Adaptive Inverse Control Using SOM based Multiple Models

Geetha K. Thampi[1], Jose C. Principe[1], JeongHo Cho[1], Deniz Erdogmus[1], Mark A. Motter[2]

[1] *Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL 32611*
[2] *NASA Langley Research Center, Hampton, Virginia, 23681-2199*

Abstract: This paper addresses the problem of modeling and controlling non-linear plants by utilizing a self-organizing map (SOM). The proposed method uses multiple models of the non-linear plant for identification, as well as a multi-model controller. The mathematical formulation of the controller algorithm and the switching strategy are discussed. Simulation and experimental studies are included for a variety of non-linear systems to demonstrate the performance of the proposed strategy.

Keywords: SOM, PE, Controller, Non-linear, Identification.

1. INTRODUCTION

It has been widely believed that most of the dynamical systems we encounter in practice exhibit non-linear behavior. The linearization of these non-linear systems is done based on the assumption that the plant normally operates near the equilibrium. But in many practical cases the system may be required to operate in the state space where the linear approximations are no longer valid. This gave rise to the notion of using multiple models to represent the system dynamics over the entire operating regime. System identification using multiple model method has received considerable attention in recent years due to its divide and conquer approach. In this approach, a set of models is designed to represent the possible system behavior.

The concept of multiple models and switching between the models has been an area of interest in Control Theory. Multiple Kalman filter-based models (Lainiotis, 1976) were studied in the past to improve the accuracy in state estimation and control problems. In recent years (Narendra *et al*, 1997, 1997, 1995) multiple models using neural networks have been used with switching between the models. Principe (Principe *et al,* 1998) have successfully modeled a chaotic time series using multiple models and applied it to the set point regulation of a NASA Langley wind tunnel during the aerodynamic testing of model aircraft (Motter, 1997). Inspired by this approach, we propose a method using self-organizing map (SOM) (Kohonen, 1995) to identify the plant at different segments of the state space trajectory, giving rise to the concept of multiple models structured by an SOM. Here the global dynamics is approximated by a preset number of local linear models, which are concurrently derived through competition using Kohonen's self-organizing map (SOM). The local models are derived from the weights of the SOM. At any time instant, the model representing the plant dynamics is chosen by the SOM depending on the state of the system.

Tracking based on inverse control constitutes one of the active areas of research in control theory. The basic objective of adaptive inverse control is to determine the control input such that the system output follows a specified trajectory. Since we identify the plant using multiple models, it is necessary to associate these models with a corresponding controller. How we do this in the presence of non-linearity poses a challenge that will be addressed in this paper.

2. PROBLEM STATEMENT

Consider a dynamical system represented by the state equation

$$\sum : y(n+1) = f(\mathbf{u}(n), \mathbf{y}(n)) \quad (1)$$

where the vector $\mathbf{u}(n)$ denotes the delayed input sequence, $\mathbf{u}(n) = [u(n)\, u(n-1) \ldots u(n-M)]^T$ and the vector $\mathbf{y}(n)$ is the delayed output sequence, $\mathbf{y}(n) = [y(n)\, y(n-1) \ldots y(n-N)]^T$ with $u(n) \in \Re, y(n) \in \Re, f : \Re^n \times \Re^m \to \Re$, where f can be any non-linear function. Qualitatively, the objective of the inverse control problem is to determine an input control law $u(\cdot)$ such that the output $y(\cdot)$ behaves in some desired fashion. As seen, the design of the controller has two phases: first the identification of the plant dynamics and second is the determination of the appropriate control action needed to meet the overall specifications. If f is unknown and we have only the input-output data, then it is an adaptive control problem. Assuming that the plant is BIBO stable, it can be identified first and then the controller can be designed, although in the strict sense, the adaptive control problem is one in which the identification and control are carried out simultaneously. Designing the controller for the system described by (1) is difficult even when f is a known non-linear system. When f is unknown, the task is to approximate f so as to design the controller. There are two alternative approaches for this; one is to approximate f by a single non-linear model $\hat{f}$, which is called the global modeling, the most common being the polynomial fitting of the trajectory. Neural networks can be successfully applied to this problem due to their universal mapping capability (Haykin, 1994). An alternative approach is to divide the state space into local regions and model individually the local dynamics in each region, i.e., to decompose f into a family of models $\hat{f}_r$ with $r = 1 \ldots N$, each fitting only the neighbors of the present point in the reconstruction space. So the overall model is a concatenation of local maps (Principe *et al*, 1998).

$$\hat{f}(\mathbf{y}(n), \mathbf{u}(n)) = \bigcup_{r=1,..,N} \hat{f}_r(\mathbf{y}(n), \mathbf{u}(n)) \quad (2)$$

Each local model $\hat{f}_r$ can be potentially simpler, even linear, but the parameters will be changing across the state space. The identification step is carried out by the SOM, which will provide a codebook representation of the plant dynamics and organizes the different dynamic regimes in topological neighborhoods. The development of local models based on SOM is elaborated next.

3. DYNAMICAL MODELING USING SOM

Since the training of the static SOM is well known we do not elaborate the manner in which an SOM is trained (Kohonen, 1995, Haykin, 1994). The steps to develop the local linear models based on the SOM has three steps:

1) Reconstruction of the state space from the input signal (embedding)
2) Training of the SOM
3) Estimation of the local linear models

So the first step is to create an embedding of the input as well as the output time series, the embedding dimension is determined by the order of the system. According to Takens' Embedding theorem, the dimension $N > 2D+1$, where D is the dimension of the attractor. This delayed input and output time series is fed to the SOM and the SOM is trained. Next step is to utilize the SOM for dynamical modeling harnessing its power for data representation, space discretization and topological preservation (Principe *et al*, 1998). The local linear models can be derived from the trained SOM by creating an extra layer of linear PEs (Processing Elements) for each of the SOM PEs. Thus the r^{th} SOM PE has an associated linear model $\hat{f}_r$, which represents the linear approximation to the local dynamics.

$$\hat{f}_r(\mathbf{y}(n), \mathbf{u}(n)) = a_{r1}\, \mathbf{u}(n) + a_{r2}\, \mathbf{y}(n) = \mathbf{a}_r^T \overline{\mathbf{u}}(n) \quad (3)$$

Where $\overline{\mathbf{u}}(n)$ is the overall SOM input, $\overline{\mathbf{u}}(n) = [\mathbf{u}(n)\, \mathbf{y}(n)]^T$ which is nothing but the concatenated version of the delayed inputs and outputs. The parameter vector $\mathbf{a}_r$ can be estimated by least squares (LS) algorithm using the points in the local region. The procedure for building the local models is as follows:

1) From input-output data, form a codebook of pairs $\{y(n+1), [\mathbf{u}(n), \mathbf{y}(n)]\}$
2) Select pairs from the codebook within a neighborhood of size N_L ($N_L > N+2$, where N is the number of inputs being fed to the SOM) centered at the current winning PE.
3) Fit the local model $y(n+1) \approx \hat{f}_r(\mathbf{u}(n), \mathbf{y}(n))$ to the selected pairs in the codebook
4) Apply the local models to obtain $\hat{y}(n+1) = \hat{f}_r(\mathbf{u}(n), \mathbf{y}(n))$

In short the local dynamic model works as follows: the current input $\overline{\mathbf{u}}(n) = [\mathbf{u}(n)\, \mathbf{y}(n)]^T$ is placed at the input of the SOM and the winner-take-all operation will select the PE that best represents the current input. This winner activates a single linear PE that contains the weights of the local linear model and estimates the response of the system for this input as $\hat{y}_r(n+1) = a_r^T \overline{\mathbf{u}}(n)$. The simulation results given in the next section describe how accurately this group of linear models emulates the actual system dynamics.

4. CASE STUDIES-DYNAMICAL MODELING

The idea presented above on local modelling is best illustrated by considering specific examples. Here we consider three examples in which a non-linear dynamical system is represented by SOM based local models. The output obtained from the models is

compared with the actual plant output and is plotted with the identification error.

4.1 Example 1

In this example a piecewise linear but globally non-linear system is taken. Local linear models are derived for this system described by,

$$y(n) = \begin{cases} 2u(n) \;\; if\, 0 \le u(n) \le 0.5 \\ -2(u(n)-1) \;\; if\, 0.5 \le u(n) \le 1 \end{cases} \quad (4)$$

Note that this system is non-invertible. We used a SOM of 7x7 to solve this problem. Figure 1. shows the actual response of the system along with the response of the SOM based multiple models.

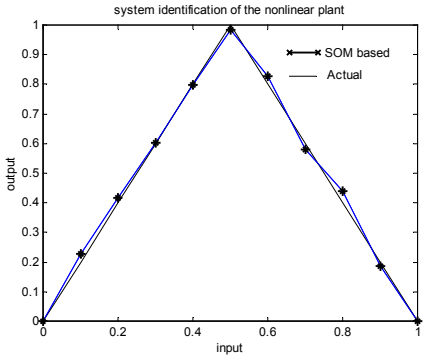

Figure 1. Actual and identified response of example 1

Here an input of magnitude less than one is applied to the system and the SOM will select the winner that best represents the input, which in turn selects the local model. Observe that although the system in the range $u \ge 0.5$ is not linear, the SOM switches between the corresponding models based on the input.

4.2 Example 2

We will now consider a linear system with varying coefficients. The system is described by $H(z) = \dfrac{a_0}{1 + b_1 z^{-1}}$, which is a first order linear system with coefficients a_0 and b_1 that vary according to the input magnitude, i.e.,

$$H(z) = \begin{cases} \dfrac{1}{1+0.5z^{-1}}, 0 < u < 0.5, \\ \dfrac{0.6}{1-0.2z^{-1}}, u > 0.5; \\ \dfrac{0.3}{1+0.8z^{-1}}, -0.5 < u < 0 \\ \dfrac{1}{1+0.4z^{-1}}, u < -0.5 \end{cases} \quad (5)$$

We assume that the input to the system ranges between –1 and 1 and we know in advance the order of the system though the parameters are unknown. A SOM of size 22x22 is trained with values uniformly distributed in $[-1, 1]$ for both dimensions. So each weight vector has two elements, one corresponding to the present value of the input and other for the previous value of the plant output. As discussed in the previous section, the modeling is carried out using least square fit and we obtain 484 models for the plant over the whole dynamic range. So the models are described by the equations

$$\hat{y}_r(k+1) = a_{0r}u(k) + a_{1r}y(k) \quad (6)$$

where, a_{0r}, a_{1r} denote the coefficients of the weight vector for the r^{th} model and $y(k), u(k)$ represent the actual system output and input respectively at the k^{th} time instant. In order to test these models, we will give a step input as shown in figure 2(a) with varying amplitude levels thus making the plant switch regimes. The output from the models is shown in figure 2(b), which clearly follows the actual trajectory of the plant thereby proving the veracity of the use of multiple models for system identification.

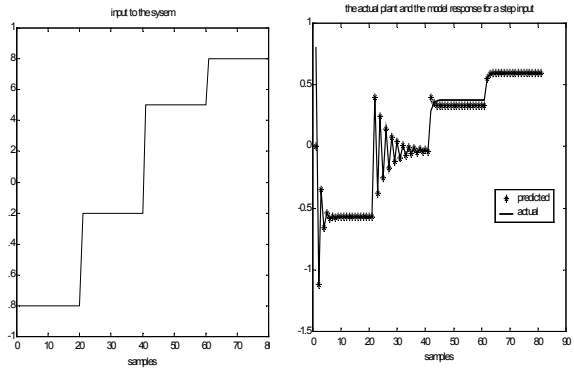

Figure 2. (a) Step input for the plant (b) Response of the plant and the model output

4.3 Example 3

We will now take a non-linear system described by the equation,

$$y(n+1) = \frac{1}{L}\left[\sin[y(n)] + u(n)(5 + \cos[y(n)u(n)])\right] \quad (7)$$

where L is the normalization constant set to 6.03. We will again use a 22x22 SOM. The system is fed with a random input. The system response is plotted along with the model response in figure 3. As can be seen the models predict the system output pretty well.

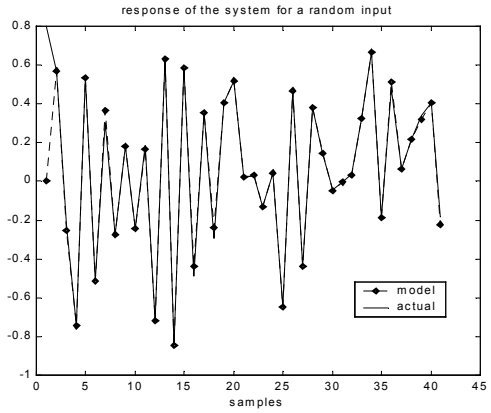

Figure 3. Random input response of the actual plant and the model

5. DESIGN OF SWITCHING CONTROLLERS

The general architecture for the control of a plant using multiple models is shown in figure 4. The unknown system to be controlled is SISO and has input u and output y. As stated in section 2 our principal objective is to determine a control input $u(k)$, which will result in the output $y(k)$ of the plant in (1) tracking a specified sequence $y^*(k)$ with sufficient accuracy (Widrow, 1996). The system has N identification models denoted by $\{M_r\}_{r=0}^{N}$, in parallel. Corresponding to each model M_r, a controller C_r is designed such that C_r achieves the control objective for M_r. So at every instant one of the models is selected by the SOM as discussed in the previous section, and the corresponding controller is used to control the actual system. This is the switching part of the system. We will be mainly concerned with the issues of learning a controller from a multiple model paradigm, using gradient descent learning. This means that the output plant error must be propagated through the multiple models to adapt the parameters of the controller. Since the multiple models are structured by the SOM, we have to derive the dual of the SOM for sensitivity propagation. Since the SOM input/output map is a discontinuous process, it may appear that it is impossible to obtain its dual system. Albeit strictly speaking this seems true, practically it is possible to transfer sensitivities through a SOM, provided sample-by-sample estimates are used like in the LMS algorithm. In the LMS we will be propagating instantaneous errors and so the algorithm will not see the discontinuities. With this background, we can approach the design of the controller in two ways.

1. Assign one controller for each of the models and design them independent of the others. This is possible because there is a model corresponding to each piecewise linear region of the state-space.
2. Assign a controller for a group of models, i.e., one controller being able to control a cluster of models, which is similar to robust control design.

Here, we design the controller using the first method, i.e., one controller per model. To design C_r, the linear model M_r is taken and the controller is designed adaptively using LMS algorithm (Haykin, 1994, Widrow 1985). Since the models are linear, the controller is also linear. The block diagram associated with the design of the controller is shown in the figure 5. Let the desired output be $d(n)$ and the controller weight vector be $\mathbf{w_c}(n)$. Then the controller output $u(n)$ is given by,

$$u(n) = \mathbf{w}_c^T(n)\,\bar{\mathbf{d}}(n), \tag{8}$$

where $\bar{\mathbf{d}}(n) = [d(n)\ y(n-1)]^T$, the overall controller input.

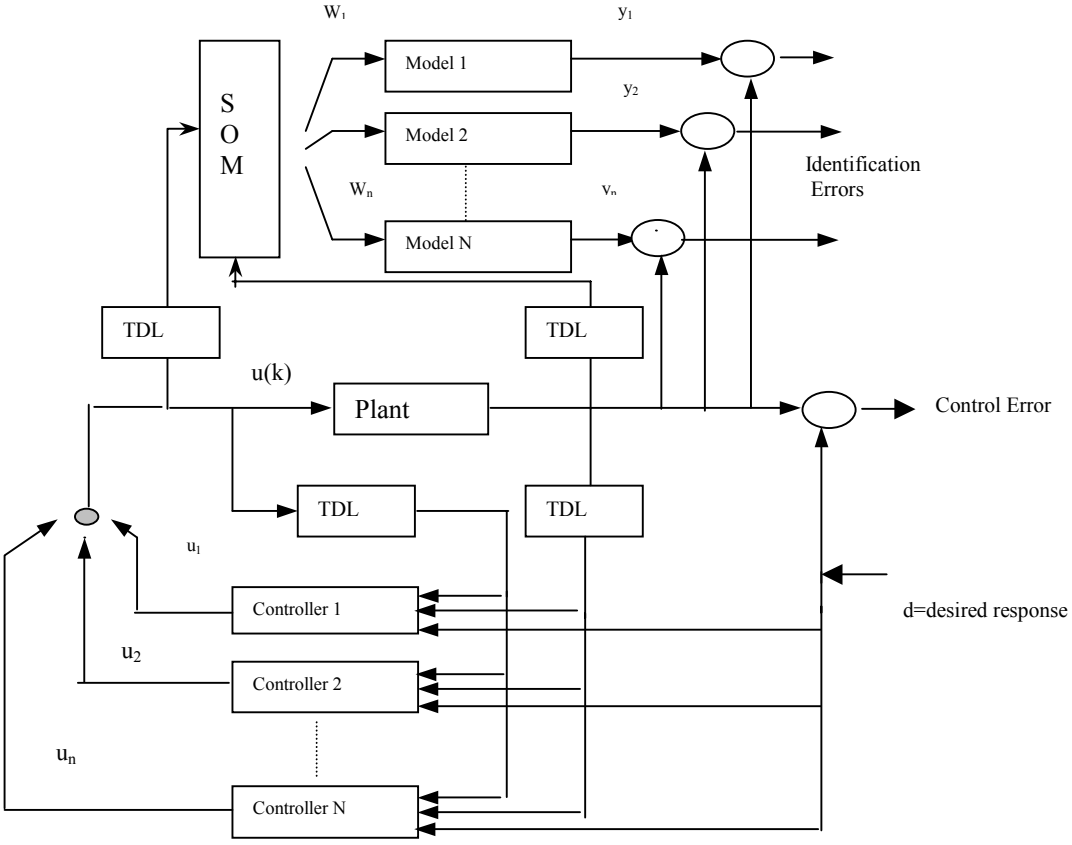

Figure 4. General architecture for multiple controllers

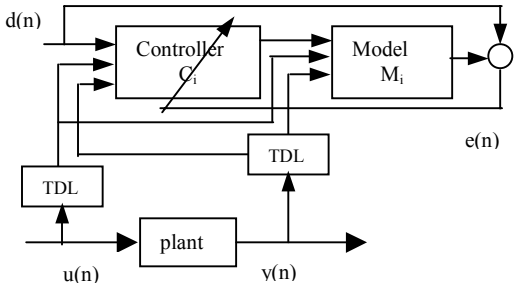

Figure 5. Single controller design

The corresponding SOM output is the winner PE weight, $w^*(n)$, which is applied to the model M_i (with parameter vector $\mathbf{a}_i$) to get a response $\hat{y}(n+1)$. The controller weight vector is adapted using simple LMS rule, which utilizes the instantaneous gradient. The instantaneous gradient is defined as

$$J(n) = \frac{1}{2}e^2(n) \tag{9}$$

So the weight vector update equation is

$$\mathbf{w_c}(n+1) = \mathbf{w_c}(n) - \eta \frac{\partial J(n)}{\partial \mathbf{w_c}(n)}$$

We can compute this gradient by using chain rule.

$$\begin{aligned} \frac{\partial J(n)}{\partial \mathbf{w_c}(n)} &= e(n)\frac{\partial e(n)}{\partial \mathbf{w_c}(n)} \\ &= -e(n)\frac{\partial \hat{y}(n)}{\partial \mathbf{w_c}(n)} \\ &= -e(n)\mathbf{a}_i \frac{\partial \mathbf{u}(n)}{\partial \mathbf{w_c}(n)} \\ &= -e(n)\mathbf{a}_i \overline{\mathbf{d}}(n) \end{aligned} \tag{10}$$

The adaptation process is repeated for all the models resulting in a model controller pair $\{M_i, C_i\}$ associated with every SOM PE. As mentioned before the natural way to decide when and to which controller one should switch is to determine the model that best describes the plant. This will be decided by SOM in accordance with the present state of the system. In a nutshell, the procedure for controlling the non-linear plant is as follows: The present state of the system $[\mathbf{u}(n)\,\mathbf{y}(n)]^T$ is applied to the SOM to get the winning PE. Now this PE will give the associated local model representing the present dynamics of the system and will fire the corresponding controller. The controller output is then fed to the actual system, which is forced to track the reference trajectory. As can be expected an error in the identification will lead to an error in the control.

6. CASE STUDIES-CONTROLLER

The proposed strategy for controller design through local modeling is tested for the examples we used in section 4.

6.1 Example 1

Consider the non-linear system described by (4). Controllers are designed for every model as described in the previous section. The whole system with the multiple controllers is evaluated in a trajectory-tracking problem. Figure 6 shows the desired trajectory and the system output. Note that the system output tracks the trajectory with almost zero error and this clearly indicates that the controllers are switching accurately.

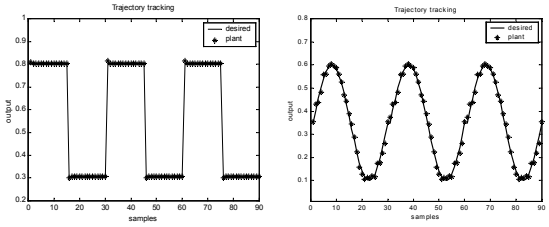

Figure 6. Trajectory tracking for example 6.1

It can be concluded from this example that though the design of controller design for this non-invertible system may seen to be an impossible task, it is possible to design the inverse if one knows at which particular part of the space the system is operating. This is possible due to the local nature of the models describing the plant.

6.2 Example 2

Consider the first order linear plant described by (5). Here our objective is to study how the different models perform in a control context. The system is fed with many different test signals and the system is found to track them pretty well, proving the multiple controller's ability to track the dynamics of the plant. Figure 7 shows the plots for trajectory tracking.

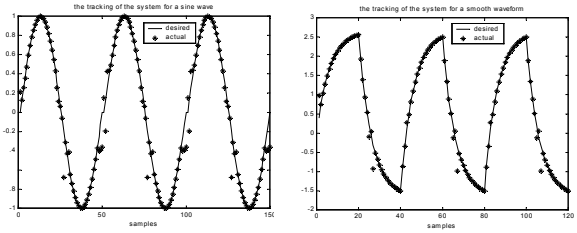

Figure 7. Trajectory tracking for example 6.2

6.3 Example 3

We will finally consider the non-linear system given by (7). The previous examples showed piece-wise linear plant dynamics. However, the plant in (7) is inherently non-linear. Again, the controllers are designed for every local linear model and the plant is tested with various trajectories. Figure 8 shows the

plots of the trajectory tracking. It is obvious that the multiple model-based controllers do well even when the system is completely non-linear.

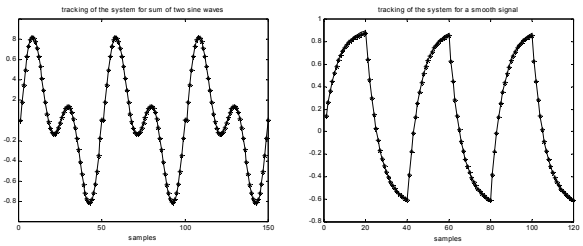

Figure 8. Trajectory tracking for example 6.3

6. CONCLUSIONS

Modeling of non-linear systems has a variety of applications. In this paper we considered the non-linear system modeling in the control problem scenario. A global or a set of local models can be used to identify the non-linear system. In this paper the modeling is based on SOM which works not only as an enhanced clustering algorithm preserving the neighborhoods of the reconstruction space dynamics, but also as an identifier of the local dynamics. The SOM is trained globally to cluster the dynamics. The discontinuities that one may encounter across the boundaries are minimized due to the inherent nature of least squares we used in the derivation of local models. This approach of utilizing SOM for non-linear system identification is relevant for designing controllers. The models turn out to be really simple in structure and so are the controllers. The method is tested for a variety of control problem yielding satisfactory performance. As an added advantage this method can be used to identify the unstable modes of a system. Work in this regard is still under progress.

In this paper we used a switching strategy depending on the values of the inputs to the SOM or the present state of the system. Instead what we can do is to evaluate a performance index and switch the models accordingly to this index. The quantization error that is inherent in SOM will introduce some error in the identification step. This will give rise to an additional error in the controller design. We can estimate an upper bound on this quantization error, and can determine the maximum control error.

Acknowledgements: This work was partially supported by Accurate Automation Corporation under grant #463.

REFERENCES

D. G. Lainiotis, (1976) “Partitioning: A Unifying Framework for Adaptive Systems”, Proceedings of the IEEE, 64:1126-1143,1182-1197, August 1976.

K. S. Narendra, S. Mukhopadhyay, (1997) “Adaptive Control using Neural Networks and Approximate Models”, IEEE Transactions on Neural Networks, 475-485, Vol. 8, No. 3, May 1997.

K. S. Narendra, J. Balakrishnan, (1997) “Adaptive Control using Multiple Models”, IEEE Transactions on Automatic Control, 171-187, Vol. 42, No. 2, February 1997.

K. S. Narendra, J. Balakrishnan, M. K. Ciliz, (1995) “Adaptive and Learning using Multiple Models, Switching and Tuning”, IEEE Control Systems, 37-50, June 1995.

J. C. Principe, L. Wang, M. A. Motter, (1998)“Local Dynamic Modeling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control”, Proceedings of IEEE, 2240-2258 Vol. 86, No. 11, November 1998.

B. Widrow, S. D. Stearns (1985), “ *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

M. Motter (1997), “Control of the NASA transonic wind tunnel the self-organizing feature map,” Ph.D. dissertation, Univ. of Florida, Gainesville, Dec. 1997.

B. Widrow, E. Walach, (1996), “Adaptive Inverse Control,” Prentice Hall Information & System Science Series, NJ

S. Haykin (1994), “Neural Networks: A Comprehensive Foundation”, Macmillan, New York.