Defining and applying prediction performance metrics on a recurrent NARX time series model.

Ryad Zemouri, Rafael Gouriveau, Noureddine Zerhouni

Defining and applying prediction performance metrics on a recurrent NARX time series model

Ryad Zemouri*, Rafael Gouriveau**, Noureddine Zerhouni**

* Laboratoire d'Automatique du CNAM,
75003 Paris, France, (e-mail: ryad.zemouri@cnam.fr)
** FEMTO-ST Institute, UMR CNRS 6174 - UFC / ENSMM / UTBM,
Automatic Control and Micro-Mechatronic Systems Department
25000 Besançon, France (e-mail: rgourive@ens2m.fr)

ABSTRACT.
Nonlinear autoregressive moving average with exogenous inputs (NARMAX) models have been successfully demonstrated for modeling the input-output behavior of many complex systems. This paper deals with the proposition of a scheme to provide time series prediction. The approach is based on a recurrent NARX model obtained by linear combination of a recurrent neural network (RNN) output and the real data output. Some prediction metrics are also proposed to assess the quality of predictions. This metrics enable to compare different prediction schemes and provide an objective way to measure how changes in training or prediction model (Neural network architecture) affect the quality of predictions. Results show that the proposed NARX approach consistently outperforms the prediction obtained by the RNN neural network.

KEYWORDS: *Time series prediction, recurrent radial basis function network, NARX models, prediction performance metrics.*

1. Introduction

Time series are pervasive in data acquisition hence a significant amount of work has been done in the realm of time series analysis, modeling, and prediction to support analysis and interpretation of such data [15,4,6,11,13,16,20,21,23,28,29,32,44,1]. The commonly encountered models of time series include auto-regressive (AR) models [11,32,1], recurrent neural networks [4,11,16,21,28,29,50,1,7] and fuzzy rule-based models [4,13,20,21,44,40]. In many cases time series could be difficult to model due to complex system dynamics; in such cases it is useful to apply non-linear prediction architectures such as neural networks in order to improve prediction performance [23,28]. It is also worth noting that with the growing diversity of constructs of computational intelligence, we are provided with a significant number of architectures of neural networks, fuzzy systems and evolutionary optimization schemes that could be exploited to enhance the performance of prediction of non-linear time series.

In one-step-ahead prediction tasks, Artificial Neural Networks (ANN) models are required to estimate the next sample value of a time series, without feeding back it to the model's input regressor [30]. In other words, the input regressor contains only actual sample points of the time series. If the user is interested in a longer prediction horizon, a procedure known as multi-step-ahead or long-term prediction, the model's output should be fed back to the input regressor for a fixed but finite number of time steps [41]. In this case, the components of the input regressor, previously composed of actual sample points of the time series, are gradually replaced by previously predicted values.

If the prediction horizon tends to infinity, from some time in the future the input regressor will start to be composed only of estimated values of the time series [30]. In this case, the multi-step-ahead prediction task becomes a dynamic modeling task, in which the ANN model acts as an autonomous system, trying to recursively emulate the dynamic behavior of the system that generated the nonlinear time series [17,18]. Multi-step-ahead prediction and dynamic modeling are much more complex to deal with than one-step-ahead prediction, and it is believed that these are complex tasks in which ANN models play an important role, in particular recurrent neural architectures [36].

Either using ANNs or others techniques to model time series, the prediction metric is a key issue to address. There is no general agreement as to an appropriate and acceptable set of metrics that can be employed in time series prediction [8,19]. Some metrics as the repeatability or the precision can be used to quantify the goodness of the prediction and can be used also to compare several techniques.

In this study, we develop a nonlinear time series prediction model based on a recurrent radial basis function network (RRBF) [47]. To improve the accuracy of the prediction model, an AutoRegressive with eXogenous inputs model (ARX) is used to correct the predictions error. Moreover, since performances of the ANN are partially depending on an initial randomly parameterization of its structure, criteria are proposed to quantify the precision, the accuracy, and the reproducibility of predictions. It enables to a priori estimate the usefulness of a neural network structure.

The remainder of the paper is organized as follows. A literature review highlighting the main trends, presenting main categories of fuzzy models and ANN in section 2. Section 3 describes the architectural of the proposed prediction model. Let call it a NARX model (Nonlinear ARX). The section 4 deals with the prediction metrics. Section 5 and 6 presents the experiments data and the prediction results.

2. Literature overview

In this section, a concise overview of some of the existing models of computational intelligence (CI) used in time series analysis and prediction is presented [15]. As a reference, we will be alluding to standard autoregressive linear models (AR models) as extensively discussed in statistics and signal processing [6,11,32]. Let us recall that the standard AR(q) model of order ''q'' is described by the following expression:

$$x(t+1) = \sum_{i=1}^{q} \varphi_i x(t) + \varepsilon(t+1) \qquad (1)$$

where φ_i for $i = 1,2,.....q$ are the parameters of the model, $x(t)$ is the value at time instant t, and ε is the residual error. Typically the residual error is assumed to be zero-mean white noise with a normal distribution. The parameters of the AR(q) model can be determined by finding the auto-correlation sequence ϕ_L for lag $L = -q,......,q$ and solving the Yule–Walker system of equations [6]:

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ . \\ . \\ \phi_q \end{bmatrix} = \begin{bmatrix} \phi_0 & \phi_{-1} & . & . & \phi_{-q+1} \\ \phi_1 & \phi_0 & . & . & \phi_{-q+2} \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi_{q-1} & \phi_{q-2} & . & . & \phi_0 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ . \\ . \\ \varphi_q \end{bmatrix} \qquad (2)$$

Fuzzy rule-based models have been found in various areas of application including, control, reliability modeling, planning and scheduling, economics forecasting, decision making, pattern recognition, etc., as well as time series prediction. [31] provides a survey on the way of building such models whereas [22] gives an interesting overview of applications of fuzzy systems in industrial engineering.

Within all fuzzy rule-based approaches, Takagi-Sugeno-Kang (TSK) models are the dominant category of models used. A TSK model is a first order Takagi Sugeno (TS) Fuzzy Inference System (FIS). In a few words, a first-order TS model can be seen as a multi-model structure consisting of linear models that are not necessarily independent [3]. It is based on the fuzzy decomposition of the input space. For each part of the state space, a fuzzy rule is formed, the global output been a combination of the whole rules. For prediction problems, given the inherent dynamics of time series one has to cope up with, a TS FIS comes here in the following form [21]:

IF $x(t)$ is $A_1(x(t))$ AND....

$$....AND\ x(t-M)\ is\ A_M(x(t-M)) \qquad (3)$$

THEN $x(t+1) = a_0 + a_1 x(t) + ... + a_M x(t-M)$

The condition part of the above rule includes a finite sequence of values (inputs) of the time series of some horizon M, namely $x(t)$, $x(t-1),.....,$ $x(n-M)$ which invoke (activate) some fuzzy sets defined in the space of amplitude of the time series. A_i are fuzzy membership functions capturing information granules forming the condition part of the ith rule. The conclusion (consequent) of the rule is a linear function of the inputs where the linear relationship is governed by its parameters $a_0, a_1,...., a_M \in \Re$. In this sense the condition part of the rule captures the ''*history*'' of the time series. Being more specific, the rule is concerned only with the dynamic of the time series being pertinent to the condition of the rule.

The TS scheme of the form given above has been under intensive investigation in fuzzy modeling. One can refer here to some studies concerning architectural enhancements of the rules as well as coming up with various ways of designing (optimizing) such models. For instance, in [4,13] discussed were ideas of making the conclusion part of the rules non-linear by using neural networks. Some other studies were focused on the use of such rule-based systems in prediction problems [4,13,21]. Radial basis functions are also used for prediction of time series [16]. Some research has also been completed in fuzzy clustering of time series [25,26]. Authors in [4] utilized a locally recurrent neural network in the consequent of the rules of a TSK–FIS (with Fuzzy Inference System) model for predicting wind speeds. They employ an IIR-MLP (Infinite Impulse Response Multi-Layer Perceptron) [4] as the consequent of each rule and considered Gaussian fuzzy sets as the antecedent of each rule. A common fuzzy prediction network used is the ANFIS described in [21]. A feed-forward neural network trained via a hybrid least squares estimate (LSE) and gradient descent is used as the consequent of the fuzzy

rules. They showed slightly better performance for ANFIS when being compared with a standard neural network. The ANFIS performs better than the standard AR model used in time series prediction. Authors in [13] use NARMAX (Non-linear AutoRegressive Moving Average with eXogenous inputs) as the consequent of the fuzzy rules. Feed-forward and recurrent neural network approaches for a TSK–FIS models for prediction are explored. They report better performance using neural networks as the consequent function in the TSK–FIS model.

The radial basis function neural network has been applied to time series prediction [16]. The authors explore several different radial basis functions applied to the prediction of several time series. The choice of basis function and basis function parameters significantly impacted the prediction performance [16]. Several fuzzy recurrent neural networks have also been applied to time series prediction [50,1,7]. A recurrent fuzzy neural network (RFNN) is developed in [50] using ordered derivatives in the training procedure. In [1] time series prediction is accomplished for a time series of linguistic concepts called a fuzzy time series and trained via genetic algorithms. The experiments show an improvement in the prediction of linguistic time series using a RFNN compared with other fuzzy time series prediction algorithms. Authors in [7] develop an improved circular back-propagation (ICBP) network where the weights are +1 or -1 to improve the predictive generalization of the network. Experiments conducted in [7] show that there are improvements in one-step and p-step prediction performance for DLS-ICBP on several synthetic data sets. The study reported in [26] comes with an interesting conclusion that clustering of raw streaming time series data could be of limited relevance since cluster centers are often sinusoidal in shape. Authors suggest several ways of countering meaningless clustering results such as having a large number of cluster centers and completing post-processing to find fewer yet more meaningful clusters. A survey of the different time series clustering methods are described in [25] including several different clustering algorithms and distance measures.

3. Prediction model architecture

3.1. *The Radial Basis Function network*

In practice, Multi-Layer Perceptron (MLP) have been found to perform poorly in a number of ways, slow convergence of weights and difficulty in modeling differential responses. Radial basis function (RBF) neural networks are able to surpass the MLPs as they are simpler in structure and have the ability to model any nonlinear function in a straightforward way. On one hand MLP networks are global approximators with nonlinear input-

output mappings and the representation of knowledge is distributed throughout the network. On the other hand, RBF networks are local approximators with nonlinear input-output mapping. The knowledge representation in this case is localized. Thus, RBF network are able to learn faster and suffer less from interference, as compared to MLPs.

A key feature of RBF networks is that the output layer is merely a linear combination of the hidden layer signals, there being only one hidden layer. Therefore, RBF networks allow for a much simpler weight updating procedure and subsequently open up greater possibilities for stability proofs and network robustness in that the network can be described readily by a set of nonlinear equations.

The RBF network is commonly used for the purpose of modeling uncertain and nonlinear functions. Utilizing RBF networks for modeling purposes could be seen as an approximation problem in a high-dimensional space. Consider the RBF network, which can be seen as a two-layer processing structure, as shown in Figure 1.a. The hidden layer consists of an array of computing units (i.e., $\phi_1, \phi_2, ... \phi_k$). These hidden units provide a set of basis functions of the input vectors (i.e., $x_1, x_2, ... x_d$) as they are expanded into the higher dimension hidden-unit space. The mapping from the input vectors to the outputs of the hidden units is nonlinear, whereas the mapping from the hidden units to the final output of the RBF network is linear.

The general mapping function of the RBF network can be represented by

$$y_p = \mathbf{g}[\boldsymbol{x}, \boldsymbol{w}] = \sum_{j=1}^{k} w_{pj} \phi_j(\boldsymbol{x}) \qquad (4)$$

where $\mathbf{x} = [x_1, x_2, ... x_d]^T$ is the input vector and ($w_{p1}, w_{p2}, ... w_{pk}$) is a set of the output weights. The commonly used RBF $\phi_j(.)$ is the Gaussian represented by

$$\phi_j(\boldsymbol{x}) = \exp(-\frac{\left\| \mathbf{x} - \mathbf{c}_j \right\|^2}{2\boldsymbol{\sigma}_j^2}) \qquad (5)$$

Each RBF contains a parameter vector called a centre ($\mathbf{c}_j$), and calculates a squared distance between the centre and the input vector ($\boldsymbol{x}$). The result is then divided by the width ($\boldsymbol{\sigma}_j$) and then passed through an exponential function.

Training a RBF with linear outputs is very fast and is accomplished through two stages:

– The first stage is unsupervised and accomplished by obtaining cluster centers of the training set input vectors. A popular method for that purpose is the *k-means* clustering,

– the second stage consists in solving a set of linear equations, the solution of which can be obtained by a matrix inversion technique such as singular value decomposition or least squares method.
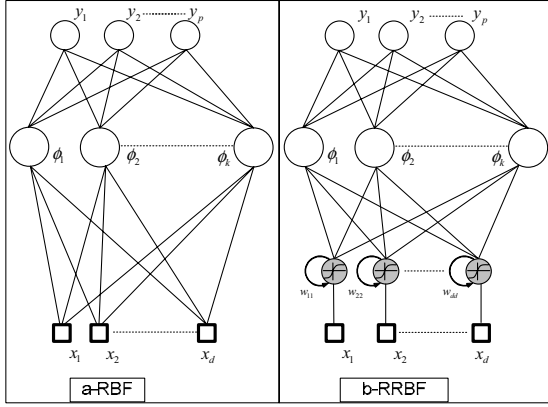
Figure 1. *Radial Basis Function and Recurrent Radial Basis Function networks*

3.2. *The Recurrent Radial Basis Function network*

The Recurrent RBF neural network considers time as an internal representation (Figure 1.b). The dynamic aspect is obtained by the use of an additional self-connection to the input neurons with a sigmoid activation function. The RRBF network can thus take into account a certain past of the input signal.

Every neuron of the input layer gives a summation at the instant t between its input x_i and its previous output weighted by a self-connection w_{ii}. The output of its activation function is:

$$a_i(t) = w_{ii}\xi_i(t-1) + x_i(t) \,,\ \xi_i(t) = f(a_i(t)) \tag{6}$$

where $a_i(t)$ and $\xi_i(t)$ represent respectively the neuron activation and its output at the instant t, and f is the sigmoid activation function defined as:

$$f(x) = \frac{(1-\exp(-\kappa x))}{(1+\exp(-\kappa x))} \tag{7}$$

The RRBF network was described on several publications [46,47,37,14,10,33]. A complete study of the looped neuron can be found on [12,5]. The reader can find in a detailed mathematical demonstration of the dynamic behavior of the looped neuron. To have the longest

memory, the self connection weight w_{ii} and the parameter κ of the sigmoid function must respect this relation: $\kappa w_{ii} = 2$. The Figure 2 represents a "*n-step*" prediction $\widehat{\Psi}(t+n)$ of $y(t)$ using previous observations $\mathbf{y} = [y(t), y(t-1), ..., y(t-m)]$:

$$\widehat{\Psi}(t+n) = \mathbf{g}[\mathbf{y},\mathbf{w}] = \sum_{j=1}^{k} w_j \phi_j(\mathbf{y}) \tag{8}$$
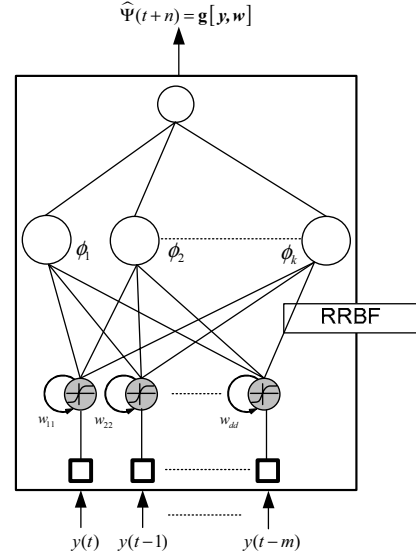
Figure 2. *"t+n" step-ahead-prediction obtained by the Neural Network*

3.3. *NARX model and optimal predictors*

The statistical approach for forecasting involves the construction of stochastic models to predict the value of an observation y_t using previous observations. A very general class of such models used for forecasting purpose is the **N**onlinear **A**uto**R**egressive **M**oving **A**verage with e**X**ogenous inputs (**NARMAX** models) [13,42,27,35] given by:

$$\begin{aligned} y(t) = \mathbf{F}[y(t-1)...y(t-n_y), e(t-1).... \\ ....e(t-n_e), x(t-1)...x(t-n_x)] + e(t) \end{aligned} \tag{9}$$

where y, e and x are output, noise and external input of the system model respectively. n_y, n_e and n_x are the maximum lags in the output, noise and input, respectively, and $\mathbf{F}$ is an unknown smooth function. It is assumed that $e(t)$ has zero mean, is independent and identically

distributed, is independent of past y and x, and has a finite variance σ^2.

The **NARMAX** models are nonlinear generalization of the well-known **ARX** models, which constitute a standard tool in linear black-box model identification. Several special cases of the general **NARMAX**(n_y, n_e, n_x) model which are frequently used are summarized here [13]:

- **NAR**(n_y) model (**N**onlinear **A**uto**R**egressive) :

$$y(t) = \mathbf{F}[y(t-1),,,,y(t-n_y)] + e(t) \qquad (10)$$

- **NARMA**(n_y, n_e) model (**N**onlinear **A**uto**R**egressive **M**oving **A**verage):

$$y(t) = \mathbf{F}[y(t-1)...y(t-n_y), e(t-1)....$$
$$......e(t-n_e)] + e(t) \qquad (11)$$

- **NARX**(n_y, n_x) model (**N**onlinear **A**uto**R**egressive with e**X**ogenous inputs):

$$y(t) = \mathbf{F}[y(t-1)...y(t-n_y), x(t-1).....$$
$$.....x(t-n_x)] + e(t) \qquad (12)$$

The **NARX** models can represent a wide variety of nonlinear dynamic behaviors and have been extensively used in various applications [35,2,34,30,45]. In a general way, a great equivalence exists between the **NARX** models and the recurrent neural network (**RNN**) [42].

For the prediction purposes, the neural network output prediction $\widehat{\Psi}(t+n)$ is always expressed by a residual prediction error ε as following (see Figure 3):

$$y(t+n) = \widehat{\Psi}(t+n) + \varepsilon(t+n) \qquad (13)$$

Considering the neural prediction $\widehat{\Psi}(t+n)$ obtained from the past input vector [$y(t)$, $y(t-1)$,...$y(t-m)$] (see Figure 2 and Figure 3), the real system output $y(t+n)$ is not available at the time instant t. So, the value of the residual error $\varepsilon(t+n)$ can't be calculated until the time instant $(t+n)$ where $y(t+n)$ is available. The equation (13) is then expressed by:

$$y(t+n) = \widehat{\Psi}(t+n) + \mathbf{f}(\varepsilon(t)) \qquad (14)$$

where $y(t+n)$ represent the ideal system prediction (without residual error), $\varepsilon(t)$ is the residual error calculated at the instant t and $\mathbf{f}$ is the function to find to estimate the prediction error $\varepsilon(t+n)$.
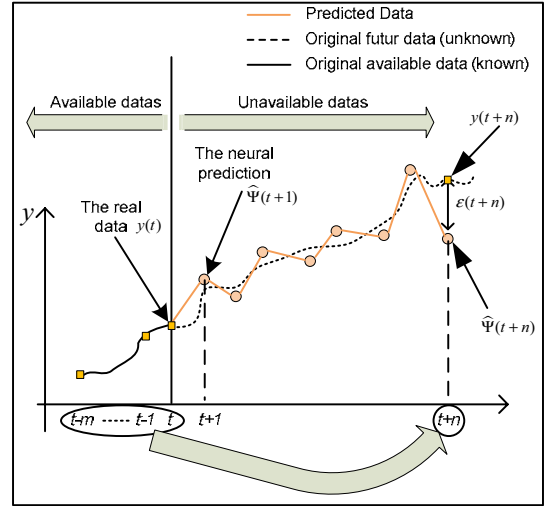
Figure 3. Residual error obtained at each prediction.

In our previous work, we have used a Proportional-Integral-Derivative controller to minimize and to estimate the prediction error $\varepsilon(t+n)$ [48]. This model is presented by the Figure 4 where the residual error $\varepsilon(t)$ is computed by a *PID* controller. The results presented in [48] reveal that the predictions obtained by this structure are better than the RRBF network. For all the tests, the best results are given by $K_i = 0$ (integrator parameter of the PID equal to zero).
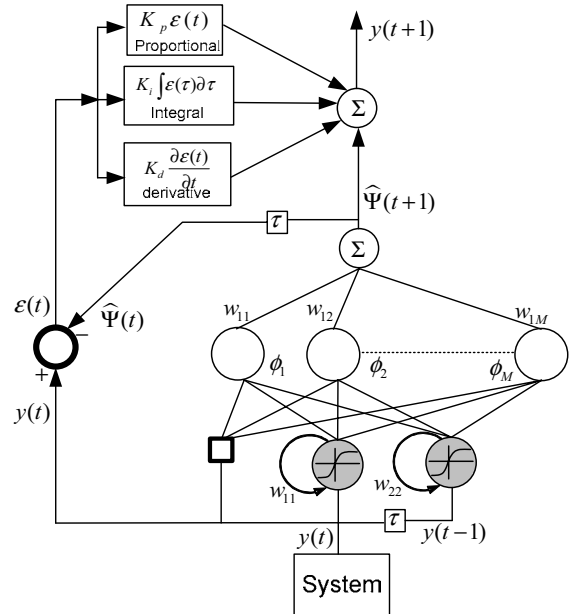
**Figure 4.** *The RRBF$_{Error}$ structure for time series prediction*

5

In this paper, we present an extension of the PID structure presented by the Figure 4. The prediction $\widehat{\Psi}(t+n)$ obtained by the RRBF network is combined with an AutoRegressive eXogenous model to process the residual prediction error. The whole prediction obtained model is described by the NARX system depicted in Figure 5. The several Predictions obtained through the RRBF network at different time instant ($\widehat{\Psi}(t), \widehat{\Psi}(t-1), \ldots, \widehat{\Psi}(t-m)$) are computed with the real available values ($y(t), y(t-1), \ldots, y(t-m)$). The "*n-step*" prediction $y(t+n)$ obtained by the whole model is then described by the following equation:

$$
\begin{aligned}
y(t+n) &= \mathbf{F}[\boldsymbol{y}, \widehat{\boldsymbol{\Psi}}] \\
&= \alpha \widehat{\Psi}(t+n) + \mathbf{f}(\varepsilon) \\
&= \alpha \widehat{\Psi}(t+n) + \sum_{i=0}^{i=m} a_i \widehat{\Psi}(t-i) + \sum_{i=0}^{i=m} b_i y(t-i)
\end{aligned}
\tag{15}
$$

The parameters (α, a_i, b_i) are calculated by linear regression (matrix inversion technique).
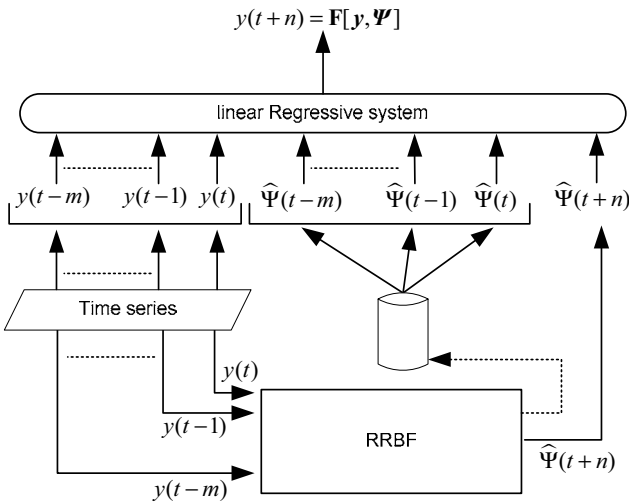
Figure 5. *"t+n" step-ahead-prediction obtained by the NARX model (the whole model)*

4. Performance metrics for neural network prediction

4.1. *Prediction evaluation*

In a general way, the choice of an error measure to compare prediction methods has been much discussed (see for example [9]). The prediction performance are used to be assessed using the Root Mean Square Error criterion

(RMSE) which is the most popular prediction error measure, the Mean Absolute Scaled Error (MASE) that, according to [19], is the more adequate way of comparing prediction accuracies, or the coefficient of determination (R2) which is a measure of how well future outcomes are likely to be predicted by the model. In any case, those error measures are only intended as summaries for the error distribution for a specific model. This distribution is usually expected to be a normal white noise in a forecasting problem, but it probably is not so in a complex problem like load forecasting. More over, this prediction metrics do not address the problem of quantifying the interest of a given model with regards to its sensibility to initial parameterization. This point is more widely discussed in this section and some metrics are proposed to face up with.

Learning is an important capability of neural networks. Learning rules are algorithms for finding suitable weights W and other network parameters. Learning of a neural network can be viewed as a nonlinear optimization problem in which the goal is to find a set of network parameters minimizing the cost function for given examples. Putting it in another way, learning is an optimization process that produces an output that is as close as possible to the desired output by adjusting network parameters. For the same learning set, the neural network structure or parameters can be different at each training run. This results from the random initialization of certain parameters of the training process.

The quality of the prediction can be completely different at each running of the training algorithm. This is most likely due to the fact that training a neural network involves minimizing an error function with a multitude of local minima. To test the prediction performances of a neural prediction model according to the prognostic metrics presented on this section, several running of the "training/test" process have to be done. Suppose that M represents the number of all the "training/test" running. For every running i of the training algorithm, a new value of the mean prediction error *E(i)* and the standard deviation *std(i)* are obtained for the *n* data of the test set as follows :

$$
E(i) = \frac{1}{n} \sum_{j=1}^{n} \left(\xi^i(j) - \zeta(j) \right)
\tag{16}
$$

$$
std(i) = \sqrt{\frac{1}{n} \sum_{j=1}^{n} \left(E(i) - \zeta(j) \right)^2}
\tag{17}
$$

where

$\xi^i(j)$ is the jth output obtained by the ith neural model

$\zeta(j)$ is the jth system output

The measures of the prognostic neural system performance are then processed on the variations of E(i)

6

and std(i). The different training and test running steps are presented as follows:

> *for i=1 to M*
> - *train the ANN on the training data set*
> - *test the ANN on the test data set*
> - *calculate the mean prediction error E(i) produced by the ith neural model on the test set*
> - *calculate the standard deviation std(i) produced by the ith neural model on the test set*
> *Next i*

On this basis, various performance metrics can be proposed.

4.2. *The timeliness*

The **timeliness** is given by the global mean of all the *M* values of *E(i)*:

$$\text{Timeliness} = \overline{E} = \frac{1}{M}\sum_{i=1}^{M} E(i) \qquad (18)$$

where *M* is the number of the training/test running, *E(i)* is the mean error of every running test *i* of the ith neural model obtained at the training step *i*.

The perfect score is **timeliness** = 0. For a small value of the **timeliness**, the probability to have a prediction close to the real value can be significant. On the contrary, if the **timeliness** value is important, the probability to have a wrong prediction is very high (Figure 6).
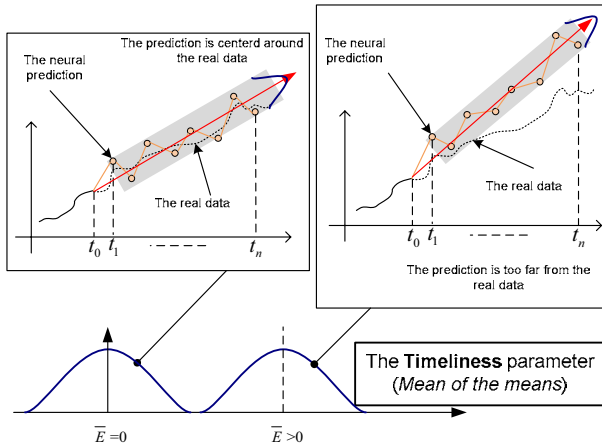

Figure 6. *an example of two different **Timeliness** values.*

4.3. *The Precision*

The **Precision** is given by the global mean of all the *M* values of std(i):

$$\text{Precision} = \overline{std} = \frac{1}{M}\sum_{i=1}^{M} std(i) \qquad (19)$$

where *std(i)* is the standard deviation of each running test *i* of the ith neural model obtained by the training step *i*.

The perfect score is **precision** = 0. For a small value of the **precision**, the probability to have predictions grouped together can be significant. On the contrary, if the **precision** value is important, the predictions are dispersed (Figure 7).
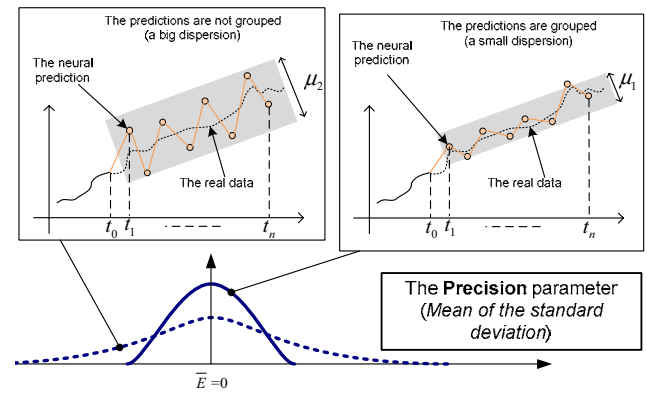

Figure 7. *an example of two different **Precision** values.*

4.4. *The repeatability*

The **Repeatability** is given by the standard deviation of both *E(i)* and *std(i)*. A simple way to calculate the repeatability parameter is :

$$\text{Repeatability} = \frac{\sigma(std) + \sigma(E)}{2} \qquad (20)$$

where $\sigma(std)$ and $\sigma(E)$ represent the standard deviation of the *M* values of respectively the *std(i)* and *E(i)* values :

$$\sigma(std) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\overline{std} - std(i)\right)^2} \qquad (21)$$

$$\sigma(E) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\overline{E} - E(i)\right)^2} \qquad (22)$$

The perfect score is **Repeatability** =0. This parameter indicates how close the different values of the E(i) and the

std(i) are grouped or clustered together. This parameter reveals the dispersion of E(i) and *std(i)* values. For small values of $\sigma(std)$ and $\sigma(E)$, it means that at each training/running time *i*, the neural model gives the same performances on the test set. The repeatability parameter reveals the random initialization influence of some learning parameters. The training process is completely repeatable for small values of the repeatability parameter. The structure of the neural model is always the same at each running of the training process.

4.5. *The Accuracy*

The **accuracy** is obtained from the three parameters and it gives a global appreciation of the prediction. A simple way to calculate the accuracy is:

$$\text{Accuracy} = \frac{1}{\text{Repeatability} + \text{Timeliness} + \text{Precision}} \quad (23)$$

If a neural model has a good Timeliness, Precision and is completely Repeatable, the prediction given by this neural model is very close to the real data. The prediction confidence is then very high. A big value of the accuracy parameter gives a great confidence of the prediction.

Figure 8 illustrates the three prediction metrics: the timeliness, the precision and the repeatability described by a radar graph. This figure reveals the quality of the prediction made by the neural network according to the value of the three metrics. Figure 9 gives the relation between the accuracy and the three metrics. One can see that the accuracy is very high for small values of the: timeliness, precision and repeatability. On the contrary, the accuracy is very low if, at least, one of the three metrics has a great value.
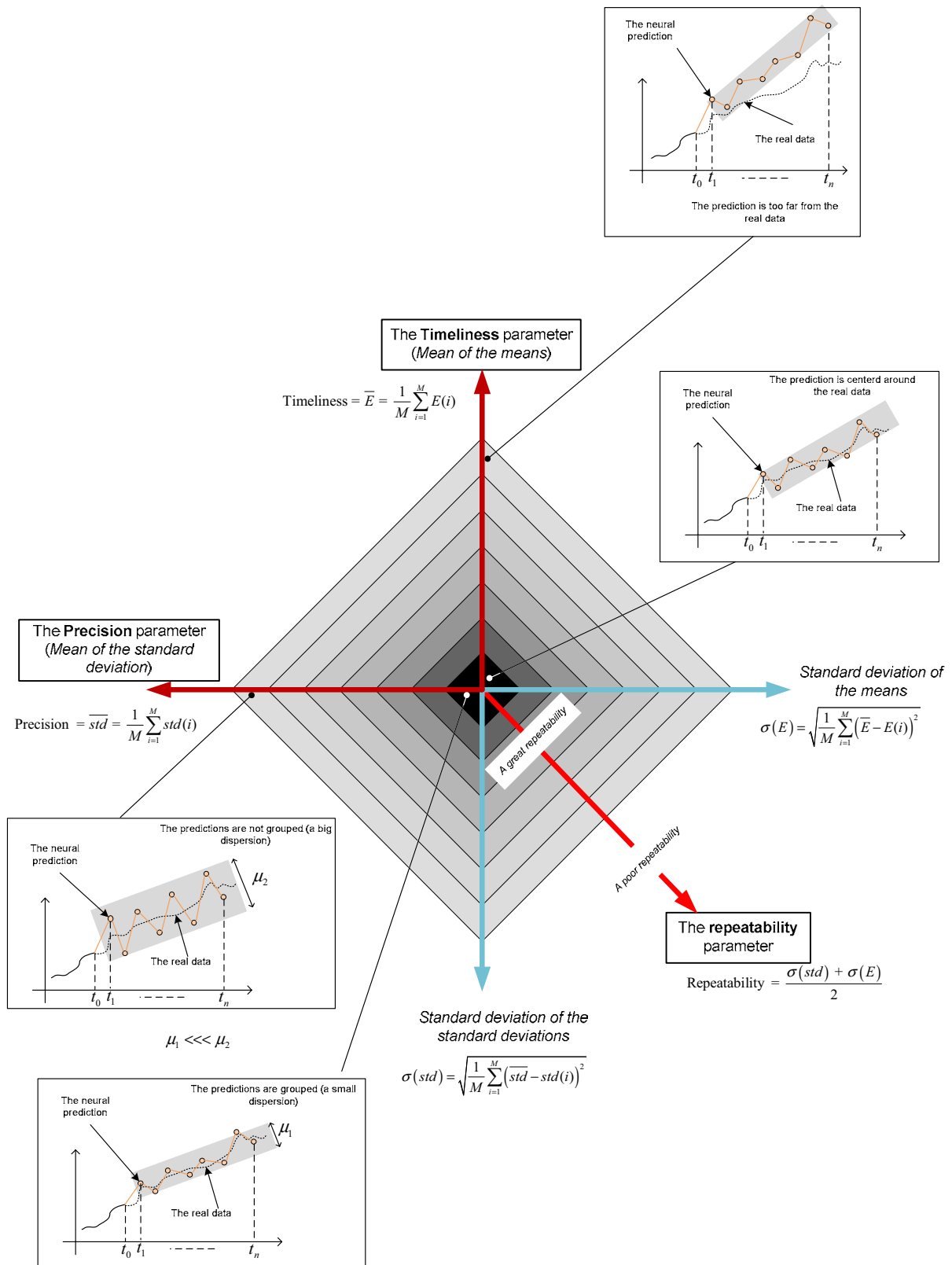
Figure 8. *Neural Network system performance measures*
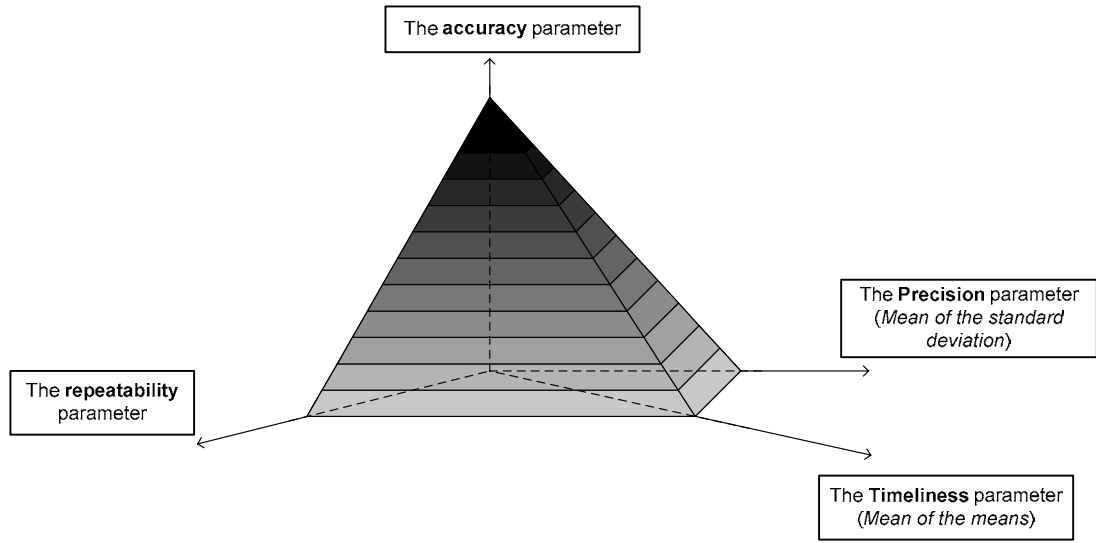
Figure 9. *The relation between accuracy and the three other metrics.*

5. Experiments & data benchmarks

Two experimental data sets have been used to test the prediction performances of the proposed structure with regards to the classical RRBF network. In the two cases, the aim of the predictions is to approximate a phenomenon by learning data gathered from the system.

For all the benchmarks, three sets were used: a **training** set, a **validation** set and a **test** set (Figure 10). The first set was used for the training parameters of the neural network (the k centers of the Gaussian nodes and the output weights). 50 samples have been used for training set and 50 samples for validation set. All data have been normalized by range [-1,+1]. Predictions steps were made from "*t+1*" to "*t+10*" by increments of 1 (in order to measure the quality of the long-term prediction). Predictions have been performed as described below:

$$y(t+n) = \mathbf{F}[\boldsymbol{y}, \widehat{\boldsymbol{\varPsi}}]$$
$$= a.\widehat{\varPsi}(t+n) + b.\widehat{\varPsi}(t) \qquad (24)$$
$$+ c.\widehat{\varPsi}(t-1) + d.y(t) + e.y(t-1)$$

$$\widehat{\varPsi}(t+n) = \mathbf{g}\Big[\big(y(t), y(t\text{-}1)\big), \boldsymbol{w}\Big]$$
$$= \sum_{j=1}^{k} w_j.\phi_j\big(y(t), y(t\text{-}1)\big) \qquad (25)$$

Once the RRBF model found by training, the second step is to find the NARX model parameter (a, b, c, d, e) by linear regression on the second training set (Figure 10). In order to find the best RRBF structure with regard to the k centers, several models of neural networks have been created by varying the number of basis functions from 2 to 50 nodes. The *k-means* training algorithm has been used to find the best Gaussian centers for each k-structure. The basis width parameter of the Gaussian nodes was fixed to 1. The neural network trained with the first set is validated in the second one. Validation uses data different from the training set, thus the validation set is independent from the estimated model. This helps to select the best one among the different model parameters. To avoid overfitting or underfitting, the optimal model parameters should be selected so as to have the best performance measure associated with the validation set. Since this dataset is independent from the estimated model, the generalization error obtained is a fair estimated. The best model which gives the best mean prediction error in the validation set is then selected and tested in the third set (the test set). The model with the best generalization performance is then selected with this crossvalidation technique (see Figure 11).
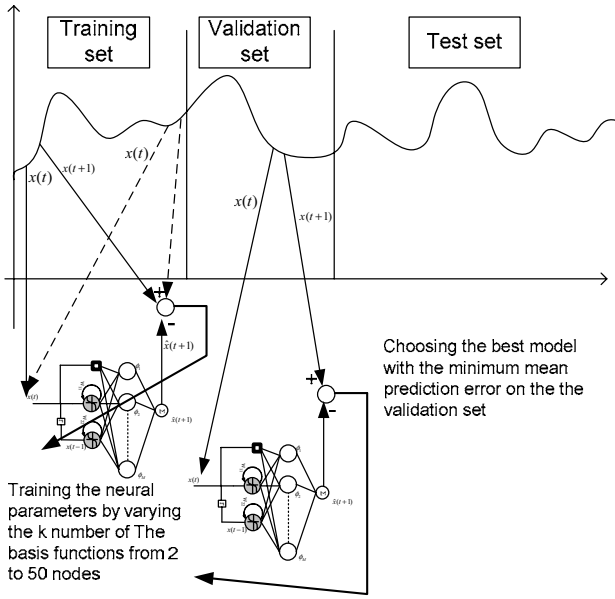
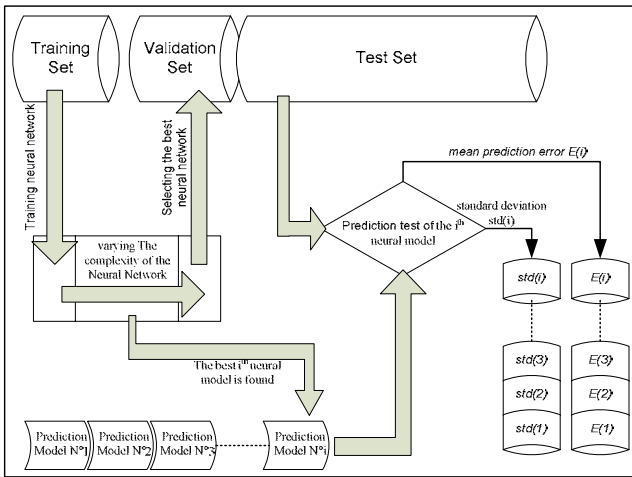**Figure 10.** *Training process and crossed validation*

Figure 11. *Procedure for performance criteria estimation.*

5.1. *Mackey Glass*

The first data set is the chaotic Mackey-Glass time series data [21]. This time series is a benchmark problem extensively used: it's a non periodic and non convergent time series (Figure 12). Considering our final applicative objective able to carry out predictions on such a signal is of good omen: what makes difficult a modeling phase are real complex systems which generally have a nonstationary and non-linear behavior. The time series is governed by the following formula:

$$\frac{dy}{dt} = b\frac{y(t-d)}{1+y(t-d)^c} - ay(t) \tag{26}$$

where a, b, c and d are real constants. The values of most commonly parameters used in literature are $a = 0.1$, $b = 0.2$, $c = 10$, and $d = 17$ where at the initial time $t_0 = 0$, $x(t_0) = 1.2$. The differential equation is approximated by the 4th order Runge-Kutta algorithm with time step equal to 1. Tests on this time series aim at predicting future values $y(t+n)$ by using past values as follows:

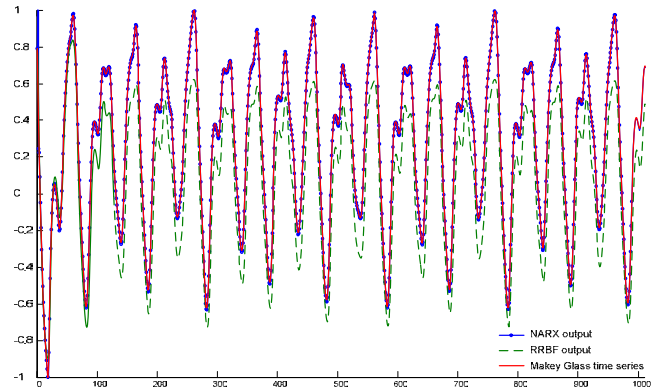$$y(t+n) = \mathbf{F}[y(t), y(t-1)] \tag{27}$$

**Figure 12.** The original Mackey Glass time series and the one step prediction $y(t+1)$.

5.2. *Lorenz*

The Lorenz time series is a long synthetic chaotic time series (see Figure 13) obtained from http://www.physics.emory.edu/~weeks/research/tseries1.html. The time series is governed by the following differential equations:

$$\frac{dy}{dt} = \sigma(x_t - y_t), \quad \frac{dx}{dt} = ry_t - x_t - y_t z_t$$
$$\frac{dz}{dt} = y_t x_t - bz_t \tag{28}$$

we take parameter setting $\sigma = 10, r = 28, b = 8/3$ and use the 4000 *y*-ordinate points derived from a Rung-Kutta integrator with time step 0.01. Tests on this time series aim at predicting future values $y(t+n)$ by using past values as follows:

$$y(t+n) = \mathbf{F}[y(t), y(t-1)] \tag{29}$$
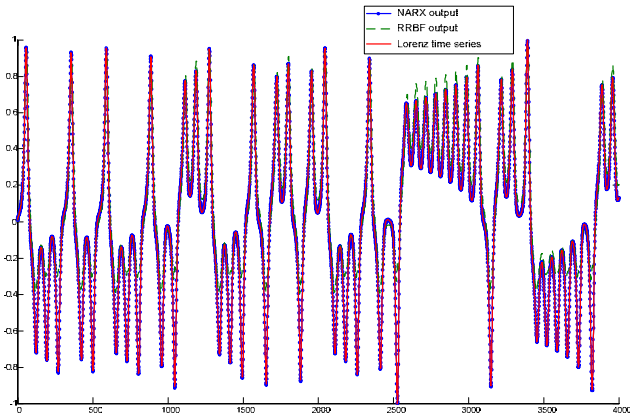
Figure 13. The original Lorenz time series and the one step prediction $y(t+1)$.

6. Results

To measure performances of the prediction model presented in section 3 by the metrics presented in section 4, the simulation tests were running *1000* times for each benchmark (*i=1 to M=1000*). At each simulation running *i*, the mean prediction error *E(i)* and the standard deviation *std(i)* are calculated.

6.1. *Mackey-Glass*

Figures N°14 to N°18 illustrate a comparative results obtained by the whole model ($y(t+n) = \mathbf{F}[y, \widehat{\Psi}]$) and those obtained by the neural network only ($\widehat{\Psi}(t+n) = \mathbf{g}[\mathbf{y}, \mathbf{w}]$). The results show that the prediction performances of the whole model are better than those obtained by the NN model. Figure 14 reveals the great disparity of the **Accuracy** parameter among the two models. The Figure 15 illustrates the three parameters of the accuracy: the **timeliness** parameter, the **precision** parameter and the **repeatability** parameter. One can see that the timeliness and the precision obtained by the whole model are better than the NN model, and this for all the horizon of prediction. The Figure 16 illustrates the interpretation of the results shown by the Figure 15. Both **timeliness** and **precision** parameters of the whole model are smaller than those obtained by the RRBF. The prediction performances of the whole model are better than the neural model: the predictions of the whole model are less dispersed, and are closer to the real system.

The results obtained for the repeatability (shown by the Figure 15) are very important for the robustness of the NN. The prediction repeatability obtained by the whole model is very high from (t+1) to (t+6). This means that the combination obtained by the NARX model gives more

repeatability for the predictions. Figure 17 shows the complexity of the Neural Network for each horizon of the prediction. The number *k* of the Gaussian nodes is presented with the mean value and the standard deviation for all the tests. The Figure 18 reveals the results obtained for the parameters (a, b, c, d, e) of the whole model.

6.2. *Lorenz data set*

The results obtained with the two prediction models for the Lorenz data set are presented by figures N°19 to N°22. Prediction performances measures obtained by the whole model are better than those obtained by the NN model. Figure 21 shows the complexity of the Neural Network for each horizon of the prediction. The number *k* of the Gaussian nodes is presented with the mean value and the standard deviation for all the tests. The Figure 22 reveals the results for the parameters (a, b, c, d, e) of the whole model.

7. Conclusion

In this paper, a neural NARX predictor model is proposed, tested and compared. The proposed predictor model provides a first step prediction made by a recurrent neural network. A second step consists to reduce the prediction error made by the recurrent NN. A linear combination between the neural network outputs and the real data, at several past times is then used. The whole NARX model obtained improves highly the predictive performances of the neural network. Experiments and comparative studies demonstrate superior performance of the proposed approach over the classical structur of the RRBF network. Further study is necessary to investigate the robustness of the proposed methods and to compare it with other neural architecture. One possible way to test the robustness is to train the prediction model using various training data groups generated from the system at different sampling rates. If the prediction models are formed similarly despite using different training data group, the proposed prediction methods are deemed robust in this sense.

The second contribution of the article is a new methodology for measuring the prediction performances of a neural network. The **Accuracy** of the prediction is given by three parameters: the **timeliness**, the **precision** and the **repeatability**. Theses metrics gives some *numerical information* to compare different schemes and provides an objective way to measure how changes in training or prediction model (Neural network architecture) affect the quality of predictions in term of: the training algorithm stability (repeatability), the dispersion (precision) and the rightness (timeliness) of the prediction. This methodology can be used in several prediction applications as failure prognosis [49], software fault prediction, Air quality prediction ...etc. There is no general agreement as to an

appropriate and acceptable set of metrics that can be employed in prediction applications, and researchers are still working on this. For example in the maintenance field, one can find some interesting studies on the prognosis metrics in [43,38,39,24].
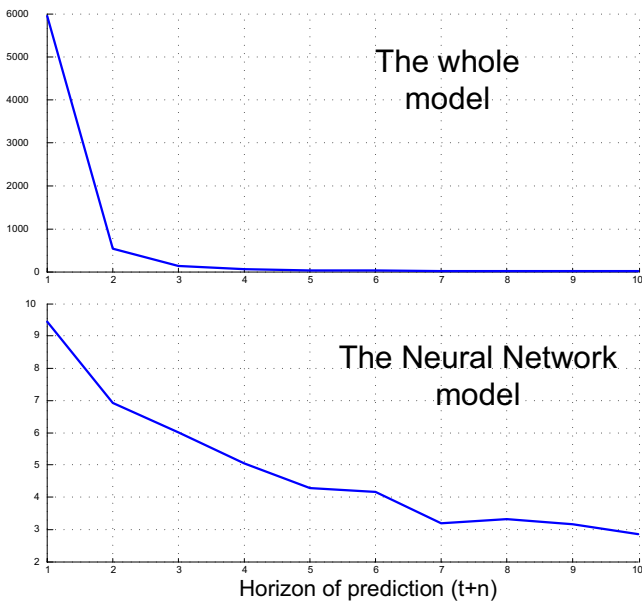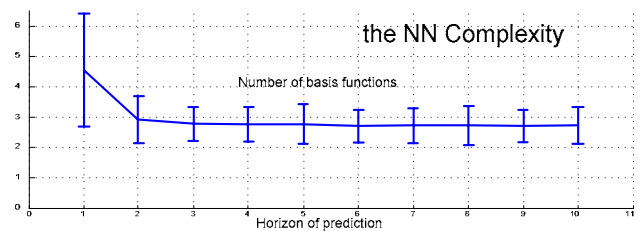

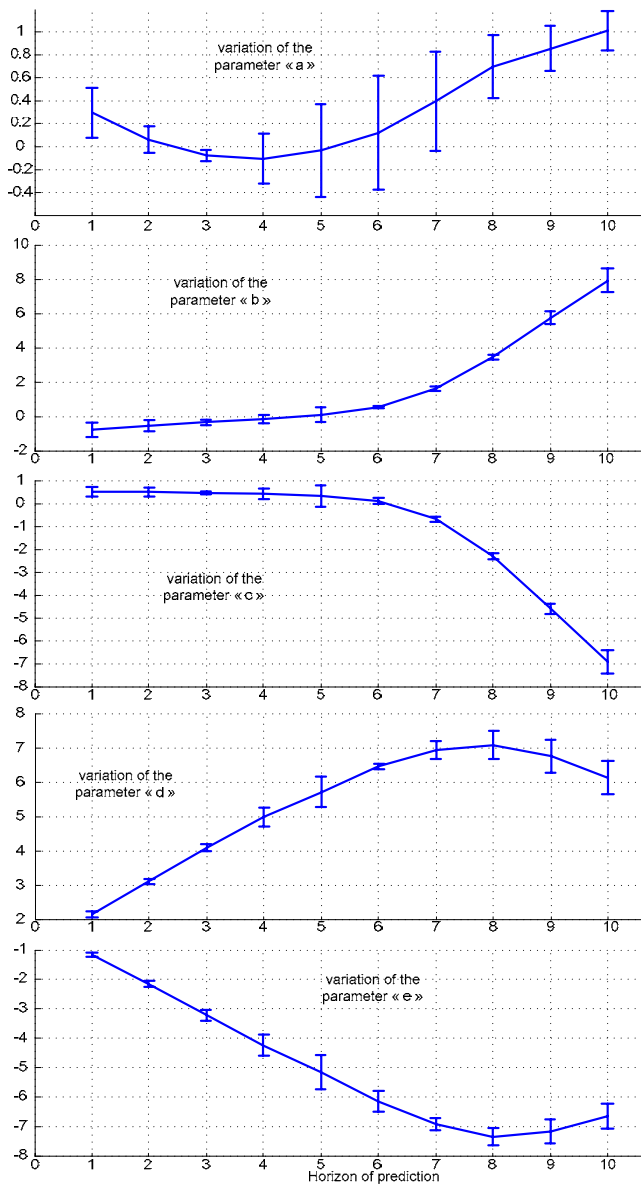**Figure 14.** *The prediction Accuracy for the Mackey Glass data set*
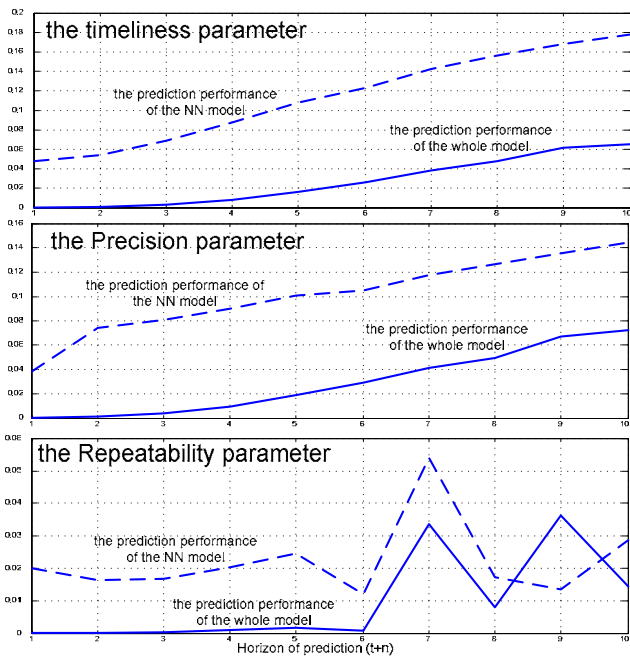
Figure 15. *Prediction metric for the Mackey Glass data set*
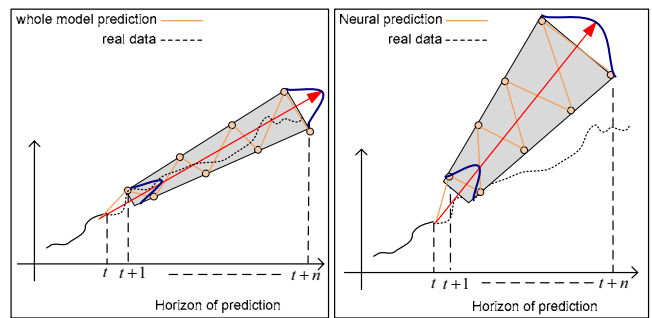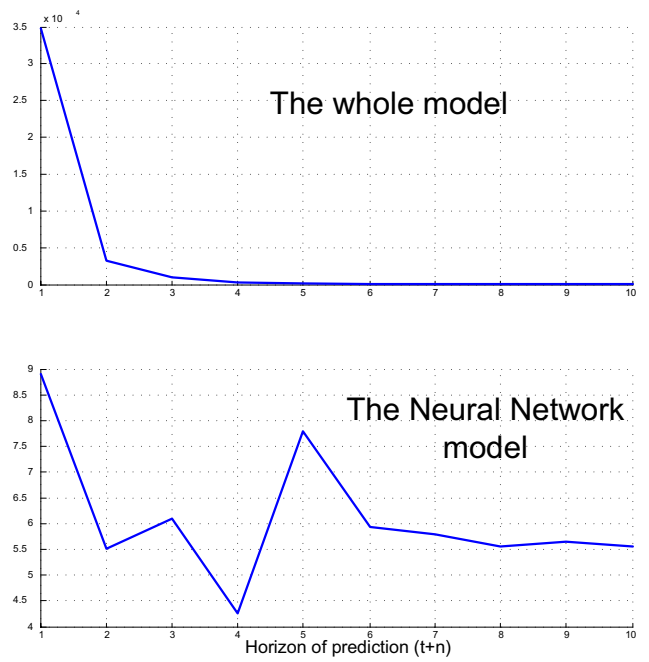

Figure 16. the relation between the (Timeliness / Precision) and the quality of the prediction. The performance obtained by the whole model is better than the NN prediction.
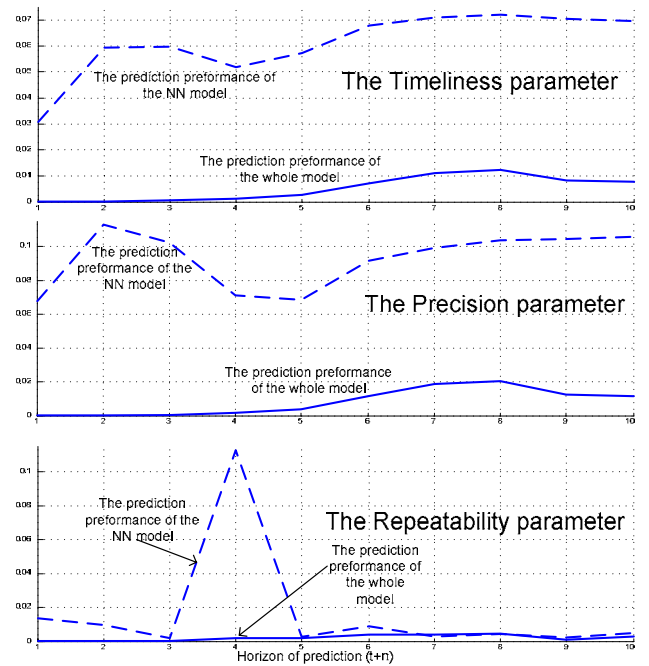

Figure 17. *The NN complexity for the Mackey Glass data set.*

Figure 18. *The NARX parameter for the Mackey Glass data set*

Figure 19. *The prediction Accuracy for the Lorenz data set*

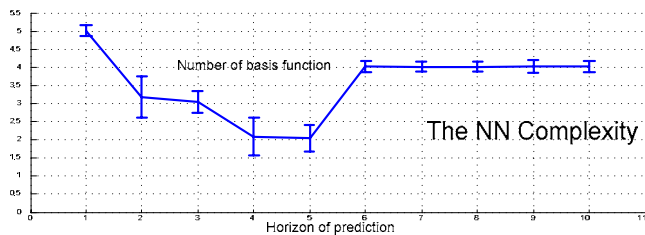**Figure 20.** *Prediction metric for the Lorenz data set*

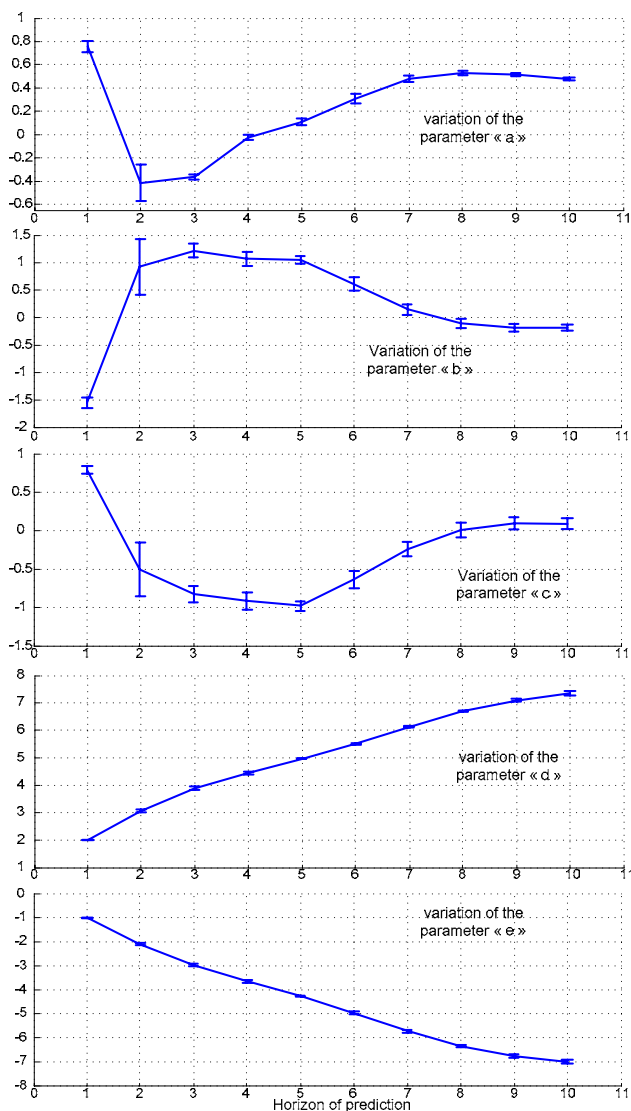**Figure 21.** *The NN complexity for the Lorenz data set.*

Figure 22. *The NARX parameter for the Lorenz data set*

References

[1] Aliev R.A., B. Fazlollahi, R.R. Aliev, B. Guirimov, Linguistic time series forecasting using fuzzy recurrent neural networks, Soft Comput.12(2) (2007)183–190.

[2] Andalib Arash and Farid Atry, « Multi-step ahead forecasts for electricity prices using NARX: A new approach, a critical analysis of one-step ahead forecasts », Energy Conversion and Management (Elsevier) Volume 50, Issue 3, March 2009, Pages 739-747

[3] Angelov P., Filev D., An approach to online identification of takagi-sugeno fuzzy models, IEEE Transaction Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 34 (2004), pp. 484-498.

[4] Barbounis T.G., Theocharis J.B., "A locally recurrent fuzzy neural network with application to wind speed prediction using spatial correlation", Neurocomputing, vol. 70, Issues 7-9, 2007, p. 1525-1542.

[5] Bernauer E., Les réseaux de neurones et l'aide au diagnostic : un modèle de neurones bouclés pour l'apprentissage de séquences temporelles, thèse de doctorat, LAAS/Toulouse 1996.

[6] Box G., G.M. Jenkins, Time Series Analysis: Forecasting and Control, second ed., Holden-Day,Oakland,CA,1976.

[7] Dai Q., S. Chen, Chained DLS-ICBP neural networks with multiple steps time series prediction, Neural Process. Lett. 21 (2005) 95–107.

[8] Dammers E., Measurement in the ex post evaluation of forecasts, Quality and Quantity, vol. 27:1 (1993), pp. 31-45.

[9] De Gooijer J.G., Hyndman R.J., 25 years of time series forecasting, International Journal of Forecasting, vol. 22 (2006), pp. 443–473.

[10] Dodonov Evgueni, and Rodrigo Fernandes de Mello « A novel approach for distributed application scheduling based on prediction of communication events » Future Generation Computer Systems (Elsevier 2009)

[11] Dorffner G., Neural networks for time series processing, Neural Network World6 (4)(1996) 447–468.

[12] Frasconi P., Gori M. et Soda G., « Local Feedback Multilayered Networks », Neural Computation, Vol. 4, pp. 120-130, 1992.

[13] Gao Y., M.J. Er, NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches, Fuzzy Sets Syst.150(2005) 331–350.

[14] Gowrishankar Ramesh, B.H.S.; Satyanarayana, P.S, "Neural network based BER prediction for 802.16e channel" 15th International Conference on Software, Telecommunications and Computer Networks, (SoftCOM 2007). 27-29 Sept. 2007 Page(s):1 – 5.

[15] Graves Daniel, Wiltold Pedrycz, "Fuzzy prediction architecture using recurrent neural networks" Neurocomputing 72 (2009), pp. 1668-1678.

[16] Harpham C., C.W. Dawson, The effect of different basis functions on a radial basis function network for time series prediction: a comparative study, Neurocomputing 69 (2006)2161–2170.

[17] Haykin S., J.C. Principe, Making sense of a complex world, IEEE Signal Process. Mag. 15 (3) (1998) 66–81.

[18] Haykin S., X.B. Li, Detection of signals in chaos, Proc. IEEE 83 (1) (1995) 95–122.

[19] Hyndman R.J., Koehler A.B., Another look at measures of forecast accuracy, International Journal of Forecasting, vol. 22 (2006), pp. 679-688.

[20] Jacquin A.P., A.Y. Shamseldin, Development of rainfall-runoff models using Takagi–Sugeno fuzzy inference systems, J.Hydrol.329(2006) 154–173.

[21] Jang J.S.R., "ANFIS: adaptive network-based fuzzy inference system", IEEE Transaction on Systems, Man and Cybernetics 23(3), 1993, p. 24-38.

[22] Kahraman C., Gülbay M., Kabak Ö., Applications of Fuzzy Sets in Industrial Engineering: A Topical Classification, in Kahraman, C. (Ed.) Fuzzy Applications in Industrial Engineering, Studies in Fuzziness and Soft Computing 201, Springer, Heidelberg (2006), pp. 1-55.

[23] Karray F.O., C. deSilva, Soft Computing and Intelligent Systems Design: Theory, Tool sand Applications, Pearson Addison Wesley, Harlow, Essex,2004.

[24] Leao, B.P.; Yoneyama, T.; Rocha, G.C.; Fitzgibbon, K.T. « Prognostics performance metrics and their relation to requirements, design, verification and cost-benefit » in international conference on Prognostics and Health Management (PHM 2008), 2008. Denver CO,

[25] Liao W.T.,Clustering of time series data—a survey,Pattern Recognition 38 (2005) 1857–1874.

[26] Lin J., E. Keogh, W.Truppel, Clustering of streaming time series is meaningless, in: Eigth ACM SIGMOD Workshop on Research Issues in Data Ming and Knowledge Discovery, ACM, San Diego, 2003, pp.56–65.

[27] Luh Guan-Chun, WU Chun-Yin, CHENG Wei-Chong « Artificial immune regulation (AIR) for model-based fault diagnosis » Artificial immune systems : (Catania, 13-16 September 2004)ICARIS 2004 : international conference on artificial immune systems No3, Catania , ITALIE (13/09/2004) 2004 , vol. 3239, pp. 28-41.

[28] Mandic D.P., J.A. Chambers, Recurrent Neural Networks for Prediction: Learning Algorithms and Architectures and Stability, Wiley, Chichester, 2001.

[29] McDonnell J.R., D.Waagen, Evolving recurrent perceptrons for time series modeling, IEEETrans.NeuralNetworks5(1)(1994)665–685.

[30] Menezes Jr. José Maria P., and Guilherme A. Barreto, « Long-term time series prediction with the NARX network: An empirical evaluation » Neurocomputing Volume 71, Issues 16-18, October 2008, Pages 3335-3343 Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).

[31] Mitra S., Hayashi S., Neuro–Fuzzy Rule Generation: Survey in Soft Computing Framework, IEEE Transactions on neural networks, vol. 11:3 (2000), pp. 758-768.

[32] Neurmaier A., T. Schneider, Estimation of parameters and eigen modes of multivariate autoregressive models, ACMTrans. Math. Software27(1)(2001) 27–57.

[33] Palluat N., D. Racoceanu1, and N. Zerhouni « A neuro-fuzzy monitoring system: Application to flexible production systems » Computers in Industry (Elsevier) Volume 57, Issue 6, August 2006, Pages 528-538 E-maintenance Special Issue

[34] Pham H.T., Tran, V.T., Yang, B-S., "A hybrid of nonlinear autoregressive model with exogenous input and autoregressive moving average model for long-term machine state forecasting", Expert Systems with Applications (Elsevier) (2009)

[35] Pisoni E., Farina M., Carnevale C., Piroddi L., "Forcasting peak air pollution levels using NARX models" Engineering Applications of Artificial Intelligence (Elsevier) Vol. 22 (2009) p. 593-602.

[36] Principe J.C., N.R. Euliano, W.C. Lefebvre, Neural Adaptive Systems: Fundamentals Through Simulations, Wiley, New York, 2000.

[37] Samantaa B., and S. Bandopadhyay « Construction of a radial basis function network using an evolutionary algorithm for grade estimation in a placer gold deposit » Computers & Geosciences (Elsevier) Volume 35, Issue 8, August 2009, Pages 1592-1602

[38] Saxena, A.; Celaya, J.; Balaban, E.; Goebel, K.; Saha, B.; Saha, S.; Schwabacher, M., « Metrics for evaluating performance of prognostic techniques », in international conference on Prognostics and Health Management (PHM 2008), 2008. Denver CO,

[39] Saxena Abhinav, Bhaskar Saha, Jose Celaya, Kai Goebel, Sankalita Saha, « On Applying the Prognostics Performance Metrics », in annual Conference of the Prognostics and Health Management Society (PHM) 2009 San Diego, CA September 27 – October 1, 2009

[40] Setnes M., R. Babuska, H.B. Verbruggen, Rule-based modeling: precision and transparency, IEEE Trans.SMC-Part C28(1)(1998)165–169.

[41] Sorjamaa A., J.H.N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing 70 (16–18) (2007) 2861–2869.

[42] Sum J.P.F., Kan W.-K, Young G.H, "a note on the equivalence of NARMAX anf RNN", Neural Computing & Applications (Springer-Verlag), (1999), vol8, p. 33-39

[43] Vachtsevanos G., Lewis F.L., Roemer M., Hess A., Wu B., Intelligent Fault Diagnosis and Prognosis for Engineering Systems, Hoboken, New Jersey, Wiley & Sons, 2006.

[44] Vernieuwe H., N.E.C. Verhoest, B. DeBaets, R.Hoeben, F.P. DeTroch, Cluster-based fuzzy models for ground water flow in the unsaturated zone, Adv. Water Resour.30(2007)701–714.

[45] Wei H.L., D.Q. Zhu, S.A. Billings and M.A. Balikhin, « Forecasting the geomagnetic activity of the Dst index using multiscale radial basis function networks » Advances in Space Research Volume 40, Issue 12, 2007, Pages 1863-1870

[46] Zemouri R., Racoceanu D., Zerhouni N. – b – « Réseaux de neurones Récurrents à Fonction de base Radiales :RRFR/ Application au pronostic », Revue d'Intelligence Artificielle, RSTI série RIA, Vol. 16, N°03, 2002.

[47] Zemouri R., Racoceanu D., Zerhouni N., "Recurrent Radial Basis Function network for Time-Series Prediction", Engin. Appl. of Artificial Intelligence, vol. 16, 2003, pp.453-463.

[48] Zemouri, R., Gouriveau, R., Zerhouni, R. (2009). Combining a recurrent neural network and a PID controller for prognostic purpose: A way to improve the accuracy of predictions. In PENTOM'09 Conf., Autrans, France.

[49] Zemouri R., Gouriveau R., « Towards Accurate and Reproducible Predictions for Prognostic: an Approach Combining a RRBF Network and an AutoRegressive Model » 1st IFAC Workshop on Advanced Maintenance Engineering, Services and Technology (A-MEST'10), Lisbon, Portugal, 1-2 July 2010.

[50] Zhou S.M., X.-D.Xu, A new type of recurrent fuzzy neural network for modeling dynamic systems, Knowl.BasedSyst.14(2001)243–251.

Ryad Zemouri received his engineer degree at Mouloud Mammeri University of Tizi-Ouzou (Algeria) in 1998 and his Ph.D. in automatic control and computer science in Franche-Comté University in 2003. He joined the Conservatoire National des Arts & Métiers (CNAM) of Paris in September 2003 where he works actually as Associate Professor. His main research interests are computational intelligence, pattern recognition, neural networks, and industrial prognostics systems.

Rafael Gouriveau received his engineer degree from National Engineering School of Tarbes (ENIT) in 1999. He then got his MS (2000) and his Ph.D. in Industrial Systems in 2003, both from the Toulouse National Polytechnic Institute (INPT). During his PhD, he worked in the field of risk management and dependability analysis. In September 2005, he joined the national high school of mechanics and microtechniques of Besançon (ENSMM) as Associate Professor. His main teaching activities are concerned with production, maintenance, manufacturing, and informatics domains. Nowadays, his research interests concern the development of industrial prognostics systems by using connexionist systems like neuro-fuzzy methods, and the investigation of reliability modeling by using possibility theory.

Noureddine Zerhouni received his engineer degree from National Engineers and Technicals School of Alger (ENITA) in 1985. After a short period in industry as engineer, he then received his Ph.D. Degree in Automatic Control from the Grenoble National Polytechnic Institute in 1991. In September 1991, he joined the National Engineering School of Belfort (ENIB) as Associate Professor. At this time, his main activity were concerned with modelling, analyse and control of manufacturing systems.

Since September 1999, Noureddine Zerhouni is Professor at the national high school of mechanics and microtechniques of Besançon (www.ens2m.fr). He found and is responsible of the research group "Design and maintenance of mechatronic systems" (COSMI) of FEMTO-ST Institute, AS2M Department. His main research activities are thereby concerned with intelligent

maintenance systems and e-maintenance: COSMI group works on detection, diagnostic and prognostic of failure, on reactive and dynamic scheduling problems, as well as on reliability modeling. Noureddine Zerhouni has been and is involved in various European and National projects on intelligent maintenance systems like FP5 European Integrated Project of ITEA program (Information Technology for European Advancement) PROTEUS, NEMOSYS (Naval E-Maintenance Oriented SYStem) with DCNS, AMIMAC-FAME (Reliability Improvement of Embeded Machine) with ALSTOM and CEGELEC.

He is also member of the international relations group of ENSMM, and develops special relationships with Romania and Algeria.