capable of efficiently controlling real systems.

The nonlinear inverse control schema, used for the control of the temperature of the heat transfer fluid, consists in combining the generalised and specialised method. First, the neural controller is trained using the generalised learning algorithm with the available output–input pairs. The trained recurrent neural network (RNN) might produce an inadequate control of the real plant because the patterns available do not represent its total inverse dynamics, but it will have enough information so that unstable or no control situations may be avoided. Secondly, the specialised learning of the neural inverse controller, using a model of the system, is performed. Hence, the controller can itself specialise in the desired objective control. These parameters are conserved and used to initialise the controller acting in real-time. Finally, the controller with the parameters initialised in that way is submitted to the specialised learning in real-time using the real plant.

The nonlinear predictive control schema developed in this work is characterised because the optimisation routine in Fig. 4 is replaced by a RNN. Thus, the current control signal is given by the output of a RNN, instead as the solution of the optimisation problem, and no nonlinear optimisation problem must to be solved at every sampling time. For this reason, this control system requires less computational effort.

The paper is organised as follows. The experimental set-up of the pilot plant reactor, the control configuration as well as the behaviour of the PID cascade controller currently incorporated in the plant, are presented in the second section. The proposed neural nonlinear inverse control system is presented in the third section whereas its implementation is discussed in the next section. In the fifth section the temperature control of the heat transfer fluid temperature in a chemical batch reactor is performed using both inverse and predictive control systems. The results show that neural-based controllers improve the performance of the real plant.

2. Experimental configuration

The Facility for Investigating Runaway Events Safely (FIRES) reactor [29] is a 100 l stainless steel, glass-lined pilot reactor which is equipped with a standard cooling/heating jacket. The main specifications and characteristics has been presented in the first part of this work [30] and, therefore, will not be discussed here.

When the process is carried out in isothermal conditions, the reactor temperature, T_r, 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 recirculating through the reactor jacket is controlled to the set-point using a slave controller. The master and slave controllers currently installed in the cascade control schema of the reactor temperature are PID self-tuning controllers. The neural controllers developed in this work act as the slave controller of the plant.

2.1. Description of the experiments

The control performances have been tested using the real plant. The experiments have been carried out with the reactor filled with 80 kg of water inside the reactor which was equipped with a standard Pfaudler stirrer at 5 s^{-1} of stirring speed. To evaluate the neural controller capability in heating as well as in cooling mode, two different set-points have been defined. Starting from the initial condition $T_e(0) = 15°C$ the first objective is to follow a control reference set to $T_e^{sp} = 23°C$ while the second objective is to reach $T_e^{sp} = 15°C$ when $T_e(0) = 23°C$. The PID controller performance for these control objectives is shown in Fig. 5. The controllers constants were adjusted following the operation's manual provided by the constructor.

The algorithms for neural control of the heat transfer fluid temperature were stored on the second workstation of the distributed control system [30]. The NN-based software sends the neural controller output, u, and in the next sampling time (10 s later) receives the measured system variables.

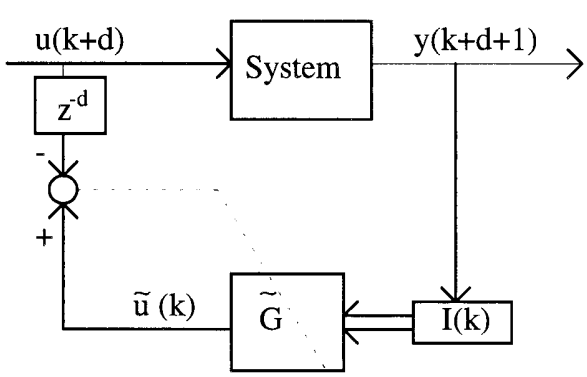

Fig. 6. First learning phase of the inverse neural controller.

3. Inverse neural control system

3.1. Inverse controller structure

When the inverse dynamic of the system is approximated by NNs, the collection of selected input variables to the network has an important repercussion on the control quality. To evaluate the system variables which have an effect on the inverse dynamic, a system model has to be assumed. Leontaritis and Billings [31] have proved that nonlinear discrete dynamic system can be represented by the following NARMA model:

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

where $y(.)$ and $u(.)$ are discrete sequences of output and input variables, respectively, and d is the delay of the system.

Let us suppose that the dynamic system represented by Eq. (1) is invertible, i.e. there exists a function $G(.)$ such that the input can be expressed in terms of a nonlinear expansion of lagged inputs and outputs as follows:

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

Updating the previous equation at current time k, it follows:

$$u(k) = G(y(k+1+d), y(k+d), \ldots, y(k+d-n_y), u(k-1), \ldots, u(k-n_u)) \quad (3)$$

Assuming that function G is known, the expression given by Eq. (3) allows the calculation of the control action at time k such that the value $y(k+d+1)$ is reached by the system at time $k+d+1$. Thus, if the objective of the control action is to reach a set-point, y^{sp}, the control action is obtained replacing the process output at time $k+d+1$ by the desired plant output, that is:

$$u(k) = G(y^{sp}, y(k+d), \ldots, y(k+d-n_y), u(k-1), \ldots, u(k-n_u)) \quad (4)$$

Eq. (4) shows that the future measurements of system output, $y(k+d), \ldots, y(k+1)$, are required for the calculation of the current signal control. However, when the control of the real plant is performed, those values are not available. Hence, it is necessary to transform the control law given by Eq. (4) into an expression applicable in real-time. As is shown in the following, that conversion is immediate.

Applying Eq. (1), the values $y(k+d)$, $y(k+d-1)$, …, $y(k+1)$ can be expressed as follows:

$$y(k+d) = F(y(k+d-1), \ldots, y(k-(n_y+1-d)), u(k-1), \ldots, u(k-(n_u+1))) \quad (5)$$

$$y(k+2) = F(y(k+1), \ldots, y(k-(n_y-1)), u(k-(d-1)), \ldots, u(k-(d+n_u-1))) \quad (6)$$

$$y(k+1) = F(y(k), \ldots, y(k-n_y), u(k-d), \ldots, u(k-(d+n_u))) \quad (7)$$

Replacing $y(k+1)$ in Eq. (6) by Eq. (7), it is observed that $y(k+2)$ can be written as a function of $y(k)$, …, $y(k-(n_y-1))$, $y(k-n_y)$, $u(k-(D-1))$, $u(k-d)$, …, $u(k-(d+n_u))$. Making successive substitutions up to reach the term $y(k+d)$ allows $y(k+d)$, $y(k+d-1)$ to be written in terms of the current available information. Consequently, keeping the notation G, the current control action can be expressed as:

$$u(k) = G(y^{sp}, y(k), \ldots, y(k-n_y), u(k-1), \ldots, u(k-n_u), \ldots, u(k-(d+n_u))) \quad (8)$$

The difference between the inverse control laws given by Eqs. (4) and (8) consists in the length of the input variable sequence, which has increased by d terms in the transformed control law, Eq. (8). The length increase is given by the delay of the system.

Since the manipulated input variable to the system is not available when the control problem is formulated, the recurrent neural network (RNN) presented in [30] is an appropriate architecture to approximate the functional G appearing in Eq. (8). Introducing the vector EMBED Equation as the input to the network and defining $d+n_u$ context neurones, the approximate inverse control law adopts the following form:

$$\tilde{u}(k) = \tilde{G}(I(k), R(k), W_{\tilde{G}}) = \tilde{G}(y^{sp}, y(k), \ldots, y(k-n_y), R_1(k), \ldots, R_{d+n_u(k)}, W_{\tilde{G}}) \quad (9)$$

where $\tilde{u}(k)$ is the output of the RNN, $R_i(k) = \tilde{u}(k-i)$, $i = 1, \ldots, d+n_u$ and $W_{\tilde{G}}$ is the set of adjustable parameters of the network.

3.2. Inverse learning algorithm

Once the controller structure has been determined, the training of the neural controller must be accomplished. It consists in determining the parameter set $W_{\tilde{G}}$ in such a manner that the control law defined by Eq. (9) results in an inverse control strategy. The learning is divided into three different phases: generalised method with the pattern set available, specialised method using a system model and a specialised method in the real plant. The first two phases are carried out off-line; the aim being to prepare the parameters of the final controller.

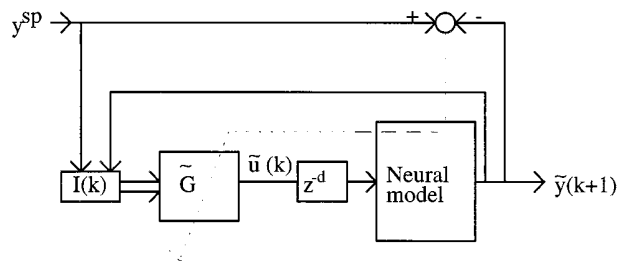

Fig. 7. Second learning phase of the inverse neural controller.

3.3. 1st phase: generalised learning with the patterns available

In this phase the parameter set is $W_{\tilde{G}}$ estimated so that the control law given by Eq. (9) acquires information about the global inverse dynamic behaviour of the real plant. The training data set $\{u(k), y(k+d+1)\}$ is obtained using the plant as black box. The input signal is varied randomly within its working range and the output of the system is stored. A delayed sequence of this variable, $I(k) = (y(k+d+1), y(k), \ldots, y(k-n_y))$, is used as input to the RNN and the input system $u(k)$ is the target output for the network, see Fig. 6. In this case, the desired set-point, y^{sp}, does not appear in the input vector to the RNN because the objective is to approximate the global inverse dynamic of system. The neural controller is trained to minimise the following performance function:

$$E_c^g = \frac{1}{2N} \cdot \sum_{k=0}^{N-d-1} (u(k) - \tilde{u}(k))^2 \quad (10)$$

The parameters of the RNN are adjusted using the dynamic backpropagation algorithm described in [30]. To accelerate the convergence of the algorithm it is convenient to realise a previous learning of the equivalent multilayer feedforward network with input vector $I(k) = (y(k+d+1), y(k), \ldots, y(k-n_y), u(k-1), \ldots, u(k-d-n_u))$, using the static backpropagation algorithm. Once the training of the multilayer feedforward network is accomplished, the parameters are used to initialise the RNN and the generalised learning of the recurrent neural controller is realised.

Since this learning phase is carried out with the output–input patterns available, the set of parameters obtained will not produce an appropriate control of the real system, although they will have enough information so that unstable or no control situations can be avoided when the inverse neural controller is acting in real time.

3.4. 2nd phase: specialised learning using a system model

Given a system model and the desired output or set-point, y^{sp}, the goal is to specialise the parameter set $W_{\tilde{G}}$, Eq. (9), obtained in the previous phase on that set-point. Defining a discrete time interval $[0, n]$ where n is a natural number indicating the time required for the model to reach the set-point, the parameters of the network are adjusted along the negative gradient direction of the error function defined as follows:

$$E_c^g = \frac{1}{n} \cdot \sum_{k=d}^{n-1} (y^{sp} - \tilde{y}(k+1))^2 \tag{11}$$

where $\tilde{y}(k+1)$ is the model output.

Applying the chain rule to calculate the gradient of Eq. (11), the following learning rule is inferred:

$$W_{\tilde{G}}^{(k+1)} = W_{\tilde{G}}^{(k)} + \alpha \cdot (y^{sp} - \tilde{y}(k+1)) \cdot \frac{\partial \tilde{y}(k+1)}{\partial \tilde{u}(k-d)} \cdot \frac{\partial \tilde{u}(k-d)}{\partial W_{\tilde{G}}}, \quad k = d, \ldots, n-1 \tag{12}$$

where $\tilde{u}(k-d)$ is the delayed output neural controller. The terms $\partial \tilde{y}(k+1)/\partial \tilde{u}(k-d)$, $k = d, \ldots, n-1$, appear in the gradient calculation because the model is located between the controller and the error measured, see Fig. 7. They represent the Jacobian of the model at every time, that is, the variation of the output model with respect to the applied input. The factors $\partial \tilde{u}(k-d)/\partial W_{\tilde{G}}$, $k = d, \ldots, n-1$, represent the variation of the RNN output with respect to the parameter $W_{\tilde{G}}$ and they can be calculated using either static or dynamic backpropagation algorithms [30].

The use of a system model to perform the specialised learning of the controller instead of the real system allows the patterns $\{I(k) = (y^{sp}, \tilde{y}(k), \ldots, \tilde{y}(k - n_y)), y^{sp}\}$, $k = d, \ldots, n-1$, to be presented several times to the neural controller. As a consequence, the inverse control law given by Eq. (9) learns to calculate the control actions $u(0), \ldots, u(n-1-d)$ such that the error given by Eq. (11) is minimised.

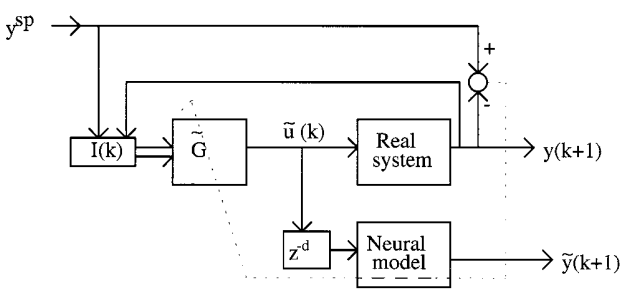

Fig. 8. Third learning phase of the inverse neural controller.

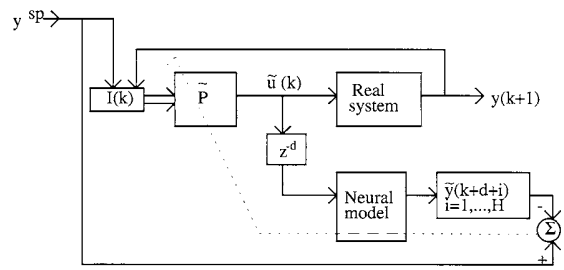

Fig. 9. Learning procedure of the nonlinear predictive controller.

3.4.1. Comments:

In this learning phase the model is used to simulate the dynamic behaviour of the real plant, $\tilde{y}(k+1)$, $k = d, \ldots, n-1$, and to backpropagate to the neural controller the error measured at output model, as is shown in Fig. 7. Hence, the chosen model to carry out this learning phase has to be capable to act as process simulator and its structure must facilitate the calculation of the Jacobian. The NARMA simulators developed in [30] have both properties. They can produce appropriate approximations of the system when it is necessary to simulate its dynamics and they allow the calculation of their Jacobian using the static bakpropagation algorithm [15].

3.5. 3rd phase: specialised learning using the real system

After the two earlier phases are finished, the parameters of the inverse control law given by Eq. (9) are initialised to the set $W_{\tilde{G}}$ obtained previously. The acceptance of that law to control the real system depends, obviously, on the capacity of the model used in the second phase to represent the dynamic behaviour of the system. If the chosen model were to be exact, the control law would produce the best control and its parameters could be fixed. However, since most system models provide approximations of the real dynamics, the performance of a neural controller whose parameters are fixed, would not be suitable. In order to correct the errors in the neural controller occasioned by the use of an approximate model in the previous phase, the specialised learning using the real system is realised. In this case, the controller is operating during the training phase and it is accomplished so that the real plant reaches asymptotically the set-point, y^{sp}.

At every discrete time k, the input to the RNN is actualised and its output, $\tilde{u}(k)$, is applied to the real system, Fig. 8. The weights adjustment is based on the negative gradient direction of the measured error at output plant:

$$e_c^s(k+1) = \frac{1}{2} \cdot (y^{sp} - y(k+1))^2 \tag{13}$$

where $y(k+1)$ is the system output.

In this case, the learning rule to adjust the controller parameters adopts the following form:

$$W_{\tilde{G}}^{(k+1)} = W_{\tilde{G}}^{(k)} + \alpha \cdot (y^{sp} - y(k+1)) \cdot \frac{\partial y(k+1)}{\partial \tilde{u}(k-d)} \cdot \frac{\partial \tilde{u}(k-d)}{\partial W_{\tilde{G}}} \quad (14)$$

The Jacobian of the process, $\partial y(k+1)/\partial \tilde{u}(k-d)$, is normally unknown and difficult to determine from experimental data. Different authors have proposed solutions to this problem. For instance, Saerens and Soquet [32] replaced the unknown derivative by it sign which was obtained using qualitative knowledge of the plant behaviour. Another scheme has been proposed in [33], it makes use of an inverse transfer relationship, determined experimentally, to estimate the error for the output controller. In this paper, based on studies realised by Jordan [15], the Jacobian of the system is approximated by the Jacobian of a system model, $\partial \tilde{y}(k+1)/\partial \tilde{u}(k-d)$.

3.5.1. Comments

As in the second learning phase a model of the system is required, although in this case the model is only used to backpropagate the measured error at process output to the controller output, see Fig. 8. Models built up with NNs are also advisable because they allow easy calculation of its Jacobian [15]. It is important to notice that the accuracy of the Jacobian approximation has an important effect on the control results. Errors in the Jacobian approximations could cause unsuitable changes in the parameters and oscillations in the neural controller output could occur. On the other hand, the accuracy of the Jacobian approximation, $\partial \tilde{y}(k+1)/\partial \tilde{u}(k-d)$ depends obviously on the accuracy of the model. Hence, even if the model parameters have been estimated before training of the neural controller, in most of the cases it is advisable to perform an on-line identification to correct the model errors occurred in the domain of interest.

4. Predictive neural control system

4.1. Predictive controller structure

In the predictive neural control system proposed in this paper, a NN acts as predictive controller of the system, see Fig. 9. It provides, at every sampling time, the current predictive control action that must be applied to the dynamic system. The aim is to reduce the computational time requirements in nonlinear predictive control systems. The use of a NN to calculate the predictive control action is based on the existence of a relationship between the current control action and the rest of the system variables when a predictive model strategy is applied, as is shown in the following.

Generally, predictive control strategies are formulated as the calculation at every discrete time k of a finite control action sequence $u(k), \ldots, u(k+N_u)$ such that the differences between the desired objective or set-point and the system outputs on a time interval extending to the future are minimised,

$$e_c^p(k+1) = \frac{1}{2} \cdot \sum_{i=1}^{H} (y^{sp}(k+d+i) - \tilde{y}(k+d+i))^2 \quad (15)$$

where H and N_u are natural numbers referred to as prediction and control horizons, respectively; $y^{sp}(k+d+i)$ and $\tilde{y}(k+d+i)$ are the sequences of set-points and the predicted system outputs provided by a system model along the prediction horizon, respectively. When the prediction horizon is bigger than the control horizon, $H > N_u$, it is necessary to add constraints to the optimisation problem. They consist in supposing that the control action remains constant after the N_u time step, $u(k+N_u) = \cdots = u(k+H)$.

For simplicity, the following analysis supposes that control horizon is equal to 0 ($N_u = 0$). Assuming that the dynamic process can be represented by a NARMA model, Eq. (1), and supposing that approximations over the prediction horizon appearing in Eq. (15), $\tilde{y}(k+d+i)$ $i = 1, \ldots, H$ are equal to the measured system outputs, from Eq. (1) it infers that the future values can be expressed as follows:

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

$$\tilde{y}(k+d+2) F(y(k+d+1), \ldots, y(k+d+1-n_y), u \times (k+1), \ldots, u(k+1-n_u))$$

$$\tilde{y}(k+d+H) = F(y(k+d+H-1), \ldots, y(k+d+H-n_y), u \times (k+H), \ldots, u(k+H-n_u))$$

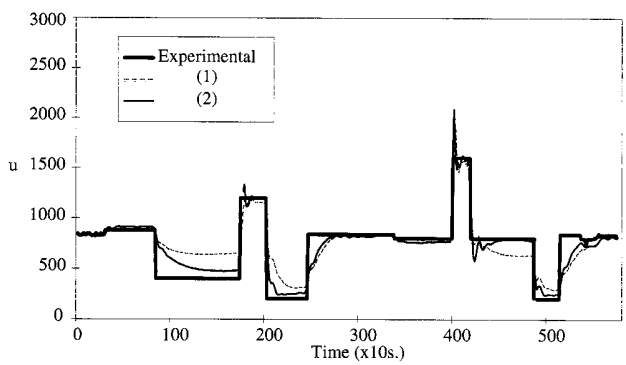

Fig. 10. Approximating the inverse dynamic of system. (1) Multilayer feedforward network (6-10-1). (2) Recurrent neural network (6-10-1, $n_1 = 2$, $n_c = 4$).

Following the steps described in Section 3.1 and imposing that $N_u = 0$, that is, $u(k) = u(k+1) = \cdots = u(k+H-1) = u(k+H)$, the values $\tilde{y}(k+d+i)$ $i = 1, \ldots, H$ can be expressed in terms of the sequence $y(k), \ldots, y(k-n_y), u(k), \ldots, u(k-d), \ldots, u(k-d-n_u)$, i.e. in terms of the information available at discrete time k. Hence, the control action $u(k)$ minimising the error function defined in Eq. (15) can be written as follows:

$$u(k) = P(y^{\mathrm{sp}}(k+d+1), \ldots, y^{\mathrm{sp}}(k+d+H), y(k), \ldots, y(k-n_y), u(k-1), \ldots, u(k-d-n_u)) \quad (16)$$

where P is a mapping that determines the solution of optimisation problem given by Eq. (15). If we assume again that the set-point remains constant over the prediction horizon, Eq. (16) is simplified as:

$$u(k) = P(y^{\mathrm{sp}}, y(k), \ldots, y(k-n_y), u(k-1), \ldots, u(k-d-n_u)) \quad (17)$$

If the function P were known, the expression given by Eq. (17) would provide at every discrete time k the control action that must be applied to the system to reach the desired control objective, when a predictive control strategy is employed. However, in the nonlinear case the functional P is rarely available and its knowledge is unfortunately very complicated. Nonetheless, based on this judgement it is possible to guarantee that there exists a relationship among the current control action and past sequences of input and output system variables when a predictive control strategy is formulated, as in the inverse control case. In addition, Eq. (17) gives the number of variables on which control action depends.

The problem of finding an expression for the functional P can now be interpreted as a functional approximation problem. Based on approximation capabilities of NNs, one can decide to approximate the functional P by a NN. As in the case of system inverse dynamic approximation, the RNN presented in [30] is also an appropriate architecture to deal with the problem. Introducing the vector $I(k) = (y^{\mathrm{sp}}, y(k), \ldots, y(k-n_y))$ as the input to the network and considering $d+n_u$ context neurones, the predictive control action is the only output of the RNN:

$$\begin{aligned}\tilde{u}(k) &= \tilde{P}(I(k), R(k), W_{\tilde{P}}) \\ &= \tilde{P}(y^{\mathrm{sp}}, y(k), \ldots, y(k-n_y), R_1(k), \ldots, R_{d+n_u}(k), W_{\tilde{P}})\end{aligned} \quad (18)$$

where $R_i(k) = \tilde{u}(k-i)$ $i = 1, \ldots, d+n_u$ and $W_{\tilde{P}}$ is the parameter set of the network.

The predictive control law given by Eq. (18) has the same structure as the inverse control law, Eq. (9), deduced in Section 3.1. However, there exits a basic difference between them: the performance function used to determine the parameter sets appearing in both control laws. In the predictive control case, the parameters of RNN, $W_{\tilde{P}}$ are adjusted using the error function evaluating the dynamic behaviour of the system in the future, Eq. (15), whereas the adaptation of the inverse control law parameters, $W_{\tilde{G}}$ were based on the immediate answers of the system, Eqs. (11) and (13).

4.2. Predictive learning algorithm

The learning of the predictive neural controller consists in determining the parameter set $W_{\tilde{P}}$ such that the control law, given by Eq. (18), provides a predictive performance controller. The training procedure is carried out while the neural controller is operating and the patterns come from the direct evolution of the plant. Once a system model provides the futures system outputs over the prediction horizon, the RNN parameters are updating in real time, see Fig. 7, along the negative gradient direction of Eq. (15). In the following, the learning rule to update the controller parameters is deduced.

Let $W_{\tilde{P}}$ be a weight of the RNN; supposing that set-points remains constant along the prediction horizon, the variation of the error function given by Eq. (15) with respect to the parameter is:

$$\frac{\partial e_{\mathrm{c}}^{\mathrm{p}}(k+1)}{\partial w_{\tilde{P}}} = -\sum_{i=1}^{H} (y^{\mathrm{sp}} - \tilde{y}(k+d+i)) \cdot \frac{\partial \tilde{y}(k+d+i)}{\partial w_{\tilde{P}}} \quad (19)$$

The variation of the model outputs with respect to the parameter $w_{\tilde{P}}$ along the prediction horizon is calculated using the chain rule as follows:

$$\frac{\partial \tilde{y}(k+d+i)}{\partial w_{\tilde{P}}} = \frac{\partial \tilde{y}(k+d+i)}{\partial \tilde{u}(k+i)} \cdot \frac{\partial \tilde{u}(k+i)}{\partial w_{\tilde{P}}}, \; i = 1, \ldots, H \quad (20)$$

where $\partial \tilde{y}(k+d+i)/\partial \tilde{u}(k+i)$, $i = 1, \ldots, H$, are the model Jacobians along the prediction horizon; and $\partial \tilde{u}(k+i)/\partial w_{\tilde{P}}$, $i = 1, \ldots, H$ represent the RNN outputs variations with respect to the parameter $w_{\tilde{P}}$. Since $N_u = 0$, the control action implemented in the model remains constant during the prediction horizon and the terms $\partial \tilde{u}(k+i)/\partial w_{\tilde{P}}$ are also constants and equal to $\partial \tilde{u}(k)/\partial w_{\tilde{P}}$. The last term can be calculated using either the static or dynamic backpropagation algorithm [30].

Hence, the neural predictive controller parameters are updated at every sampling time using the following learning rule:

$$+ \alpha \cdot \frac{\partial \tilde{u}(k)}{\partial w_{\tilde{P}}} \cdot \sum_{i=1}^{H} (y^{\mathrm{sp}} - \tilde{y}(k+d+i)) \cdot \frac{\partial \tilde{y}(k+d+i)}{\partial \tilde{u}(k)} \quad (21)$$

The predictive controller operation and its learning procedure, see Fig. 9, can be summarised as follows: at every sampling time k, the input vector $I(k) = (y^{sp}, y(k), \ldots, y(k-n_y))$ is presented to the RNN and its output, $\tilde{u}(k)$, is implemented in the dynamic system and in the system model; the partial derivative $\partial\tilde{u}(k)/\partial w_{\tilde{F}}$ is calculated; once the errors $(y^{sp} - \tilde{y}(k+d+i))$ and the model Jacobians $\partial\tilde{y}(k+d+i)/\partial\tilde{u}(k)$, $i = 1, \ldots, H$, have been evaluated, the neural parameters are adjusted using the Eq. (21) and the procedure is repeated at the next sampling time $k+1$.

4.2.1. Comments

In order to predict the dynamic behaviour of the system along the prediction horizon, a system model is needed. This model must be able to act as system simulator on the time interval $[k, k+H]$. On the other hand, the model looks after backpropagating the errors to the controller output. As in the inverse neural controller scheme, parallel NARMA models developed in [30] are appropriate for the purpose of predictive controlling. Since the success of the predictive control strategy depends on the accuracy of system output prediction provided by the model system, it is suitable to carry out an on-line identification of neural model parameters.

The predictive control scheme previously described is an on-line control method because, as we have already mentioned, the adjustment of the controller parameters is accomplished on real time. Hence, if the initial weights of the neural controller are set to random values, an optimal performance of predictive controller cannot be expected. To obtain an appropriate control of the real plant, it is convenient to provide some initial information to the neural predictive controller. This information can be obtained in an off-line way, this is training the neural controller to realise a predictive control strategy with a system model instead of real plant.

5. Experimental results: neural control of the heat transfer fluid temperature in a jacketed batch reactor

In this section the neural inverse and predictive control schemes described in Sections 3 and 4, respectively, are applied to perform the control of the heat transfer fluid temperature in a jacketed batch reactor. In addition, the performance of both controllers against the self-tunning PID controller currently installed in the plant is also tested.

5.1. Inverse neural control

In [30] the construction of NARMA models simulating the dynamic behaviour of the heat transfer fluid temperature was treated. One conclusion drawn from that study was that the most important variables affecting the current temperature $T_e(k)$ were the delayed control signal $u(k-4)$, the reactor temperature $T_r(k)$, and the heat transfer fluid temperature at previous time, $T_e(k-1)$. Hence, the following simplified NARMA model can be assumed to represent the time evolution of the heat transfer fluid temperature:

$$T_e(k+1) = T_e(k) + F(u(k-4), T_e(k-1) - T_r(k-1)) \tag{22}$$

Following the steps described in Section 3.1 and assuming that reactor temperature remains constant during the time interval $[k, k+3]$, this is $T_r(k+3) \approx T_r(k+2) \approx T_r(k+1) \approx T_r(k)$, the inverse control law can be written as:

$$u(k) = G(T_e^{sp} - T_e(k), T_e(k) - T_r(k), u(k-1), \ldots, u(k-4)) \tag{23}$$

Introducing the input vector $I(k) = (T_e^{sp} - T_e(k), T_e(k) - T_r(k))$ and four context neurones, the functional G can be approximated using the RNN, as:

$$\tilde{u}(k) = \tilde{G}(T_e^{sp} - T_e(k), T_e(k) - T_r(k), R_1(k), \ldots, R_4(k), W_{\tilde{G}}) \tag{24}$$

where $R_i(k) = \tilde{u}(k-i)$ $i = 1, 2, 3, 4$. The number of hidden layers and neurones in those layers have been determined by trial and error, stabilising only one hidden layer of ten neurones. The neuron activation functions have been chosen as the sigmoidal ranged on the interval [0, 1].

5.1.1. First phase

In this phase the variable T_e^{sp} in the input vector to the network, see Eq. (24), is substituted by the measured output value $T_e(k+5)$. The generalised learning of the inverse neural controller has been carried out using an experimental data set which has been obtained with 80 kg of water inside the reactor, 5 s^{-1} stirrer rate and manipulating the control signal, u, directly on the supervising workstation [30].

To accelerate the convergence of the RNN in approximating the dynamic inverse represented by the pattern set, a multilayer feedforward network with ten neurones in the hidden layer and $I(k) = (T_e(k+5) - T_e(k), T_e(k) - T_r(k), u(k-1), \ldots, u(k-4))$ as the input pattern, is previously trained. That network was trained using the static backpropagation algorithm and learning rule varying from 0.1 to 0.01. The value of the error function given by Eq. (10) after 10 000 learning cycles was $E_c^g = 1.7 \cdot 10^{-3}$ and the approximated input vari-

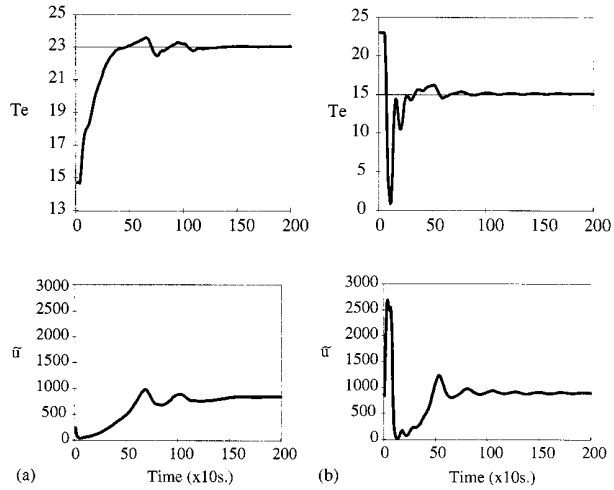

Fig. 11. Inverse neural controller performance: parameters fixed to $W_{\tilde{G}}(generalised)$. (a) $T_e^{sp} = 23°C$, (b) $T_e^{sp} = 15°C$.

able it is shown in Fig. 10. Subsequently, the training of the RNN is performed during 3000 learning cycles and the quadratic error given by Eq. (10) reaches the value $E_c^g = 8.8 \cdot 10^{-4}$. As can be seen from Fig. 10, the RNN is a more appropriate architecture to approximate the inverse dynamics rather than typical multilayer feedforward networks.

Although the generalised learning of the inverse neural controller has not been realised to build up the definitive controller of system, the performance of the control law given by Eq. (24) whose parameters are fixed to the set obtained in this phase, also referred to as $W_{\tilde{G}}(generalised)$, is evaluated. The evolution of the heat transfer fluid temperature and control action for every control objective are shown in Fig. 11. This control law does not produce an efficient control of heat transfer fluid temperature, but it has enough information such that not control or unstable situations can be avoided.

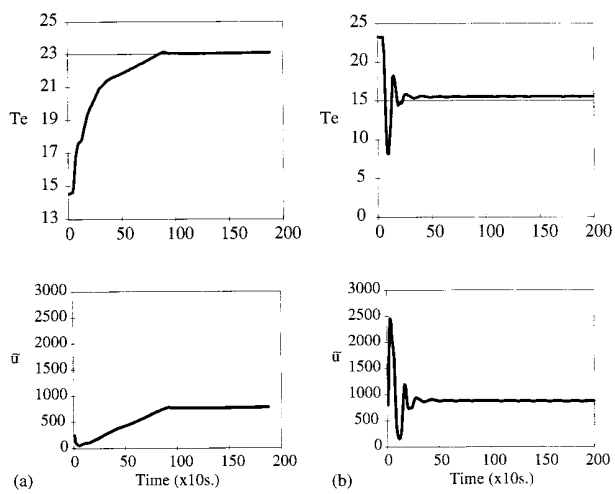

Fig. 12. Inverse neural controller performance: parameters fixed to $W_{\tilde{G}}(specialised\_m)$. (a) $T_e^{sp} = 23°C$, (b) $T_e^{sp} = 15°C$.

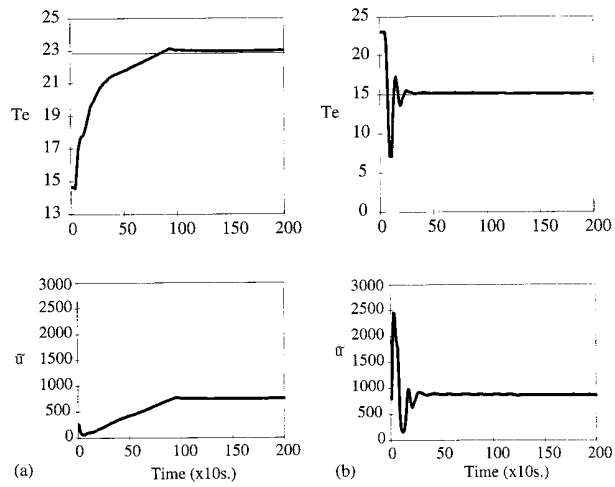

Fig. 13. Inverse neural controller performance: parameters initialised to $W_{\tilde{G}}(specialised\_m)$ an updating in rel time. (a) $T_e^{sp} = 23°C$, (b) $T_e^{sp} = 15°C$.

5.1.2. Second phase

As shown in Section 3.2, a model of the system capable of simulating its dynamic behaviour is required. In this work, neural NARMA parallel models developed in [30] have been used for the purpose of simulating dynamic evolution of the heat transfer fluid temperature in this learning phase. From the initial condition $T_e(0) = 15°C$, the target of the specialised learning phase is to follow the control objective to $T_e^{sp} = 23°C$ for $0 \leq k \leq 300$ and $T_e^{sp} = 15°C$ for $301 \leq k + 600$. Each simulation cycle, for k ranging from 0 to 600 is called a trail. Initialising the inverse control law, Eq. (24), by the parameter set $W_{\tilde{G}}(generalised)$, they are updated at every time k using the learning rule given by Eq. (12) and a learning rate fixed to 0.01. The weights obtained after 20 trails, also named $W_{\tilde{G}}(specialised\_m)$, are conserved and used to control the real plant. Fig. 12 shows the performance of the inverse control law given by Eq. (24) whose parameters have been fixed to the set $W_{\tilde{G}}(specialised\_m)$. Since it has been targeted on the desired control objective, the controller performance has been improved as expected. The overshoot of the temperature is significantly reduced with these new parameters.

5.1.3. Third phase

The control results previously obtained are satisfactory. However, in the cooling mode it is observed that the controller provokes an offset in the evolution of the heat transfer fluid temperature, i.e. the temperature is stabilised around 15.4°C instead to 15.0°C, see Fig. 12(b). This fact is given by the use of a 'not exact'

model in the previous phase. To correct this type of errors, the specialised learning of the neural inverse controller using the real plant is accomplished. Using the set $W_{\tilde{G}}(specialised\_m)$ to initialise the controller, the parameters are updated in real time according to the learning rule given by Eq. (14) and a learning rate fixed to 0.01. Since the controller parameters are adjusted using the difference between the real plant output and the set-point, Eq. (13), errors produced by the use of a 'not exact' model are corrected and the offset in the evolution of heat transfer fluid temperature has been eliminated, as can be seen in Fig. 13.

5.2. Predictive neural control

Based on the fact that the dynamics of the heat transfer fluid temperature can be represented by the NARMA model given by Eq. (22) and on the results from Section 4.1, the predictive control law adopts the following form:

$$u(k) = P(T_e^{sp} - T_e(k), T_e(k) - T_r(k), u(k-1), \ldots, u(k-4)) \quad (25)$$

Using the vector $I(k) = T_e^{sp} - T_e(k), T_e(k) - T_r(k))$ as the input pattern and considering four context neurones, the functional P is approximated using the RNN:

$$\tilde{u}(k) = \tilde{P}(T_e^{sp} - T_e(k), T_e(k) - T_r(k), R_1(k), \ldots, R_4(k), W_{\tilde{P}} \quad (26)$$

where $R_i(k) = \tilde{u}(k-i)\ \ i = 1, 2, 3, 4$. The used neural architecture had one hidden layer containing ten neurones and the sigmoidal activation functions were ranged in [0, 1].

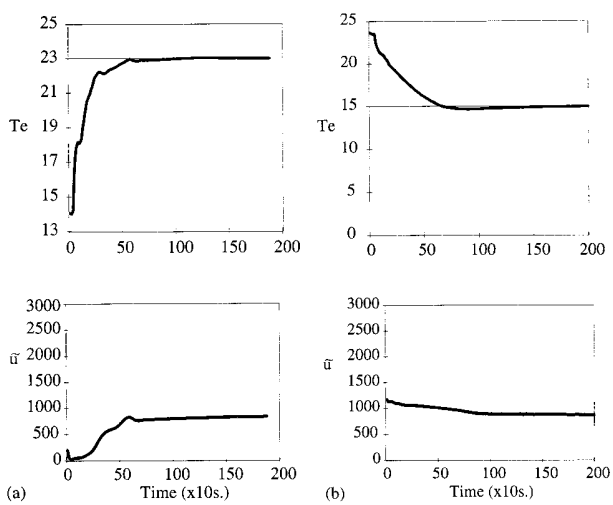

Fig. 14. Predictive neural controller performance: parameters initialised to $W_{\tilde{G}}(generalised)$. (a) $N_u = 0$, $H = 40$, $T_e^{sp} = 23°C$, (b) $N_u = 0$, $H = 70$, $T_e^{sp} = 15°C$.

After some tests conducted using a system model, the prediction horizon for the heating mode was set to 40 sampling times, i.e. 400 s. Because inverse and predictive control laws, Eqs. (24) and (26), have the same structure, the predictive control law can be initialised by parameters obtained in the previous section. Initialising the predictive control law given by Eq. (26) to the parameters $W_{\tilde{G}}(generalised)$, the steps described in Section 4.2 are carried out to reach the first control objective, $T_e^{sp} = 23°C$. At each sampling time, a NARMA parallel model [30] provided the approximated system output over the prediction horizon and the predictive controller parameters were updated in real time using the learning rule given by Eq. (21) and a learning rate fixed to 0.0005. The evolution of heat transfer fluid temperature and control signal are shown in Fig. 14(a). For the cooling desired objective control $T_e^{sp} = 15°C$ the prediction horizon has to be increased to 70 sampling times, i.e. 700 s. Fig. 14(b) shows the performance of the predictive controller, Eq. (26), in the cooling mode. The controller parameters are also updated in real time using the learning rule given by Eq. (21) and a learning rate fixed to 0.0005.

6. Discussion and conclusions

PID controllers have been widely used in industrial applications. The main advantages are its simplicity and the familiarity of users with their operation. However, this methodology has serious restrictions and in many cases it produces poor control performances. The results presented in the previous section have shown that the use of nonlinear control techniques based on appropriate modelling of process improves the control of the heat transfer fluid temperature in the jacketed chemical reactor, see Figs. 5, 13 and 14.

The success of the neural inverse control system depends exclusively on the weights used to initialise the controller acting in real time. The inverse neural controller built up in Section 5.1 whose parameters have been fixed to $W_{\tilde{G}}(specialised\_m)$ see Fig. 12, has identical behaviour to the inverse controller whose parameters are updating in real time, see Fig. 13, only some errors concerning to the off-set in the cooling mode are corrected. This implies that the performance of controller acting in real-time depends highly on the parameter set obtained in previous phases. Hence, if the weights of inverse neural controller would be set to random values, unstable or inefficient control of system could be expected. To guarantee an appropriate control of the real system, the inverse controller acting in real time must be initialised using previous knowledge about the dynamic behaviour that should be controlled.

In this context, the method to obtain initial weights is also important. The specialised learning using a system

model, phase 2, allows the controller to specialise in the desired control objective. However, previous simulations have shown that if the controller is only submitted to this training phase, optimal control performances cannot be ensured. The generalised learning, phase 1, i.e. approximation of system inverse dynamic with available patterns, permits the controller to gain information about the global inverse dynamic of system, which generally helps to obtain better performances. For instance, in Fig. 13(b) it is observed that the inverse controller has produced a large overshoot in the heat transfer fluid temperature which is not possible to reduce even if successive trails in the second training phase (specialised learning using a system model) are made. To reduce this temperature overshoot, it would have been necessary to initialise the controller with parameters having more information about the inverse dynamic system in the cooling mode. That is, the generalised learning would have been carried out with more patterns representing the cooling of heat transfer fluid temperature. It should be noticed that in this work, that learning phase is realised with only few patterns in this region, control actions higher than 900, see Fig. 10. Both specialised and generalised learning contribute to the efficient and suitable inverse neural controller performance on the real plant.

With regards to the predictive control system proposed in this paper it is interesting to point out that no nonlinear optimisation problem has to be solved at every sampling time. The use of a NN to calculate the predictive control action reduces the computational effort involved on those strategies. Hence, it is a predictive control schema relevant and suitable for real time applications. It should be noticed that the time spent for updating the RNN is lower than the requirement of time to solve the nonlinear optimisation problem. In last case there is a need to perform many updates at every sampling time to calculate the correct predictive control action that minimise the error function defined by Eq. (15). The parameters of the RNN are updated along the negative gradient direction of Eq. (15) and at every time only an updating must be carried out. The neural controller learns to realised the predictive control strategy along the time. The computational time requirements of predictive control systems currently used have prevented from testing their performance against the neural schema.

On the other hand, the use of a relationship between the current predictive control action and the measured variables of system allows off-line estimations of neural predictive controller parameters. This in fact facilitates the learning of the controller in real-time and helps to provide a more efficient control of the real system.

Both, inverse and predictive control schemes have their own characteristics, advantages and disadvantages. Nonetheless, the question whether inverse or predictive control strategies should be applied, merits further elaboration. Depending on the nature of dynamic systems, on the available information about the system and the requirement of users, either inverse or predictive control strategy will be more appropriate. However, there are some issues that could affect the choice of the approach to be used, which are briefly analysed.

As it has been mentioned above, the inverse control system depends highly on the parameters used to initialise the controller acting in real time. Since predictive control schema proposed in this work is an on-line method in which the control action is given by a NN, the initialisation of the network weights has also an effect on the neural predictive controller performance. However, it is necessary to point out that parameter initialisations play a less important role in the later case. This is due to the performance function used to update the predictive controller parameters, Eq. (15), which has more knowledge about the dynamic evolution of the plant than the error function used to update the inverse controller parameters, Eqs. (11) and (13). Eqs. (11) and (13) evaluate the instantaneous error measured at system output whereas Eq. (15) measures the behaviour of system along the prediction horizon. This allows that neural predictive controller can itself anticipate and coherently correct inappropriate control actions produced by unsuitable parameter initialisations. Another issue is the system model used to perform the inverse and predictive control strategies. The performance of predictive controller depends highly on the accuracy of the model because the calculation of control action is based on the dynamic behaviour of system along the prediction horizon predicted by the model. If errors occur in system outputs predicted, an unstable and unable predictive controller can be expected. In the inverse control schema a system model is also required in the second and third learning phase. However, in this case the dependence is lower because errors in the controller provoked by the use of an inappropriate model can be corrected in real time when the third learning phase is realised.

In the case study, i.e. the control of the heat transfer fluid temperature, both neural controllers have produced good performances. In the heating mode, predictive control was faster than inverse control, see Fig. 13(a)Fig. 14(a). In the cooling mode, the inverse controller has provided the fastest control, although the predictive controller was able to remove the overshoot in the temperature, see Fig. 13(b)Fig. 14(b).

Acknowledgements

The authors are most grateful to Dr Pedro Pimenta for his valuable comments and suggestions. I.M.G.

gratefully acknowledges the financial support of a grant from the TMR programme of the European Commission.

References

[1] A. Isidori, Nonlinear Control Systems, An introduction. 2nd ed., Springer-Verlag, Berlin, 1989.
[2] M. Agarwal, D.E. Seborg, Self-tunning controllers for nonlinear systems, Automatica 23 (1987) 209–214.
[3] M. Agarwal, D.E. Seborg, Multivariable nonlinear self-tunning controller, AIChE J. 33 (1987) 1379–1386.
[4] J.D. Morningred, B.E. Paden, D.E. Seborg, D.A. Mellichamp, An adaptive nonlinear predictive controller, Chem. Eng. Sci. 47 (1992) 755–762.
[5] J. Alvarez, E. Aranda, Nonlinear adaptive control of a chemical reactor using singular perturbation techniques. Dynamics and Control of Chemical Reactors (DYCORD + '92), Maryland, USA, 1992, pp. 125–247.
[6] J. Villermaux, C. Georgakis, Current problems concerning batch reactions, Int. Chem. Eng. 31 (1991) 434–440.
[7] K.J. Åström, T.J. McAvoy, Intelligent control, An overview and evaluation, in: D.A. White, D.A. Sofge (Eds.), Handbook of Intelligent Control, Van Nostrand Reinhold, New York, 1992.
[8] 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.
[9] D.C. Psichogios, L.H. Ungar, Direct and indirect model based control using artificial neural networks, Ind. Eng. Chem. Res. 30 (1991) 2564–2573.
[10] E.P. Nahas, M.A. Henson, D.E. Seborg, Nonlinear internal model control strategy for neural network models, Comput. Chem. Eng. 16 (1992) 1039–1057.
[11] J.L. Dirion, B. Ettedgui, M. Cabassud, M.V. Le Lann, G. Casamatta, Elaboration of a neural network system for semi-batch reactor temperature control: an experimental study, Chem. Eng. Process. 35 (1996) 225–234.
[12] B. Widrow, J. McCool, B. Medoff, Adaptive control by inverse modelling. Twelfth Asimolar Conference on Circuits, Systems and Computers, 1978.
[13] D. Psaltis, A. Sideris, A.A. Yamamura, A multi-layered neural network controller, IEEE Control Syst. Mag. 8 (1988) 17–20.
[14] B. Widrow, Adaptive inverse control. Proc. Second IFAC Workshop on Adaptive Systems in Control and Signal Processing, Lund Institute of Technology, Lund, Sweden, 1986, pp. 1–5.
[15] M.I. Jordan, Generic constraints on underspecified target trajectories. IJCNN Proc., IEE, New York, June 1989, pp. 217–225.
[16] K.S. Narendra, K. Pathasarthy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1990) 4–27.
[17] J. Tanomaru, S. Omatu, Process control by on-line trained neural controllers, IEEE Trans. Ind. Electron. 39 (1992) 511–521.
[18] Low Teck-Seng, Tong-Heng Lee, Hock-Koon Lim, A Methodology for neural network training for control of drives with nonlinearities, IEEE Trans. Ind. Electron. 39 (1993) 243–249.
[19] J.M. Martín Sánchez, Adaptive predictive control systems, EEUU Patent no. 4 197 576, 1976.
[20] J.W. Eaton, J.B. Rawlings, Model-predictive control of chemical processes, Chem. Eng. Sci. 47 (1992) 705–720.
[21] C.E. García, D.M. Prett, M. Morari, Model predictive control, theory and practice—a survey, Automatica 25 (1989) 335–348.
[22] D.W. Clarke, C. Mohtadi, P.S. Tuffs, Generalized predictive control—Part I. the basic algoritnm, Automatica 23 (1987) 137–148.
[23] D.W. Clarke, C. Mohtadi, P.S. Tuffs, Generalized predictive control—Part II. Extensions and interpretations, Automatica 23 (1987) 149–160.
[24] B.W. Bequette, Nonlinear predictive control using multi-rate sampling, Can. J. Chem. Eng. 69 (1991) 136–147.
[25] T. Pröll, M.N. Karim, Model-predictive pH control using real-time NARX approach, AIChE J. 40 (1994) 269–282.
[26] M.J. Willis, G.A. Montague, C. Massimo, M.T. Tham, A.J. Morris, Artificial neural network based predictive control. IFAC Advanced Control of Chemical Processes, Toulouse, France, 1991, pp. 261–266.
[27] C. Kambhampati, K. Warwick, Use of stochastic neural networks for process control, IFAC Dynamics and Control of Chemical Reactors (DYCORD + '92), Maryland, USA, 1992, pp. 111–116.
[28] D.G. Luenberger, Linear and Nonlinear Progamming, Addison-Wesley, Reading, MA, 1973.
[29] 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.
[30] I.M. Galván, J.M. Zaldívar, Application of recurrent neural networks in batch reactors: Part I. NARMA modelling of the dynamic behaviour of the heat transfer fluid temperature, Chem. Eng. Process. 36 (1997) 505–518.
[31] 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.
[32] M. Saerens, A. Soquet, A neural controller based on backpropagation algorithm. Technical report IRIDIA-NN 005, Université Libre of Bruxelles, 1989, p. 23.
[33] F.C. Chen, Y.H. Pao, Learning control with neural networks, Proc. IEEE Cont. Robot. Autom. 3 (1989) 1448–1453.