

NONLINEAR SYSTEM IDENTIFICATION USING LOCAL ARX MODELS BASED ON THE SELF-ORGANIZING MAP

Luís Gustavo M. Souza
Guilherme A. Barreto

Federal University of Ceará
Department Teleinformatics Engineering
e-mails: {[luisgustavo.guilherme](mailto:luisgustavo.guilherme@deti.ufc.br)}@deti.ufc.br
Av. Mister Hull, S/N - Campus of Pici, Center of Technology
CP 6005, CEP 60455-760, Fortaleza, CE, Brazil

Abstract- In this paper, we build multiple local ARX (Auto-Regressive model with eXogenous variables) models for nonlinear system identification based on Kohonen's Self-Organizing Map. These models are evaluated in the task of approximating the inverse dynamics of two benchmarking input-output data sets: a hydraulic actuator and a flexible robotic arm. Simulation results demonstrate that the proposed multiple local ARX modeling approach outperforms both single ARX model and MLP-based NARX model approaches in the aforementioned tasks.

Keywords- Kohonen map, local linear models, NARX models, system identification, hydraulic actuator, robot arm.

1 Introduction

Dynamical system identification is the field interested in building mathematical models of nonlinear systems, starting from experimental time series data, measurements, or observations (Ljung, 1999). Typically, a certain linear or nonlinear model structure which contains unknown parameters is chosen by the user. In general, the parameters should be computed so that the errors between estimated (or predicted) and actual outputs of the system are minimized in order to capture the dynamics of the system as close as possible. The resulting model can be used as a tool for analysis, simulation, prediction, monitoring, diagnosis, and control system design.

Artificial neural network (ANN) models have been successfully applied to the identification and control of a variety of nonlinear dynamical systems, such as chemical, economic, biological or technological processes (Chen et al., 1990; Narendra and Lewis, 2001; Ibnkahla, 2003; Purwar et al., 2007). Such achievements are mainly due to a number of theoretic and empirical studies showing that supervised feedforward architectures, such as multilayer perceptron (MLP) or radial basis function (RBF) networks, can approximate arbitrarily well any continuous input-output mapping (see Pinkus (1999) and Schilling et al. (2001) for surveys).

In this paper, we propose two system identification techniques which uses the self-organizing map (SOM) (Kohonen, 1997) as a basis for local function approximation. Such an SOM-based local dynamic modeling and control approaches have been successfully applied to complex system identification and control tasks (Cho et al., 2007; Díaz-Blanco et al., 2007; Cho et al., 2006; Lan et al., 2005; Barreto et al., 2004; Principe et al., 1998), such as chaotic time series prediction, aircraft control, inverted pendulum control, nonlinear missile control, biological reactor control, among others.

The SOM is an unsupervised neural algorithm designed to build a representation of neighborhood (spatial) relationships among vectors of an unlabeled data set. The neurons in the SOM are arranged in an output layer A in one-, two-, or even three-dimensional (1-D, 2-D, 3-D) arrays. Each neuron $i \in A$ has a weight vector \mathbf{w}_i with the same dimension of the input vector \mathbf{x} . The network weights are trained according to a competitive-cooperative scheme in which the weight vectors of a winning neuron and its neighbors in the output array are updated after the presentation of an input vector. Usually, a trained SOM is used for clustering and data visualization purposes.

The SOM was previously applied to learn static input-output mappings (Barreto et al., 2003; Walter and Ritter, 1996; Midenet and Grumbach, 1994), which are usually represented as

$$y(t) = g(u(t)) \quad (1)$$

in which the current output $y(t) \in \mathfrak{R}$ depends solely on the current input $u(t) \in \mathfrak{R}$. In this paper, however, we are interested in

systems which can be described by the NARX model:

$$y(t) = f[y(t-1), \dots, y(t-n_y); u(t), u(t-1), \dots, u(t-n_u)], \quad (2)$$

where n_y and n_u are the (memory) orders of the dynamical model. From (2), the system output y at time t depends, on the past n_y output values and on the past n_u values of the input u . In many situations, it is also desirable to approximate the inverse mapping of a nonlinear plant, given by

$$u(t) = f^{-1}[y(t-1), \dots, y(t-n_y); u(t-1), \dots, u(t-n_u)]. \quad (3)$$

In system identification, the goal is to obtain estimates of $f(\bullet)$ and $f^{-1}(\bullet)$ from available input-output time series data pairs $\{u(t), y(t)\}$, $t = 1, \dots, N$, where N is the time series length.

For the SOM and other unsupervised networks to be able to learn dynamical mappings, they must have some type of *short-term memory* (STM) mechanism (Wang, 2003; Wang and Arbib, 1990). That is, the SOM should be capable of temporarily storing past information about the system input and output vectors. There are several STM models, such as delay lines, leaky integrators, reaction-diffusion mechanisms and feedback loops (Principe et al., 2002; Barreto and Araújo, 2001), which can be incorporated into the SOM to allow it to approximate a dynamical mapping $f(\bullet)$ or its inverse $f^{-1}(\bullet)$. In order to draw a parallel with standard system identification approaches, we limit ourselves to describe the VQTAM approach in terms of time delays as STM mechanisms.

The remainder of the paper is organized as follows. In Section 2 SOM architecture and its learning process are described. In Section 3 two SOM-based local ARX models are introduced. Simulations and performance are presented in Section 4. The paper is concluded in Section 5.

2 The Self-Organizing Map

The self-organizing map, introduced by Kohonen (1997), is commonly used to transform high dimensional input vectors into a lower dimensional discrete representation that preserves topological neighborhoods. Basically, it works as a vector quantization algorithm that adaptively quantizes the input space by discovering a set of representative prototype vectors (also called reference vectors or centroids). The basic idea of SOM is to categorize vectorial stochastic data into different groups by means of a winner-take-all selection rule.

The SOM is composed of two fully connected layers: an input layer and a competitive layer. The input layer simply receives the incoming input vector and forwards it to the competitive layer through weight vectors. The goal of SOM is to represent the input data distribution by the distribution of the weight vectors. Competitive learning drives the winning weight vector to become more similar to the input data. Throughout this paper, we represent the weight vector between input layer and neuron i as

$$\mathbf{w}_i = [w_{i,1}, w_{i,2}, \dots, w_{i,j}, \dots, w_{i,p}]^T, \quad (4)$$

where $w_{i,j} \in \mathfrak{R}$ denotes the weight connecting node j in the input layer with neuron i , and p , is the dimension of the input vector. In what follows, a brief description of the original SOM algorithm is given. Firstly, we use the euclidean distance metric to find the current winning neuron, $i^*(t)$, as given by the following expression:

$$i^*(t) = \arg \min_{\forall i} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|, \quad (5)$$

where $\mathbf{x}(t) \in \mathfrak{R}^p$ denotes the current input vector, $\mathbf{w}_i(t) \in \mathfrak{R}^p$ is the weight vector of neuron i , and t denotes the time steps associated with the iterations of the algorithm. Secondly, it is necessary to adjust the weight vectors of the winning neuron and of those neurons belonging to its neighborhood in the output array:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)], \quad (6)$$

where $0 < \alpha(t) < 1$ is the learning rate, which usually decay in time, and $h(i^*, i; t)$ is a weighting function that defines the boundaries of the neighborhood of the winning neuron at time t .

A common choice for the neighborhood function is the gaussian function given by

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*}(t) - \mathbf{r}_i(t)\|^2}{2\sigma^2(t)}\right), \quad (7)$$

where $\mathbf{r}_i(t) \in \mathfrak{R}^D$ and $\mathbf{r}_{i^*}(t) \in \mathfrak{R}^D$, are respectively, the positions of neurons i and i^* in a predefined output array (of dimension D) where the neurons are arranged in the nodes, and $\sigma(t) > 0$ defines the radius of the neighborhood function at time t .

The variables $\alpha(t)$ and $\sigma(t)$ should both decay with time to guarantee convergence of the weight vectors to stable steady states. In this paper, we adopt an exponential annealing for both, as given by

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_T}{\alpha_0}\right)^{(t/T)} \quad \text{and} \quad \sigma(t) = \sigma_0 \left(\frac{\sigma_T}{\sigma_0}\right)^{(t/T)}, \quad (8)$$

where α_0 (σ_0) and α_T (σ_T) are the initial and final values of $\alpha(t)$ ($\sigma(t)$), respectively. The operations defined from Eq. (5) to Eq.(8) are repeated until a steady state of global ordering of the weight vectors has been achieved. In this case, we say that the map has converged.

In addition to usual clustering properties, the resulting map also preserves the topology of the input samples in the sense that adjacent input patterns are mapped into adjacent neurons on the map. Due to this topology-preserving property, the SOM is able to cluster input information and spatial relationships of the data on the map. This clustering ability of the SOM has shown to be quite useful for the identification of nonlinear dynamical systems (Barreto and Araújo, 2004). However, the number of neurons required by the SOM to provide a good approximation of a given input-output mapping is very high, specially when compared to the MLP and RBF neural networks. To alleviate this limitation of the plain SOM algorithm to some extent, we introduce two SOM-based local ARX models.

3 Local ARX Modeling Approaches

In this section, we describe two approaches to the system identification problem that use the SOM as a building block. The basic idea behind both is the partitioning of the input space into non-overlapping regions, called Voronoi cells, whose centroids correspond to the weight vectors of the SOM. Then an interpolating hyperplane is associated with each Voronoi cell or to a small subset of them, in order to estimate the output value of a function.

3.1 Local Linear Mapping

The first architecture to be described is called *Local Linear Mapping* (LLM) (Walter et al., 1990). The basic idea of the LLM is to associate each neuron in the SOM with a conventional FIR/LMS linear filter. The SOM array is used to quantize the input space in a reduced number of prototype vectors (and hence, Voronoi cells), while the filter associated with the winning neuron provides a local estimator of the output of the mapping being approximated.

For the sake of clarity, Figure 1 shows a sketch of the LLM model for a scalar-valued nonlinear function of a single variable, i.e. $y = f(x)$. Note that the global nonlinear function, represented by the curved solid line, is approximated piecewise by three affine functions, associated with the available prototypes w_i , $i = 1, 2$ and 3 . In this example, the prototypes are along the x -axis since they are scalar numbers. For each input x , the linear function associated with the closest prototype w_{i^*} is used to generate the estimated output value \hat{y} .

Thus, for the inverse modeling task of interest, each input vector $\mathbf{x}(t) \in \mathfrak{R}^{p+q}$ is defined as

$$\mathbf{x}(t) = [u(t-1), \dots, u(t-q); y(t-1), \dots, y(t-p)]^T. \quad (9)$$

Clustering (or vector quantization) of the input space X is performed by the LLM as in the usual SOM algorithm, with each neuron i owning a prototype vector \mathbf{w}_i , $i = 1, \dots, S$, where S is the number of neurons of the output array A .

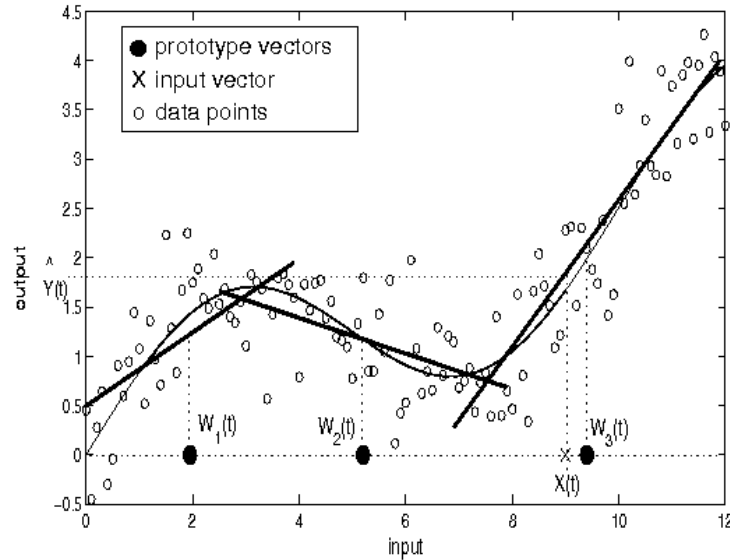


Figure 1: Sketch of the local linear mapping implemented by the LLM architecture.

Additionally, there is a coefficient vector $\mathbf{a}_i(t) \in \mathfrak{R}^{p+q}$ associated to each weight vector \mathbf{w}_i , which plays the role of the coefficients of a (linear) ARX model:

$$\mathbf{a}_i(t) = [b_{i,1}(t), \dots, b_{i,q}(t), a_{i,1}(t), \dots, a_{i,p}(t)]^T. \quad (10)$$

The output value of the LLM-based multiple local ARX model is then computed as

$$\hat{u}(t) = \sum_{k=1}^q b_{i^*,k}(t)u(t-k) + \sum_{l=1}^p a_{i^*,l}(t)y(t-l) = \mathbf{a}_{i^*}^T(t)\mathbf{x}(t), \quad (11)$$

where $\mathbf{a}_{i^*}(t)$ is the coefficient vector of the linear filter associated with the winning neuron $i^*(t)$. From Eq. (11), one can easily note that the coefficient vector $\mathbf{a}_{i^*}(t)$ is used to build a local approximation for the output of the desired nonlinear mapping.

Since the adjustable parameters of the LLM algorithm are the set of prototype vectors $\mathbf{w}_i(t)$ and their associated coefficient vectors $\mathbf{a}_i(t)$, $i = 1, 2, \dots, S$, we need two learning rules. The rule for updating the prototype vectors \mathbf{w}_i follows exactly the one given in Eq. (6). The learning rule of the coefficient vectors $\mathbf{a}_i(t)$ is an extension of the normalized LMS algorithm that also takes into account the influence of the neighborhood function $h(i^*, i; t)$:

$$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \alpha h(i^*, i; t)\Delta\mathbf{a}_i(t), \quad (12)$$

where $0 < \alpha \ll 1$ denotes the learning rate of the coefficient vector, and $\Delta\mathbf{a}_i(t)$ is the error correction rule of Widrow-Hoff (Widrow and Hoff, 1960), given by

$$\Delta\mathbf{a}_i(t) = [u(t) - \mathbf{a}_i^T(t)\mathbf{x}(t)] \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|^2}, \quad (13)$$

where $u(t)$ is the actual output of the inverse nonlinear mapping being approximated. Once trained, the weight vectors \mathbf{w}_i of the SOM and the associated coefficient vectors $\mathbf{a}_i(t)$, $i = 1, 2, \dots, S$, are “frozen”. The LLM-based local ARX model can then be used to estimate the outputs of input-output mapping.

3.2 Prototype-Based Local Least-Squares Model

The algorithm to be described in this section, called K -winners SOM (KSOM), was originally applied to nonstationary univariate time series prediction (Barreto et al., 2004; Barreto et al., 2003). In this paper we aim to evaluate this architecture in the context of nonlinear system identification. For training purposes, the KSOM algorithm depends on the VQTAM (*Vector-Quantized Temporal Associative Memory*) model, proposed by Barreto and Araújo (2004), which is a straightforward extension of the SOM algorithm that can be used for system identification and control purposes. Roughly speaking, the VQTAM is just a SOM algorithm that simultaneously performs vector quantization on the input and output spaces of a given nonlinear mapping.

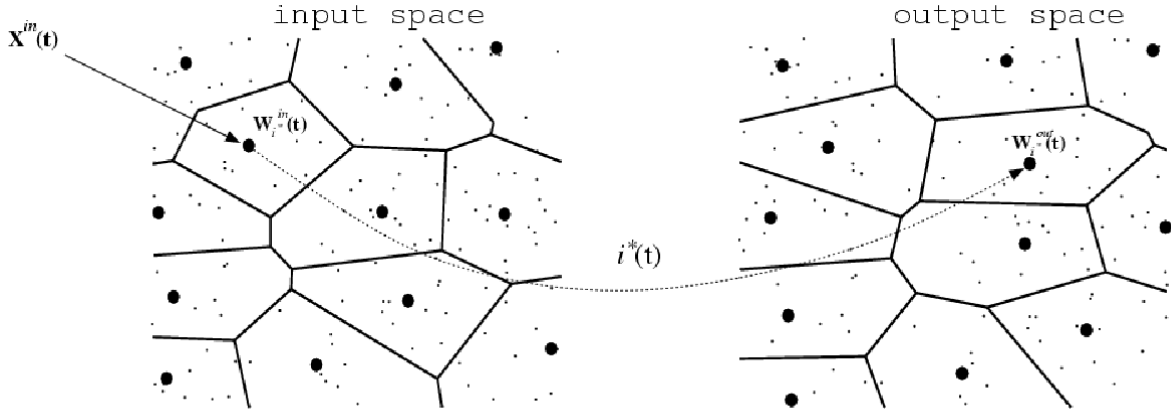


Figure 2: Sketch of the associative mapping implemented by the VQTAM method.

In the VQTAM model, the input vector at time step t , $\mathbf{x}(t)$, is composed of two parts. The first part, denoted $\mathbf{x}^{in}(t) \in \mathfrak{R}^{p+q}$, carries data about the input of the dynamic mapping to be learned. The second part, denoted $x^{out}(t) \in \mathfrak{R}$, contains data concerning the desired (scalar) output of this mapping. The weight vector of neuron i , $\mathbf{w}_i(t)$, has its dimension increased accordingly. These changes are formulated as follows:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ x^{out}(t) \end{pmatrix} \text{ and } \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix}, \quad (14)$$

where $\mathbf{w}_i^{in}(t) \in \mathfrak{R}^{p+q}$ and $w_i^{out}(t) \in \mathfrak{R}$ are, respectively, the portions of the weight vector which store information about the inputs and the outputs of the desired mapping. Depending on the variables chosen to build the vector $\mathbf{x}^{in}(t)$ and the scalar $x^{out}(t)$ one can use the SOM algorithm to learn the forward or the inverse mapping of a given plant (system). For instance, if the interest is in inverse identification, then we define

$$\mathbf{x}^{in}(t) = [u(t-1), \dots, u(t-q); y(t-1), \dots, y(t-p)]^T, \quad (15)$$

and

$$x^{out}(t) = u(t). \quad (16)$$

The winning neuron at time step t is determined based only on $\mathbf{x}^{in}(t)$, i.e.

$$i^*(t) = \arg \min_{\forall i} \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|. \quad (17)$$

For updating the weights, however, both $\mathbf{x}^{in}(t)$ and $x^{out}(t)$ are used:

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t) h(i^*, i; t) [\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)], \quad (18)$$

$$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t) h(i^*, i; t) [x^{out}(t) - w_i^{out}(t)], \quad (19)$$

where $0 < \alpha(t) \ll 1$ is the learning rate, and $h(i^*, i; t)$ is a time-varying gaussian neighborhood function defined as in Eq. (7). In words, the learning rule in Eq. (18) performs topology-preserving vector quantization on the input space, while the rule in Eq. (19) acts similarly on the output space of the mapping being learned.

As training proceeds, the SOM learns to associate the input prototype vectors \mathbf{w}_i^{in} with the corresponding output prototype vectors w_i^{out} (see Fig. 2). The SOM-based associative memory procedure implemented by the VQTAM can then be used for function approximation purposes.

After training the VQTAM model, the coefficient vector $\mathbf{a}(t)$ of a local ARX model for estimating the output of the mapping is computed for each time step t by the standard least-squares estimation (LSE) technique, using the weight vectors of the K neurons closest to the current input vector, instead of using the original data vectors. Note that in order to avoid numerical problems we must have $K \gg 1$.

Let the set of K winning weight vectors at time t to be denoted by $\{\mathbf{w}_{i_1}^*, \mathbf{w}_{i_2}^*, \dots, \mathbf{w}_{i_K}^*\}$. Recall that due to the VQTAM training style, each weight vector $\mathbf{w}_i(t)$ has a portion associated with $\mathbf{x}^{in}(t)$ and other associated with $\mathbf{x}^{out}(t)$. So, the KSOM uses the corresponding K pairs of prototype vectors $\left\{ \mathbf{w}_{i_k}^{in}(t), \mathbf{w}_{i_k}^{out}(t) \right\}_{k=1}^K$, with the aim of building a local function at time t :

$$\mathbf{w}_{i_k}^{out}(t) = \mathbf{a}^T(t) \mathbf{w}_{i_k}^{in}(t), \quad k = 1, \dots, K \quad (20)$$

where $\mathbf{a}(t) = [b_1(t), \dots, b_q(t), a_1(t), \dots, a_p(t)]^T$ is a time-varying coefficient vector. Equation (20) can be written in a matrix form as

$$\mathbf{w}^{out}(t) = \mathbf{R}(t)\mathbf{a}(t), \quad (21)$$

where the output vector $\mathbf{w}^{out}(t)$ and the regression matrix \mathbf{R} at time t are defined as follows

$$\mathbf{w}^{out}(t) = [w_{i_1,1}^{out}, w_{i_2,1}^{out}, \dots, w_{i_K,1}^{out}]^T, \quad (22)$$

and

$$\mathbf{R}(t) = \begin{pmatrix} w_{i_1,1}^{in}(t) & w_{i_1,2}^{in}(t) & \dots & w_{i_1,p+q}^{in}(t) \\ w_{i_2,1}^{in}(t) & w_{i_2,2}^{in}(t) & \dots & w_{i_2,p+q}^{in}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K,1}^{in}(t) & w_{i_K,2}^{in}(t) & \dots & w_{i_K,p+q}^{in}(t) \end{pmatrix}. \quad (23)$$

In practice, we usually have $p+q > K$, i.e. \mathbf{R} is a non-square matrix. In this case, we resort to the pseudo-inverse method (Principe et al., 2000; Haykin, 1994). Thus, the coefficient vector $\mathbf{a}(t)$ is computed by

$$\mathbf{a}(t) = (\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I})^{-1}\mathbf{R}^T(t)\mathbf{w}^{out}(t), \quad (24)$$

where \mathbf{I} is a identity matrix of order K and $\lambda > 0$ is a small regularization constant which is added to the diagonal of $\mathbf{R}^T(t)\mathbf{R}(t)$ to make sure that this matrix is full rank. The parameter λ is found by trial-and-error and it is set to small values to avoid excessively biased estimates of $\mathbf{a}(t)$. It turned out that the KSOM model is relatively insensitive to the choice of λ as long as it is kept small (suggested range: $\lambda \in [0.001, 0.005]$).

Once $\mathbf{a}(t)$ has been computed, one can estimate the output of the nonlinear mapping being approximated by the output of an ARX model:

$$\hat{u}(t) = \sum_{k=1}^q b_k(t) u(t-k) + \sum_{l=1}^p a_l(t) y(t-l) = \mathbf{a}^T(t) \mathbf{x}^{in}(t). \quad (25)$$

Note that, for each time step t , the KSOM builds a single local ARX model over a subset of K weight vectors chosen from the whole set of S weight vectors. Thus, the coefficient vector $\mathbf{a}(t)$ is time-variant. This is one of the differences between KSOM and LLM.

In a sum, while the KSOM-based local ARX model uses $K < S$ prototype vectors to build a single local ARX model with a time-variant coefficient vector, the LLM-based approach builds S local ARX models with time-invariant coefficients (one for each prototype vector). Another difference is that the LLM approach uses a LMS-like learning rule to update the coefficient vector of the winning neuron. Once training is completed all coefficient vectors \mathbf{a}_i , $i=1, 2, \dots, S$, are frozen for posterior use. The KSOM, instead, uses a LSE-like procedure to find the coefficient vector $\mathbf{a}(t)$ each time an input vector is presented, so that a single linear mapping is built at each time step (see Fig. 3).

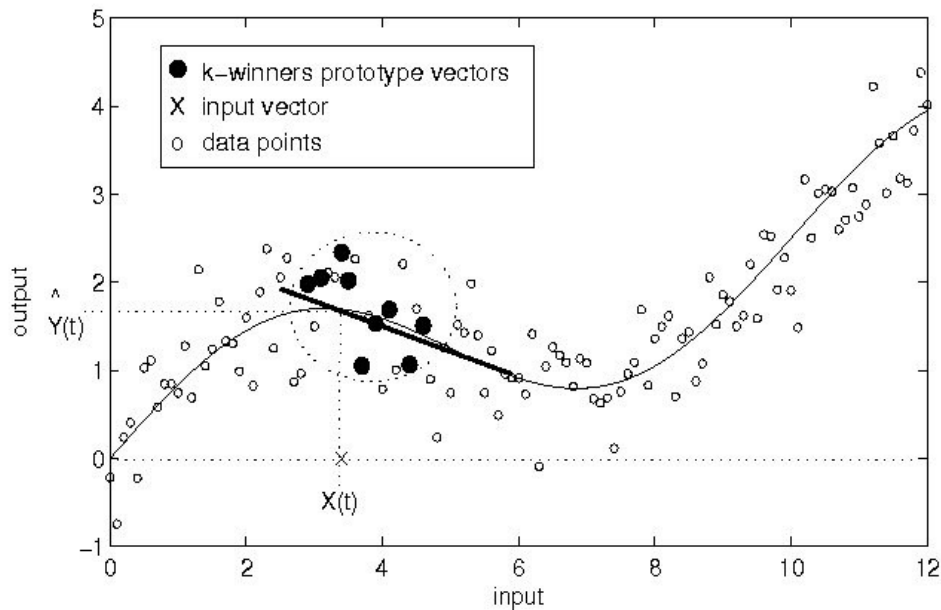


Figure 3: Sketch of the local linear mapping implemented by the KSOM architecture.

Some authors have proposed local modeling approaches that closely resemble the KSOM model (Cho et al., 2006; Principe et al., 1998; Chen and Xi, 1998). Principe et al. (1998) proposed a neural architecture that is equivalent to KSOM in the sense that the coefficient vector $\mathbf{a}(t)$ is computed from K prototype vectors of a trained SOM using the LSE technique. However, the required prototype vectors are not selected as the K nearest prototypes to the current input vector, but rather automatically selected as the winning prototype at time t and its $K-1$ topological neighbors. If the trained SOM presents topological defects, as usually occurs for multidimensional data, the KSOM generally provides more accurate results.

Chen and Xi (1998) also proposed a local regression model whose coefficient vectors are computed using the prototypes of a competitive learning network through the *Recursive Least-Squares* (RLS) algorithm. However, the competitive network used by Chen and Xi does not have the topology-preserving properties of the SOM algorithm, which has shown to be important for system identification purposes (Barreto and Araújo, 2004).

It is worth mentioning that here is no specific difference between the SOM-based local models evaluated in Barreto et al. (2003, 2004) and the ones presented in the current paper. The main difference is related with the kind of application discussed in them. They have different demands. For example, for univariate time series prediction we require only one memory order parameter, while for NARX modeling we need two memory parameters. Another important point is that in Barreto et al. (2003, 2004), the data set was artificially generated chaotic (noise free) time series. In the current paper, both data sets are comprised of real-world noisy input/output time series.

Our focus is in evaluating the local modeling approach on system identification, more specifically, on local NARX modeling for system identification. It should be pointed out that the local modeling approach itself is not novel, it has been used before for time series prediction, system identification and control purposes, as can be observed in many of the cited references. However, application of local modeling to system identification is in its first infancy, in the sense that there is still room for applications and theoretic advances. For example, performance comparison with more traditional global (linear or not) system identification approaches is missing in the literature. It is exactly in this regard that the paper contributes the most, with a performance evaluation of several neural-based models (including different MLP topologies) for local ARX modeling of nonlinear systems.

4 Computer Simulations and Discussion

Hydraulic Actuator: The proposed SOM-based local linear NARX models are evaluated in the identification of the inverse dynamics of a hydraulic actuator and compared with standard MLP-based global NARX models. Figure 4 shows the measured values of the valve position (input time series, $\{u(t)\}$) and the oil pressure (output time series, $\{y(t)\}$). The oil pressure signal sequence shows a highly oscillating behavior caused by mechanical resonances (Sjöberg et al., 1995).

For the sake of completeness, the LLM- and KSOM-based local ARX models are compared with an one-hidden-layer MLP trained by the standard backpropagation algorithm (MLP-1h), another one-hidden-layer MLP trained by the Levenberg-Marquardt (MLP-LM) algorithm and, finally, a two-hidden-layer MLP (MLP-2h) trained by the standard backpropagation algorithm. All these models are also compared with the standard linear *Auto-Regressive with eXogenous variables* (ARX) model, trained on-line through the plain LMS algorithm.

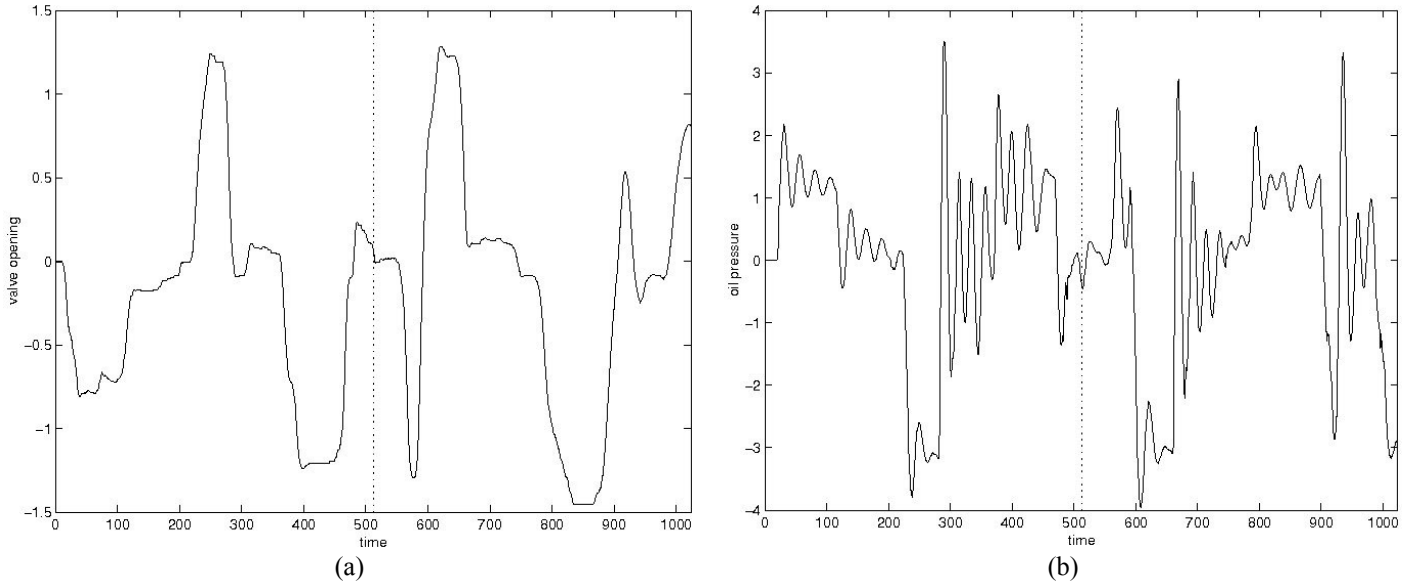


Figure 4: Measured values of valve position (a) and oil pressure (b).

For all MLP-based global NARX models, the transfer function of the hidden neurons is the hyperbolic tangent function, while the output neuron uses a linear one. After some experimentation with the data, the best configuration of the MLP-1h and MLP-LM models have 20 units in the hidden layer. For the MLP-2h, the number of neurons in second hidden layer is heuristically set to half the number of neurons in the first hidden layer. This value for the number of neurons in the second hidden layer turned out to be satisfactory. The learning rate for the MLPs was set to 0.1. No momentum term is used. The programming codes for all models were implemented in Matlab[®] 7.0.

During the estimation (testing) phase, for evaluation purposes, the neural models should compute the estimation error (residuals) $e(t)=u(t) - \hat{u}(t)$, where $u(t)$ is desired output and $\hat{u}(t)$ is the estimate provided by each neural model. For quantitative assessment of the performance of all models accuracy we use the NMSE (*normalized mean squared error*):

$$NMSE = \frac{\sum_{t=1}^M e^2(t)}{M \cdot \hat{\sigma}_u^2} \quad (26)$$

where $\hat{\sigma}_u^2$ is the variance of the original time series $\{u(t)\}$, $t = 1, \dots, M$, where M is the length of the sequence of residuals.

The models are trained using the first 512 samples of the input/output signal sequences and tested with the remaining 512 samples. The input and output memory orders are set to $p = 4$ and $q = 5$, respectively. For each SOM-based model, the initial and final learning rates are set to $\alpha_0 = 0.5$ and $\alpha_T = 0.01$. The initial and final values of radius of the neighborhood function are $\sigma_0 = 0.5S$ and $\sigma_T = 0.001$, where S , the number of neurons in the SOM is set to 20. The learning rate α is set to 0.1.

For the KSOM-based local ARX model, the best number of winning neurons was found to be $K = 15$. In the first simulation, the NMSE values were averaged over 100 training/testing runs, in which the weights of the neural models were randomly initialized at each run. The obtained results are shown in Table 1, which shows the mean, minimum, maximum and variance of the NMSE values, measured along the 100 training/testing runs. In this table, the models are sorted in increasing order of the mean NMSE values.

Neural Models	NMSE			
	<i>mean</i>	<i>min</i>	<i>max</i>	<i>Variance</i>
KSOM	0.0019	0.0002	0.0247	1.15×10^{-5}
LLM	0.0347	0.0181	0.0651	1.58×10^{-4}
ARX	0.0380	0.0380	0.0380	0.0083
MLP-LM	0.0722	0.0048	0.3079	0.0041
MLP-1h	0.3485	0.2800	0.4146	4.96×10^{-4}
MLP-2h	0.3516	0.0980	2.6986	0.0963

One can easily note that the performances of KSOM- and LLM-based local models on this real-world application are far better than those of MLP-based global models. The better performance of the KSOM-based model in comparison to the LLM-based is due to the use of the LSE algorithm to estimate the coefficient vector $\mathbf{a}(t)$, as indicated in Eq. (24). Among the MLP-based global models, the use of second-order information also explains the better performance of the MLP-LM, which uses curvature information extracted from an approximation of the Hessian matrix.

Figure 5 shows typical sequences of estimated values of the valve position provided by the best local ARX and global NARX models. Figure 5a shows the sequence generated by the KSOM-based model, while Figure 5b shows the sequence estimated by the MLP-LM model. One can note that the local approach provides better approximation in regions of higher curvature, e.g. around the time steps 350, 410 and 500. These regions are highlighted in the figures and denoted R1, R2 and R3, respectively.

Flexible Robotic Arm: An additional data set is used to evaluate all the aforementioned models in the estimation of the inverse dynamics of a flexible robot arm. The arm is installed on an electric motor. It had modeled the transfer function from the measured reaction torque of the structure on the ground to the acceleration of the flexible arm¹. Figure 6 shows the measured values of the reaction torque of the structure (input time series, $\{u(t)\}$) and the acceleration of the flexible arm (output time series, $\{y(t)\}$).

The number of neurons for all SOM-based local ARX models is set to 30. For the KSOM model, K is now equal to 25. For each SOM-based model, the training parameters are the same as those used for the hydraulic actuator data set. The MLP-1h and MLP-LM models have 30 hidden neurons, while the MLP-2h has 30 and 15 neurons in the first and second hidden layers, respectively. The MLP-based models used a constant learning rate equal to 0.1. No momentum term is used. The LMS algorithm is used again for training the linear ARX model.

The models are trained using the first 820 samples of the input-output signal sequences (approximately, 80% of the total) and tested with the remaining 204 samples. The input and output memory orders are set to $p = 4$ and $q = 5$, respectively. The obtained results are shown in Table 2, where are displayed the mean, minimum, maximum and variance of the NMSE values, measured along the 100 training/testing runs. The weights of the neural models were randomly initialized at each run. In this table, the models are again sorted in increasing order of the mean NMSE values.

Neural Models	NMSE			
	<i>mean</i>	<i>min</i>	<i>max</i>	<i>Variance</i>
KSOM	0.0064	0.0045	0.0117	1.83×10^{-6}
MLP-LM	0.1488	0.0657	0.4936	0.0107
MLP-1h	0.1622	0.1549	0.1699	1.03×10^{-5}
LLM	0.3176	0.2685	0.3558	2.23×10^{-4}
ARX	0.3848	0.3848	0.3848	0.0445
MLP-2h	0.6963	0.5978	1.5310	0.0368

¹ These data were obtained in the framework of the Belgian Programme on Interuniversity Attraction Poles (IUAP)-nr.50 initiated by the Belgian State (available at <http://homes.esat.kuleuven.be/~smc/daisy/daisydata.html>).

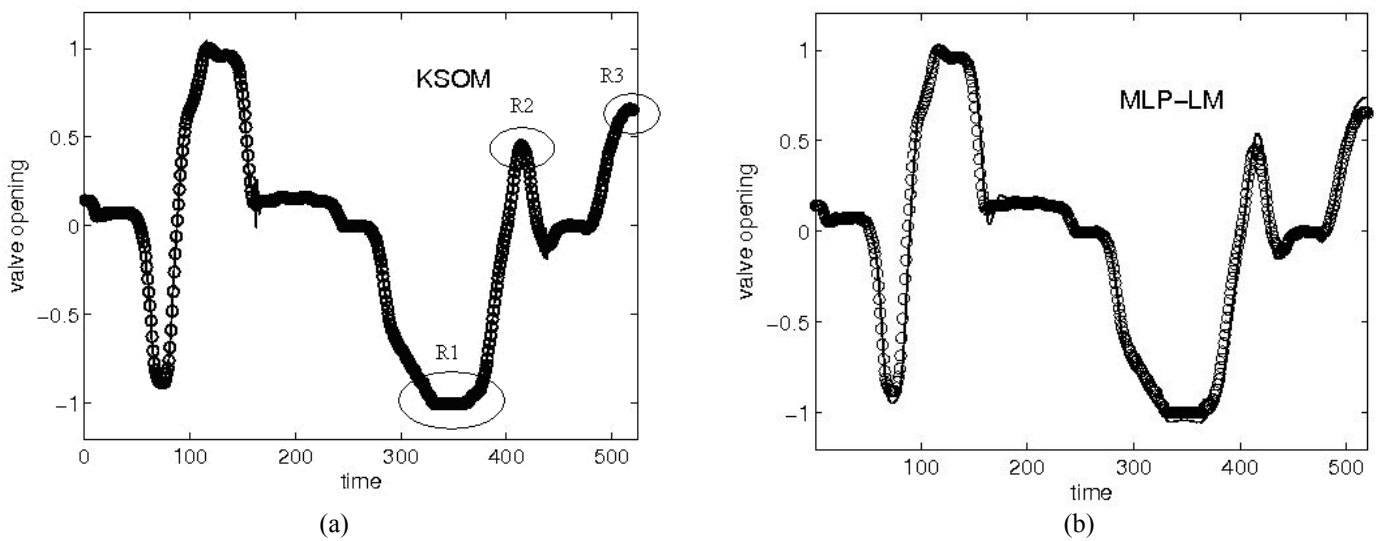


Figure 5: Typical estimated sequences of the valve position provided by the KSOM and MLP-LM models. Open circles 'o' denote actual sample values, while the solid line indicates the estimated sequence.

Again, the performance of KSOM-based local ARX model on this real-world application is far better than those of MLP-based global models. The better performance of the KSOM-based model in comparison to the LLM-based model remains, but this time the LLM model performed worse than the MLP-LM and MLP-1h models, being only better than the ARX and MLP-2h models. As before, time the use of second-order information was crucial to the good performance of the MLP-LM model with respect to the other MLP-based models.

Figure 7 shows typical sequences of estimated values of the reaction torque of the structure provided by the best local ARX and global NARX models. Figure 7a shows the sequence generated by the KSOM model, while Figure 7b shows the sequence estimated by the MLP-LM model.

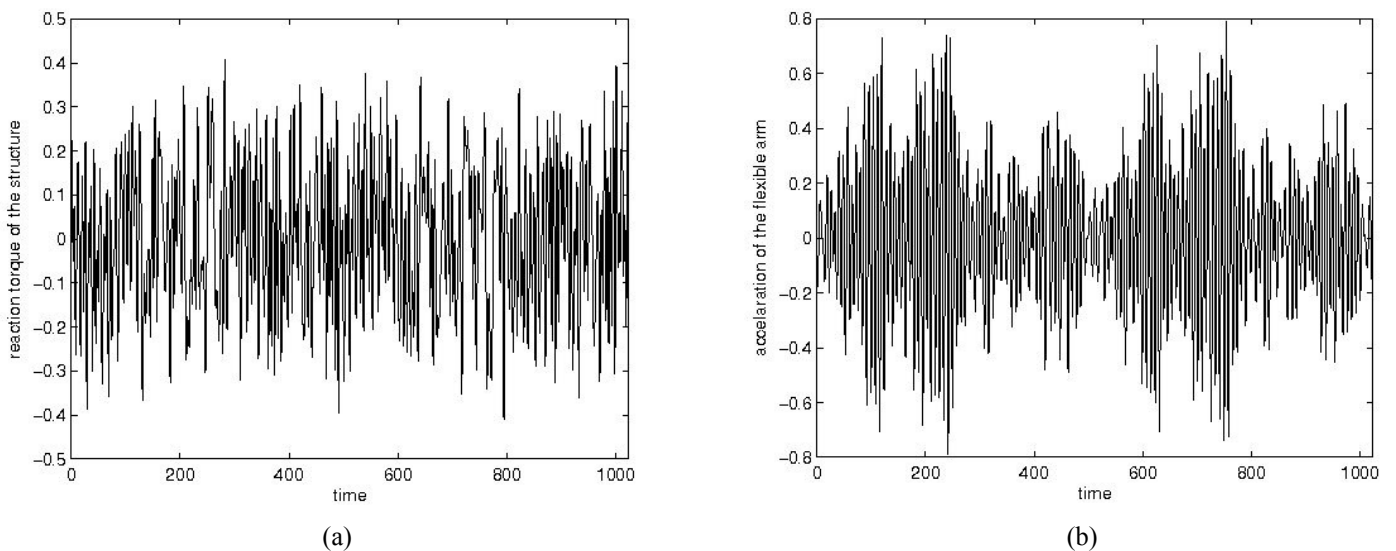


Figure 6: Measured values of reaction torque of the structure (a) and acceleration of the flexible robotic arm (b).

4 Conclusion

In this paper, we have attempted to tackle the problem of nonlinear system identification using the local modeling methodology. For that purpose we evaluated two local ARX models based on Kohonen's self-organizing map in the identification of the inverse dynamics of two real-world data sets, a hydraulic actuator and a flexible robotic arm. The first local ARX model proposed builds multiple linear ARX models, one for each Voronoi region associated with the prototype vectors of the SOM.

The second one builds only a single ARX model using the prototypes vectors closest to the current input vector. It has been shown for the hydraulic actuator data set that the local ARX models outperform the conventional MLP-based global NARX models, and for the robot arm data set, considerably more difficult than the first data set, the KSOM-based local NARX model presented the best performance among all models. Currently, we are evaluating the SOM-based local ARX modeling approach on other data sets for benchmarking purposes. Also, we are assessing their computational complexity (i.e. the number of operations required) during both training and testing phases.

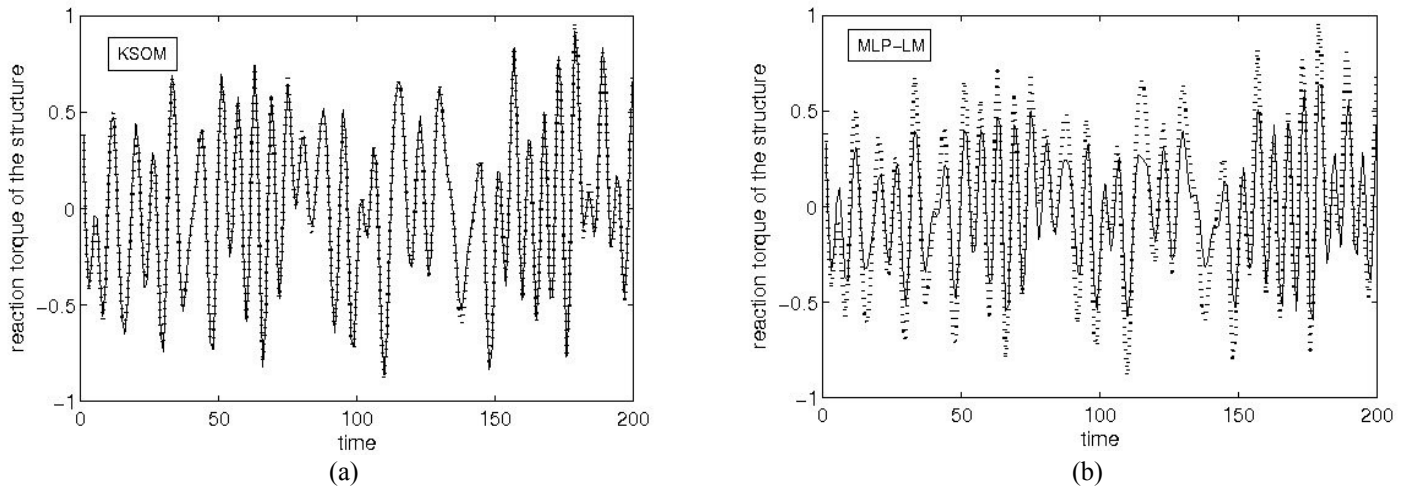


Figure 7: Typical estimated sequences of reaction torque of the structure provided by the KSOM and MLP-LM models. Dashed lines denote actual sample values, while the solid line indicates the estimated sequence.

5 Acknowledgements

The authors would like to thank FUNCAP (grant #1469/07) and CAPES/PRODOC for the financial support.

6 References

- Barreto, G. A. and Araújo, A. F. R. (2001). Time in self-organizing maps: an overview of models, *International Journal of Computer Research* 10(2): 139–179.
- Barreto, G. A. and Araújo, A. F. R. (2004). Identification and control of dynamical systems using the self-organizing map, *IEEE Transactions on Neural Networks* 15(5): 1244–1259.
- Barreto, G. A., Araújo, A. F. R. and Ritter, H. J. (2003). Self-organizing feature maps for modeling and control of robotic manipulators, *Journal of Intelligent and Robotic Systems* 36(4): 407–450.
- Barreto, G. A., Mota, J. C. M., Souza, L. G. M. and Frota, R. A. (2003). Previsão de séries temporais não-estacionárias usando modelos locais baseados em redes neurais competitivas, *Anais do VI Simpósio Brasileiro de Automação Inteligente (SBAI'03)*, pp. 941–946.
- Barreto, G., Mota, J., Souza, L. and Frota, R. (2004). Nonstationary time series prediction using local models based on competitive neural networks, *Lecture Notes in Computer Science* 3029: 1146–1155.
- Chen, J.-Q. and Xi, Y.-G. (1998). Nonlinear system modeling by competitive learning and adaptive fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics-Part C* 28(2): 231–238.
- Chen, S., Billings, S. A. and Grant, P. M. (1990). Nonlinear system identification using neural networks, *International Journal of Control* 51: 1191–1214.

Cho, J., Principe, J., Erdogmus, D. and Motter, M. (2006). Modeling and inverse controller design for an unmanned aerial vehicle based on the selforganizing map, *IEEE Transactions on Neural Networks* 17(2): 445–460.

Cho, J., Principe, J., Erdogmus, D. and Motter, M. (2007). Quasi-sliding mode control strategy based on multiple linear models, *Neurocomputing* 70(4-6): 962–974.

Díaz-Blanco, I., Cuadrado-Vega, A. A., Díez-González, A. B., Fuertes-Martínez, J. J., Domínguez-González, M. and Reguera-Acevedo, P. (2007). Visualization of dynamics using local dynamic modelling with self-organizing maps, *Lecture Notes on Computer Science* 4668: 609–617.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*, Macmillan Publishing Company, Englewood Cliffs, NJ.

Ibnkahla, M. (2003). Nonlinear System Identification Using Neural Networks Trained with Natural Gradient Descent, *EURASIP Journal on Applied Signal Processing* (12):1229-1237.

Kohonen, T. K. (1997). *Self-Organizing Maps*, 2nd extended edition, Springer-Verlag, Berlin, Heidelberg.

Lan, J., Cho, J., Erdogmus, D., Principe, J. C., Motter, M. A. and Xu, J. (2005). Local linear PID controllers for nonlinear control, *International Journal of Control and Intelligent Systems* 33(1): 26–35.

Ljung, L. (1999). *System Identification: Theory for the User*, 2nd edn, Prentice-Hall, Englewood Cliffs, NJ.

Midenet, S. and Grumbach, A. (1994). Learning associations by self-organization: the LASSO model, *Neurocomputing* 6: 343–361.

Narendra, K. S. and Lewis, F. L. (2001). Special issue on neural networks feedback control, *Automatica* 37(8).

Pinkus, A. (1999). Approximation theory of the MLP model in neural networks, *Acta Numerica*, Vol. 8, pp. 143–195.

Principe, J. C., Euliano, N. R. and Garani, S. (2002). Principles and networks for self-organization in space-time, *Neural Networks* 15(8–9): 1069–1083.

Principe, J. C., Euliano, N. R. and Lefebvre, W. C. (2000). *Neural Adaptive Systems: Fundamentals Through Simulations*, John Wiley & Sons, New York, NY.

Principe, J. C., Wang, L. and Motter, M. A. (1998). Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control, *Proceedings of the IEEE* 86(11): 2240–2258.

Purwar, S., Kar, I. N. and Jha, A. N. (2007). On-line system identification of complex systems using Chebyshev neural networks, *Applied Soft Computing* 7(1):364-372.

Schilling, R. J., Jr., J. J. C. and Al-Ajlouni, A. F. (2001). Approximation of nonlinear systems with radial basis function neural networks, *IEEE Transactions on Neural Networks* 12: 1–15.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Deylon, B., Glorennec, P.-Y., Hjalmarsson, H. and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: A unified overview, *Automatica* 31(12): 1691–1724.

Walter, J. and Ritter, H. (1996). Rapid learning with parametrized self-organizing maps, *Neurocomputing* 12: 131–153.

Walter, J., Ritter, H. and Schulten, K. (1990). Nonlinear prediction with self-organizing map, *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'90)*, Vol. 1, pp. 587–592.

Wang, D. L. (2003). *The Handbook of Brain Theory and Neural Networks*, 2nd edn, MIT Press, chapter Temporal pattern processing, pp. 1163–1167.

Wang, D. L. and Arbib, M. A. (1990). Complex temporal sequence learning based on short-term memory, *Proceedings of the IEEE* 78: 1536–1543.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits, *IRE WESCON Convention Record-Part 4*, pp. 96–104.