Nonlinear System Identification Using Additive Dynamic Neural Networks—Two On-Line Approaches

Robert Griñó, *Member, IEEE*, Gabriela Cembrano, and Carme Torras

***Abstract*—This paper proposes a class of additive dynamic connectionist (ADC) models for identification of unknown dynamic systems. These models work in continuous time and are linear in their parameters. Also, for this kind of model two on-line learning or parameter adaptation algorithms are developed: one based on gradient techniques and sensitivity analysis of the model output trajectories versus the model parameters and the other based on variational calculus, that lead to an off-line solution and an invariant imbedding technique that converts the off-line solution to an on-line one. These learning methods are developed using matrix calculus techniques in order to implement them in an automatic manner with the help of a symbolic manipulation package. The good behavior of the class of identification models and the two learning methods is tested on two simulated plants and a data set from a real plant and compared, in this case, with a feedforward static (FFS) identifier.**

***Index Terms*—Additive dynamic neural networks, identification, invariant imbedding theory, sensitivity analysis, variational calculus.**

I. Introduction

In THE LAST few years, a growing interest in the study of nonlinear systems in control theory has been observed. This interest stems from the need to give new solutions to some long-standing necessities of automatic control [1]: to work with more and more complex systems, to satisfy stricter design criteria, and to fulfill the previous points with less and less *a priori* knowledge of the plant.

In this context, a great effort is being made within the area of system identification, towards the development of nonlinear models of real processes. In addition to more classical identification methods such as NARMAX modeling [2], [3], a new set of methods has been developed recently which apply artificial neural networks to the tasks of identification and control of dynamic systems. These works are supported by two of the most important capabilities of neural networks: their ability to learn [4], [5] (based on the optimization of an appropriate error function) and their good performance for the approximation of nonlinear functions [6], [7].

At present, most of the works on system identification using neural networks are based on multilayer feedforward neural networks with backpropagation learning or more efficient variations of this algorithm. These methods have been applied to real processes and they have shown an adequate behavior [8]–[12]. It is important to remark that most of them use static discrete-time models that capture the dynamics of the real process through the use of tapped-delay lines in the model inputs and outputs [13], [14]. A number of drawbacks associated with this type of models may appear in the identification of complex dynamic systems, such as difficulties in selecting the appropriate number of required delays and, in some cases, poor identification performance when implemented on line, after training off line, due to training deficiencies. In order to avoid these limitations, recurrent neural networks with internal dynamics are adopted in several works [15]–[17]. A common feature of the above contributions is that they all work in discrete time leading to discrete-time models of the real continuous system. This causes a great dependence of the resulting models on the sampling period used in the process and no information is given about the model trajectories between the sampling instants. Furthermore, the theoretical support for a subsequent use of the generated models in controller design is insufficient.

For these reasons, this paper presents the use of continuous-time additive dynamic neural networks [18]–[21] to identify real processes. Additionally, the identification methods presented in the paper use on-line training, so that when the training error is low, the network model can be reasonably expected to have captured the dynamic behavior of the real process/system. This approach has several advantages with respect to the discrete-time tapped-delay line models [22]–[25].

- The number of configuration parameters (degrees of freedom) of the model is considerably lower. It is only necessary to specify the dimension of the state space, since the number of inputs and outputs is determined by their counterparts in the real process.
- The models obtained with this approach are in state–space form and work in continuous time, which is very interesting in order to apply differential geometric theory for nonlinear control [26].

The rest of this paper is organized as follows. Section II presents the architecture of the additive dynamic connectionist

Manuscript received March 11, 1998; revised August 3, 1998. This work was supported in part by a grant from the Asociación/Colegio de Ingenieros Industriales de Cataluña and the Comisión Interministerial de Ciencia y Tecnologia (CICYT) under Project TAP97-0969-C03-01. This paper was recommended by Associate Editor J. M. Zurada.

R. Griñó is with the Instituto de Organización y Control de Sistemas Industriales, Universitat Politècnica de Catalunya, Barcelona, Spain (e-mail: grino@ioc.upc.es).

G. Cembrano and C. Torras are with the Instituto de Robótica e Informática Industrial, Universitat Politècnica de Catalunya, Consejo Superior de Investigaciones Científicas, Barcelona, Spain (e-mail: cembrano@iri.upc.es; torras@iri.upc.es).

Publisher Item Identifier S 1057-7122(00)01496-3.

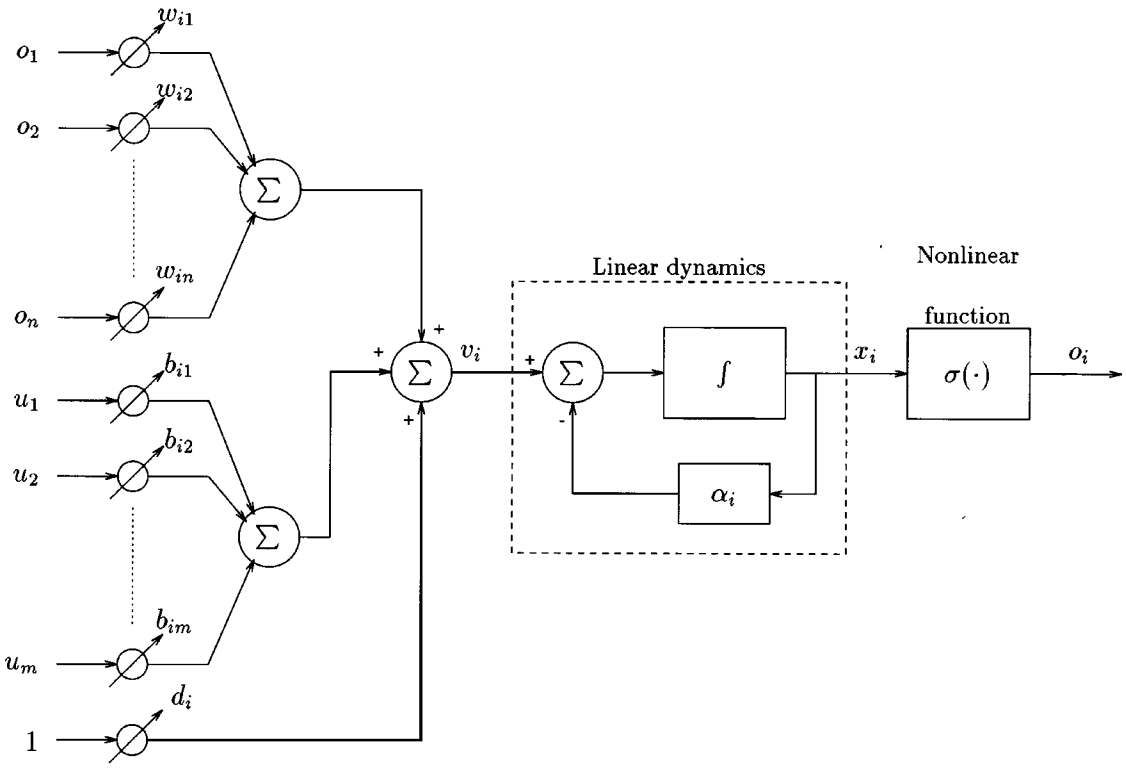

Fig. 1. Basic node in the additive neural model.

(ADC) models used in the system identification tasks. Section III develops two on-line parameter adaptation methods for the model of the previous section. More specifically, this section derives two on-line parameter update approaches, the first one based on a gradient method and sensitivity analysis of the model and the second based on variational calculus and invariant imbedding techniques. Then, Section IV discusses the implementation issues of the parameter adaptation methods of the previous section. In Section V, the developed methodologies are applied for the identification of several systems, including one represented by a real experimental data set of a hydroelectric power plant. Finally, Section VI summarizes the conclusions of the present work.

II. Architecture of the Connectionist Models

The architecture of the connectionist models is described by the structure of their basic elements and how they are interconnected.

A. Basic Elements

The basic processing element of a neural model is the node, also called the neuron by analogy with the biological neurons. In general, the basic model of a node is composed of a weighted adder, a linear dynamic SISO system, and a nonlinear static function. In this paper these elements are shown in Fig. 1.

- The weighted adder is described by the equation

$$v_i = \sum_{j=1}^{n} w_{ij} o_j + \sum_{k=1}^{m} b_{ik} u_k + d_i, \qquad i = 1 \cdots n \tag{1}$$

where the weighted sum v_i is a linear combination of the outputs of the nodes o_i of the net, the external inputs u_k, and the bias term d_i. Taking (1) and arranging them in matrix form gives

$$\boldsymbol{v} = \boldsymbol{W}\boldsymbol{o} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{d} \tag{2}$$

where

$\boldsymbol{W} = [w_{ij}] \in M_n(\mathbb{R})$[1] weight matrix of the network;
$\boldsymbol{B} = [b_{ij}] \in M_{n,m}(\mathbb{R})$ parameter matrix that applies the input vector to the model;
$\boldsymbol{d} = [d_i] \in \mathbb{R}^n$ bias vector of the network.

- The linear dynamic system has the weighted sum v_i as input and x_i as output. In this work, a first-order linear system with a variable time constant ($\tau_i = 1/\alpha_i$) and a static gain of value ($1/\alpha_i$) for each node in the network is chosen. Then, each node in the network has the following differential equation:

$$\dot{x}_i + \alpha_i x_i = v_i. \tag{3}$$

- The nonlinear static activation function selected in this work is the hyperbolic tangent and it maps the state of a node to its output ($\sigma\colon \mathbb{R} \to (-1, 1)$, $\sigma \in C^{\infty}$).

B. Additive Models

The composition of a number of the basic elements described above constitutes the additive dynamic connectionist (ADC)

[1] $M_{n,m}(\mathbb{R})$ is the set of n rows by m columns matrices with elements over the real field and $M_n(\mathbb{R})$ is the set of n rows by n columns matrices with elements over the real field;

model, each basic element being a node of the network. Arranged in matrix form all of the nodes, together with a linear output equation with constant parameters, has the following structure (see Fig. 2):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{W}, \boldsymbol{B}) = -\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{x}) + \boldsymbol{W}\boldsymbol{\sigma}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{d} \quad (4)$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\boldsymbol{x} \quad (5)$$

where

$\boldsymbol{x} \in \mathbb{R}^n$	state vector;
$\boldsymbol{d} \in \mathbb{R}^n$	bias vector;
$\hat{\boldsymbol{y}} \in \mathbb{R}^s$	output of the model;
$\boldsymbol{W} \in M_n(\mathbb{R})$	weight matrix;
$\boldsymbol{B} \in M_{n,m}(\mathbb{R})$	input weight matrix;
$\boldsymbol{C} = [c_{ij}] \in M_{s,n}(\mathbb{R})$	fixed output matrix;
$\boldsymbol{\sigma}(\boldsymbol{x}) = [\tanh(x_1), \cdots, \tanh(x_n)]^T$	C^∞ nonlinear vector field from $\mathbb{R}^n$ to (–1, 1).

It is important to note that the special form (a mixed matrix and Kronecker product[2] [27], [28]) of the first term in (4) is motivated by the need, for subsequent developments, to keep the vector structure for the time constant vector $\boldsymbol{\alpha}$. In particular, matrix $\boldsymbol{U} \in M_{n,n^2}(\mathbb{R})$ is formed as $\boldsymbol{U} = \sum_{j=1}^{n} \boldsymbol{D}_j \otimes \boldsymbol{e}_j^T$ where $\boldsymbol{D}_j = \operatorname{diag}\{0, \cdots, 0, \overset{j)}{1}, 0, \cdots, 0\} \in M_n(\mathbb{R})$ and $\boldsymbol{e}_j$ is the jth vector of the standard basis of $\mathbb{R}^n$.

A study of the absolute stability of the class of continuous-time additive dynamic neural network models defined by (4) and (5) has been carried out, obtaining a set of sufficient conditions developed from a frequency domain point of view [29].

III. Parameter Adaptation in Dynamic Neural Networks

The basic idea of the identification process is to arrange the connectionist model in parallel with the real plant, i.e., the model receives the same inputs as the plant and its outputs predict the values of the plant outputs (see Fig. 3). Clearly, the objective is to have the same output signals from the plant $\boldsymbol{y}$ and the model $\hat{\boldsymbol{y}}$ at any t. Since the plant is structurally and parametrically unknown, this will only be possible if the model is able to identify the class of systems to which the plant belongs. Neural networks have been shown to be good universal approximators, for example, in [7], [30]–[33]. In particular, the capabilities of continuous-time recurrent neural networks for the approximation of dynamic systems were exposed in [34]. In these conditions, it is reasonable to assume that the proposed models are capable of approximating the plant output, except for a residual error due to the structural modeling mismatch between the plant and the neural network model.

Then, parameter adaptation or learning techniques are required to perform the identification. In this work, two on-line adaptation methods are proposed.

[2]Also called the tensor product.

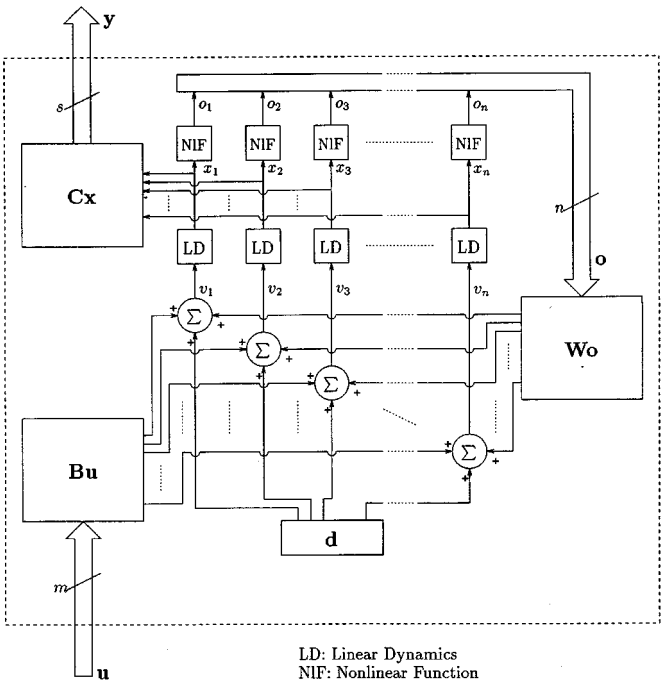

Fig. 2. Additive dynamic neural network model.

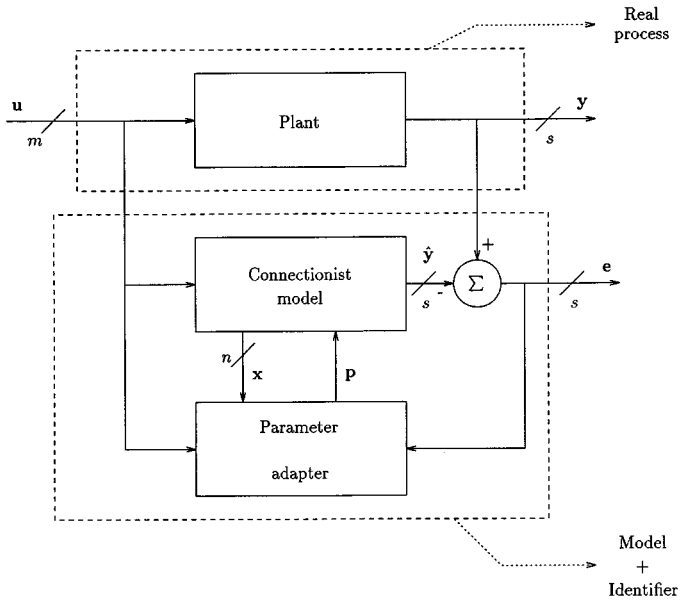

Fig. 3. Structure of the identification method.

A. Gradient Parameter Adaptation Based on Sensitivity Analysis

1) Statement of the Problem: Taking the output of the real process $\boldsymbol{y}$ and the output from the model $\hat{\boldsymbol{y}}$, the identification error vector is defined as $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$. This error is a function of time and the model parameters

$$\boldsymbol{e} = \boldsymbol{e}(t, \boldsymbol{\alpha}, \boldsymbol{W}, \boldsymbol{B}) \in \mathbb{R}^s \quad (6)$$

and is zero when the model represents exactly the dynamic behavior of the system to be identified. Therefore, the parametric identification problem involves finding a parameter set such that $\hat{\boldsymbol{y}} = \boldsymbol{y}$ and, in order to achieve this, the parameter data set must

be modified recursively to bring the identification error to zero or to a small residual value which may be attributed to the noise inherent to the real process.

For this purpose, it is necessary to define a cost functional of the identification error which measures the goodness of fit of the identification mechanism and which takes its minimum value for a zero identification error. The selected cost functional is

$$V = \tfrac{1}{2}\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} \tag{7}$$

where $\boldsymbol{Q}$ is a symmetric positive definite matrix that weighs the components of the identification error vector.[3]

2) Parameter Updating Equations: As for the parameter adaptation mechanism, a common method is to perform the parameter modification in the opposite direction of the cost function gradient vector with respect to the model parameters. Specifically, for the proposed model, the parameter update equations are

$$\dot{\boldsymbol{\alpha}} = -\epsilon_{\boldsymbol{\alpha}}\frac{\partial V}{\partial \boldsymbol{\alpha}}, \qquad \boldsymbol{\alpha}(0) = \boldsymbol{\alpha}^0 \tag{8}$$

$$\dot{\boldsymbol{W}} = -\epsilon_{\boldsymbol{W}}\frac{\partial V}{\partial \boldsymbol{W}}, \quad \boldsymbol{W}(0) = \boldsymbol{W}^0 \tag{9}$$

$$\dot{\boldsymbol{B}} = -\epsilon_{\boldsymbol{B}}\frac{\partial V}{\partial \boldsymbol{B}}, \qquad \boldsymbol{B}(0) = \boldsymbol{B}^0 \tag{10}$$

where the constants $\epsilon.$ are the learning rates for each parameter set.

Equations (8)–(10) express in a general form the update of the model parameters, but they require some developments to become useful. In particular, it is necessary to expand the derivatives of the cost function V with respect to all the model parameter sets using the extended chain rule for matrix operations.

For illustration, the derivative of the cost functional with respect to vector $\boldsymbol{\alpha}$, $\partial V/\partial \boldsymbol{\alpha}$ is derived below. Table I summarizes the results for the $\boldsymbol{\alpha}$, $\boldsymbol{W}$, and $\boldsymbol{B}$ parameters

$$\frac{\partial V}{\partial \boldsymbol{\alpha}}\left(\boldsymbol{e}(\boldsymbol{\alpha})\right) = \left(\frac{\partial \boldsymbol{e}^T}{\partial \boldsymbol{\alpha}} \otimes \boldsymbol{I}_1\right)\left(\boldsymbol{I}_1 \otimes \frac{\partial V}{\partial \boldsymbol{e}}\right) = \frac{\partial \boldsymbol{e}^T}{\partial \boldsymbol{\alpha}}\frac{\partial V}{\partial \boldsymbol{e}} \tag{11}$$

where the second term in the right-hand side is

$$\frac{\partial V}{\partial \boldsymbol{e}} = \frac{1}{2}\left(\boldsymbol{Q} + \boldsymbol{Q}^T\right)\boldsymbol{e} = \boldsymbol{Q}\boldsymbol{e} \tag{12}$$

according to the definition of the cost function V in (7) and taking into account the symmetry of $\boldsymbol{Q}$. In the first term in the right-hand side of (11), it is necessary to apply the chain rule once more to obtain

$$\begin{aligned}\frac{\partial \boldsymbol{e}^T}{\partial \boldsymbol{\alpha}}\left(\hat{\boldsymbol{y}}^T(\boldsymbol{\alpha})\right) &= \left(\frac{\partial \hat{\boldsymbol{y}}^T}{\partial \boldsymbol{\alpha}} \otimes \boldsymbol{I}_1\right)\left(\boldsymbol{I}_1 \otimes \frac{\partial \boldsymbol{e}^T}{\partial \hat{\boldsymbol{y}}}\right) \\ &= \frac{\partial \hat{\boldsymbol{y}}^T}{\partial \boldsymbol{\alpha}}\frac{\partial \boldsymbol{e}^T}{\partial \hat{\boldsymbol{y}}}\end{aligned} \tag{13}$$

which, according to the previous definition of the error vector $\partial \boldsymbol{e}^T/\partial \hat{\boldsymbol{y}} = (\partial \boldsymbol{e}/\partial \hat{\boldsymbol{y}}^T)^T = -\boldsymbol{I}_s$, results in

$$\begin{aligned}\dot{\boldsymbol{\alpha}} &= -\epsilon_{\boldsymbol{\alpha}}\frac{\partial V}{\partial \boldsymbol{\alpha}} \\ &= -\epsilon_{\boldsymbol{\alpha}}\frac{\partial \boldsymbol{e}^T}{\partial \boldsymbol{\alpha}}\frac{\partial V}{\partial \boldsymbol{e}} \\ &= -\epsilon_{\boldsymbol{\alpha}}\frac{\partial \hat{\boldsymbol{y}}^T}{\partial \boldsymbol{\alpha}}\frac{\partial \boldsymbol{e}^T}{\partial \hat{\boldsymbol{y}}}\frac{\partial V}{\partial \boldsymbol{e}} \\ &= -\epsilon_{\boldsymbol{\alpha}}\left(-\frac{\partial \hat{\boldsymbol{y}}^T}{\partial \boldsymbol{\alpha}}\boldsymbol{I}_s\boldsymbol{Q}\boldsymbol{e}\right) \\ &= \epsilon_{\boldsymbol{\alpha}}\frac{\partial \hat{\boldsymbol{y}}^T}{\partial \boldsymbol{\alpha}}\boldsymbol{Q}\boldsymbol{e} \\ &= \epsilon_{\boldsymbol{\alpha}}\left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\alpha}^T}\right)^T\boldsymbol{Q}\boldsymbol{e}\end{aligned} \tag{14}$$

where $\partial \hat{\boldsymbol{y}}^T/\partial \boldsymbol{\alpha}$ is the transpose Jacobian matrix of the output trajectories of the model versus the parameters, $\partial \hat{\boldsymbol{y}}/\partial \boldsymbol{\alpha}^T$. This term cannot be expanded further using the chain rule and it is therefore necessary to rely on sensitivity analysis in order to compute it. The application of this technique to the ADC model will be carried out in the following section for the series of parameter sets in the model.

For illustration, if the derivative of the cost functional with respect to matrix $\boldsymbol{W}$, $\partial V/\partial \boldsymbol{W}$ is expanded and arranged in matrix form

$$\begin{aligned}\frac{\partial V}{\partial \boldsymbol{W}}\left(\boldsymbol{e}(\boldsymbol{W})\right) &= \left(\boldsymbol{I}_n \otimes \frac{\partial V}{\partial \boldsymbol{e}^T}\right)\left(\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{W}} \otimes \boldsymbol{I}_1\right) \\ &= \left(\boldsymbol{I}_n \otimes \frac{\partial V}{\partial \boldsymbol{e}^T}\right)\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{W}}\end{aligned} \tag{15}$$

where the term in the rightmost factor is, by the chain rule

$$\begin{aligned}\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{W}}\left(\hat{\boldsymbol{y}}(\boldsymbol{W})\right) &= \left(\boldsymbol{I}_n \otimes \frac{\partial \boldsymbol{e}}{\partial \hat{\boldsymbol{y}}^T}\right)\left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}} \otimes \boldsymbol{I}_1\right) \\ &= \left(\boldsymbol{I}_n \otimes \frac{\partial \boldsymbol{e}}{\partial \hat{\boldsymbol{y}}^T}\right)\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}}.\end{aligned} \tag{16}$$

Substituting (16) in (15) gives

$$\begin{aligned}\frac{\partial V}{\partial \boldsymbol{W}} &= \left(\boldsymbol{I}_n \otimes \frac{\partial V}{\partial \boldsymbol{e}^T}\right)\left(\boldsymbol{I}_n \otimes \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{y}^T}\right)\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}} \\ &= \left(\boldsymbol{I}_n \otimes \boldsymbol{e}^T\boldsymbol{Q}\right)\left(\boldsymbol{I}_n \otimes (-\boldsymbol{I}_s)\right)\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}} \\ &= -\left[(\boldsymbol{I}_n\boldsymbol{I}_n) \otimes \left(\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{I}_s\right)\right]\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}} \\ &= -\left(\boldsymbol{I}_n \otimes \left(\boldsymbol{e}^T\boldsymbol{Q}\right)\right)\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}}.\end{aligned} \tag{17}$$

Consequently, the update equation of the parameter set $\boldsymbol{W}$ is

$$\dot{\boldsymbol{W}} = \epsilon_{\boldsymbol{W}}\left(\boldsymbol{I}_n \otimes \left(\boldsymbol{e}^T\boldsymbol{Q}\right)\right)\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}} \tag{18}$$

where the term $\partial \hat{\boldsymbol{y}}/\partial \boldsymbol{W}$ cannot be expanded further by the chain rule, requiring the application of sensitivity analysis techniques for dynamic systems.

As has been shown, the parameter-update equations contain the constants $\epsilon.$ that clearly modify their dynamic behavior, specifically the learning or adaptation rate. In general practice, these constants are given positive values close to zero to obtain a slow dynamic behavior for the parameters. However, a reasoned justification of this fact appears in the following section and arises naturally from the development of the sensitivity equations of the model.

3) Sensitivity Analysis of the Connectionist Model: The basic mathematical problem in sensitivity theory is the compu-

[3]A special case could be $\boldsymbol{Q} = \boldsymbol{I}$.

TABLE I
UPDATE EQUATIONS FOR THE PARAMETER SETS OF THE CONNECTIONIST MODEL

Parameter	Update equation
$\boldsymbol{\alpha}$	$\dot{\boldsymbol{\alpha}} = \epsilon_{\boldsymbol{\alpha}} \left(\frac{\partial \hat{\mathbf{y}}}{\partial \boldsymbol{\alpha}^T}\right)^T \mathbf{Q}\mathbf{e}, \quad \boldsymbol{\alpha}(0) = \boldsymbol{\alpha}^0$
$\mathbf{W}$	$\dot{\mathbf{W}} = \epsilon_{\mathbf{W}} \left(\mathbf{I}_n \otimes \left(\mathbf{e}^T\mathbf{Q}\right)\right) \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{W}}, \quad \mathbf{W}(0) = \mathbf{W}^0$
$\mathbf{B}$	$\dot{\mathbf{B}} = \epsilon_{\mathbf{B}} \left(\mathbf{I}_n \otimes \left(\mathbf{e}^T\mathbf{Q}\right)\right) \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{B}}, \quad \mathbf{B}(0) = \mathbf{B}^0$

tation of the change in the system behavior due to parameter variations. In particular, in this work the following definition taken from [35] is used.

Definition III.1: The absolute sensitivity function is

$$\boldsymbol{S}_j \triangleq \left.\frac{\partial \boldsymbol{\zeta}(\boldsymbol{\alpha})}{\partial \alpha_j}\right|_{\boldsymbol{\alpha}_0} \tag{19}$$

where

$\boldsymbol{\zeta}(\boldsymbol{\alpha})$ any function that characterizes the system behavior;
$\boldsymbol{\alpha}$ system parameter vector;
$\boldsymbol{\alpha}_0$ nominal value. □.

According to the previous definition, the sensitivity to parameter variations is a function, not a coefficient, and is time dependent. Also, as can be seen in the definition, the parameter variations are treated in an infinitesimal manner, leading to the representation of the sensitivity function as a partial derivative.

There are qualitative differences in the effects produced by variations of the different parameters. The following classification proposed in [35] can be established.

Definition III.2: There are three categories of parameter variations in continuous-time dynamic systems.

α-variations: These are parameter variations around a nominal value $\boldsymbol{\alpha}_0$ that do not affect the order of the mathematical model. A necessary condition is that $\boldsymbol{\alpha}_0 \neq \mathbf{0}$.
β-variations: These are variations of the initial conditions from their nominal values $\boldsymbol{\beta}_0$.
λ-variations: These are parameter variations from a nominal value $\boldsymbol{\lambda}_0 = \mathbf{0}$ that affect the order of the mathematical model. □

In this work, the parameter variations that occur in the identification model are of the α type since the following assumptions are made.

- The initial conditions of the identification models do not depend on the parameters of the model.
- The initial conditions of the models are not known and the parameter fit must be as insensitive as possible to their values.
- None of the model parameters are strictly zero, but should they be, the order of the mathematical model would not be affected.[4]
- The inputs to the real system and the model do not depend on the system parameters. This can be assumed because the identification of the real system is made in open loop and, for this reason, the system inputs are independent of its dynamic behavior.

In order to find the trajectory sensitivity equations of the connectionist model of (4) and (5) with respect to its parameters, it is necessary to take the partial derivative with respect to the parameters in the state and output equations of the model. For example, the expansions for vector $\boldsymbol{\alpha}$ and matrix $\boldsymbol{B}$ are performed.

Starting with vector $\boldsymbol{\alpha}$, if the partial derivative with respect to this vector is taken on both sides of (4) and the chain rule is applied

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\alpha}^T}\dot{\boldsymbol{x}} &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\alpha}^T} + \left(\boldsymbol{I}_1 \otimes \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T}\right)\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}^T} \otimes \boldsymbol{I}_1\right) \\ &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}^T} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\alpha}^T} \\ \frac{\partial \boldsymbol{x}(0)}{\partial \boldsymbol{\alpha}^T} &= \mathbf{0}.\end{aligned} \tag{20}$$

It is important to remark that the derivative of the initial conditions with respect to the parameter vector is equal to zero since they have been assumed independent of the parameters. Now, if the parameter vector $\boldsymbol{\alpha}$ is kept constant in time ($\boldsymbol{\alpha}_0$), a swap of the operator derivative with respect to time and derivative with respect to the parameter vector can be performed as follows:

$$\dot{\boldsymbol{\Lambda}}_{\boldsymbol{\alpha}} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T}\right|_{\boldsymbol{\alpha}_0} \boldsymbol{\Lambda}_{\boldsymbol{\alpha}} + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\alpha}^T}\right|_{\boldsymbol{\alpha}_0}, \qquad \boldsymbol{\Lambda}_{\boldsymbol{\alpha}}(0) = \mathbf{0} \tag{21}$$

where

$$\boldsymbol{\Lambda}_{\boldsymbol{\alpha}} \triangleq \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}^T} = \left[\frac{\partial x_i}{\partial \alpha_j}\right] \in M_n(\mathbb{R}).$$

Taking partial derivatives in the output equation (5)

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\alpha}^T} &= \left(\boldsymbol{I}_1 \otimes \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}^T}\right)\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}^T} \otimes \boldsymbol{I}_1\right) \\ &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}^T}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}^T}\end{aligned} \tag{22}$$

and putting it in matrix form with

$$\boldsymbol{\Xi}_{\boldsymbol{\alpha}} \triangleq \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\alpha}^T} = \left[\frac{\partial \hat{y}_i}{\partial \alpha_j}\right] \in M_{s,n}(\mathbb{R})$$

gives

$$\boldsymbol{\Xi}_{\boldsymbol{\alpha}} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}^T}\right|_{\boldsymbol{\alpha}_0} \boldsymbol{\Lambda}_{\boldsymbol{\alpha}}. \tag{23}$$

For the parameter matrix $\boldsymbol{B} \in M_{n,m}(\mathbb{R})$, the sensitivity state and output equations take the following form:

$$\frac{\partial}{\partial \boldsymbol{B}}\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{B}} + \left(\boldsymbol{I}_n \otimes \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T}\right)\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{B}} \otimes \boldsymbol{I}_1\right)$$

[4]This fact can be observed in the equations of the model given the chosen structure.

$$\frac{\partial \boldsymbol{x}(0)}{\partial \boldsymbol{B}} = \boldsymbol{0} \tag{24}$$

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{B}} = \left(\boldsymbol{I}_n \otimes \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}^T}\right)\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{B}}. \tag{25}$$

Restating the last two equations in matrix form with $\boldsymbol{\Lambda_B} \triangleq \partial \hat{\boldsymbol{x}}/\partial \boldsymbol{B}^T \in M_{n^2,m}(\mathbb{R})$ and $\boldsymbol{\Xi_B} \triangleq \partial \hat{\boldsymbol{y}}/\partial \boldsymbol{B}^T \in M_{sn,m}(\mathbb{R})$ and taking into account the operator swap, the following equations are obtained:

$$\dot{\boldsymbol{\Lambda}}_{\boldsymbol{B}} = \left(\boldsymbol{I}_n \otimes \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T}\right|_{\boldsymbol{B}_0}\right)\boldsymbol{\Lambda_B} + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{B}_0}$$

$$\boldsymbol{\Lambda}(0)_{\boldsymbol{B}} = \boldsymbol{0} \tag{26}$$

$$\boldsymbol{\Xi_B} = \left(\boldsymbol{I}_n \otimes \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}^T}\right|_{\boldsymbol{B}_0}\right)\boldsymbol{\Lambda_B}. \tag{27}$$

4) Identification Process Equations: Arranging the differential and algebraic equations that correspond to the model, the sensitivity analysis and the parameter update together produce the following set of equations:

$$\dot{\boldsymbol{x}} = -\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{x}) + \boldsymbol{W}\boldsymbol{\sigma}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{d} \tag{28}$$

$$\dot{\boldsymbol{\Lambda}}_{\boldsymbol{\alpha}} = \left(-\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{I}_n) + \boldsymbol{W}\frac{\partial \boldsymbol{\sigma}(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\right)\boldsymbol{\Lambda_\alpha} + (-\boldsymbol{U}(\boldsymbol{I}_n \otimes \boldsymbol{x})) \tag{29}$$

$$\dot{\boldsymbol{\Lambda}}_{\boldsymbol{W}} = \left(\boldsymbol{I}_n \otimes \left(-\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{I}_n) + \boldsymbol{W}\frac{\partial \boldsymbol{\sigma}(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\right)\right)\boldsymbol{\Lambda_W} + \overline{\boldsymbol{U}}_{nn}(\boldsymbol{I}_n \otimes \boldsymbol{\sigma}(\boldsymbol{x})) \tag{30}$$

$$\dot{\boldsymbol{\Lambda}}_{\boldsymbol{B}} = \left(\boldsymbol{I}_n \otimes \left(-\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{I}_n) + \boldsymbol{W}\frac{\partial \boldsymbol{\sigma}(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\right)\right)\boldsymbol{\Lambda_B} + \overline{\boldsymbol{U}}_{nm}(\boldsymbol{I}_m \otimes \boldsymbol{u}) \tag{31}$$

$$\dot{\boldsymbol{\alpha}} = \epsilon_{\boldsymbol{\alpha}}\boldsymbol{\Xi}_{\boldsymbol{\alpha}}^T\boldsymbol{Q}\boldsymbol{e} \tag{32}$$

$$\dot{\boldsymbol{W}} = \epsilon_{\boldsymbol{W}}\left(\boldsymbol{I}_n \otimes \left(\boldsymbol{e}^T\boldsymbol{Q}\right)\right)\boldsymbol{\Xi_W} \tag{33}$$

$$\dot{\boldsymbol{B}} = \epsilon_{\boldsymbol{B}}\left(\boldsymbol{I}_n \otimes \left(\boldsymbol{e}^T\boldsymbol{Q}\right)\right)\boldsymbol{\Xi_B} \tag{34}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\boldsymbol{x} \tag{35}$$

$$\boldsymbol{\Xi}_{\boldsymbol{\alpha}} = \boldsymbol{C}\boldsymbol{\Lambda}_{\boldsymbol{\alpha}} \tag{36}$$

$$\boldsymbol{\Xi_W} = (\boldsymbol{I}_n \otimes \boldsymbol{C})\boldsymbol{\Lambda_W} \tag{37}$$

$$\boldsymbol{\Xi_B} = (\boldsymbol{I}_n \otimes \boldsymbol{C})\boldsymbol{\Lambda_B} \tag{38}$$

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} \tag{39}$$

where $\overline{\boldsymbol{U}}_{kl} \triangleq \sum_{i=1}^{k}\sum_{j=1}^{l}\boldsymbol{E}_{ij} \otimes \boldsymbol{E}_{ij} \in M_{k^2,l^2}(\{0,1\})$ and $\boldsymbol{E}_{ij} = \boldsymbol{e}_i\boldsymbol{e}_j^T \in M_{k,l}(\{0,1\})$ is the Kronecker matrix, $\boldsymbol{e}_i$ being the ith vector of the natural basis of $\mathbb{R}^k$, $\boldsymbol{e}_j$ being the jth vector of the natural basis of $\mathbb{R}^l$, and

$$\frac{\partial \boldsymbol{\sigma}(\boldsymbol{x})}{\partial \boldsymbol{x}^T} = \operatorname{diag}\left\{\frac{d\sigma(x_1)}{dx_1}, \cdots, \frac{d\sigma(x_n)}{dx_n}\right\}$$

with $d\sigma(z)/dz = 1 - (\tanh(z))^2$ given the definition of $\sigma(\cdot)$.

Of the set of (28)–(39), (28) is the state equation of the connectionist model, the differential matrix equations (29)–(31) are the state sensitivity equations with respect to the model parameters, the differential matrix equations (32)–(34) are the parameter update equations according to the gradient method and, finally, the algebraic equations (35)–(39) correspond to the output equation of the model, the output sensitivity equations, and the identification error of the method.

Equations (28)–(39) make up, together with the initial conditions

$$\begin{aligned}&\boldsymbol{x}(0) = \boldsymbol{x}^0 \quad \boldsymbol{\Lambda_\alpha}(0) = \boldsymbol{0} \quad \boldsymbol{\Lambda_W}(0) = \boldsymbol{0} \quad \boldsymbol{\Lambda_B}(0) = \boldsymbol{0}\\ &\boldsymbol{\alpha}(0) = \boldsymbol{\alpha}^0 \quad \boldsymbol{W}(0) = \boldsymbol{W}^0 \quad \boldsymbol{B}(0) = \boldsymbol{B}^0\end{aligned} \tag{40}$$

an initial value problem which can be solved in an on-line manner. Then, the learning of the model parameters can be performed from a random-value set in real-time operation if necessary.

At this point, it is important to examine the following considerations about the adaptation mechanism formulated above. First, the impact of the value of constants $\epsilon.$ that appear in the parameter update equations and, second, the effect of the initial conditions of the state equation of the model and the parameter update equations in the global behavior of the adaptation process.

As for the impact of the constants $\epsilon.$ (learning rates), in the global dynamics it is important to remark that the sensitivity equations of the model only give the Jacobian matrix of the output trajectories of the model with respect to the parameters when these are kept constant during the whole process. Actually, this fact is not so in an on-line parameter adaptation since the parameters are changing continuously, thus, only an approximation of the expected result is obtained. Then, if a correct operation of the adaptation mechanism is desired, it is necessary to assign to the constants $\epsilon.$ a positive small value in order to provide a slow dynamics for the parameters for the purpose of considering them quasi-stationary.

The following comments can be made concerning the initial conditions of the equations involved in the adaptation process.

- The sensitivity state equations are given, as mentioned above, zero initial conditions.
- The state equation of the model has unknown initial conditions. Likewise, as the chosen approach is of the black-box type, the knowledge of the initial conditions of the physical system under study is not relevant. In general, a random-valued or zero vector of initial conditions can be taken.
- The parameter update equations also need a set of initial conditions which must be nonzero so as not to reduce the model to the trivial case. Obviously, the closer the initial conditions are to the optimal values, the faster the adaptation process will be.[5]

5) Complexity Issues: The complexity, in number of differential equations, of this on-line adaptation mechanism is the following.

- Necessary equations for the adaptation of the parameter vector $\boldsymbol{\alpha}$: $n_{\boldsymbol{\alpha}} = n^2 + n$.
- Necessary equations for the adaptation of the weight matrix $\boldsymbol{W}$: $n_{\boldsymbol{W}} = n^3 + n^2$.
- Necessary equations for the adaptation of the input matrix $\boldsymbol{B}$: $n_{\boldsymbol{B}} = n^2m + nm$.

[5]In the error and cost-function sense.

- Equations of the additive dynamic neural model: $n_{\text{mod}} = n$.

Therefore, the total complexity is $n_{\text{total}} = n^3 + (2+m)n^2 + (2+m)n$ where n is the model state-vector dimension (number of nodes in the neural network) and m is the number of inputs. For example, in the usual case which has a constant vector $\boldsymbol{\alpha}$ and input matrix $\boldsymbol{B}$, the complexity is $n_{\text{total}} = n^3 + n^2 + n$.

B. Variational and Invariant Imbedding Techniques

1) Statement of the Problem: As given in Section III-A1, the learning error is defined as $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$. The chosen functional is the integral of a quadratic form of the learning error, i.e.,

$$J = \int_0^{t_f} \boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{e}\, dt \tag{41}$$

and it must be minimized subject to the following constraints:

- the dynamics of the neural network: (4) and (5);
- the stationarity of the net parameters: $\dot{\boldsymbol{\alpha}} = \boldsymbol{0}$, $\dot{\boldsymbol{W}} = \boldsymbol{0}$, $\dot{\boldsymbol{B}} = \boldsymbol{0}$.

2) A Variational Solution of the Learning Problem: This section develops a variational solution as in [36] to the minimization problem stated above. First, the constraints are adjoined to the cost functional J with the corresponding multiplier functions $\boldsymbol{\lambda}$.

$$\begin{aligned} J = \int_0^{t_f} \Big\{ & \boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{e} + \boldsymbol{\lambda}_{\boldsymbol{x}}^T (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{W}, \boldsymbol{B}) - \dot{\boldsymbol{x}}) + \boldsymbol{\lambda}_{\boldsymbol{\alpha}}^T (-\dot{\boldsymbol{\alpha}}) \\ & + \boldsymbol{\lambda}_{\boldsymbol{W}}^T \left(-\operatorname{col} \dot{\boldsymbol{W}}^T \right) + \boldsymbol{\lambda}_{\boldsymbol{B}}^T \left(-\operatorname{col} \dot{\boldsymbol{B}}^T \right) \Big\} dt. \end{aligned} \tag{42}$$

The right side[6] of the above equation is integrated by parts and then the variation of J due to variations in the parameters $\{\boldsymbol{\alpha}, \boldsymbol{W}, \boldsymbol{B}\}$ of the model with a fixed final time t_f is

$$\begin{aligned} \delta J = & - \Big\{ \left[\boldsymbol{\lambda}_{\boldsymbol{x}}^T \delta \boldsymbol{x} \right]_0^{t_f} + \left[\boldsymbol{\lambda}_{\boldsymbol{\alpha}}^T \delta \boldsymbol{\alpha} \right]_0^{t_f} + \left[\boldsymbol{\lambda}_{\boldsymbol{W}}^T \delta \operatorname{col} \dot{\boldsymbol{W}}^T \right]_0^{t_f} \\ & + \left[\boldsymbol{\lambda}_{\boldsymbol{B}}^T \delta \operatorname{col} \dot{\boldsymbol{B}}^T \right]_0^{t_f} \Big\} \\ & + \int_0^{t_f} \Big\{ \left(-\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{C} + \dot{\boldsymbol{\lambda}}_{\boldsymbol{x}}^T + \boldsymbol{\lambda}_{\boldsymbol{x}}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T} \right) \delta \boldsymbol{x} \\ & + \left(\boldsymbol{\lambda}_{\boldsymbol{x}}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\alpha}^T} + \dot{\boldsymbol{\lambda}}_{\boldsymbol{\alpha}}^T \right) \delta \boldsymbol{\alpha} + \left(\boldsymbol{\lambda}_{\boldsymbol{x}}^T \frac{\partial \boldsymbol{f}}{\partial \operatorname{row} \boldsymbol{W}} + \dot{\boldsymbol{\lambda}}_{\boldsymbol{W}}^T \right) \\ & \cdot \delta \operatorname{col} \boldsymbol{W}^T + \left(\boldsymbol{\lambda}_{\boldsymbol{x}}^T \frac{\partial \boldsymbol{f}}{\partial \operatorname{row} \boldsymbol{B}} + \dot{\boldsymbol{\lambda}}_{\boldsymbol{B}}^T \right) \delta \operatorname{col} \boldsymbol{B}^T \Big\} dt \end{aligned} \tag{43}$$

where the stationarity of the parameters and the dynamic equations of the neural model are assumed in the simplification. Now, if the variation in J due to variations $\delta \boldsymbol{x}$, $\delta \boldsymbol{\alpha}$, $\delta \operatorname{col} \boldsymbol{W}^T$, and $\delta \operatorname{col} \boldsymbol{B}^T$ must be zero, it is necessary that the terms in brackets and parentheses in (43) vanish. Taking into account these conditions gives

$$\dot{\boldsymbol{x}} = -\boldsymbol{U}(\boldsymbol{\alpha} \otimes \boldsymbol{x}) + \boldsymbol{W} \boldsymbol{\sigma}(\boldsymbol{x}) + \boldsymbol{B} \boldsymbol{u} + \boldsymbol{d} \tag{44}$$

$$\dot{\boldsymbol{\alpha}} = \boldsymbol{0} \tag{45}$$

$$\operatorname{col} \dot{\boldsymbol{W}}^T = \boldsymbol{0} \tag{46}$$

$$\operatorname{col} \dot{\boldsymbol{B}}^T = \boldsymbol{0} \tag{47}$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{x}} = \operatorname{diag}\{\alpha_1, \cdots, \alpha_n\} \boldsymbol{\lambda}_{\boldsymbol{x}} - \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}^T} \boldsymbol{W}^T \boldsymbol{\lambda}_{\boldsymbol{x}} + \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{e} \tag{48}$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{\alpha}} = \operatorname{diag}\{x_1, \cdots, x_n\} \boldsymbol{\lambda}_{\boldsymbol{x}} \tag{49}$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{W}} = -(\boldsymbol{I}_n \otimes \boldsymbol{\sigma}(\boldsymbol{x})) \boldsymbol{\lambda}_{\boldsymbol{x}} \tag{50}$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{B}} = -(\boldsymbol{I}_n \otimes \boldsymbol{u}) \boldsymbol{\lambda}_{\boldsymbol{x}} \tag{51}$$

where (44)-(47) result from the stationarity of the parameters and the dynamic neural model and (48)-(51) follow from the need to nullify the terms in parentheses inside the integral of (43). It is also important to state the boundary conditions, extracted from the transversality conditions, which take the following form:

$$\begin{aligned} \boldsymbol{\lambda}_{\boldsymbol{x}}(0) = \boldsymbol{\lambda}_{\boldsymbol{x}}(t_f) = \boldsymbol{0} \qquad & \boldsymbol{\lambda}_{\boldsymbol{\alpha}}(0) = \boldsymbol{\lambda}_{\boldsymbol{\alpha}}(t_f) = \boldsymbol{0} \\ \boldsymbol{\lambda}_{\boldsymbol{W}}(0) = \boldsymbol{\lambda}_{\boldsymbol{W}}(t_f) = \boldsymbol{0} \qquad & \boldsymbol{\lambda}_{\boldsymbol{B}}(0) = \boldsymbol{\lambda}_{\boldsymbol{B}}(t_f) = \boldsymbol{0}. \end{aligned} \tag{52}$$

Arranging the state vectors of the differential equations of the boundary value problem as

$$\boldsymbol{z} = \left[\boldsymbol{x}^T \quad \boldsymbol{\alpha}^T \quad \left(\operatorname{col} \boldsymbol{W}^T \right)^T \quad \left(\operatorname{col} \boldsymbol{B}^T \right)^T \right]^T \in \mathbb{R}^{n^2+2n+nm}$$

and

$$\boldsymbol{\zeta} = \left[\boldsymbol{\lambda}_{\boldsymbol{x}}^T \quad \boldsymbol{\lambda}_{\boldsymbol{\alpha}}^T \quad \boldsymbol{\lambda}_{\boldsymbol{W}}^T \quad \boldsymbol{\lambda}_{\boldsymbol{B}}^T \right]^T \in \mathbb{R}^{n^2+2n+nm}$$

the system (44)–(51) can be stated in compact form as

$$\dot{\boldsymbol{z}} = \boldsymbol{g}(\boldsymbol{z}, \boldsymbol{u}) \tag{53}$$

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{h}(\boldsymbol{\zeta}, \boldsymbol{z}, \boldsymbol{u}), \qquad \boldsymbol{\zeta}(0) = \boldsymbol{\zeta}(t_f) = \boldsymbol{0} \tag{54}$$

which constitutes a two-point boundary value problem (TPBVP) which cannot be solved in a on-line manner because of the fixed end time and the boundary conditions.

3) On-Line Operation Using Invariant Imbedding Techniques: In order to solve the learning problem on line with an infinite time horizon, an invariant imbedding (II) technique [37], [38] may be used. The II methodology is based on the transformation of the problem into a more general one that has an easier solution. Then, when the more general problem is solved, the previous problem is automatically solved.

When this approach is applied to the above TPVBP the following partial differential equation results [39]:

$$\boldsymbol{g}(\boldsymbol{r}(\boldsymbol{c}, t_f), \boldsymbol{u}) = \left. \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{c}^T} \right|_{\boldsymbol{c}} \boldsymbol{h}(\boldsymbol{c}, \boldsymbol{r}(\boldsymbol{c}, t_f), \boldsymbol{u}) + \left. \frac{\partial \boldsymbol{r}}{\partial t_f} \right|_{t_f} \tag{55}$$

where

- $\boldsymbol{c}$ general value for the end condition of $\boldsymbol{\zeta}$ ($\boldsymbol{\zeta}(t_f) = \boldsymbol{c}$);
- $\boldsymbol{r}$ function that relates the value of $\boldsymbol{\zeta}(t_f)$ with $\boldsymbol{z}(t_f)$ ($\boldsymbol{z}(t_f) = \boldsymbol{r}(\boldsymbol{c}, t_f)$).

This is the partial differential equation of the invariant imbedding technique and it does not have a known general solution. However, the solution can be approximated through the following linear function:

$$\boldsymbol{r}(\boldsymbol{c}, t_f) = \hat{\boldsymbol{z}}(t_f) + \boldsymbol{P}(t_f) \boldsymbol{c} \tag{56}$$

[6]The column operator **col** applied to a matrix $\boldsymbol{A} \in M_{n,m}(\mathbb{R})$ yields a vector that has as components the elements of $\boldsymbol{A}$ stacked by columns. Formally, $\operatorname{col} \boldsymbol{A} = \sum_{j=1}^{m} (\boldsymbol{e}_j \otimes \boldsymbol{I}_n) \boldsymbol{A} \boldsymbol{e}_j \in \mathbb{R}^{nm}$, where $\boldsymbol{e}_j$ is the jth vector of the standard basis of $\mathbb{R}^m$.
Likewise, the row operator is defined as $(\operatorname{row} \boldsymbol{A})^T = \operatorname{col} \boldsymbol{A}^T \in \mathbb{R}^{nm}$.

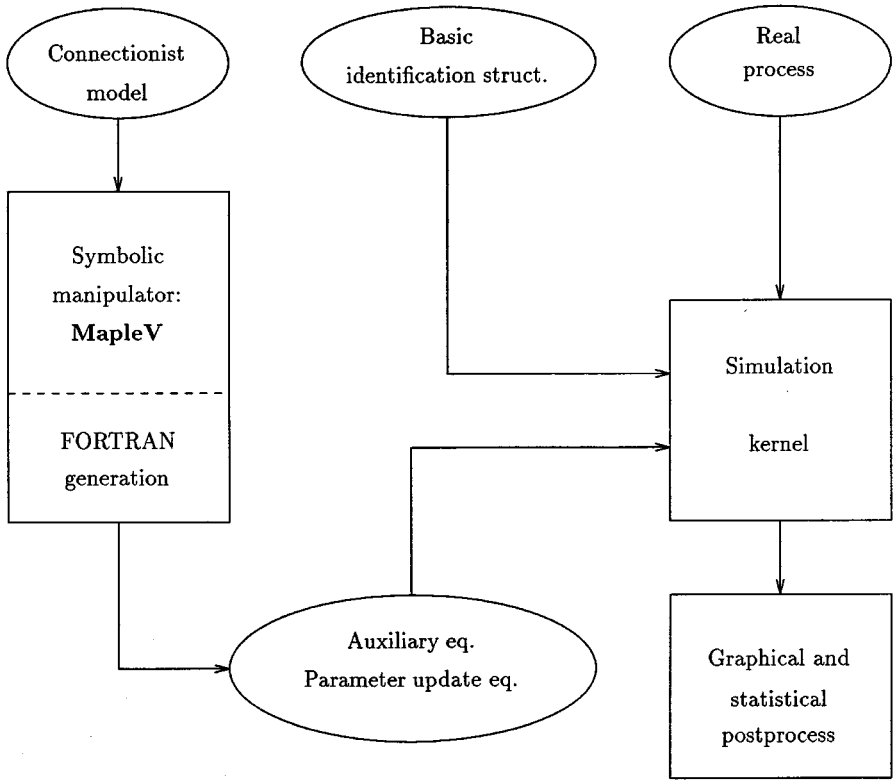

Fig. 4. Sketch of the generation, implementation, and validation of the elements involved in the identification process.

where $\hat{\boldsymbol{z}}(t_f)$ is the correct solution of the problem, i.e., the solution when $\boldsymbol{c} = \boldsymbol{0} \equiv \boldsymbol{\zeta}(t_f) = \boldsymbol{0}$. It is important to point out that the linear structure chosen for the function $\boldsymbol{z}$ is appropriate while $\boldsymbol{c}$ is small ($\boldsymbol{c} \approx \boldsymbol{0}$), i.e., near the optimal solution. For this reason, in the following derivations, it will be assumed that $\boldsymbol{c}$ is small and, therefore, the terms of $\mathcal{O}(\|\boldsymbol{c}\|^2)$ and higher will not be taken into account. Now, if (56) is substituted in (55)

$$\boldsymbol{g}(\hat{\boldsymbol{z}} + \boldsymbol{P}\boldsymbol{c}, \boldsymbol{u}) = \boldsymbol{P}\boldsymbol{h}(\boldsymbol{c}, \hat{\boldsymbol{z}} + \boldsymbol{P}\boldsymbol{c}, \boldsymbol{u}) + \dot{\hat{\boldsymbol{z}}} + \dot{\boldsymbol{P}}\boldsymbol{c}. \tag{57}$$

However, since the functions $\boldsymbol{g}$ and $\boldsymbol{h}$ are nonlinear, it is necessary to perform a Taylor expansion around $\boldsymbol{\zeta} = \boldsymbol{c}$, $t = t_f$, and $\boldsymbol{z} = \hat{\boldsymbol{z}}$ until the first order obtaining:

$$\begin{aligned} &\boldsymbol{g}(\hat{\boldsymbol{z}}, \boldsymbol{u}) + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}^T}(\hat{\boldsymbol{z}}, \boldsymbol{u})\boldsymbol{P}\boldsymbol{c} \\ &\quad = \boldsymbol{P}\boldsymbol{h}(\boldsymbol{c}, \hat{\boldsymbol{z}}, \boldsymbol{u}) + \boldsymbol{P}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}^T}(\boldsymbol{c}, \hat{\boldsymbol{z}}, \boldsymbol{u})\boldsymbol{P}\boldsymbol{c} + \dot{\hat{\boldsymbol{z}}} + \dot{\boldsymbol{P}}\boldsymbol{c}. \end{aligned} \tag{58}$$

The derivation continues by substituting functions $\boldsymbol{g}$ and $\boldsymbol{h}$ by their expressions and removing the terms of $\mathcal{O}(\|\boldsymbol{c}\|^2)$ and higher. Then, the terms of degrees zero and one in $\boldsymbol{c}$ are equated separately.

4) Application to the Dynamic Additive Models: The results obtained in the previous section can be applied to the TPBVP equations (53) and (54) of the neural network identification problem. However, beforehand, it is necessary to recast the equations into a more explicit form as

$$\dot{\boldsymbol{z}} = \boldsymbol{g}(\boldsymbol{z}, \boldsymbol{u}) \tag{59}$$

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{H}(\boldsymbol{\zeta}, \boldsymbol{z}, \boldsymbol{u})\boldsymbol{S}\boldsymbol{\zeta} + \boldsymbol{h} \tag{60}$$

where

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \\ \boldsymbol{H}_3 \\ \boldsymbol{H}_4 \end{pmatrix} = \begin{pmatrix} \operatorname{diag}\{\alpha_1, \cdots, \alpha_n\} - \dfrac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}^T}\boldsymbol{W}^T \\ \operatorname{diag}\{x_1, \cdots, x_n\} \\ -(\boldsymbol{I}_n \otimes \boldsymbol{\sigma}(\boldsymbol{x})) \\ -(\boldsymbol{I}_n \otimes \boldsymbol{u}) \end{pmatrix} \tag{61}$$

and

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{C}^T\boldsymbol{Q}^T\boldsymbol{e} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}$$

where

$$\boldsymbol{S} = (\boldsymbol{I}_n \quad \boldsymbol{0} \quad \boldsymbol{0} \quad \boldsymbol{0}). \tag{62}$$

Substituting (59) and (60) in (58) and equating the zero- and first-degree terms produces the following equations[7]:

- Zero degree:

$$\dot{\hat{\boldsymbol{z}}} = \boldsymbol{g} - \boldsymbol{P}\boldsymbol{h} \tag{63}$$

- First degree:

$$\dot{\boldsymbol{P}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}^T}\boldsymbol{P} - \boldsymbol{P}\boldsymbol{H}\boldsymbol{S} - \boldsymbol{P}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}^T}\boldsymbol{P} \tag{64}$$

[7]Dropping the independent variables in the vector field expressions.

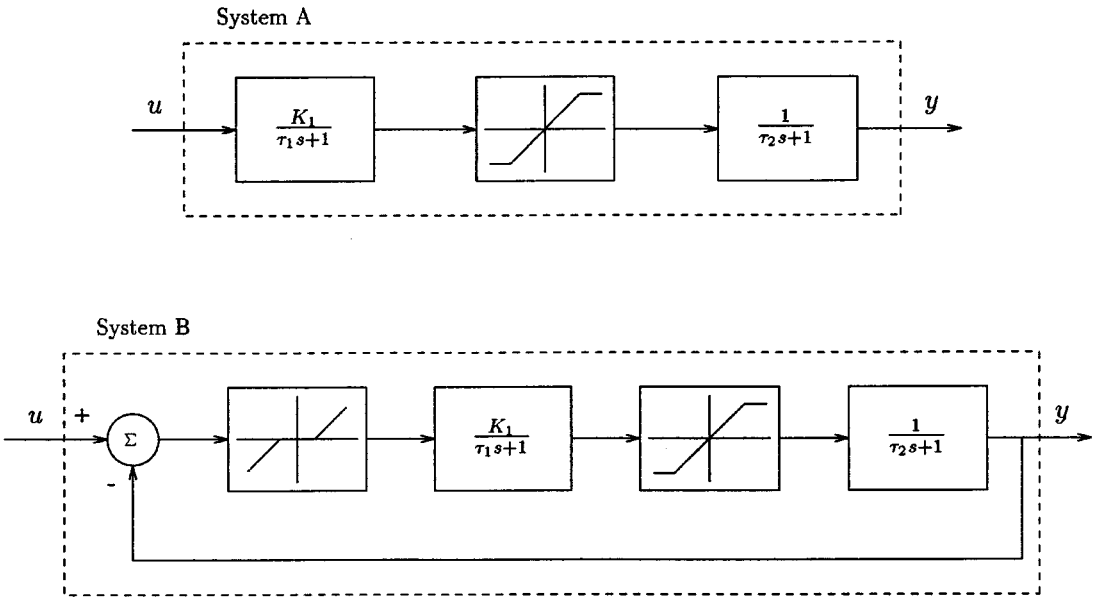

Fig. 5. Second-order nonlinear systems used in the identifications.

where

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}^T} = \begin{pmatrix} \dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^T} & \dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{\alpha}^T} & \dfrac{\partial \boldsymbol{f}}{\partial (\mathrm{col}\boldsymbol{W}^T)^T} & \dfrac{\partial \boldsymbol{f}}{\partial (\mathrm{col}\boldsymbol{B}^T)^T} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \in M_{n^2+(2+m)n}(\mathbb{R}) \tag{65}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}^T} = \begin{pmatrix} \dfrac{\partial \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{e}}{\partial \boldsymbol{x}^T} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in M_{n^2+(2+m)n}(\mathbb{R}) \tag{66}$$

and $\boldsymbol{P}$ can be partitioned as

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} & \boldsymbol{P}_{13} & \boldsymbol{P}_{14} \\ \boldsymbol{P}_{21} & \boldsymbol{P}_{22} & \boldsymbol{P}_{23} & \boldsymbol{P}_{24} \\ \boldsymbol{P}_{31} & \boldsymbol{P}_{32} & \boldsymbol{P}_{33} & \boldsymbol{P}_{34} \\ \boldsymbol{P}_{41} & \boldsymbol{P}_{42} & \boldsymbol{P}_{43} & \boldsymbol{P}_{44} \end{pmatrix} \in M_{n^2+(2+m)n}(\mathbb{R}) \tag{67}$$

with $\boldsymbol{P}_{ij} \in M_n(\mathbb{R})$, $i, j \in \{1, 2\}$; $\boldsymbol{P}_{31}$, $\boldsymbol{P}_{32} \in M_{n^2,n}(\mathbb{R})$; $\boldsymbol{P}_{13}$, $\boldsymbol{P}_{23} \in M_{n,n^2}(\mathbb{R})$; $\boldsymbol{P}_{33} \in M_{n^2}(\mathbb{R})$; $\boldsymbol{P}_{41}$, $\boldsymbol{P}_{42} \in M_{nm,n}(\mathbb{R})$; $\boldsymbol{P}_{14}$, $\boldsymbol{P}_{24} \in M_{n,nm}(\mathbb{R})$; $\boldsymbol{P}_{43} \in M_{nm,n^2}(\mathbb{R})$; $\boldsymbol{P}_{34} \in M_{n^2,nm}(\mathbb{R})$, and $\boldsymbol{P}_{44} \in M_{nm,nm}(\mathbb{R})$. With this approach, the complexity of the above equations is, in terms of number of differential equations, $n_t = (n^2+2n+nm)+(n^2+2n+nm)^2$. However, without loss of generality, it is possible to assume that matrix $\boldsymbol{P}$ is symmetric ($\boldsymbol{P}_{ij} = \boldsymbol{P}_{ji}, \forall i, j$) in order to reduce the complexity of the learning problem. This assumption is feasible because in (56) of the invariant imbedding procedure the $\boldsymbol{P}$ matrix can be chosen to be symmetric without degrading the approximation it involves.

With the above considerations, the equations for the on-line identification problem can be formulated as

$$\begin{aligned}
\dot{\hat{\boldsymbol{z}}}_{\boldsymbol{x}} =& -\boldsymbol{U}(\boldsymbol{\alpha} \otimes \hat{\boldsymbol{z}}_{\boldsymbol{x}}) + \left(\boldsymbol{\sigma}^T(\hat{\boldsymbol{z}}_{\boldsymbol{x}}) \otimes \boldsymbol{I}_n\right) \boldsymbol{V}_{nn} \hat{\boldsymbol{z}}_{\boldsymbol{W}} \\
&+ \left(\boldsymbol{u}^T \otimes \boldsymbol{I}_n\right) \boldsymbol{V}_{mn} \hat{\boldsymbol{z}}_{\boldsymbol{B}} + \boldsymbol{d} \\
&- \boldsymbol{P}_{11} \boldsymbol{C}^T \boldsymbol{Q}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) && (68) \\
\dot{\hat{\boldsymbol{z}}}_{\boldsymbol{\alpha}} =& -\boldsymbol{P}_{21} \boldsymbol{C}^T \boldsymbol{Q}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) && (69) \\
\dot{\hat{\boldsymbol{z}}}_{\boldsymbol{W}} =& -\boldsymbol{P}_{31} \boldsymbol{C}^T \boldsymbol{Q}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) && (70) \\
\dot{\hat{\boldsymbol{z}}}_{\boldsymbol{B}} =& -\boldsymbol{P}_{41} \boldsymbol{C}^T \boldsymbol{Q}^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) && (71) \\
\dot{\boldsymbol{P}}_{11} =& -\left(\boldsymbol{H}_1^T \boldsymbol{P}_{11} + \boldsymbol{P}_{11} \boldsymbol{H}_1 + \boldsymbol{H}_2^T \boldsymbol{P}_{21} + \boldsymbol{P}_{21}^T \boldsymbol{H}_2\right. \\
&\left. + \boldsymbol{H}_3^T \boldsymbol{P}_{31} + \boldsymbol{P}_{31}^T \boldsymbol{H}_3 + \boldsymbol{H}_4^T \boldsymbol{P}_{41} + \boldsymbol{P}_{41}^T \boldsymbol{H}_4\right) \\
&+ \boldsymbol{P}_{11} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{11} && (72) \\
\dot{\boldsymbol{P}}_{21} =& -\left(\boldsymbol{P}_{21} \boldsymbol{H}_1 + \boldsymbol{P}_{22}^T \boldsymbol{H}_2 + \boldsymbol{P}_{32}^T \boldsymbol{H}_3 + \boldsymbol{P}_{42}^T \boldsymbol{H}_4\right) \\
&+ \boldsymbol{P}_{21} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{11} && (73) \\
\dot{\boldsymbol{P}}_{31} =& -\left(\boldsymbol{P}_{31} \boldsymbol{H}_1 + \boldsymbol{P}_{32}^T \boldsymbol{H}_2 + \boldsymbol{P}_{33} \boldsymbol{H}_3 + \boldsymbol{P}_{43}^T \boldsymbol{H}_4\right) \\
&+ \boldsymbol{P}_{31} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{11} && (74 \\
\dot{\boldsymbol{P}}_{41} =& -\left(\boldsymbol{P}_{41} \boldsymbol{H}_1 + \boldsymbol{P}_{42} \boldsymbol{H}_2 + \boldsymbol{P}_{43} \boldsymbol{H}_3 + \boldsymbol{P}_{44}^T \boldsymbol{H}_4\right) \\
&+ \boldsymbol{P}_{41} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{11} && (75) \\
\dot{\boldsymbol{P}}_{22} =& \boldsymbol{P}_{21} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{21}^T && (76) \\
\dot{\boldsymbol{P}}_{32} =& \boldsymbol{P}_{31} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{21}^T && (77) \\
\dot{\boldsymbol{P}}_{42} =& \boldsymbol{P}_{41} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{21}^T && (78) \\
\dot{\boldsymbol{P}}_{33} =& \boldsymbol{P}_{31} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{31}^T && (79) \\
\dot{\boldsymbol{P}}_{43} =& \boldsymbol{P}_{41} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{31}^T && (80) \\
\dot{\boldsymbol{P}}_{44} =& \boldsymbol{P}_{41} \boldsymbol{C}^T \boldsymbol{Q}^T \boldsymbol{C} \boldsymbol{P}_{41}^T && (81)
\end{aligned}$$

where $\boldsymbol{V}_{kl} = \sum_{i=1}^{k} \sum_{j=1}^{l} \boldsymbol{E}_{ij} \otimes \boldsymbol{E}_{ij}^T \in M_{kl,kl}(\{0, 1\})$ and $\boldsymbol{E}_{ij} = \boldsymbol{e}_i \boldsymbol{e}_j^T \in M_{k,l}(\{0, 1\})$ is the Kronecker matrix. The above system of matrix differential equations have the initial conditions $\boldsymbol{P}_{ii}(0) = \epsilon \boldsymbol{I}$, $i = 1, 2, 3$, $\epsilon > 0$, and a small absolute value and $\hat{\boldsymbol{z}}.(0)$ as close as possible to the unknown correct values. This second group of initial conditions cannot be known *a priori*, so their values must be

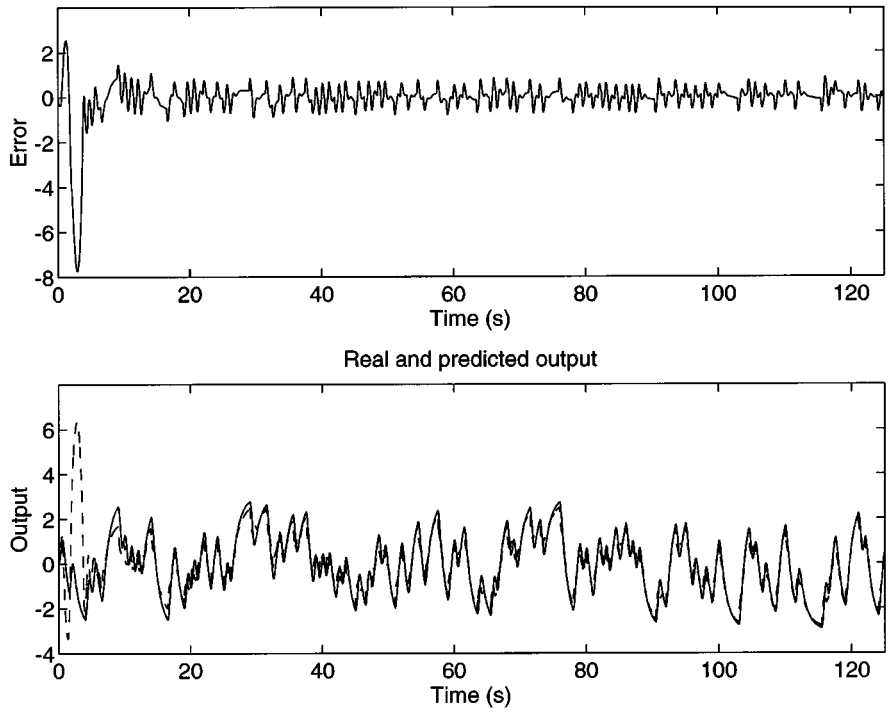

Fig. 6. Identification results in System A: $\tau_1 = 0.2$, $K_1 = 2$, $\tau_2 = 0.2$, saturation limits $= 3, -3$. Gradient parameter adaptation based on sensitivity analysis.

set randomly. However, the closer they are to their correct values, the faster the system convergence will be.

5) Complexity Issues: The complexity, in terms of the number of differential equations, of this system of matrix differential equations is the following.

- For the first group ($\hat{\boldsymbol{z}}.$): $n_{\boldsymbol{Z}} = n^2 + 2n + nm$.
- For the second group ($\boldsymbol{P}..$): $n_{\boldsymbol{P}} = n_{\boldsymbol{P}.1} + n_{\boldsymbol{P}.2} + n_{\boldsymbol{P}.3} + n_{\boldsymbol{P}.4} = (n^4/2) + (2+m)n^3 + ((m^2+4m+5)/2)n^2 + ((m+2)/2)n$. In detail, by columns of the block matrix $\boldsymbol{P}$:

  First $\boldsymbol{P}$ column ($\boldsymbol{P}.1$): $n_{\boldsymbol{p}.1} = ((n^2+n)/2)+n^2+n^3+n^2m$;

  Second $\boldsymbol{P}$ column ($\boldsymbol{P}.2$): $n_{\boldsymbol{p}.2} = ((n^2+n)/2)+n^3+n^2m$;

  Third $\boldsymbol{P}$ column ($\boldsymbol{P}.3$): $n_{\boldsymbol{p}.3} = ((n^4+n^2)/2)+n^3m$;

  Fourth $\boldsymbol{P}$ column ($\boldsymbol{P}.4$): $n_{\boldsymbol{p}.4} = ((n^2m^2+nm)/2)$.

Then, for the whole system the complexity in terms of the number of differential equations is $n_{\text{total}} = (n^4/2)+(2+m)n^3+((m^2+4m+7)/2)n^2+((3m+6)/2)n$. It is important to remark that the complexity of this method is greater than in the other approach based on sensitivity analysis and gradient update of the parameters of the neural models [21].

IV. Implementation Characteristics

The two approaches to on-line system identification using additive neural network models developed in the previous sections have been implemented in an attempt to obtain a fully automated approach to the generation of the identification models, the auxiliary equations and the parameter update equations.

The equations for the parameter update of the models have been obtained through the implementation of the developed methods using a symbolic manipulation program, specifically, MapleV [40]. This fact also justifies the level of abstraction used in the formalization of the developments.

The procedure to obtain an executable binary file to perform the numerical experimentation is sketched in Fig. 4. First, the specification of the architecture of the neural model (number of inputs, outputs and nodes together with the values of the parameters which have been assumed constant) is input to the symbolic manipulation package developed for the selected method. The output is a set of source-code files ready to be compiled and linked with the main program and the numerical integration kernel. For the numerical integration itself, two different tools have been used: the continuous-time simulation language ACSL [41] and the ODEPACK [42] integration package (LSODES routine). The reason for this choice lies in the higher performance of the LSODES routine for large problems, since it works with an explicit Jacobian matrix using sparse-matrix algebra.

In the numerical experiments, which will be presented in the following section, the source-code program also includes the necessary routines to simulate the real data set that must be input to the simulation problem.

Also, it is important to remark that, in the case of an implementation of the identification methods with the real hardware plant in the loop, it might be necessary to change the numerical integration algorithms to others which are specifically suited for real-time operation, such as described in [43].

V. Validation of Identification Methods

This section presents several results obtained with the two developed on-line parameter update methods together with a reference solution consisting of an identification using feedforward neural-network models with tapped delay lines.

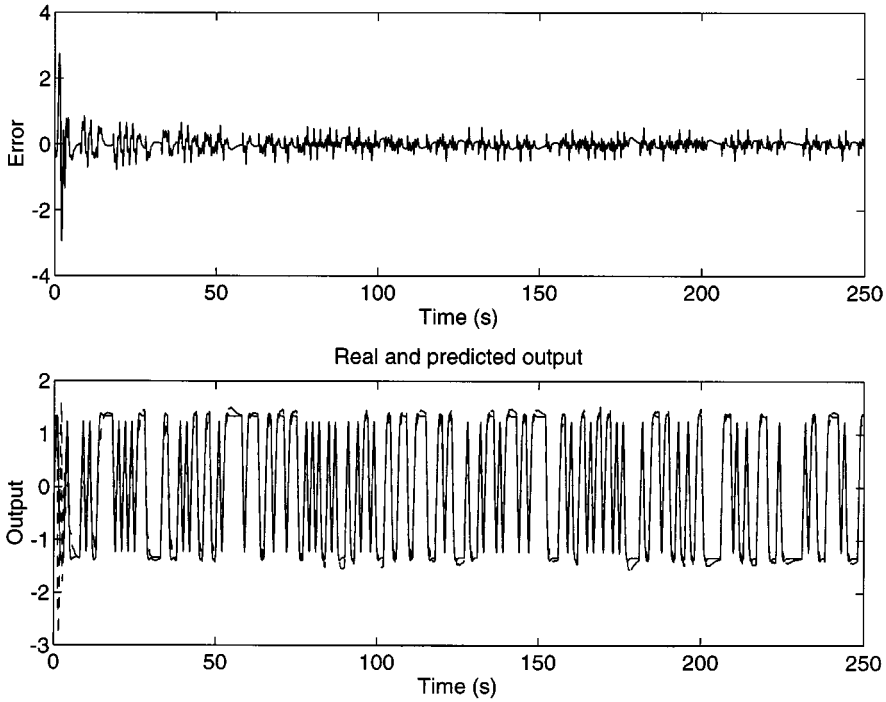

Fig. 7. Identification results in System B: same values as system A and a dead zone bounds of +1, −1. Gradient parameter adaptation based on sensitivity analysis.

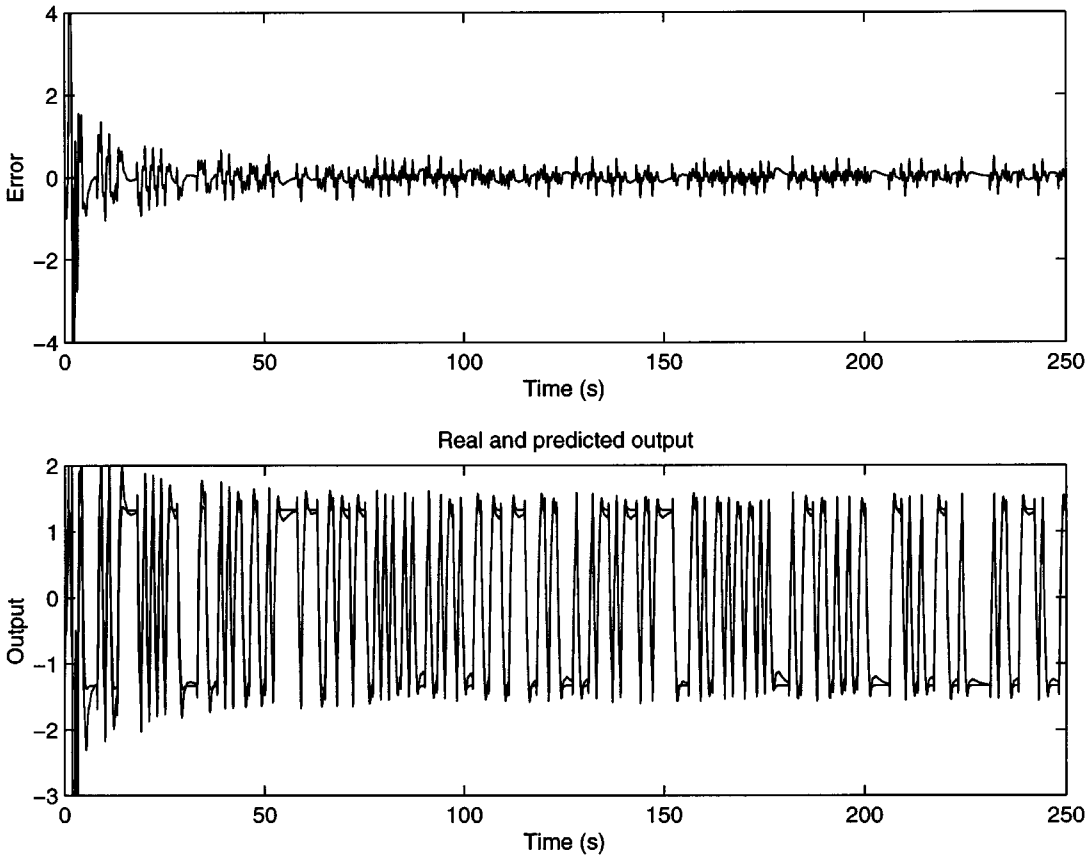

Fig. 8. Identification results in System B: same values as system A and a dead zone bounds of $+1$, -1. Variational and invariant imbedding technique.

A. Test Cases

Three representative examples are used in this paper: two simulated second-order nonlinear systems and a real data set consisting of the electrical power demand and production readings in a hydroelectric power plant.

1) Simulated Nonlinear Second-Order System: Fig. 5 shows the simulated nonlinear systems used for the experi-

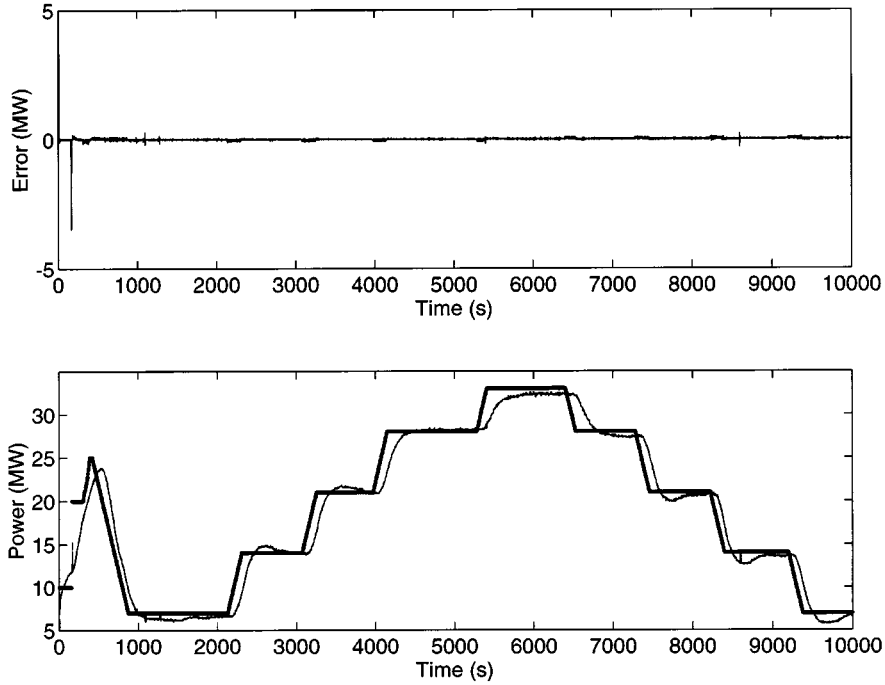

Fig. 9. Identification error (top) and reference (thick), output and model output (thin) (bottom) for the real power plant data set. Gradient parameter adaptation based on sensitivity analysis.

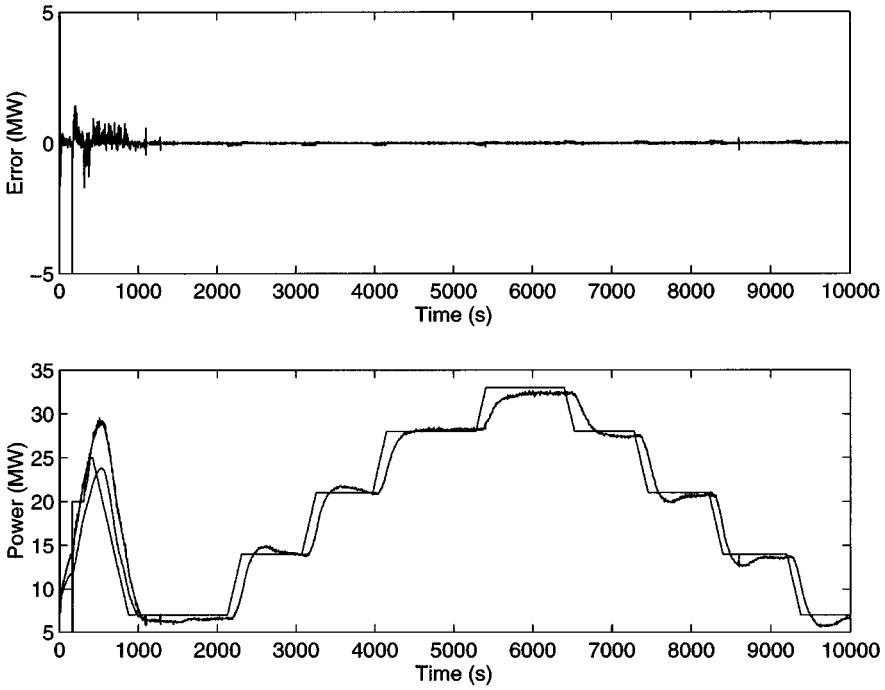

Fig. 10. Identification error (top) and reference, output, and model output (bottom) for the real power plant data set. Variational and invariant imbedding technique.

ments. The first one contains a cascade of first-order systems with a saturation element between them, and the second one is composed of the same elements plus a dead band working in closed loop. This kind of systems is very common in industrial processes, especially in motion control systems.

In all the experiments a maximum length binary sequence has been used as input to the plant and the model. This kind of input has been chosen because its industrial use is more extended than the pure white noise. Also, it is important to point out that no tuning of the initial conditions of the models has been carried out.

2) Real Data Set from a Hydroelectric Power Plant: In this case, the system to be identified is a hydroelectric group from a power plant of the utility company ENHER.[8] The real data set consists of two time series corresponding to the electrical power desired from the plant, and the real power production.

[8]Empresa Nacional Hidroeléctrica Ribagorzana.

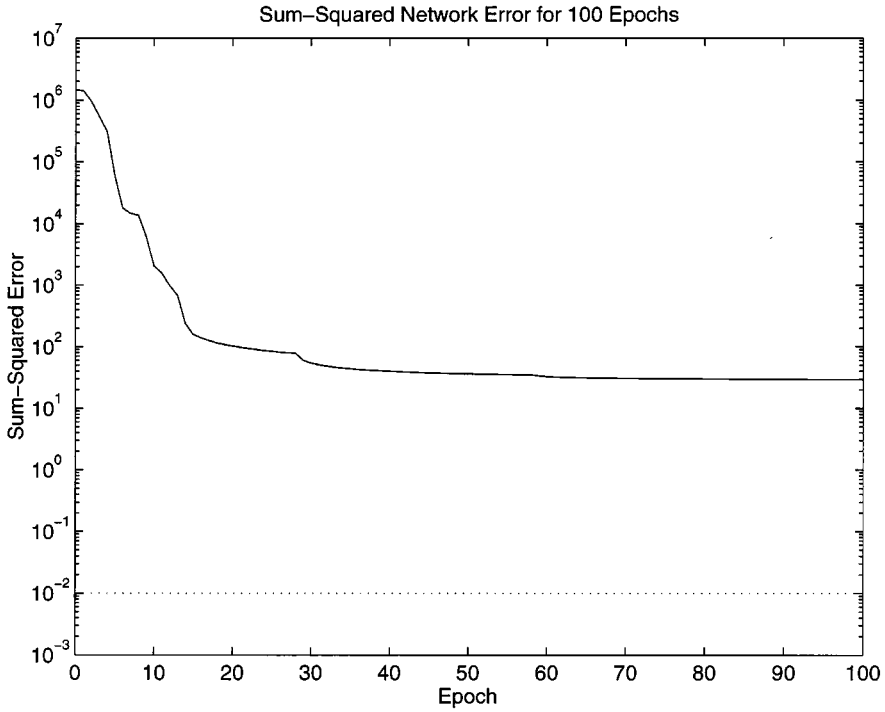

Fig. 11. Sum squared network error for the first one hundred epochs of the training phase.

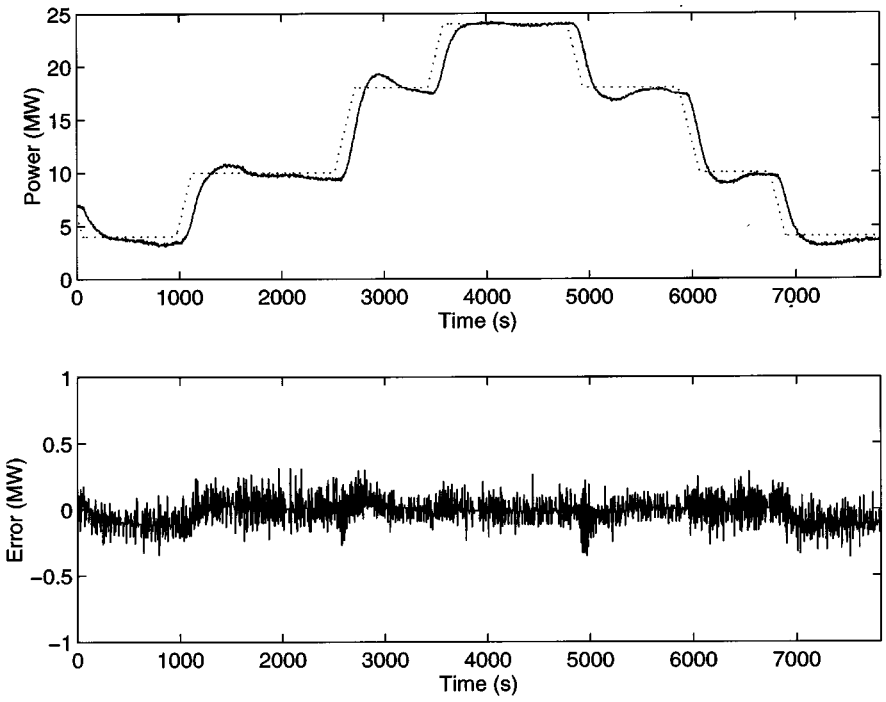

Fig. 12. Series-parallel prediction over the test set: (top) power demand (dotted), real output power (solid), and predicted output power (dash-dot); and (bottom) identification error.

The sampling period of the data set is 3 s and the total time recorded is 18 663 s.

B. Experimental Results

This section presents some results to illustrate the identification performance of the proposed class of models together with their on-line parameter adaptation algorithms.

1) Results with the Simulated Plants: For System A, Fig. 6 shows the identification error and the real and model output for an experiment with a five-node ADC model with gradient parameter adaptation based on sensitivity analysis and $\epsilon = 0.1$. At time $t = 100$ s the adaptation mechanism has stopped and the model performs its prediction task quite well, which implies that the underlying dynamics of the plant have been acquired by the ADC model.

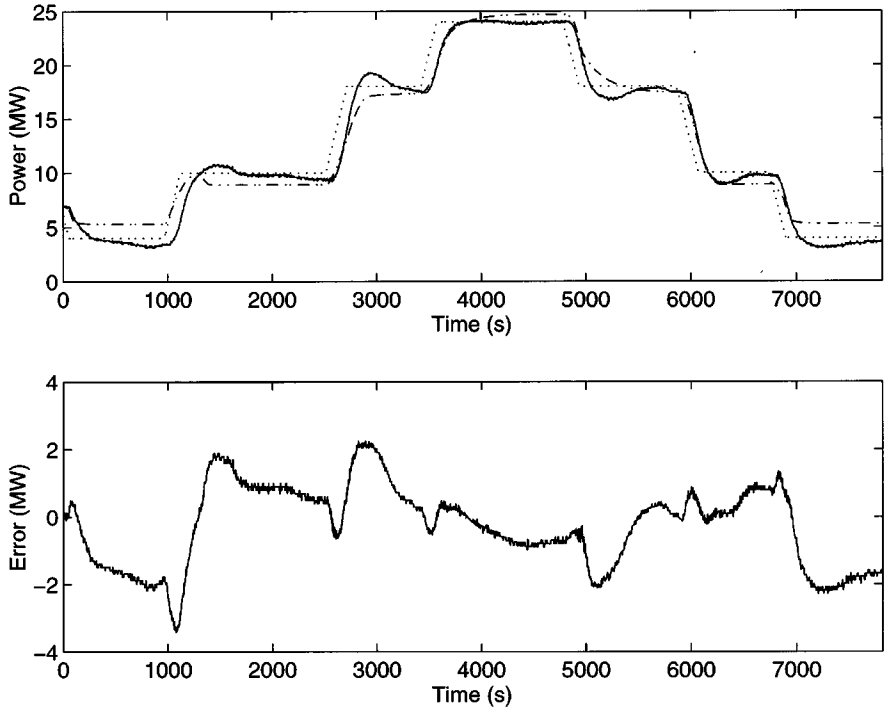

Fig. 13. Parallel prediction over the test set: (top) power demand (dotted), real output power (solid), and predicted output power (dash-dot); and (bottom) identification error.

The five-node ADC model is the minimum configuration that acquires the dynamic behavior of the real process, and this fact is recognized because the weights maintain almost constant values some time after the performance of the model is good. That is to say, it is the model and not the adapter that learns the process behavior.

For System B, Fig. 7 shows the identification error and the real and predicted output, also with a five-node ADC model using gradient parameter adaptation ($\epsilon = 0.5$). Fig. 8 shows the same data for an ADC model of the same complexity with the variational and invariant imbedding technique.

Among the numerical experiments performed in this work, a lack of excitation in the input to the system has been tested and the results show a high degree of robustness since no drift is observed in the model parameters.

2) Results with the Real Data Set: Now, the parameter adaptation algorithms are applied to the real data set from the power plant. In particular, only the first 3334 points from the input and output time series are used, which correspond to the first 10 002 s.

Fig. 9 shows the real and predicted output power and the demanded power (bottom), and the identification error (top) for a five-node ADC model with gradient parameter adaptation based on sensitivity analysis. As can be observed, the real output power is virtually indistinguishable from the model output power, except for the first moments of the on-line adjustment.

Fig. 10 shows the same information as Fig. 9 for a five-node ADC model, now, with variational and invariant imbedding adaptation technique. It can also be observed that the identification error is quite small except at the beginning of the on-line adjustment of the model.

For both of the learning methods, when the predicted output is very similar to the real output, the adaptation mechanism has been stopped with little degradation of the identification error, which implies that the ADC model has acquired the dynamics of the process.

3) Comparison of Performance Against Feedforward Neural Network Identifiers: In order to establish a reference for comparison of the identification performance of the proposed methods, an identifier using static feedforward neural networks has been designed. This method has been chosen as a reference since it is undoubtedly one of the most efficient state-of-the-art nonlinear identification techniques. Neural network models with tapped delay lines in the inputs and outputs have been used. In particular, the considered models are

$$\hat{y}_k = f(y_{k-1}, \cdots, y_{k-s}, u_k, \cdots, u_{k-m}) \tag{82}$$

where subindex k denotes an observation of the variable in question at time kT, T being the sampling period, s and m are the maximum delays in the output and the input of the model, i.e., the depth of the historical windows in the input and output time series, and f is a static function that represents the neural model.

The approach followed for the training phase is a series-parallel one.The model is trained with real data of current and past inputs, as well as past outputs, in order to predict the corresponding current output.

The series-parallel approach to identification in the connectionist context involves the teacher-forcing concept of learning. This means that the test of performance is carried out by forcing the inputs and the delayed values of the inputs and outputs of the system to some prespecified values, previously measured in the plant and contained in the test set. Then, the delayed data are not

generated by the identifier. Conversely, in the parallel identification configuration, the developed observations are generated by the identifier itself.

When using feedforward neural networks, it is usually more efficient to train the identifier with a series-parallel configuration. However, in order to test the ability of the identifier to perform on-line, it is necessary to analyze its results in the parallel configuration. A problem that is frequently encountered in this situation, is that an identifier that performed extremely well in the series-parallel configuration does not provide satisfactory results in the parallel configuration. This implies that the dynamics of the process have not been captured correctly.

In the experiments shown below, a FFS neural identifier was developed for the real data set. The FFS model was carefully designed by performing a thorough search of the best number of delays and hidden nodes. The neural network was trained off line (in a series-parallel configuration) and tested on line (in a parallel configuration).

The training of the FFS models (with hyperbolic tangent sigmoid transfer functions in their nodes) is performed over a training data set, made up of first 3364 values from the input–output time series, with a Levenberg–Marquardt optimization algorithm. The criterion to select the best model is its prediction behavior over the test set, which is composed of the remaining values (2857) of the input–output time series.

The learning evolution for the best model can be observed in Fig. 11. This model has seven delays in the input, seven delays in the output, and 15 nodes in its hidden layer. Fig. 12 shows the behavior of the model in the prediction of the test set in a series-parallel configuration. Conversely, Fig. 13 shows the prediction behavior of the same model over the test set in a parallel configuration. In the ADC model presented in this work, the parametric identification is performed on line, i.e., in a parallel configuration, involving an infinite-step prediction. Therefore, the results of the FFS neural identifier and the ADC model must be compared for the parallel configuration. From the analysis of Figs. 9, 10, and 13 it follows that the tracking performance and the identification error of the ADC model is significantly better than that of the FFS model, especially in the transitory parts of the system output.

VI. Conclusions and Further Work

The connectionist models presented in the paper have been designed in order to obtain an efficient tool for the identification of complex systems, where the dynamic process may be partially or completely unknown. The selection of dynamic neural networks provides the model with better abilities to capture the unknown dynamics and to generate an internal state representation of the system, as opposed to other static connectionist models. The use of continuous-time models makes it possible to use an existing theoretical background for subsequent system analysis and controller design.

The experimental results presented in the paper show how the proposed ADC model can efficiently identify two synthetic nonlinear systems and one highly nonlinear real plant. The comparison of identification performance with that of an FFS neural network illustrates two facts. First, the FFS network requires an important phase of iterative development in order to achieve a structure that can efficiently approximate the nonlinear dynamics, whereas this process is almost automatic in the proposed ADC model. Second, the tracking results with the ADC model are significantly better than those obtained with the optimally selected FFS neural identifier. It may be argued that the ADC method involves a relatively complex mathematical formulation. However, in our implementation, a symbolic manipulation package based on MapleV automatically generates the FORTRAN code whose on-line execution performs the actual identification task, thus reducing mathematical manipulation and operation to the minimum.

Concerning the comparison of the two proposed ADC learning methods, the gradient parameter adaptation based on sensitivity analysis outperforms the variational and invariant imbedding technique in the initial stages of learning. However, once the model is adapted, their performances are qualitatively similar, as shown in the numerical experiments. The detailed quantitative behaviors depend on the dynamics of the identified systems, without a clear predominance of one method over the other. The latter method has, however, the important advantage of being easily extendable to the treatment of noisy systems.

So far the ADC identification models have only been developed and tested for deterministic dynamic systems. A further step of the research involves the extension of this method for stochastic processes as well. In particular, it is envisaged to extend the invariant imbedding parameter update method to consider the stochastic case. Additionally, research is ongoing in the convergence characteristics of the parameter update methods developed in this work.

Acknowledgment

The authors would like to thank the power utility ENHER for providing the power production data.

References

[1] P. J. Antsaklis, "Neural networks in control systems," *IEEE Control Systems Mag.*, no. 2, pp. 3–5, Apr. 1990.

[2] M. J. Korenberg and L. D. Paarmann, "Orthogonal approaches to time-series analysis and system identification," *IEEE Signal Proc. Mag.*, no. 7, pp. 29–43, July 1991.

[3] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Proc. Mag.*, no. 7, pp. 10–26, July 1991.

[4] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–26, Mar. 1990.

[5] ——, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 252–262, Mar. 1991.

[6] M. M. Polycarpou and P. A. Ioannou, "Identification and control of nonlinear systems using neural network models: Design and stability analysis," Univ. of Southern California, Tech. Rep. 91-09-01, Sept. 1991.

[7] F. Girosi and T. Poggio, "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, pp. 465–469, 1989.

[8] N. V. Bhat, P. A. Minderman, T. J. McAvoy, and N. S. Wang, "Modeling chemical process systems via neural computation," *IEEE Control Systems Mag.*, vol. 2, pp. 24–29, Apr. 1990.

[9] S. Weerasooriya and M. A. El-Sharkawi, "Identification and control of a dc motor using back-propagation neural networks," *IEEE Trans. Energy Conversion*, vol. 6, no. 4, pp. 663–669, Dec. 1991.

[10] R. E. Loke and G. Cembrano, "Neural adaptive control of a bioreactor," in *Preprints of the 2nd IFAC Symposium on Intelligent Components and Instruments for Control Applications (SICICA'94)*, Cs. Bányász, Ed.. Budapest, Hungary, June 1994, pp. 182–186.

[11] V. Ruiz and C. Torras, "On-line learning with minimal degradation in feedforward networks," *IEEE Trans. Neural Networks*, vol. 6, no. 3, pp. 657–668, 1995.
[12] G. Cembrano, G. Wells, J. Sarda, and A. Ruggeri, "Dynamic control of a robot arm based on neural networks," *Control Engineering Practice*, vol. 5, no. 4, pp. 485–492, 1997.
[13] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*. Cambridge, MA: MIT Press, 1990.
[14] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–26, Mar. 1990.
[15] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, "Memory neural networks for identification and control of dynamical systems," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 306–319, 1994.
[16] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of recurrent multilayer perceptron in modeling complex process dynamics," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 255–266, 1994.
[17] S. W. Piche, "Steepest descent algorithms for neural network controllers and filters," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 198–212, 1994.
[18] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[19] R. M. Sanner and J.-J. E. Slotine, "Direct adaptive control using Gaussian networks," Nonlinear Systems Lab., MIT, Tech. Rep. SL-910303, Mar. 1991.
[20] M. Sato, "A learning algorithm to teach spatiotemporal patterns to recurrent neural networks," *Biol. Cybern.*, vol. 62, pp. 259–263, 1990.
[21] R. Griñó, "Nonlinear system identification using additive dynamic neural networks," in *Postprints of the 2nd IFAC Symposium on Intelligent Components and Instruments for Control Applications (SICICA'94)*, Budapest, Hungary, June 1994, pp. 437–442.
[22] Q. H. Wu, B. W. Hogg, and G. W. Irwin, "A neural network regulator for turbogenerators," *IEEE Trans. Neural Networks*, vol. 3, pp. 95–100, Jan. 1992.
[23] F. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Control Systems Mag.*, no. 2, pp. 44–48, Apr. 1990.
[24] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, no. 5, pp. 1212–1227, 1995.
[25] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems—A survey," *Automatica (J. IFAC)*, vol. 28, no. 6, pp. 1083–1112, Dec. 1992.
[26] A. Isidori, *Nonlinear Control Systems*, NY: Springer-Verlag, 1989.
[27] W. J. Vetter, "Derivative operations on matrices," *IEEE Trans. Automat. Contr.*, vol. 15, pp. 241–244, Apr. 1970.
[28] J. W. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Trans. Circuits Syst.*, vol. 25, pp. 772–781, Sept. 1978.
[29] R. Griñó, "Stability analysis of continuous time additive dynamic neural networks," in *Actes del 1r Seminari de treball en Automàtica, Robòtica i Percepció*, A. Català and J. Aguilar, Eds. Barcelona, Spain, Feb. 1996, pp. 117–127.
[30] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–191, 1989.
[31] V. Kurkova, "Kolmogorov's theorem is relevant," *Neural Computation*, vol. 3, pp. 617–622, 1991.
[32] F. Albertini and E. D. Sontag, "For neural networks, function determines form," *Neural Networks*, vol. 6, pp. 975–990, 1993.
[33] K. Doya, "Universality of fully-connected recurrent neural networks," Dept. of Biology, UCSD, Tech. Rep., Feb. 1993.
[34] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, pp. 801–806, 1993.
[35] P. M. Frank, *Introduction to System Sensitivity Theory*. New York, NY: Academic, 1978.
[36] A. E. Bryson and Y. Ho, *Applied Optimal Control*. New York, NY: Ginn , 1969.
[37] A. P. Sage and J. L. Melsa, *System Identification, vol. 80 of Mathematics in Science and Engineering*. New York, NY: Academic, 1971.
[38] R. Bellman and G. M. Wing, "An introduction to invariant imbedding," in *Classics in Applied Mathematics*. Philadelphia, PA: SIAM, 1992.
[39] R. Griñó, "On-line system identification using additive dynamic neural networks. An invariant imbedding approach," in *Proc. 1996 Int. Workshop Neural Networks Identification, Control, Robotics, Signal/Image Processing (NICROSP'96)*, V. Piuri, Ed., Venice, Italy, Aug. 1996, pp. 437–442.
[40] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, *MapleV. Library Reference Manual*. New York, NY: Springer-Verlag, 1992.
[41] Mitchell & Gauthier Assoc., *Advanced Continuous Simulation Language (ACSL)—Reference Manual*. Concord, MA: Mitchell & Gauthier , 1991.
[42] A. C. Hindmarsh, "Odepack, a systematized collection of ode solvers," in *Scientific Computing*, R. S. Stepleman, Ed. Amsterdam, The Netherlands: North-Holland, 1983.
[43] R. M. Howe, "A new family of real-time predictor-corrector integration algorithms," *Simulation*, vol. 57, no. 3, pp. 177–186, 1991.

Robert Griñó (S'90–M'96) received the M.Sc. degree in electrical engineering and the Ph.D. degree in automatic control from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1989 and 1997, respectively.

During 1990 and 1991 he worked as a Research Assistant at the Instituto de Cibernética (UPC) and, since 1992, he has been an Assistant Professor at the Systems Engineering and Automatic Control Department and at the Instituto de Organización y Control de Sistemas Industriales, the Universitat Politècnica de Catalunya. He has been involved in the INNOVATION project, Constraint Logic Operation of Water Systems (CLOCWISE) and the Spanish Research and Technology Council projects Advanced Nonlinear Control Techniques and Nonlinear Control of Dynamical Systems. His research interests include nonlinear control, stability theory, sensitivity theory, differential algebraic systems, and identification.

Dr. Griñó is a member of SIAM and SCS and is an affiliate member of IFAC.

Gabriela Cembrano received the M.Sc. degree in power engineering and the Ph.D. degree in automatic control from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1984 and 1988, respectively.

She has been working in applied research in automatic control since 1985 and was the Head of the Control Division of the Instituto de Cibernética from 1991–1996. Most recently, she has been involved in the ESPRIT projects Robot Control Based on Neural Network Systems (CONNY) and Knowledge Capture for Advanced Supervision of Water Distribution Networks (WATERNET) and the Spanish Research and Technology Council projects Advanced Nonlinear Control Techniques and Safety in Complex Dynamic Systems. Her major research interests are optimal and adaptive control, intelligent control, and modeling of dynamic systems. She is an Assistant Researcher of the Consejo Superior de Investigaciones Científicas, Instituto de Robótica y Informática Industrial, Universitat Politècnica de Catalunya, and she teaches Ph.D. courses in optimal and adaptive control.

Carme Torras received the M.Sc. degree in mathematics from the Universitat de Barcelona, Barcelona, Spain, in 1978, the M.Sc. degree in computer science from the University of Massachusetts at Amherst in 1981, and the Ph.D. degree in computer science from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1984.

Based on her thesis, she authored the book *Temporal-Pattern Learning in Neural Models* (Berlin, Germany: Springer-Verlag, 1985). Neurocomputing and robot motion planning are her major research interests. She has been involved in several ESPRIT projects, among them Robot Control Based on Neural Network Systems" (CONNY), Self-Organization and Analogical Modeling Using Subsymbolic Computing (SUBSYM), Planning RObot Motion (PROMotion), and "Behavioural Learning: Sensing and Acting" (B-LEARN). She is a Professor of Research in the Consejo Superior de Investigaciones Científicas, Universitat Politècnica de Catalunya, and she teaches Ph.D. courses in the fields of robotics and artificial intelligence at the Universitat Politècnica de Catalunya.