Neurocomputing, in press.

Training Wavelet Networks for Nonlinear Dynamic Input-Output Modeling.

Y. Oussar, I. Rivals, L. Personnaz, G. Dreyfus
Laboratoire d'Électronique
École Supérieure de Physique et Chimie Industrielles
10, rue Vauquelin
F - 75231 PARIS Cedex 05, FRANCE.
Phone: 33 1 40 79 45 41 Fax: 33 1 40 79 44 25
E-mail: Yacine.Oussar@espci.fr

Abstract

In the framework of nonlinear process modeling, we propose training algorithms for feedback wavelet networks used as nonlinear dynamic models. An original initialization procedure is presented, that takes the locality of the wavelet functions into account. Results obtained for the modeling of several processes are presented; a comparison with networks of neurons with sigmoidal functions is performed.

Keywords: Training, Wavelet networks, Nonlinear dynamic modeling, Neural networks, Feedback networks, Recurrent networks.

1. INTRODUCTION.

During the past few years, the nonlinear dynamic modeling of processes by neural networks has been extensively studied. Both input-output [7] [8] and state-space [5] [14] models were investigated. In standard neural networks, the non-linearities are approximated by superposition of sigmoidal functions. These networks are universal approximators [2] and have been shown to be parsimonious [3].

Wavelets are alternative universal approximators; wavelet networks have been investigated in [17] in the framework of *static* modeling; in the present paper, we propose a training algorithm for feedback wavelet networks used as nonlinear *dynamic* models of processes. We first present the wavelets that we use and their properties. In section 3, feedforward wavelet networks for static modeling are presented. In section 4, the training systems and algorithms for dynamic input-output modeling with wavelet networks, making use of the results of section 3, are described. For illustration purposes, the modeling of several processes by wavelet networks and by neural networks with sigmoidal functions is presented in section 5.

2. FROM ORTHOGONAL WAVELET DECOMPOSITION TO WAVELET NETWORKS.

The theory of wavelets was first proposed in the field of multiresolution analysis; among others, it has been applied to image and signal processing [6]. A family of wavelets is constructed by translations and dilations performed on a single fixed function called the *mother wavelet.* A wavelet ϕ_j is derived from its mother wavelet ϕ by

$$\phi_j\,(z) = \phi\left(\frac{x - m_j}{d_j}\right) \qquad (1)$$

where its translation factor m_j and its dilation factor d_j are real numbers ($d_j > 0$). We are concerned with modeling problems, i.e. with the fitting of a data set by a finite sum of wavelets. There are several ways to determine the wavelets for this purpose:

- From orthogonal wavelet decomposition theory, it is known that, with a suitable choice of ϕ, and if m_j and d_j are integers satisfying some conditions, the family $\left\{\phi_j\right\}$ forms an orthogonal wavelet basis. A weighted sum of such functions with appropriately chosen m_j and d_j can thus be used; in this way, only the weights have to be computed [18].
- Another way to design a wavelet network is to determine the m_j and d_j according to a space-frequency analysis of the data; this leads to a set of wavelets which are not necessarily orthogonal [10] [1].
- Alternatively, one can consider a weighted sum of wavelets functions whose parameters m_j and d_j are adjustable real numbers, which are to be trained together with the weights.

In the latter approach, wavelets are considered as a family of parameterized nonlinear functions which can be used for nonlinear regression; their parameters are estimated through "training". The present paper introduces training algorithms for *feedback* wavelet networks used for dynamic modeling, which are similar in spirit to training algorithms used for feedback neural networks.

Choice of a mother wavelet

In the present paper, we choose the first derivative of a gaussian function, $\phi(x) = \pm\, x \exp\left(\pm \frac{1}{2}\, x^2\right)$ as a mother wavelet. It may be regarded as a differentiable version of the Haar mother wavelet, just as the sigmoid is a differentiable version of a step function, and it has the universal approximation property [17]. This mother wavelet has also been used in reference [17]. More complex wavelet functions, such as the second derivative of the gaussian (as in [1]) may be used, but they will not be considered here.

The wavelet network.

In the case of a problem with N_i inputs, multidimensional wavelets must be considered. The simplest, most frequent choice ([1], [6], [17], [18]) is that of separable wavelets, i.e. the product of N_i monodimensional wavelets of each input:

$$\Phi_j(x) = \prod_{k=1}^{N_i} \phi(z_{jk}) \quad \text{with} \quad z_{jk} = \frac{x_k - m_{jk}}{d_{jk}} \qquad (2)$$

where m_j and d_j are the translation and dilation vectors. We consider wavelet networks of the form:

$$y = \psi(x) = \sum_{j=1}^{N_w} c_j \Phi_j(x) + a_0 + \sum_{k=1}^{N_i} a_k x_k \ . \tag{3}$$

(3) can be viewed as a network with N_i inputs, a layer of N_w wavelets of dimension N_i, a bias term, and a linear output neuron. When linear terms are expected to play an important role in the model, it is customary to have additional direct connections from inputs to outputs, since there is no point in using wavelets for reconstructing linear terms. Such a network is shown in Figure 1.

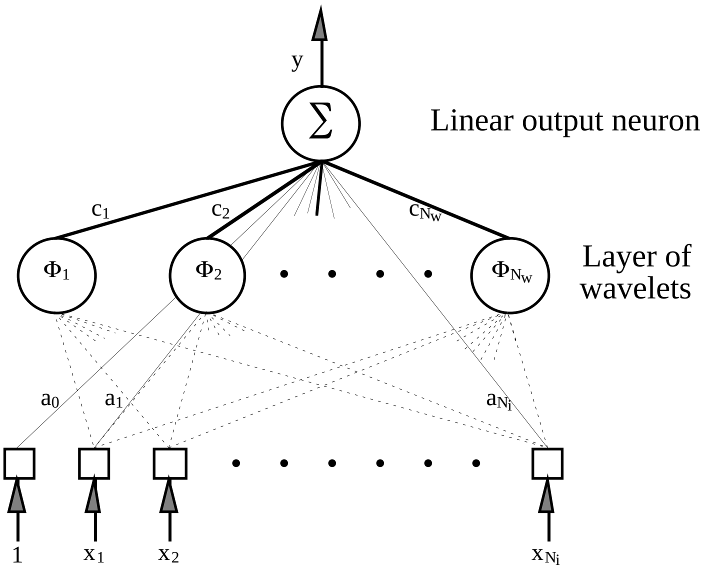

Figure 1. A feedforward wavelet network.

3. STATIC MODELING USING FEEDFORWARD WAVELET NETWORKS.

Static modeling with wavelet networks has been investigated by other authors in [17]. In order to make the paper self-contained, we devote the present section to introducing notations and to recalling basic equations which will be used in Section 4 for dynamic modeling.

We consider a process with N_i inputs and a scalar output y_p. Steady-state measurements of the inputs and outputs of the process build up a training set of N examples $(\boldsymbol{x}^n, y_p^n)$, $\boldsymbol{x}^n = \left[x_1^n, \ldots, x_{N_i}^n\right]^\mathrm{T}$ being the input vector for example n and y_p^n the corresponding measured process output. In the domain defined by the training set, the static behavior of the process is assumed to be described by:

$$y_p^n = f(\boldsymbol{x}^n) + w^n \qquad n = 1 \text{ to } N \tag{4}$$

where f is an unknown nonlinear function, and $\{w^n\}$ denotes a set of independent identically distributed random variables with zero mean and variance σ_w^2.

We associate the following wavelet network to the assumed model (4):

$$y^n = \psi\left(\boldsymbol{x}^n, \theta\right) \qquad n = 1 \text{ to } N \tag{5}$$

where y^n is the model output value related to example n, the nonlinear function ψ is given by relation (3), and θ is the set of adjustable parameters:

$$\theta = \{m_{jk},\ d_{jk},\ c_j,\ a_k,\ a_0\} \quad \text{with} \quad j = 1 , \ldots , N_w \ \text{ and } \ k = 1 , \ldots , N_i \qquad (6)$$

θ is to be estimated by training so that ψ approximates the unknown function f on the domain defined by the training set.

3.1. Training feedforward wavelet networks.

As usual, the training is based on the minimization of the following quadratic cost function:

$$J\,(\theta) = \frac{1}{2} \sum_{n=1}^{N} (y_p^n - y^n)^2 = \frac{1}{2} \sum_{n=1}^{N} (e^n)^2 \qquad (7)$$

The minimization is performed by iterative gradient-based methods.

The partial derivative of the cost function with respect to θ is:

$$\frac{\partial J}{\partial \theta} = - \sum_{n=1}^{N} e^n \frac{\partial y^n}{\partial \theta} \qquad (8)$$

where $\frac{\partial y^n}{\partial \theta}$ is a short notation for $\left.\frac{\partial y}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{x} = \boldsymbol{x}^n}$. The components of the latter vector are:

- parameter a$_0$:

$$\frac{\partial y^n}{\partial a_0} = 1 \qquad (9)$$

- direct connection parameters:

$$\frac{\partial y^n}{\partial a_k} = x_k^n \quad k = 1 , \ldots, N_i \qquad (10)$$

- weights:

$$\frac{\partial y^n}{\partial c_j} = \Phi_j(\boldsymbol{x}^n) \quad j = 1 , \ldots , N_w \qquad (11)$$

- translations:

$$\frac{\partial y^n}{\partial m_{jk}} = \pm \frac{c_j}{d_{jk}} \left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{\boldsymbol{x}=\boldsymbol{x}^n} \qquad k = 1 , \ldots , N_i \ \text{ and } \ j = 1 , \ldots , N_w \qquad (12)$$

with

$$\left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{\boldsymbol{x} = \boldsymbol{x}^n} = \phi\left(z_{j1}^n\right) \phi\left(z_{j2}^n\right) \ldots \phi\,'\left(z_{jk}^n\right) \ldots \phi\left(z_{jN_i}^n\right) \qquad (13)$$

where $\phi\,'\left(z_{jk}^n\right)$ is the value of the derivative of the scalar mother wavelet at point z_{jk}^n:

$$\phi\,'\left(z_{jk}^n\right) = \left.\frac{d\phi(z)}{dz}\right|_{z=z_{jk}^n} \qquad (14)$$

- dilations:

$$\frac{\partial y^n}{\partial d_{jk}} = \pm \frac{c_j}{d_{jk}} z_{jk}^n \left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{\boldsymbol{x}=\boldsymbol{x}^n} \qquad k = 1 , \ldots , N_i \ \text{ and } \ j = 1 , \ldots , N_w \qquad (15)$$

At each iteration, the parameters are modified using the gradient (8), according to:

$$\Delta\theta = -M \frac{\partial J}{\partial \theta} \tag{16}$$

where M is some definite positive matrix ($M = \mu\, Id$, $\mu>0$ in the case of a simple gradient descent, or $M = \mu\, \widetilde{H}^{-1}$, $\mu>0$ where $\widetilde{H}^{-1}$ is an approximation, updated iteratively, of the inverse Hessian, for quasi-Newton methods).

3.2. Initialization of the network parameters.

Initializing the wavelet network parameters is an important issue. Similarly to Radial Basis Function networks (and in contrast to neural networks using sigmoidal functions), a random initialization of all the parameters to small values (as usually done with neural networks) is not desirable since this may make some wavelets too local (small dilations) and make the components of the gradient of the cost function very small in areas of interest. In general, one wants to take advantage of the input space domains where the wavelets are not zero.

Therefore, we propose an initialization for the mother wavelet $\phi(x) = \pm\, x \exp\left(\pm \frac{1}{2} x^2\right)$ based on the input domains defined by the examples of the training sequence. We denote by $[\alpha_k, \beta_k]$ the domain containing the values of the k-th component of the input vectors of the examples. We initialize the vector m of wavelet j at the center of the parallelepiped defined by the N_i intervals $\{[\alpha_k, \beta_k]\}$: $m_{jk} = \frac{1}{2}\left(\alpha_k + \beta_k\right)$. The dilation parameters are initialized to the value $0.2\left(\beta_k - \alpha_k\right)$ in order to guarantee that the wavelets extend initially over the whole input domain. The choice of the a_k ($k = 1, \ldots, N_i$) and c_j ($j = 1, \ldots, N_w$) is less critical: these parameters are initialized to small random values.

3.3. Stopping conditions for training.

The algorithm is stopped when one of several conditions is satisfied: the Euclidean norm of the gradient, or of the variation of the gradient, or of the variation of the parameters, reaches a lower bound, or the number of iterations reaches a fixed maximum, whichever is satisfied first.

The final performance of the wavelet network model depends on whether: (i) the assumptions made about the model (relation 4) are appropriate, (ii) the training set is large enough, (iii) the family contains a function which is an approximation of f with the desired accuracy in the domain defined by the training set, (iv) an efficient (i.e. second-order) training algorithm is used.

4. DYNAMIC MODELING USING WAVELET NETWORKS.

We propose to extend the use of wavelet networks to the dynamic modeling of single-input-single-output (SISO) processes. The training set consists of two sequences of length N: the input sequence $\{u(n)\}$ and the measured process output $\{y_p(n)\}$. As in the static case, the aim is to approximate f by a wavelet network.

Depending on the assumptions about the noise, either feedforward or feedback predictors may be required [9]. For example, if it is assumed that the noise acting on the process is state noise (see for instance equation (35) of section 5.2), i.e. if a Nonlinear AutoRegressive with eXogeneous inputs (NARX, or Equation Error) model

$$y_p(n) = f\left(y_p(n \pm 1)\right), y_p(n \pm 2), ..., y_p\left(n \pm N_s\right), u(n \pm 1), ..., u\left(n \pm N_e\right)\right) + w_n \qquad (17)$$

is assumed to be valid, then *the optimal associated predictor is a feedforward one,* whose inputs are past outputs *of the process* y_p and the external inputs u:

$$y(n) = f\left(y_p(n\text{-}1), y_p(n\text{-}2), ..., y_p(n\text{-}N_s), u(n\text{-}1), ..., u(n\text{-}N_e)\right) \ . \qquad (18)$$

f is a unknown nonlinear function, which is to be approximated by a wavelet network ψ given by (3).

Conversely, if it is assumed that the noise is output noise, i.e. if an Output Error model

$$\begin{aligned} &s(n) = f\left(s(n \pm 1)\right), s(n \pm 2), ..., s\left(n \pm N_s\right), u(n \pm 1), ..., u\left(n \pm N_e\right)\right) \\ &y_p(n) = s(n) + w(n) \end{aligned} \qquad (19)$$

is assumed to be valid, then *the optimal associated predictor is a feedback one,* whose inputs are past outputs *of the model y* and the external inputs u:

$$y(n) = f\left(y(n\text{-}1), y(n\text{-}2), ..., y(n\text{-}N_s), u(n\text{-}1), ..., u(n\text{-}N_e)\right) \qquad (20)$$

In the absence of noise, either feedforward or feedback predictors can be used. If the goal is the design of a simulation model, i.e. of a model that can compute the output more than one time step ahead, a feedback predictor should be trained [9].

In all cases, θ is to be estimated so that ψ approximates the unknown function f on the domain defined by the training set.

We define the copy n *(n = 1,..., N)* as the wavelet network configuration giving $y(n)$ at its output in the case of a feedforward predictor, and as the feedforward part of the network canonical form in the case of a feedback predictor [8]. In order to keep the notations equivalent with the previous section we note: $y^n = y(n)$.

4.1. Training feedforward wavelet predictors.

In this case, the N copies are independent, and the training is similar to that of a static model. Therefore, the input vector of copy n can be viewed as the vector $\boldsymbol{x}^n$ defined in section 3 and $\{y_p(n)\}$ as the process output defined as y_p^n. More precisely, the inputs of copy n can be renamed as:

- external inputs: $x_k^n = u(n\text{-}k)$ with $k = 1, ..., N_e$
- state inputs: $x_k^n = y_p(n\text{-}k\text{+}N_e)$ with $k = N_e\text{+}1, ..., N_e\text{+}N_s$

Since the state inputs of the copies are forced to the corresponding desired values, the predictor is said to be trained in a *directed* [8], or *teacher-forced* [4] fashion.

4.2. Training feedback wavelet predictors.

In this case, the N copies are not independent: the N output values $y^n = y(n)$ of the network may be considered as being computed by a large feedforward network made of N cascaded copies of the feedforward part of the canonical form of the feedback network [8]: the state inputs of copy n are equal to the state outputs of copy n-1. The inputs and outputs of copy n are renamed as:

- external inputs: $x_k^n = u(n\text{-}k)$ with $k = 1, \ldots, N_e$.
- state inputs: $x_k^n = y(n\text{-}k+N_e)$ with $k = N_e+1, \ldots, N_e+N_s$.
- state outputs: $x_k^n = y(n\text{-}k+N_e+N_s+1)$ with $k = N_e+N_s+1, \ldots, N_e+2N_s$.

$x_{N_e+N_s+1}^n = y(n) = y^n$ is the n-th value of the output of the network.
$\theta^n = \left\{m_{jk}^n, d_{jk}^n, c_j^n, a_k^n, a_0^n\right\}$ with $j = 1, \ldots, N_w$ and $k = 1, \ldots, N_e+N_s$
is the set of parameters of copy n. The feedback predictor network and copy n are shown on figure 2.

Since the state inputs of the first copy only are forced to desired values, the predictor is said to be trained in a *semi-directed* fashion [8], (also known as *backpropagation through time* [15]: the gradient of the cost function is computed by a single backpropagation through the N copies).
The gradient of $J(\theta) = \frac{1}{2}\sum_{n=1}^{N}\left(y_p^n - y^n\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(e^n\right)^2$ with respect to θ can be expressed as the sum of the gradient with respect to each of the N copies θ^n of θ:

$$\frac{\partial J}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial J}{\partial \theta^n} = \sum_{n=1}^{N} \frac{\partial J}{\partial y^n}\frac{\partial y^n}{\partial \theta^n} \qquad (21)$$

The analytical expressions of $\frac{\partial y^n}{\partial m_{jk}^n}, \frac{\partial y^n}{\partial d_{jk}^n}, \frac{\partial y^n}{\partial c_j^n}, \frac{\partial y^n}{\partial a_k^n}, \frac{\partial y^n}{\partial a_0^n}$ which are the components of $\frac{\partial y^n}{\partial \theta^n}$ are identical to those given (without superscript n for θ) in relations (9) – (15), for the training of feedforward nets.
The set of partial derivatives $\left\{\frac{\partial J}{\partial y^n}\right\}$ can be computed by backpropagation through the feedforward network consisting of the N cascaded copies.
We introduce the intermediate variables $\{q_k^n\}$, q_k^n being the partial derivative of $-J$ with respect to x_k^n, the state variable x_k of the n-th copy:

$$q_k^n = -\frac{\partial J}{\partial x_k^n} \qquad (22)$$

Copy N:

- output:

$$q_{\text{out}}^N = q_{N_e+N_s+1}^N = e^N \qquad (23)$$

- other output state variables:

$$q_k^N = 0 \quad \textit{with} \quad k = N_e+N_s+2, \ldots, N_e+2N_s \qquad (24)$$

- for the N_s state inputs :

$$q_k^N = \left(a_k^N + \sum_{j=1}^{N_w} \frac{c_j^N}{d_{jk}^N}\frac{\partial \Phi_j}{\partial z_{jk}^N}\right) q_{\text{out}}^N \quad \text{with} \quad k = N_e+1, \ldots, N_e+N_s \qquad (25)$$

Copies n = N-1 to 2:

- output:

$$q_{\text{out}}^n = e^n + q_{N_e+1}^{n+1} \qquad (26)$$

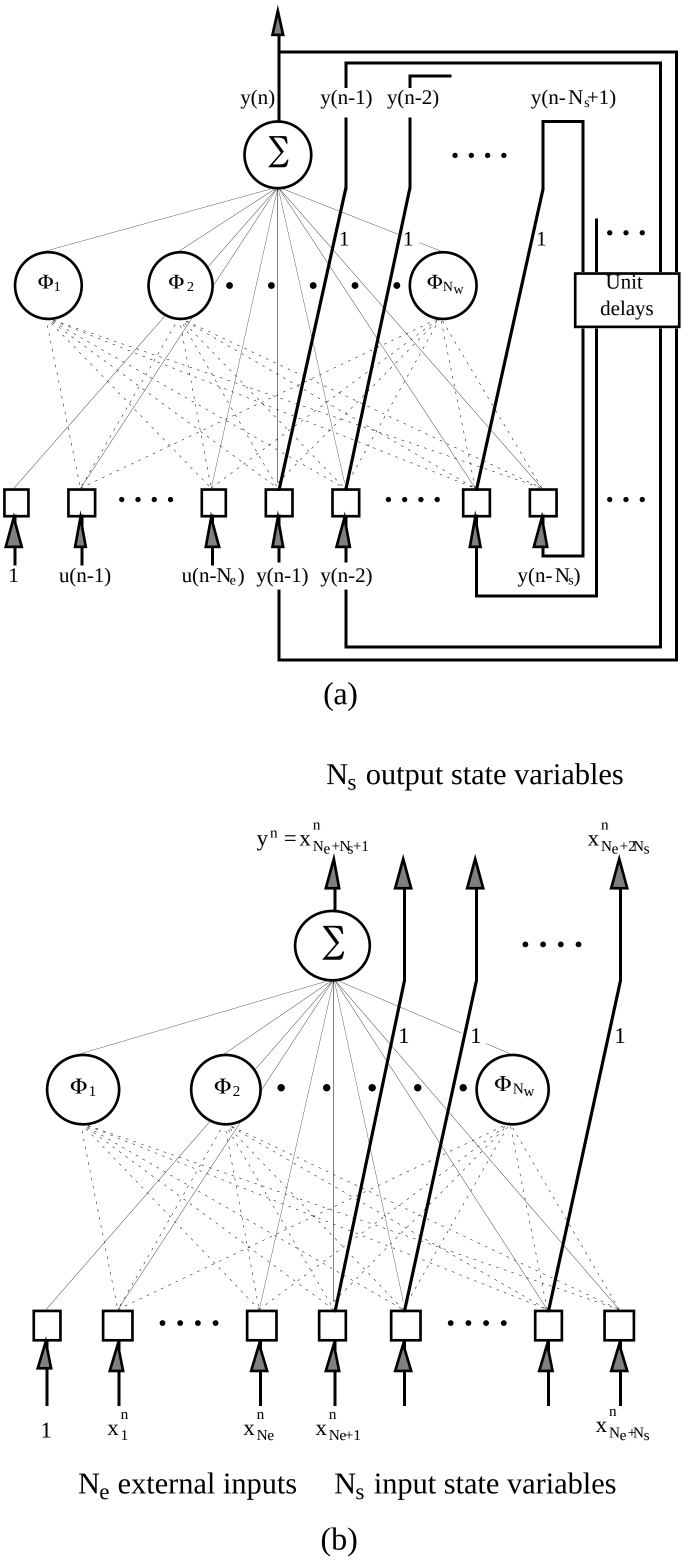

Figure 2. (a) feedback predictor network; (b) n-th copy for training.

- other output state variables:

$$q_k^n = q_{k-N_s}^{n+1} \quad \text{with} \quad k = N_e+N_s+2 \text{ , ... , } N_e+2\,N_s \tag{27}$$

- the N_s-1 first state inputs :

$$q_k^n = q_{k+N_s+1}^n + \left(a_k^n + \sum_{j=1}^{N_w} \frac{c_j^n}{d_{jk}^n} \frac{\partial \Phi_j}{\partial z_{jk}^n} \right) q_{\mathrm{out}}^n \text{ with } k = N_e+1 , \dots , N_e+N_s-1 \qquad (28)$$

- the last state input :

$$q_{N_e+N_s}^n = \left(a_{N_e+N_s}^n + \sum_{j=1}^{N_w} \frac{c_j^n}{d_{jk}^n} \frac{\partial \Phi_j}{\partial z_{jk}^n} \right) q_{out}^n \qquad (29)$$

Copy 1:

- output:

$$q_{\mathrm{out}}^1 = e^1 + q_{N_e+1}^2 \qquad (30)$$

5. SIMULATION RESULTS.

In this section we make use of the above algorithms for training input-output wavelet networks on data gathered from simulated and from real processes, and we make use of the algorithms presented in [8] for training input-output neural networks with one hidden layer of sigmoidal neurons on the same data.

The wavelet networks are input-output models as defined by (18) or (20), where the unknown function *f* is approximated by wavelet networks whose mother wavelet is described in section 2 (derivative of a gaussian).

The neural networks used have one hidden layer of sigmoidal units and direct connections from the inputs:

$$y\,(\boldsymbol{x}) = \sum_{j=1}^{N_\sigma} c_j \tanh\left(v_j(\boldsymbol{x})\right) + a_0 + \sum_{k=1}^{N_i} a_k x_k \text{ with } v_j(\boldsymbol{x}) = \sum_{k=1}^{N_i} w_{jk} x_k \qquad (31)$$

We denote by Training Mean Square Error (TMSE) the mean square error on the training set:

$$\mathrm{TMSE} = \frac{1}{N} \sum_{n=1}^{N} \left(y_p(n) - y^n\right)^2 = \frac{2}{N} J \qquad (32)$$

The performance of the model is estimated by the Performance Mean Square Error (PMSE), computed on a test sequence.

The training procedure starts with a simple gradient method (500 iterations) which is followed by a quasi-Newton method (BFGS with line search by Nash [11]).

5.1. Modeling of a simulated process without noise.

The process considered here is simulated with a second order nonlinear equation. This process has been used to illustrate a selection procedure for neural models [16]. The output of the process is given by:

$$y_p(n) = f\,\left(y_p(n\text{-}1), y_p(n\text{-}2), u(n\text{-}1)\right) = \frac{24 + y_p(n\text{-}1)}{30} y_p(n\text{-}1) - 0.8 \frac{u(n\text{-}1)^2}{1 + u(n\text{-}1)^2} y_p(n\text{-}2) + 0.5u(n\text{-}1) \qquad (33)$$

Since noise is absent, either feedforward or feedback predictors can be used. In order to obtain a simulation model of the process, we chose to train a feedback predictor:

$$y(n) = \psi\left(y(n\text{-}1), y(n\text{-}2), u(n\text{-}1), \theta\right) \qquad (34)$$

A training and a test sequence of 1000 samples each were generated. The input sequence for both training and test consists of pulses with random amplitude in the range [-5,5] and with random duration between 1 and 20 sampling periods. Figures 3a and 3b show the training sequence.

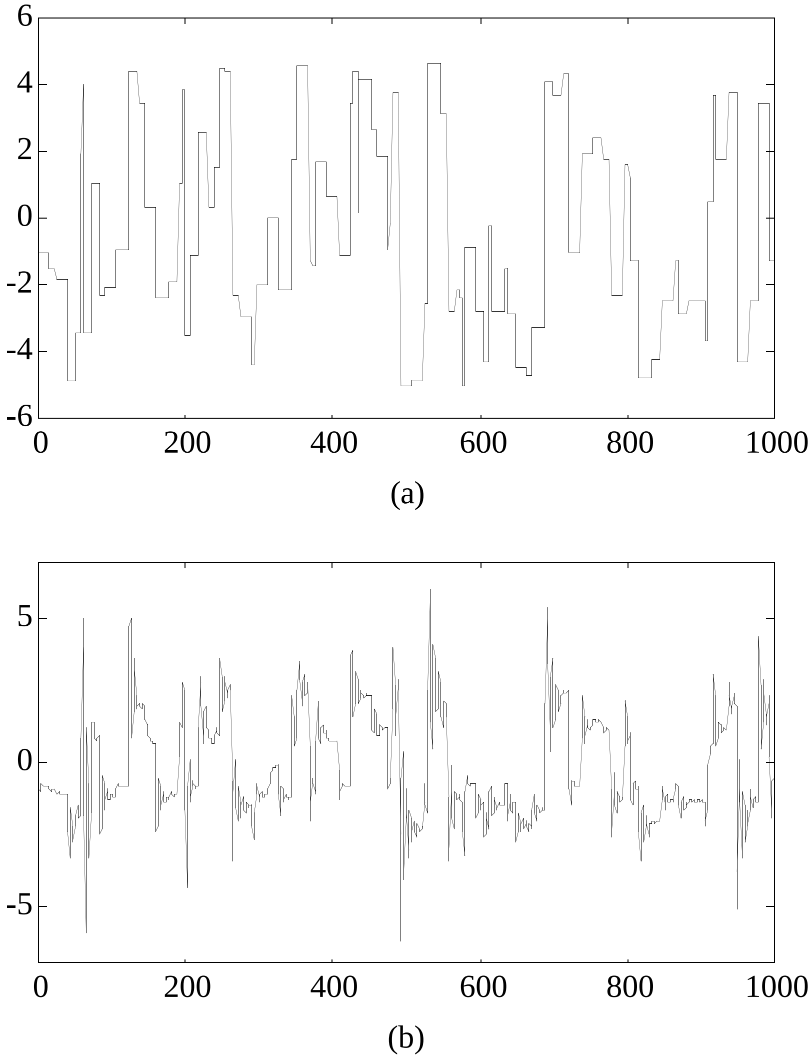

Figure 3: (a) Training input sequence; (b) Training output sequence.

Several feedback wavelet networks were trained, with fifty different initializations for each network. The results corresponding to the minimal PMSE's are given in table 1. Additional wavelets do not improve the performance.

Number of wavelets	Number of parameters	TMSE	PMSE
1	11	$7.6\ 10^{-2}$	$1.5\ 10^{-1}$
2	18	$2.0\ 10^{-2}$	$3.6\ 10^{-2}$
3	25	$2.2\ 10^{-3}$	$6.7\ 10^{-3}$
4	32	$2.8\ 10^{-4}$	$1.3\ 10^{-3}$
5	39	$5.2\ 10^{-5}$	$2.9\ 10^{-4}$
6	46	$3.8\ 10^{-6}$	$2.9\ 10^{-5}$

Table 1. Wavelet modeling results for the noiseless simulated process.

Several feedback neural networks were trained, with fifty different initializations for each network. The results corresponding to the minimal PMSE's are given in table 2. Additional hidden neurons do not improve the performance.

Number of sigmoids	Number of parameters	TMSE	PMSE
1	9	$1.1\ 10^{-1}$	$1.8\ 10^{-1}$
2	14	$7.1\ 10^{-2}$	$1.0\ 10^{-1}$
3	19	$1.1\ 10^{-3}$	$8.4\ 10^{-3}$
4	24	$3.9\ 10^{-4}$	$2.3\ 10^{-3}$
5	29	$4.5\ 10^{-6}$	$1.8\ 10^{-5}$
6	34	$4.2\ 10^{-6}$	$1.6\ 10^{-5}$

Table 2. Neural modeling results for the noiseless simulated process.

In this example, the two types of networks perform with roughly the same accuracy.

5.2. Modeling of a simulated process with noise.

The previous trainings were performed with noiseless data. In this section, we study the case where a zero-mean noise acts on the process. As described in section 4, we consider two cases: NARX models and Output Error models.

In the first one, the state variables of the model used for simulating the process are the output of the process at times n and n-1, and the noise is added to the state variables. It is a NARX model given by the following equation:

$$y_p(n) = f\left(y_p(n-1), y_p(n-2), u(n-1)\right) + w(n) \qquad (35)$$

where f is the function introduced in the previous section.

In the second case, the state variables of the model used for simulating the process are not subject to noise, but noise is added to the output variable: it is an Output Error model given by the following equations:

$$\begin{cases} s(n) = f\left(s(n-1), s(n-2), u(n-1)\right) \\ y_p(n) = s(n) + w(n) \end{cases} \qquad (36)$$

where $s(n)$ and $s(n-1)$ are the state variables.

Since we are interested in black-box modeling, we generate training and test data from (35) or (36). The input sequences used are identical to those shown in the previous section. The processes are simulated with a noise of variance $\sigma_w^2 = 10^{-2}$. Once the training and test sequences are generated, we pretend not to know equations (35) and (36). Since we must make a decision as to whether we train a feedforward predictor or a feedback predictor, we have to make an assumption about the effect of noise on the process (output noise or state noise). The results presented below have been obtained by making the right assumption: for modeling the data generated by equation (35), we have trained a feedforward wavelet predictor, and, for modeling the data generated by equation (36), we have used a feedback predictor (the adverse effect of making the wrong assumption about the noise has been demonstrated in [8]).

Since we are modeling a process with noise, the goal is the following: find the smallest network such that the error on the test set and the error on the training be as close as possible to the variance of the noise. Because the process is simulated, we know the variance of the noise, so that we know whether this goal is achieved.

As in the case of the process without noise, several networks with an increasing number of wavelets were trained. The optimal N_w, for which the PMSE is smallest (no overfitting occurs), is 5; the results presented on table 3 show that the variance of the noise is indeed reached.

	TMSE	PMSE
NARX Model	$9.6\ 10^{-3}$	$1.0\ 10^{-2}$
Output Error Model	$1.0\ 10^{-2}$	$1.2\ 10^{-2}$

Table 3. Wavelet modeling results for noisy simulated processes, when the right assumption about the effect of noise is made.

5.3. Modeling of a real process.

The process to be modeled is the hydraulic actuator of a robot arm. The external input u is the position of a valve and the output y_p is the oil pressure. A sequence of 1024 points is available. We consider the first half of the data sequence as a training sequence. We use a feedback predictor with N_e=1 and N_s=2 so that:

$$y(n) = \psi(y(n\text{-}1),\ y(n\text{-}2),\ u(n\text{-}1),\ \theta) \tag{37}$$

Predictors having increasing numbers of wavelets were trained, with 50 initializations for each predictor. The best PMSE is obtained with a network of 2 wavelets (18 parameters); the corresponding values of the TMSE and PMSE are:

TMSE = 0.11 PMSE = 0.13

Figure 4 shows the responses of the process and of the wavelet network on the test sequence.

Table 4 shows the results obtained on the same problem with other input-output models. The neural network model whose performance is reported has three hidden neurons (best PMSE of 50 trainings with different initializations).

Input-output model	PMSE	Numbers of parameters	Reference
Hinging hyperplanes	0.34	14	[12]
Neural Network	0.14	19	This paper
Wavelet network	0.13	18	This paper

Table 4. A comparison of different input-output models of the hydraulic actuator.

In this modeling problem, wavelet and neural networks perform equivalently. However, these results are still not as satisfactory as those obtained in [13] with a state-space model using a neural network with sigmoid functions; state-space modeling with wavelet networks will not be considered in the present paper.

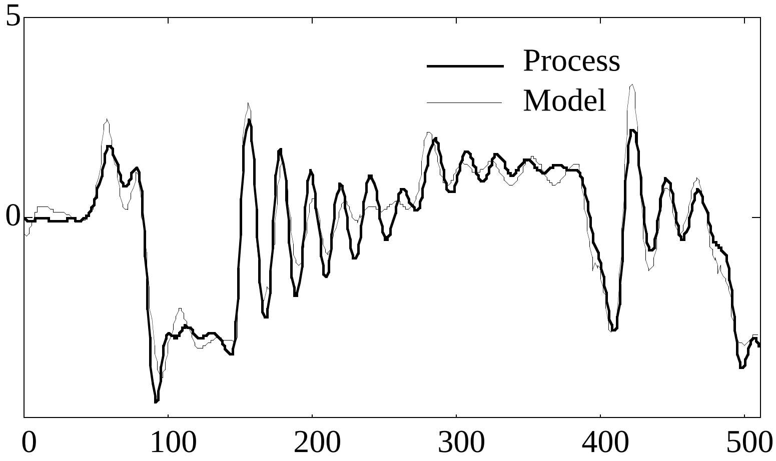

Figure 4. Model and process outputs on the test sequence.

6. CONCLUSION.

In this paper, we extend the use of wavelet networks for function approximation to dynamic nonlinear input-output modeling of processes. We show how to train such networks by a classic minimization of a cost function through second order gradient descent implemented in a backpropagation scheme, with appropriate initialization of the translation and dilation parameters. The training procedure is illustrated on the modeling of simulated and real processes. A comparison with classic sigmoidal neural networks leads to the conclusion that the two types of networks can perform equivalently in terms of accuracy and parsimony for nonlinear input-output modeling of processes with a small number of inputs, provided the technical precautions outlined above (proper initialization and efficient training algorithms) are taken.

References.

[1] M. Cannon and J.-J. E. Slotine, *Space-Frequency Localized Basis Function Networks for Nonlinear System Estimation and Control,* Neurocomputing 9 (3) (1995) 293-342.

[2] G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function,* Mathematics of control, signals and systems, 2 (1989) 303-314.

[3] K. Hornik, M. Stinchcombe, H. White and P. Auer, *Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives,* Neural Computation, 6 (6) (1994) 1262-1275.

[4] M. I. Jordan, *The Learning of Representations for Sequential Performance,* Doctoral Dissertation, University of California, San Diego, 1985.

[5] A. U. Levin, *Neural networks in dynamical systems; a system theoretic approach,* PhD Thesis, Yale University, New Haven, CT, 1992.

[6] S. Mallat, *A Theory for Multiresolution Signal Decomposition: The Wavelet Transform,* IEEE Trans. Pattern Anal. Machine Intell. 11 (7) (1989) 674-693.

[7] K. S. Narendra and K. Parthasarathy, *Identification and Control Of Dynamical Systems Using Neural Networks,* IEEE Trans. on Neural Networks, 1 (1) (1990) 4-27.

[8] O. Nerrand, P. Roussel-Ragot. L. Personnaz, G. Dreyfus, *Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms,* Neural Computation, 5 (2) (1993) 165-199.

[9] O. Nerrand , P. Roussel-Ragot, D. Urbani, L. Personnaz, G. Dreyfus, *Training recurrent neural networks: why and how? An Illustration in Process Modeling,* IEEE Trans. on Neural Networks 5 (2) (1994) 178-184.

[10] Y. C. Pati and P. S. Krishnaparasad, *Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformations,* IEEE Trans. on Neural Networks 4 (1) (1993) 73-85.

[11] E. Polak, *Computational Methods in Optimization: A Unified Approach* (Academic Press, New–York, 1971).

[12] P. Pucar and M. Millnert, *Smooth Hinging Hyperplanes - An Alternative to Neural Nets,* in: Proceedings of 3rd European Control Conference, Vol. 2 (Rome, 1995) 1173-1178.

[13] I. Rivals, L. Personnaz, G. Dreyfus, J.L. Ploix, *Modélisation, Classification et Commande par Réseaux de Neurones : Principes Fondamentaux, Méthodologie de Conception et Illustrations Industrielles,* in: J.P. Corriou, ed., Les réseaux de Neurones pour la Modélisation et la Commande de Procédés (Lavoisier Tec et Doc, 1995) 1-37.

[14] I. Rivals and L. Personnaz, *Black Box Modeling With State-Space Neural Networks*, in: R. Zbikowski and K. J. Hunt eds., Neural Adaptive Control Technology I (World Scientific, Singapore, 1996) 237-264.

[15] D. E. Rumelhart, and J. L. McClelland, *Parallel Distributed Processing*, (MIT Press, Cambridge, MA, 1986).

[16], D. Urbani, P. Roussel-Ragot. L. Personnaz and G. Dreyfus, *The Selection of Neural Models of Non-linear Dynamical Systems by Statistical Tests*, in: Proceedings of the IEEE Conference on Neural Networks for Signal Processing IV, (Greece ,1994) 229-237.

[17] Q. Zhang and A. Benveniste, *Wavelet Networks,* IEEE Trans. on Neural Networks 3 (6) (1992) 889-898.

[18] J. Zhang, G. G. Walter, Y. Miao and W. N. Wayne Lee, *Wavelet Neural Networks For Function Learning*, IEEE Trans. on Signal Processing 43 (6) (1995) 1485-1497.