A predictive control scheme based on neural networks

Alejandro M. Suárez
Electronics Engineering Department, Federico Santa María Technical University, Valparaíso, Chile
Manuel A. Duarte-Mermoud
Electrical Engineering Department, University of Chile, Santiago, Chile, and
Danilo F. Bassi
Computer Science Department, University of Santiago, Santiago, Chile

Abstract

Purpose – To develop a new predictive control scheme based on neural networks for linear and non-linear dynamical systems.

Design/methodology/approach – The approach relies on three different multilayer neural networks using input-output information with delays. One NN is used to identify the process under control, the other is used to predict the future values of the control error and finally the third one is used to compute the magnitude of the control input to be applied to the plant.

Findings – This scheme has been tested by controlling discrete-time SISO and MIMO processes already known in the control literature and the results have been compared with other control approaches with no predictive effects. Transient behavior of the new algorithm, as well as the steady state one, are observed and analyzed in each case studied. Also, online and offline neural network training are compared for the proposed scheme.

Research limitations/implications – The theoretical proof of stability of the proposed scheme still remains to be studied. Conditions under which non-linear plants together with the proposed controller present a stable behavior have to be derived.

Practical implications – The main advantage of the proposed method is that the predictive effect allows to suitable control complex non-linear process, eliminating oscillations during the transient response. This will be useful for control engineers to control complex industrial plants.

Originality/value – This general approach is based on predicting the future control errors through a predictive neural network, taking advantage of the NN characteristics to approximate any kind of relationship. The advantage of this predictive scheme is that the knowledge of the future reference values is not needed, since the information used to train the predictive NN is based on present and past values of the control error. Since the plant parameters are unknown, the identification NN is used to back-propagate the control error from the output of the plant to the output of the controller. The weights of the controller NN are adjusted so that the present and future values of the control error are minimized.

Keywords Cybernetics, Control systems, Neural nets

Paper type Research paper

1. Introduction

Nowadays, the quality requirements for automatic controllers has been increasing due to the greater complexity of industrial plants and because of the necessity of meeting

The research contained in this paper was partially supported by CONICYT under grants FONDECYT 1970351 and FONDECYT 1980361, and by Universidade Santiago, grant DICYT 06961BA.

increasingly high quality standards for goods and services. At the same time, the availability of computing resources has dramatically increased. This has resulted that approaches that are computationally expensive can now be used to control complex processes. There have been important developments of controllers based on more exact models, particularly those using predictive models that have been successfully applied in many industrial plants (Peterson *et al.*, 1989; Lundstrom *et al.*, 1995; Draeger *et al.*, 1995). One important advantage of this kind of approach is its ability to deal with restrictions on control and internal variables. In many applications of predictive modeling, a linear model is used for predicting the process behavior within the horizon of interest (Cutler and Ramaker, 1980; García *et al.*, 1989). However, since the majority of real processes have some non-linear behavior, there have been several efforts in order to extend the predictive control techniques to incorporate non-linear models (Peterson *et al.*, 1989; Brengel and Seider, 1989).

The most difficult part in the achievement of a non-linear predictive control is the derivation of the mathematical model. In many cases it is still impossible to obtain an acceptable model of the process based on physical principles, due to its inherent complexity. A more promissory way to face these problems is to use a neural network as a black-box non-linear model of the behavior of the processes (Thibaut and Grandjean, 1992; MacMurray and Himmelblau, 1992). Such a neural network models can be adjusted (trained) from plant input/output data. This data, required for the neural network training, can be obtained straightforward from special open loop experiments performed on the plant. However, in many cases and for practical reasons, the plant is driven by conventional PID controllers providing stability and basic control. In Draeger *et al.* (1995), it is shown that the measures of input/output variables of a plant, handled by a linear controller, can provide very fine data for training neural networks. This procedure is quite practical since the plant is always under control. Also, this method is more effective, since the input excitation is much more similar to what is expected from a non-linear controller than the one obtained by doing experiments in open loop with no control.

Lately, several authors have used prediction of some signals based on neural networks as a part of control schemes (Ydstie, 1990; Saint-Donald *et al.*, 1991) for nonlinear systems, and even some stability results have been informed (Eaton *et al.*, 1995). Predictive effects have also been used to compensate time-delays in nonlinear systems (Tan and de Kayser, 1994a, b, c; Wan, 1990).

In this paper, a new control scheme that uses predictive effects for adjusting the controller is proposed, considering the results developed in Suárez (1998). The main idea consists in training a neural controller, taking into account the future errors between the process output and the desired reference, using a predictive network.

In Section 2, the control approach is explained indicating the main features. Section 3 contains the mathematical description of the weight adjustments in the controller network. Section 4 describes the procedures for training and using the predictive neural network. Finally, in Section 6 the main conclusion of the mathematical analysis performed are presented.

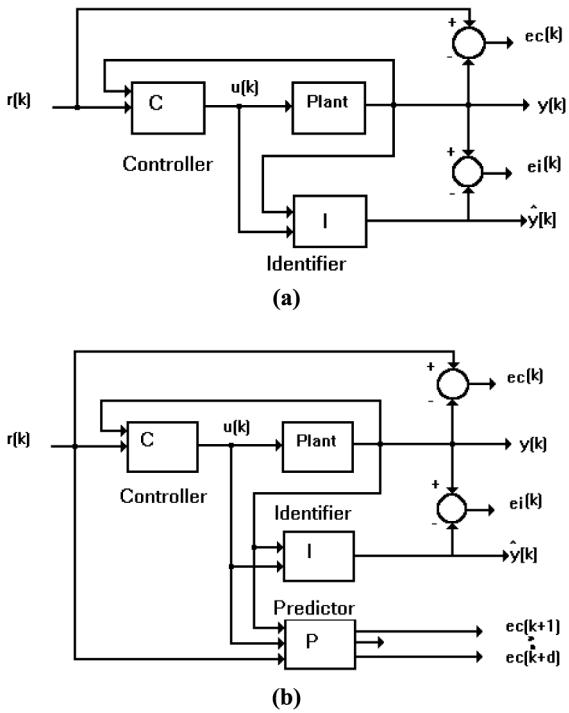
2. Description of the new control approach

The adaptive systems have the advantage to adapt it selves to environmental changes. For this reason it is believed that the direct model control scheme, as presented in

Suárez (1998), suitably modify is a promising control scheme for complex pants. It is important also to mention that the PID approach is very robust for most applications. This is explained by the fact that the control signal is synthesized from information about present, past and future errors. Using this idea, it is attempted to improve the direct control approach with model, by optimizing the neural network controller with respect to the present as well as to the past and future errors. In order to obtain future errors, a predictive network with d step ahead or predictive horizon is proposed.

In this section, the proposed neural network control scheme is described, considering its general structure and the control error predictive effects. Since in this study a comparison is made between the multivariable neural control configurations with and without predictive effects, both schemes are described in what follows. The reader can found a more detailed presentation of the control method in Suárez (1998).

Figure 1(a) shows the MIMO neural control scheme without predictive effects for a plant, where $u(t) \mathscr{R}^p$ and $y(t) \in \mathscr{R}^m$ are the input and output variables, respectively. The NN denoted as I correspond to the identifier which give us an estimation $\hat{y}(t)$ of output variables $y(t)$, based on input/output information collected online. The identifier parameters are adjusted using static backpropagation of the identification error $e_i(t) = y(t) - \hat{y}(t) \in \mathscr{R}^m$, with the aim of minimizing some criteria function of the identification error. The NN denoted as C represents the controller and supply the control variable $u(t) \in \mathscr{R}^m$ to be applied to the plant. The controller parameters are



Notes: (a) Without prediction, (b) With prediction

Figure 1.
Neural control schemes using a model of the plant

adjusted using dynamic backpropagation of the control error $e_c(t) = y(t) - r(t) \in \mathscr{R}^m$, through the identified model (since the plant is unknown) and with the objective of minimizing some criteria function of the control error. (Figure 1(a)).

Figure 1(b) shows the MIMO neural control scheme with predictive effects for a plant, where $u(t)\mathscr{R}^p$ and $y(t) \in \mathscr{R}^m$ are the input and output variables, respectively. The scheme is similar to that presented previously but with a predictive effect introduced by another NN denoted by P which supply an estimation of the future control error (prediction error). The controller parameters are adjusted using dynamic backpropagation of the control errors up to d step ahead, $e_c(t), e_c(t+1), \ldots, e_c(t+d)$, through the identified model since the plant is unknown, such that the following criterion function is minimized:

$$J(t) = \frac{1}{2}\sum_{i=0}^{i=d} \lambda_i e_c^2(t+i), \quad \text{with } \lambda_i = \frac{1}{1+d}$$

In Figure 1(b), it is shown the direct control with model approach and with a predictive network.

3. Algorithm for error propagation

The aim of the control configuration shown in Figure 1(b) is that the control error $e_c(k)$ be minimum or zero after a certain time. For training the control network C it is required a cost function associated to this error, that should be minimized. In what follows this cost function is described and it is shown how to generate the errors needed for this adjustment.

3.1 Generation of the error

The cost function used in this study is defined as:

$$f(.) = \frac{1}{2}\bar{e}(.) \tag{1}$$

where

$$\bar{e}(.) = \lambda_0 e(k)^2 + \lambda_1 e(k+1)^2 + \cdots + \lambda_d e(k+d)^2 \tag{2}$$

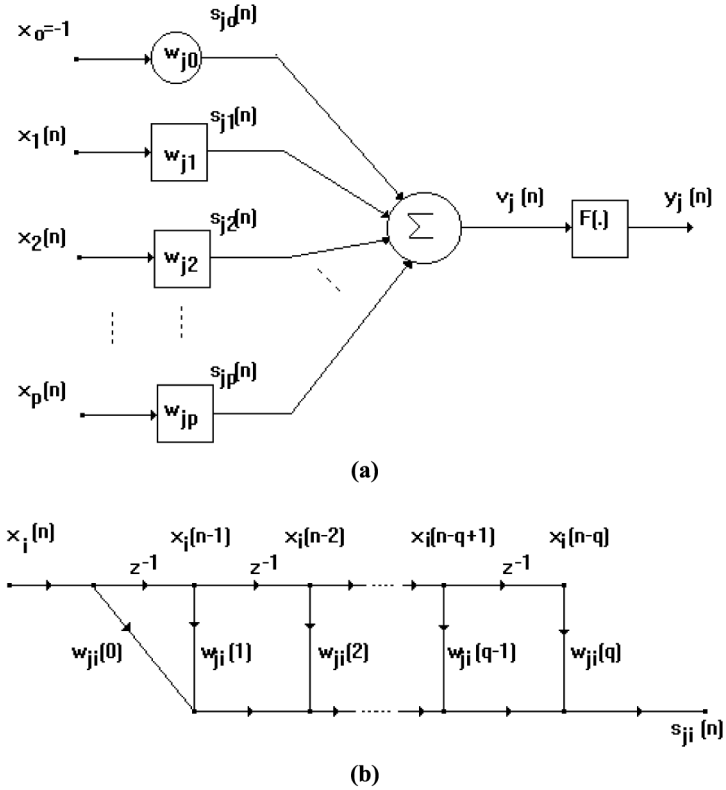
with

$$\lambda_i = \frac{1}{d+1} \quad \text{for } i = 0, 1, 2, \ldots, d$$

The error $\bar{e}(.)$ is a weighted mean of the present and future errors and can be modified by choosing suitable λ_i.

3.2 Spatial-temporal model of a neuron

In the classical neuron model a single synaptic weight is associated to each input. For temporal processing, this simple weight structure has to be modified accordingly (Haykin, 1994). A general approach for representing the temporal behavior is to model each synapse as a FIR filter, like the one shown in Figure 2(a), where w_{ij} denotes the weight vector of ith synapse associated to jth neuron, where $i = 1,2,\ldots,p$. The weight w_{jo}, which is connected to a dummy fixed input $x_0 = -1$ representing the threshold θ_j.



Notes: (a) Dynamic neuron model, including FIR synaptic filters. (b) Flow diagram of a FIR synaptic filter

Figure 2.

Figure 2(b) shows the flow diagram of the FIR, where z^{-1} represents the unit delay operator.

According to this model, the signal $s_{ji}(n)$ at the output of the ith synapse of the jth neuron is represented as a linear combination of the delayed input values $x_i(n)$, as shown by the following discrete convolution:

$$s_{ji}(n) = \sum_{l=0}^{q} w_{ji}(l)x_i(n-l) \tag{3}$$

where n is the discrete-time variable. Thus, by adding up the contributions of the whole set of p synapses, the output $y_i(n)$ of jth neuron can be depicted by the following equations:

$$v_j(n) = \sum_{i=0}^{p} s_{ji}(n) \tag{4}$$

$$y_j(n) = F(v_j(n)) \tag{5}$$

where $v_j(n)$ is the activation potential of the jth neuron, and $F(.)$ is the non-linear activation function of such neuron. It can be shown that if the weight vector w_{ji} and state vector $x_i(n)$ are replaced by scalars w_{ji} and x_i, respectively, and the inner-product operation is replaced by the ordinary product, the dynamic model depicted by equations (4) and (5) is reduced to the static model.

3.3 Relationship between networks

In the backpropagation of the error, there are components from both the identifier network and the controller network (Figure 3). One crucial step in the backpropagation is when both networks are joined. For that reason we shall analyze separately each situation.

In Figure 3 the following parameters are considered: n_u, number of plant inputs; n_s, number of plant outputs.

Between the output neurons of the controller network and each neuron in the first hidden layer of the identifier network we use dynamical synapse, as shown in Figure 2(a), where: $x_i \in \{u_1, \ldots, u_{n_u}\}$, output of output layer salida of controller network, $r \in \{0, \ldots q\}$, delays associated to the x_i; $F[.]$, activation function; y_j, output of j ith neuron of input layer of identifier network. Formulae relating y_j with all the inputs of the identifier network which come from the outputs of the controller network are as follows:

$$y_j(n) = F(v_j(n)) \tag{6}$$

$$v_j(n) = \sum_{i=0}^{p} \sum_{r=0}^{q} w_{ji}(r) x_i(n - r) \tag{7}$$

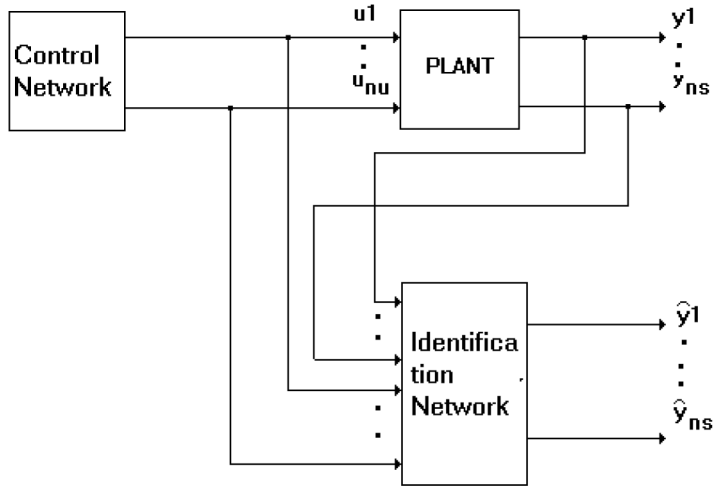


Figure 3.
Block diagram of controller network and identifier network

3.4 Backpropagation of the error

In order to observe how the error is backpropagated through the network, when a weight in the controller network is updated (Haykin, 1994; Wan, 1990), we use the signal flow diagram of Figure 4, where $y_d(k)$ is the desired value of the output, at time k.

The equations that are involved in $w_{il}(n)$ weight adjustment are the following:

Cost criterion:

$$\xi(n) = \sum_{k=1}^{n0} \bar{e}_k(n) \tag{8}$$

where n_0 is the number of neurons in the output layer of the identifier network.

Since there are also delayed inputs entering to the identifier network, the minimized cost function is considered as $\xi(n)$ computed over all the time indexes involved in the backpropagation path. Then the global cost index, which is minimized over every time index in the present and the past, is:

$$\xi_{\text{total}} = \sum_{m=n-\text{delays}}^{n} \xi(m) \triangleq \sum_{n} \xi(n) \tag{9}$$

Each weight adjustment is performed as:

$$w_{il}(n+1) = w_{il}(n) + \Delta w_{il}(n) \tag{10}$$

where

$$\Delta w_{il}(n) = -\eta \frac{\partial \xi_{\text{total}}}{\partial w_{il}(n)} \tag{11}$$

rearranging equations (1) and (2) associated with the error, we have:

$$\bar{e}_k(n) = \frac{1}{2(d+1)} \sum_{m=n}^{n+d} (y_d(m) - y_k(m))^2 \tag{12}$$

and observing the output layer in the flow diagram of Figure 4 we obtain:

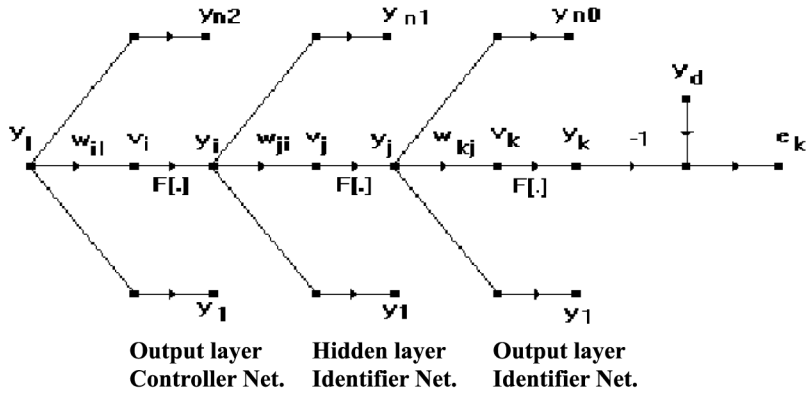


Figure 4.
Flow diagram of neuron signals

$$y_k(n) = F(v_k(n)) \tag{13}$$

and

$$v_k(n) = \sum_{j=0}^{n1} w_{kj} y_j(n) \tag{14}$$

where n_1 is the number of neurons in the first hidden layer of the identifier network.

Following with the analysis of the hidden layer of the identifier network, as shown in Figure 4, it yields:

$$y_j(n) = F(v_j(n)) \tag{15}$$

Using the equation (7), but replacing p by n_2 we obtain:

$$v_j(n) = \sum_{i=0}^{n2} \sum_{r=0}^{q} w_{ji}(r) y_i(n-r) \tag{16}$$

where n_2 is the number of neurons in the output layer of the controller network and q is the number of delays.

Looking at the output layer of controller network (Figure 4) we can conclude that:

$$y_i(n) = F(v_i(n)) \tag{17}$$

and

$$v_i(n) = \sum_{l=0}^{n3} w_{il}(n) y_l(n) \tag{18}$$

where n_3 is the number of neurons in the hidden layer of the controller network.

In order to update the weight of equation (10) it is necessary to compute the variation of the global cost index with respect to that weight. By using the derivative chain rule, we have that:

$$\frac{\partial \xi_{\text{total}}}{\partial w_{il}(n)} = \frac{\partial \xi_{\text{total}}}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial v_i(n)} \frac{\partial v_i(n)}{\partial w_{il}(n)} \tag{19}$$

by differentiating equations (17) and (18), and replacing the result in equation (19) we obtain:

$$\frac{\partial \xi_{\text{total}}}{\partial w_{il}(n)} = \frac{\partial \xi_{\text{total}}}{\partial y_i(n)} F'(v_i(n)) y_l(n) \tag{20}$$

Defining as local gradient the following term:

$$\underset{i}{=}\delta(n) = -\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} F'(v_i(n)) \tag{21}$$

and replacing into equation (20), it yields:

$$\frac{\partial \xi_{\text{total}}}{\partial w_{il}(n)} = -\overset{=}{\underset{i}{\delta}}(n) y_l(n) \tag{22}$$

In equation (21), the undefined partial derivative can be computed again, using the chain rule for all the past n values, since at this stage, the delayed inputs of the dynamic neuron are now involved. That is:

$$\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} = \sum_{j=0}^{n1} \sum_{n} \frac{\partial \xi(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial y_i(n)} \tag{23}$$

By differentiating the equation (15) and replacing in equation (23) we obtain:

$$\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} = \sum_{j=0}^{n1} \sum_{n} \frac{\partial \xi(n)}{\partial y_j(n)} F'(v_j(n)) \frac{\partial v_j(n)}{\partial y_i(n)} \tag{24}$$

Defining as new local gradient the following term:

$$\bar{\delta}_j(n) = -\frac{\partial \xi(n)}{\partial y_j(n)} F'(v_j(n)) \tag{25}$$

replacing into equation (24) yields:

$$\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} = -\sum_{j=0}^{n1} \sum_{n} \bar{\delta}_j(n) \frac{\partial v_j(n)}{\partial y_i(n)} \tag{26}$$

Using the result of Appendix (equation (A3)), we obtain:

$$\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} = -\sum_{j=0}^{n1} \sum_{n=r}^{q+r} \bar{\delta}_j(n) w_{ji}(n-r) \tag{27}$$

Defining the following vectors:

$$\Delta_j(n) = \left[\bar{\delta}_j(n) \quad \bar{\delta}_j(n+1) \quad \ldots \quad \bar{\delta}_j(n+q) \right]^{\mathrm{T}} \tag{28}$$

$$\boldsymbol{w}_{ji} = \left[w_{ji}(0) \quad w_{ji}(1) \quad \ldots \quad w_{ji}(q) \right]^{\mathrm{T}} \tag{29}$$

and replacing into equation (27) we have:

$$\frac{\partial \xi_{\text{total}}}{\partial y_i(n)} = -\sum_{j=0}^{n1} \Delta_j^{\mathrm{T}}(n) \boldsymbol{w}_{ji} \tag{30}$$

As in the previous analysis, the partial undefined derivative of equation (25) can be decomposed by the chain rule, which yields:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_{k=1}^{n0} \frac{\partial \xi(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{31}$$

Differentiating equations (13) and (14) and replacing the results in equation (31) yields:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_{k=1}^{n0} \frac{\partial \xi(n)}{\partial y_k(n)} F'(v_k(n)) w_{kj} \tag{32}$$

Defining as new local gradient the following term:

$$\delta_k(n) = -\frac{\partial \xi(n)}{\partial y_k} F'(v_k(n)) \tag{33}$$

and replacing in equation (32), we have:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_{k=1}^{n0} \delta_k(n) w_{kj}(n) \tag{34}$$

In equation (32) the undefined partial derivative can be computed, using again the chain rule, yielding as result:

$$\frac{\partial \xi(n)}{\partial y_k(n)} = \frac{\partial \xi(n)}{\partial \overline{e}_k(n)} \frac{\partial \overline{e}_k(n)}{\partial y_k(n)} \tag{35}$$

with equations (8) and (12) we obtain:

$$\frac{\partial \xi(n)}{\partial y_k(n)} = \frac{-1}{d+1} \sum_{t=n}^{n+d} e_k(t) \tag{36}$$

By means of this set of equations, it is possible to adjust any weight of the controller network. It can be noted that the future errors are involved in the local gradient of equation (33) by means of equation (36). Furthermore, in equation (30) it is observed that the local gradient vector is composed by local scalar future gradients associated with y_i.

Clearly the exact time frame used for the weights adjustment is not important. A possible solution for the causality problem would be to adjust weight based only in past and present values of local scalar gradients. Therefore, by using $\Delta_j(n-q)$ in equation (30) and replacing in equation (21), we get:

$$\underset{i}{=}\delta(n-q) = -F'(v_i(n-q)) \sum_{j=0}^{n1} \Delta_j^{\mathrm{T}}(n-q) w_{ji} \tag{37}$$

It is noted that the state $y_l(n-q)$ must be stored in order to facilitate the computation of the product $\bar{\delta}_i(n-q)y_l(n-q)$ for adapting the weight associated with the connection of the jth neuron with the jth neuron.

As a summary we have the following procedure:

- In order to obtain the value of the output $y_k(n)$, the following equations must be evaluated in the following order: equations (18), (17), (16), (15), (14) and (13).
- To update a weight in the controller network, for example $w_{il}(n)$, the following equations must be evaluated, keeping the right order: equations (36), (33), (34), (25), (30), (21), (22), (11) and (10).

4. Description of the predictive network

In the proposed scheme, an error signal from the plant output to the output of controller network is propagated through the plant identifier network. The propagated error signal is the average of actual error and future errors of the plant: this is the predictive effect. The control error is computed from the difference between the plant output and the corresponding reference signal. In order to obtain the future errors a predictive neural network is used. It is important to note that the future reference input needs not to be know in advance.

4.1 Definition of vectors and matrices

The matrices used for training and running the predictive error neural network are shown in what follows. Furthermore, other parameters involved in the prediction process are also defined:

N = number of predictions.

q = number of delay taps at the process input.

p = number of delay taps at the process output and reference.

n_e = number of inputs.

n_s = number of outputs.

DE = matrix of plant input data of dimension $n_e \times (N + q + 1)$.

DS = matrix of plant output data of dimension $n_s \times (N + p + 1)$.

DR = matrix of reference data of dimension $n_s \times (N + p + 1)$.

DSv = matrix of known output data of dimension $n_s \times N$.

DRv = matrix of known reference data of dimension $n_s \times N$.

DEE = matrix of plant input data for network training of dimension $n_e \times (q + 1)$.

DSE = matrix of plant output data for network training of dimension $n_s \times (p + 1)$.

DRE = matrix of reference data for network training of dimension $n_s \times (p + 1)$.

DEJ = matrix of plant input data for evaluating the network of dimension $n_e \times (q + 1)$.

DSJ = matrix of plant output data for evaluating the network of dimension $n_s \times (p + 1)$.

DRJ = matrix of reference data for evaluating the network of dimension $n_s \times (p + 1)$.

The known error prediction matrix used for training the network is generated by:

$$E = \mathrm{DRv} - \mathrm{DSv} \tag{38}$$

The pattern matrix for training the network at each iteration is:

$$\boldsymbol{P}_E = \begin{bmatrix} \mathrm{DEE} \\ \mathrm{DSE} \\ \mathrm{DRE} \end{bmatrix} \tag{39}$$

The pattern matrix for evaluating the network at each iteration is:

$$\boldsymbol{P}_J = \begin{bmatrix} \mathrm{DEJ} \\ \mathrm{DSJ} \\ \mathrm{DRJ} \end{bmatrix} \tag{40}$$

and the vector of prediction errors produced by the network is given by:

$$\bar{E} = \left[e_1(k+1)\cdots e_{ns}(k+1)e_1(k+2)\cdot e_{ns}(k+2)\cdots e_1(k+N)\cdots e_{ns}(k+N)\right]^T \tag{41}$$

4.2 Data usage

During the training phase, the last N process data is used as known values for the predictive network and at each iteration this data is updated. While evaluating, this last N data is used as the input vector network, producing at the output of the network the future N process errors. The procedure of training and evaluating the network is performed at each iteration.

5. Simulations results

In this section, the proposed control scheme, direct neural predictive control using a model of the plant, is tested over several mathematical simulated processes. This setting allows us to get open loop training data.

5.1 Simulation tests

SISO and MIMO linear and nonlinear processes have been selected for testing the performance of the proposed control configuration. The models of the processes used in the simulations are defined by equations (42)-(47) in terms of either their transfer functions or their input-output representations. In order to facilitate the result comparisons, the nonlinear cases chosen are standard processes found in literature (Narendra and Parthasarathy, 1990, 1991).

Linear SISO processes:

$$\text{First order } H(z) = \frac{0.00995}{z - 0.99} \tag{42}$$

$$\text{Second order } H(z) = \frac{0.00487z + 0.00474}{z^2 - 1.91z + 0.923} \tag{43}$$

Nonlinear SISO processes:

$$\text{First order } y_p(k+1) = \frac{y_p(k)}{1+y_p(k)^2} + u^3(k) \tag{44}$$

$$\text{Second order } y_p(k+1) = \frac{y_p(k)y_p(k-1)[y_p(k)+2.5]}{1+y_p^2(k)+y_p^2(k-1)} + u(k) \tag{45}$$

Linear MIMO process:

$$H(z) = \begin{bmatrix} \frac{0.00995}{z-0.99} & \frac{0.00995}{z-0.99} \\ \frac{0.00995}{z-0.99} & \frac{0.00995}{z-0.99} \end{bmatrix} \tag{46}$$

Nonlinear MIMO process:

$$\begin{bmatrix} y_{p1}(k+1) \\ y_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} \frac{y_{p1}(k)}{1+y_{p2}^2(k)} \\ \frac{y_{p1}(k)y_{p2}(k)}{1+y_{p2}^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \tag{47}$$

5.2 Neural networks characteristics

In all simulations feedforward neural networks are used for control, identification and prediction, with linear activation functions in the output layer and hyperbolic tangent (tan*h*) activation functions in the hidden layers. This class of neural networks is standard and easy to implement to achieve real time performance.

It is know that feedforward neural networks, with at least one hidden layer having sufficient neurons, are able to approximate any nonlinear function with arbitrary accuracy (Hornik *et al.*, 1990). For the nonlinear cases to be studied we consider networks with one input layer, two hidden layers and one output layer. While it may be sufficient to have one hidden layer, with two it is possible to get lesser error in the training process.

In describing the feedforward neural networks we use the following definitions:

$$\boldsymbol{N}^{N}_{\boldsymbol{i}_1,\boldsymbol{i}_2,\ldots,\boldsymbol{i}_{N+1}}$$

where: $\mathbf{N}$ is the number of layers (with synaptic weights); $\mathbf{i}_1$ is the number of inputs; $\mathbf{i}_{N+1}$ is the number of outputs; $\mathbf{i}_2,\mathbf{i}_3,\ldots,\mathbf{i}_N$ number of neurons at each hidden layer $(N-1)$.

For example, $N^3_{5,20,10,1}$ represents a networks with 3 layer of synaptic weights, 5 inputs, 20 neurons in the first hidden layer, 10 neurons in the second layer and 1 output neuron.

Representing the process input by $u(k)$ and the process output by $y(k)$, with n and m as delays, the neural network function description has the following form:

$$y(k+1) = \boldsymbol{N}[y(k), y(k-1), \ldots, y(k-m+1); u(k), u(k-1), \ldots, u(k-n+1)] \tag{48}$$

The controller and predictor networks were trained online, while the identifier network was adjusted offline. For training the identifier network at every case, we used

a random variable, uniformly distributed in the interval $[-2, 2]$, assuming that control signal will be in that interval. In addition, for non-linear SISO processes, training was done with the following signal: $\sin(t) + \sin(5t) + \sin(10t)$. This allows to adapt the networks to similar signals that will be found in future reference, decreasing the training time with respect to random signal. The reference signal used for testing the performance is always the same $\sin(2t)$.

5.3 Control configuration

For controlling the output of the different processes (equations (42)-(47)), two control schemes were considered (Figure 1). Both schemes use a dynamic actualization method and a single controller network. The main difference between both methods is the type of error signal used for training the controller network.

In the schemes studied in this work (Figure 1), the error signal $e_c(k)$ is propagated from plant output to controller network output, through a plant identifier network (Murray-Smith *et al.*, 1992). The difference between scheme (a) and (b) shown in Figure 1, is in its back propagated error, since in the scheme (a), the actual error between the plant output and the reference is propagated, while in scheme (b) the actual error and future errors of plants (predictive effect) are back propagated. In order to determine the future errors, a predictive network is used. The implementation of these control schemes was done under Simulink of Matlab environment, and the neural networks were developed as a *S*-function of Simulink.

5.4 Tests and results

Each process was tested with both schemes, already described. The transient as well as the steady state phase were observed and compared, changing the learning gain of the controller network and the prediction horizon of the predictive network.

5.4.1 Linear first-order process. The parameters of the networks used for controlling the first-order system defined by equation (42) are given in Table I.

The off-line training of the identifier network had 5,000 epochs, where each epoch consisted of 300 pairs of data.

Controller network:	Structure:	$\boldsymbol{N}^1_{5,1}$
	Inputs:	$[r(k), r(k-1), r(k-2), y(k), y(k-1)]$
	Sampling time:	0.01
	Learning gain:	0.5
	Training:	5,000 online iterations
Identifier network:	Structure:	$\boldsymbol{N}^1_{5,1}$
	Inputs:	$[u(k), u(k-1), u(k-2), y(k), y(k-1)]$
	Sampling time:	0.01
	Learning gain:	0.1
Predictive network:	Structure:	$\boldsymbol{N}^1_{9,1}$
	Inputs:	$[u(k), u(k-1), u(k-2), y(k), y(k-1), y(k-2), r(k), r(k-1), r(k-2)]$
	Sampling time:	0.01
	Learning gain:	0.01
	Prediction steps:	3
	Training:	5,000 online iterations

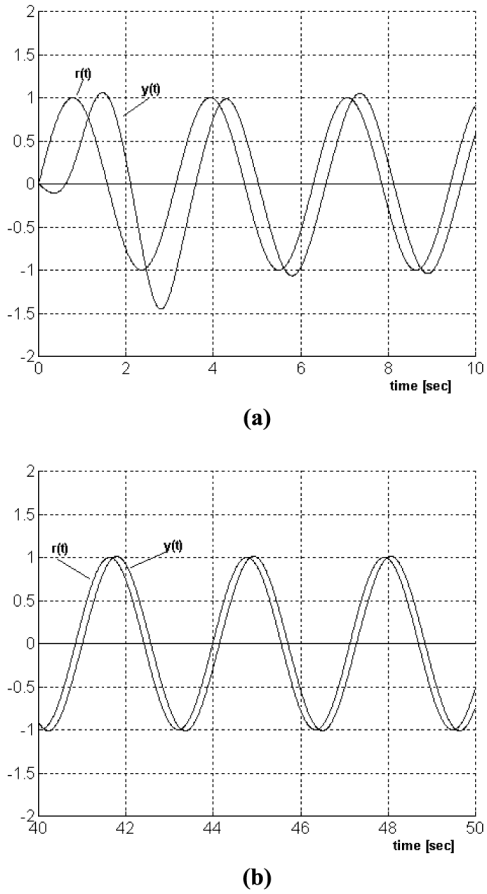
Table I.

For the scheme without predictive network, in Figure 5 are shown the initial part and ending part of the process output, during training phase of controller network. The results of the scheme with predictive network are shown in Figure 6.

From Figures 5 and 6 it is observed that the predictive effect does not significantly improve the controlled system behavior. This is because the plant is a simple linear system.

Several other simulations were performed when the learning gain of the controller network was changed. From there it was observed that for both, transient and steady stages, the convergence speed increases when the learning gain is augmented. It was also studied in the case when the prediction horizon of the predictive network is changed, keeping fixed the learning gain of controller network. It was observed that increasing the prediction horizon only affects the transient stage, making it slower.

5.4.2 Linear second-order process. The parameters of the networks used for controlling the second-order plant described by equation (43) are the same that those used for the linear first-order process shown in the previous section.



Notes: (a) Initial part, (b) Final part

Figure 5.
Output of first-order linear process, with no predictive network

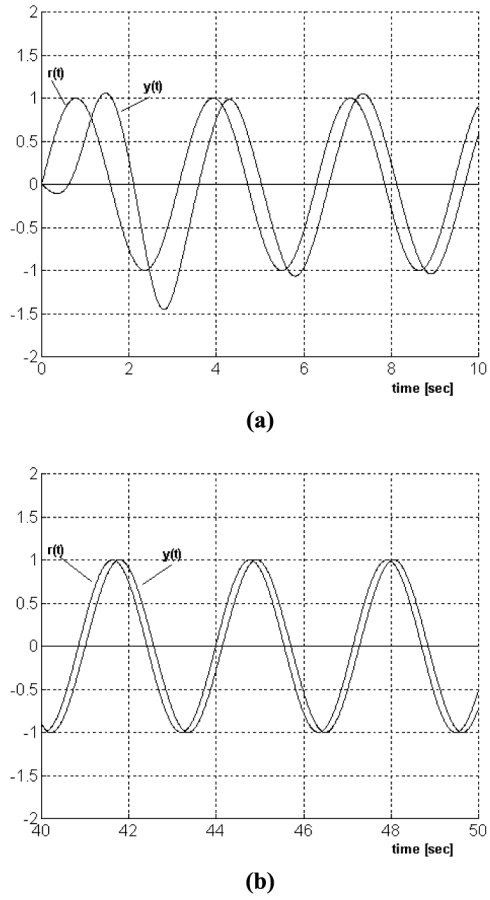


Figure 6.
Output of first-order linear process, with use of predictive network

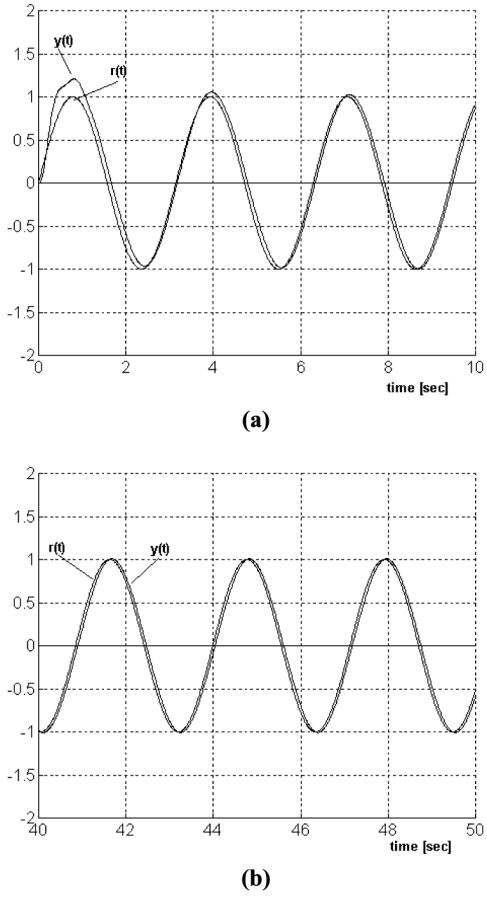
Notes: (a) Initial part, (b) Final part

For the control scheme without predictive network, Figure 7 displays the initial and final stages of the process output, while training the controlling network. For the scheme with predictive network, the results are displayed in Figure 8.

From Figures 7 and 8, when adding the predictive action, no significant changes are observed. Again this is due to that fact that the plant is a simple linear system.

From other extensive simulations done, it can be shown that the results obtained for the first-order linear system are also obtained for the linear second-order case. This means that the convergence speed increases when the learning rate does, and increasing the prediction horizon, while keeping fixed the learning rate of controller network, modify only the transient stage, making it slower.

5.4.3 Non linear first-order process. The parameters of networks used for controlling the nonlinear process defined by equation (44) are given in Table II.



Notes: (a) Initial part, (b) Final part

Figure 7.
Output of second-order linear process, without predictive network

The training phase of the identifier network took 10,000 epochs, where each epoch had 300 pairs of data. The first 5,000 epochs had random inputs while the last 5,000 epochs had sums of sinusoid signals as inputs.

For the control scheme without predictive network, Figure 9 shows the initial and final stages of the process output, during the training phase of controlling network. The results for the control scheme with predictive network are shown in Figure 10.

Extensive simulations done but not shown here for space reasons, show that when the learning rate of controller network increased, the system becomes less stable. With the predictive effect, it becomes more stable. When the number of predictions or the predictive horizon increase, while keeping fixed the learning rate of the controller network, the system become more stable in both transient and steady state stages.

Observing Figures 9 and 10 we can conclude that the predictive effect noticeably diminish the oscillations in the system response.

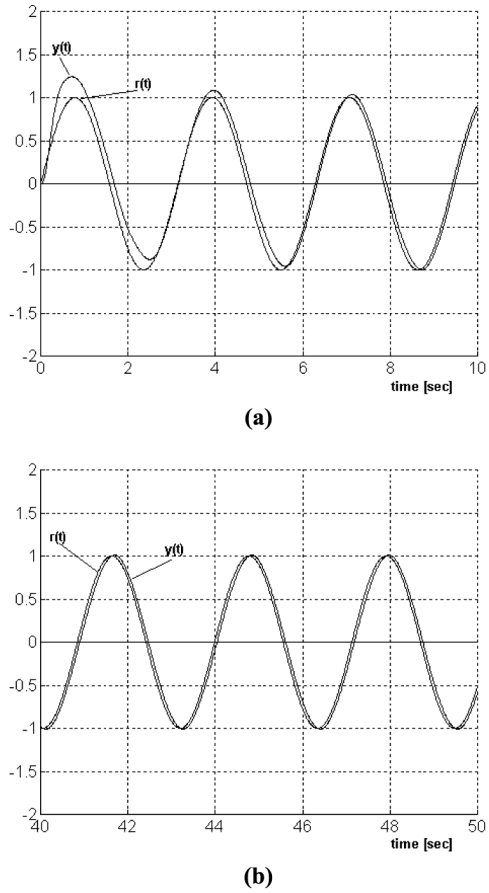


Figure 8.
Output of second-order linear process, with use of predictive network

Notes: (a) Initial part, (b) Final part

5.4.4 Non-linear second-order process. The parameters of the networks used for controlling and identifying the nonlinear second-order plant defined by equation (45) are identical to those used for the nonlinear first-order process presented in the previous section. However, the predictive network is different, and its parameters are given in Table III

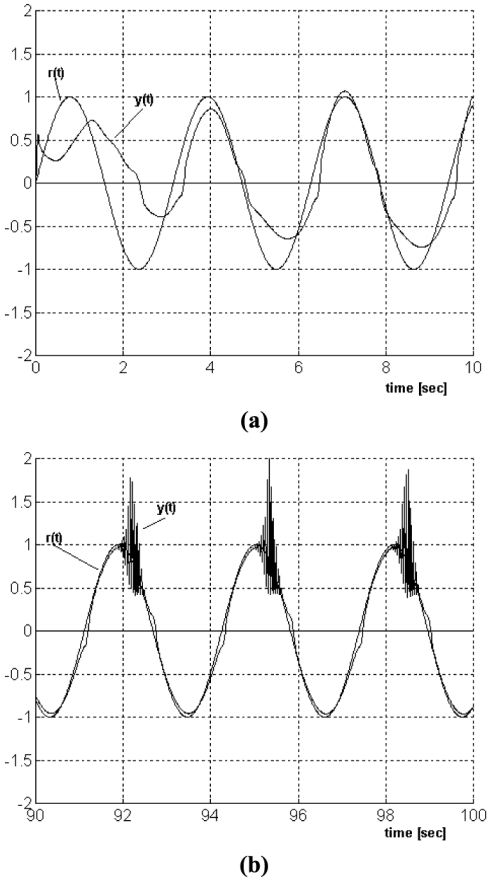
The control scheme without predictive network produces an oscillatory behavior of the process, during the training phase of controller network, as shows Figure 11, for the initial stage. The steady state response is not shown since it is awful. The results for the control scheme with predictive network are much better as shown in Figure 12.

The advantage of using a control scheme with predictive affects is clearly appreciated from Figures 11 and 12, for a nonlinear second-order plant. This advantage is more evident as long as the non-linearities and order of the plant increase.

Other simulations performed when the learning rate of the controller network is changed, show similar results to those obtained for the first-order non-linear system.

Controller network:	Structure:	$N^3_{5,20,10,1}$
	Inputs:	$[r(k), r(k-1), r(k-2), y(k), y(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.1
	Training:	10,000 online iterations
Identifier network:	Structure:	$N^3_{5,20,10,1}$
	Inputs:	$[u(k), u(k-1), u(k-2), y(k), y(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
Predictive network:	Structure:	$N^3_{5,20,10,1}$
	Inputs:	$[u(k), u(k-1), u(k-2), y(k), y(k-1), y(k-2), r(k), r(k-1), r(k-2)]$
	Sampling time:	0.01
	Learning rate:	0.01
	Prediction steps:	3
	Training:	10,000 online iterations

Table II.



Notes: (a) Initial part, (b) Final part

Figure 9.
Output of first-order nonlinear process, without predictive network

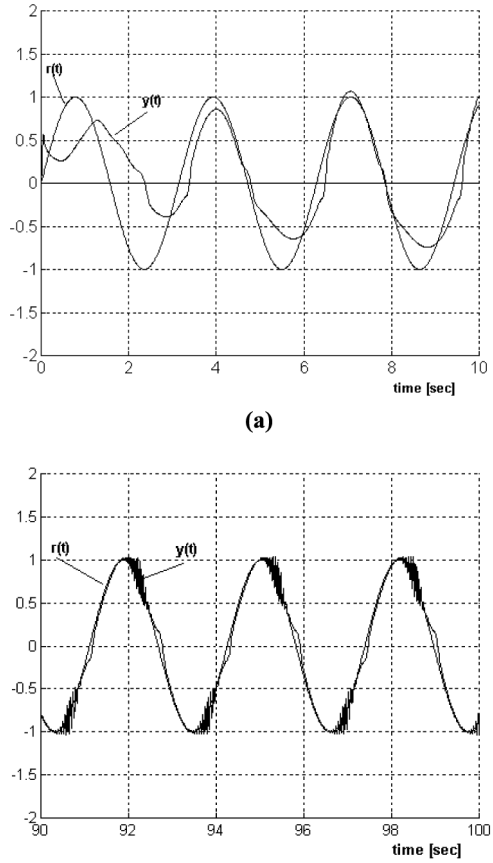


Figure 10.
Output of first-order nonlinear process, with use of predictive network

Notes: (a) Initial part, (b) Final part

Predictive network:	Structure:	$N^3_{9,20,10,1}$
	Inputs:	$[u(k), u(k-1), u(k-2), y(k), y(k-1), y(k-2), r(k), r(k-1), r(k-2)]$
	Sampling time:	0.01
	Learning rate:	0.01
	Prediction steps:	3
	Training:	5,000 online iterations

Table III.

Again, when the number of predictions are increased, while keeping fixed the learning rate of the controller network, the system becomes more stable.

5.4.5 Linear MIMO process. The parameters of networks used for controlling the linear MIMO process defined by equation (46) are given in Table IV.

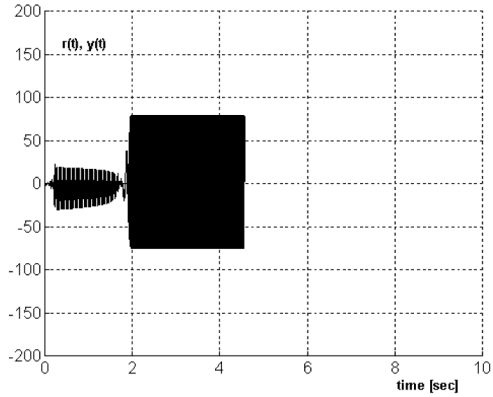
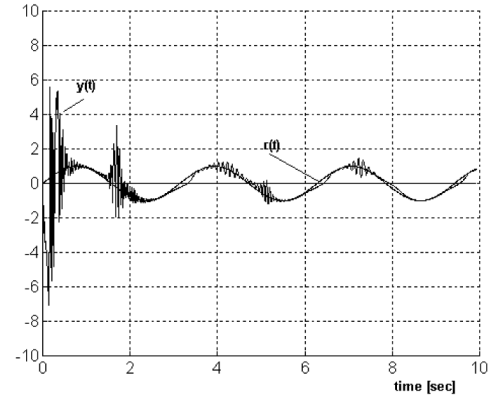
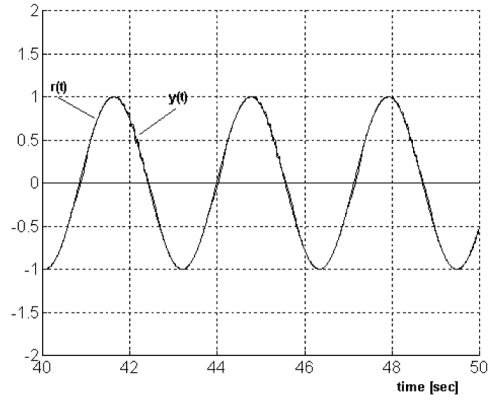


Figure 11.
Output of second-order nonlinear process, without predictive network. Initial stage only



(a)



(b)

Notes: (a) Initial part, (b) Final part

Figure 12.
Output of second-order nonlinear process, with use of predictive network

Controller network:	Structure:	$\boldsymbol{N}_{8,1}^{1}$
	Inputs:	$[r1(k), r1(k-1), r2(k), r2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.5
	Training:	5,000 online iterations
Identifier network:	Structure:	$\boldsymbol{N}_{8,1}^{1}$
	Inputs:	$[u1(k), u1(k-1), u2(k), u2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
Predictive network:	Structure:	$\boldsymbol{N}_{12,1}^{1}$
	Inputs:	$[r1(k), r1(k-1), r2(k), r2(k-1), u1(k), u1(k-1), u2(k), u2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
	Prediction steps:	3
	Training:	5,000 online iterations

Table IV.

The off-line training of the identifier network had 5,000 epochs and each epoch was composed by 600 pairs of data. Those 600 vectors were generated by exciting the plant with white noise, uniformly distributed between $[-2, +2]$, using different seeds for each input.

For the control scheme without predictive network, the results are shown in Figure 13, which shows the initial and final stages of the process output, during the training phase of the controller network. For the scheme with the predictive network, the results are shown in Figure14.

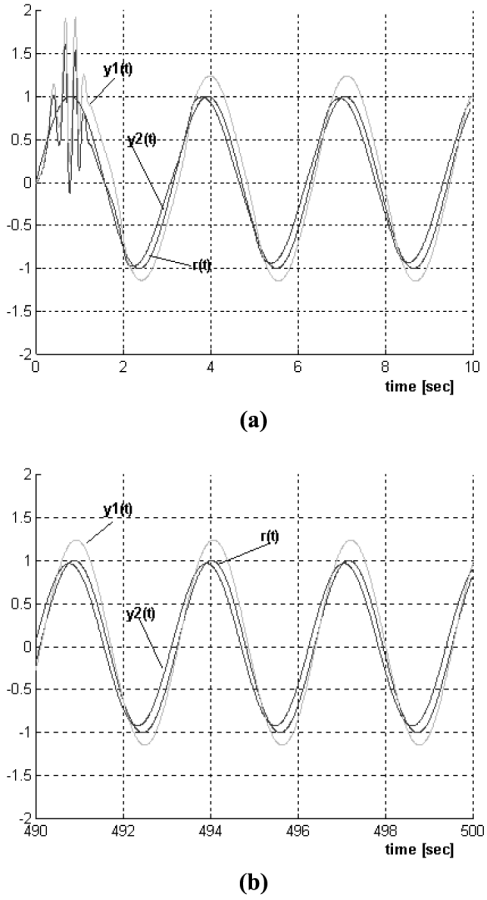
From Figures 13 and 14, it is observed that the transient stage response is less oscillatory when using the proposed scheme. For the steady state output they are both very similar. However, the control scheme without predictive network was trained five times more than the scheme with predictive network.

5.4.6 Non-linear MIMO process. The parameters of network used for controlling the MIMO non-linear plant, defined in equation (47) are given in Table V.

The off-line training of the identifier network had 5,000 epochs, and each epoch was composed by 600 pairs of data. Those 600 vectors were generated by exciting the plant with white noise, uniformly distributed between $[-2, +2]$, using different seeds for each input.

For the control scheme without predictive network, Figure 15 shows only the initial stage of the process output, during the training phase of the controller network, since the steady output was very oscillatory. For the scheme using the predictive network, the results are shown in Figure16, for the initial and final stages.

A significant improvement it is observed in the system response when the proposed scheme is used, with respect to the scheme without prediction. This advantage was also present when we applied the proposed scheme to control a phenomenological (non-linear) model of CODELCO-Andina grinding plant. For the control strategy a 2×2 multivariable model of the grinding plant was chosen, where the output variables are the sump level (L), the pulp density to hydrocyclones (D), and the input variables are the pump speed (V) and the sump water flow (F). A less oscillatory



Notes: (a) Initial part, (b) Final part

Figure 13.
Output of MIMO linear process, without predictive network

response variables L and D was obtained using predictive effects if compared with the scheme without prediction (Duarte *et al.*, 2001)

6. Conclusions

In this paper, a new predictive controller scheme has been presented. This general approach is based on predicting the future errors through a predictive neural network, taking advantage of the NN characteristics to approximate any kind of relationship. The advantage of this predictive scheme is that the knowledge of the future reference values is not needed, since the information used to train the predictive NN (used to estimate the future values of the control error) is based on present and past values of the control error. Changes in the reference signal have to be smooth enough to achieve good results.

The controller parameters are simply obtained from a global optimization process, by minimizing the prediction error (control error at present and future instants of time).

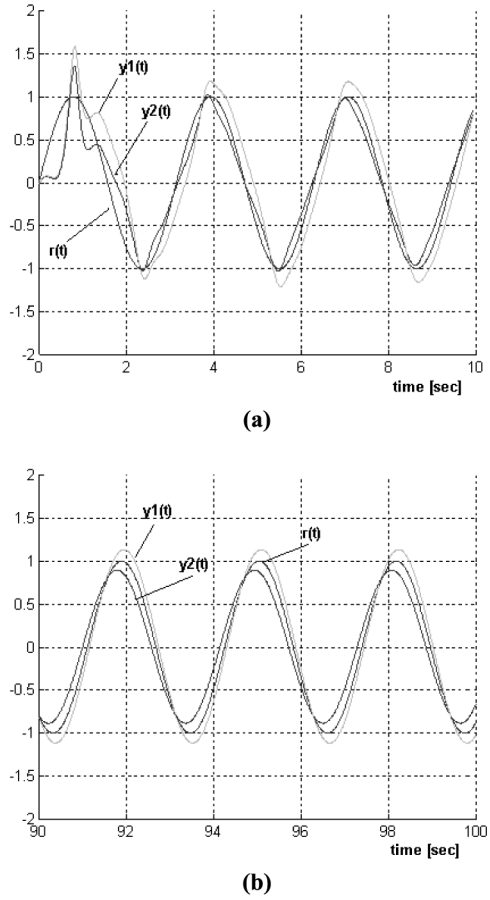


Figure 14.
Output of MIMO linear process, with use of predictive network

Notes: (a) Initial part, (b) Final part

This was rendered feasible since only present and past data were used for predictions, not being required to know beforehand the plant reference. This is very convenient if the process is part of a bigger system, where the reference may not be known a priori.

Conversely to this new scheme we have a simpler scheme with no predictive network, where the controller parameters are computed using optimization in the traditional sense: to minimize the current value of the control error.

In performing the network optimization we use an appropriate back propagation error through the time. We observe that the error local gradient, which is propagated from the identifier network to the controller network, depends on the local future gradients of the input layer of the identifier network. This problem is overcome by using present and past values of the scalar local gradients, to avoid non causality.

In linear SISO processes, both control schemes, with and without error prediction, behave similarly, not finding significant differences. When the learning rate of the

Controller network:	Structure:	$N^3_{8,20,10,1}$
	Inputs:	$[r1(k), r1(k-1), r2(k), r2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
	Training:	10,000 online iterations
Identifier network:	Structure:	$N^3_{8,20,10,1}$
	Inputs:	$[u1(k), u1(k-1), u2(k), u2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
Predictive network	Structure:	$N^3_{12,20,10,1}$
	Inputs:	$[r1(k), r1(k-1), r2(k), r2(k-1), u1(k), u1(k-1), u2(k), u2(k-1), y1(k), y1(k-1), y2(k), y2(k-1)]$
	Sampling time:	0.01
	Learning rate:	0.01
	Prediction steps:	3
	Training	10,000 online iterations

Table V.

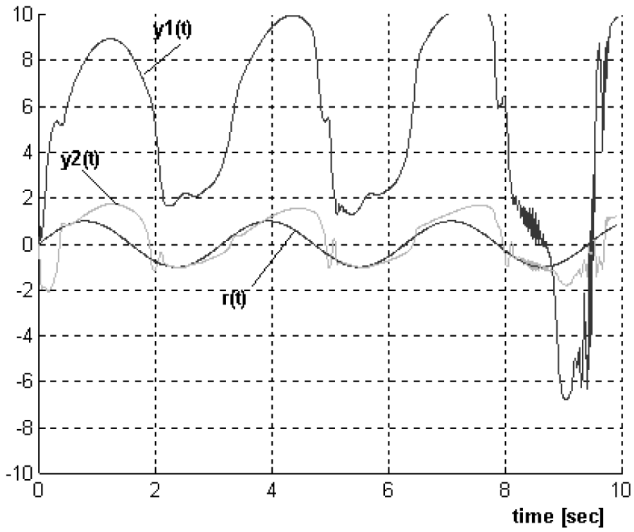


Figure 15.
Output of nonlinear MIMO process, without predictive network. Initial stage only

controller network is increased, the system converges more rapidly. On the other hand, increasing the number of predicting steps makes the transient slower. For linear (MIMO) process we have a more stable transient stage, i.e. with fewer oscillations, when prediction NN is used.

For non-linear SISO processes, we observed that increasing the learning rate of controller network makes the system less stable. When using a relatively high learning rate (i.e. 0.5), the predictive effect makes more stable the processes, in the transient and steady state stages. Also, the system is stabilized by augmenting the number of prediction steps. In the MIMO case when a predictive network scheme is used, we

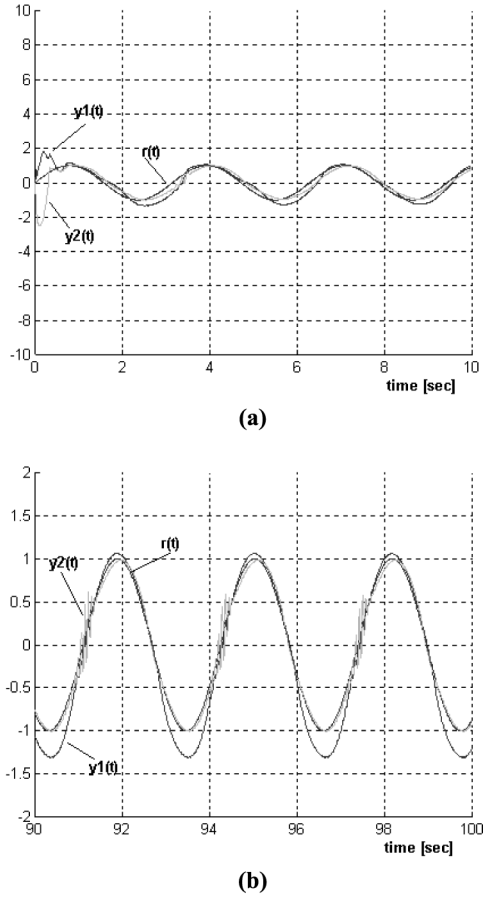


Figure 16.
Output of nonlinear MIMO process, with use of predictive network

Notes: (a) Initial part, (b) Final part

observe a significant improvement, for both transient and steady stage, with respect to the non predictive approach.

In summary, both schemes, with a without predictive effects, can suitably control linear systems, but only the approach with predictive effects can satisfactorily control non-linear systems.

Finally, it is important to point out that the architecture of the proposed control scheme, using a NN as a plant model, a NN for the controller and another NN for the identifier, facilitates their integration in one larger network.

References

Brengel, D.D. and Seider, W.D. (1989), "Multistep nonlinear predictive control", *Ind. Chem. Eng. Res.*, Vol. 28, pp. 1812-22.

Cutler, C. and Ramaker, B. (1980), "Dynamic matrix control: a computer control algorithm", *Proceedings of Joint Automatic Control Conference, CA, Paper WP5-B.*

Draeger, A., Engell, S. and Ranke, H. (1995), "Model predictive control using neural networks", *IEEE Control Systems Magazine*, October, pp. 61-6.

Duarte, M., Suárez, A. and Bassi, D. (2001), "Control of grinding plants using predictive multivariable neuronal control", *Powder Technology*, Vol. 115 No. 2, pp. 193-206.

Eaton, J.W., Rawling, J.B. and Ungar, L.H. (1995), "Stability of neural net based model predictive control", *Proceedings of the American Control Conference, Baltimore, USA*, pp. 2481-5.

García, C.E., Prett, D.M. and Morari, M. (1989), "Model predictive control: theory and practice – a survey", *Automatica*, Vol. 25, pp. 335-48.

Haykin, S. (1994), *Neural Networks. A Comprehensive Foundation*, Maxwell Macmillan International, New York, NY.

Hornik, K., Stincombe, M. and White, H. (1990), "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks", *Neural Networks*, Vol. 3, pp. 211-23.

Lundstrom, P., Lee, J.H., Morari, M. and Skogestad, S. (1995), "Limitations of dynamic matrix control", *Computer Chem. Eng.*, Vol. 19 No. 4, pp. 409-21.

MacMurray, J. and Himmelblau, D. (1992), "Identification of a packed distillation column for control via artificial neural networks", *Proceedings of the American Control Conference, San Francisco, CA*, pp. 1455-9.

Murray-Smith, R., Sbarbaro, D. and Neumerkel, D. (1992), "Neural networks for modeling and control of a nonlinear dynamic system", paper presented at IEEE Symposium on Intelligent Control, Glasgow, Scotland, pp. 122-7.

Narendra, K.S. and Parthasarathy, K. (1990), "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Net.*, Vol. 1 No. 1, pp. 4-27.

Narendra, K.S. and Parthasarathy, K. (1991), "Gradient methods for optimization of dynamical systems containing neural networks", *IEEE Trans. Neural Net.*, Vol. 2 No. 2, pp. 252-62.

Peterson, T., Hernandez, E., Arkun, Y. and Schork, F.J. (1989), "Nonlinear predictive control of a semi batch polymerisation reactor by an extended DMC", *Proceedings of American Control Conference*, pp. 1534-9.

Saint-Donald, J., Bath, N. and McAvoy, T.J. (1991), "Neural net based model predictive control", *International Journal of Control*, Vol. 54 No. 6, pp. 1453-68.

Suárez, A. (1998), "New architecture of predictive control for nonlinear systems using neural networks", PhD thesis, Department of Electrical Engineering, University of Chile, Santiago (in Spanish).

Tan, Y. and de Kayser, R. (1994a), "Adaptive neural control for nonlinear processes with large deadtime", *Proceedings of the 3 IFAC Symposium on Artificial Intelligence in Real-Time Control, Valencia, Spain*, pp. 219-24.

Tan, Y. and de Kayser, R. (1994b), "Neural network based predictive control for nonlinear processes with time-delay", *Proceedings of the IEEE Conference on System Man and Cybernetics, San Antonio, TX, October 2-5, USA*.

Tan, Y. and de Kayser, R. (1994c), "Neural networks based adaptive predictive control", in Clarke, D. (Ed.), *Advanced in Model Predictive Control*, Oxford University Press, London, pp. 77-88.

Thibaut, J. and Grandjean, B.P.A. (1992), "Neural networks in process control: a survey", in Najim, K. and Dufour, E. (Eds), *Advanced Control of Chemical Processes*, IFAC Symposium Series No. 8, pp. 251-60.

Wan, E.A. (1990), "Temporal backpropagation for FIR neural networks", *Proceedings of the IEEE Joint Conference on Neural Networks, San Diego, CA*,Vol. 1, pp. 575-80.

Ydstie, E. (1990), "Forecasting and control using adaptive connectionist networks", *Computers Chem. Eng.*, Vol. 4, pp. 583-99.

Appendix

In equation (26), for analyzing the partial derivative inside the sum we consider the equation (16). The index q, which defines the upper limit in the inner sum of equation (16), corresponds to the total number of time delays in each synaptic filter between jth neuron and all other neuron in hidden layer. The index $n2$, that defines the upper limit of the outer sum, corresponds to the total number of synapses arriving to jth neuron.

Recalling that the discrete convolution is commutative with respect to r, we can rewrite equation (16) in the following equivalent form:

$$v_j(n) = \sum_{i=0}^{n2} \sum_{r=0}^{q} w_{ji}(n-r) y_i(r) \tag{A1}$$

Differentiating equation (A1) with respect to y_i, we obtain:

$$\frac{\partial v_j(n)}{\partial y_i(n)} = \begin{cases} w_{ji}(n-r), & 0 \le n-r \le q \\ 0, & \text{otherwise} \end{cases} \tag{A2}$$

According to equation (A2), the partial derivative of equation (26), inside the sum, for each n that is outside the range $r \le n \le q + r$, is exactly zero. Therefore, for the case of ith hidden neuron, applying equation (A2) into equation (26) yields:

$$\sum_{j=0}^{n1} \sum_{n=r}^{q+r} \bar{\delta}_j(n) w_{ji}(n-r) = \sum_{j=0}^{n1} \sum_{n=0}^{q} \bar{\delta}_j(n+r) w_{ji}(n) \tag{A3}$$

Corresponding author
Manuel A. Duarte-Mermoud can be contacted at: mduartem@ing.uchile.cl