INFORMATICA, 2014, Vol. 25, No. 3, 401–414

DOI: http://dx.doi.org/10.15388/Informatica.2014.20

Adaptive Inverse Control Using an Online Learning Algorithm for Neural Networks

José Luis CALVO-ROLLE[1]*, Oscar FONTENLA-ROMERO[2], Beatriz PÉREZ-SÁNCHEZ[2], Bertha GUIJARRO-BERDIÑAS[2]

[1]*Department of Industrial Engineering, E.U. Politécnica, University of A Coruña Campus de Ferrol, Avda. 19 de Febrero s/n, 15405 Ferrol, Spain*

[2]*Laboratory for Research and Development in Artificial Intelligence (LIDIA) Department of Computer Science, Faculty of Informatics, University of A Coruña Campus de Elviña s/n, 15071 A Coruña, Spain*

e-mail: jlcalvo@udc.es, ofontenla@udc.es, bperezs@udc.es, cibertha@udc.es

Received: June 2012; accepted: April 2014

Abstract. We propose an adaptive inverse control scheme, which employs a neural network for the system identification phase and updates its weights in online mode. The theoretical basis of the method is given and its performance is illustrated by means of its application to different control problems showing that our proposal is able to overcome the problems generated by dynamic nature of the process or by physical changes of the system which originate important modifications in the process. A comparative experimental study is presented in order to show the more stable behavior of the proposed method in several working ranks.

Key words: model predictive control, adaptive inverse control, neural networks, neural predictor.

1. Introduction

Nowadays automatic control systems have become part of our everyday life. They allow us to guarantee that the process or plant presents the desired behavior along time. Among other applications they ensure that, for example (a) the temperature in our homes remains within adequate levels in both summer and winter, (b) the airplanes maintain desired speed and altitude, and (c) the automobile emissions satisfy the specifications. Automatic control systems can take different shapes but common to all them is the function to manipulate a system so that it reaches the desired behavior. When designing a controller for a particular system, it is important to acquire knowledge about how the system will react when it is manipulated in different ways. This information allows us to know how to control the system so that it shows a determinate behavior (Nørgaard *et al*., 2000).

The principles of predictive control were introduced in the seventies. Subsequently over more than two decades, these were refined to a high level thanks to an important research in the field. Among others classic methods, it worth mentioning the proportional-

*Corresponding author.

integral-derivative (PID) and the model predictive control (MPC) since, they are considered as two of the most popular control strategies. The first one, PID control algorithm, is widely applied in the industrial field. This is not only due to its simple structure but also to the fact that the algorithm shows a good performance in a wide variety of applications (Liua and Daleyb, 2001). Most of the PID tuning rules use frequency-response methods as those proposed by Ziegler and Nichols (1942), Voda and Landau (1995), Zhuang and Atherton (1993), Pessen (1994), Kaya and Scheib (1988), Chien *et al*. (1952), Åstrom and Hägglund (2000). The latter, MPC is an advanced control strategy based on the optimization of an objective function within a specified prediction horizon. This strategy has been recognized as the winning alternative for constrained multivariable control of industrial systems (Seborg *et al*., 2004; Maciejowski, 2002; Normey-Rico and Camacho, 2007). However, there exist many real industrial systems that present highly nonlinear characteristics which can be inherent to the system or due to its deterioration, signs of wearing, etc. In such situations, classic methods, based on a linear mathematical model of the controlled process, are not very efficient because they can not guarantee stable control outside the range of model validity (Muske and Rawlings, 1993). In order to improve PID and MPC performance for dynamic processes, several tuning strategies were presented by different authors (Åstrom and Hägglund, 1984; Kraus and Mayron, 1984) but these approaches do not achieve a good control performance. These reasons motivate the development of nonlinear model predictive control (NMPC) where a more accurate model of the process is used for prediction and optimization. However, many of current NMPC schemes are based on physical models of the controlled processes, which are difficult to obtain or they are not available.

A recent approach to model nonlinear dynamical systems is the use of artificial neural networks. Neural networks have turned into an useful tool to deal with the problems of controlling nonlinear dynamical systems thanks to their properties, such as the adaptive nature and the universal approximation capabilities. Unlike classic methods, artificial neural networks solve any nonlinear correspondence between the inputs and the outputs of a system. The key to the successful application of NMPC based on a neural network model is an accurate nonlinear model and an efficient optimization algorithm. Taking into account these favorable characteristics, the application of neural networks for model identification and adaptive control of dynamic and complex systems has been studied extensively (Nørgaard *et al*., 2000; Narendra and Parthasarathy, 1990; Sarangapani, 2006). Thus, it is possible to find diverse approaches of neural control applied to different interest fields (Hsu, 2008; Zhai and Yu, 2009; Ben-Nakhi and Mahmoud, 2002; Vasickaninova *et al*., 2011; Han and Qiao, 2011; Leeghim *et al*., 2009; Wang *et al*., 2006).

Besides, it is worthwhile to mention that in the particular case of highly dynamic processes, it would be appropriate to employ models that allow to make a precise control in real time. Therefore, with the aim of treating properly these situations, in this work we propose a generic topology which allows the control of any industrial process including highly nonlinear and dynamic systems. This is the main characteristic of the proposed model improving the existing techniques that employ among others, minimum squares.

Then, we propose a new nonlinear adaptive control model which employs, as part of the system identification stage, a learning algorithm for a two-layer feedforward neural network that updates the weights in online mode to overcome the problems generated by the dynamic nature of the process. This algorithm, presented in Pérez-Sánchez *et al*. (2010), includes a factor that weights the errors committed in each one of the training samples. Thanks to this term, the method obtains a good performance in environments that present an evolutionary behavior over time. Thus, in this paper we present a model for predictive control that furthermore allows to adjust the identification of process in an online mode favouring its control.

The paper is structured as follows. In Section 2 we describe our proposal of adaptive inverse control scheme with neural networks. In Section 3 its behavior is illustrated by its application to several simulated systems and their results are compared to those obtained by classic control systems. Finally, in Section 4 the results are discussed and some conclusions are given.

2. Proposed Method Based on Adaptive Inverse Control

Adaptive inverse control using neural networks for nonlinear dynamical system has received much attention in recent years. The aim of this technique is to inversely identify the dynamic of the process using its outputs as inputs of the model. Direct adaptive inverse control is a relatively new approach (Widrow and Walach, 2008), which combines signal processing methods with the control theory, and it is designed to control systems with complex characteristics. It is a control technique widely used for design and analysis in industrial process control systems as it can be implemented for a process in a straightforward way. As the process is generally unknown, it is necessary to adapt or to adjust the parameters of the neural network in order to create a true plant inverse.

In Fig. 1 it can be observed our proposal of adaptive inverse control scheme for an industrial process. In figure, $SP(t)$ indicates the set point signal at the instant t, $y_d(t)$ is the desired output of the system, $u(t)$ refers to the control signal and $y(t)$ denotes the process output. In the proposed scheme, an error signal, the difference between the plant output $y(t)$ and the desired output of the system $y_d(t)$, is used by an adaptive algorithm to adjust the parameters with the aim of minimizing the mean square of this error. Therefore, the key of the inverse control is how to obtain the inverse model of the plant. The control system is composed of two parts which are detailed below:

1. The *driver* block which aim is to force the output response to a certain way, despite of the process natural behavior. Thus, the driver block is a mathematical model that calculates the desired output of the process for a particular behavior using as reference the set point signal. The block has several inputs: the set point, the process output and their past values. As output, the driver block obtains the desired output of the process. The concept is very similar to the Reference Trajectory of Model Predictive Control (Camacho and Bordons, 2004), Predictive Functional Control (Richalet *et al*., 1987) or Extended Prediction Self Adaptive Control (De Keyser

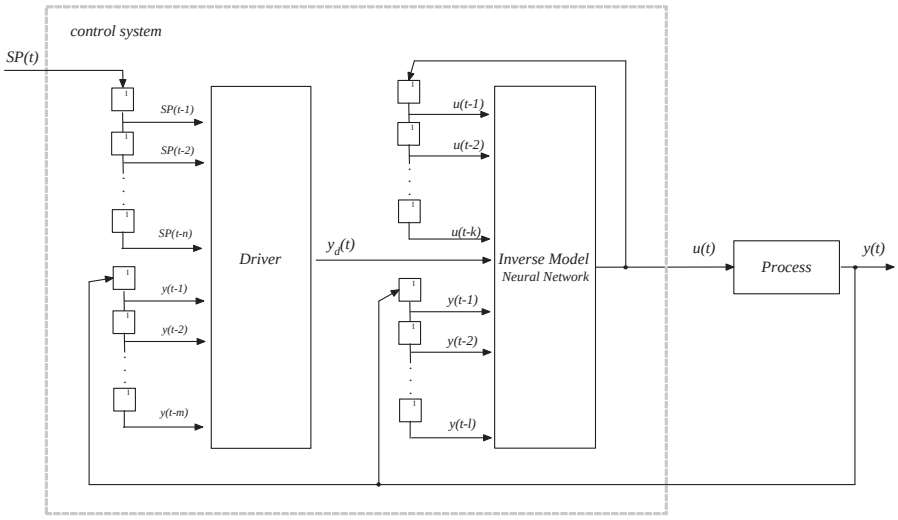


Fig. 1. Adaptive inverse control scheme.

and Van Cuawenberghe, 1985). In most cases, typically, the mathematical model implements a first order system. For the proposed method, the desired output $y_d(t)$ is described by the following equation,

$$y_d(t) = \alpha_1 SP(t-1) + \alpha_2 SP(t-2) + \cdots + \alpha_n SP(t-n) + \beta_1 y(t-1) + \beta_2 y(t-2) + \cdots + \beta_m y(t-m) \quad (1)$$

where α_n and β_m are the parameters establishing the dynamic of this driver block for the desired response of the controlled system. Usually the driver block is used to smooth the set point signal in order to get an output without overshoot, achieving good results for response and peak times, etc.

2. The *neural predictor* is employed to identify the system, in an inverse way, using a real time approach. As it can be observed in Fig. 1, the inputs of the neural predictor are composed of the desired output $y_d(t)$ and previous values of the process output $y(t)$ and the control signal $u(t)$. The online learning algorithm employed to train the feedforward neural network is explained in detail in the following subsection.

It is worthwhile to mention that in the case of real systems the previous values of the plant output signal should be filtered before their substitution. The inclusion of this filtered phase allows to the system reduces possible noises, parasitic nonlinearities, etc. of the output signal. As we propose a generic model this filtered phase is not reflected in our scheme (as it can be observed in Fig. 1).

2.1. *Online Learning Algorithm for the Neural Network*

The learning algorithm used to train the neural network was presented in a previous work (Pérez-Sánchez *et al.*, 2010). It is an online algorithm for two-layer feedforward neural

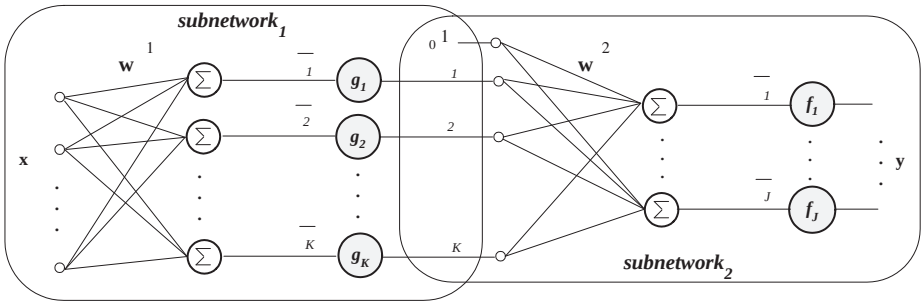


Fig. 2. Two-layer feedforward neural network.

networks which has the ability of dealing with non-stationary environments. Consider the two-layer feedforward neural network in Fig. 2 where the inputs are represented as the column vectors $\mathbf{x}(t)$ and the outputs are denoted as $\mathbf{y}(t)$, where t indicates the learning epoch, I, J are the number of inputs and outputs, respectively, and K is the number of hidden neurons. Functions $g_1, g_2, \ldots, g_K$ and $f_1, f_2, \ldots, f_J$ are the nonlinear activation functions of the hidden and output layer of the network.

If we consider this network as the composition of two one-layer subnetworks (see Fig. 2) we can define the objective functions for both subnetworks independently. Suppose $z_k(t)$, the desired output for hidden neuron k at the current learning epoch t, is available (see Fig. 2). In this case, employing $\bar{z}_k(t) = g_k^{-1}(z_k(t))$ we can define the objective function for the k output of the subnetwork 1 as the sum of squared errors *before* the nonlinear activation function g_k as,

$$Q_k^{(1)}(t) = h_k(t)\big(g_k'\big(\bar{z}_k(t)\big)\big(\mathbf{w}_k^{(1)^T}(t)\mathbf{x}(t) - \bar{z}_k(t)\big)\big)^2, \quad k = 1, \ldots, K, \tag{2}$$

where $\mathbf{w}_k^{(1)}(t)$ is the input vector of weights for hidden neuron k at the instant t and $g_k'(\bar{z}_k(t))$ is a scaling term which weighs the errors (Fontenla-Romero *et al.*, 2010). Moreover, $h_k(t)$ is a forgetting factor that determines the importance of the error at the t-th time instant. This factor is employed in order to establish the form and the speed of the adaptation to the recent samples in a dynamic context (Martínez-Rego *et al.*, 2011). Analogously, the cost function for each output j of the subnetwork 2 is defined as,

$$Q_j^{(2)}(t) = h_j(t)\big(f_j'\big(\bar{d}_j(t)\big)\big(\mathbf{w}_j^{(2)^T}(t)\mathbf{z}(s) - \bar{d}_j(t)\big)\big)^2, \quad j = 1, \ldots, J, \tag{3}$$

where $d_j(t)$ is the desired output for neuron j, $\bar{d}_j(t) = f_j^{-1}(d_j(t))$, $f_j'(\bar{d}_j(t))$ is the scaling term of the errors and $h_j(t)$ is a forgetting factor. Regarding the forgetting functions $h_k(t)$ and $h_j(t)$, there exist several options that can be used, for example, an exponential or lineal function among others. In a stationary environment a constant function should be used in order to give the same weight to all the data points analyzed during the learning process. Whereas in a non-stationary context the function should be monotonically crescent to take into account the increment in the importance of recent information in contrast with the previous one.

As the objective functions presented in Eqs. (2) and (3) are cuadratic in weights **w**, which are free parameters of the system, both functions are convex. Thus for these objective functions, the global optimum can be easily obtained deriving them with respect to the parameters of the network and setting the derivatives to zero (Fontenla-Romero *et al.*, 2010). So, using matrix notation, we obtain the following systems of linear equations,

$$\mathbf{A}_k^{(1)}(t)\mathbf{w}_k^{(1)}(t) = \mathbf{b}_k^{(1)}(t), \quad k = 1, \ldots, K, \tag{4}$$

$$\mathbf{A}_j^{(2)}(t)\mathbf{w}_j^{(2)}(t) = \mathbf{b}_j^{(2)}(t), \quad j = 1, \ldots, J, \tag{5}$$

where

$$\mathbf{A}_k^{(1)}(t) = \mathbf{A}_k^{(1)}(t-1) + h_k(t)\mathbf{x}(t)\mathbf{x}^T(t)g_k'^2\big(\bar{z}_k(t)\big), \tag{6}$$

$$\mathbf{b}_k^{(1)}(t) = \mathbf{b}_k^{(1)}(t-1) + h_k(t)g_k^{-1}(z_k(t))\mathbf{x}(t)g_k'^2\big(\bar{z}_k(t)\big) \tag{7}$$

and

$$\mathbf{A}_j^{(2)}(t) = \mathbf{A}_j^{(2)}(t-1) + h_j(t)\mathbf{z}(t)\mathbf{z}^T(t)f_j'^2\big(\bar{d}_j(t)\big), \tag{8}$$

$$\mathbf{b}_j^{(2)}(t) = \mathbf{b}_j^{(2)}(t-1) + h_j(t)f_j^{-1}\big(d_j(t)\big)\mathbf{z}(t)f_j'^2\big(\bar{d}_j(t)\big), \tag{9}$$

being $\mathbf{A}^{(1)}(t-1)$, $\mathbf{A}^{(2)}(t-1)$, $\mathbf{b}^{(1)}(t-1)$, $\mathbf{b}^{(2)}(t-1)$ the matrices and vectors that store the coefficients of the systems of linear equations obtained in previous iterations to calculate the values of the weights of both layers. Therefore, this permits handling the earlier knowledge acquired by the network and using it to incrementally approach the optimum value of the weights.

Finally, from Eqs. (4) and (5) the optimal weights can be obtained as,

$$\mathbf{w}_k^{(1)}(t) = \mathbf{A}_k^{(1)^{-1}}(t)\mathbf{b}_k^{(1)}(t), \quad \forall k, \tag{10}$$

$$\mathbf{w}_j^{(2)}(t) = \mathbf{A}_j^{(2)^{-1}}(t)\mathbf{b}_j^{(2)}(t), \quad \forall j. \tag{11}$$

Algorithm 1 details the incremental online learning method employing the concepts earlier described. Finally, it is worth mentioning that the complexity of the algorithm is determined by the complexity to solve several systems of linear equations for each layer of the network. Several computationally efficient methods can be used to solve this kind of systems (except for ill-conditioned matrices) with a complexity from $O(n^2)$ to $O(n^3)$, being n the number of weights of the network (Carayannis et al., 1982; Bojanczyk, 1984). In the case of ill-conditioned matrices, the problem can be solved applying the pseudoinverse Moore–Penrose (Penrose, 1955).

3. Experimental Results

Several experiments were carried out to check the efficacy of the proposed method. In order to accomplish this analysis we have compared the behavior of our proposal with

Algorithm 1 Online algorithm with forgetting ability for two-layer feedforward neural networks.

Inputs: $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots x_I(t))$, being t the current instant
$\mathbf{d}(t) = (d_1(t), d_2(t), \ldots d_J(t))$
bias, $x_0(t) = 1, z_0(t) = 1$

1. Initialization Phase:
2. $\mathbf{A}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times(I+1)}$, $\mathbf{b}_k^{(1)}(0) = \mathbf{0}_{(I+1)\times 1}$, $\forall k = 1, \ldots, K$,
3. $\mathbf{A}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times(K+1)}$, $\mathbf{b}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times 1}$, $\forall j = 1, \ldots, J$,
4. The initial weights, $\mathbf{w}_k^{(1)}(0)$, are calculated by means of some initialization method.
5. For the current instant t, and $k = 1, \ldots, K$
6. $z_k(t) = g(\mathbf{w}_k^{(1)}(0), \mathbf{x}(t))$,
7. For each output k of the subnetwork 1 ($k = 1, \ldots, K$),
8. $\mathbf{A}_k^{(1)}(t) = \mathbf{A}_k^{(1)}(t-1) + h_k(t)\mathbf{x}(t)\mathbf{x}^T(t)g_k'^2(\bar{z}_k(t))$ (Eq. (6)),
9. $\mathbf{b}_k^{(1)}(t) = \mathbf{b}_k^{(1)}(t-1) + h_k(t)g_k^{-1}(z_k(t))\mathbf{x}(t)g_k'^2(\bar{z}_k(t))$ (Eq. (7)),
10. Calculate $\mathbf{w}_k^{(1)}(t)$ solving the system of linear equations (Eq. (4)),
11. end of For.
12. For each output j of the subnetwork 2 ($j = 1, \ldots, J$),
13. $\mathbf{A}_j^{(2)}(t) = \mathbf{A}_j^{(2)}(t-1) + h_j(t)\mathbf{z}(t)\mathbf{z}^T(t)f_j'^2(\bar{d}_j(t))$ (Eq. (8)),
14. $\mathbf{b}_j^{(2)}(t) = \mathbf{b}_j^{(2)}(t-1) + h_j(t)f_j^{-1}(d_j(t))\mathbf{z}(t)f_j'^2(\bar{d}_j(t))$ (Eq. (9)),
15. Calculate $\mathbf{w}_j^{(2)}(t)$ solving the system of linear equations (Eq. (5)),
16. end of For.
17. end of For.

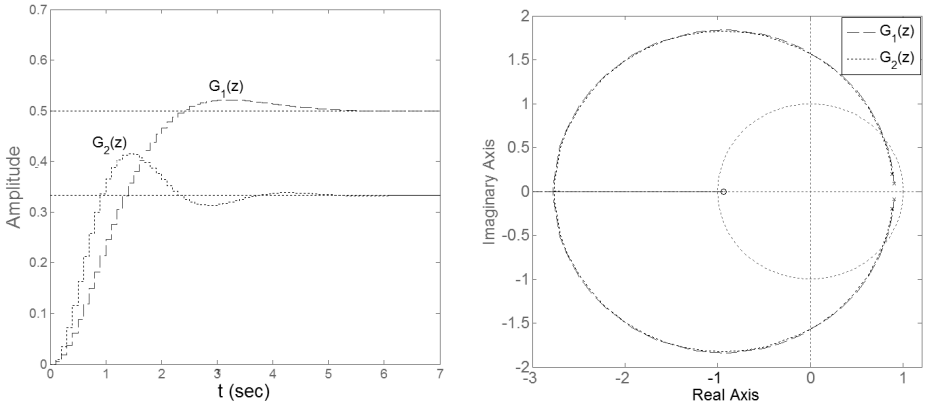
the standard methods PID, MPC and also for an adaptive scheme formed by a Recursive Least Squares (RLS) method and a self-tuned PID (Banyasz and Keviczky, 1982; Keviczky and Banyasz, 1992). This last approach uses the online RLS learning method to identify the process in real-time and the self-tuned PID to regulate the process. Thus, in this section we present two representative examples which uncover the good results achieved by the presented approach. In both cases only two parameters in the driver block, see Eq. (1), were used and their values were empirically established as $\alpha_1 = 0.2592$ and $\beta_1 = 0.7408$. Also the forgetting function in Eqs. (2) and (3) was the exponential function with a factor of 0.9. Regarding the standard controller PID, the great problem is the adjustment of its parameters. In order to solve this problem, we accomplished several proofs with some of the most known and usual methods, specifically, of Ziegler and Nichols (1942), Kaya and Scheib (1988) and Chien *et al*. (1952). The best methods for the adjustment of the parameters and the values obtained are displayed in Table 1. For the MPC controller the parameters were empirically determined and the final values are included in Table 2. Finally, for the adaptive RLS-PID method a forgetting factor of 0.99 was used for the RLS and a $K_m = 0.001$ for the self-tuned PID. This last parameter is used to attenuate the overshooting effect.

Table 1
PID controller parameters for experiments 1 and 2: proportional gain (K_p), integral time (t_i) and derivative time (t_d).

Experiment	Method of adjustment	K_p	t_i	t_d
1	Kaya–Scheib set point regulation minimize IAE	4.6741	1.5448	0.1016
2	Kaya–Scheib set point regulation minimize ISE	4.2968	0.3999	0.0489

Table 2
MPC controller parameters for experiments 1 and 2.

Parameter	Values
Control time step	0.1 s
Prediction horizon	1.0 s
Control horizon	0.2 s
Overall weight tuning	0.7
Overall estimator gain	0.5



(a) (b)

Fig. 3. Step response and root locus for the systems in experiment 1.

In the first experiment, the system to be controlled, $G_1(z)$, presents a low percentage of overshoot and a certain speed when the input is a unit step. However, at a given instant of time, the model of the plant was changed abruptly to $G_2(z)$. From that moment, the response of the system is faster and then it has more percentage of overshoot and its gain varies. Both systems are generated in accord with Eq. (12) with sampling time of 0.1 s. In Fig. 3(a) it can be seen the response for both systems to unit step. Moreover, Fig. 3(b) shows the root locus of both systems superposed. In this last figure, it is possible to check that the systems are very similar, in fact they have the same shape but their starting points are different

$$G_1(z) = \frac{0.004675z + 0.004373}{z^2 - 1.801z + 0.8187}, \qquad G_2(z) = \frac{0.009319z + 0.008717}{z^2 - 1.765z + 0.8187}. \tag{12}$$

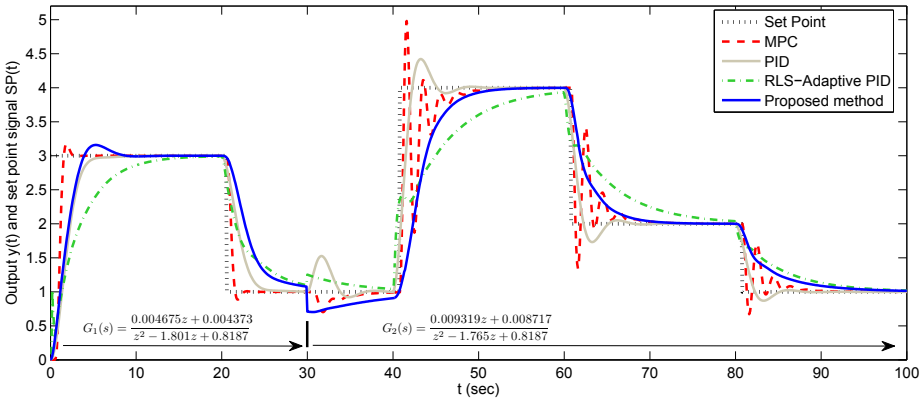
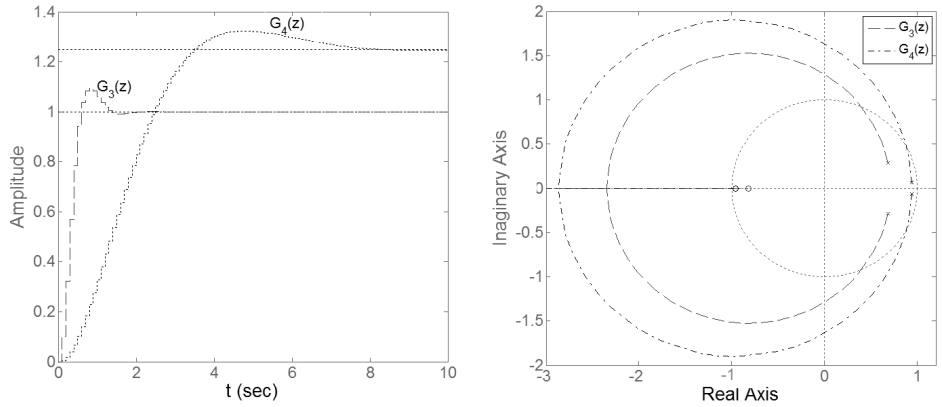


Fig. 4. Set point signal and system output for the MPC, PID, RLS-PID and proposed method in experiment 1.



(a) (b)

Fig. 5. Step response and root locus for the systems in experiment 2

Figure 4 shows the results for this experiment. The transition between the systems is presented at $t = 30$ s and it can be observed from this moment that the behavior of the MPC and PID is more unstable than the proposed method. In the instant $t = 30$ s all the methods present an abrupt change in their behavior but our proposal is able to adapt its parameters, in an online fashion, achieving a good identification of the new system.

In the second experiment, we generated the system $G_3(z)$ that shows a low percentage of overshoot when the input is an unit step. At a given instant of time the plant changes strongly to $G_4(z)$. Due to this fact, the system becomes slower and it presents a more percentage of overshoot and a different gain. Both systems are generated according to Eq. (13) with sampling time of 0.1 seconds. Figure 5(a) shows the response for both systems to unit step. The root locus of both systems superposed is shown in Fig. 5(b). In these graphics it

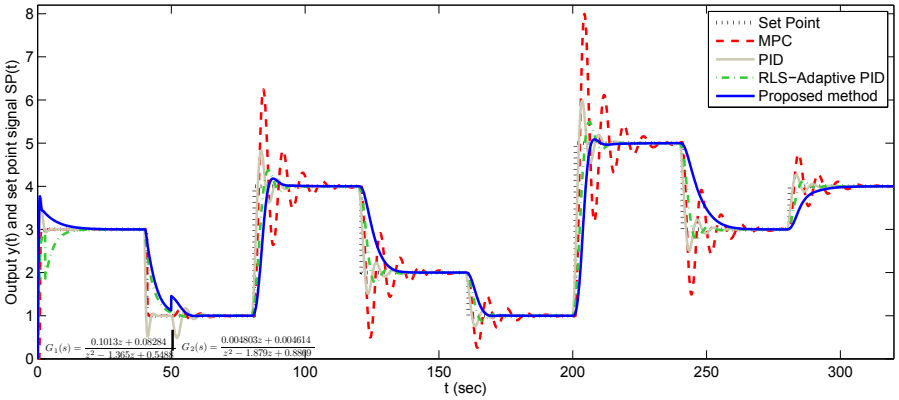


Fig. 6. Set point signal and system output for the MPC, PID, RLS-PID and proposed method in experiment 2.

can be seen that the change is more significative than in the first test.

$$G_3(s) = \frac{0.1013z + 0.08284}{z^2 - 1.365z + 0.5488}, \qquad G_4(s) = \frac{0.004803z + 0.004614}{z^2 - 1.879z + 0.8869}. \tag{13}$$

Figure 6 contains the results for this second test. In this case the change in the system is forced at $t = 50$ s. Again, at that point, the classical control models suffer a significant degradation but the proposed method recovers its good performance for the new system in a few number of samples. The other approaches present an oscillating behavior for the new system because they are not able to adapt to the new situation. Furthermore, the presented method presents a lesser overshooting effect than the adaptive RLS-PID, despite of the low value of K_m parameter, which ensures a more stable mode of operation.

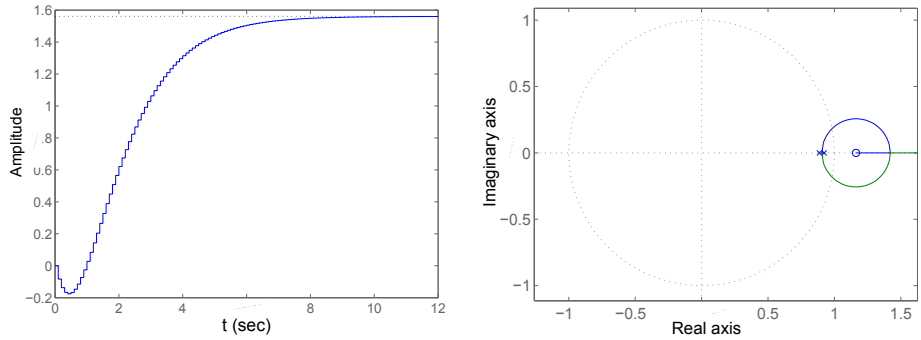
3.1. *Robustness to Disturbances*

Finally, two experiments were accomplish to analyze the performance of the proposed method for nonlinear or nonminimum-phase systems, including also some random disturbances added at the output of the system. The random disturbances were generated from a Normal distribution having a mean of 0 and a standard deviation of 0.02.

For the first experiment the plant introduced into the literature by Narendra and Parthasarathy (1990), and subsequently studied by other authors (Plett, 2003), was used. The difference equations defining its dynamics are:

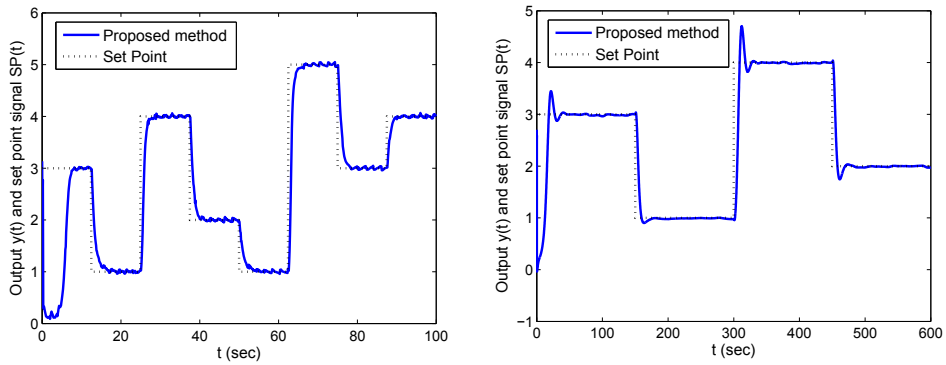
$$s(k) = \frac{s(k-1)}{1 + s^2(k-1)} + u^3(k-1),$$
$$y(k) = s(k) + noise(k),$$

where $noise(k)$ is the random disturbance described previously.



(a) (b)

Fig. 7. Step response and root locus for the nonminimum-phase system.



(a) (b)

Fig. 8. Set point signal and system output for the proposed method.

The second experiment was generated with the transfer function in Eq. (14) with sampling time of 0.1 s.

$$G(z) = \frac{-0.08347z + 0.09705}{z^2 - 1.81z + 0.8187}. \tag{14}$$

Figure 7 shows the response of this system to unit step and the corresponding root locus. As can be observed a typical nonminimum-phase response is obtained where the step response departs in a different direction from the one of the steady state, which produces an initial undershoot. A classical result states that the step response of an asymptotically stable, strictly proper transfer function exhibits initial undershoot if and only if the system has an odd number of positive zeros.

Figure 8 shows the results for the two experiments. The graphs show that in both cases the method is able to control the plant even with the presence of disturbances in the output of the plant. In the case of the nonminimum-phase system (Fig. 8(b)) the on-line learning algorithm can overcome the problem produced the initial undershoot and leads the plant to the reference point.

4. Conclusions

In this research we have proposed a scheme for adaptive inverse control that employs a supervised neural network for the system identification. In order to obtain a real-time control, a new online algorithm was employed for the learning process of the neural network. The main characteristics of the method are:

- The presented topology is able to control processes with hard nonlinearities and shifts in its operational behavior that cause different dynamic outcomes. Therefore, it is suitable for the control of dynamic processes that change very much their behavior due to several reasons such as variations in their working regime, deteriorations, maintenance operations, etc. Besides, as it was checked in the experimental study the model is able to adapt the identification of the process in real time with a stable behavior.
- It shows a better performance than classic control methods, such as PID and MPC, and other well-established adaptive systems as the Recurrent Least Squares with Self-tuning PID for the control of adaptive processes.
- The learning method, employed to identify the process, presents an complexity from $O((K+J)N^2)$ to $O((K+J)N^3)$, being K, J, N the number of hidden units, output neurons and parameters of each subnetwork, respectively. This is due to the system has to solve $K+J$ systems of linear equations in each iteration of the learning algorithm. Although previous online learning algorithms were used in the literature, this new method has a fast convergence and also can be parallelized (each system of linear equations can be solved independently) allowing a significant improvement in learning time for large networks.

Acknowledgements. The authors would like to acknowledge support for this work from the Secretaría de Estado de Investigación of the Spanish Goverment (Grant code TIN2012-37954), partially supported by the European Union FEDER funds.

References

Åstrom, K.J., Hägglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5), 645–651.

Åstrom, K.J., Hägglund, T. (2000). Benchmark systems for PID control. In: *Preprints FAC Workshop on Digital Control: Past, Present and Future of PID Control*, pp. 181–182.

Banyasz, C.S., Keviczky, L. (1982). Direct method for self-tuning PID regulators. In: *6th IFAC Symposium on Identification and System Parameter Estimation*, pp. 1249–1254.

Ben-Nakhi, A.E., Mahmoud, M.A. (2002). Energy conservation in buildings through efficient a/c control using neural networks. *Applied Energy*, 73(1), 5–23.

Bojanczyk, A. (1984). Complexity of solving linear systems in different models of computation. *SIAM Journal on Numerical Analysis*, 21(3), 591–603.

Camacho, E., Bordons, C. (2004). *Model Predictive Control*. Springer, London.

Carayannis, G., Kalouptsidis, N., Manolakis, D. (1982). Fast recursive algorithms for a class of linear equations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 30(2), 227–239.

Chien, K.-L., Hrones, J., Reswick, J. (1952). On the automatic control of generalised passive systems. In: *Transactions of the ASME*, pp. 175–185.

De Keyser, R., Van Cuawenberghe, A. (1985). Extended prediction self-adaptive control. In: *IFAC Symposium on Identification and System Parameter Estimation*, pp. 1317–1322.

Fontenla-Romero, O., Guijarro-Berdiñas, B., Pérez-Sánchez, B., Alonso-Betanzos, A. (2010). A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition*, 43(5), 1984–1992.

Han, H.-G., Qiao, J.-F. (2011). Adaptive dissolved oxygen control based on dynamic structure neural network. *Applied Soft Computing*, 11(4), 3812–3820.

Hsu, C.-F. (2008). Adaptive growing-and-pruning neural network control for a linear piezoelectric ceramic motor. *Engineering Applications of Artificial Intelligence*, 21(8), 1153–1163.

Kaya, A., Scheib, T. (1988). Tuning of PID controllers of different structures. *Control Engineering*, 7, 62–65.

Keviczky, L., Banyasz, C.S. (1992). An adaptive PID regulator based on time delay estimation. In: *Proceedings of the 31st IEEE Conference on Decision and Control*, vol. 4, pp. 3243–3248.

Kraus, T., Mayron, T. (1984). Self-tuning PID controllers based on a pattern recognition approach. *Control Engineering Practice*, 30(6), 106–111.

Leeghim, H., Choi, Y., Bang, H. (2009). Adaptive attitude control of spacecraft using neural networks. *Acta Astronautica*, 64(7–8), 778–786.

Liua, G., Daleyb, S. (2001). Optimal-tuning PID control for industrial systems. *Control Engineering Practice*, 9(11), 1185–1194.

Maciejowski, J. (2002). *Predictive Control with Constraints*. Pearson Education Limited, London.

Martínez-Rego, D., Pérez-Sánchez, B., Fontenla-Romero, O., Alonso-Betanzos, A. (2011). A robust incremental learning method for non-stationary environments. *NeuroComputing*, 74(11), 1800–1808.

Muske, K.R., Rawlings, J.B. (1993). Model predictive control with linear models. *AIChE Journal*, 39(2), 262–287.

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

Nørgaard, M., Ravn, O., Poulsen, N.K., Hansen, L.K. (2000). *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook*. Springer, New York.

Normey-Rico, J., Camacho, E. (2007). *Control of Dead-Time Process*. Springer, Berlín.

Penrose, R. (1955). A generalized inverse for matrices. In: *Proceedings of the Cambridge Philosophical Society*, Vol. 51, pp. 406–413.

Pérez-Sánchez, B., Fontenla-Romero, O., Guijarro-Berdiñas, B. (2010). An incremental learning method for neural networks in adaptive environments. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, pp. 815–822.

Pessen, D. (1994). A new look at PID-controller tuning. *Transactions of the American Society of Mechanical Engineering, Journal of Dynamic Systems Measures and Control*, 116, 553–557.

Plett, G.L. (2003). Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. *IEEE Transactions on Neural Networks*, 14(2), 360–376.

Richalet, J., Abu el Ata-Doss, S., Arber, C., Kuntze, H., Jacubash, A., Schill, W. (1987). Predictive functional control: application to fast and accurate robots. In: *10th World Congress of the International Federation of Automatic Control (IFAC)*.

Sarangapani, J. (2006). *Neural Network Control of Nonlinear Discrete-Time Systems*. CRC Press, Boca Raton.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A. (2004). *Process Dynamics and Control*. 2nd ed., Wiley, New York.

Vasickaninova, A., Bakosova, M., Meszaros, A., Klemes, J. (2011). Neural network predictive control of a heat exchanger. *Applied Thermal Engineering*, 31(13), 2094–2100.

Voda, A., Landau, I.D. (1995). A method for the auto-calibration of PID controllers. *Automatica*, 31(1), 41–53.

Wang, S.W., Yu, D.L., Gomm, J.B., Page, G.F., Douglas, S.S. (2006). Adaptive neural network model based predictive control for air–fuel ratio of SI engines. *Engineering Applications of Artificial Intelligence*, 19(2), 189–200.

Widrow, B., Walach, E. (2008). *Adaptive Inverse Control. A Signal Processing Approach*. Wiley, New York.

Zhai, Y.-J., Yu, D.-L. (2009). Neural network model-based automotive engine air/fuel ratio control and robustness evaluation. *Engineering Applications of Artificial Intelligence*, 22(2), 171–180.

Zhuang, M., Atherton, D.P. (1993). Automatic tuning of optimum PID controllers. *Control Theory and Applications, IEEE Proceedings D*, 140(3), 216–224.

Ziegler, J.G., Nichols, N.B. (1942). Optimum settings for automatic controllers. *IEEE Transactions of ASME*, 64, 759–768.

J.L. Calvo-Rolle was born in A Coruña, Spain, in 1974. He received the MS and PhD degrees in Industrial Engineering from the University of León, León, Spain, in 2004, and 2007, respectively. He is Associate Professor of Automatic Control and the head of Industrial Engineering Department, Faculty of Engineering, University of A Coruña, Spain. His main research interests have been centered in applying expert system technology to the diagnosis and control systems and in intelligent training systems for control engineering, optimization and education.

O. Fontenla-Romero was born in Ferrol, Spain, in 1974. He received his BS, MS and PhD degrees in computer science from the University of A Coruña, in 1997, 1998, and 2002, respectively. He works as an Associate Professor at the Department of Computer Science of University of A Coruña since 2004. His current research interests include new linear learning methods and noise immunity for neural networks and functional networks. His main current areas are neural networks, functional networks and non-linear optimization.

B. Pérez-Sánchez received her MS and PhD degree in computer science from the University of A Coruña, in 2005 and 2010, respectively. She works as an Assistant Professor at the Department of Computer Science of University of A Coruña since 2007. Her current research interests include theoretical works on neural networks and new linear learning methods and learning optimization.

B. Guijarro-Berdiñas received her BS, MS and PhD degrees in computer science from the University of A Coruña, in 1990, 1992, and 1998, respectively. At present, she works as a Tenured Professor at the Department of Computer Science of University of A Coruña. She has authored more than 50 scientific publications, including journals, books and book chapters in the field of Artificial Intelligence.

Adaptyvusis inversinis valdymas operatyviuoju mokymosi algoritmu, skirtu neuroniniams tinklams

José Luis CALVO-ROLLE, Oscar FONTENLA-ROMERO, Beatriz PÉREZ-SÁNCHEZ, Bertha GUIJARRO-BERDIÑAS

Siūloma adaptyvioji inversinio valdymo schema, sistemos identifikavimo fazėje naudojanti neuroninį tinklą, atnaujindama jos svorius operatyviojoje būsenoje. Pateiktas taikomo metodo teorinis pagrindas. Šio metodo efektyvumas iliustruojamas, sprendžiant įvairias valdymo problemas. Parodoma, kad pasiūlytas metodas yra pajėgus išspręsti uždavinius, sąlygojamus proceso dinamika ar sistemos fiziniais pokyčiais, sukeliančiais svarbias proceso modifikacijas.