

Local Dynamic Modeling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control

JOSE C. PRINCIPE, SENIOR MEMBER, IEEE, LUDONG WANG, MEMBER, IEEE, AND
MARK A. MOTTER, SENIOR MEMBER, IEEE

Invited Paper

The technique of local linear models is appealing for modeling complex time series due to the weak assumptions required and its intrinsic simplicity. Here, instead of deriving the local models from the data, we propose to estimate them directly from the weights of a self-organizing map (SOM), which functions as a dynamic-preserving model of the dynamics. We introduce one modification to the Kohonen learning to ensure good representation of the dynamics and use weighted least squares to ensure continuity among the local models. The proposed scheme is tested using synthetic chaotic time series and real-world data.

The practicality of the method is illustrated in the identification and control of the NASA Langley wind tunnel during aerodynamic tests of model aircraft. Modeling the dynamics with an SOM leads to a predictive multiple model control strategy (PMMC). Comparison of the new controller against the existing controller in test runs shows the superiority of our method.

Keywords— *Dynamic modeling, local linear models, multiple models, NASA wind tunnel, neural control, self-organizing maps.*

I. INTRODUCTION

System identification and time series prediction are engineering embodiments of the old problem of function approximation. Each seeks to estimate, in its own way, the parameters of the system that created the time series. The traditional approach is statistical and based on the linear model [3]: we assume that the time series is produced by a linear system excited by white Gaussian noise. The variability observed in the time series is assigned to the stochastic nature of the excitation, which can not be modeled. Therefore, the goal of linear modeling is limited to the estimation of the parameters of the model which will match the power spectrum of the time series.

Manuscript received November 1, 1997; revised April 17, 1998. This work was partially supported by the National Science Foundation under grant ECS-9510715 and by ONR under grant N00014-94-1-0858.

The authors are with the Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL 32611 USA.

Publisher Item Identifier S 0018-9219(98)07861-X.

More recently, another perspective is surfacing which is called dynamic modeling [16], [21]. The time series is considered the output of a deterministic, autonomous (i.e. without input), dynamical system. The complexity of the time series is linked to the high order and nonlinear nature of the dynamical system and not to the exogenous random excitation, as in the linear case. In this approach the model system must either be nonlinear or linear with time-varying parameters, because linear time invariant systems have trivial autonomous dynamics (either fixed points of limit cycles).

Perhaps the best example of such a perspective is chaotic time series modeling [7], [11], [21]. On the complexity scale, chaotic time series span the gap between periodic signals and random noise. Chaotic time series offer the ultimate difficulty for developing a model from a time series because the signal time structure is time varying and highly complex with a $1/f$ spectrum. It is therefore the perfect environment to test new modeling approaches since a minor error in model parameters is amplified by the natural divergence of the trajectories in phase space. Its potential usefulness is also enormous because many real-world phenomena are considered chaotic (the weather, sea clutter, lasers, heart beats, some types of brain waves, etc.). Time series produced by chaotic systems that were previously considered “noise” may in fact have deterministic structure which can be modeled. Hence, for improved performance, one can develop predictive models whose output can be subtracted from the time series instead of applying filtering techniques to attenuate the noise. Recent results by Haykin [30] show precisely the power of nonlinear dynamic modeling to improve the detectability of targets from sea clutter.

The tools to understand, develop, and apply nonlinear dynamic models are very different from linear time series modeling and will be briefly reviewed. We summarize the

basic approaches utilized in dynamic modeling, but the main goal of the paper is to develop an innovative local linear dynamic modeling reminiscent of vector quantization. Instead of representing the unknown manifolds with state-dependent predictive models, which are independently derived from local information [10], here the global dynamics are approximated by a preset number of local linear models, which are concurrently derived through competition using Kohonen's self-organizing map (SOM) [18]. The local linear models are constructed from the weights of the SOM, which facilitates training when compared to previous techniques. More significantly, our scheme provides a partial overlap among neighboring subregions with different local models, which effectively alleviates the discontinuity problem. In our method, the SOM is not only employed as a static representation of the signal, but as a robust dynamic-preserving space. We discuss first the implementation of our model using the traditional Kohonen learning (KL), but we then propose a modified learning strategy for the SOM utilizing the prediction error. Weighted least-square estimation of local models is proposed to improve the modeling performance. We show that our method achieves dynamic modeling for both synthetic (Lorenz) and experimental (laser data) chaotic time series.

The applications (and implications) of dynamic modeling to intelligent control are recent and rich. Here the SOM local modeling framework is applied to develop a set point controller for the 16-ft Transonic Tunnel at NASA Langley Research Center, Hampton, VA. The wind tunnel is used for aerodynamic tests of airplane models and requires precise wind speeds in spite of the large changes in the angle-of-attack of the model which produce different load conditions in the wind tunnel. The control problem is one of regulating the air speed around a Mach number set point, while the plant has a time-varying load which produces very different dynamics. The control action is very different from set point to set point and is also a function of the disturbance. The approach of modeling the tunnel dynamics with a set of local models seems particularly appropriate for this task, since the winner-take-all operation of the SOM can switch very quickly between very different local dynamic models, tracking the change in the tunnel dynamics.

The fundamental issue in this application is how to go from dynamic modeling (which assumes autonomous dynamics) to a control scheme (which requires a nonautonomous system with forcing inputs). We cluster the control inputs and use them to select one SOM from a set of local dynamic models which are trained with Mach number responses from the full operating range. The winner-take-all operation of the SOM local model creates a switching controller similar to the approach described by Narendra [25]. A set of predefined control inputs is applied to the local model selected by the winning probability element (PE) with the goal of predicting the tunnel dynamics 50 steps in the future. The control input that best meets the set point specification is applied to the driving turbine.

The experimental system was successfully tested in an operational environment. The Mach number was controlled

to within the research requirement at various transonic Mach numbers over a period of approximately 9 h, while the aircraft model under test was positioned in various attitudes (three runs of about 3 h duration). We will present results of this test run and compare them with the operator and the existing controller.

II. DYNAMIC MODELING

Time series prediction seeks to quantify the system that created the time series. In the linear model framework, the unknown system transfer function is the inverse of the model transfer function, which is arrived at through prediction [32]. When the linear methodology is extended to nonlinear systems, the relation between the model system and the unknown system has to be stated in different terms since the nonlinear model system can no longer be described by a transfer function. The works of Takens [45], Casdagli [7], and many others have shown that an equivalent methodology exists for nonlinear dynamic modeling. A K th order autonomous dynamical system can be represented by a set of K th ordinary differential equations

$$\frac{d}{dt}\mathbf{s}(t) = \Phi(\mathbf{s}(t)) \quad (1)$$

where $\mathbf{s}(t) = [s_1(t), s_2(t), \dots, s_K(t)]$ is a vector of system states and Φ is called the vector field. Bold letters will represent vectors. The system state at any time can be specified in a K dimensional space. The vector field maps a manifold M to a tangent space T . If Φ is a nonlinear function of the system state the system is called nonlinear. Assume there is a closed form solution to (1) $\phi_t: M \rightarrow M$. For a given initial condition s_0 , the function $\phi_t(s_0)$ represents a state-space trajectory of the system (the mapping ϕ is called the flow). If the vector field is continuously differentiable then the system flow and its inverse exist for any finite time. This implies that the trajectories of an autonomous system never intersect each other.

When working with experimental data produced by dynamical systems, we generally do not know the state equations (1) [1]. We are restricted to observe the outputs of the dynamical system. So a fundamental issue is what can be inferred about the dynamics from the observation of an output time series. Packard *et al.* [31] showed experimentally and Takens [45] proved that a sampled observable $x(n)$ of the dynamical system and its delayed versions

$$\mathbf{x}(n) = [x(n), x(n - \tau), \dots, x(n - (N - 1)\tau)] \quad (2)$$

can be used to create a trajectory in a Euclidean space of size N , which preserves the dynamical invariants (correlation dimension and Lyapunov exponents) of the original dynamical system provided N is sufficiently large. In (2) τ is the normalized delay; it is irrelevant theoretically, but it plays an important role when the time series is noisy and or of finite size. The dimension N of the space should be larger than $2D_e$, where D_e is the dimension of the attractor, i.e., the geometric object created by the trajectories after transients die out. This is a remarkable result, since it shows

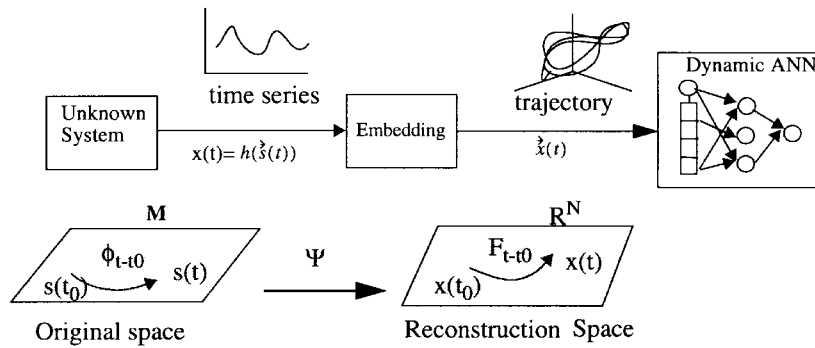


Fig. 1. Nonlinear modeling steps and their mathematical translation.

that the system's state information can be recovered from a sufficiently long observation of the output time series. Its simplicity should be contrasted with the conventional approach of Kalman observables in control theory [46].

In more mathematical terms, this statement means that there is a one-to-one smooth map Ψ with a smooth inverse from the K th-dimensional manifold M of the original system to the Euclidean reconstruction space R^N . Such mapping is called an embedding, and the theorem is known as Takens Embedding Theorem [45].

According to Takens' Embedding Theorem, when $N > 2D_e$ a map $F: R^N \rightarrow R^N$ exists that transforms the current reconstructed state $\mathbf{x}(n)$ to the next state $\mathbf{x}(n+\tau)$, where τ is the normalized delay. For simplicity we will set $\tau = 1$, which means

$$\mathbf{x}(n+1) = F(\mathbf{x}(n)) \quad (3)$$

or

$$\begin{bmatrix} x(n+1) \\ \dots \\ x(n-N+2) \end{bmatrix} = F \left(\begin{bmatrix} x(n) \\ \dots \\ x(n-N+1) \end{bmatrix} \right).$$

Note that (3) specifies a multiple input, multiple output system F built from several (nonlinear) filters and a nonlinear predictor [36]. The predictive mapping is the center piece of modeling since once determined, F can be obtained from the predictive mapping by simple matrix operations. The predictive mapping $f: R^N \rightarrow R$ can be expressed as

$$x(n+1) = f(\mathbf{x}(n)). \quad (4)$$

Equation 4 defines a deterministic nonlinear autoregressive (NAR) model of the signal. The existence of this predictive model lays a theoretical basis for dynamic modeling in the sense that it opens the possibility to build from a vector time series a model to approximate the mapping f . The result and steps in dynamical modeling are depicted in Fig. 1.

Dynamic modeling implies a two-step process [16]. The first step is to transform the observed time series into a trajectory in the reconstruction space by using one of the embedding techniques [52]. The most common is a time delay embedding which can practically be implemented with a delay line (also called a memory structure in

neurocomputing) with a size specified by Takens' Embedding Theorem. The dimension D_e of the attractor can be estimated by the correlation dimension algorithm [13], but other methods exist [1]. The second step in dynamic modeling is to build the predictive model of (4) from the trajectory in reconstruction space [16]. These theoretical considerations imply that after the embedding step any adaptive nonlinear system trained as a predictor can identify the mapping f and thus are of no help in deciding which is the best strategy to design the predictive model. So a different approach is required to design predictive models.

III. GLOBAL AND LOCAL MODELS

The task of nonlinear dynamic modeling is to approximate the reconstructed flow $F(x)$ in (3) which evolves the trajectory in reconstruction space $\mathbf{x}(n) \rightarrow \mathbf{x}(n+1)$. Unlike the linear case, there is no algorithmic way to determine the functional form for $F(\cdot)$, hence the framework of function approximation has been utilized [37]. The flow $F(x)$ is approximated by a functional form $\tilde{F}(\cdot)$ given by

$$\tilde{F}(\mathbf{x}(n), \mathbf{w}) = \sum_{i=1}^N w_i \varphi_i(\mathbf{x}(n)) \quad (5)$$

and the error

$$E = \sum_{x \in D} \text{dis}(\mathbf{x}(n+1) - \tilde{F}(\mathbf{x}(n), \mathbf{w})) \quad (6)$$

is minimized, where D is the domain of the approximation, and $\text{dis}(\cdot)$ is an appropriate metric (normally taken as the Euclidean distance). Estimated quantities will be denoted by the superscript \sim . Due to the particular form of dynamic modeling (4), the function approximation of (6) is equivalent to prediction in state space, i.e., we can define an instantaneous error

$$\varepsilon(n) = x(n+1) - \tilde{f}(\mathbf{x}(n), \mathbf{w}) \quad (7)$$

which is minimized with the appropriate error metric over the domain. There are two basic alternatives to proceed with the optimization: either a single map $\tilde{F}(\cdot)$ is used to approximate all the points in the reconstruction space (global modeling), i.e., $D = R^N$; or the map $\tilde{F}(\cdot)$ is decomposed in a family of maps $\tilde{F}_r(\cdot)$ with $r = 1, \dots, R$, each fitting only the neighbors of the present point in

reconstruction space (local modeling), i.e., D is a local region centered at the present point in reconstruction space. The overall predictive map is then a concatenation of local maps [10]

$$\tilde{F}(\mathbf{x}(n)) = \bigcup_{r=1, \dots, R} \tilde{F}_r(\mathbf{x}(n)). \quad (8)$$

A. Global Dynamic Models

Global models based on polynomial fitting of the trajectory have been reported in the literature [12], [43]. Neural networks have also been successfully applied to this problem due to their universal mapping characteristics. After the pioneering work of Lapedes and Farber [21], multilayer perceptrons (MLP's) have been extensively utilized [17], [34]. Following the work of Broomhead and Lowe [4], radial basis functions (RBF's) have also become very popular as global models [16]. Note that these two topologies have rather different approximation characteristics since their basis functions are respectively global (sensitive to the entire space) and local (primarily sensitive to a local area). Nevertheless both systems are trained with data from the entire reconstruction space, so they belong to the global model category.

The global models have been the most explored for nonlinear dynamic modeling, but they will not be reviewed here. Please consult [52] for a broad review and [36] for a performance comparison with local dynamic models.

B. Local Dynamic Models

An alternate modeling methodology divides the state space in local regions and models individually the local dynamics in each region. This method is closely related to differential topology [14] and it is more general than the global approach in the sense that fewer statistical and geometric assumptions are required about the data [10]. Each local model $\tilde{F}_r(\cdot)$ ($r = 1, \dots, R$), can be potentially simpler, usually linear, but the parameters will have to change across state space. The model in (5) becomes

$$\tilde{F}_r(\mathbf{x}(n), \mathbf{w}) = \mathbf{J}_r \mathbf{x}(n) + \mathbf{b}_r \quad (9)$$

where \mathbf{J} is the Jacobian of F at $\mathbf{x}(n)$ and \mathbf{b} is a bias vector. The assumption behind this model is that the state dynamics are locally smooth. Assuming the L2 norm for the cost function, the parameters (\mathbf{J}, \mathbf{b}) can be estimated by least squares (or with the LMS algorithm [53]) from the data points within the local region, by simply substituting (9) into (6), i.e.,

$$\min_{\mathbf{b}, \mathbf{J}} E$$

$$E = \sum_{k=1}^{N_L} (\mathbf{x}_k(n+1) - \mathbf{J}_k \mathbf{x}_k(n) - \mathbf{b}_k)^2 \quad (10)$$

where N_L is the number of points in the neighborhood of point $\mathbf{x}(n)$, which for stable results must be larger than N , the size of the reconstruction space. Notice that this is a

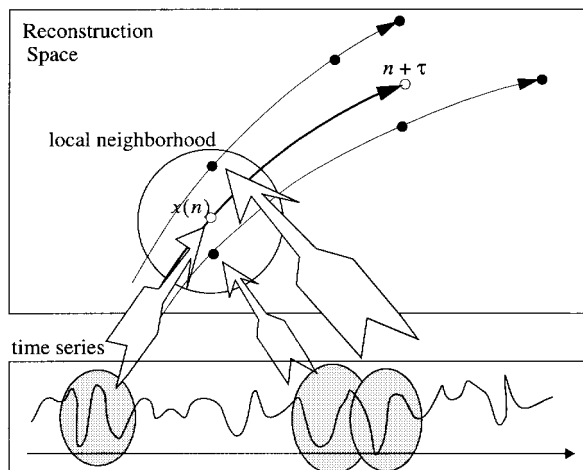


Fig. 2. The relationship between time series samples and neighborhoods in reconstruction space.

conceptually simple but time consuming operation. Points which are neighbors in reconstruction space may be very far away in the time series (Fig. 2).

So, the time series has to be either reconstructed separately for each local model to find the N_L nearest neighbors, or the full trajectory has to be kept in computer memory for a search of the neighbors. Then (10) has to be computed for each point $\mathbf{x}(n)$ in the local neighborhood. The global dynamic model is obtained by piecing together all the local models according to (8).

Another difficulty is how to cover appropriately the trajectories in reconstruction space (placement of centers and neighborhood radius). Normally a uniform coverage of the space is utilized, but the size of the neighborhood is difficult to decide *a priori*. Applications of this technique have been reported for prediction of chaotic time series [7], [11], noise removal of experimental time series [15], [20], estimating the largest Lyapunov exponent [5], [40], and control of chaotic dynamics [29], [42]. For dynamic modeling which seeks to model the long term behavior of the dynamical system, the problem is how to guarantee smoothness at the boundaries among the local models. Crutchfield [10] has shown experimentally that dynamic modeling fails if this condition is not imposed. The extended Kalman filter also develops a local linear model of the trajectory [46], but it utilizes a formulation based on a statistical data model.

C. State Dependent Prediction of Nonlinear AR Processes

The approach of locally linear AR fitting has been discussed in the time series literature by Priestley [33]. A similar approach was proposed by Farmer and Sidorowich [11] for predicting chaotic time series. They concluded that there was little benefit of using higher order polynomials to fit the local dynamics. Tong also introduced the idea of threshold linear models [47]. Singer *et al.* [44] presented the important idea of local linear modeling as a state dependent AR scheme. When the state vectors are constructed from the time series of a single variable by a delay embedding, the locally linear functions are effectively reduced to state de-

pendent AR models, and thus a codebook prediction scheme reminiscent of vector quantization is formed. This insight is very valuable because it couples local linear modeling with a combination of vector quantization followed by adaptive linear models.

An N th order nonlinear AR model can be written as [44]

$$y(n+1) = f(y(n), y(n-1), \dots, y(n-N+1)) + u(n)$$

which accepts the state space representation

$$\begin{aligned} \mathbf{x}(n+1) &= \begin{bmatrix} 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 0 \end{bmatrix} \mathbf{x}(n) + \begin{bmatrix} 0 \\ \dots \\ 0 \\ f(\mathbf{x}(n)) \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ \dots \\ 0 \\ 1 \end{bmatrix} \mathbf{u}(n) \\ y(n) &= [0 \quad \dots \quad 0 \quad 1] \mathbf{x}(n) \end{aligned} \quad (11)$$

where $\mathbf{x}(n) \in R^N$, $y(n) \in R$, $\mathbf{u}(n) \in R^q$ are respectively the vectors for the state, output and input, and $f: R^N \rightarrow R$. Due to the Markovian structure of NAR models of order N , $y(n+1)$ can be estimated from the most recent N values, i.e.,

$$P(y(n+1)|y(i), 0 < i < n) = P(y(n+1)|\mathbf{x}(n)) \quad (12)$$

as shown by (11). The minimum mean square error estimate of $y(n+1)$ using the entire signal history is

$$\begin{aligned} \hat{y}(n+1) &= E[y(n+1)|\mathbf{x}(n)] \\ &= E[f(\mathbf{x}(n)) + u(n)|\mathbf{x}(n)] = f(\mathbf{x}(n)) \end{aligned} \quad (13)$$

or in other words, even though $f(\mathbf{x}(n))$ is not available, the system state dynamics can be observed in prediction given the recovered state vector $\mathbf{x}(n)$.

The time series history $y(n+1) = f(\mathbf{x}(n)) + u(n)$ represents a set of noisy state observations nonuniformly distributed in the reconstruction space (see Fig. 2). So one option is to create a codebook of state vectors and signal values, and use the present state to lookup the next value of the time series. The pairs $(y(n+1), \mathbf{x}(n))$ contain the information to help us observe local instances of the state, although they are tainted by noise. If enough of these observations are available to cover all of the state space and the noise is reduced by local averaging, the nonlinear dynamical system can be identified. With this perspective we can immediately enunciate a procedure to build a dynamic model from a codebook of local linear models [44].

- 1) Form a codebook of pairs $(y(k+1), \mathbf{x}(k))$ from the signal history.
- 2) Select pairs $(y(k+1), \mathbf{x}(k))$ from the codebook near the current state $\mathbf{x}(n)$.
- 3) Fit a local model $y(k+1) \approx \tilde{f}_r(\mathbf{x}(k))$ to the selected pairs in the codebook.
- 4) Apply the local model to obtain $\tilde{x}(n+1) = \tilde{f}_r(\mathbf{x}(n))$.

We just have to address the practical aspects of implementing the codebook and estimating the local linear model. Assuming that the underlying state evolution is sufficiently smooth, then the mapping function $f(\mathbf{x})$ in the vicinity of $\mathbf{x}(n)$ can be approximated by the first few terms of its multidimensional Taylor series expansion

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}(n)) + \nabla f^T(\mathbf{x}(n))(\mathbf{x} - \mathbf{x}(n)) + \dots \approx \mathbf{a}^T \mathbf{x} + b \quad (14)$$

which is a local linear predictor. Accordingly, the vector and scalar quantities of \mathbf{a} and b are estimated from the selected pairs $(y(n+1), \mathbf{x}(n))$ in the neighborhood of the present state. The codebook will be developed with an SOM.

IV. KOHONEN'S SELF-ORGANIZING MAP

A. SOM Networks and Kohonen Learning

Kohonen [18] developed the SOM to transform an input signal of arbitrary dimension into a lower (one- or two-) dimensional discrete representation preserving topological neighborhoods. Let $\Phi: \mathbf{X} \rightarrow \mathbf{A}$ denote the SOM mapping from an input space \mathbf{X} and the discrete output space \mathbf{A} . The SOM Φ defines in Kohonen words "an elastic net of points \mathbf{A} that are fitted to the input signal space \mathbf{X} to approximate its density function in an ordered way," [18]. In order to achieve this goal, the discrete grid \mathbf{A} of PE's indexed by $i \in \mathbf{A}$ is described by reference vectors \mathbf{w}_i which take their values in the input space \mathbf{X} . The response of an SOM to an input $x \in \mathbf{X}$ is determined by the reference vector w_{i° of the PE which produces the best match to the input

$$i^\circ = \arg \min_i \text{dist}(\mathbf{w}_i - \mathbf{x}) \quad i = 1, \dots, A. \quad (15)$$

where the superscript $^\circ$ denotes the winning PE, and $\text{dist}(\cdot)$ is an appropriate distance metric such as the Euclidean metric. This means that each PE represents a local neighborhood of the input space, also called a Voronoi cell [18]. In this respect an SOM is a vector quantizer, where the weights play the role of the codebook vectors [2]. The nonlinear mapping Φ is obtained by modifying the weight vectors \mathbf{w}_i with a learning algorithm. Since the ultimate goal of the SOM is to approximate the input data density, the weights should be "attracted" to the regions of the input space with high sample density. The training can be accomplished generally with a competitive learning rule as

$$\mathbf{w}_i(n+1) = \begin{cases} \mathbf{w}_i(n) + \eta(\mathbf{x}(n) - \mathbf{w}_i(n)), & i = i^\circ \\ \mathbf{w}_i(n), & \text{otherwise.} \end{cases} \quad (16)$$

The remarkable difference between the SOM and other direct competitive learning schemes is the details of the weight updating. Instead of adjusting the winner exclusively, a scaled adjustment is applied to all the output PE's

$$\Delta \mathbf{w}_i(n) = \eta \Lambda_{i^\circ, i}(\mathbf{x}(n) - \mathbf{w}_i(n)) \quad (17)$$

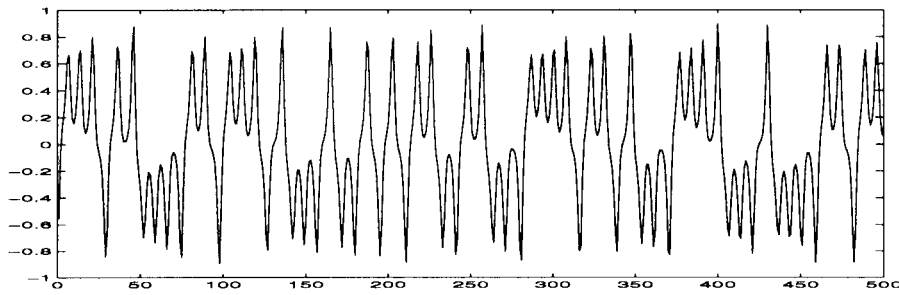


Fig. 3. The time series of the Lorenz system (x -coordinate).

where $\Lambda_{i^\circ, i}$ is a spatial neighborhood function and n is the learning rate. A typical neighborhood function is

$$\Lambda_{i^\circ, i}(n) = \exp\left(-\frac{\|r_i - r_{i^\circ}\|^2}{2\sigma^2(n)^2}\right) \quad (18)$$

where $\|r_i - r_{i^\circ}\|$ represents the Euclidean distance in the output space between the i th PE and the winner. The above choice of the neighborhood function adjusts appreciably the output units close to the winner while those further away experience little change. Equation (16) is a special case of (17) for

$$\Lambda_{i^\circ, i} = \begin{cases} 1, & i = i^\circ \\ 0, & \text{otherwise.} \end{cases}$$

There are two phases during learning. First, the algorithm should cover the full input space and establish the neighborhood relations that preserve the input data structure. This requires competition among the majority of PE's and a large learning rate such that the PE's can orient themselves to preserve local relationships. The second phase of learning is the convergence phase where the local detail of the input space is preserved. Hence the neighborhood function should cover just one PE and the learning rate should be also small. In order to achieve these properties, both the neighborhood function and the learning rates should be scheduled during learning. So finally, the weight update in SOM training is

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \eta(n)\Lambda_{i^\circ, i}(n)(\mathbf{x}(n) - \mathbf{w}_i(n)) \quad (19)$$

where

$$\eta(n) = \frac{1}{a + bn} \quad \sigma(n) = \frac{1}{c + dn}. \quad (20)$$

The SOM has very interesting properties for data modeling. When the network converges to its final stable state following a successful learning process, it displays four major remarkable properties.

- 1) A discretization and space dimension reduction is attained via the feature map Φ [20]. That is, the continuous input space is mapped onto a discrete output space of lower dimension. This property makes the simple architecture of codebook representation feasible.
- 2) The SOM map Φ is a good approximation to the input data distribution. Luttrell's analysis of the SOM shows that it approximates the input space data distribution as a vector quantizer [22]. This property is

important since it provide a compact representation of the input space data distribution.

- 3) The feature map Φ embodies a statistical law [20]. In other words, input regions of the same size but with different number of samples occupy different domains in the output space. The larger the number of samples, the larger the domain in the output space \mathcal{A} . This property helps to make the SOM an optimum codebook of the given input space.
- 4) The feature map Φ naturally forms a topologically ordered output space, i.e., regions that are adjacent in the input space are mapped onto PE's in the lattice that are spatial neighbors [38]. This feature reduces errors caused by noise because neighborhood PE's tend to be activated in the output space \mathcal{A} .

B. Codebook in Reconstruction Space

The straightforward way to take advantage of the above properties for time series modeling is to create an SOM from the reconstruction space. So the first step is to create an embedding (2) that will map the time series to a trajectory in a multidimensional reconstruction space. We will exemplify the use of the SOM when applied to the Lorenz system. The Lorenz equations are

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(r - z) - y \\ \dot{z} &= xy - bz \end{aligned} \quad (21)$$

where σ, r , and b are constants and can be solved by numerical integration. With $\sigma = 10, r = 28$, and $b = 8/3$, the system exhibits chaotic dynamics. The time series projected along the x coordinate is shown in Fig. 3.

Fig. 4 shows the two-dimensional (2-D) projection of the Lorenz attractor initially reconstructed in three-dimensional (3-D) space.

We selected an embedding dimension of $N = 4$ (a tap delay line with three delays, four taps) and trained a 22×22 SOM with a Lorenz time series with 3000 samples. After training, a new sequence of 500 samples of the Lorenz time series is presented to the SOM. Fig. 5 depicts the sequence of winning PE's (circles) in the output SOM space connected with lines. We observe that the sequence of winning PE's creates a trajectory that is a faithful 2-D projection of the Lorenz attractor shown in Fig. 4. This example shows that the SOM mapped the trajectory at its

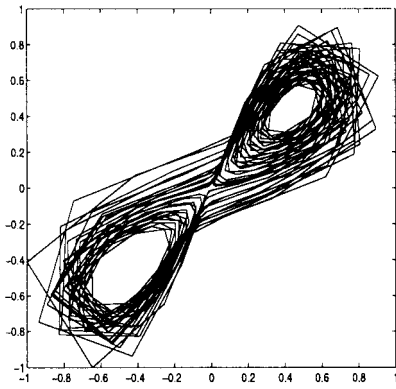


Fig. 4. Two-dimensional projection of the Lorenz attractor.

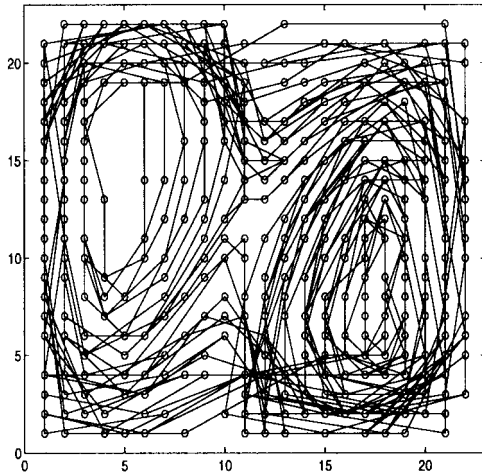


Fig. 5. Trajectory of winning PE's in the SOM output space.

input (the reconstruction space) into the discrete output 2-D space, thereby preserving the structure of the four-dimensional (4-D) reconstruction space. Repeatability of this result was very good [51]. Different training runs produced a winning PE evolution which always resembled Fig. 5 (apart from rotations and translations).

Our next step is to seek ways to utilize the SOM for dynamic modeling, harnessing its power of data representation, space discretization, and topological neighborhood preservation.

V. SOM-BASED LOCAL LINEAR MODELS

According to the state dependent theory of nonlinear processes (Section III-C), the three steps needed to identify a nonlinear system are:

- 1) embed the time series in a reconstruction space;
- 2) create a codebook of pairs $(y(n+1), \mathbf{x}(n))$;
- 3) fit a local linear model to the selected pairs in the codebook.

These steps naturally lead to our proposed block diagram of SOM-based local linear modeling depicted in Fig. 6 [35]. There are three function blocks in the topology: 1) the embedding layer, 2) the SOM, and 3) the layer of local linear predictors. The first step performs an embedding on the time series using a delay line. But notice that the

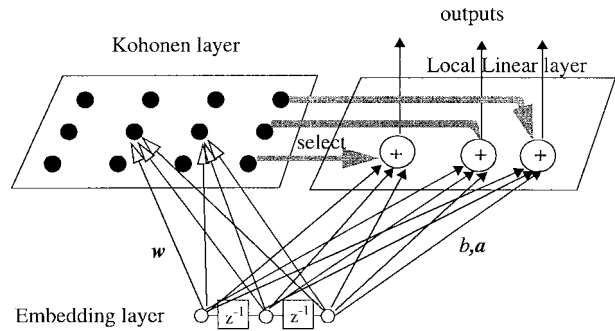


Fig. 6. Structure of Kohonen's SOM for dynamic modeling.

information for dynamic modeling is contained in the pairs $(y(n+1), \mathbf{x}(n))$, so we will augment the embedding space normally of dimension N with one more dimension to represent the functional values, i.e., the embedding space has dimension $N+1$.

The codebook is implemented by an SOM with a sufficiently large number of PE's (experimentally determined). One-dimensional (1-D) and 2-D neighborhoods have been successfully utilized in our studies using the traditional training procedure suggested by Kohonen. After training, the SOM will represent the system dynamics in the discrete output lattice just like a codebook, but enhanced with neighborhood relationships, i.e., states that are adjacent in the reconstruction space, will be represented by PE's that are neighbors in the output space (as illustrated in Fig. 5).

We propose to derive the local linear models from the trained SOM by creating an extra layer of linear PE's which receives inputs from the tap delay line (Fig. 6). The i th SOM PE has a companion linear model $\tilde{f}_i(\mathbf{a}_i, b_i)$ which represents the linear approximation of the local dynamics. The linear PE weights (\mathbf{a}_i, b_i) are computed directly from the weights of the SOM by a least square fit within a local neighborhood of size $N_L (N_L > N+2, \text{ where } N \text{ is the embedding dimension})$ centered at the current winning PE according to

$$w_{i,0} = \mathbf{a}_i^T \mathbf{w}_{ij} + b_i \quad j = 1, \dots, N_L \quad (22)$$

where $w_{i,0}$ represents the weight of the i th SOM PE connected to the first tap (most recent sample) of the embedding layer. Obviously, we first train the Kohonen network to find \mathbf{w}_{ij} and only then compute in one sweep the parameters a and b of the local linear models using least squares. Note that with this training procedure the input data is not used to derive the local models. Comparing (22) with (14), we observe that the weights of the SOM are being used instead of the input data samples to derive the local models. The weight $w_{i,0}$ is used as a proxy for $y(n+1)$, while the other weights connected to the same PE represent $\mathbf{x}(n)$. This ordering relation in the SOM weights is a reflection of the flow of time in the delay line. See the Appendix for a summary of the algorithm.

For prediction, the local dynamic model works as follows: the current input $\mathbf{x}(n)$ is placed at the input of the SOM and the winner-take-all operation will select the local

PE that best represents the current $(y(n+1), \mathbf{x}(n))$ pair. This winner activates a single linear PE that contains the weights of the local linear model and produces $\tilde{x}(n+1)$, which is a prediction of the next sample of the time series, i.e.,

$$\tilde{x}(n+1) = \tilde{f}_{i^*}(\mathbf{x}(n)) = \mathbf{a}_{i^*}^T \mathbf{x}(n) + b_{i^*}. \quad (23)$$

If iterated prediction is required [16], the generated sample is fed back to the input (delayed by one sample) and the procedure repeated as many times as needed.

There are several important advantages of using a SOM-based local dynamic model.

- 1) The SOM mitigates the discontinuity in local modeling addressed in Section III-B. Crutchfield [10] showed experimentally that dynamic modeling fails when this is not enforced. Since the SOM is trained globally, ensuring smoothness at the neighborhood boundaries [19], the PE prediction weights at the local linear layer share SOM weights. So, the continuity among the local models has a better chance of being met.
- 2) The SOM-ordered output space allows an easy implementation of linear models. Using the SOM output space for dynamic modeling has two advantages: it positions the local models in state space and identifies the local model for the current input state $(y(n+1), \mathbf{x}(n))$. This is crucial to help us build the local linear models from the output lattice when the observations are noisy and we have to perform local averaging.
- 3) The SOM input space representation simplifies enormously the construction of linear models for dynamic modeling. A major drawback of the local linear method was the need to always reconstruct the neighboring states from the time series. Performing dynamic modeling with the weights of the SOM will avoid this step. The weights of the winning PE of a trained SOM represent the center of the cluster that codes the present state. Due to the neighborhood-preserving properties of the SOM, the neighbors of the winning PE in the output space represent the neighboring input states. Hence, there is no need to go back to the time series to reconstruct the neighboring states of the current state. The weights of the winning PE and its neighbors contain all the information we need to fit the local linear model to estimate the next sample of the time series. This is a major computational savings.
- 4) A fourth advantage of the application of the SOM to local dynamic modeling is the discretization of the reconstruction space. Although the reconstruction space and the trajectory are assumed noncountable and continuous subsets of Euclidean space, the feature map is countable and discrete. This is in contrast with the other local linear modeling scenarios.
- 5) The winner-take-all operation of the SOM makes possible the selection (and training) of a local linear

model attached to each output of the network. The method has the advantages of compact state space representation of the original time series, simple state selection (winner-take-all), and state locality. The latter feature is ensured by the neighborhood preserving property which is the direct consequence of Kohonen's SOM training [18].

Hence, the SOM is an explicit quantification of the dynamics and becomes an infrastructure for local model construction. We can use the SOM to identify the system that produced the time series under investigation, or to artificially generate a time series similar to the original one by simply feeding back (through a delay of one sample) the output to the input of the system depicted in Fig. 6.

Ritter and co-workers proposed the parameterized SOM (PSOM) for the task of rapid learning multidimensional mappings in robotics and vision. Their goal was to create a continuous mapper based on the SOM which would map the discrete output of the SOM back to the input space using interpolating polynomials [39]. Their impressive results on learning the inverse kinematics of a 3-DOF robot finger with little training data show the power of the technique [49]. Later the same group proposed a Chebyshev PSOM and a local PSOM [50]. The local PSOM includes basically the same components as our work, but both the topology and the training are very different. In our approach the SOM provides simply the selection of the local linear models which work directly with the input instead of the output of the SOM. The training of our local linear models is done directly from the SOM weights, while they propose a gradient descent based on the Levenberg–Marquardt algorithm. Moreover, the link to the theory of dynamic modeling was never stated, which provides, in our opinion, new insights and a broader, more principled scope to our topology. In a statistical framework, Cherkassky [8] proposes a local linear extension to the SOM for the purpose of improving the approximation to the regression surface. Albeit similar to Ritter's work it differs in the global nature of the regression. Previous applications of SOM for dynamic modeling [23], [48] only used the competitive properties of the neural model to create the codebook. Recently, Vesanto [55] proposed a scheme that essentially followed our topology [35].

A. Dynamic Learning (DL) in the SOM

We propose below an improvement to KL for the special task of dynamic modeling by incorporating the model fitting error in the training of the SOM. This modification improves performance but the benefit of independent training of the SOM and of the local models is lost since DL trains both the SOM and the linear models at the same time. We will also utilize weighted least squares to derive the local linear models for a more precise adaptation of the linear models.

1) *Incorporating the Fitting Error in SOM Training:* The SOM obtained with the Kohonen's learning law provides an approximation of the input space with close statistical

density matching, i.e., more frequent trajectories will be represented by larger areas in the output space. For the purpose of dynamic modeling where the SOM is employed as a dynamical representation structure, the density matching is not the best criterion.

Regions that are difficult to model (normally the time series segments with large curvature) should be the ones that have larger areas in the output SOM lattice. However, this goal should not interfere with the creation of the map, in particular the topological ordering. Let $m(\mathbf{x})$ denote the PE density factor, i.e., the number of PE's representing a small volume of the input space \mathbf{X} . In the ideal case, $m(\mathbf{x})$ should be proportional to the curvature of $f(\mathbf{x})$. Although this quantity is not readily available, it can be approximated by the amplitude of the linear prediction error

$$\begin{aligned}\varepsilon(\mathbf{x}(n)) &= f(\mathbf{x}(n)) - \tilde{f}_{i^{\circ}}(\mathbf{x}(n)) \\ &= x(n+1) - (\mathbf{a}_{i^{\circ}}^T \mathbf{x}(n) + b_{i^{\circ}}).\end{aligned}\quad (24)$$

Thus the desired property becomes $m(\mathbf{x}) \propto \|\varepsilon(\mathbf{x}(n))\|$, which implies that $\varepsilon(\mathbf{x}(n))$ should be involved in the training process. Two adaptation variables can be considered: the learning rate η and the width of neighborhood function $\Lambda_{i^{\circ},i}$. Luttrell [22] shows that the neighborhood function $\Lambda_{i^{\circ},i}$ interferes with the power of the noise mismatch, so any disruption to the decreasing trend of the neighborhood function may cause instability during training. Instead, the conventional learning rate η can be modified to be a function of $\varepsilon(\mathbf{x}(n))$. A larger value of η_{ε} will tend to recruit more PE's for the neighborhood function. Considering the constraint that the magnitude of the learning rate is less than unity, a straightforward way to involve ε in the training process is

$$\eta_{\varepsilon} = \frac{1 - \exp(-\mu(\eta + \bar{\varepsilon}))}{1 + \exp(-\mu(\eta + \bar{\varepsilon}))}\quad (25)$$

where μ is a constant that controls the slope of the exponential increasing function, and η is the conventional learning rate. A normalized value of $\bar{\varepsilon}$ should be used to offset the amplitude variation

$$\bar{\varepsilon} = \frac{\|\varepsilon(\mathbf{x}(n))\|}{\|x(n+1)\|}.\quad (26)$$

When $\bar{\varepsilon}$ is small, the training defaults to the basic KL. However when $\bar{\varepsilon}$ is large, learning is emphasized on the neighborhoods that represent large curvature trajectories. This DL process will produce a feature map with the density factor $m(\mathbf{x})$ consistent with the local complexity of the given dynamics. Similar modifications have been proposed in [28] to increase the accuracy of the SOM in nonparametric regression, but here the estimate of the second derivative is more direct. The authors reported good results.

2) *Weighted Least Square Estimation*: The least square algorithm was previously described for the construction of the local linear models [35]. We need to use at least $N+1$ PE's to derive the local linear model, and we choose the winning PE neighbors for that purpose (the receptive field).

In least squares, all the elements of the receptive field are equally weighted for the construction of the local model. Intuitively this may not be an optimal choice, since PE's closer to the center represent dynamics that are closer to the winning PE. A poor weighting of the PE contributions may cause the estimation of the linear models to deviate from the optimal orientation and hinder the reduction of the discontinuity at the boundaries. With this concern in mind, we propose to make the contribution of each PE to the collective response inversely proportional to its metric distance to the center, which yields a weighted least squares solution [41]. Weighted least squares is a very well known technique, and it has been proposed to improve the accuracy of the SOM for nonparametric regression [8].

Mathematically, the weighted least square solution is equivalent to inserting a weighting matrix to the optimization process

$$\min(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})^T \mathbf{S}(\mathbf{y} - \mathbf{x}\boldsymbol{\theta})\quad (27)$$

where $\mathbf{y} = \mathbf{x}\boldsymbol{\theta}$ and \mathbf{S} is a nonsingular symmetric matrix. The weighted least squares solution satisfies

$$\mathbf{x}^T \mathbf{S}(\mathbf{y} - \mathbf{x}\boldsymbol{\theta}) = 0.\quad (28)$$

Therefore the solution becomes

$$\tilde{\boldsymbol{\theta}}_{\text{WLS}} = (\mathbf{x}^T \mathbf{S} \mathbf{x})^{-1} (\mathbf{x}^T \mathbf{S} \mathbf{y})\quad (29)$$

and it is unique if and only if $\mathbf{x}^T \mathbf{S} \mathbf{x}$ is invertible. Using the Euclidean metric for the distance d_i to the center, we construct a diagonal matrix \mathbf{S} defined as

$$\mathbf{S} = \{s_{ij}\}_{1 \leq i, j \leq N_L}\quad (30)$$

where $s_{ij} = 0$ for $i \neq j$, and

$$s_{ii} = 1 - \frac{d_i^m}{\sum_{k=1}^m d_k^m}.\quad (31)$$

The experimental results show that m ranging from two to four provides the best performance. In our application

$$\mathbf{y} = \begin{bmatrix} w_{1(0)} \\ \dots \\ w_{N_L(0)} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} [\mathbf{w}_1^T \ 1] \\ \dots \\ [\mathbf{w}_{N_L}^T \ 1] \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix}.$$

VI. EXPERIMENTAL RESULTS

The proposed topology for local dynamic modeling is tested in the Lorenz system, which exhibits chaotic dynamics with large Lyapunov exponent for $\sigma = 10, b = 8/3, r = 28$. The Lorenz time series is sampled at 10 Hz. The SOM system used here is composed of a 2-D grid of 22×22 PE's. The embedding dimension is chosen as four, and so the dimension of the state input during the training process is five. The parameters for the learning rate and the neighborhood function in (19) are $a = 1, b = 1/500, c = 1/8, d = 1/4000$. Further testing of the system is presented in [51]. Here we would like to show the reliability of the method to model the Lorenz system and also the improvements obtained with the new training rule.

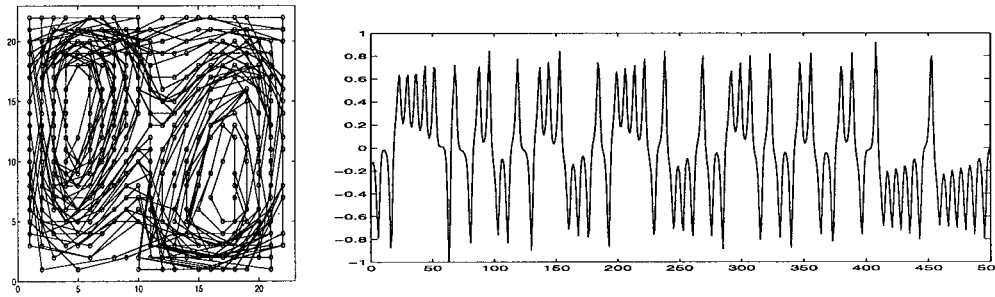


Fig. 7. Trajectory of the winning PE and generated Lorenz time series.

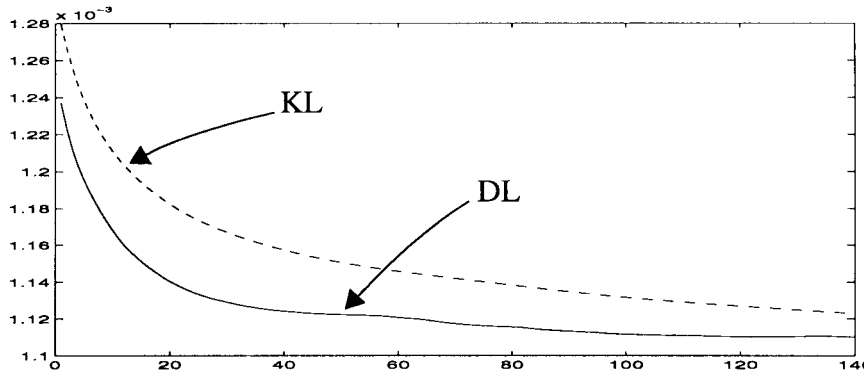


Fig. 8. Comparison of the learning curves: KL and the new rule.

The SOM is trained with the conventional KL and with the improvement of Section V-A, which will be called DL here. The training is performed with a 3000-sample segment of the time series for 150 epochs. After training, the local models for each PE are constructed using the weighted least algorithm of (29) with $m = 4$. With all the weights fixed, an initial point in reconstruction space is loaded and the whole system is iterated as an autonomous dynamical system, i.e., its output is fed back to its input to generate a time series. This generation model is called iterated or recursive prediction, and it mimics the assumption made for dynamic modeling. Fig. 7 shows the 500-sample trajectory of the winning PE in the output space and the corresponding autonomous prediction.

Although visually the generated time series resembles the Lorenz signal, the accuracy of dynamic modeling is quantified by comparing the dynamic invariants [correlation dimension and largest Lyapunov exponent (LLExp)] between the original time series and the generated time series as suggested in [34]. We use the Grassberger and Procaccia algorithm [13] to estimate the correlation dimension and Wolf's algorithm [54] to estimate the LLExp.

Fig. 8 shows that the DL outperforms the conventional KL for the task of dynamic modeling since it produces a smaller normalized MSE during training (0.0011 versus 0.0012).

An example of the modification of the SOM output neighborhoods produced by the new training is illustrated in Fig. 9. The waveform on top shows an enlarged segment of the Lorenz time series used for training. The segment within the broken lines is selected because it produces one

Table 1 Comparison of Dynamic Invariants

Time series	LLExp (bits/sec)	Correlation Dimension
original	2.17	2.07
dynamic learning	2.09	2.08
Kohonen learning	1.83	2.01

of the largest prediction errors. The sample that is predicted is shown by the cross. Fig. 9(b) and (c) depicts the contour plot of the SOM activity bubble around the winning PE that corresponds to the selected segment of the time series. Fig. 9(b) depicts the contour plots obtained with the new dynamic learning, while Fig 9(c) corresponds to regular Kohonen learning. There are two things to note: first, although these figures correspond to two different training runs of the SOM, and the winning PE actually appears in two distinct places of the output space, both contour plots are rather similar; second, the area in the output space that codes the peak of the time series is larger for dynamic learning than for regular learning (the first contour line is 0.9 of the maximum). This means that the neighborhood field of this PE was enhanced by dynamic learning which is the expected result (more PE's are recruited enhancing the resolution for the high-curvature portion of the trajectory and decreasing the prediction error). Table 1 summarizes the results of the estimated dynamic invariants in the generated time series versus the original time series.

Many more tests of consistency both in time characteristics and on multiple training runs are reported in [51]. We summarize by saying that our proposed SOM-

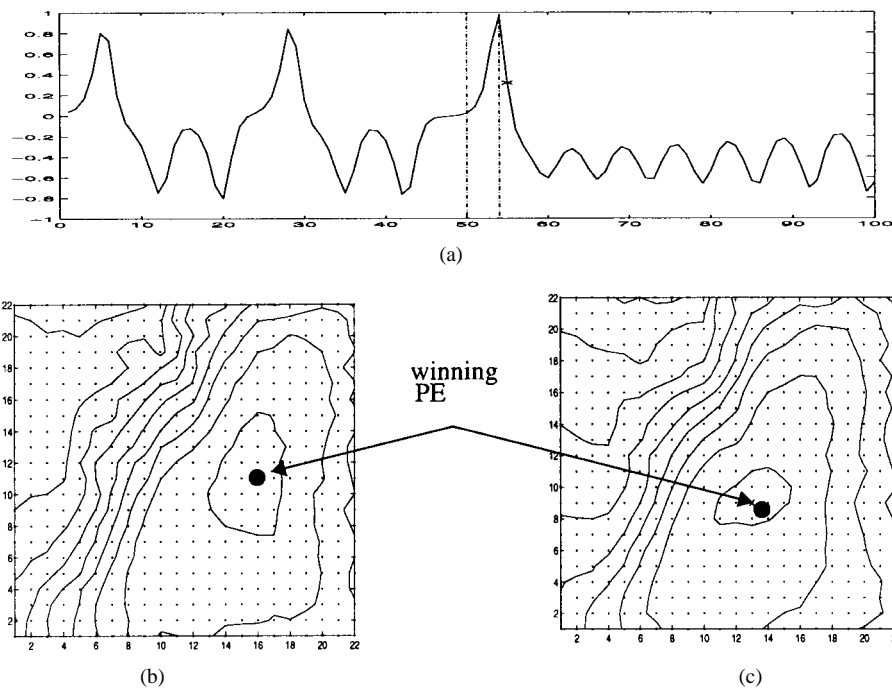


Fig. 9. Comparison of the recruiting areas at the SOM output for the two learning rules.

based locally linear dynamic model is very robust. It never diverged during our experiments, while both the state-dependent model proposed in [10] and a different model created around a simple clustering algorithm [2] showed divergence. Once the system switches to the wrong PE during iterated prediction it will produce a burst of samples that do not conform with the model. The trajectory simply diverges and never recovers. Hence, we conclude that both the neighborhood preservation property of the SOM and our proposed weighted least square estimation of the local modes is preferable for successful local linear dynamic modeling.

Finally, we would like to present results for the modeling of a real-world signal. We selected the laser time series from the Santa Fe Time Series Competition [52]. This time series is difficult to model due to the collapses of the orbits in the attractor. A 28×28 SOM with $a = 1$, $b = 1/500$, $c = 1/8$, $d = 1/400$ with an embedding of five and a $\mu = 2$, was trained for 130 epochs on a 3000 sample segment of the laser time series. The output of the autonomous dynamic model obtained through iterated prediction is shown in Fig. 10(a) which resembles the bursting behavior of the original signal. The spectra of the original and generated time series are shown in Fig. 10(b) and (c), which demonstrates that the modeling preserved the essential of the deterministic motion on the attractor.

VII. DESIGN OF SWITCHING CONTROLLERS BASED ON SOM

A. Problem Specification

The previous development of local dynamic modeling has many applications, but here we will address its use in system identification and how it can be extended to create a

multiple model switching controller. The plant is the 16-ft Transonic Tunnel at NASA Langley Research Center. The problem is how to ramp up or down and regulate the wind speed (Mach number) in the tunnel such that aerodynamic studies in scaled down models can be conducted with high accuracy. The required Mach number accuracy is ± 0.003 of the set point. The difficulty is that modifying the orientation (angle-of-attack) of the model relative to the airflow produces changes in the effective cross section of the tunnel test section, thereby changing abruptly the wind speed. This in turn requires quick modification of the main drive motor rpm to stabilize the wind speed at the Mach number set point. The tunnel dynamics vary widely with the Mach number and the angle of attack of the model. Due to the huge power of the main drive system (50 MW) the only available controls are three regimes (increase/maintain/decrease the rpm by a nominal amount, which are represented respectively by 1/0/-1), and the time each control is applied. So we can model the possible control inputs as a family of time series with three amplitude levels, which differ only in their time structure.

B. The Control Strategy

This control application is fundamentally one of regulation around a set point in a nonstationary environment. The leading characteristic of this application is the fast and unpredictable changes in dynamics encountered when the model's angle of attack is modified. The existing automatic controller is a highly tuned but fixed table look-up of drive motor commands based on the error at a given Mach number [6]. An adaptive controller that meets the set point specifications will be very slow to adapt. Alternatively, we developed in advance a set of local dynamic models for the

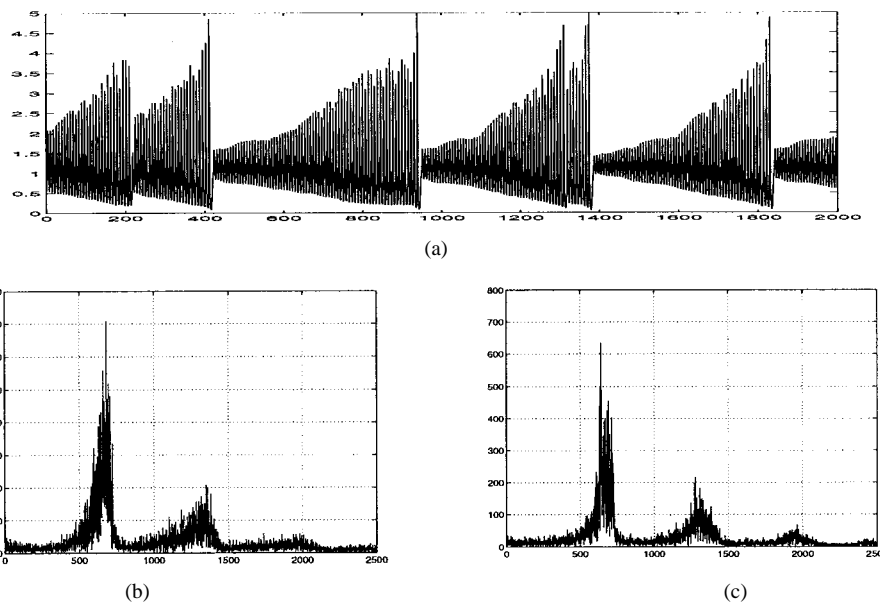


Fig. 10. Iterated prediction of the laser time series and fast Fourier transform (FFT) spectra.

tunnel dynamics and switch them according to the measured Mach number history.

The design of the controller has two phases: first the wind tunnel dynamics have to be identified; then an appropriate control action has to be decided. The local modeling approach based on an SOM is especially well suited to model the changes in tunnel dynamics when the model position is modified. In fact, the SOM will provide a codebook representation of the tunnel dynamics and will organize the different dynamic regimes in topological neighborhoods. The difficulty is that the wind tunnel cannot be considered as an autonomous system since it is excited by the controller at all times.

We decided to cluster the possible control inputs into a set of representative inputs, and design a different SOM per each representative control input. As long as we can create a meaningful and limited catalog of control inputs this is a feasible alternative. The quantized nature of the control input works on our side. Moreover, experience shows that the tunnel operators develop a “style” when dealing with a specific situation. The practical usefulness of a style implies that it is possible to define automatically a set of representative control inputs and also corroborates the opinion that a limited catalog of control inputs suffices to ramp the Mach number up and down and regulate around the set point.

Each SOM will model the forced tunnel dynamics, i.e., the combination of the tunnel with one of the representative controls, which can be alternatively thought as a collection of “autonomous” wind tunnels. Each SOM is trained with full range Mach number data under many different experimental conditions (i.e., different model aircrafts and angle-of-attack). When a given SOM PE fires, its corresponding predictor is the best available local linear descriptor for the tunnel dynamics with the applied control input. Hence the system can predict the evolution of the dynamics in the

near future. Since the requirement is to regulate the Mach number, the system may choose several possible control commands from the catalog and see which one best meets the Mach number specification. This reasoning points to the implementation of a predictive multiple model switching controller (PMMSC). The block diagram is presented in Fig. 11.

We are not the first ones to propose a switching control architecture. A multiple model structure with switching has been proposed by Narendra [25]. The architecture of Narendra’s MMSC also utilizes N predictive models of the plant which are obtained by observing the system over a long period of time. Each has a corresponding controller which is designed offline. The outputs of the predictive models are compared with the plant output and the best fit to the plant output (according to a performance index) selects one of the models and the corresponding controller at each time instant. This corresponds to the switching part of the architecture. The implementation of the switching scheme employs some hysteresis to prevent arbitrarily fast switching between models. In a more recent paper [26], stability results for an all-fixed models controller were established for linear systems under some mild assumptions. In particular, it is shown that if there is at least one model that is sufficiently close to the actual plant and there is a nonzero waiting time between switches from one model to another, then the overall system is stable, given that each fixed model is stabilized by its corresponding controller. An even more recent paper [27] introduces two classes of approximate nonlinear input–output models which reduce the computational complexity of designing a controller based on the fact that the approximate models are linear in the control input.

Our approach can be considered an improvement over this scheme for the following reasons. The SOM approach models all the dynamic regimes observed during training

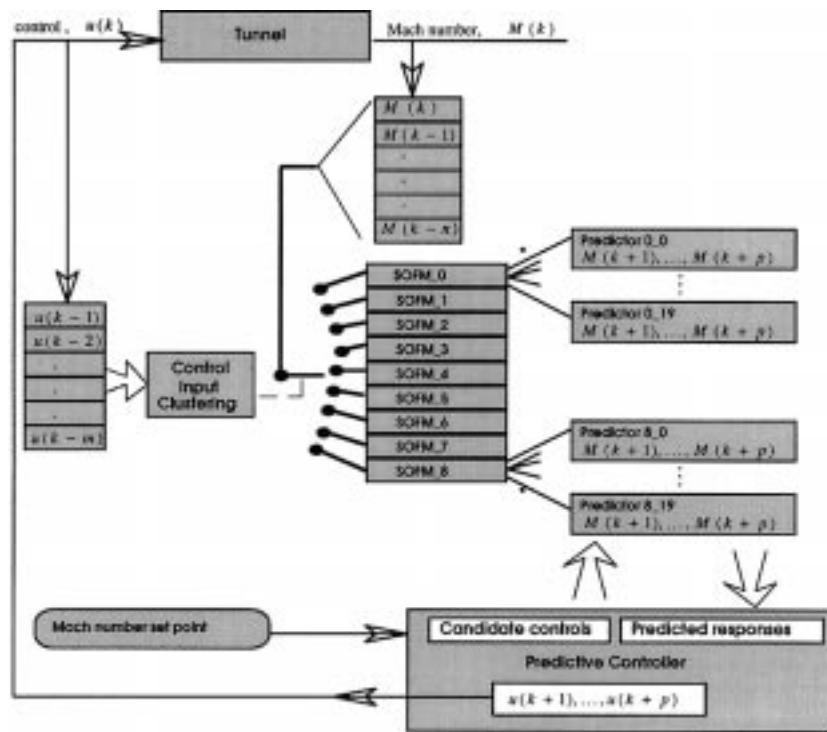


Fig. 11. Block diagram of the PMMSC.

and automatically divides the dynamical space by the number of available PE's (their number specifies the granularity of the Voronoi tessellation). So we are guarantying coverage of the full dynamic space observed during training, which is a difficulty in Narendra's multiple models. Moreover, neighboring SOM PE's represent neighboring regions in the dynamic space. Therefore, wrong assignments in the winning PE due to noise tend to activate similar dynamic models. Kohonen demonstrated this property of the SOM for noisy coding of images, and here the problem is the same [19]. There is no such neighborhood relationship in Narendra's scheme so the selection of a wrong predictive model may temporarily cause a poor control regime. In our case the system tries several control sequences with the winning model and chooses the best one, so even if the dynamic model is not the most appropriate, there is an extra flexibility to match the set point with small error. Finally, our architecture is much more compact, consisting of an SOM and a set of linear predictors obtained directly from the SOM weights. After training, the PMMSC was implemented in an old Intel 486 and controlled the wind tunnel for 9 h, collecting the data presented in this paper.

C. Design of the PMMSC

As seen in Fig. 11, both the control input $u(k)$ and the Mach number response $M(k)$ are used as inputs to the PMMSC. The control input is embedded in an $m = 50$ dimensional space to represent the representative control sequences being applied to the tunnel. These control input sequences are clustered in C control classes ($C = 7$). Here we use the available knowledge of operating the wind

tunnel to select the representative control inputs to ramp the Mach number up and down and to regulate the set point. The most recent 50 samples of the control input are clustered to the closest representative control input (in a $L1$ norm sense). Two additional control classes which are only ten samples long are used for identification purposes only.

The winning cluster selects one of the 7 ($C = 0, 1, \dots, 6$) SOM networks. The input to each of these SOM is obtained from the past history of the Mach number responses. An embedding of the Mach number response in a space of $N = 50$ is performed. Each of these SOM is trained in the full operating range of the Mach numbers and works as a codebook for the wind tunnel dynamics under the specified control input. Two additional SOM's, $C = 7, 8$, operate essentially in parallel on the $m = 10$ subspace to identify the most recent input-output dynamics while ramping up or down. The size of each SOM here is 20×1 , i.e. a linear array of PE's. This number of output PE's provided enough precision in the local linear models to achieve the required 0.003 tolerance in the Mach number [24].

Ensembles of Mach number responses resulting from the application of each control prototype were extracted from many hours of wind tunnel test data, sampled at 3 Hz (~ 2 million samples or 40000 vectors). Next, each ensemble of responses corresponding to one of the seven control prototypes was clustered using the SOM, trained over 10000 epochs, as explained in Section IV-A. Each PE of the SOM is augmented with a local linear model as described in Section V, and the predictor trained as described in the same section. Fig. 12 shows the weights of two converged SOM for the Mach number responses

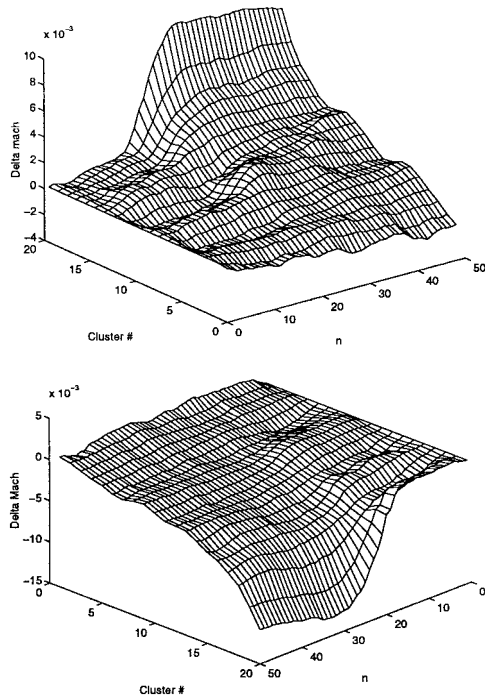


Fig. 12. SOM for the Mach number responses of the regulating control prototypes.

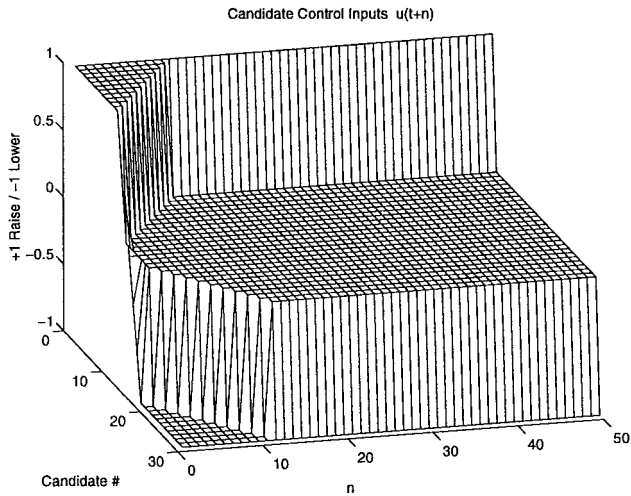


Fig. 13. Repertoire of control inputs.

belonging to the two regulating control inputs. Notice the smooth organization of the weights across the neural field, from which the local linear models are derived.

The local linear model associated with each PE is used to predict the tunnel dynamics for the next 50 samples under the repertoire of our candidate control inputs. We developed a family of 29 control waveforms, shown in Fig. 13.

These control inputs are developed from experience with the goal of ramping up or down and regulating the Mach number. Each candidate control is fed to the local linear model of the winning PE producing a vector M of Mach number responses which is tested against the desired set point (M_{sp}) within the last 30 points of the 50 point

trajectory to emphasize steady state matching, i.e.,

$$E = \sum_{k=21}^{50} \|M(k) - M_{sp}(k)\|. \quad (32)$$

The control input that produces the smallest Euclidean distance to the set point specification in steady state is elected as the best control and sent to the controller. The control $u(k)$ is updated every sample when ramping or steady-state control sequences are selected, but when regulatory control sequences are selected the entire 50-sample control sequence is applied. Thus, when actively regulating about a set point, the switching between models is performed at most every 50 sample periods.

VIII. EXPERIMENTAL CONTROL OF THE WIND TUNNEL

A. Comparison of PPMSC Control to Existing Automatic Controller and Expert Operator

Fig. 14 compares the regulating performance of the existing gain-scheduled control, an expert operator, and the PPMSC under similar conditions over a nominal fifteen minute interval. The first row of figures display the Mach numbers (with the ± 0.003 tolerance lines), followed by the control commands in the second row and test model angle-of-attack (disturbance) for each of the three control schemes. The control commands are amplitude coded. Control commands whose duration is 0.3 s are displayed at amplitude one. Control pulses of 0.1 s are shown with magnitude 0.33, and 0.2 s pulses are shown with magnitude 0.66. Control pulses of less than 0.1 s in duration are ineffective to produce a change in the fan drive rpm at any condition and are not used.

Derived metrics to quantify the comparisons between the three control strategies are the time out of tolerance and the $L1$ norm of the control input $u(k)$. The time out of tolerance is the cumulative sum of time that the measured Mach number deviates beyond the required tolerance of 0.003

$$T_{out} = \sum_{k=1}^N \alpha \Delta t(k) \quad (33)$$

where

$$\begin{cases} \alpha = 0, & |M(k) - M_{sp}(k)| \leq 0.003 \\ \alpha = 1, & \text{otherwise} \end{cases} \quad (34)$$

and $\Delta t = t(k) - t(k-1)$. The $L1$ norm of the control command is

$$L1(u) = \sum_{k=0}^N |u(k)|. \quad (35)$$

For this particular experiment, the angle-of-attack begins to mildly disturb the Mach number at approximately 5° . Table 2 lists the reduction in the standard deviation of the Mach number, time out of tolerance, control effort, and time required to complete the sweep through the desired range of angle-of-attack while maintaining Mach

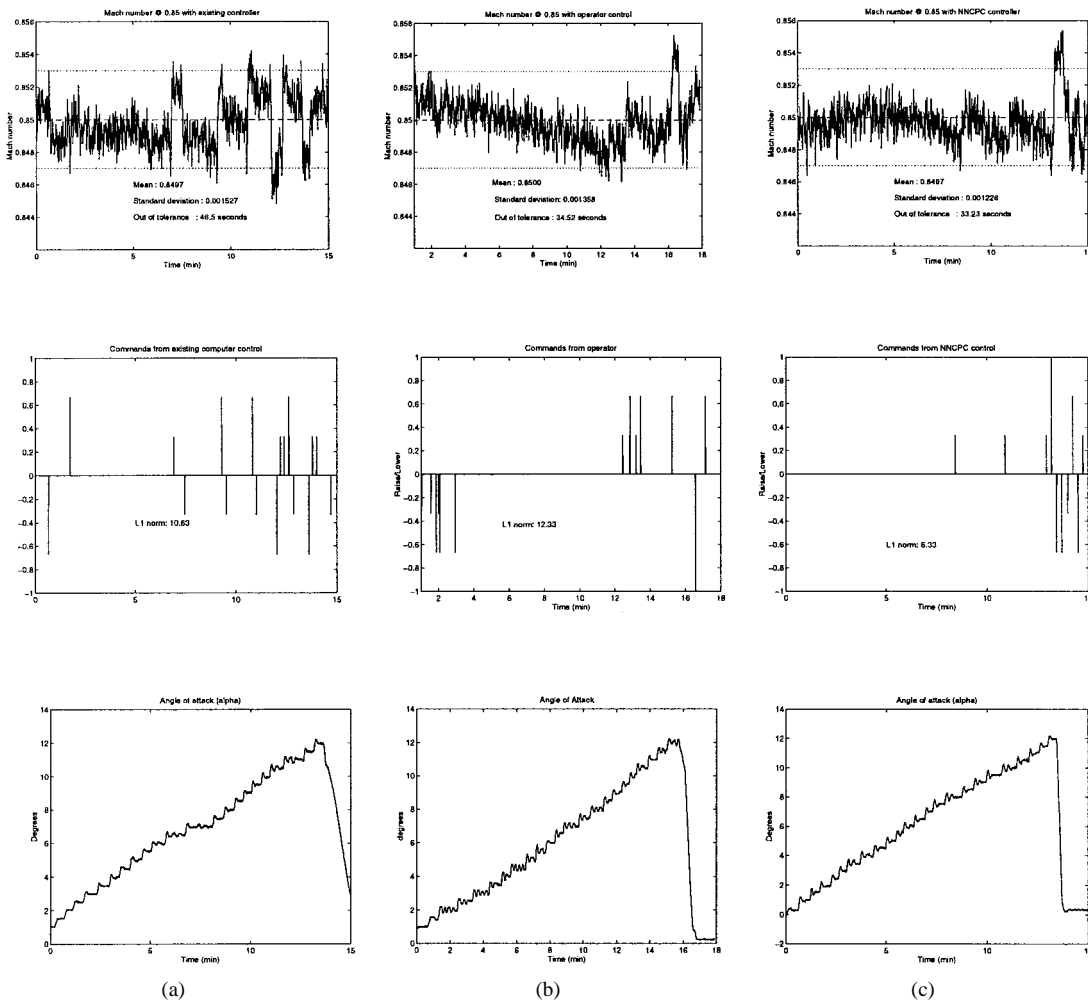


Fig. 14. Comparison of (a) the existing controller, (b) expert operator, and (c) PMMSC for a 15 min run.

Table 2 Comparison of Control Schemes

	Controller (Cont)	Operator (Oper)	PMMSC	% reduction (Cont/Oper)
Mean	0.8497	0.8500	0.8497	-
SD	0.001527	0.001358	0.001226	20/10
T _{out} (sec)	46.5	34.52	33.2	29/4
L1	10.6	12.33	6.3	40/49
Alpha sweep (sec)	886	930	806	9/13

number steady for this comparison. For this particular test condition, the PMMSC performs slightly better than the expert operator but with much less control effort and less time to complete the alpha sweep. Compared to the existing automatic control, the PMMSC maintains the Mach number within the desired tolerance with less control effort, thereby completing the alpha sweep in less time, which is the most important figure of merit for the utilization of the wind tunnel facility (relates to the time necessary to conduct the experiment).

An additional metric on the control, the control density, was calculated by taking the sum of the absolute value of

the control over a 50-sample sliding window

$$\xi(k) = \sum_{i=0}^{49} |u(k-i)|. \quad (36)$$

The control density is used to compare the sparseness of the control between the PMMSC, the existing controller, and an expert operator. This quantity measures the accuracy of the present control input, so in the PMMSC case it is a measure of the accuracy of the local linear models to predict the tunnel dynamics. As illustrated in Fig. 15, the PMMSC is clearly the most sparse, but it allows for increased density of control when demanded by the external disturbance. In this respect it is similar to the variation in control density employed by the expert operator. This is in contrast to the existing automatic control, with fixed gains for a particular operating point resulting in a narrow range of control density.

Fig. 16 compares the result of controlling the Mach number to several different set points over a nominal 28 min interval, which involved operating point changes and regulation. The goal here is to show how the PMMSC handles ramping up and down the tunnel Mach number. Mach number set points of 0.95, 0.9, and 0.6 are common

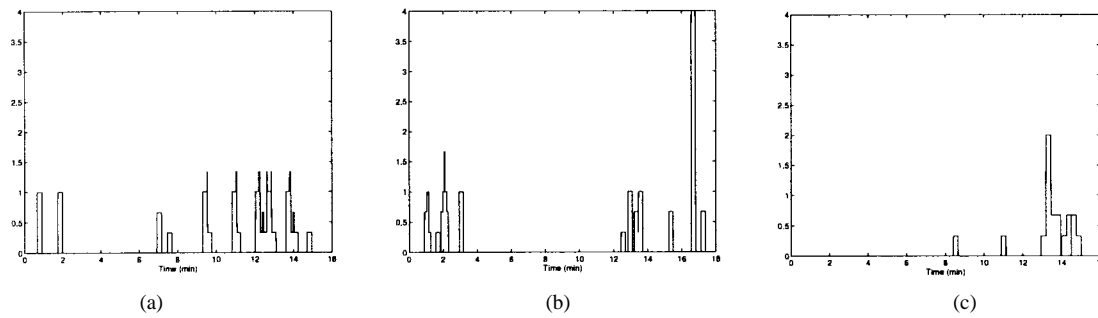


Fig. 15. Comparison of control densities: (a) controller, (b) operator, and (c) PMMSC.

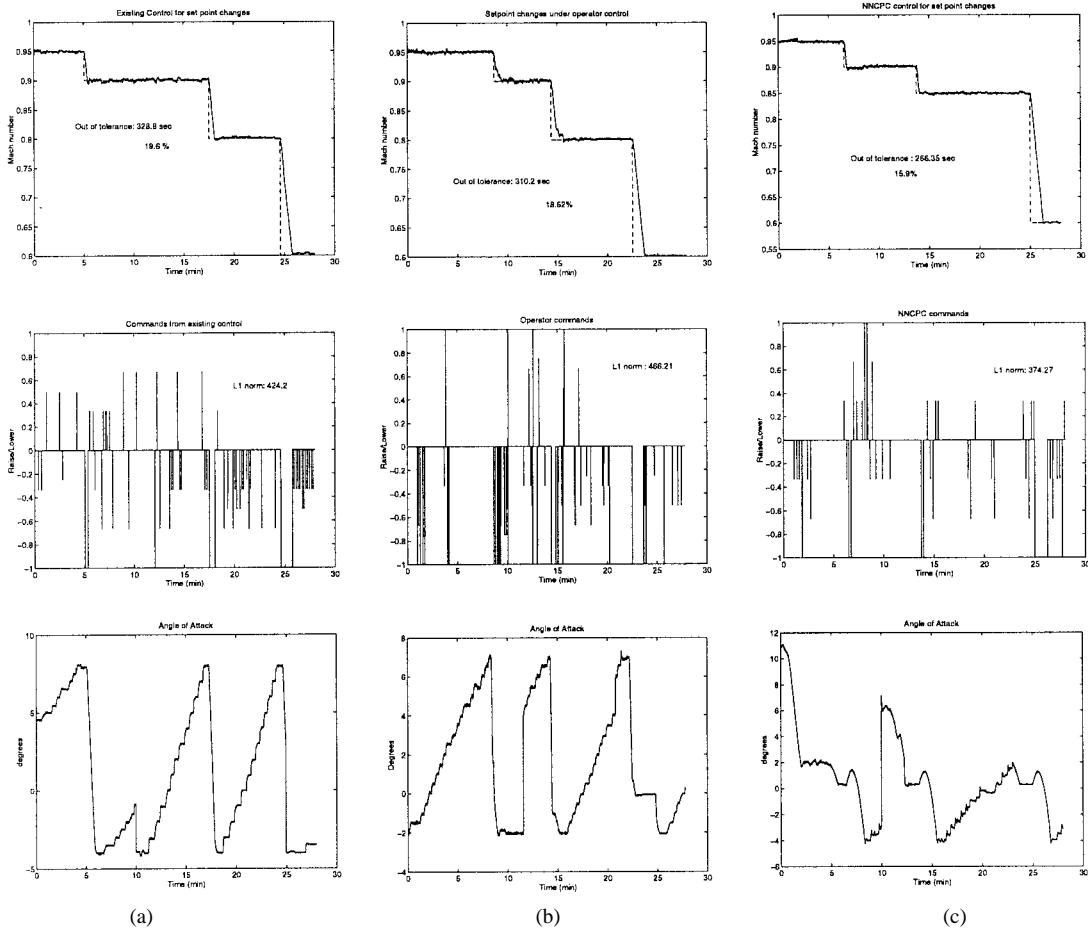


Fig. 16. Comparison for controlling to several different set points: (a) controller, (b) operator, and (c) PMMSC for a 28 min run.

to all three controllers. The PMMSC controls the Mach number to the intermediate value of 0.85 instead of 0.8 for the operator and existing controller. This difference is minimal, so this still provides a reasonable basis for comparison of the controllers. The angle-of-attack was varied extensively during all three runs. Again, the PMMSC maintains the Mach number within tolerance for a higher percentage of the time, with less expenditure of control effort. Table 3 lists the time out of tolerance and the $L1$ norm for the three runs. The PMMSC reduces the time out of tolerance on the order of 15–20% compared to the existing automatic control or an expert operator.

Table 3 Comparison for Control Schemes

	Controller	Operator	PMMSC	% reduction
T_{out} (sec)	329	310	266	19/16
$L1$ norm	424	466	374	12/20

IX. CONCLUSIONS

Nonlinear dynamical modeling is a rich area with many possible applications. Nonlinear modeling seeks to estimate the nonlinear system that produced the time series under study. Hence it requires a nonlinear global model or a set of

local linear models. Local linear modeling is conceptually elegant and requires mild assumptions about the underlying dynamics, but it is hindered by possible discontinuities among the local linear models.

We propose an innovative modeling scheme based on an SOM, which works not only as an enhanced clustering algorithm that preserves neighborhoods of the reconstruction space dynamics, but also as an estimator of the local linear models. We presented the mathematical principles that support the clustering model and derived a procedure to directly estimate the parameters of the local linear models from the SOM weights. The proposed enhanced SOM architecture implements naturally the necessary steps of identifying a nonlinear dynamical system from a time series.

Our approach has several advantages. First, the SOM-based modeling is computationally more efficient than other local models, which makes it very appealing for practical system identification applications. Second, since in our approach the SOM is trained globally to cluster the dynamics, the problem of discontinuities among the local models is reduced. In order to minimize the discontinuities, we applied a weighted least squares estimation to derive the local linear models. Third, we improved KL for this task. The conventional KL responds only to the density of samples in the input space, while here the fitting error to the trajectory is also important. Therefore we included an extra term in the KL which assigns more PE's to regions of the trajectory where the error is larger. This improves the fidelity of the mapping.

It is very interesting to observe that in the appropriately trained SOM for dynamic modeling, the trajectory of the winning PE in the output space describes a complicated path that is a projection of the reconstruction space trajectory. The beauty of the approach is that the complexity of the trajectory is naturally decomposed in the appropriate switching among very simple linear models, which leads to accurate dynamic modeling and very direct engineering applications in system identification and controls.

Our proposed SOM approach is relevant for the switching controllers for two reasons: first it guaranteed that the repertoire of dynamics used for training the SOM are appropriately represented at the output of the SOM; second, the same system that is performing the mapping is utilized to derive the local linear models, which makes the identification of the plant very compact and computationally efficient. In a sense, the diverse plant dynamics are captured in a compact look-up table of linear models.

We applied this concept to the control of the NASA Langley Transonic Wind Tunnel with very good results. The control problem is one of regulating the set point of the plant under nonstationary loads (produced by the modification of the angle of attack of the aircraft model inside the tunnel). There are several peculiarities to this problem that made this solution so simple and accurate. First, the wind tunnel is stable, which means that the problem is one of regulation. Second, there is plenty of data to train a locally linear dynamic model of the tunnel

dynamics under different load conditions. Third, the control input is a time series with three levels, which enables a meaningful clustering of the control input is a small set of representative control commands.

Hence we can discretize the control space, and simplify the modeling of the tunnel dynamics with the control input as a set of seven "different" wind tunnels. For each we modeled the dynamics with our proposed SOM approach. Once the selected SOM determines the winning PE, the system has a local linear model of the plant which can be used to automatically predict the Mach number produced by a small, but effective, set of control commands. Hence the control command that best meets the set point is used as the next command input. Regulating around a set point is distilled into the selection of a local linear model that is used to predict the tunnel dynamics and determines which is the best possible control input (hence the name predictive multiple model switching controller). The PMMSC is easily implemented for on-line operation in a 486 PC and was utilized to control the wind tunnel during 9 h for three different aerodynamic modeling experiments. We demonstrated the accuracy of our controller during these runs and showed that it outperformed the existing controller and even trained operators.

We realize that PMMSC is not a general control architecture. The PMMSC is set in advance, so it is not able to adjust to events that are very different from the ones used during training. But what is remarkable is that switching appropriately among a set of 140 local models was able to control to within 0.003 the Mach number of the wind tunnel during normal experimental conditions, which create nonstationary loading. It is very probable that conditions different from the ones used to train the system were encountered during the 9 h test runs, but as long as the new dynamics could be approximated reasonably and locally by the developed dynamic code books the performance remained within bounds. This attests to the power of local linear dynamic modeling and of the proposed SOM-based architecture. This reasoning also clearly points the direction of further work. Once we have this simple method of capturing *a priori* the dynamics about the task, the goal should be to integrate the SOM dynamic infrastructure in more complex control architectures, such as the adaptive critics or adaptive controls. These methods tend to work with no *a priori* information and therefore are very slow to converge. Inclusion of the SOM will speed up the convergence, preserving at the same time their characteristics of adapting to new situations.

APPENDIX

Let X , A , and Φ denote the continuous input space, spatially discrete output space and a competitive projection respectively. The proposed nonlinear modeling scenario follows three steps: 1) reconstruction of the state space from the input signal; 2) training the SOM; and 3) estimation of the locally linear predictors.

1) *Reconstruction Of The State Space From The Training Signal:* Embed the time series in a space of dimension

$N + 1$, where N is Takens' embedding dimension ($N \geq 2D_e$). A sequence of $N + 1$ -dimensional state vectors $[x(n + \tau), \mathbf{x}(n)]$ is created from the time series, where $\mathbf{x}(n) = [x(n), x(n - \tau), \dots, x(n - (N - 1)\tau)]^T$ and τ is the appropriate time delay.

2) *Training the SOM Model:* This step is accomplished via the Kohonen learning process (or preferably with the improved dynamic learning of Section V-A). With each vector-scalar pair $[x(n + \tau), \mathbf{x}(n)]$ presented at the network input, train the system with (15). The learning process adaptively discretizes the continuous input space $\mathbf{X} \subset \mathbf{R}^{N+1}$ into a set of K disjoint cells \mathbf{A} to construct the mapping $\Phi: \mathbf{X} \rightarrow \mathbf{A}$. This process continues until the learning rate decreases close to zero and the neighborhood function covers about one output PE. After learning, the output space lattice \mathbf{A} represents the input space \mathbf{X} . The mapping Φ represents the input space data distribution and preserves local neighborhood relations.

3) *Estimate the Locally Linear Predictors:* Once the SOM is trained, for each PE $u_i \in \mathbf{A}$, the corresponding local linear predictor coefficients $[\mathbf{a}_i^T, b_i]$ are estimated based on $\alpha_i \subset \mathbf{A}$, which is a set of N_L elements in the neighborhood of u_i including u_i itself. Each element u_i , has a corresponding weight vector $[w_{i0}, \mathbf{w}_{ij}]^T \in \mathbf{R}^{N+1}$, where $\mathbf{w}_{ij} = [w_{i1}, w_{i2}, \dots, w_{iN}]$. The local prediction model $[\mathbf{a}_i^T, b_i]$ is fitted in the least-square sense to the set of weights in α_i , i.e.,

$$w_{i0} = b_i + \mathbf{a}_i^T \mathbf{w}_{ij}. \quad (37)$$

To ensure a stable solution of the above equations, α_i must have more than $N + 1$ elements. Thereafter for each output unit $u_i \in \mathbf{A}$, a unique linearly local model function $\tilde{f}(\mathbf{a}_i(\cdot), b_i(\cdot))$ is constructed.

REFERENCES

- [1] H. D. I. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring, "The analysis of observed chaotic data in physical systems," *Rev. Mod. Phys.*, vol. 65, no. 4, p. 1331, 1993.
- [2] S. C. Ahalt *et al.*, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, p. 277, 1990.
- [3] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day, 1976.
- [4] D. S. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.
- [5] P. Bryant, R. Brown, and H. D. I. Abarbanel, "Lyapunov exponents from observed time series," *Phys. Rev. Lett.*, vol. 65, p. 1523, 1990.
- [6] F. J. Capone, L. S. Bangert, S. C. Asbury, C. T. L. Mills, and E. A. Bare, "The NASA Langley 16-foot tunnel," NASA, Langley, VA, NASA Tech. Paper 3521, Sept. 1995.
- [7] M. Casdagli, "Nonlinear prediction of chaotic time series," *Phys. D*, vol. 35, p. 335, 1989.
- [8] V. Cherkassky and H. Lari-Najafi, "Constrained topological mapping for nonparametric regression analysis," *Neural Networks*, vol. 4, p. 27, 1991.
- [9] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of adaptive methods for function estimation from samples," *IEEE Trans. Neural Networks*, vol. 7, pp. 969–984, July 1996.
- [10] J. P. Crutchfield and B. S. McNamara, "Equations of motion from a data series," *Complex Systems*, vol. 1, p. 417, 1987.
- [11] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59, no. 8, p. 845, 1987.
- [12] M. Giona, F. Lentini, and V. Cimagalli, "Functional reconstruction and local prediction of chaotic time series," *Phys. Rev. A*, vol. 44, pp. 3496–3502, 1991.

- [13] P. Grassberger and I. Procaccia, "Characterization of strange attractors," *Phys. Rev. Lett.*, vol. 50, no. 5, pp. 346–349, 1983.
- [14] V. Guilleman and A. Pollack, *Differential Topology*. Englewood Cliffs, NJ: Prentice-Hall, 1974.
- [15] S. M. Hammel, "A noise reduction method for chaotic systems," *Phys. Lett. A*, vol. 148, p. 421, 1990.
- [16] S. Haykin and J. Principe, "Dynamic modeling of chaotic time series with neural networks," *IEEE Digital Signal Processing Mag.*, pp. 66–81, May 1998.
- [17] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
- [18] T. Kohonen, *Self-Organizing Feature Maps*. New York: Springer-Verlag, 1995.
- [19] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.
- [20] E. J. Kostelich and J. A. Yorke, "Noise reduction: Finding the simplest dynamical system consistent with the data," *Phys. D*, vol. 41, p. 183, 1990.
- [21] R. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modeling," Los Alamos Nat. Lab., Los Alamos, NM, Tech. Rep. LA-UR87-2662, 1987.
- [22] S. P. Luttrell, "Self-organization: A derivation from first principles of a class of learning algorithms," in *Proc. IEEE Conf. Neural Networks*, Washington, DC, 1989, p. 495.
- [23] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-Gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks*, vol. 4, no. 4, July 1993.
- [24] M. Motter, "Control of the NASA transonic wind tunnel with the self-organizing feature map," Ph.D. dissertation, Univ. Florida, Gainesville, Dec. 1997.
- [25] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz, "Adaptation and learning using multiple models, switching, and tuning," in *IEEE Control Syst. Mag.*, vol. 15, pp. 37–51, Mar. 1995.
- [26] ———, "Adaptive control using multiple models," in *IEEE Trans. Automat. Contr.*, vol. 42, pp. 171–187, Feb. 1997.
- [27] K. S. Narendra and S. Mukhopadhyay, "Adaptive control using neural networks and approximate models," in *IEEE Trans. Neural Networks*, vol. 8, pp. 475–485, May 1997.
- [28] H. Najai and V. Cherkassky, "Adaptive knot placement for non-parametric regression," in *Proc. Neural Information Processing Systems*, vol. NIPS-6, 1994, pp. 247–254.
- [29] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Phys. Rev. Lett.*, vol. 64, no. 11, pp. 1196–1199, 1990.
- [30] S. Haykin and X. B. Li, "Detection of signals in chaos," *Proc. IEEE*, vol. 83, no. 1, pp. 94–122, Jan. 1995.
- [31] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Phys. Rev. Lett.*, vol. 45, p. 712, 1980.
- [32] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.
- [33] M. B. Priestley, "State-dependent models: A general approach to nonlinear time series analysis," *J. Time Ser. Anal.*, vol. 1, p. 47, 1980.
- [34] J. C. Principe and J.-M. Kuo, "Dynamic modeling of chaotic time series with neural networks," in *Proc. Neural Information Processing Systems*, vol. NIPS 7, 1995, pp. 311–318.
- [35] J. C. Principe and L. Wang, "Non-linear time series modeling with self-organizing feature maps," in *Proc. IEEE Workshop Neural Networks for Signal Processing*, 1995, pp. 11–20.
- [36] J. C. Principe, L. Wang, and J.-M. Kuo, "Chaotic time series modeling with neural networks," in *Signal Analysis and Prediction*, Birkhauser, submitted for publication.
- [37] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*. Singapore: World Scientific, 1989.
- [38] H. Ritter, "Learning with the self-organizing map," in *Artificial Neural Networks*, vol. 1, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Amsterdam, The Netherlands: North Holland, 1991, pp. 379–384.
- [39] H. Ritter, "Parametrized self-organizing maps for vision tasks," in *Proc. ICANN'94*, Italy, pp. 803–810.
- [40] M. Sano and Y. Sawada, "Measurement of the Lyapunov spectrum from a chaotic time series," *Phys. Rev. Lett.*, vol. 55, no. 10, p. 1082, 1985.
- [41] L. L. Scharf, *Statistical Signal Processing—Detection, Estimation, and Time Series Analysis*. Reading, MA: Addison-Wesley, 1989.
- [42] T. Shinbrot, E. Ott, C. Grebogi, and J. A. Yorke, "Using chaos to direct trajectories to targets," *Phys. Rev. Lett.*, vol. 65, p.

- 3250, 1990.
- [43] J. J. Sidorowich, "Modeling of chaotic time series for prediction, interpolation, and smoothing," in *Proc. IEEE ICASSP*, vol. 4, 1992, p. 121.
 - [44] A. C. Singer, G. Wornell, and A. Oppenheim, "Codebook prediction: A nonlinear signal modeling paradigm," in *Proc. IEEE ICASSP*, vol. 5, 1992, p. 325.
 - [45] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence* (Springer Lecture Notes in Mathematics), vol. 898, D. A. Rand and L.-S. Young, Eds. New York: Springer-Verlag, 1980, pp. 365–381.
 - [46] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction, and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
 - [47] H. Tong, *Non-Linear Time Series Analysis: A Dynamical Systems Approach*. Oxford, UK: Oxford, 1990.
 - [48] J. Walter, H. Ritter, K. Schulten, "Non-linear prediction with self-organization maps," in *Proc. IJCNN*, 1990, vol. I-589, pp. 589–594.
 - [49] J. Walter and H. Ritter, "Local PSOM's and Chebyshev PSOM's improving the PSOM," in *Proc. Int. Conf. Artificial Neural Networks (ICANN'95)*, vol. 1, pp. 95–102.
 - [50] ———, "Rapid learning with parametrized self-organizing maps," *NeuroComputing*, vol. 12, pp. 131–153, 1996.
 - [51] L. Wang, "Local linear dynamic modeling with self organizing feature maps," Ph.D. dissertation, Univ. Florida, Gainesville, 1996.
 - [52] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA: Addison-Wesley, 1984.
 - [53] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
 - [54] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D*, vol. 16, p. 285, 1985.
 - [55] J. Vesanto J. "Using the SOM and local linear models in time series prediction," *Proc. Workshop Self-Organizing Maps WSOM'97*, Helsinki, Finland, pp. 142–147.



Jose C. Principe (Senior Member, IEEE) received the B.S. degree from University of Porto, Portugal, in 1973 and the M.Sc. and Ph.D. degrees from University of Florida, Gainesville, in 1974 and 1979, respectively.

He was a Professor at University of Aveiro, Portugal, from 1980 to 1987. He is Presently a Professor of Electrical and Computer Engineering at the University of Florida, Gainesville, where he teaches courses in signal processing and artificial neural networks (ANN's). He is

the Founder and Director of the University of Florida Computational NeuroEngineering Laboratory (CNEL). His primary area of interest is the processing of nonstationary, non-Gaussian signals and nonlinear models with dynamic neural networks. The CNEL Lab proposed a new, biological plausible, model for the classification of time varying patterns (the gamma model). The CNEL Lab has been studying the use of ANN's for dynamical modeling/controls and image/signal processing applications.

Dr. Principe is a member of SPIE, the IEEE Signal Processing Society, and the Board of Governors of the International Neural Networks Society. He is Secretary of the Neural Networks Technical Committee and a Member of the Advisory Board of the University of Florida Brain Institute.



Ludong Wang (Member, IEEE) received the B.S. degree in electronic engineering from Harbin Engineering University, Harbin, China, in 1986, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Florida, Gainesville, FL, in 1992 and 1996, respectively.

In 1996 he joined Hughes Network Systems (HNS), Germantown, MD, as a Member of Technical Staff in the Mobile Satellite Division. Since then he has been involved in the research and simulation of digital mobile satellite cellular systems. His recent two designs for geomobile systems are being filed for patents. He is also a co-investigator of two other filed patents. His main interests include neural networks, statistical estimation, adaptive signal processing, and synchronization in digital transmission systems.



Mark A. Motter (Senior Member, IEEE) was born in Columbia, PA, on September 3, 1955. He received the B.S.E.E. (magna cum laude) and M.S.E.E. degrees in 1983 and 1985, respectively, from Old Dominion University, Norfolk, VA. He received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in 1998.

He served in the United States Navy from 1973 to 1979 and was honorably discharged at the rank of Electronics Technician First Class.

Since 1985 he has been employed at NASA Langley Research Center, where he has been primarily involved in the modeling and control of wind tunnels and associated experimental equipment. He is a registered Professional Engineer in the state of Virginia. His current research interests are the assessment of mental engagement from physiological signals, adaptive-subspace self-organizing maps, and modeling and visualization of aircraft automation state and the airspace system.

Dr. Motter is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi.