

Neural Modeling of Dynamic Systems with Nonmeasurable State Variables

Cesare Alippi, *Senior Member, IEEE*, and Vincenzo Piuri, *Senior Member, IEEE*

Abstract—The paper studies the ability possessed by recurrent neural networks to model dynamic systems when some relevant state variables are not measurable. Neural architectures based on virtual states—which naturally arise from a space state representation—are introduced and compared with the more traditional neural output error ones. Despite the evident potential model ability possessed by virtual state architectures we experimented that their performances strongly depend on the training efficiency. A novel validation criterion for neural output error architectures is suggested which allows to assess the neural network not only in terms of its approximation accuracy but also with respect to stability issues.

Index Terms—Dynamic systems, recurrent neural networks, stability, training.

I. INTRODUCTION

THE development of a black-box model for a dynamic system is an assessed procedure to be envisaged whenever the equations ruling the system are unknown, inadequate to explain the process behavior or computationally intractable. To this end, linear system identification techniques [1] have been proved to be valuable and theory-supported tools to solve the problem if the system is linear or the interest is around a working point. Unfortunately, linear models suffer from inaccuracy if the system to be approximated is significantly nonlinear and evolves through different working points.

To overcome such limits, recurrent neural networks [2] have been suggested as alternative solutions whose validity has been shown in a wide range of applications. In the instrumentation and measurement fields, neural networks have been successfully used to implement (e.g., see [3]), enhance (e.g., see [4]), and calibrate (e.g., see [5]) both real and virtual sensors (e.g., see [6]), process signals and images (e.g., see [7]), predict, control, and model plants (e.g., see [8]).

The approximating ability of such neural networks depends on *a priori* information about the system, the number of available data, the influence of noise on data and, mainly, on their structure (the choice for a wrong neural model may, in fact, drastically impair its approximating accuracy).

Inputs and outputs of the system (here accounting for both external outputs and states) are usually implicitly supposed to be measurable which ends to be a reasonable hypothesis in many applications. When this assumption holds, the network weights can be easily obtained by minimizing a discrepancy

function, e.g., the mean squared error function, provided that an efficient training algorithm [9] has been chosen.

Conversely, when some (if not all) the state variables are nonmeasurable, we should identify the neural structure to be used and the most appropriate training procedure.

Sometimes, the presence of nonmeasurable states can be trivialized if there it exists an equivalent input–output representation for the equations ruling the process (mathematically the dependence on the nonmeasurable states can be reduced to a direct dependency on the output dynamic); for instance, this nicely happens when all equations ruling the process are linear. In such a case, the problem can be solved by considering the straightforward input–output nonlinear output error—NOE—representation instead of the more complex state-space one [10]. More realistically, we do not know whether the contribution provided by nonmeasurable states can be reduced to an input–output representation or not and, therefore, we should consider different neural structures (e.g., NOE, virtual states, etc.), identify a good neural family (e.g., determine the number of layers and neurons per layer) and, finally, the best model within such a family (i.e., configure the weights). Hence, the search for an acceptable neural network is an extremely complex and time consuming procedure; we would like to simplify it by eliminating the least efficient/interesting neural topologies from the candidate ones. This operation should be intended as follows. If a neural structure is classified as not efficient with respect to a class of applications it does not mean that it is useless (it could be extremely appropriate for solving a specific problem) but the performances for a generic application are modest. We experienced that several neural structures show this behavior (e.g., the memory neuron networks [2]). These networks either provide really excellent performance on specific applications or, more frequently, modest ones. This should be somehow related to the approximation ability of such network, i.e., the property to model a generic dynamic system, and the effectiveness of the train procedure.

In this paper, we study and compare the features of two neural structures to approximate dynamic systems in order to rank their effectiveness when some state variables are nonmeasurable. The reference architecture is the output error neural network NOE [2] in which the network outputs are feedback to constitute additional inputs and the concept of state. Such networks are extremely interesting for their proven approximation ability within a wide range of applications.

As challenging candidates we suggest neural architectures based on the space-state representation which extends the ones

Manuscript received May 12, 1997; revised August 17, 1999.

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, I-20133 Milano, Italy (e-mail: alippi@elet.polimi.it).

Publisher Item Identifier S 0018-9456(99)09177-9.

presented in the literature [10] by introducing virtual neurons. Virtual neurons are output neurons which do not directly contribute to generate the network output but are feedback and delayed to constitute additional inputs. This ends in additional states for the network, an appreciable feature for dealing with unknown dynamics.

In the comparison we will not consider the wide class of fully recurrent models since it has been shown that any recurrent network can be transformed into a canonical form realizable with static and NOE neural networks [10].

The structure of the paper is as follows. Section II formalizes the problems of learning from examples and introduces the NARX and the NOE neural models. Section III addresses aspects related to the modeling of dynamic systems when some (if not all) dynamic states are not measurable and introduces the virtual neuron networks. Section IV provides a stability criterion for NOE models. Section V presents a comparison among performances provided by different neural model structures. Two experiments are presented addressing a linear model and a nonlinear one, respectively. In particular, the simplified but strongly nonlinear model refers to a drum-type boiler, characterized by a dynamic associated with a nonmeasurable state.

II. NARX AND NOE NEURAL STRUCTURES

For sake of simplicity, let us focus the attention on a MISO system (multiple inputs single output), which receives the n -dimensional input vector $u(t)$ and produces the one-dimensional output $y(t)$, affected by an additive i.i.d. noise η taking into account both the approximating model deficiencies and the instrumentation noise.

In the following, we assume that the system is both observable and controllable.

A specific model approximating the dynamic system will be indicated as $\hat{y}(t) = f(\hat{\theta}, t)$ where $\hat{\theta}$ is the n -dimensional vector of parameters containing weights and biases; $\hat{\theta}$ is obtained by minimizing the mean square error—MSE—function

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N (y(t) - \hat{y}(t, \theta))^2 \quad (1)$$

evaluated over the Z^N set which contains N input–output pairs.

The relationship between the inputs and the output of the system can be captured by considering the *regressor vector* φ , which contains a limited time-window of actual and past inputs, outputs, and—possibly—predicted outputs. In this paper, we consider model structures $\hat{y}(t) = f(\varphi)$ in which the function $f(\cdot)$ is a regression-type neural network, characterized by N_φ inputs, N_h nonlinear hidden units and a single effective linear output [8]. The presence of a dynamic is modeled by delay elements (or *time lags*), which may affect both inputs (time history on external inputs u) and output (hence allowing to deal with the system dynamics).

Several neural model structures have been suggested in the literature (e.g., see [2] for a review), which basically differ in the regressor vector.

In the following, we will focus the attention on NARX, NOE, and virtual neurons model structures. Briefly, the NARX structure considers both past inputs and outputs of the system to infer $\hat{y}(t)$, we have $\varphi = [u(t), u(t-1), \dots, u(t-n_u), y(t-1), \dots, y(t-n_y)]$ and, consequently, the plant needs to be accessible on line to provide the required *ys*. Differently, the NOE structure processes only past inputs and predicted outputs, i.e., $\varphi = [u(t), u(t-1), \dots, u(t-n_u), \hat{y}(t-1), \dots, \hat{y}(t-n_y)]$; it is not required the presence of any *ys* coming from the plant during its operational phase: model and plant are separate entities.

Since static regression-type neural networks are universal static function approximators [11] we implemented them as core of the envisioned model structures. The static assumption, which guarantees the universal approximation property, may be relaxed to deal with dynamic systems: up to now, the approximation ability has been proven to hold also for dynamic systems under the hypothesis that there is an acyclic computational graph from φ to $\hat{y}(t)$. Note that NOE models do not satisfy this hypothesis since their regressor vectors contains past predicted values.

III. MODELING DYNAMIC SYSTEMS HAVING NONMEASURABLE VARIABLES

For ease of notation, but without any loss in generality, we can assume that the deterministic behavior of the process to be modeled is ruled by the system of differential equations

$$\begin{cases} \dot{x} = \bar{g}_x(x, y, u) \\ \dot{y} = \bar{g}_y(x, y, u) \end{cases} \quad (2)$$

where y is the measurable state (which also coincides with the output), and x is an internal state (the extension to several states is trivial and does not modify the results here achieved).

A discrete realization of (2) can be easily obtained (e.g., by applying the Euler explicit discretization) and leads to consider

$$\begin{cases} x(t) = g_x(x(t-1), y(t-1), u(t)) \\ y(t) = g_y(x(t-1), y(t-1), u(t)). \end{cases} \quad (3)$$

Even if some noise affects and corrupts the generation of y and x over time, (3) is representative of many real processes and represents a good starting point for generating new time discrete model structures. Relationships concerning disturbances will be captured by the neural model during the training phase directly from the available data set Z^N . Of course, introduction of new models requires further validation and experimental comparison with already existing ones. This allows either to prove their efficacy or suggests the neural structure not to be considered, hence simplifying the search space.

A. System Modeling with NARX-Based Neural Topologies

If the internal state x is measurable, we could simply consider a neural network approximating each nonlinear function g_x and g_y of system (3), with the regressor vector $\varphi = [x \ y \ u]$. In this case, identification of the best neural network would be a reasonably easy task.

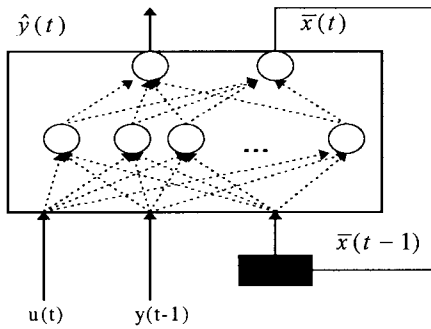


Fig. 1. Black-box realization of (3).

Conversely, if the internal state x is not measurable, generation of a model approximating the (3) becomes more difficult. As we already mentioned, if it is possible to manipulate the system (2) and reduce it to an equivalent explicit nonlinear differential equation of the type

$$\dot{y} = \tilde{g}(\dot{y}, y, \dot{u}, u) \quad (4)$$

then the approximating problem can be solved by considering a NARX type neural network, e.g., with $\varphi = [y(t-1) \ y(t-2) \ u(t) \ u(t-1)]$.

B. System Modeling with Virtual States Neural Topologies

If the state x is not measurable we can introduce a new neural structure resembling the mathematical description of (3). The solution is extremely natural and intuitive and leads to the introduction of a *virtual* nonmeasurable internal state \bar{x} . Such a state is not necessarily an estimate of the real internal state x , even if ideally it could be but, *a priori*, x and \bar{x} are not even correlated. The virtual state \bar{x} is a state variable which simply provides the dynamic necessary to give enough flexibility to the model. The new network structure receives the regressor vector $\varphi = [\bar{x} \ y \ u]$ and, with respect to system (3), it can be represented as in Fig. 1, where we assume that the black rectangle represents a time delay. It is immediate the natural derivation of the neural structure from system (3) and hence its potential interest. The virtual states neural structure extends [10] where it is assumed a measurable x .

Note that the initial condition for \bar{x} is unknown: during training it is required the network also to learn how to become stable. This increases the difficulty of training and it might impair accuracy. We will experimentally show in Section IV that, even if the virtual states network naturally arises from (3), its training phase may be difficult. This is due to the presence of virtual states that, by providing additional degrees of freedom, increase the probability of ending in a suboptimal minimum.

C. System Modeling with NOE-Based Neural Topologies

The model approximation of systems having nonmeasurable states becomes particularly difficult when the neural model must approximate the process or the plant without having access to it on-line during its operational phase. This requirement is of particular interest in all those applications (e.g.,

what-if applications [12]) which require a model evolving independently from the plant.

To afford this case, we can consider two approaches. The first solution, following the guidelines given above, takes into account suitably delayed virtual outputs \bar{x} . Some *a priori* information about the process could be exploited to guide the model definition (e.g., the number of nonmeasurable states may be approximately deduced by relying on physical-based equations). If some states are measurable, we could envisage mixed neural structures which include both virtual and measurable states: if y possesses its own dynamic, a feedback on \hat{y} (as in the NOE model) is necessary.

The second solution is to construct a NOE model to model the system without introducing any nonmeasurable states. This solution is particularly appealing since we do not require any *a priori* information. Training is carried out by using recurrent learning algorithms similar to the ones necessary for the virtual states. In this case there are no problems related to the state initialization since, at the beginning, the estimate values \hat{y} are the values y themselves.

IV. A STABILITY CRITERION FOR NOE MODELS

Once training has been completed one simply observes performances and discovers whether the model approximates the system with enough accuracy or not but, in general, nothing can be said about its stability. In the following, we provide a stability criterion for NOE neural networks. The criterion derives by considering the relationships between NARX and NOE models.

It is obvious that, if $y(t) = \hat{y}(t)$, $\forall t$, the NOE model coincides with the NARX model: the real output feedback at the network input can be substituted by its estimate, achieving a situation similar to the one of (3). It is experimentally well known that, due to the noise presence and the training inefficiencies, this is difficult to obtain. In the worst case, if the training procedure has not been perfected or the model is inaccurate (e.g., because of a wrong choice of Z^N), model and system might even diverge. Nevertheless, it is more likely that the training procedure provides a good NOE model such that error $e(\cdot) = \hat{y}(\cdot) - y(\cdot)$ over time is small.

Differently from linear models, the presence of a saturating nonlinear function in the hidden units (e.g., hyperbolic tangent) cannot introduce a continuous increase/decrease of the output if the system is unstable: if the model trajectory diverges from the system one is within a cylindrical neighborhood centered onto the system trajectory). In fact, the absolute difference between the two trajectories is always structurally bounded by a constant $|e(t)|C$, $\forall t$. If we consider a hyperbolic tangent function $Th(x) = (e^x - e^{-x})/(e^x + e^{-x})$, such a constant is equal to the sum of the absolute values of the bias b_o and the weights θ_i feeding the output neuron, i.e., $C = \sum_{i=1}^{N_h} |\theta_i^o| + |b_o|$.

Under the assumption that the system to be approximated is stable, the performance evolution over time can be studied by analyzing the discrepancy $e(\cdot) = \hat{y}(\cdot) - y(\cdot)$ between the system output [supposed to be ruled by (3)] and the NOE

model (obtained after training)

$$\begin{cases} \text{System: } y(t+1) = g_y(\varphi), \varphi = [y(t), u(t), u(t-1)] \\ \text{NOE: } \hat{y}(t+1) = g_{\hat{y}}(\hat{\varphi}), \hat{\varphi} = [\hat{y}(t), u(t), u(t-1)] \end{cases} \quad (4)$$

with the same initial conditions $\hat{y}(0) = y(0) = y^o$. The evolution over time of the error can be obtained by considering the Taylor expansion around the true regressor vector φ of the NOE model

$$\hat{y}(t+1) = g_{\hat{y}}(\varphi) + g'_{\hat{y}}(\varphi)(\hat{\varphi} - \varphi) + O \quad (5)$$

and by subtracting the equation ruling the system, thus obtaining

$$\hat{y}(t+1) - y(t+1) = (g_{\hat{y}}(\varphi) - g_y(\varphi)) + g'_{\hat{y}}(\varphi)(\hat{\varphi} - \varphi) + O. \quad (6)$$

Let k be the maximum time lag associated with the output, and $\nabla g_{\tau} = \partial g / \partial \hat{y}(t - \tau)$ the gradient w.r.t. the time lagged prediction $\hat{y}(t - \tau)$. The vector $(\varphi - \hat{\varphi})$ can be interpreted as the error vector $(\varphi - \hat{\varphi}) = [0, \dots, 0, (\hat{y}(t) - y(t)), \dots, (\hat{y}(t - k) - y(t - k))]^T$. Equation (6) becomes

$$\begin{aligned} e(t+1) = & \nabla g_0 e(t) + \nabla g_1 e(t-1) + \dots + \nabla g_k e(t-k) \\ & + (g_{\hat{y}}(\varphi) - g_y(\varphi)) + O \end{aligned} \quad (7)$$

which represents the error dynamic over time. If we assume a small discrepancy between the real model and the approximated one, i.e., each component of the column vector $(\varphi - \hat{\varphi})$ is sufficiently small we can discard the O term. By indicating with F the forcing input $(g_{\hat{y}}(\varphi) - g_y(\varphi))$, (7) becomes the time discrete ordinary differential equation

$$e(t+1) = \nabla g_0 e(t) + \nabla g_1 e(t-1) + \dots + \nabla g_k e(t-k) + F. \quad (8)$$

It should be noted that the gradients are evaluated over $\hat{\varphi}$ and not over φ . Due to the universal approximation property, being $g_{\hat{y}}(\varphi)$ a neural NARX approximation of $g_y(\varphi)$, the term $(g_{\hat{y}}(\varphi) - g_y(\varphi))$ may become reasonably small, i.e., the obtained model possesses good prediction abilities when fed with data coming from the system.

The prediction error $e(\cdot)$ is composed of two contributions, deriving from the noise η and the systematic error ε due to the training inefficiency, the limited data set, and the model bias.

The NOE model is close to the process output, being stable at that instant of time, only if the eigenvalues associated with (8) lie within the unary circle.

If the system is error free (i.e., $\eta = 0$) and we excite system and model with a step function, the error tends to zero only if the eigenvalues are within the unary circle at the steady state. We experimentally observed that there is an error at the steady state even when eigenvalues lie within the unary circle. In our opinion, this is due to the presence of a nonnull forcing term $F = \bar{F}$, namely, the NOE model has a nonnull prediction error even when it receives the true regressor vector. In such a case, the error at the steady state satisfies the relationship

$$e_{ss} = \bar{F} \left(1 - \sum_{\tau=0}^k \nabla g_{\tau} \right)^{-1}. \quad (9)$$

We note that the eigenvalues associated with (8) depend on the working point along the trajectory and, with respect to a

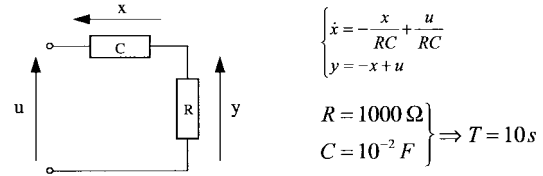


Fig. 2. Linear system.

specific instant of time, some of them could lie outside the unary circle. However, several experiments have shown that the presence of some instants of time in which the eigenvalues lie outside the unary circle does not significantly affect the error dynamic and, consequently, the final performance.

The fraction of points of the validation set having at least one eigenvalue outside the unary circle over the total number of points can be used as a confidence measure for the stability of the NOE model on new data. This ratio can be used as a criterion to assess the local stability of the obtained model. The smaller the ratio the smaller the probability that the NOE neural network will become unstable during its operational evolution. In the same direction, we can assert that the model behaves well during the operating phase if the eigenvalues are reasonably far, according to some metric, from the unary circumference both on training and validation data.

V. EXPERIMENTAL RESULTS

The following experiments are representative of a large set of experiments carried out to compare NOE models with virtual neurons ones. The experiments are parameterized in the regressor vectors and the number of hidden units. We will indicate with $\hat{y}(t - \tau)$ the predicted output delayed by τ instants of time, with $\bar{x}_i(t - \tau)$ and $u_i(t - \tau)$ the i th virtual τ -delayed state and input, respectively (when there is only one state or input the i index will be omitted). We will then apply the stability criterion to NOE models to test their local stability. Training was accomplished by considering the highly efficient Levenberg–Marquardt (LM), DFP, and BFGS algorithms [13], corrected to deal with the recurrent features of the model. We implemented, for the NOE networks, the teacher forcing modality [2] to provide a nice initialization of the regressors. The provided MSE and plots are those obtained with the best algorithm. It should be outlined that we did not experience any significant difference in performance among the algorithms (LM was slightly inferior in performance but it is computationally less intensive).

A test set was introduced to monitor, over training time, the model performance, hence limiting undesired overtraining effects caused by overdimensioned model.

A. Linear Model

The first experiment refers to the linear circuit depicted in Fig. 2 characterized by a capacitor of $C = 0.01F$ and a resistor of $R = 1000 \Omega$ (the constant of time T is hence 10 s). Training ($N = 500$), test ($N = 500$), and validation ($N = 200$) data have been obtained by applying step signals with uniformly extracted amplitudes and time

TABLE I
PERFORMANCES IN VALIDATION AND THE REGRESSOR VECTORS FOR THE CONSIDERED STRUCTURES

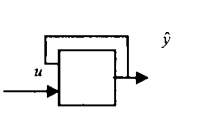
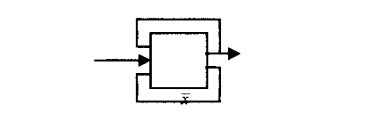
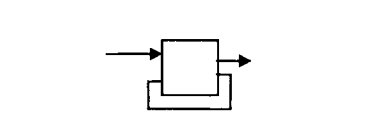
								MSE $\times 10^{-3}$
$\hat{y}(t-1)$ $\hat{y}(t-2)$	$\hat{y}(t-1)$	$\hat{y}(t-1)$ $\bar{x}(t-1)$	$\hat{y}(t-1)$ $\bar{x}(t-1)$ $\bar{x}(t-2)$	$\hat{y}(t-1)$ $\bar{x}_1(t-1)$ $\bar{x}_2(t-1)$	$\bar{x}(t-1)$	$\bar{x}(t-1)$ $\bar{x}(t-2)$	$\bar{x}_1(t-1)$ $\bar{x}_2(t-1)$	
2.514	2.505	2.505	2.505	2.506	13.657	2.863	3.186	$u(t)$ $u(t-1)$

TABLE II
MSE OF THE BEST NEURAL NETWORK

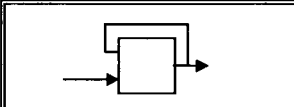
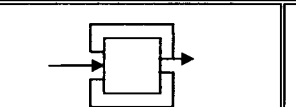
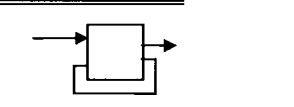
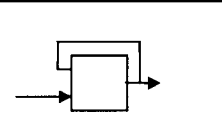
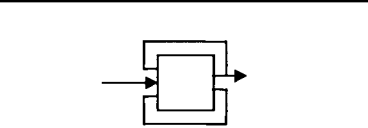
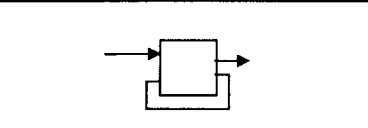
MSE			
Training	$1.598 \cdot 10^{-9}$	$2.595 \cdot 10^{-4}$	$5.407 \cdot 10^{-4}$
Test	$1.360 \cdot 10^{-9}$	$2.746 \cdot 10^{-4}$	$5.058 \cdot 10^{-4}$
Validation	$2.505 \cdot 10^{-3}$	$2.505 \cdot 10^{-3}$	$2.863 \cdot 10^{-3}$

TABLE III
PERFORMANCES IN VALIDATION AND THE REGRESSOR VECTORS FOR THE CONSIDERED STRUCTURES

								MSE $\times 10^{11}$
$\hat{y}(t-1)$ $\hat{y}(t-2)$	$\hat{y}(t-1)$	$\hat{y}(t-1)$ $\bar{x}(t-1)$	$\hat{y}(t-1)$ $\bar{x}(t-1)$ $\bar{x}(t-2)$	$\hat{y}(t-1)$ $\bar{x}_1(t-1)$ $\bar{x}_2(t-1)$	$\bar{x}(t-1)$	$\bar{x}(t-1)$ $\bar{x}(t-2)$	$\bar{x}_1(t-1)$ $\bar{x}_2(t-1)$	
0.164	0.797	0.253	1.722	0.326	230.034	8.917	13.947	$u_1(t)$ $u_2(t)$
0.908	0.527	0.431	0.482	1.098	5.624	12.753	22.311	$u_1(t)$ $u_2(t)$ $u_2(t-1)$

durations. MSE's in validation are given in Table I, MSE's in training, test, and validation in Table II. In Table II (and Table III), each column refers to a different neural structure grading, left to right, from NOE to pure virtual states neural structures. In particular, the first column refers to a NOE model, the second to a NOE/virtual state mixed structure, the third to a pure virtual neurons structure. The second row characterizes the state components of the regressor vector while the last column completes with the inputs. The best models have been identified with a grey shaded cell in the table. With respect to Table I, we note that the NOE model is slightly better than the virtual states one but the difference is neglectable. All models approximate the process with high accuracy and are practically indistinguishable. Table II shows

that virtual neurons structures have more difficulties in the training phase compared to NOE networks despite the use of high sophisticated training algorithms. We think this is due to the difficulty of the training algorithm to converge to a good minimum because of the presence of loops (which introduce a dynamic not directly controllable by the training phase). If the state variables were measurable then the training algorithm would have been applied to a modified error function comprising the error between the real states and the ones estimated by the neural network. We should appreciate the fact that in the linear case the pure virtual states model has no feedbacks on the measured output and, despite that, the virtual states have been configured by the training algorithm to exactly mimic the system.

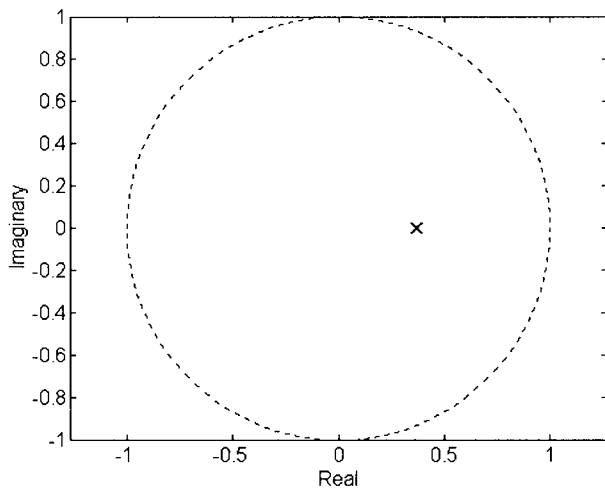


Fig. 3. Pole evolution over validation time for the NOE model in the linear system case.

We then studied the stability of the best NOE model characterized by five hidden units and a single feedback on the output. According to the criterion given in Section IV we computed the evolution of the pole over the validation time; Fig. 3 shows such evolution in the I - R plane. Note that there is only one real pole (identified with an X): the NOE model during the operational evolution keeps the pole constant at a value smaller than 0.5 for each instant of time. It is also interesting to observe that such a pole does not coincide with the system one (whose value is $1/T = 0.1$).

The pole has been evaluated by taking into account the estimated regressor vector instead of the real one in computing the gradients; we verified that there was no difference between the poles, thus proving that the obtained model is also a good predictor.

B. Nonlinear Model: Drum-Type Boiler

To test performances on a nonlinear system we considered a power plant driven by a drum-type boiler. Let P be the steam pressure within the boiler (supposed to be uniform), P_{sh} the steam pressure at the turbine inlet, Q_{rad} the radiating power (which is nonlinearly related to the fuel flow W_c via lookup tables), and A_v the aperture of the turbine valve (0 means that no steam flow inputs the steam turbine, 1 that the valve is fully open). The physically-based discretized equations, which describes the boiler in a simplified (but quite accurate) way, are

$$\begin{cases} P(t+1) = P(t) + b\sqrt{P(t)(P(t) - P_{sh}(t))} \\ \quad \cdot (c + dP(t)) + eQ_{rad}(W_c(t)) \\ P_{sh}(t+1) = P_{sh}(t) + f\sqrt{P(t)(P(t) - P_{sh}(t))} \\ \quad + gA_v(t)P_{sh}(t) \end{cases} \quad (10)$$

where b , c , d , e , f , and g are constants which characterize the specific plant; A_v and W_c are the external input variables, the pressure P_{sh} is measurable, and the pressure P is nonmeasurable.

The process is highly nonlinear. The output P_{sh} is characterized by two main dynamics: the first one, associated with A_v , is very fast (few seconds), while the second one, related

to W_c , is quite slow (several minutes). We know, *a priori*, that by linearization of (10) the linear transfer function has a zero associated with the opening of the turbine valve. For its nature (10) can be easily expressed as (3).

The data extraction phase, necessary to generate Z^N , is a critical step. In fact, if we randomly excite the inputs, we end with unrealistic behaviors for the plant and the pressures tends to infinity (e.g., this happens by giving maximum fuel for long time while keeping close the turbine valve). The problem was overcome by slightly correlating the inputs so as to constitute feasible input profiles.

Differently from the linear case, which was reasonably simple, it is required a high number of data to configure the nonlinear models. This immediately means that we need a large training set and hence, long training time. A data decimation phase is then generally considered to limit the computational complexity of the training procedure. The frequency spectrum of the input and the output signals was first computed by applying a FFT: the decimation frequency was taken to be five times the fastest relevant dynamic of the process. From the resulting set we extracted 2500 data for training, 2500 for testing, and 2500 for validation. Validation on these data provides a MSE close to the test set and hence it has been not be considered in the following. As a critic validation experiment we present an input sequence composed by independent steps on W_c and A_v : these signals have not been generated during training where inputs are correlated to grant feasibility. This input profile could be the typical signal that an operator gives to the plant during a manual control. With this validation we wish to stress the models so as to emphasize the limits of the approximation accuracy.

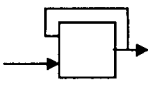
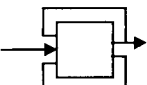
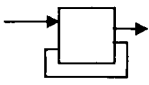
We experimented that a single layer (5–7 hidden units) was sufficient for the envisioned application, a second layer only increases the computational complexity of the training procedure without benefits in accuracy.

As with the linear case, we considered different model structures. Here, the variable \hat{y} refers to the measurable pressure P_{sh} , the nonmeasurable state \bar{x} is associated with the pressure P , $u_1 = W_c$ and $u_2 = A_v$. Tables III and IV introduce the performances of the different models. In this case we note that the NOE model performs significantly better than the pure virtual states one: the application is more complex and the training algorithm is not efficient enough to identifying a good setting for the weights.

Fig. 4 presents the validation error $y - \hat{y}$ over the considered validation set for the NOE neural network. As we can see it seems that there is large error around sample 1500, in reality, it corresponds to a 2.5% relative error (there, the pressure reaches 1.9×10^7 Pa). Anyway, such a high value for the pressure is a limit case also for the system which should work in correspondence with smaller values. Fig. 5 shows the correspondent validation error for the pure virtual neurons neural network; the hybrid models performance can be placed between the two. It is interesting to note that the error is more distributed but, conversely, it is higher.

The evolution of the two poles of the best NOE in validation is given in Fig. 6(a) for the I - R plane while the punctual evolution of their values over validation time is given in

TABLE IV
MSE OF THE BEST NEURAL NETWORK

MSE $\times 10^{11}$			
Training	0.093	0.084	6.563
Test	0.082	0.130	16.909
Validation	0.164	0.253	5.624

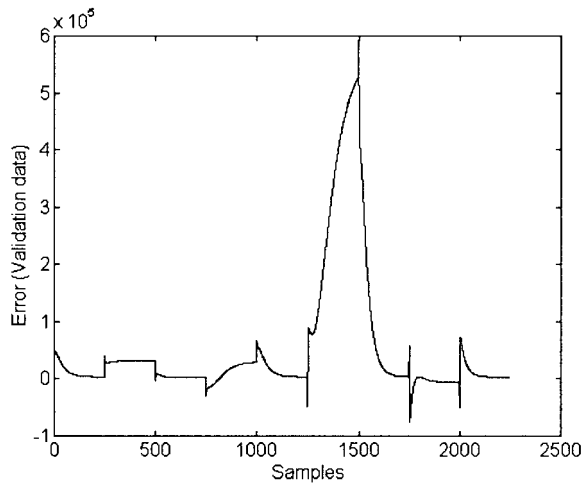


Fig. 4. Validation error for the NOE neural network.

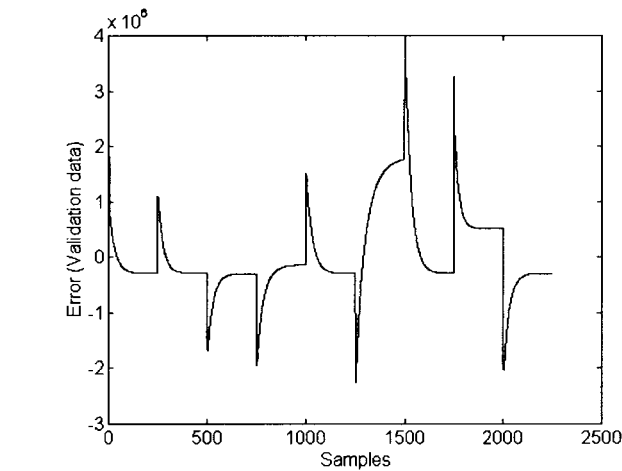
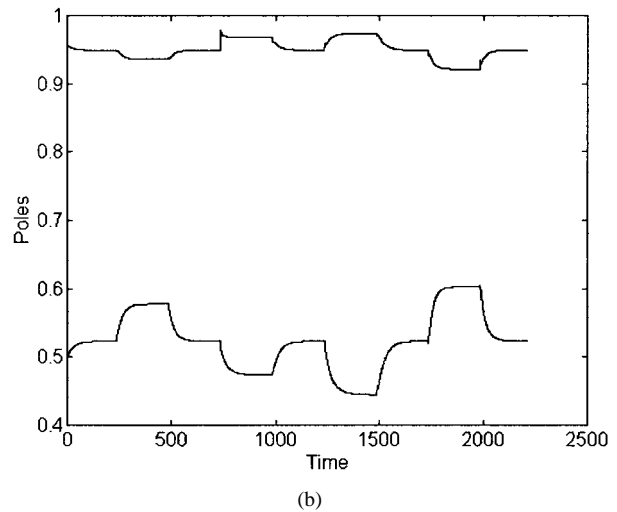
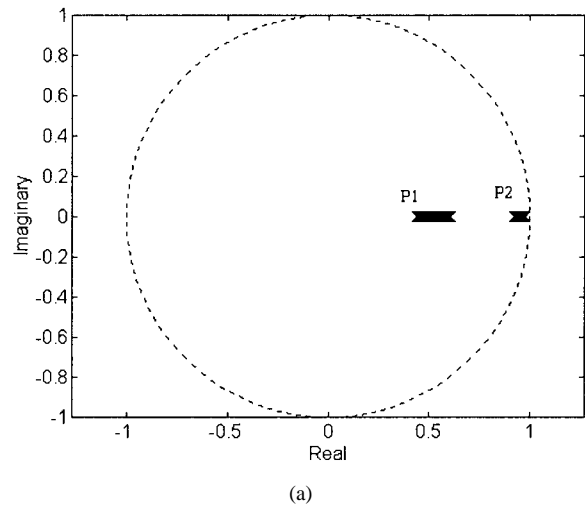


Fig. 5. Validation error for the virtual state neural network.

Fig. 6. (a) Poles' evolution over time for the best NOE model in the $I-R$ plane and (b) in the poles-time plane.

Fig. 6(b). The two poles P1 and P2 are distinct, real and, differently from the linear case, evolve during time so as to generate a small interval. We observe that P2 is close to the unitary circle but it never reaches it [see Fig. 6(b)] and that when P1 increases, P2 decreases. We experienced in other neural models that there it might exist a set of instants of time for which a pole is outside the unitary circle but if the ratio between such points and N is small the neural network will be stable. Local instability implies instability when the ratio increases, as happens when the weights have been not configured properly.

In general, only when the performances of the model are satisfactory as well as the distribution of its poles, we can state that the model can be reasonably considered to be a good approximation of the system.

VI. CONCLUSIONS

The paper deals with neural modeling of dynamic systems when some of the system state variables are not measurable. If *a priori* information about the structure of the equations ruling the system is available, topologies characterized either by the input-output or the space-state representations can be

constructed by introducing virtual states to provide enough degrees of freedom to the model. Even if such models are effective in many applications, training inefficiency may impair their performances and further research effort should be carried out on this front. The paper shows that an effective solution to this approximation problem can be obtained by considering NOE models which experimentally show, once more, to be extremely interesting neural network structure. The model accuracy and its stability need then to be verified to certify the quality of the approximating network; to this end we suggested a novel criterion to perform this task.

REFERENCES

- [1] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [2] *IEEE Trans. Neural Networks*, vol. 5, Mar. 1994.
- [3] B. J. Sheu and J. Choi, *Neural Information Processing and VLSI*. Boston, MA: Kluwer, 1995.
- [4] R. L. Watrous, "Speaker normalization and adaptation using second-order connectionist networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 21–30, Jan. 1993.
- [5] N. J. Bock, E. Porada, M. Beaulieu, and T. A. Eftimov, "Automatic calibration of a fiber-optic strain sensor using a self-learning system," *IEEE Trans. Instrum. Meas.*, vol. 43, pp. 341–346, Apr. 1994.
- [6] K. Aminian, P. Robert, E. Jequier, and Y. Schutz, "Estimation of speed and incline of walking using neural network," *IEEE Trans. Instrum. Meas.*, vol. 44, pp. 743–746, June 1995.
- [7] R. Linggard, D. J. Myers, and C. Nightingale, Eds., *Neural Networks for Vision, Speech and Natural Language*. London, U.K.: Chapman & Hall, 1992.
- [8] C. Alippi and V. Piuri, "Experimental neural networks for prediction and identification," *IEEE Trans. Instrum. Meas.*, vol. 45, Apr. 1996.
- [9] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.
- [10] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnez, and G. Dreyfus, "Training recurrent neural networks: Why and how? An illustration in dynamical process modeling," *IEEE Trans. Neural Networks*, vol. 5, Mar. 1994.
- [11] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Contr. Signals Syst.*, vol. 2, 1989.
- [12] C. Alippi, A. Cori, V. Piuri, and F. Pretolani, "Monitoring and real time simulation of power plants: A neural based environment," in *Proc. IEEE-IMTC96*, Brussels, Belgium, June 4–6, 1996.
- [13] W. Press, S. Teukolosky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1992.



Cesare Alippi (M'97–SM'99) received the Dr.Eng. degree (summa cum laude) in electronic engineering in 1990 and the Ph.D. degree in computer engineering in 1995, both from the Politecnico di Milano, Milano, Italy.

He is an Associate Professor of information processing systems with the Politecnico di Milano.

His research interests include neural networks (mathematics, learning, implementation, applications), genetic algorithms, nonlinear signal and image processing, and VLIW architectures.



Vincenzo Piuri (S'84–M'86–SM'96) received the Dr.Eng. degree in electronic engineering in 1984 and the Ph.D. degree in information engineering in 1989 from the Politecnico di Milano, Milan, Italy. He is an Associate Professor in operating systems at the Politecnico di Milano.

His research interests include distributed and parallel computing systems, computer arithmetic, neural networks, and fault tolerance.

Dr. Piuri is a member of IMACS, INNS, and AEI.