INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. **13**, 219–239 (1999)

NEURAL ADAPTIVE CONTROL OF NON-LINEAR PLANTS VIA A MULTIPLE INVERSE MODEL APPROACH

PEDRO J. ZUFIRIA*, JESÚS FRAILE-ARDANUY†, RICARDO RIAZA AND JUAN I. ALONSO‡

Grupo de Redes Neuronales, Departamento de Matemática Aplicada a las Tecnologías de la Información, E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain

SUMMARY

In this paper, neural architectures for controlling non-linear plants with parameter variation are proposed. In the first part of the document, the concept of specialized learning over an operation region is considered in order to identify the inverse dynamics of a given plant. Some aspects concerning discretization and invertibility of continuous-time plants are also addressed. In the second part of this work, a control architecture which combines the former approach of inverse identification through specialised learning with a multiple model scheme is presented. Finally, simulation results are discussed, evaluating the performance of the proposed schemes; specifically, the presented controllers are applied to the simulation of the control of a robot arm. Copyright © 1999 John Wiley & Sons, Ltd.

Key Words: inverse identification; multiple model; neural adaptive control; non-linear plants

1. INTRODUCTION

The control of non-linear and uncertain systems is a very challenging field in the context of modern control theory. There are different approaches in the literature for designing non-linear controllers, which make use of specific properties of the plant. Lyapunov theory has been extensively employed for characterizing local linearization techniques and robust control.[1] One of the key concepts when addressing linearization control for non-linear plants is their controllability. If the plant S has a controllable linearization in the neighbourhood of an equilibrium point, it can be proved that there exists a linear state feedback law making the plant S stable in the neighbourhood of the equilibrium point.[2] This property has been used in the design of linear

* Correspondence to: Pedro J. Zufiria, Grupo de Redes Neuronales, Departamento de Matemática Aplicada a las Tecnologías de la Información, E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain. E-mail: pzz@mat.upm.es
† Also at E.T.S. Ing. Caminos, Canales y Puertos, UPM, 28040 Madrid, Spain.
‡ Also at Lucent Technologies, Microelectronics Group, Polígono Industrial Tres Cantos Zona Oeste, Tres Cantos, 28760 Madrid, Spain.

Contract/grant sponsor: Proyecto Multidisciplinar de Investigación y Desarrollo, Universidad Politécnica de Madrid.
Contract/grant number: 14908
Contract/grant sponsor: Programa Sectorial de PGC de Dirección General de Enseñanza Superior e Investigación Científica
Contract/grant number: PB97-0566-C02-01.

CCC 0890–6327/99/040219–21$17.50

controllers to stabilise non-linear plants; a well-known example is the inverted pendulum stabilized around the vertical position.[3] In addition to Lyapunov methods, feedback linearization techniques, strongly supported in differential-algebraic geometry tools, appear as a promising field in the context of affine systems.[4–6]

Adaptive control in the framework of LTI systems is a mature field where the conditions for guaranteeing stability, in relation to the order of the system, relative degree, high-frequency gain, etc., are known.[7,8] In the case of non-linear plants, there is no general methodology to deal with the design of adaptive controllers, and very restrictive properties of the system must be assumed to guarantee a stable solution.[1] Finally, neural network-based approaches are emerging as a very promising alternative to more traditional methods.[9–11]

The present work is based on some results presented in the literature,[2,12–14] attempting to establish a formal mathematical foundation for the synthesis of control systems based on neural networks. In such context, alternative methods for designing neural adaptive controllers for non-linear plants are presented; these methods make use of a specialized identification scheme of the inverse system dynamics in the operational region of the plant. Here, we propose a general scheme for designing a controller based on a non-linear feedback law directly derived from the inverse dynamics of the plant and exploiting the well-known approximation properties of neural networks.[15,16]

The model employed considers the generic state-space description of a plant S

$$\begin{aligned} x(k+1) &= f[x(k), u(k)] \\ y(k) &= h[x(k)] \end{aligned} \tag{1}$$

where $x(k) \in X \subset R^n$, $u(k) \in U \subset R^r$, $y(k) \in Y \subset R^p$, n being the order of the plant, r the number of inputs, and p the number of outputs. Note that, in some cases, this discrete-time setting may come from the discretization of a continuous-time plant. For the sake of simplicity, SISO systems will be assumed, that is, $r = p = 1$. The plant is characterized in a region by a pair $\{S, \theta_i\}$, where S represents system equations (1) and θ_i is a vector of parameters representing the internal or external factors that can modify the plant dynamics (e.g. perturbation signals, plant parameter changes, or any kind of sensor or actuator malfunction that could influence the plant behaviour). Each pair is referred as a *functional environment* of the plant; hence, the plant will be represented by the set of pairs $\{S, \theta_i\}_{i=1,\ldots,L}$, where L represents the number of different functional environments in which our system is able to work. To make notation easier we consider $\{S, \theta_i\} = S_i$ for $i = 1, \ldots, L$.

The rest of the paper is organized as follows. In Section 2, the problem of inverse identification and control using neural networks is analysed. Formulations in both discrete- and continuous-time contexts are considered. Section 3 presents the multiple model approach for control design. Simulation results concerning the applicability of the proposed models to specific control problems are illustrated in Section 4. Particularly, the control of a robot arm is simulated. Finally, concluding remarks appear in Section 5.

2. NEURAL INVERSE IDENTIFICATION AND CONTROL

The control of dynamical systems based on the characterization of the plant inverse has several known drawbacks, related to minimum phase requirements in the LTI case. In general, stability of the original system and robustness with respect to initial conditions seem to be critical aspects

to be considered. Nevertheless, if stability of the plant in a given region of work is guaranteed *a priori*, the inverse approach can be useful helping to solve tracking problems. In this section, some basic concepts and results on inverse identification and control are presented.

2.1. Discrete-time dynamical systems

Given a dynamical system in the form of (1), the objective of the proposed control method is to find a control signal $u(k)$ that ensures $\| y(k) - y_D(k) \| < \varepsilon, \forall k$; that is, $u(k)$ has to drive the plant for *tracking* a reference signal $y_D(k)$, where $y(k)$ is the system output, and ε is a bound on the tracking error. This control signal can be easily computed if a right inverse of the plant

$$u(k) = S_R^{-1}[y_D(k)], \quad k \in N \tag{2}$$

is available for each step k. In this case, the design of the controller is reduced to characterize the inverse dynamics of the plant, and it is not dependent on the desired trajectory. Once the right inverse is available, the control signal can be directly obtained, applying (2) to the desired tracking trajectory.[17,18] This method is generally applicable in static problems where the existence of the right inverse can be proved.

In a dynamical context, the inverse of a plant may not be characterized by a standard state-space formulation, often requiring, for instance, non-causal computations. In general, for LTI plants, it is possible to define the so-called L-delay inverse systems, such that when cascaded with the plant, provide an output that is delayed L steps with respect to the input.[19] In non-linear dynamical systems as defined in (1), the determination of an inverse system may have more than one solution (or may have no solution at all) and therefore it is not possible to design a generic controller as indicated in (2). In this context, the characterization of invertibility of a non-linear plant must consider two aspects: first, the mandatory restriction to local characterizations; second, the above-mentioned limitation concerning the possibility of defining only a delayed inverse of the plant.[13]

In general, it should be assumed that the system output is stable in the domain defined by the reference trajectory; if it is not, one can consider that the system has been stabilized before the identification process, so that all systems addressed in the present study are BIBO (bounded input, bounded output).

The identification scheme to be proposed,[20] considers a generic model of non-linear systems that can be described by a difference equation as

$$y(k+1) = \phi[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)] \tag{3}$$

where n represents the order of the system. The use of these models can be justified if the system satisfies certain observability conditions.[13] For non-linear systems with an input–output representation as in (3), there exists a non-linear feedback law of the form:[13]

$$u(k) = \Theta[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1), y_D(k+d)] \tag{4}$$

where d is the relative degree of the plant in the operational region of state space defined by the reference trajectory.

A neural network (NN) architecture can be employed to learn the (delayed) inverse dynamics (4) of the plant for a given trajectory. The inverse identification scheme is depicted in Figure 1 (for the case of relative degree $d = 1$), where the NN is configured as a TDNN (time delay neural network).[21] For doing so, an input–output model of the plant described in (3) is only required, so

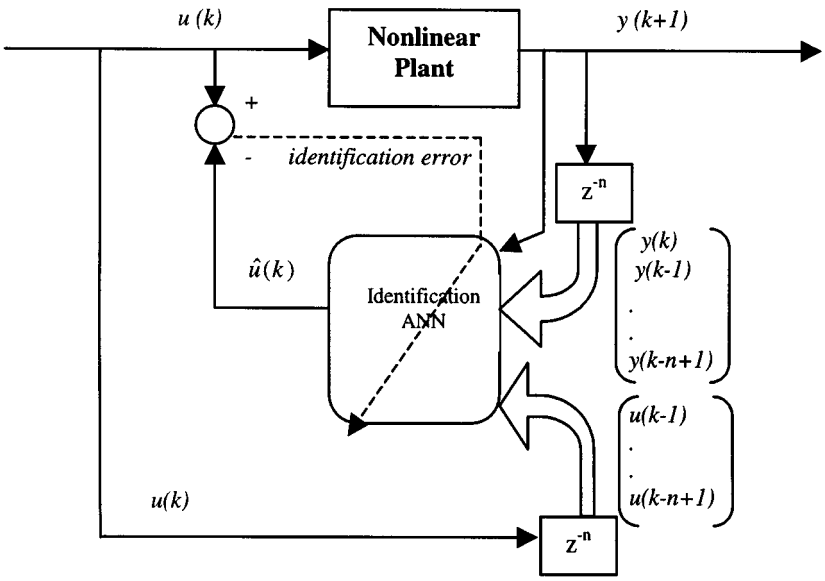

Figure 1. Inverse identification scheme

that even in the case of unknown or partially known ϕ (or f and h in (1)), this identification criterion is applicable.

The network training is performed considering the desired trajectory $y_D(k)$ in the actual functional environment. Details about the neural architecture and the training algorithm will be given in Section 4. The final objective of the identification is to obtain a NN-based accurate approximation of the delayed plant inverse dynamics in the operation domain.

A control procedure can be developed making use of the inverse dynamics of the plant in the domain of the output space defined by the reference signal $y_D(k)$. This inverse provides a delayed *specialized inverse model* of the plant that allows the computation of the control signal (2) from advanced values of tracking reference signal $y_D(k+d)$, where d is the relative degree of the plant.

The proposed control scheme is shown in Figure 2 (for the case $d=1$), aiming that the difference $\|y(k)-y_D(k)\|$ is below a defined limit error. The mapping Θ is approximated by the identification NN in the actual plant functional environment, and the NN controller is always a copy of the identification NN, being updated through inverse identification. An important feature is that the identification process continues while control is in progress; this way we achieve a more stable and robust behaviour of the controlled system. Hence, any fluctuation of plant dynamics can be approximated by the NN, performing an adaptive control action. It must be pointed out that if no model of the plant is available, the identification-control process can be done in an on-line fashion.

As will be shown in simulations, the on-line inverse identification procedure, while control is being performed, allows the plant to move from the original functional environment S_i to a neighbouring environment S_j without requiring a new identification process. Nevertheless, transient periods can be too long until the NN re-adapts its internal weights, and big changes

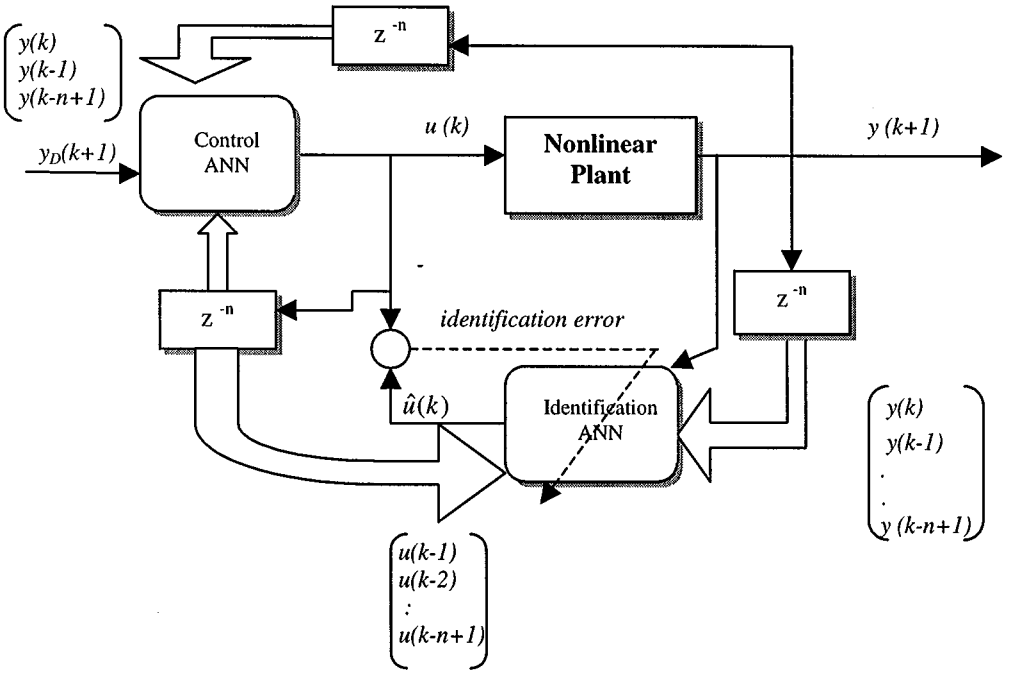

Figure 2. Control scheme

in the functional environment could even make the NN training fail; this fact will motivate the introduction of a multiple model to deal with these situations. This task is addressed in Section 3.

2.2. *Continuous-time dynamical systems*

When continuous-time systems are considered, invertibility and discretization issues make the control problem more involved. Concerning inverse identification, in the case of continuous-time dynamical systems, equivalent results to the ones existing for discrete-time systems, in the LTI context, exist for L-integral inverse computations.[19] In general, when a dynamical system of the form

$$x' = f(x, u, t) \tag{5}$$

is considered, controllability aspects related to the existence of a control law $u(t)$ such that $x_r'(t) - f(x_r(t), u(t), t) = 0$ (relying on the implicit function theorem) are crucial. In addition, the path-tracking problem so addressed makes sense if stability of the system is guaranteed in a region including the reference trajectory $x_r(t)$, so that if the initial conditions of the system do not lie on the reference trajectory a good tracking error is still guaranteed. A small perturbation analysis of the dynamics of $x - x_r$ concludes that stability along any trajectory in the region of interest guarantees that such difference is bounded.

Nevertheless, if the control design is directly framed in a continuous-time context, where the existence of continuous-time controllers may be guaranteed, the above-mentioned existing results

concerning identification and control of discrete systems may not be directly applicable. Particularly, the identification procedures may not be implementable in a straightforward manner with neural network architectures.

Here, some aspects concerning discretization of continuous-time systems for computer control purposes are presented. In addition, discretization of continuous-time control laws is also analysed. The study suggests the difficulty of defining general design procedures for continuous-time systems in the context of the above results for discrete-time neural identification and control. Here, we illustrate some of the ideas for a large variety of non-linear plants such as, for instance, mechanical systems

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= f(x_1, x_2) + g(x_1, x_2)u \end{aligned} \tag{6}$$

Note that, if a reference trajectory $x_{1r}(t)$, $x_{2r}(t)$ is pre-defined, assuming that $g(x_1, x_2)$ is invertible in the region of interest, the system may be so that $u(t)$ can be computed as

$$u(t) = g^{-1}(x_{1r}(t), x_{2r}(t))\,[x_{2r}'(t) - f(x_{1r}(t), x_{2r}(t))] \tag{7}$$

Since this expression involves derivatives, it is not framed in the standard integral state space formulation. Note that this fact does not affect the computation of the control law in an open-loop formulation. Nevertheless, as mentioned above, this approach is only applicable when stability conditions are guaranteed *a priori* in the region of interest, and the system satisfies controllability conditions as in (7). Interesting open issues, beyond the scope of this section, concerning performance and stability may be studied when some of the terms in (7) are obtained by feedback from the system state-space variables. For computational purposes, equation (7) has to be discretized, where different discrete control formulations can be obtained, as shown below. In that case, a rigorous analysis of the overall control system would require a hybrid (continuous for the plant and discrete for the controller) formulation, whose study must be addressed in a more complex context. As an alternative, discretization of the plant equations may serve as an approximate modelling which leads to a pure discrete formulation very suitable for applying the above presented results.

Hence, discretization of (6) with a sampling period Δt so that $x(t_0 + k\Delta t) = x(k)$, and using Runge–Kutta methods provides an expression which depends on the order of the method

$$\begin{aligned} x_1(k+1) &= x_1(k) + \Delta t x_2(k) + \tfrac{1}{2}\Delta t^2[f(x_1(k), x_2(k)) + g(x_1(k), x_2(k))u(k)] + \cdots \\ x_2(k+1) &= x_2(k) + \Delta t[f(x_1(k), x_2(k)) + g(x_1(k), x_2(k))u(k)] + \cdots \end{aligned} \tag{8}$$

where such selected order determines the properties of the resulting discrete system.

2.2.1. Examples for methods of order 1. For instance, in the case of RK1 (Euler's method) we get the following discrete-time formulation:

$$\begin{aligned} x_1(k+1) &= x_1(k) + \Delta t x_2(k) \\ x_2(k+1) &= x_2(k) + \Delta t[f(x_1(k), x_2(k)) + g(x_1(k), x_2(k))u(k)] \end{aligned} \tag{9}$$

where a control variable $u(k)$ may be constructed accordingly for obtaining an inverse of the discrete plant. Equation (7) suggests the construction of the inverse of the plant defined in (6), so

that a discretization of (7) might provide an approximation to the inverse of the discrete system defined in (9). Equation (7) can be discretized in several ways using also a first-order finite-difference backward formulation:

$$u(k) = g^{-1}(x_{1r}(k-d), x_{2r}(k-d))\left[\frac{x_{2r}(k+1-d) - x_{2r}(k-d)}{\Delta t} - f(x_{1r}(k-d), x_{2r}(k-d))\right] \quad (10)$$

so that the causality of the control law is preserved for $d \geqslant 1$. Note that, following equation (4), non-causal terms can be employed as values of a reference trajectory known ahead in time. A key issue arises now concerning the relationship between both discrete-time dynamical systems (9) and (10); it is easy to prove that system (10) corresponds to the d-unit delay right inverse of system (9). For proving so, let us consider the system resulting from the cascade of systems (9) and (10). Then, the relationship in the overall cascaded system between the output variables $[z_1(k), z_2(k)]$ and the input variables $[x_{1r}(k), x_{2r}(k)]$ is obtained via direct substitution, leading to

$$\begin{aligned} z_1(k+1) &= z_1(k) + \Delta t z_2(k) \\ z_2(k+1) &= z_2(k) + \Delta t\Bigg[f(z_1(k), z_2(k)) + g(z_1(k), z_2(k))\, g^{-1}(x_{1r}(k-d), x_{2r}(k-d)) \\ &\quad \times\left(\frac{x_{2r}(k+1-d) - x_{2r}(k-d)}{\Delta t} - f(x_{1r}(k-d), x_{2r}(k-d))\right)\Bigg]. \end{aligned} \quad (11)$$

Rearranging the equations we get

$$\begin{aligned} z_1(k+1) &= z_1(k) + \Delta t z_2(k) \\ g^{-1}(z_1(k), z_2(k))[z_2(k+1) - z_2(k) - \Delta t(f(z_1(k), z_2(k)))] \\ &= g^{-1}(x_{1r}(k-d), x_{2r}(k-d))(x_{2r}(k+1-d) - x_{2r}(k-d) - \Delta t f(x_{1r}(k-d), x_{2r}(k-d)))]. \end{aligned} \quad (12)$$

Note that equations (12) are satisfied for all k, under the condition $[z_1(k), z_2(k)] = [x_{1r}(k-d), x_{2r}(k-d)]$.

If a forward difference formulation is employed (i.e. $d = 0$) note that a 0-delay inverse is obtained at the cost of violating causality in the computation of $u(k)$. A key aspect in both cases is that the resulting dynamical system is not asymptotically stable, so that any disturbance in the system may steer the system far from the desired reference trajectory. Note that alternative formulations for system discretization need not guarantee such type of cascade input/output behaviour.

As mentioned above, following the formulation in equation (4) it may be possible to consider non-causal control laws; also feedback can be incorporated to the system so that stability properties could possibly improve. This aspect is related with the open issue mentioned above concerning feedback terms in (7) for the continuous-time laws. In general, the resulting control laws will not correspond with pure L-delay inverses of the system. In that feedback context, we can define several types of feedback control laws as, for instance,

$$u(k) = g^{-1}(z_1(k), z_2(k))\left[\frac{x_{2r}(k+1) - z_2(k)}{\Delta t} - f(z_1(k), z_2(k))\right] \quad (13)$$

where the resulting cascade system does not have asymptotic stability. Working with second-order differences for $u(k)$, it can be shown that if we define

$$u(k) = g^{-1}(z_1(k), z_2(k))\left[\frac{x_{2r}(k+l) - 2z_2(k) + z_2(k-l)}{l^2\Delta t^2} - f(z_1(k), z_2(k))\right], \tag{14}$$

the smaller value of l leading to asymptotic stability of the resulting system is $l = 4$.

2.2.2. Methods of order larger than 1. A clear improvement when considering higher order methods is the fact that the control variable u affects directly on the position state variables x_1. On the other hand, terms corresponding to the functions f and g cannot be easily cancelled by selecting an appropriate expression for u. For simple cases, such as for instance $f \equiv 0$ and $g \equiv 1$, it can be shown that appropriate approximations for $u(k)$, $u(k + \frac{1}{2})$ and $u(k + 1)$ making use of both x_{1r} and x_{2r}, can lead to asymptotically stable feedback formulations in the case, for instance, of improved tangent (order 2) method, or the RK4 scheme.

As a conclusion, a formal study of the possible discrete control strategies for obtaining a delay inverse identification of the plant (8) is quite involved. The performance seems to strongly depend on the selected discretization strategies and order of the methods to obtain the discrete control strategy from (7) and the discrete system (8) from (6). In addition, the stability of the resulting scheme cannot be easily guaranteed.

These limitations support the need of using neural network architectures oriented to approximate the inverse of the plant (6), specially when functions f and g are not precisely known. The neural identification allows the use of the control strategy proposed in Figure 2.

3. MULTIPLE MODEL APPROACH

The multiple model approach attempts to provide a better performance in situations where the plant is intended to evolve in several (qualitatively or quantitatively) different environments. Generally speaking, different controllers might be used, each one associated with a specific model of the plant or adapted to different complexities in the environment. In the specific case of the scheme here proposed, based on inverse identification, the fact that the controller is defined by an inverse model of the plant leads to a one-to-one correspondence between models and controllers. Therefore, the single model previously presented must be understood as a one-model/one-controller scheme, while the multiple model to be introduced in this section corresponds to a multiple-model/multiple-controller approach.

The multiple-model approach to be proposed here, makes use of a representation of the plant by $\{S, \theta_i\}$ for each ith functional environment.[13] Based on this model, we can generate the control signal $u(k)$ for each value k pertaining to the range of operation, so that tracking of the reference trajectory $y_D(k)$ is ensured. In the single-model approach, whenever the functional environment changes, the NN can adjust its weights through an on-line identification process. However, in situations with drastic environmental changes in the environment, the performance of the single-model control scheme may not be accurate enough.

Assuming that the complete dynamic operational range of the plant can be split into different functional environments, the plant dynamics can be characterized in a finite set of pairs $\{S, \theta_i\}_{i=1,\ldots,L}$, where $L \in N$. For the sake of simplicity it can be further assumed that there are few differences in plant dynamics between environments in the same neighbourhood, that is, the

functional environments $\{S, \theta_i\}$ and $\{S, \theta_{i+1}\}$ are closer in plant behaviour than $\{S, \theta_i\}$ and $\{S, \theta_{i+2}\}$. These differences become meaningful when the plant has to perform some task that implies the change of dynamic conditions between environments not belonging to the same neighbourhood, that is, with different plant dynamics conditions. In these situations, the single-model scheme, based on a unique NN, may be deficient.

A multiple-model scheme may, however, deal with these situations by means of an array of pairs inverse model/neural controller. Different functional environments are addressed by distinct controllers, and the control problem is reduced to switching between the NNs corresponding to the inverse model of the actual operation environment. Two possible settings must be considered in the design of this kind of systems:

1. Knowledge of the plant might be sufficient to previously define the sequence of environments the plant has to switch through to complete its operative cycle. In this case, switching between the corresponding NN's is automatic and only the near past of the system should lead to a short transient period when the environment changes. Simulation results in Section 4.1 illustrate this situation.
2. If this knowledge is not available, an external switching criterion, deciding when the plant enters a new environment, must be used.[22] Environment switching may be detected, in this case, by means of the identification error.[23,14] This strategy is detailed in Section 4.2.

The final objective of the control scheme is to provide a policy that assures stability and improves robustness of the overall controlled system inside any of the different dynamic conditions the plant can switch to in the functional environment space.

4. SIMULATION RESULTS

In this section, we present some simulation results which illustrate the behaviour of the models presented above. Section 4.1 addresses the case of a discrete system with high relative degree, while in Section 4.2 the control of a robot arm is simulated. In both cases, the performance of the single and multiple model approaches are compared.

4.1. System with high relative degree

The system considered here has been previously discussed[13] as an example of non-linear system with relative degree greater than one ($d = 3$). Here, it is employed to illustrate the performance of the single- and multiple-model schemes, under the assumption of the sequence of operation environments being known. For comparative purposes, the same sequence was applied to both schemes.

The state-space representation of the model is given by the system of equations:

$$\begin{aligned} x_1(k+1) &= a \cdot x_3(k) \\ x_2(k+1) &= x_1(k) + (1 - c \cdot x_2(k)) \cdot u(k) \\ x_3(k+1) &= b \cdot x_1(k) \cdot x_3(k) - x_2(k) \\ y(k) &= x_1(k) \end{aligned} \tag{15}$$

and the desired trajectory is a sine function $y_D(k) = \sin(k\pi/100)$. Following (4), the inverse identification approach requires the desired output of the plant for $k + d$ to be known at step k. If we consider this problem in a MRAC (Model Reference Adaptive Controller) approach,[8] the desired trajectory $y_D(k + d)$ corresponds with the output of a reference model that is a pure delay of d units: $r(k) = y_D(k + d)$, where $r(k)$ is the reference signal.

Function approximations were carried out by using a three-layer Multilayer Perceptron configured as TDNN, and trained via the static backpropagation algorithm. The training was improved by adding a momentum term. An NN(6, 25, 10, 1) network was used, being 6, 25, 10, 1 the number of neurons of the input, first hidden, second hidden and output layers, respectively. In particular, the six input neurons correspond to the signals $y(k), y(k-1), y(k-2), u(k-1), u(k-2)$ and $y_D(k+3)$, and are given by the fact that both the order and the relative degree of the system are 3, while the output is the control signal $u(k)$. The training algorithm was applied until a convergence error of 0·001 was reached for the dynamic range defined by the reference trajectory.

Four possible different environments are considered, corresponding to the following different parameter vectors of coefficients $[a, b, c]$: $S_1 = [0{\cdot}5, 0{\cdot}3, 0{\cdot}4]$, $S_2 = [0{\cdot}9, 0{\cdot}7, 0{\cdot}4]$, $S_3 = [0{\cdot}7, 0{\cdot}2, 0{\cdot}4]$ y $S_4 = [0{\cdot}7, 0{\cdot}3, 0{\cdot}6]$. The sequence of environments started with S_1, changing to S_2 at $k = 100$, to S_3 at $k = 200$, to S_4 at $k = 400$, to S_2 at $k = 600$ and, finally, to S_3 at $k = 800$.

4.1.1. Single-model approach. In this model, a single NN is used for the identification of all the above-mentioned environments. The network adapted its weights through the on-line identification and kept an acceptable tracking of the reference signal for $k \in [0, 100]$; nevertheless, a clear degradation of the control action was observed after the first environmental switch (at $k = 100$), as shown in Figure 3.

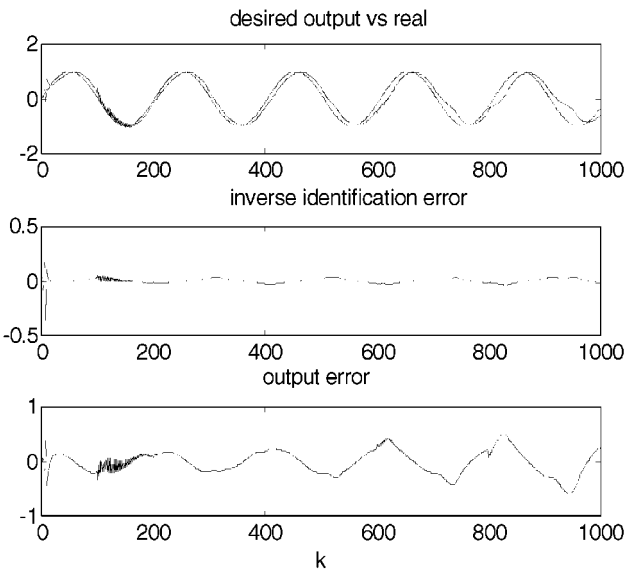

Figure 3. System with high relative degree: single model

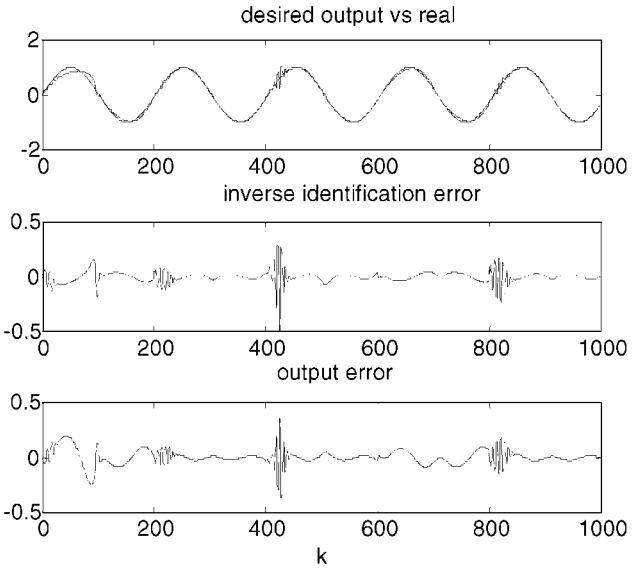

Figure 4. System with high relative degree: multiple model

When environmental changes are less drastic, the adaptive nature of the scheme could allow the use of the single model approach to control the system.[24]

4.1.2. Multiple-model approach. The results of a simulation performed with a multiple-model controller are shown in Figure 4, where the plant evolves through the same sequence of environments. In this case, whenever the plant moves to the next functional environment, the corresponding NN is switched to perform a new control action. The assumed knowledge of the sequence of environments allows an automatic switching between the networks, as mentioned above. The switching occurs in the context of a change in the dynamic condition of the plant, belonging the near past to a different dynamic setting; as a consequence, a transient period arises until the system dynamics gets completely into the new environment. If the plant may be reset to a known initial state, every time it enters a new functional environment, the transient period can be reduced, or even suppressed, providing a better performance for the system.

4.2. Robot arm controller

As a practical application of the proposed method, we present here the simulation of a robot arm controller based on an on-line learning scheme. In Section 4.2.1, the single inverse model previously introduced is applied to the control of the robot arm, which operates always in the same environment, while a sequence of functional environments is addressed through a multiple model in Section 4.2.2. No *a priori* knowledge of this sequence is assumed and, therefore, a performance index is used for environmental switching.

We have used the two-link SCARA robot arm model[25] for simulation purposes (Figure 5). The dynamic equation of such a n-link rigid robot arm, in vector form, is

$$T = M(q)q'' + V(q, q') + F(q') + G(q) \tag{16}$$

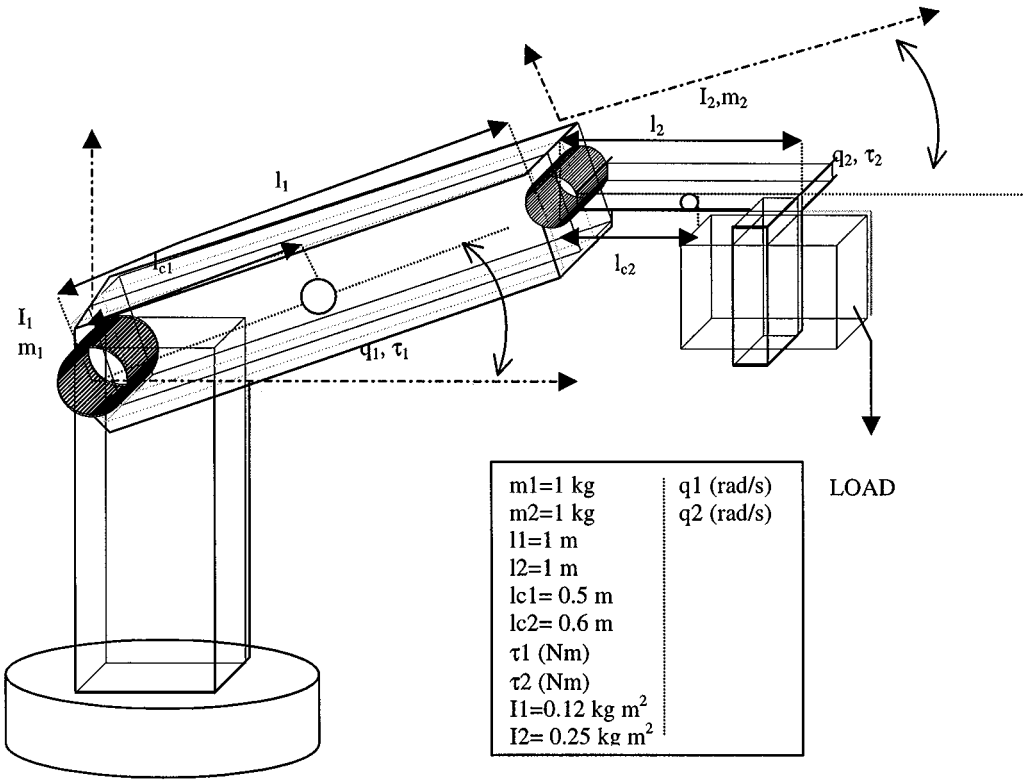

Figure 5. Robot arm

$M(q)$ being the manipulator inertia matrix, while $V(q, q')$ is the vector representing centrifugal and Coriolis effects, F represents friction forces and G represents gravity; (q, q', q'') are the vectors containing joint positions, velocities and accelerations. Specifically, the equations of the system are

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} q_1'' \\ q_2'' \end{bmatrix} + \begin{bmatrix} -hq_2' & -hq_1' - hq_2' \\ hq_1' & 0 \end{bmatrix} \begin{bmatrix} q_1' \\ q_2' \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{17}$$

where

$$\begin{aligned} g_1 &= m_1 l_{c1} g \cos q_1 + m_2 g[l_{c2} \cos(q_1 + q_2) + l_1 \cos q_1] \\ g_2 &= m_2 l_{c2} g \cos(q_1 + q_2) \\ H_{11} &= m_1 l_{c1}^2 + I_1 + m_2[l_1^2 + l_{c2}^2 + 2 l_1 l_{c2}^2 \cos(q_2)] + I_2 \\ H_{12} &= H_{21} = m_2 l_1 l_{c2} \cos(q_2) + m_2 l_{c2}^2 + I_2 \\ H_{22} &= m_2 l_{c2}^2 + I_2 \\ h &= m_2 l_1 l_{c2} \sin(q_2) \end{aligned} \tag{18}$$

Note that the system defined in (16) can be rewritten in the form of equation (6),

$$q'' = M^{-1}(q)[-V(q, q') - F(q') - G(q)] + M^{-1}(q)T \tag{19}$$

so that the study developed there is applicable to this case. This fact supports the use of the discrete inverse neural identification approach proposed in Section 2.

Equation (16) could be seen as a way of characterizing the inverse dynamics of the robot arm: for a desired trajectory (q_d, q'_d, q''_d) the required torque can be obtained, being expressed as $T = \phi(q, q', q'')$, where ϕ represents such inverse dynamics. Nevertheless, if the problem is to be tackled within the framework presented in previous sections, only state variables should be employed for defining inverse dynamics. Our purpose is to generate the required torque for tracking the desired trajectory. This torque is based on the system inverse identification through the training of a Neural Network, as exposed in Section 2, where a discretization of the plant equations for computer neural control was introduced.

As proved in previous works,[2] there exits a nonlinear feedback law of the form

$$u(k) = \Theta[q_1(k), q'_1(k), q_2(k), q'_2(k), q_{1d}(k+1), q'_{1d}(k+1), q_{2d}(k+1), q'_{2d}(k+1)] \tag{20}$$

where $q_1(k)$, $q'_1(k)$, $q_2(k)$, $q'_2(k)$ represent real position and velocity of the articulation 1 and 2 and $q_{1d}(k+1)$, $q'_{1d}(k+1)$, $q_{2d}(k+1)$, $q'_{2d}(k+1)$ represent desired position and velocity of both articulations one step ahead.

A NN(8, 25, 15, 2) network was used, and the training was performed as in the previous section. The inputs of the controller network correspond to the signals $q_1(k)$, $q'_1(k)$, $q_2(k)$, $q'_2(k)$, $q_{1d}(k+1)$, $q'_{1d}(k+1)$, $q_{2d}(k+1)$ and $q'_{2d}(k+1)$, while the outputs are the torques of both articulations. The sampling time was 10^{-3} s.

4.2.1. Single-model approach. The structure of the controller is shown in Figure 6 and relies in two different sub-systems; first, the plant is driven by a PD controller (with parameters $k_d = 100$, $k_p = 20\,k_d$) that ensures a bounded output for the domain defined by the reference trajectory. Second, the inverse identification network begins to generate control actions as identification progresses. The total applied torque is obtained as the sum of the neural controller and the PD loop contributions. In the limit, when the identification is complete, almost all the control torque is generated by the neural controller, assuming a small remaining identification error. Similar controllers have been proposed in previous works.[10,25]

Assuming that the desired trajectories for each one of the two articulations (Figure 7) and the model (19) are known, the desired reference torque for the given trajectory can be computed through (16) for comparative evaluation purposes. After convergence, the real torque applied to the robot is very close to the ideal value (Figure 8), so the vector of position error $(q_{1d}(k) - q_1(k), q_{2d}(k) - q_2(k))$, will converge to zero (Figure 9).

It is important to note that the method allows us to perform nonlinear feedback control of nonlinear plants using static backpropagation,[2,12] which is computationally less costly than the dynamic version. Nevertheless, simulations have shown that backpropagation has a great dependence on the initial conditions of the system and on the network structure. Other neural paradigms[10,26] might reduce this dependence, but should not significantly affect the applicability and viability of the proposed model; for the sake of simplicity, static backpropagation has been used, being however the results representative of the general controller behaviour.

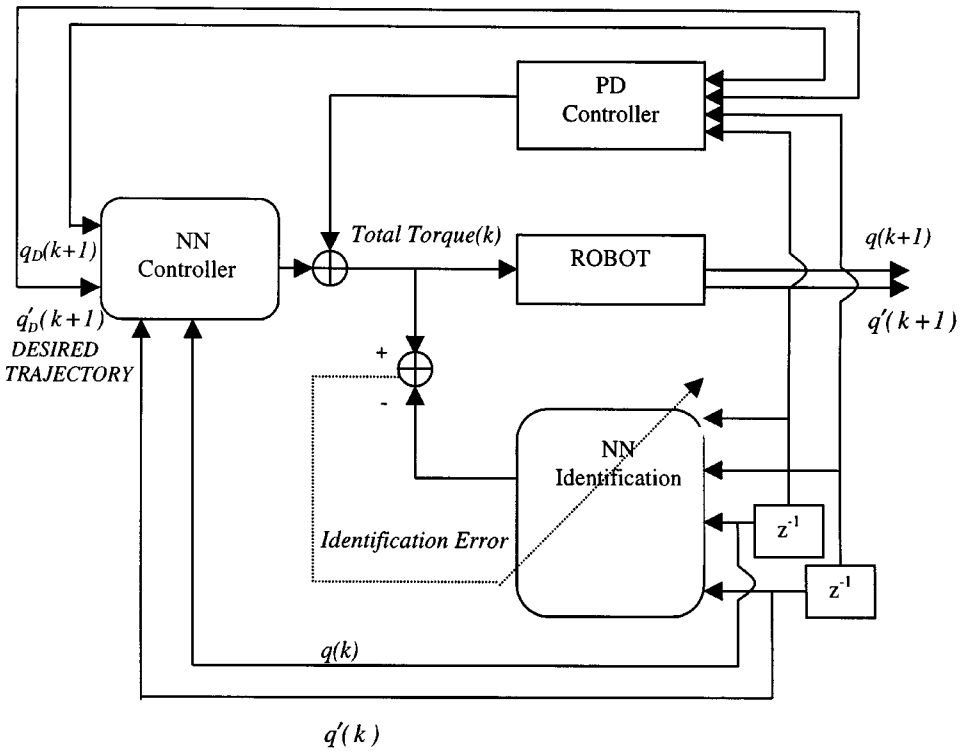

Figure 6. Single model for robot arm control

4.2.2. Multiple model approach. In this section, the control of the robot arm under a previously unknown sequence of environments is addressed using the multiple-model setting. In this case, a PD controller is combined with an array of neural controllers[23] which is displayed in Figure 10. Note that the PD controller guarantees some minimal good performance of the system (stability, etc.). Then, a NN-based control loop improves the performance for the different environments in an adaptive manner.

Desired trajectories for both joints are defined by

$$\begin{aligned} q_{1\mathrm{d}}(t) &= 3 + 6(\sin(t) + \sin(2t)) \\ q'_{1\mathrm{d}}(t) &= 6(\cos(t) + 2\cos(2t)) \\ q''_{1\mathrm{d}}(t) &= -6(\sin(t) + 4\sin(2t)) \\ q_{2\mathrm{d}}(t) &= 2 + 4(\sin(t) + \sin(3t)) \\ q'_{2\mathrm{d}}(t) &= 4(\cos(t) + 3\cos(3t)) \\ q_{2\mathrm{d}}(t) &= -4(\sin(t) + 9\sin(3t)) \end{aligned} \tag{21}$$

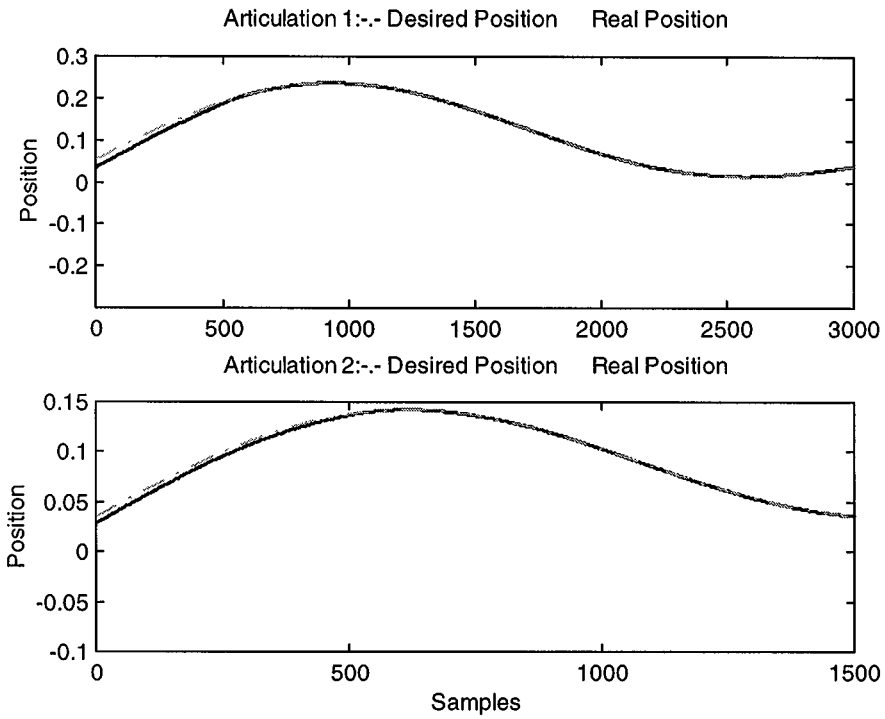

Figure 7. Desired and real trajectories of both articulations

If the robot arm has to deal with different loads, changing the m_2 parameter of our model, a single neural model might not be adequate for the inverse identification of the real plant. If a single identification-control model is used for the different environments, its adaptation may be very slow and a large transient error might be generated. In the worst case, the single NN architecture could be no valid any longer. Hence, multiple models are required to identify and control different situations. In this case, if specific models are available for different environments, knowing *a priori* the mass of different loads, corresponding controllers can be designed. During system operation, identification of the correct environment is needed in order to select the appropriate controller, in the so-called switching phase. If changes in the plant are small, the NN controller has to be tuned in order to improve its action (tuning phase). In the switching phase, one needs to determine the moment in which the current model is not appropriate as well as the model to replace it (i.e. it is necessary to know when to switch and what to switch to). In the tuning phase, the controller is to be adjusted at each sampling step.[14,23]

The learning process was performed by training a specific controller for each different environment. So, there are several identification-controller model pairs, represented by (I_j, C_j) where $j = 1, \ldots, N$ denotes the number of different environments. Environment switching was detected through the inverse identification error provided by the identifying network. The employed switching scheme was proposed in previous works.[14,23] The switching module is

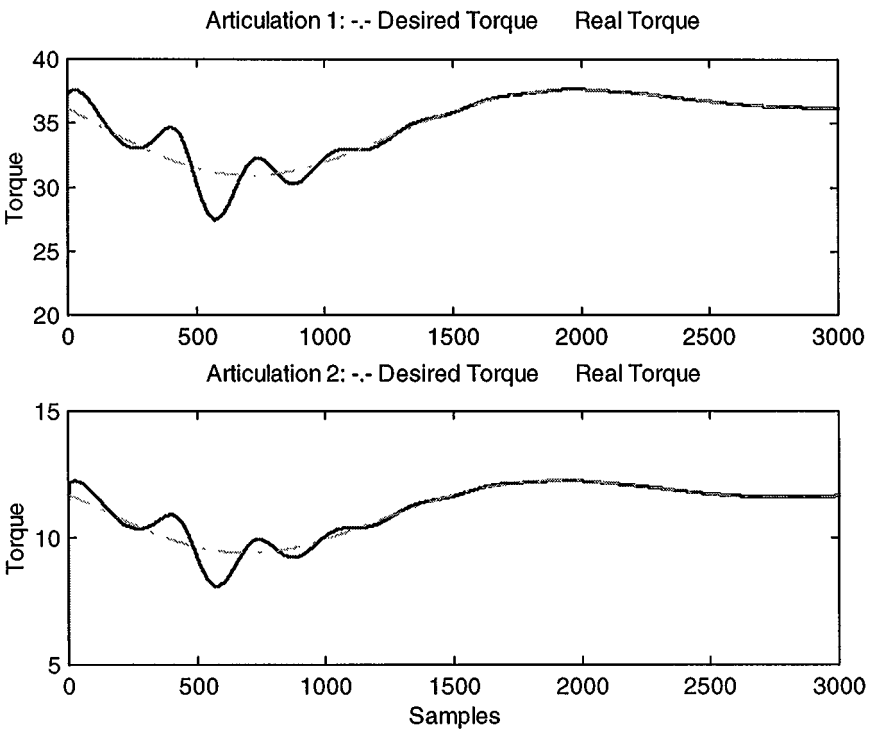

Figure 8. Real and desired torques

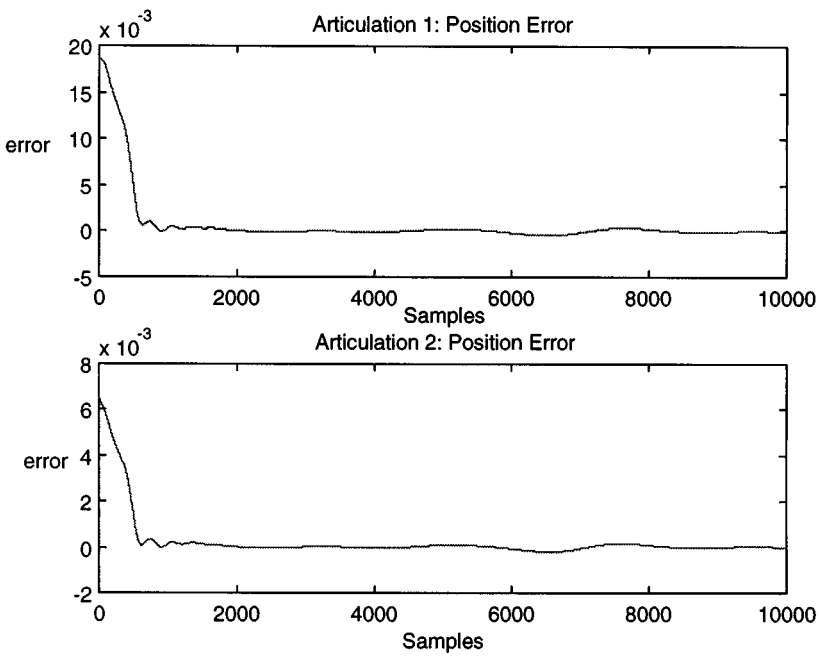

Figure 9. Position error

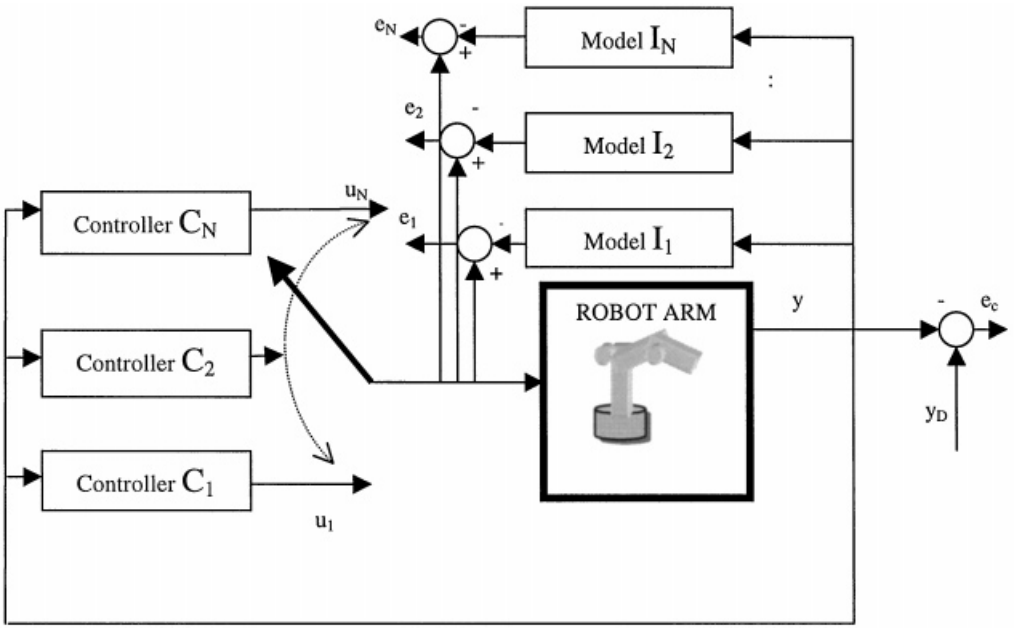

Figure 10. Array of neural controllers for the robot arm

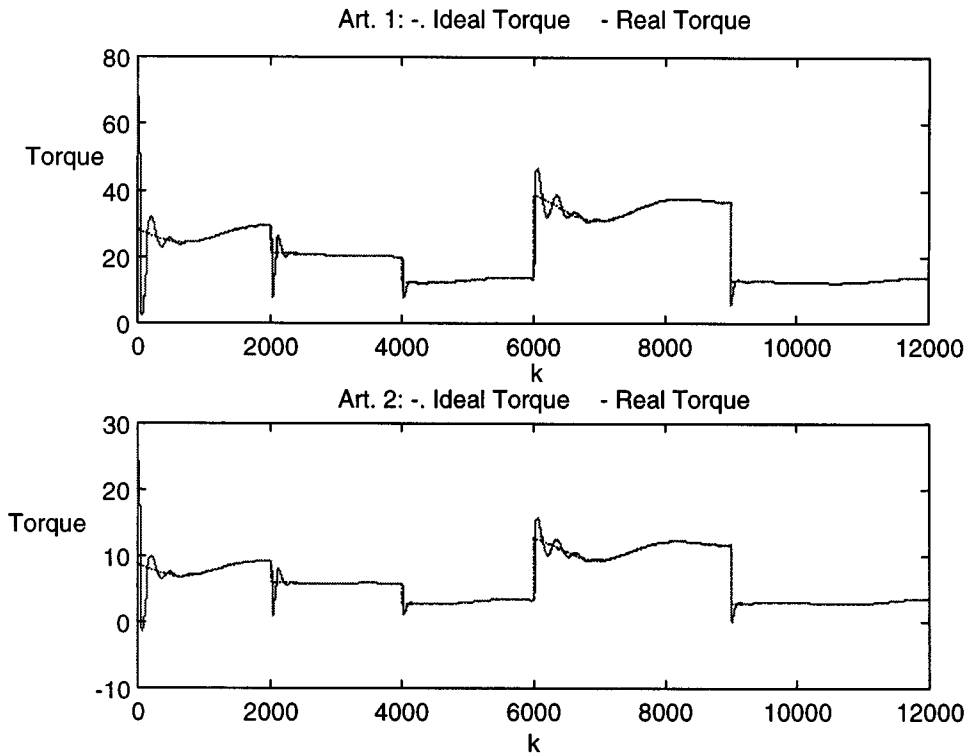

Figure 11. Real and desired torque applied to the two joints

monitoring a performance index based on the identification error for each model

$$J_j(t) = \alpha e_j^2(t) + \beta \int_0^t e^{-\lambda(t-\tau)} e_j^2(\tau)\, d\tau \tag{22}$$

The two terms in this performance index consider both instantaneous and long-term measures. The integral term depends on a forgetting factor, λ, which determines the long-term memory of this index.

The switching module selects the controller with the smallest index at every instant of time. To avoid fast switching, a hysteresis algorithm is also included. In case that the pair (I_j, C_j) is being used at instant t, and $J_k(t) = \min_i\{J_i(t)\}$, then the new pair, (I_k, C_k) will only be selected if $J_j(t) \geqslant J_k(t) + \delta$, for some $\delta > 0$; otherwise the former pair will be retained.

In simulations, the method has been applied to the robot arm, under the following sequence of values: $m_2 = 1{\cdot}5$ for $k \leqslant 2000$, $m_2 = 1$ for $2000 < k \leqslant 4000$, $m_2 = 0{\cdot}5$ for $4000 < k \leqslant 6000$, $m_2 = 2$ for $6000 < k \leqslant 9000$, and $m_2 = 0{\cdot}5$ for $9000 < k \leqslant 12\,000$.

The comparative analysis of the applied control torques versus the desired ones (Figure 11) shows that, each time m_2 changes in value, the system switches to the adequate controller, giving a good global performance. Position error (tracking error) is small over the entire interval, but changes in m_2 lead to small bounded fluctuations during the transition period (Figure 12).

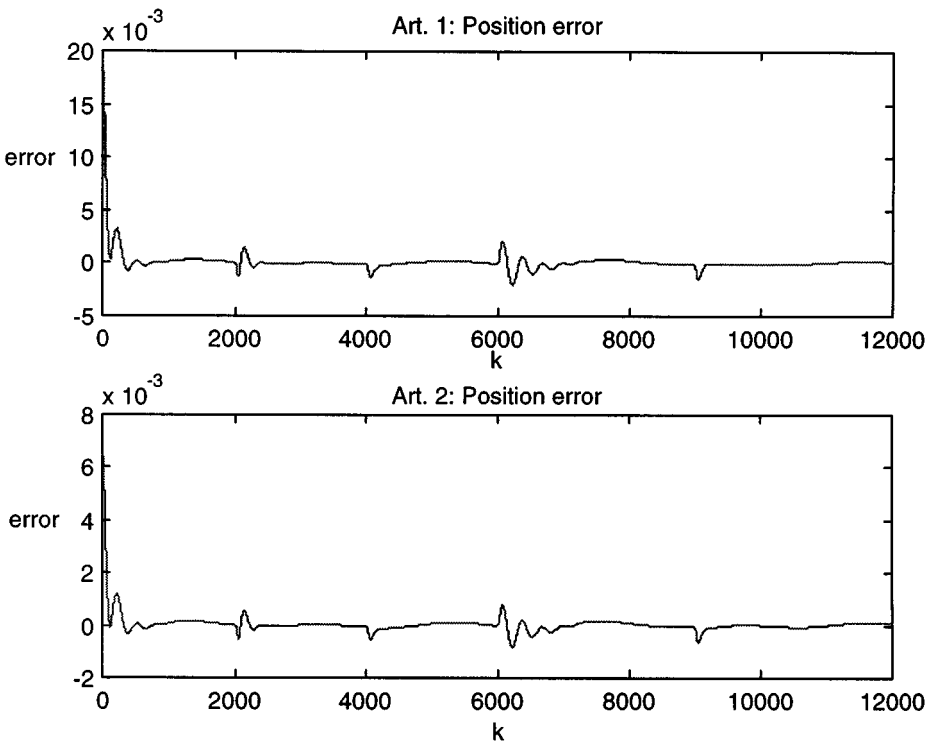

Figure 12. Position error

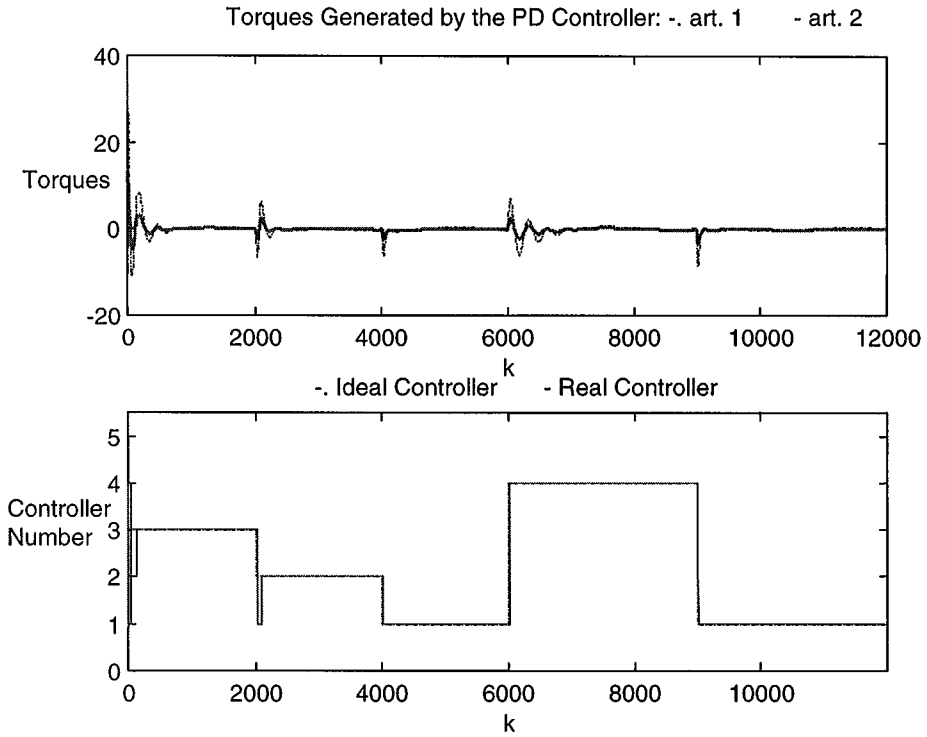

Figure 13. Contribution of the PD controller and selected and ideal controller

Figure 13 shows the switching scheme and the contribution of the PD controller to the global control action. After some transient period, where several incorrect controllers are chosen, the adequate controller is selected. Contribution of the PD controller rises considerably every time m_2 changes in value.

5. CONCLUDING REMARKS

In this paper, several adaptive control schemes for non-linear systems have been presented. The first method is based on performing an on-line inverse identification through supervised training of a single feedforward neural network. Simulation results have shown that the proposed technique performs satisfactorily, even in systems with relative degrees greater than one. Related works[24] have shown that the overall resulting control system might also be applicable to situations with plant parameter changes. Also, the control of a two-link SCARA robot has been developed to illustrate the method in a robotics application context, showing good results in simulations.

Secondly, a multiple-model architecture has been proposed; this architecture for controlling non-linear time-varying plants has proved to perform very well in different situations. Tracking results in the control of a robot arm with changing model are very promising. More research is

needed for incorporating *a priori* knowledge of the plant, considering different trajectory sets, increasing the dimension of the functional environment space, etc.

The proposed schemes can be combined with alternative neural network paradigms which might improve their robustness and speed of convergence.

ACKNOWLEDGEMENTS

This work has been partially supported by Proyecto Multidisciplinar de Investigación y Desarrollo 14908 of the Universidad Politécnica de Madrid, and Project PB97-0566-C02-01 of the Programa Sectorial de PGC of Dirección General de Enseñanza Superior e Investigación Científica in the MEC of Spain. Ricardo Riaza would also like to acknowledge the support of a graduate fellowship from the Universidad Politécnica de Madrid.

REFERENCES

1. Vidyasagar, M., *Nonlinear System Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
2. Levin, A. U. and K. S. Narendra, 'Control of nonlinear dynamical systems using neural networks: controllability and stabilization', *IEEE Trans. Neural Networks*, **4**, 192–206 (1993).
3. Friedland, B., *Control System Design*: *An Introduction to State Space Methods*, McGraw-Hill, New York, 1991.
4. Fu, K. S., R. C. Gonzalez and C. S. G. Lee, *Robotics*: *Control*, *Sensing*, *Vision and Intelligence*, McGraw-Hill, New York, 1987.
5. Isidori, A., *Nonlinear Control Systems*, Springer, Berlin, 1989.
6. Slotine, J.-J. E. and W. Li, *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
7. Åström, K. J. and B. Wittenmark, *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.
8. Narendra, K. S. and A. M. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall. Englewood Cliffs, NJ, 1989.
9. Zurada, J. M., *Artificial Neural Systems*, West Publishing Company, 1992.
10. Zalzala, A. M. S. and A. S. Morris, *Neural Networks for Robotics Control*, *Theory and Applications*, Ellis Horwood, Chichester, 1996.
11. Agarwal, M., 'A systematic classification of neural-network-based control', *IEEE Control Systems Mag*., **17**, 75–93 (1997).
12. Narendra, K. S. and K. Parthasarathy, 'Identification and control of dynamical systems using neural networks', *IEEE Trans. Neural Networks*, **1**, 4–27 (1990).
13. Levin, A. U. and K. S. Narendra, 'Control of nonlinear dynamical systems using neural networks-part II: observability, identification and control'. *IEEE Trans. Neural Networks*, **7**, 30–42 (1996).
14. Narendra, K. S. and J. Balakrishnan, 'Adaptive control using multiple models', *IEEE Trans. Neural Networks*, **42**, 171–187 (1997).
15. Hornik, K., M. Stinchcombe and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, **2**, 359–366 (1988).
16. Girosi, F. and T. Poggio, 'Networks and the best approximation property', Artificial Intelligence Lab, *Memo. No. 1164*, MIT, Cambridge, MA, 1989.
17. Psaltis, D., A. Sideris and A. A. Yamamura, 'A multilayered neural network controller', *IEEE Control System Mag*., **8**, 17–21 (1988).
18. Kotta, U. 'Right inverse of a discrete time nonlinear system', *Int. J. Control*, **51**, 1–9 (1990).
19. Sain, M. K. and J. L. Massey, 'Invertibility of linear time-invariant dynamical systems', *IEEE Trans. Automat. Control*, **14**, 141–149 (1969).
20. Hao, J. and J. Vandewalle, 'Inverse identification for control of nonlinear dynamical systems by back propagation neural networks', *WCNN*, **3**, 317–320 (1993).
21. Hush, D. R. and B. G. Horne, 'Progress in supervised neural networks. What's new since Lippmann?', *IEEE Signal Process. Mag*., **10**, 8–39 (1993).
22. Alonso, J. I., P. J. Zufiria and K. Membrut, 'Inverse adaptive control of non-linear plants via a multiple model approach. Application to a robot', *Engng. Benefits Neural Networks*, **EANN98,** 114–117 (1998).
23. Narendra, K. S., J. Balakrishnan and M. Kemal Ciliz, 'Adaptation and learning using multiple models. Switching and tuning', *IEEE Control Systems Mag*., **15**, 37–51 (1995).

24. Alonso, J. I., J. Cires and P. J. Zufiria, 'Inverse adaptive control of non-linear plants using neural networks. Application to a robot arm controller', *Neural Networks Engng. Systems*, **EANN97**, 79–86 (1997).
25. Ananthraman, S. and D. P. Garg, 'Training backpropagation and CMAC neural networks for control of a SCARA robot', *Engng. Appl. Artif. Intell*, **6**, 105–115 (1993).
26. Sanner, R. M. and J.-J. E. Slotine, 'Direct adaptive control with gaussian networks', *NSL Report No. 910303*, March 1991.

