LOCAL LINEAR NARX MODELS BASED ON THE SELF-ORGANIZING MAP

Luís Gustavo M. Souza*, Guilherme A. Barreto*

Emails: luisgustavo@deti.ufc.br, guilherme@deti.ufc.br

Abstract— In this paper, we develop efficient local linear NARX (Nonlinear Auto-Regressive model with eXogenous variables) models based on Kohonen's Self-Organizing Map. These models are evaluated as function approximators in the task of identifying the inverse dynamics of a hydraulic actuator. Simulation results demonstrate that the proposed SOM-based linear NARX structures consistently outperform standard MLP-based global models for system identification.

Keywords— Kohonen map, local linear models, NARX models, system identification, hydraulic actuator.

Resumo— Neste trabalho, são desenvolvidos modelos NARX localmente lineares a partir da rede auto-organizável de Kohonen. Estes modelos são avaliados como aproximadores de funções na tarefa de identificação da dinâmica inversa de um atuador hidráulico. Simulações computacionais demonstram que as estruturas NARX propostas têm desempenho superior ao de modelos NARX globais baseados na rede MLP.

Keywords— Mapa de Kohonen, modelos lineares locais, modelo NARX, identificação de sistemas, atuador hidráulico.

1 Introduction

Many control schemes are dependent on the availability of a good model of the system to be controlled (Norgaard et al., 2000). For any such scheme to work sufficiently well, the system model obtained by the system identification procedure must be of a certain standard. The system identification procedure therefore needs to be chosen with care.

In producing a model of a particular system, essentially the system is being approximated by the model in terms of both structure and parameters. Since neural networks have been shown to be universal approximators (Hornik et al., 1989), they can be used for the purposes of system identification. As a technique in general, they may be able to provide an universal means for modeling nonlinear systems, assuming network training techniques and computational complexity do not restrict their use.

Identification of nonlinear dynamic systems with neural networks was consistently evaluated by Narendra and Parthasarathy (1990) who, under mild conditions, have shown that such NARX neural networks are able to solve difficult function approximation problems. Since then, neural-based system identification procedures have been dominated by standard supervised architectures, such as the Multilayer Perceptron (MLP) and the Radial Basis Functions (RBF) networks.

More recently, the self-organizing map (SOM) has emerged as a viable alternative to more traditional neural-based approaches to identification and control of nonlinear dynamical systems (Barreto and Araújo, 2004; Principe et al., 1998; Kohonen et al., 1996). In this paper, local linear models based on the SOM are developed and used as NARX models to approximate the inverse dynamics of an hydraulic actuator. Local models have been a source of much interest because they have the ability to adhere to the local shape of an arbitrary surface, which is difficult especially in cases when the dynamical system characteristics vary considerably throughout the state space.

The technique of local linear neural models is closely related to the piecewise linear modelling approach, widely used within the field of adaptive control. The idea of local linear modelling is to specify a set of hyperplanes (i.e. adaptive filters), each of which is only locally used, in order to closely approximates nonlinear surface that defines the input-output dynamics of the process of interest.

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 linear NARX 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 quantize the input space by discovering a set of representative prototype (reference) vectors. 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. A competitive learning drives the winning weight vector to become more similar to the input data. Throughout this paper, we define the weight vector between input layer and neuron i as follows:

$$\mathbf{w}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,j}, \ldots, w_{i,p+q})^T \quad (1)$$

where $w_{i,j} \in \mathbb{R}$ denotes the weight connecting node j in the input layer with neuron i, and $(p+q)$, is the dimension of the input vector. In what follows, a brief description of the original SOM algorithm is given.

Firstly we use Euclidean distance metric to find the current winning neuron, $i^*(t)$, as given by the following expression:

$$i^*(t) = \arg\min_{\forall i} \|\mathbf{x}(t) - \mathbf{w}_i(t)\| \quad (2)$$

where $\mathbf{x}(t) \in \mathbb{R}^{p+q}$ denotes the current input vector, $\mathbf{w}_i(t) \in \mathbb{R}^{p+q}$ is the weight vector of neuron i, and t symbolizes 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:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \quad (3)$$

where $0 < \alpha(t) < 1$ is the learning rate and $h(i^*, i; t)$ is a gaussian weighting function that limits the neighborhood of the winning neuron:

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right) \quad (4)$$

where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$, are respectively, the positions of neurons i and i^* in a predefined output array 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 for both an exponential decay, given by:

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_T}{\alpha_0}\right)^{(t/T)} \text{ and } \sigma(t) = \sigma_0 \left(\frac{\sigma_T}{\sigma_0}\right)^{(t/T)} \quad (5)$$

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 by Eqs. (2) and (5) 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 patterns are mapped into adjacent regions 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 abilities 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 somewhat alleviate this limitation of the plain SOM algorithm, we introduce next two SOM-based local linear NARX models.

3 Local Linear NARX Models

In this section, we describe two approaches to the system identification problem that uses the SOM as a building block. The basic idea behind both is the segmentation of the input space into nonoverlapping 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 set of them, in order to estimate the output of the process.

Let us assume that the dynamical system we are dealing with can be described by the NARX model:

$$\begin{aligned} y(t) \quad = \quad & f[y(t-1), \ldots, y(t-p); u(t), \\ & u(t-1), \ldots, u(t-q+1)] \end{aligned} \quad (6)$$

where $f(\cdot)$ is an unknown nonlinear mapping, and q and p, $q < p$, are the orders of the input and output regressors, respectively. However, for the inverse modelling task we are interested in, the neural network models should implement the inverse mapping $f^{-1}(\cdot)$, given by:

$$u(t) = f^{-1}[u(t-1), \ldots, u(t-q); y(t-1), \ldots, y(t-p)] \quad (7)$$

whose goal is to estimate the input of a given system based on previous values of the input and output variables. This kind of nonlinear inverse model of a system and the corresponding online identification of its parameters are useful, for example, for real-time control purposes (Norgaard et al., 2000).

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 linear estimator of the output of the mapping being approximated.

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

$$\mathbf{x}(t) = [u(t-1), \ldots, u(t-q); y(t-1), \ldots, y(t-p)]^T \quad (8)$$

Clustering (or vector quantization) of the input space $\mathcal{X}$ is performed by the LLM as in the usual SOM algorithm, with each neuron i owning a prototype vector $\mathbf{w}_i$, $i = 1, \ldots, N$.

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

$$\mathbf{a}_i(t) = [b_{i,1}(t), \ldots, b_{i,q}(t), a_{i,1}(t), \ldots, a_{i,p}(t)]^T \quad (9)$$

The output value of the LLM-based NARX model is then computed as follows:

$$\begin{aligned} \hat{u}(t) &= \sum_{m=1}^{q} b_{i^*,m}(t)u(t-m) + \sum_{l=1}^{p} a_{i^*,l}(t)y(t-l) \\ &= \mathbf{a}_{i^*}^T(t)\mathbf{x}(t) \end{aligned} \quad (10)$$

where $\mathbf{a}_{i^*}(t)$ is the coefficient vector associated to the winning neuron which is used to build a local linear approximation of the output of the desired nonlinear mapping.

The rule for updating the prototype vectors $\mathbf{w}_i$ follows exactly the one given in Eq. (3). The learning rule of the coefficient vectors $\mathbf{a}_i(t)$ is an extension of the plain LMS normalized algorithm, that also takes into account the influence of the neighborhood function $h(i^*, i; t)$:

$$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \alpha' h(i^*, i; t)\Delta\mathbf{a}_i(t) \quad (11)$$

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:

$$\Delta\mathbf{a}_i(t) = \left[u(t) - \mathbf{a}_i^T(t)\mathbf{x}(t)\right] \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|^2} \quad (12)$$

where $u(t)$ is the actual output of the inverse nonlinear mapping being approximated.

3.2 Prototype-Based Local Least-Squares Regression Model

The algorithm to be described in this section, called K-winners SOM (KSOM), was originally applied to nonstationary 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 Barreto and Araújo (2004), which is a simple 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.

In the VQTAM model, the input vector at time step t, $\mathbf{x}(t)$, is composed of two parts. The first part, denoted $\mathbf{x}^{in}(t) \in \mathbb{R}^{p+q}$, carries data about the input of the dynamic mapping to be learned. The second part, denoted $x^{out}(t) \in \mathbb{R}$, contains data concerning the desired 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:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ x^{out}(t) \end{pmatrix} \text{ and } \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix} \quad (13)$$

where $\mathbf{w}_i^{in}(t) \in \mathbb{R}^{p+q}$ and $w_i^{out}(t) \in \mathbb{R}$ are, respectively, the portions of the weight (prototype) 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 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 the inverse identification, then one defines:

$$\begin{aligned} \mathbf{x}^{in}(t) &= [u(t-1), \ldots, u(t-q); \\ &\qquad y(t-1), \ldots, y(t-p)]^T \end{aligned} \quad (14)$$

$$x^{out}(t) = u(t) \quad (15)$$

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

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}. \quad (16)$$

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

$$\Delta\mathbf{w}_i^{in}(t) = \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)] \quad (17)$$

$$\Delta w_i^{out}(t) = \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)] \quad (18)$$

where $0 < \alpha(t) < 1$ is the learning rate, and $h(i^*, i; t)$ is a time-varying Gaussian neighborhood function defined as in Eq. (4). In words, the learning rule in Eq. (17) performs topology-preserving vector quantization on the input space and the rule in Eq. (18) acts similarly on the output space of the mapping being learned.

After training the VQTAM model, the coefficient vector $\mathbf{a}(t)$ of a linear NARX 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, $K \geq 1$, neurons closest to the current input vector, instead of using the original data vectors.

Let the set of K winning weight vectors at time t to be denoted by $\{\mathbf{w}_{i_1^*}, \mathbf{w}_{i_2^*}, \ldots, \mathbf{w}_{i_K^*}\}$. 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 $x^{out}(t)$. So, the KSOM uses the corresponding K pairs of prototype vectors $\{\mathbf{w}^{in}_{i^*_k}(t), w^{out}_{i^*_k}(t)\}_{k=1}^{K}$, with the aim of building a local linear function approximator at time t:

$$w^{out}_{i^*_k} = \mathbf{a}^T(t)\mathbf{w}^{in}_{i^*_k}(t), \qquad k = 1, \ldots, K \tag{19}$$

where $\mathbf{a}(t) = [b_1(t), \ldots, b_q(t), a_1(t), \ldots, a_p(t)]^T$ is a time-varying coefficient vector. Equation (19) can be written in a matrix form as:

$$\mathbf{w}^{out}(t) = \mathbf{R}(t)\mathbf{a}(t) \tag{20}$$

where the output vector $\mathbf{w}^{out}$ and the regression matrix $\mathbf{R}$ at time t are defined by the following equations:

$$\mathbf{w}^{out}(t) = [w^{out}_{i^*_1,1}(t)\ w^{out}_{i^*_2,1}(t)\ \cdots\ w^{out}_{i^*_K,1}(t)]^T \tag{21}$$

$$\mathbf{R}(t) = \begin{pmatrix} w^{in}_{i^*_1,1}(t) & w^{in}_{i^*_1,2}(t) & \cdots & w^{in}_{i^*_1,p+q}(t) \\ w^{in}_{i^*_2,1}(t) & w^{in}_{i^*_2,2}(t) & \cdots & w^{in}_{i^*_2,p+q}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w^{in}_{i^*_K,1}(t) & w^{in}_{i^*_K,2}(t) & \cdots & w^{in}_{i^*_K,p+q}(t) \end{pmatrix} \tag{22}$$

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 given by:

$$\mathbf{a}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{w}^{out}(t) \tag{23}$$

where $\mathbf{I}$ is a identity matrix of order K and $\lambda > 0$ (e.g. $\lambda = 0.001$) is a small constant added to the diagonal of $\mathbf{R}^T(t)\mathbf{R}(t)$ to make sure that this matrix is full rank. Once $\mathbf{a}(t)$ has been computed, we can estimate the output of the nonlinear mapping being approximated by the output of a NARX model:

$$\begin{aligned} \hat{u}(t) &= \sum_{m=1}^{q} b_m(t)u(t-m) + \sum_{l=1}^{p} a_l(t)y(t-l) \\ &= \mathbf{a}^T(t)\mathbf{x}^{in}(t) \end{aligned} \tag{24}$$

Note that the KSOM is considered a local linear NARX model due to the use of a subset of K weight vectors chosen from the whole set of N weight vectors. This is also one of the main differences between KSOM and the LLM approaches. While the former uses $K \ll N$ prototype vectors to build the local linear model, the latter uses a single prototype. Another main 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 $\mathbf{a}_i$, $i = 1, \ldots, N$, are freezed for posterior use. The KSOM, instead, uses a LSE-like procedure to find a single the coefficient vector $\mathbf{a}(t)$, so that the linear mapping is built dynamically at each time step.

Some authors have proposed local linear approaches that closely resemble the KSOM model (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 topological defects are present, as usually occurs for multidimensional data, the KSOM provides more accurate results.

Chen and Xi (1998) also proposed a local linear 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).

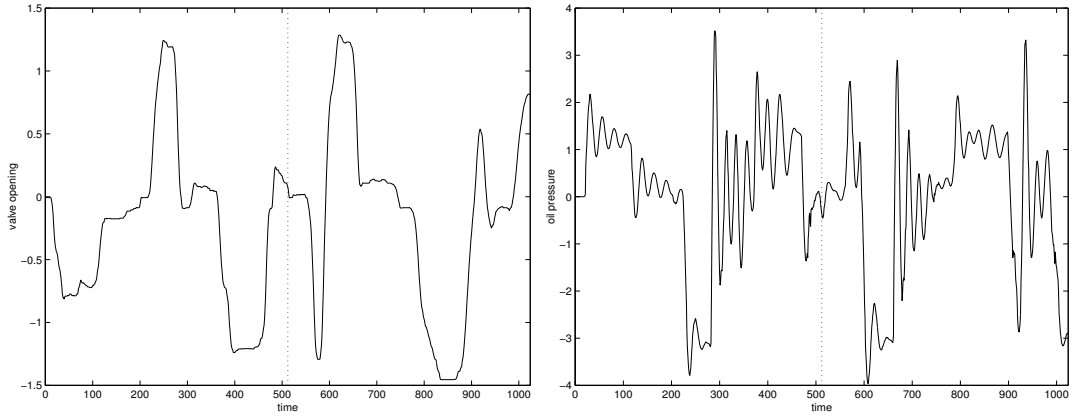
To the best of our knowledge, no performance comparison between the LLM and KSOM approaches as nonlinear system identification tools have been reported in the literature. In this sense, this is one of the main contributions of this paper.

4 Simulations

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 1 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).

The LLM- and KSOM-based local linear NARX models are compared with an one-hidden-layer MLP trained by the backpropagation algorithm (MLP-1h), another one-hidden-layer MLP trained by the Levenberg-Marquardt (MLP-LM) algorithm and, finally, a two-hidden-layers MLP (MLP-2h) trained by the backpropagation algorithm. For all MLP-based global NARX models, the transfer function of the neurons of the hidden layers 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 sec-



(a) (b)

Figure 1: Measured values of valve position (a) and oil pressure (b).

ond layer is heuristically set to half the number of neurons in the first hidden layer. The MLP was trained with constant learning rate equal to 0.1. No momentum term is used.

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 of each neural model, and $\hat{u}(t)$ is the estimates provided by each neural model. For quantitative assessment of the performance of all neural-based NARX models accuracy we use the NMSE (*normalized mean squared error*):

$$NMSE = \frac{\sum_{t=1}^{M} e^2(t)}{M \cdot \hat{\sigma}_u^2} = \frac{\hat{\sigma}_e^2}{\hat{\sigma}_u^2} \qquad (25)$$

where $\hat{\sigma}_u^2$ is the variance of the original time series $\{u(t)\}_{t=1}^{M}$ and 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 $\sigma_0 = N/2$ and $\sigma_T = 0.001$, where N, the number of neurons in the SOM, is set to 20. The learning rate α' is set to 0.01.

For the KSOM-based 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, where are displayed the mean, minimum, maximum and variance of the NMSE values, measured along the 100 training/testing runs. In this table, the models are sorted according to the mean NMSE values.

Table 1: Performance of the MLP-based and local linear models using real-world data.

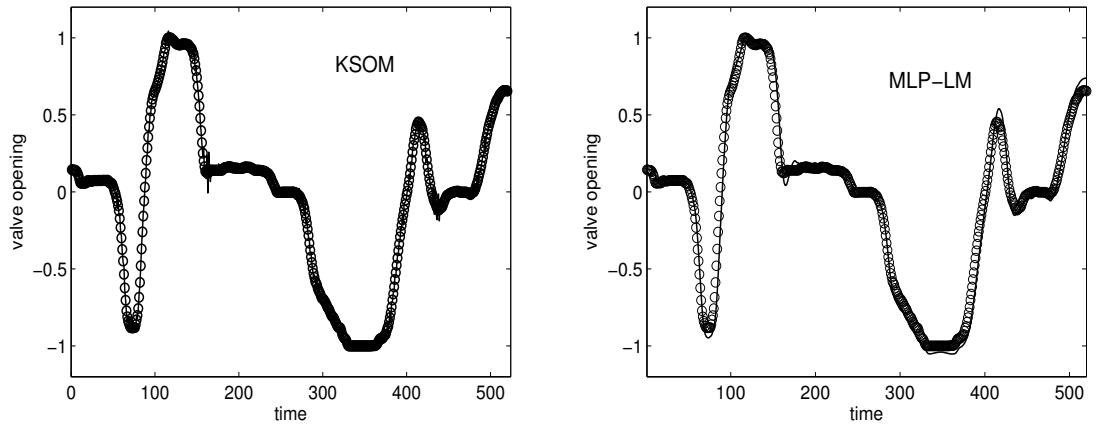
Neural Models	**NMSE**			
	mean	*min*	*max*	*variance*
KSOM-15	0.0019	0.0002	0.0247	1.15×10^{-5}
LLM	0.0347	0.0181	0.0651	1.58×10^{-4}
MLP-LM	0.0722	0.0048	0.3079	0.0041
MLP-1h	0.3485	0.2800	0.4146	4.96×10^{-4}
MLP-2h	0.3516	0.0980	2.6986	0.0963

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 a second-order algorithm to estimate the coefficient vector $\mathbf{a}(t)$, as indicated in Eq. (23). Among the MLP-based global models, the use of second-order information also explains the better performance of the MLP-LM, which uses information extracted from the Hessian matrix.

For a simple evaluation of the results in qualitative terms, Figure 2 shows typical sequences of estimated values of the valve position provided by the best local and global NARX models. Figure 2a shows the sequence generated by the KSOM-based model, while Figure 2b shows the sequence estimated by the MLP-LM model.

5 Conclusion

In this paper, we have attempted to tackle the problem of nonlinear system identification using the local linear modelling methodology. For that purpose we introduced two local linear NARX models based on Kohonen's self-organizing map



(a) (b)

Figure 2: Typical estimated sequences of the valve position provided by the KSOM and MLP-LM models. Open circles '∘' denote actual sample values, while the solid line indicates the estimated sequence.

and evaluated them in the identification of the inverse dynamics of a hydraulic actuator. The first local NARX model proposed builds a fixed number of linear LMS filters, one for each Voronoi region associated with the prototype vectors of the SOM. The second one builds only a single LMS/Newton linear filter using the prototypes vectors closest to the current input vector, it has been shown that the proposed local linear NARX models consistently outperform the conventional MLP-based global NARX models.

Acknowledgment

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

References

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., 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.

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

Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks* **2**(5): 359–366.

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

Kohonen, T. K., Oja, E., Simula, O., Visa, A. and Kangas, J. (1996). Engineering applications of the self-organizing map, *Proceedings of the IEEE* **84**(10): 1358–1384.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks* **1**(1): 4–27.

Norgaard, M., Ravn, O., Poulsen, N. K. and Hansen, L. K. (2000). *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag.

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

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

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