Chapter 4

Adaptive local linear modelling and control of nonlinear dynamical systems

Deniz Erdogmus, Jeongho Cho, Jing Lan, Mark Motter and Jose C. Principe

4.1 Introduction

Systems theory is a well-established and mature area of engineering research, where many strong general mathematical results are available. Especially the analysis of linear identification and control systems have been pursued by many researchers leading to a *complete* understanding of various mechanisms that are effective in the stability, controllability and observability of these. Due to the availability of such an extensive knowledge base about linear systems, modern industrial control applications are still typically designed utilising the results from linear control systems theory. Nevertheless, academic research has been concentrating around problems involving the stability, identification and control of nonlinear dynamical systems in the last few decades. These efforts have now also matured into a broad theory of nonlinear systems, their identification and control. Initial efforts in this area pursued parametric approaches, inspired by the established linear systems theory, where the system dynamical equations are generally assumed to be known from physical principles, possibly with some uncertainty in the values of certain parameters. In this framework, the system identification and system control problems are decoupled, therefore can be solved sequentially. More recently, adaptive system identification and control methodologies have also been investigated, once again leading to a very good understanding of the adaptation in linear systems and a satisfactorily general insight to adaptation in nonlinear control systems. The latter problem, however, is implicitly extremely difficult to tackle and although nice mathematical results are obtained, practicality of these nonlinear techniques is yet difficult to achieve.

Control theory deals with the problem of manipulating the behavior of dynamical systems to satisfy certain desired outputs from the system. Classically, as mentioned above, the design procedure will follow the system identification and controller selection stages in the parametric approach to system modeling and control. In the case of traditional identification based on models derived from physical principles, the data are used to estimate the *unknown* parameters [1,2], whereas modern approaches stemming from the advances in neural network theory introduce *black-box* function approximation schemes in parts of the models [3-7]. The neural network modeling capabilities may be further enhanced using multiple such sub-models in the context of switching between adaptive models, as we will present in more detail in this chapter, to obtain closed-loop control systems that enhance transient behavior and cope better with modeling uncertainties and sudden model changes [8,9]. Following the system identification stage, depending on the modeling approach taken, the controller is designed typically using classical techniques based on linear system theory, such as gain scheduling [10], switching between multiple fixed or adaptive controllers [11,12], as well as classical or neural-network-based nonlinear techniques [13-15].

A large class of real-world systems can be reasonably approximated by nonlinear, time-invariant mathematical models. Therefore, our discussions here will focus on this class of systems, although we will briefly describe how to extend the presented approaches to the more general nonlinear time-varying system scenarios. Note that, however, the latter is an extremely difficult problem to solve. Even the global modeling of time-invariant nonlinear systems and designing corresponding controllers is itself a daunting task, let alone dealing successfully with time-variability in cases except where the variations are slow so that available adaptation tools can cope with the task of tracking the changing models.

A principle that is adopted with great enthusiasm in the statistical function approximation literature is the *divide-and-conquer* approach that dictates solving complicated problems by breaking them up into smaller and easier pieces that can be managed by simpler topologies. The method presented here follows along the lines of this principle. Therefore, conceptually, this modeling technique can be regarded as a piece-wise modeling approach, where the pieces are then *patched* together to form an approximate but successful global model. Specifically, when each of the *model pieces* are selected to be linear, the resulting model is a piece-wise linear dynamical approximation to the globally nonlinear dynamical system. The advantages of such a partitioning approach are three-fold: system identification complexity is reduced significantly due to the scaling down of the optimisation problem from one large task to multiple small and simple tasks, the piece-wise model easily scales up to encompass more volume in the state-space of the dynamical system by the addition of new patches as data from previously *unseen* portions of the state-space is acquired and the design of a control system for the nonlinear system can be reduced to the design of multiple simple and local controllers among which switching or cooperation is possible to generate a single control command to the actual plant. Especially with the selection of local linear dynamical models, the global nonlinear controller design reduces to the much simpler problem of designing multiple linear controllers for linear systems, a problem for which there are many extremely strong tools available in the linear control systems literature [16-18].

In the local modeling approach, there are two possibilities for utilising the individual local models to generate a single global value: select one model at a given time (winner-take-all), or take a weighted combination of the models (mixture-of-experts). Both approaches will be discussed in detail in the following sections. In particular, the winner-take-all approach will make explicit use of the self-organising maps (SOM) [19] in order to select which model-controller pair to switch to at every time instant (as opposed to the output-tracking-error-based switching criterion proposed by Narendra and co-workers [8,11]) and the mixture-of-experts approach will utilise the finite Gaussian mixture models (GMM) for a statistical interpretation of the local model contributions through the components of the mixture density model. Although the output-error approach is also commonly utilised in switching expert systems, it requires adjusting switching criterion parameters in noisy situations or for different systems, whereas in the SOM-based switching modality, these considerations are automatically taken care of in the SOM-training phase through the statistical interpretation of the data by the self organisation algorithm. The trade-off in this is the requirement that the multi-dimensional state space of the system is sufficiently covered by the SOM, whereas the output error approach operates in the lower dimensional output space. In addition, the SOM can be trained to classify the current state of the system directly from an input vector that is representative of this state, rather than the indirect measure of output-error. Consequently, the model selection sequence obtained using a SOM is expected to be more in tune with the actual state space transition that the dynamical system experiences. The GMM-based models will also be partitioned based on the same representative state vector as the SOM, consisting of past values of the plant's input and output, leading to the questions of validity and accuracy of such state representations.

It is well known that linear dynamical systems expressed by an observable state-space equation set can be equivalently described by autoregressive moving average (ARMA) difference equations (or differential equations in continuous time), which are essentially recursive expressions for the current output of the system in terms of its past inputs and outputs. The existence of such input-output recursive representations for nonlinear systems is also dependent on the extended definitions of observability for nonlinear systems. Results demonstrate that a wide class of nonlinear systems, called generically observable systems, also possess such nonlinear ARMA (NARMA) models that are valid at least locally and sometimes even globally, in the state space [2,20-23]. In summary, we can conclude that the behavior of nonlinear dynamical systems can at least locally be described well by NARMA equations, which is of crucial importance in the case of state-space reconstruction for dynamical systems where the internal state variables are not accessible. In such cases, the time-delay embedding method [24] has to be used in order to create local NARMA or ARMA models that are representative of the system dynamics.

4.2 Motivation for local linear modeling and time-delay embedding

Consider, without loss of generality, a single-input single output (SISO) nonlinear time-invariant dynamical system with state vector $\mathbf{x} \in \Re^n$, input $u \in \Re$ and output $y \in \Re$ with the following set of state equations and output mapping:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, u_k) \\ y_k &= h(\mathbf{x}_k) \end{aligned} \tag{4.1}$$

Notice that the consecutive outputs are (following the reasoning in [3] and with $\circ$ denoting composite functions):

$$\begin{aligned}
y_k &= h(\mathbf{x}_k) = \phi_1(\mathbf{x}_k) \\
y_{k+1} &= h \circ f(\mathbf{x}_k, u_k) = \phi_2(\mathbf{x}_k, u_k) \\
&\vdots \\
y_{k+n-1} &= h \circ f \circ \cdots \circ f(\mathbf{x}_k, u_k) = \phi_n(\mathbf{x}_k, u_k, u_{k+1}, ..., u_{k+n-2})
\end{aligned} \tag{4.2}$$

Defining $\mathbf{y}_k^n = [y_k \quad \cdots \quad y_{k+n-1}]^T$ and $\mathbf{u}_k^{n-1} = [u_k \quad \cdots \quad u_{k+n-2}]^T$, (4.2) can be collected in a vector-valued function form as $\mathbf{y}_k^n = \mathbf{\Phi}(\mathbf{x}_k, \mathbf{u}_k^{n-1})$.

Implicit Function Theorem [24]. Let **f** be a C^1 mapping of an open set $E \subset \Re^{n+m}$ into $\Re^n$ such that **f**(**a**,**b**)=**0** for some point (**a**,**b**) in *E*. Let $\mathbf{f_x}(\mathbf{x},\mathbf{y})$ denote the Jacobian of **f** with respect to **x** at the point (**x**,**y**) in *E*. If $\mathbf{f_x}(\mathbf{a},\mathbf{b})$ is invertible, then there exists an open set $U \subset \Re^{n+m}$ and $W \subset \Re^m$ with $(\mathbf{a},\mathbf{b}) \in U$ and $\mathbf{b} \in W$ such that to every $\mathbf{y} \in W$ there corresponds a unique **x** satisfying **f**(**x**,**y**)=**0**, $(\mathbf{x},\mathbf{y}) \in U$. If this **x** is defined to be **g**(**y**), then $\mathbf{g} \in C^1$ and is a mapping of *W* into $\Re^n$, **g**(**b**)=**a** and **f**(**g**(**y**),**y**)=0 for all $\mathbf{y} \in W$.

The Implicit Function Theorem basically states that the condition of local invertibility for a nonlinear function is that its Jacobian is locally nonsingular. Employing this theorem on the vector-valued function representation of (4.2), we conclude that if the Jacobian $\partial\mathbf{\Phi}/\partial\mathbf{x}$ is nonsingular at a stationary point in the state space of the unforced system, then $\mathbf{x}_k$ can be expressed locally in terms of $\mathbf{y}_k^n$ and $\mathbf{u}_k^{n-1}$. However, since by definition $\mathbf{x}_{k+n}$ depends on the inputs $u_k,\dots,u_{k+n-1}$ and the initial state $\mathbf{x}_k$, there exists a unique local nonlinear input-output mapping of the form

$$y_{k+1} = F(y_k, y_{k-1}, ..., y_{k-n+1}, u_k, u_{k-1}, ..., u_{k-n+1}) \tag{4.3}$$

valid in an open set in the state space encompassing the stationary point of linearisation. The same conclusion result could also be obtained with the brute force method of linearising the nonlinear dynamics around a stationary point in the state space and defining a state transformation from the actual state vector to incremental changes in the states, such that the system is locally represented by an ARMA process with state-dependent coefficients, which essentially becomes a NARMA equation as in (4.3). Conversely, it is possible to express the local NARMA process of (4.3) by a set of switching local ARMA processes, where each linearisation is carried out at the current operating point. Effectively, at a given operating point $(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) = (y_k^*, \ldots y_{k-n+1}^*, u_k^*, \ldots u_{k-m+1}^*)$, the approximate ARMA process is

$$\begin{aligned}
\hat{y}_{k+1} &= F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) + \left[\nabla_{\mathbf{y}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) \;\; \nabla_{\mathbf{u}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*})\right] \cdot \begin{bmatrix} \hat{\mathbf{y}}_k^n - \mathbf{y}_k^{n*} \\ \hat{\mathbf{u}}_k^n - \mathbf{u}_k^{n*} \end{bmatrix} \\
&= \left[\nabla_{\mathbf{y}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) \;\; \nabla_{\mathbf{u}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*})\right] \cdot \begin{bmatrix} \hat{\mathbf{y}}_k^n \\ \hat{\mathbf{u}}_k^n \end{bmatrix} \\
&\quad + \left(F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) - \left[\nabla_{\mathbf{y}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*}) \;\; \nabla_{\mathbf{u}} F(\mathbf{y}_k^{n*}, \mathbf{u}_k^{n*})\right] \cdot \begin{bmatrix} \mathbf{y}_k^{n*} \\ \mathbf{u}_k^{n*} \end{bmatrix} \right)
\end{aligned} \tag{4.4}$$

where the model output is a linear combination of the reconstructed state variables plus a bias term. For smoothly varying nonlinear dynamical systems, this ARMA model can further be accurately approximated by a purely linear combination of the reconstructed state variables as $\hat{y}_{k+1} = \mathbf{a}^T \hat{\mathbf{y}}_k^n + \mathbf{b}^T \hat{\mathbf{u}}_k^n$, where the bias term is implicitly embedded in the local model coefficient vectors **a** and **b** through the least squares type consideration of the approximation error and the mean state vector value in the neighborhood of approximation. This latter approximation is necessary when it is desired to design a local linear linear controller for the local linear ARMA model. The elimination of the bias term makes the local model truly linear in terms of its inputs and outputs, not just linear in its coefficients.

In this piece-wise linear approximation of the original nonlinear system, the coefficients of the locally effective ARMA model are determined by the current state of the nonlinear system, which is expressed in terms of the past values of the system input and output. This is an important observation, since in general, if the mathematical model of the plant is not known, its *physically meaningful* internal state variables are not accessible either. Under such conditions, the past values of the input and the output signals can be utilised to generate a representative state vector to identify the local behavior of the system.

In chaos theory and nonlinear time-series analysis, this method of reconstructing a state vector is referred to as *time-delay embedding* and there are strong theoretical results that demonstrate the mathematical validity of this approach for the case of autonomous nonlinear systems [22,23]. In particular, Takens' embedding theorem states, in plain words, that there exists an invertible (i.e. a one-to-one and onto) between the original state dynamics and the reconstructed state dynamics provided that the embedding dimension (the number of lags in the reconstructed state vector) is sufficiently large (specifically greater than two times the original state dimension). A similar result was also demonstrated by Aeyels that stated almost any autonomous system of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}),\ y = h(\mathbf{x})$ is generically observable if $2n$+1 samples of the output is taken in a manner similar to (4.2) [20].

These theoretical results on the local NARMA representations combined with the observability of nonlinear systems and the utility of time-delay embedding reconstructions of the state vector allow the construction of a piece-wise linear dynamical model approximation for a nonlinear system, which can be determined and optimised completely from input-output data collected from the original system. The literature is rich in multiple model approaches for nonlinear modeling [1,25], where the general consensus is that local modeling typically outperforms global modeling with a single highly complicated neural network in input-output modeling scenarios [2,26-30], despite the intrinsic simplicity and the input-output delay embedding approach has been adopted commonly based on results from nonlinear time-series analysis, such as Takens' theorem and its extensions [28]. A question of practical importance in this *black-box* input-output modeling approach is how to choose the number of embedding lags for both the input and the output. The next section deals with this question.

The motivation presented above mainly dealt with noise-free deterministic dynamical systems, whereas in practice, the available data is certainly noisy. From a mathematical perspective the suitability of the local modeling approach is justified by the above discussion. The practical aspects when noisy data is utilised is going to be investigated in the following sections whenever necessary.

4.3 Selecting the embedding dimension

If the number of physical dynamical states of the actual nonlinear system is known *a priori*, one can select the length of the embedding tap-delay lines for the input and output in accordance with the theoretical results by Takens and Aeyels. For complete practicality of the proposed local linear modeling approach for unknown systems, however, a truly data-driven methodology for determining the embedding dimensions for the input and output signals is required.

The problem of determining accurate input-output models from training data generated by the system has been addressed by many researchers [1-3,7,9,30,31], where the selection of the number of lags for the input and output signals (which is essentially a question of model-order selection) has always been an issue of practical importance. A useful solution to determine the embedding dimensions for input-output models is outlined by He and Asada [32], where the model order is determined based on the *Lipschitz index* calculated using the training data and the corresponding optimal model outputs for various embedding dimensions.

1. Select candidate output and input embedding dimensions *n* and *m*.

2. From now on consider all past values of input and output as input variables $x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m}$. Let the model output be denoted by y. Denote the ith input vector sample by $\mathbf{x}_i$ and the output by y_i, $i=1,\ldots,N$.
3. For every pair of samples evaluate the Lipschitz quotient: $q_{ij}=|y_i-y_j|/||\mathbf{x}_i-\mathbf{x}_j||$, $i\neq j$, $i,j=1,\ldots,N$.
4. Let $q^{(n+m)}(k)$ denote the kth largest quotient q_{ij}.
5. Evaluate the Lipschitz index: $q^{(n+m)}=\left(\prod_{k=1}^{p}\sqrt{n}q^{(n+m)}(k)\right)^{1/p}$, where p is an integer in the range $[0.01N,0.02N]$.
6. Go to step 1 and evaluate the Lipschitz index for a different set of embedding dimensions. The appropriate values of embedding dimensions will be indicated by the convergence index of the decreasing Lipschitz index as the embedding dimensions are increased one by one.

According to the theory, the appropriate embedding dimension pair for the input and the output is indicated by the convergence of the index. In other words, the embedding dimension, where the index stops decreasing (significantly), is to be selected as the model order.

4.4 Determining the local linear models

The local linear modeling approach can be broken into two consecutive parts: clustering/quantising the reconstructed state vector $\mathbf{x}_k^r=[\mathbf{y}_k^{nT}\quad \mathbf{u}_k^{nT}]^T$ adaptively using a statistically sound approach and optimising the local linear models corresponding to each cluster of samples with least squares (or some other criterion) using data from only that cluster.[1] There

[1] Although we focus on local linear modeling in this chapter, the principles and methodologies outlined here can immediately be extended to local nonlinear modeling. In fact, we will briefly investigate this latter choice later.

are many possible techniques for tackling each of these individual problems available in the literature. For example, the first step (data clustering) can be achieved by using a standard clustering algorithm (such as k-means clustering [23]) or vector quantisation methods [33], numerous variants of self organising maps (referred to as SOM) [19], or probability density mixture models (specifically the Gaussian Mixture Models – GMM) [34]; the second step (model optimisation) can be achieved using the analytical least squares solution (also referred to as the Wiener solution) [35], the least-mean-squares (LMS) algorithm [36], the recursive least squares algorithm [35], or the Kalman filter [37] if the mean-squared-error (MSE) is the optimality criterion of choice.[2]

In this section, we will focus on two of the clustering methods listed above: SOM and GMM. Based on which approach is selected in modeling, the principle behind the local linear models will be either competitive (for the SOM) or cooperative (for the GMM). Later on, the controller designs will also be slightly different due to this difference in the nature of the two approaches. In any case, the local model representation regimes will be selected optimally according to the criteria that these clustering methods utilise marking the main difference of the proposed adaptive local linear modeling approach from the standard gain-scheduling-like traditional approaches where the operating points of these local linear models are typically selected to be the stationary points of the state dynamics. In the case of completely unknown dynamics, this option is out of the question any way. Therefore, the methods presented here can be successfully applied both to cases where the actual state vector is accessible and where it is not available (so that input-output modeling is required).

[2] Alternative optimality criteria include other moments [38] and entropy [39] of the modeling error.

4.4.1 Clustering the Reconstructed State Vector Using Self-Organising Maps

Off-Line Training Phase: Suppose that input-output training data pairs of the form $\{(u_1,y_1),\ldots,(u_N,y_N)\}$, where u is the input signal and y is the output signal, is available from a SISO system for system identification. Under the conditions stated earlier, a nonlinear time-invariant dynamical system can be approximated locally by a linear ARMA system of the form[3]

$$y_k = a_1 y_{k-1} + \ldots + a_n y_{k-n} + b_1 u_{k-1} + \ldots + b_m u_{k-m} + c = \mathbf{a}^T \mathbf{y}_{k-1}^{n} + \mathbf{b}^T \mathbf{u}_{k-1}^{m} + c \qquad (4.5)$$

The reconstructed state vector $\mathbf{x}_k^{n,m} = \left[y_{k-1} \;\cdots\; y_{k-n} \;\; u_{k-1} \;\cdots\; u_{k-m} \right]^T$ can be adaptively achieved using a SOM with any topology of choice (triangular or rectangular grids are possible). The SOM consists of an array of neurons with weight vectors $\mathbf{w}_i$ that are trained competitively on its input vectors $\mathbf{x}_k^{n,m}$.[4] In training, these input vector samples are presented to the SOM one at a time in multiple epochs and preferably in each epoch, the presentation order of the samples is randomly shuffled to prevent memorising and/or oscillatory learning behavior.

At every iteration, the winner neuron is selected as the one that minimises instantaneously the Mahalanobis distance $d(\mathbf{w},\mathbf{x}_k) = (\mathbf{w}-\mathbf{x}_k)^T \mathbf{\Sigma}^{-1}(\mathbf{w}-\mathbf{x}_k)$ between the weight vector and the current training sample. Then the winner weight and its topological neighbors are updated using the following stochastic incremental learning rules [19], where $\mathbf{w}_w$ and $\mathbf{w}_n$ denote the winner and neighbor neuron weights, respectively:

$$\begin{aligned} &\mathbf{w}_w(t+1) \leftarrow \mathbf{w}_w(t) + \eta(t)\mathbf{\Sigma}^{-1}(\mathbf{x}_k - \mathbf{w}_w(t)) \\ &\mathbf{w}_n(t+1) \leftarrow \mathbf{w}_n(t) + \eta(t)h(\|\mathbf{w}_n(t) - \mathbf{w}_w(t)\|, \sigma(t))\mathbf{\Sigma}^{-1}(\mathbf{x}_k - \mathbf{w}_n(t)) \end{aligned} \qquad (4.6)$$

[3] The bias term c is required for mathematical consistency, however, in practice it is optional and can be removed due to reasons discussed before.

[4] The superscript ^{n,m} will be dropped from now on whenever unnecessary.

In the Mahalonobis distance, the scaling matrix Σ can be selected as the input covariance to emphasize various directions in the updates in accordance with the data structure, or it can be set to identity. The neighborhood function $h(.,\sigma)$ is a monotonically decreasing function in its first argument and it is unity when evaluated at zero and zero when evaluated at infinity. This allows the neighboring neurons to be updated proportionally to their distance to the instantaneous winner. The neighborhood radius σ is slowly annealed as well as the learning rate η. The neighborhood radius is initially set such that most of the network is included in this region, but in time, as the neurons start specializing in their distinct regions of quantization, this radius decreases to small enough values to cover only the winner neuron effectively. Typically, the neighborhood function is selected as a one-sided Gaussian function with the standard deviation parameter controlling the neighborhood radius. Both the radius and the learning rate can be annealed linearly or exponentially in terms of iterations or epochs.

The trained SOM can be regarded as a vector quantizer with the special topology preserving property. In particular, the SOM quantizes its input space while preserving the topological structure of the manifold that the samples come from, resulting in strong neighborhood relationships between the neurons; neighboring input vectors are mapped to neighboring neurons, thus in the next step of local linear modeling, neighboring models will be structurally similar. The input space of the SOM, which is equivalently the reconstructed state space of the system under consideration, is partitioned into smaller non-overlapping sets that are typically illustrated by a Voronoi diagram. The input samples $\{(\mathbf{x}_{i1}, y_{i1}),...,(\mathbf{x}_{iN_i}, y_{iN_i})\}$, which are in the ith Voronoi region, are associated with the corresponding neuron with weight vector $\mathbf{w}_i$, where N_i is the number of training samples in this region. In this local modeling scheme, besides the SOM weight vectors, each neuron also has a vector of local linear model coefficients associated with it, denoted by $\mathbf{a}_i$ and $\mathbf{b}_i$ separately for the output and input portions of the reconstructed state vector respectively. These models can be optimised using the training data clustered to the ith Voronoi region and the MSE criterion. This results in the following least-squares optimal local linear model coefficients [35,36]:[5]

$$\mathbf{c}_i = \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i^{yy} & \mathbf{R}_i^{yu} \\ \mathbf{R}_i^{uy} & \mathbf{R}_i^{uu} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P}_i^{dy} \\ \mathbf{P}_i^{du} \end{bmatrix} = \mathbf{R}_i^{-1}\mathbf{P}_i \tag{4.7}$$

where the blocks of the input autocorrelation matrix are obtained from the training samples using

$$\begin{aligned} \mathbf{R}_i^{yy} &= \frac{1}{N_i}\sum_k \mathbf{y}_k^n \mathbf{y}_k^{nT} & \mathbf{R}_i^{uu} &= \frac{1}{N_i}\sum_k \mathbf{u}_k^n \mathbf{u}_k^{nT} \\ \mathbf{R}_i^{yu} &= \frac{1}{N_i}\sum_k \mathbf{y}_k^n \mathbf{u}_k^{nT} & \mathbf{R}_i^{uy} &= \frac{1}{N_i}\sum_k \mathbf{u}_k^n \mathbf{y}_k^{nT} \end{aligned} \tag{4.8}$$

and the input-desired crosscorrelation vector blocks are estimated from samples using

$$\mathbf{P}_i^{dy} = \frac{1}{N_i}\sum_k y_k \mathbf{y}_k^n \quad \mathbf{P}_i^{du} = \frac{1}{N_i}\sum_k y_k \mathbf{u}_k^n \tag{4.9}$$

Finally, the output of the ith local linear model with the optimised coefficients is given by $\hat{y}_k = [\mathbf{a}_i^T \quad \mathbf{b}_i^T]\mathbf{x}_k$. More generally, introducing the model output weighting term p_{ik}, the overall local linear model system output is expressed as a switching (weighted) combination of the M individual model outputs:

$$\hat{y}_k = \sum_{i=1}^{M} p_{ik}\mathbf{c}_i^T \mathbf{x}_k \tag{4.10}$$

For hard competition, the weighting coefficients p_{ik} take only the values 0 or 1 at every time instant k. The selection completely depends on the ith neuron winning for input vector $\mathbf{x}_k$. A simple modification that one can introduce to the SOM-based local linear models to make the overall model cooperative rather than competitive is to allow other weighting values for the

[5] If the bias term is included in the linear model, then the Wiener solution in (7) must be modified accordingly.

models. For example, a weighted average type combination based on the distances of the current sample to the neuron weights would have

$$p_{ik} = \frac{f(d(\mathbf{w}_i, \mathbf{x}_k), \sigma)}{\sum_{j=1}^{M} f\left(d(\mathbf{w}_j, \mathbf{x}_k), \sigma\right)} \tag{4.11}$$

where a monotonically increasing emphasis function $f(.,\sigma)$ is combined with the Mahalonobis distance $d(.,.)$. The choice of a linear emphasis function would exactly be weighted averaging based on Mahalonobis distances. For clarity, the overall SOM-based local modeling topology is illustrated in Figure 4.1.

On-Line Training Phase: In most cases, a batch-training phase as described above is beneficial for control system performance on the actual system. The on-line training procedure, although it could be employed immediately to the unknown system with random initialisation of all the weights and coefficients to be optimised, could require a large number of samples and/or time to become sufficiently accurate, while in the mean time the system operates under an improper controller system. Nevertheless, the on-line training algorithm presented here could be especially useful in fine-tuning the existing models or introducing additional local models to the archive whenever modeling performance of the existing system drops below acceptable levels. In addition, for identifying time-varying systems, the local models can be continuously adapted using on-line data. For only fine-tuning of the existing models, one only needs to continue updating the SOM weights as well as the local linear model coefficients on a sample-by-sample basis in real time. The SOM weights can be continued to be updated using the original update rules given in (4.6). The local linear model coefficients, however, must be updated by one of the many existing on-line linear model weight update rules from the literature. These on-line training rules for linear models include LMS and RLS (for recursive least squares) [35].[6]

When using LMS or RLS, only the coefficients of the linear model associated with the instantaneous winner neuron (whose weight vector is updated recently using the SOM learning algorithm) are updated. The LMS update rule for the coefficients of the winner model (assuming neuron *i* is the winner) is given by

$$\mathbf{c}_i \leftarrow \mathbf{c}_i + \mu(y_k - \mathbf{c}_i^T \mathbf{x}_k)\mathbf{x}_k \tag{4.12}$$

where μ is the LMS step size. Since LMS uses stochastic gradient updates for the model coefficients, it exhibits a misadjustment associated with the power of the inputs and the step size. However, its computational complexity is very low, suitable for fast real-time applications. On the other hand, RLS is a fixed-point algorithm that can track the analytical Wiener solution via sample-by sample updates. The drawback is its increased complexity compared to LMS. The coefficient updates for the winner model according to RLS is given by the following iterations

$$\begin{aligned} \mathbf{k}_i &\leftarrow \left(\lambda^{-1}\mathbf{R}_i\mathbf{x}_k\right) / \left(1 + \lambda^{-1}\mathbf{x}_k^T\mathbf{R}_i\mathbf{x}_k\right) \\ \mathbf{c}_i &\leftarrow \mathbf{c}_i + \mathbf{k}_i\left(y_k - \mathbf{c}_i^T\mathbf{x}_k\right) \\ \mathbf{R}_i &\leftarrow \lambda^{-1}\mathbf{R}_i - \lambda^{-1}\mathbf{k}_i\mathbf{x}_k^T\mathbf{R}_i \end{aligned} \tag{4.13}$$

where the input autocorrelation matrix and weight vectors are initialised to $\mathbf{R}_i=\delta^{-1}\mathbf{I}$, $\mathbf{c}_i=\mathbf{0}$, δ being a small positive value.

[6] These update rules can be extended to the updating of nonlinear model weights [40]. The extension of LMS is trivial. The RLS algorithm is, in principle an implementation of the Kalman filter considering the adaptive weights as states. Hence, extensions to nonlinear systems (such as neural networks) are achieved through the formulation of the learning problem as an extended Kalman filtering problem [41].

Besides MSE, alternative model optimisation criteria such as alternative lags of error correlation [40], higher order error moments [36], or error entropy [38], can be utilised. Similar on-line update algorithms can be derived for these alternative criteria.

In some situations, simply fine-tuning of existing local models might not be sufficient to meet performance requirements in a sustained manner. Especially if, in actual operation, situations that are not encompassed in the training data set are encountered then a new local model might need to be introduced to the system of models. This could be achieved by utilising a growing SOM (GSOM) [41]. The most suitable grid structure for the GSOM is triangular. The neuron weights are still updated using (4.6). Contrary to static a SOM, in the GSOM, once in a while (e.g., at the end of every epoch), a new neuron is inserted (generated) in the weight space to the midpoint of line segment connecting the neuron with the highest winning frequency and the neuron farthest to it. A similar neuron-*killing* criterion can be developed to eliminate infrequently activated neurons. This procedure is repeated until a balanced distribution of input samples per neuron is obtained in the Voronoi diagram. In the process of generating and killing neurons, the triangular topology of the SOM must be preserved, so the new neighborhood connections must be selected accordingly.

4.4.2 Clustering the Reconstructed State Vector Using Gaussian Mixture Models

Off-Line Training Phase: Suppose that input-output training data pairs of the form $\{(u_1,y_1),\ldots,(u_N,y_N)\}$, where u is the input signal and y is the output signal, is available from a SISO system for system identification. The linear models described by (4.5) are still valid locally. In contrast to the SOM clustering of the reconstructed state vector $\mathbf{x}_k^{n,m} = [y_{k-1} \ \cdots \ y_{k-n} \ u_{k-1} \ \cdots \ u_{k-m}]^T$, which trains the cluster centers (neurons) competitively, the Gaussian mixture model considers the possibility of multiple modes generating the same state. In particular, it is assumed that the probability distribution of the state vector is given by

$$p(\mathbf{x}_k) = \sum_{i=1}^{M} \alpha_i G(\mathbf{x}_k ; \boldsymbol{\mu}_i , \boldsymbol{\Sigma}_i) \tag{4.14}$$

where $G(\mathbf{x};\boldsymbol{\mu}, \Sigma)$ is a multivariate Gaussian density with mean $\boldsymbol{\mu}$ and covariance Σ. The coefficient α_i denotes the probability of occurrence of the ith mode in the GMM, which in turn reflects the probability of the corresponding local model being effective. Given the training data and once the state vectors are reconstructed using embedding, the maximum likelihood solution for the parameters α_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ can be determined using the expectation maximisation (EM) algorithm [34]. The EM algorithm can be outlined as follows:[7]

1. E-Step: Compute the expectation of the log-likelihood of the complete data conditioned by the observed samples assuming the most recent solution for the mixture parameters, which is given by $Q(\vartheta \mid \vartheta_t) = \sum_k E[\log p(\mathbf{v} \mid \vartheta) \mid \mathbf{x}_k , \vartheta_t]$, where the parameter vector is defined to include all means, covariances and weights in the mixture model: $\vartheta = [\alpha_1 \ \cdots \ \alpha_M \ \ \boldsymbol{\mu}_1 \ \cdots \ \boldsymbol{\mu}_M \ \ vec(\boldsymbol{\Sigma}_1) \ \cdots \ vec(\boldsymbol{\Sigma}_M)]^T$.
2. M-Step: Update parameter estimates to ϑ_{t+1}, which is the maximiser of $Q(\vartheta \mid \vartheta_t)$.

Similar to the SOM-based modeling, in GMM-based local linear models, each Gaussian mode has a vector of local linear model coefficients associated with it, denoted by $\mathbf{a}_i$ and $\mathbf{b}_i$, again, for output and input portions of the reconstructed state vector, respectively. The output of the ith local linear model is given by $\hat{y}_k = [\mathbf{a}_i^T \ \ \mathbf{b}_i^T]\mathbf{x}_k$. The overall model output is a weighted

[7] The EM algorithm is essentially a fixed-point update rule for the mixture density parameters to maximize the likelihood of the data.

combination of the *M* individual outputs as in (4.10), $\hat{y}_k = \sum_{i=1}^{M} p_{ik} \mathbf{c}_i^T \mathbf{x}_k$, where $p_{ik} = \alpha_i G(\mathbf{x}_k ; \boldsymbol{\mu}_i , \boldsymbol{\Sigma}_i)$. The linear model coefficients can be collectively optimised using a modified Wiener solution similar to that in (4.7). The modification involves the model activation probabilities, p_{ik} and is explicitly given as $\boldsymbol{\theta} = \mathbf{R}^{-1}\mathbf{P}$, where $\boldsymbol{\theta} = [\mathbf{a}_1^T \quad \mathbf{b}_1^T \quad \cdots \quad \mathbf{a}_M^T \quad \mathbf{b}_M^T]$. The input autocorrelation matrix and the input-output crosscorrelation vector are defined using the modified input vector $\mathbf{z}_k = [p_{1k}\mathbf{x}_k^T \quad \cdots \quad p_{Mk}\mathbf{x}_k^T]$.[8] Specifically

$$\mathbf{R} = \frac{1}{N}\sum_k \mathbf{z}_k \mathbf{z}_k^T \qquad \mathbf{P} = \frac{1}{N}\sum_k y_k \mathbf{z}_k \tag{4.15}$$

For completeness, the GMM-based local modeling topology, which is similar to the SOM-based topology in many aspects, is shown in Figure 4.2.

On-Line Training Phase: After the off-line training procedure described above, the GMM-based model can be put to practice, while small adjustments to the existing parameters and model coefficients could be carried out in operation on a sample by-sample basis, although this would be computationally extremely expensive. The EM algorithm could still be iterated by including one more sample to the probability density evaluations at every time instant. Alternatively, the EM algorithm could be replaced by a gradient-based maximum likelihood algorithm that can be operated in a stochastic manner (similar to LMS) to update the GMM parameters. Similarly, the linear model coefficients can be updated on-line using LMS or RLS with the modified input vectors $\mathbf{z}_k$ [35]. Particularly, the LMS update for the linear model coefficients is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu(y_k - \boldsymbol{\theta}^T \mathbf{z}_k)\mathbf{z}_k \tag{4.16}$$

[8] Notice that the GMM-based model output is equivalently expressed as $\hat{y}_k = \boldsymbol{\theta}^T \mathbf{z}_k$

and the corresponding RLS update is similar to (4.13), but the input and weight vectors are modified:

$$\begin{aligned}
&\mathbf{k} \leftarrow \left(\lambda^{-1}\mathbf{R}\mathbf{z}_k\right) / \left(1 + \lambda^{-1}\mathbf{z}_k^T \mathbf{R}\mathbf{z}_k\right) \\
&\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{k}\left(y_k - \boldsymbol{\theta}^T \mathbf{z}_k\right) \\
&\mathbf{R} \leftarrow \lambda^{-1}\mathbf{R} - \lambda^{-1}\mathbf{k}\mathbf{z}_k^T \mathbf{R}
\end{aligned} \tag{4.17}$$

Alternative model optimisation criteria such as alternative lags of error correlation [40], higher order error moments [36], or error entropy [38], can also be utilised in this case with appropriate modifications.

4.4.3 Extending the Methodology to Local Nonlinear Modeling

It was made clear that the general nonlinear system of (4.1) is, in general, approximated locally by a NARMA process and we went one step further in the approximation to replace the local NARMA approximation by piece-wise linear dynamics. One obvious modification would be to allow the local models to be nonlinear input-output dynamical recursive systems, such as time-delay neural networks (TDNN) [42]. The TDNN, being an extension of multilayer perceptrons (MLP) to time series processing, still possesses the universal approximation capabilities of MLPs, however, for the restricted class of dynamical systems with myopic memories (i.e., systems where a finite number of past values of the input affect the output, in a manner similar to the observability conditions discussed earlier in this chapter) [43,44]. A TDNN basically consists of a number of FIR filters in parallel whose outputs are modified by sigmoid nonlinearities and then linearly combined by the output layer. More generally,multiple layers of nonlinear FIR filter banks can be employed, but it is known that a sufficiently large single hidden layer TDNN has enough approximation capability. Training is typically performed via backpropagation of MSE [42].

Another feasible alternative that has a smaller approximation capability, but significantly simple to optimise is an Hammerstein dynamical system [45,46]. A Hammerstein structure consists of a static nonlinearity that transforms the input followed by a linear dynamical system. Once the input nonlinearities are set, the training of the linear dynamical portions is similar to the linear models discussed earlier (using the properly modified input autocorrelation matrix in the case of MSE optimality criterion).

Another possibility in nonlinear modeling is to use Volterra series approximation [47]. Volterra series expansion is an extension of Taylor series expansion to dynamical systems. It is based on multi-dimensional convolution integrals and the first order Volterra approximation is simply a linear convolutive system. Typically, Volterra series approximations are truncated at most the third order convolution and separability of the multidimensional impulse responses is assumed for simplicity. This, of course, limits the approximation capability of the model in addition to the fact that the least squares optimisation of the model coefficients is not necessarily simplified; local minima problems still exist.

Finally, as a direct consequence of Taylor series expansion, the local linear models can be extended to include higher order polynomial factors of delayed values of the input and the output. This is the Kolmogorov-Gabor polynomial modeling approach [48]. The number of coefficients to be optimised grows combinatorially with the order of the polynomial model, creating the main drawback of this approach.

4.5 Designing the local linear controllers

An added advantage of the proposed local linear modeling approach is it greatly simplifies the design of control systems for nonlinear plants. In general, this is a daunting task and typically practical solutions involve linearisation of the dynamics and then employing well-established controller design techniques from linear control systems theory. While designing globally stable nonlinear controllers with satisfactory performance at every point in the state space of the closed loop control system is extremely difficult and perhaps impossible to achieve especially in the case of unknown plant dynamics, the local linear modeling technique presented above, coupled with strong controller design techniques from linear control theory [18,49] and recent theoretical results on switching control systems [8,11,18], it becomes possible to achieve this goal through the use of this much simpler approach of local modeling. The topology of local linear controllers that naturally arise from the local linear modeling approach is illustrated in Figure 4.3.

4.5.1 Inverse Error Dynamics Controller Design

Once the optimal local linear models have been identified from the training input-output data available for system identification, one can use any standard linear controller design techniques to meet predefined regulation, stabilisation, or tracking performance goals. Possibilities include stabilisation with linear state feedback (the individual ARMA systems can be expressed in controllable canonical form to design their corresponding state-space controllers), regulation or tracking a time-varying desired output response signal by a PID controller or more generally inverse error dynamics controller. In this section, we will focus on the latter, inverse error dynamics controller scheme as it includes the PID controllers [50-53] and the exact tracking control [3,54], commonly utilised in practice.

The principle behind inverse error dynamics controller design is pole placement. Simply, it can be described as selecting a set of stable poles for the tracking error signal dynamics. If we denote the desired plant output at time k by d_k and the actual plant output by y_k, then the instantaneous tracking error is simply given by $e_k = d_k - y_k$.[9] The goal of this controller design technique is to guarantee that the error signal obeys the following dynamical equation:

$$e_{k+1} + \lambda_1 e_k + \lambda_2 e_{k-1} + \ldots + \lambda_l e_{k-l+1} = 0 \qquad (4.18)$$

The parameters $\boldsymbol{\lambda}=[\lambda_1,\ldots,\lambda_l]^T$ are selected such that the roots of the polynomial $1+\lambda_1 x+\ldots\lambda_l x^l$ are inside the unit circle. This guarantees the global stability of the closed loop control system provided that there is no noise, disturbance or modeling error. We will address how to handle these unwanted effects by designing the parameter vector appropriately later in this section. Of particular interest are inverse error dynamic equations of order 0 to 3. These are explicitly

$$\begin{array}{lll} \text{Order } 0: & e_{k+1} = 0 & \text{Exact tracking controller} \\ \text{Order } 1: & e_{k+1} + \lambda e_k = 0 & \text{First order linear decaying error dynamics} \\ \text{Order } 2: & e_{k+1} + \lambda_1 e_k + \lambda_2 e_{k-1} = 0 & \text{Second order linear decaying error dynamics} \\ \text{Order } 3: & e_{k+1} + \lambda_1 e_k + \lambda_2 e_{k-1} + \lambda_3 e_{k-2} = 0 & \text{PID controller} \end{array} \qquad (4.19)$$

The exact tracking controller simply solves for the error equation to determine the control input necessary for the next error value to be identically zero. This strategy, obviously, is not robust to modeling errors or external disturbances such as measurement noise. The first and second order error dynamic controllers are easy to design, since analytical expressions of their corresponding poles and the associated dynamical behaviors are well understood in linear system theory. Especially the second order controller allows the designer to choose both overshoot and settling time simultaneously. The third order error dynamics controller is effectively a PID controller, since PID controllers, when discretised, allow the designer to place the closed loop

[9] Although we are considering discrete-time controller design here, the idea is easily applied to continuous-time controller design as well. In addition, generalization to MIMO systems is also achieved simply by defining a vector-valued error signal, whose entries may or may not interact in the desired error dynamics equation through the use of non-diagonal or diagonal coefficient matrices.

system poles to any location on a three dimensional manifold in the *n*-dimensional space of the plant dynamics.

In the most general case, defined by (4.18), the control input is solved for as follows:

$$e_{k+1} = d_{k+1} - \hat{y}_{k+1} = -(\lambda_1 e_k + \lambda_2 e_{k-1} + \ldots + \lambda_l e_{k-l+1}) = -\boldsymbol{\lambda}^T \mathbf{e}_k^l \tag{4.20}$$

Recall that explicitly the model output is given by

$\hat{y}_{k+1} = \sum_{i=1}^{M} p_{ik} \mathbf{c}_i^T \mathbf{x}_k^{n,m} = \sum_{i=1}^{M} p_{ik} \mathbf{a}_i^T \mathbf{y}_k^n + \sum_{i=1}^{M} p_{ik} \mathbf{b}_i^T \mathbf{u}_k^m$. We are particularly interested in the terms involving u_k. Separating these terms from the others in the expression, the model output is

$\hat{y}_{k+1} = \left(\sum_{i=1}^{M} p_{ik} \mathbf{a}_i^T \mathbf{y}_k^n + \sum_{i=1}^{M} p_{ik} \tilde{\mathbf{b}}_i^T \mathbf{u}_{k-1}^{m-1}\right) + \left(\sum_{i=1}^{M} p_{ik} b_{1i}\right) u_k = v_k + \tilde{b}_1 u_k$, where all new variables are defined in accordance with the equalities here. Introducing this new expression in (4.20) and solving for the input u_k, we obtain

$$u_k = \frac{1}{\tilde{b}_1}\left(d_{k+1} + \boldsymbol{\lambda}^T \mathbf{e}_k^l - v_k\right) \tag{4.21}$$

Notice that we can define $u_{ik} = \left(d_{k+1} - \left(\mathbf{a}_i^T \mathbf{y}_k^n + \tilde{\mathbf{b}}_i^T \mathbf{u}_{k-1}^{m-1}\right) + \boldsymbol{\lambda}^T \mathbf{e}\right) / b_{1i}$ as the control input suggested by the ith model. With this definition, the control input of (4.21) can be expressed as

$$u_k = \frac{\sum_{i=1}^{M} p_{ik} b_{1i} u_{ik}}{\sum_{i=1}^{M} p_{ik} b_{1i}} = \sum_{i=1}^{M} \alpha_{ik} u_{ik} \tag{4.22}$$

where $\sum_{i=1}^{M} \alpha_{ik} = 1$, thus (4.21) is equivalent to a weighted average of the individual control input suggestions by the *M* local linear models, which motivates the parallel controller topology shown in Figure 4.3.

For various choices of the error dynamics order and pole locations, the control law will drive the tracking closed-loop system in various different ways, while the error signal will always satisfy the linear difference equation given in (4.18), which is selected by the designer. An alternative controller design technique for both local linear models and local nonlinear models (should this be the designer's choice) is sliding mode control [55], which is essentially a nonlinear extension of the linear inverse error dynamics controller design methodology presented here. Sliding mode control is also well understood in the nonlinear control literature and it is known to be more robust to modeling errors and external disturbances compared to its linear counterpart [56]. In the linear case, which is being considered in this chapter, while any stable choice of stable poles will work satisfactorily and as expected in no-error/disturbance scenarios, in practice, due to the piece-wise nature of the local models and sensor noise, these undesirable effects will always corrupt the calculated control input in (4.21). This can be countered by carefully selecting the error dynamic pole locations such that the parameter vector $\boldsymbol{\lambda}$ represents a discrete-time filter that eliminates most of the power that is contained in these disturbances. Clearly, this requires the *a priori* knowledge of the statistical characteristics of these disturbances. Specifically, one needs the power spectral density information so that the filter can be designed to suppress high-energy frequency bands.

4.5.2 Selecting the Error Dynamics Pole Locations to Suppress Disturbance Effects

Recall the tracking error dynamical equation in (4.18), $\hat{e}_{k+1} = d_{k+1} - \hat{y}_{k+1} = -\boldsymbol{\lambda}^T \hat{\mathbf{e}}_k^l$, obtained using the local linear models with noisy input-output measurements from the system according to the prediction equation $\hat{y}_{k+1} = \boldsymbol{\theta}^T \hat{\mathbf{z}}_k = y_{k+1} + n_k$, where $\hat{\mathbf{z}}_k$ is the vector of noisy input-output measurements, y_{k+1} is the true plant output corresponding to the planned control input and n_k is the overall error term that includes all external disturbances and modeling errors. With this notation, also including the measurement noise present in $-\boldsymbol{\lambda}^T \hat{\mathbf{e}}_k^l$ in n_k, the dynamical equation that the true tracking error obeys becomes stochastic: $e_{k+1} = -\boldsymbol{\lambda}^T \mathbf{e}_k^l + n_k$. In effect, this represents an all-poles filter from the noise term n_k to e_k, whose pole locations are exactly determined by $\boldsymbol{\lambda}$. Hence, besides the stability constraint, the poles should be located inside the unit circle such that the all-pole filter from noise to tracking error that has the transfer function $H(z) = 1/(1+\lambda_1 z^{-1} + \ldots + \lambda_l z^{-l})$.

This observation inspires an alternative linear error dynamics approach to controller design in the case of measurement noise, external disturbances and modeling errors. If the equation in (4.18) is properly modified, it might be possible to achieve arbitrary transfer functions $H(z)$ from noise to tracking error (i.e. transfer functions with zeros and poles simultaneously). To achieve this, the error dynamic equation must be extended to further error predictions into the future using the model. Specifically, if the following error dynamic equation is used

$$\eta_q \hat{e}_{k+q} + \ldots + \eta_1 \hat{e}_{k+1} + \lambda_1 e_k + \lambda_2 e_{k-1} + \ldots + \lambda_l e_{k-l+1} = 0 \tag{4.23}$$

then the noise and error terms, again collected in a single n_k, drive the following ARMA system from the disturbance to the tracking error:

$$\begin{bmatrix} \boldsymbol{\eta}^T & \boldsymbol{\lambda}^T \end{bmatrix} \mathbf{e}_{k+q}^{q+l} = \boldsymbol{\eta}^T \mathbf{n}_{k+q}^{q} \tag{4.24}$$

Effectively, this corresponds to the following transfer function from n_k to e_k:

$$H(z) = \frac{\eta_1 + \eta_2 z^{-1} + \ldots + \eta_q z^{1-q}}{1 + \eta_1 z^{-1} + \ldots + \eta_q z^{1-q} + \lambda_1 z^{-q} + \ldots + \lambda_l z^{1-l-q}} \tag{4.25}$$

Consequently, the parameter vectors $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$ must be selected to place the zeros and poles of the transfer function in (4.25) to maximally filter out noise, using the spectral density information. The noise PSD can be estimated from training data, which is collected from the original system under noisy measurement conditions. These procedures are out of the scope of this chapter, therefore, we will not go into the details. However, spectral estimation is a mature and well-established research area [57].

4.6 Simulations

The local linear modeling and control technique that is presented above has been tested on a variety of nonlinear system identification and control problems including chaotic time-series prediction, synthetic nonlinear system identification, simplified missile lateral dynamics identification and control, NASA Langley transonic wind tunnel and NASA LoFlyte waverider aircraft. In this section, we will provide a compilation of simulation and experimental results obtained from the application of these principles to the listed problems.

4.6.1 Chaotic Time Series Prediction

In this example, we will demonstrate the use of the Lipschitz index for determining the model order for a SOM-based local linear modeling system using data generated by the Mackey-Glass attractor and the Lorenz attractor. The Mackey-Glass chaotic system is governed by the following differential equation [58]:

$$\dot{x}(t) = bx(t) + \frac{\alpha x(t-\gamma)}{1 + x^{\rho}(t-\gamma)} \tag{4.26}$$

The delay γ controls the depth of the underlying chaotic motion. The parameters are set to α=0.2, β=-0.1 and ρ=10. The system in (4.26) is iterated using the Runge-Kutta-4 integration method with a time step of 1 and the signal is downsampled by a factor of 6. An embedding dimension of 6 and an embedding delay of 2 were determined based on the Lipschitz index and auto-mutual-information[10] of the signal, shown in Figure 4.4. A SOM trained on 1000 samples of input vector

[10] Information dimension is a standard method for determining the delay amount in the embedding. The first zero or the minimum of the auto-mutual-information [60] is used as the embedding delay.

generated as described exhibit the Mackey-Glass attractor's topology, as shown in Figure 4.5. In addition, evaluation of the single step prediction performance using local linear models (LLM) attached to the neurons of the SOM demonstrates the high accuracy: a prediction signal-to-error ratios (SER) of 33.80dB, 33.39dB and 31.05dB are obtained using a 15x15 rectangular SOM-LLM, a 40-neuron single layer TDNN and a 100-basis Radial Basis Function (RBF) network, respectively.

The second chaotic system that will be considered here is the Lorenz attractor. The dynamics of this system are governed by the following system of differential equations [58]:

$$\begin{aligned} \dot{x} &= \sigma(y-x) \\ \dot{y} &= -y - xz + Rx \\ \dot{z} &= xy - bz \end{aligned} \tag{4.27}$$

where we selected R=28, σ=10 and b=8/3 to obtain chaotic dynamics and the first state as the output. Using Runge-Kutta-4 integration with time step 0.01, 9000 samples were generated for analysis. Using the Lipschitz index and mutual information analysis (as shown in Figure 4.6) the embedding dimension is determined to be 3 and the embedding delay is 2. The first 8000 samples from the output time-series are used to create the reconstructed state samples, according to which the SOM and the local linear models are trained for single step prediction. As seen in Figure 4.7, the SOM represents the Lorenz attractor accurately. In addition, the single step prediction performance, again evaluated in terms of SER is found to be 53.74dB, 49.23dB and 50.46dB for the 20x20 SOM-LLM, 100-neuron TDNN and 150-basis RBF network, respectively, on the remaining 1000-sample test set.

These examples illustrate the modeling capability of the LLM technique on chaotic signal prediction, which is a benchmark problem in system identification. In addition, the validity of the Lipschitz index for selecting the embedding dimension (i.e., model order) is demonstrated by the successful reconstruction of the two chaotic attractors with the dimensions indicated by this index.

4.6.2 Synthetic Nonlinear System Identification

The first nonlinear system that is considered here is defined by the following state dynamics and output mapping [13]. Since the input is cubed, a finite number of linear models will not have good global approximation performance. The approximation will only be valid in the range of inputs available in the training data.

$$\begin{aligned} x_{k+1} &= \frac{x_k}{1+x_k^2} + u_k^3 \\ y_k &= x_k \end{aligned} \tag{4.28}$$

Input-output training data is generated using white input signal distributed uniformly in [-2,2]. Since the input is independent from itself at any delay, the embedding delay is taken to be unity. The embedding dimensions for the input and the output are both 2. Consequently, a SOM is trained using the input vector $[y_k, y_{k-1}, u_k, u_{k-1}]^T$ and corresponding local linear models are optimised using least squares. For a comparison, a local quadratic polynomial Hammerstein modeling (LPM) approach is also implemented [46]. The reconstructed state vector for this model is $[y_k, u_k, u_k^2]^T$. The size of the rectangular grid structure is selected based on the generalisation MSE of both models on a validation set. A size of 15x15 is selected for both SOMs. The system identification capabilities of the SOM-LLM, the SOM-LPM, a 150-basis RBF network and a 13-neuron Focused gamma neural network (FGNN) [59] are compared.[11] Their performances on the test set are respectively 12.73dB, 47.95dB, 43.56dB and 44.59dB.

[11] The FGNN is a generalization of the TDNN where the input tap-delay line is replaced by a tap-gamma line.

Clearly, the cubic input term is not sufficiently approximated by the selected number of linear models and increasing the size of the network results in poor generalisation. As expected, the quadratic polynomial is certainly much more effective in approximating the cubic nonlinearity locally with the same number of local models.

The second nonlinear system that is considered is an FIR filter followed by a static nonlinearity defined by:

$$y_{k+1} = \arctan(u_k - 0.5u_{k-1}) \quad (4.29)$$

The system is excited by white input uniform on [-2,2] and system identification was carried out using the SOM-LPM topology explained above. The system identification testing results yielded SER values of 47.96dB, 49.99dB and 52.30dB for 8-neuron FGNN, 41-basis RBF and 15x15 SOM-LPM, respectively.

4.6.3 Simplified Lateral Missile Dynamics Identification and Control

Under the assumptions of constant mass, zero roll and pitch angles and angular speed, the yaw dynamics of a missile can be expressed by [5]:

$$\begin{aligned} \dot{x}_1 &= x_2 - 0.1\cos(x_1)(5x_1 - 4x_1^3 + x_1^5) - 0.5\cos(x_1)u \\ \dot{x}_2 &= -65x_1 + 50x_1^3 - 15x_1^5 - x_2 100u \\ y &= x_1 \end{aligned} \qquad (4.30)$$

The input is the rudder deflection and it is limited by ±0.5rad. Two local linear models, SOM-LLM and GMM-LMM were identified using 6000 samples if input-output data, where the system is excited by white input uniform on [-0.5,0.5]. A discretisation time step of 0.05s was used (making the training set correspond to 300s of flight time). The embedding delays for both the input and the output were found to be 2 using the Lipschitz index. The SOM consisted of a 15x15 rectangular grid, while the GMM had 5 Gaussian modes. Both models were tested on

1000 independently generated test samples generated using a random input sequence. The SOM-LLM and GMM-LLM performances were 31.7dB and 31.0dB in terms of SER, respectively. In other modeling problems, it was observed that the GMM-LLM approach required a smaller number of linear models, in general, compared to the SOM-LLM approach.

In addition, to system identification, local linear PID controllers were designed for both models placing the poles of the closed-loop response at 0, 0, 0.05+*i*0.3 and 0.05-*i*0.3. For each linear model, the PID coefficients are set to these locations by adjusting their corresponding λ vectors. The closed-loop regulation and tracking performances are tested by forcing the system output to track step changes and smooth sinusoidal changes in the desired output. The tracking performances of the SOM-LLM and GMM-LLM based PID controllers are shown in Figure 4.8. As a comparison, the tracking performance of a TDNN-based global adaptive model-controller pair is also presented in Figure 4.9. Clearly, the local PID controllers outperform the adaptive globally nonlinear technique in terms of both overshoot and convergence time.

4.6.4 NASA LoFlyte Waverider Aircraft Identification and Control

The NASA LoFlyte is an unmanned aerial vehicle (UAV) designed by Accurate Automation Corporation (AAC) and an illustrative picture is shown in Figure 4.10. The LoFlyte program is an active test program at the Air Force Flight Test Center of the Edwards Air Force Base with the objective of developing the technologies necessary to design, fabricate and flight test a Mach 5 waverider aircraft [60,61]. The LoFlyte UAV is also used to understand the low speed characteristics of a hypersonic shape and to demonstrate several innovative flight control technologies. The task of CNEL is to develop modeling and control strategies for LoFlyte based solely on input-output data.

According to classical aerodynamics, the flight dynamics of any rigid body are determined by movements along or around three body axes: roll, pitch (longitudinal motion) and yaw (lateral motion). The elevator δ_e is mainly responsible for controlling the longitudinal motion state variables (pitch angle, θ and pitch rate, q), the rudder δ_r primarily controls the lateral motion state variables (yaw angle, ψ and yaw rate r), the aileron δ_a mainly controls the roll motion state variables (roll angle, ϕ and roll rate, p). Finally, the throttle δ_t largely controls the aircraft's longitudinal speed, while in some aircraft, deflectable thrust vectors might allow yaw and roll contributions from the engine power. Typically, under certain symmetry assumptions for the aircraft body, the state dynamics of the rigid-body aircraft are represented around its center of gravity as follows (see [62] or any standard text book on flight dynamics and control):

$$\begin{aligned}
\dot{u} &= -(wq - vr) - g\sin\theta + F_x / m \\
\dot{v} &= -(ur - wp) + g\cos\theta\sin\phi + F_y / m \\
\dot{w} &= -(vp - uq) + g\cos\theta\cos\phi + F_z / m \\
\dot{p} &= \left((I_{yy} - I_{zz})qr + I_{xz}(qr - pq) + L\right)/ I_{xx} \\
\dot{q} &= \left((I_{zz} - I_{xx})rp + I_{xz}(r^2 - p^2) + M\right)/ I_{yy} \\
\dot{r} &= \left((I_{xx} - I_{yy})pq + I_{xz}(pq - qr) + N\right)/ I_{zz} \\
\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\
\dot{\theta} &= q\cos\phi - r\sin\theta \\
\dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta
\end{aligned} \tag{4.31}$$

In (4.31), u, v, w are the speed components of the aircraft along its body x, y, z axes, respectively. Similarly, p, q, r are angular speeds around these axes and ϕ, θ, ψ are the Euler angles that define the rotation matrix between the body coordinate frame and the inertial coordinate frame (e.g., the north-east-down system in the case of short-duration, short distance flights within the atmosphere, under the flat-earth assumption). The gravity g is along the *down* direction of the inertial frame. The engine power and aerodynamic effects generate the forces F_x, F_y and F_z as well as the moments L, M, N. The coefficients m, I_{xx}, I_{yy}, I_{zz} and I_{xz} are the aircraft mass and moments of inertia determined by its geometry.

The LoFlyte aircraft is simulated using a software by AAC and is assumed to be the true plant. It is assumed that the throttle is constant and state variables *p*, *q*, *r*, *u*, *v*, *w* are available for external measurement. The goal of the identification and control problem is to determine local linear models from the three inputs (aileron, elevator and rudder) to these six state variables (outputs) and to control them in order to track a desired trajectory of flight.

Input-output training data is generated using the ACC flight simulator by manually flying the model aircraft (with a joystick) to imitate a test flight. The embedding dimensions for both input and the output are selected to be 3. SOM-based local linear models are trained from all three inputs to all six outputs, quantising the reconstructed state space formed by the vector of delayed past output values. In these local models, in order to reduce model complexity, the coupling between the state variables is ignored, while all three inputs are still assumed to affect all six outputs. In essence, in matrix-vector form, the models are of the form

$$\hat{\mathbf{y}}_{k+1} = \sum_{i=0}^{3} \mathbf{A}_i^T \mathbf{y}_{k-i} + \sum_{i=0}^{3} \mathbf{B}_i^T \mathbf{u}_{k-i} \tag{4.32}$$

where $\hat{\mathbf{y}}$ denotes the vector of outputs (the six measured states) and **u** denotes the vector of inputs (the three deflector angles). Since the output coupling is ignored, $\mathbf{A}_i$ are diagonal matrices, while $\mathbf{B}_i$ are full matrices that allow coupling effects from all inputs to all outputs.

Using 5000 training samples and a 10x10 rectangular SOM grid, whose size was determined according to the validation set performance (see Figure 4.11), the system identification performance was found to be 28.12dB, 20.21dB, 28.56dB, 74.73dB, 27.83dB and 37.98dB for the six outputs, respectively, in terms of SER on a 1000-sample test set. The same data was used in training a global 12-neuron TDNN model, which achieved 27.63dB, 19.57dB, 27.34dB, 47.56dB, 27.24dB and 36.57dB, respectively for the outputs in the test set.

An order-0 inverse error dynamic controller was designed for *p*, *q* and *r* according to (4.19) and (4.21). For comparison, an adaptive TDNN inverse controller based on the TDNN was also designed for the aircraft. The performances of the controllers are tested in set-point regulation and arbitrary output tracking. The tracking results of both controller systems are presented in Figure 4.12 and Figure 4.13 [54]. Clearly the local linear controllers exhibit better overshoot and convergence speed characteristics compared to the nonlinear counterpart.

4.6.5 NASA Langley Transonic Wind Tunnel Identification and Control

In this final application example, the performance of the proposed local linear modeling and control approach in the identification and control of the NASA Langley 16-Foot Transonic Tunnel will be presented [12]. This wind tunnel, whose picture is shown in Figure 4.14 to provide some perspective, is driven by a simple bang-zero-bang type control input with three possible values: -1, 0, +1. The plant output is the Mach number (in the range 0.20 to 1.30) achieved around the experimental aircraft model whose dynamics are being studied as shown in Figure 4.15. The possible input value sequences of length *p* were considered, resulting in a total of 3^p possible sequences. These sequences were partitioned to 9 sets, which were experimentally determined to meet performance needs with low computational requirements. Seven of these prototype input sequences were 50-samples long, while the remaining two were only 10-samples long [12]. For each of these input partitions, the tunnels Mach number responses were clustered using a 20-node linear SOM. Finally, each neuron of each SOM has a linear predictor of Mach number associated with it, as in the SOM-LLM framework, that evaluates the suitability of each input sequence by comparing the predicted Mach number with the desired value in the following *p* time steps (either 50 or 10 depending on the input sequence being evaluated). The control input that produces the best Mach number match to the desired is selected and employed.

The identified local linear models and associated controllers are tested on the actual tunnel and the performance is compared to that of an expert human operator and the existing controller scheme. Typically, acceptable performance is maintaining a Mach number regulation error within ±0.003 of the set point while completing an angle-of-attack (α) sweep in as small time as possible (due to power considerations).

In the first experiment, each controller is required to maintain 0.85 Mach within specifications for 15 minutes while an α-sweep is completed. Minimum control activity is a plus. The performance of the three controllers is shown in Figure 4.16.

The average Mach number of the existing controller, the expert operator and the SOM-LMM controller (denoted by PMMSC in the figure) are 0.8497, 0.8500 and 0.8497, respectively, with standard deviations 0.001527, 0.001358 and 0.001226. The amount of time these controllers were out of tolerance were 46.5s, 34.52s and 33.2s. The L_1 norm of the control inputs were 10.6, 12.33 and 6.33, respectively. Clearly, the local linear controllers outperformed both competitors in terms of meeting tolerance bounds with minimum controller activity.

As a second test, all controllers were required to track a step-wise changing Mach number set point, again with as small as possible control effort, in a 28-minute experiment. The Mach number tracking performances of the controllers are shown in Figure 4.17.

The existing controller was out of the tolerance bounds for 329s, the expert operator was out of tolerance for 310s and the SOM-LLM was out of bounds for 266s. Their respective L_1 control input norm values for the duration of this experiment were 424.2, 466.2 and 374.3, again indicating the superior performance of the proposed local linear control approach.

4.7 Conclusions

In this chapter, the problem of nonlinear system identification and control system design was addressed under the divide-and-conquer principle. This principle motivated the use of multiple local models for system identification in order to simplify the modeling task. Especially in the case of unknown dynamics, where only input-output data from the plant is available, the proposed method is able to approximate the nonlinear dynamics of the plant using a piece-wise linear dynamical model that is optimised solely from the available data. Especially when local linear models are used as described, it also became possible to design a piece-wise linear controller for the plant, whose design is based on the identified model.

The questions of the existence and validity of input-output models as described and utilised was addressed theoretically using the implicit function inversion theorem that points out the observability conditions under which such models are possible to build from input-output data alone. The performance of the proposed local linear modeling scheme and the associated local linear controllers were tested on a variety of nonlinear dynamical systems including chaotic systems, a NASA aircraft and the NASA Langley transonic wind tunnel.

It was seen that the designed closed-loop control systems are extremely successful; in fact, in the experimental comparisons of performance at the NASA Langley tunnel, the proposed local linear controllers outperformed the existing computer controller and a human expert operator. These are encouraging results that motivate the use of this modeling and control technique for various other control applications. The capabilities of the local linear modeling approach are not limited to system identification and control applications. There are a wide range of nonlinear signal processing problems, such as magnetic resonance imaging, speech processing and computer vision, where the *local linear signal processing* can be employed to obtain simple but successful solutions to difficult nonlinear problems.

ACNOWLEDGMENTS

This work was partially supported by the Accurate Automation Corporation under grant #463 and by NASA under grant NAG-1-02068.

4.8 References

[1] O. Nelles: '*Nonlinear System Identification*' (Springer, New York, 2001).

[2] I.J. Leontaritis and S.A. Billings: "Input-output parametric models for nonlinear systems part I: deterministic nonlinear systems", International Journal of Control, 1985, vol. 41, no. 2, pp. 303-328.

[3] K.S. Narendra: "Neural networks for control: Theory and practice", Proceedings of IEEE, 1996, vol. 84, no. 10, pp. 1385-1406.

[4] T.A. Johansen and B.A. Foss: "Constructing NARMAX models using ARMAX models", International Journal of Control, 1993, vol. 58, no. 5, pp. 1125-1153.

[5] X. Ni, M. Verhaegen, A.J. Krijgsman and H.B. Verbruggen: "A new method for identification and control of nonlinear dynamic systems", Engineering Applications of Artificial Intelligence, 1996, vol. 9, no. 3, pp. 231-243.

[6] D.M. Walker, N.B. Tufillaro and P. Gross: “Radial-basis models for feedback systems with fading memory”, IEEE Transactions on Circuits and Systems, 2001, vol. 48, no. 9, pp. 1147-1151.

[7] B.S. Kim and A.J. Calise: “Nonlinear flight control using neural networks”, Journal of Guidance, Control and Dynamics, 1997, vol. 20, no. 1, pp. 26-33.

[8] K.S. Narendra, J. Balakrishnan and M.K. Ciliz: “Adaptation and learning using multiple models, switching and tuning”, IEEE Control Systems Magazine, 1995, vol. 15, no. 3, pp. 37-51.

[9] J.C. Principe, L. Wang and M.A. Motter: “Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control”, Proceedings of IEEE, 1998, vol. 86, no. 11, pp. 2240-2258.

[10] C. H. Lee and M. J. Chung; “Gain-scheduled state feedback control design technique for flight vehicles”, IEEE Transactions on Aerospace and Electronic Systems, 2001, vol. 37, no. 1, pp. 173-182.

[11] K.S. Narendra and C. Xiang: “Adaptive control of discrete-time systems using multiple models”, IEEE Transactions on Automatic Control, 2000, vol. 45, no. 9, pp. 1669-1686.

[12] M.A. Motter: ‘*Control of the NASA Langley 16-foot transonic tunnel with the self-organizing feature map*”. Ph.D. dissertation, University of Florida, Gainesville, Florida, 1997.

[13] K.S. Narendra and K. Parthasarathy: “Identification and control of dynamical systems using neural networks”, IEEE Transactions on Neural Networks, 1990, vol. 1, no. 1, pp. 4-27.

[14] W.T. Miller: “Real-time neural network control of a biped walking robot”, IEEE Control Systems Magazine, 1994, vol. 14, no. 1, pp. 41-48.

[15] J.D. Boskovic and K.S. Narendra; “Comparison of linear, nonlinear and neural network based adaptive controllers for a class of fed-batch fermentation process”, Automatica, 1995, vol. 31, no. 6, pp. 817-840.

[16] C.T. Chen: ‘*Introduction to linear system theory*’ (Holt, Rinehart and Winston, New York, 1970).

[17] R.C. Dorf and R.H. Bishop: ‘*Modern control systems*’ (8th ed., Addison, Wesley, New York, 1998).

[18] K. Ogata: ‘*Modern control engineering*’ (4th ed., Prentice Hall, 2001).

[19] T. Kohonen: ‘*Self-organizing maps*’ (Springer, New York 1995).

[20] D. Aeyels: “Generic observability of differentiable systems”, SIAM Journal of Control and Optimization, 1979, vol. 19, pp. 139-151.

[21] E.D. Sontag: “On the observability of polynomial systems”, SIAM Journal of Control and Optimization, 1979, vol. 17, pp. 139-151.

[22] F. Takens: “On numerical determination of the dimension of an attractor”, in *Dynamical Systems and Turbulance*, (D. Rand and L.S. Young eds.), Warwick 1980, *Lecture Notes in Mathematics*, (Springer-Verlag, Berlin, 1981), vol. 898, pp. 366-381.

[23] J. Stark, D.S. Broomhead, M.E. Davies and J. Huke: “Takens embedding theorems for forced and stochastic systems”, Nonlinear Analysis, Theory Methods and Applications, 1997, vol. 30, no. 8, pp. 5303-5314.

[24] W. Rudin: ‘*Principles of mathematical analysis*’ (McGraw-Hill, New York, 1976).

[25] R. Murray-Smith and T.A. Johansen: ‘*Multiple model approaches to modeling and control*’ (Taylor & Francis, New York, 1997).

[26] J.J. Sidorowich: “Modeling of chaotic time series for prediction, interpolation and smoothing,” Proceedings of ICASSP’92, 1992, pp. 121-124.

[27] A.C. Singer, G.W. Wornell and A.V. Oppenheim; “codebook prediction: a nonlinear signal modeling paradigm”, Proceedings of ICASSP’92, 1992, pp. 325-328.

[28] J. Stark, D.S. Broomhead, M.E. Davies and J. Huke; “Takens Embedding Theorems for Forced and Stochastic Systems”, Nonlinear Analysis: Theory, Methods and Applications, 1997, vol. 30, no. 8, pp. 5303-5314.

[29] J. Walter, H. Ritter and K. Schulten: “Nonlinear prediction with self-organizing maps”, Proceedings of IJCNN’90, 1990, pp. 589-594.

[30] M. Casdagli: “Nonlinear prediction of chaotic time series”, Physica D, 1989, vol. 35, no. 3, pp. 35-356.

[31] D.M. Walker, N.B. Tufillaro and P. Gross: “Radial-basis models for feedback systems with fading memory”, IEEE Transactions on Circuits and Systems, 2001, vol. 48, no. 9, pp. 1147-1151.

[32] X. He and H. Asada: “A new method for identifying orders of input-output models for nonlinear dynamic systems”, Proceedings of ACC’93, 1993, pp. 2520-2523.

[33] T. Martinetz, H.Ritter and K. Schulten: “Neural-gas network for vector quantization and its application to time-series prediction,” IEEE Transactions on Neural Networks, 1993, vol. 4, no. 4, pp. 558-568.

[34] G.J.Mclachlan and D. Peel: ‘*Finite mixture models*’ (Wiley and New York and 2001).

[35] S. Haykin: ‘*Adaptive filter theory*’ (4th ed. and Prentice-Hall and New York and 2001).

[36] B. Widrow and S. Stearns: ‘*Adaptive signal processing*’ (Prentice-Hall, New York, 1985).

[37] B. Schoner: ‘*Probabilistic characterization and synthesis of complex driven systems*’. Ph.D. dissertation, MIT, Cambridge, MA, 1996.

[38] D. Erdogmus and J.C. Principe: “An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems”, IEEE Transactions on Signal Processing, 2002, vol. 50, no. 7, pp. 1780-1786.

[39] S. Singhal and L. Wu: "Training multilayer perceptrons with the extended Kalman algorithm", Proceedings of NIPS'91, 1991, pp. 133-140.

[40] J.C. Principe, Y.N. Rao and D. Erdogmus: "Error whitening wiener filters: theory and algorithms", in *Least-Mean-Square Adaptive Filters*, S. Haykin and B. Widrow (eds), Wiley, 2003.

[41] B. Fritzke: "Growing cell structures – a self-organizing network for supervised and unsupervised learning", IEEE Transactions on Neural Networks, 1994, vol. 7, no. 9, pp. 1441-1460.

[42] S. Haykin: '*Neural networks: A comprehensive foundation*' (2nd ed., Prentice Hall, Englewood Cliffs, 1998).

[43] K. Hornik: "Approximation capabilities of multilayer feedforward networks", Neural Networks, 1991, vol. 4, pp. 251-257.

[44] G. Cybenko: "Approximation by superposition of a sigmoidal function", Mathematics of Control, Signals, Systems, 1989, vol. 2, pp. 303-314.

[45] E. Eskinat, S. Johnson and W.L. Luyben: "Use of Hammerstein models in identification of nonlinear systems", AIChE Journal, 1991, vol. 37, pp. 255-268.

[46] J. Cho, J.C. Principe and M.A. Motter: "Local Hammerstein modeling based on self-organizing map", Proceedings of NNSP'03, 2003, pp. 809-818.

[47] J. Wray and G.G.R. Green: "Calculation of the Volterra kernels of nonlinear dynamic systems using an artificial neural network", Biological Cybernetics, 1994, vol. 71, no. 3, pp. 187-195.

[48] H.R. Madala and A.G. Ivakhnenko: '*Inductive learning algorithms for complex systems modeling*' (CRC Press, Boca Raton, 1994).

[49] K.S. Narendra and J. Balakrishnan: "Adaptive control using multiple models", IEEE Transactions on Automatic Control, 1997, vol. 42, no. 2, pp. 171-187.

[50] R.E. Brown, G.N. Maliotis and J.A. Gibby: "PID self-tuning controller for aluminum rolling mill", IEEE Transactions on Industry Applications, 1993, vol. 29, no. 3, pp. 578-583.

[51] K.M. Vu: "Optimal setting for discrete PID controllers", IEE Proceedings D, 1992, vol. 139, no. 1, pp. 31-40.

[52] J. Bao, J.F. Forbes and P.J. McLellan: "Robust multiloop PID controller design: A successive semidefinite programming approach", Industrial and Engineering Chemistry Research, 1999, vol. 38, pp. 3407-3419.

[53] J. Lan, J. Cho, D. Erdogmus, J.C. Principe, M.Motter and J. Xu: "Local linear PID controllers for nonlinear control", to appear in International Journal of Control and Intelligent Systems, 2005.

[54] J. Cho, J.C. Principe, D. Erdogmus and M.A. Motter: "Modeling and inverse controller design for an unmanned aerial vehicle based on the self organizing map", submitted to IEEE Transactions on Neural Networks, 2003.

[55] J.Y. Hung, W. Gao and J.C. Hung: "Variable structure control: A survey", IEEE Transactions on Industrial Electronics, 1993, vol. 40, no. 1, pp. 2-22.

[56] D. Erdogmus: '*Optimal trajectory tracking guidance of an aircraft with final velocity constraint*'. MS Thesis, Middle East Technical University, Ankara, Turkey, 1999.

[57] S.M. Kay: '*Modern spectral estimation: Theory and application*' (Prentice Hall, Englewood Cliffs, 1988).

[58] D. Kaplan and L. Glass: '*Understanding nonlinear dynamics*' (Springer-Verlag, New York, 1995).

[59] J.C. Principe, N. Euliano and C. Lefebvre: '*Neural and adaptive systems: Fundamentals through simulations*' (Wiley, New York, 1999).

[60] C. Cox, J. Neidhoefer, R. Saeks and G. Lendaris: "Neural adaptive control of LoFLYTE", Proceedings of ACC'01, 2001, vol. 4, pp.2913-2917.

[61] C. Cox, K. Mathia and R. Saeks: "Learning flight control and LoFLYTE," Proceedings of WESCON'95, 1995, pp.720-723.

[62] L.V. Schmidt: '*Introduction to aircraft flight dynamics*' (American Institute of Aeronautics and Astronautics, Reston, VA, 1998).

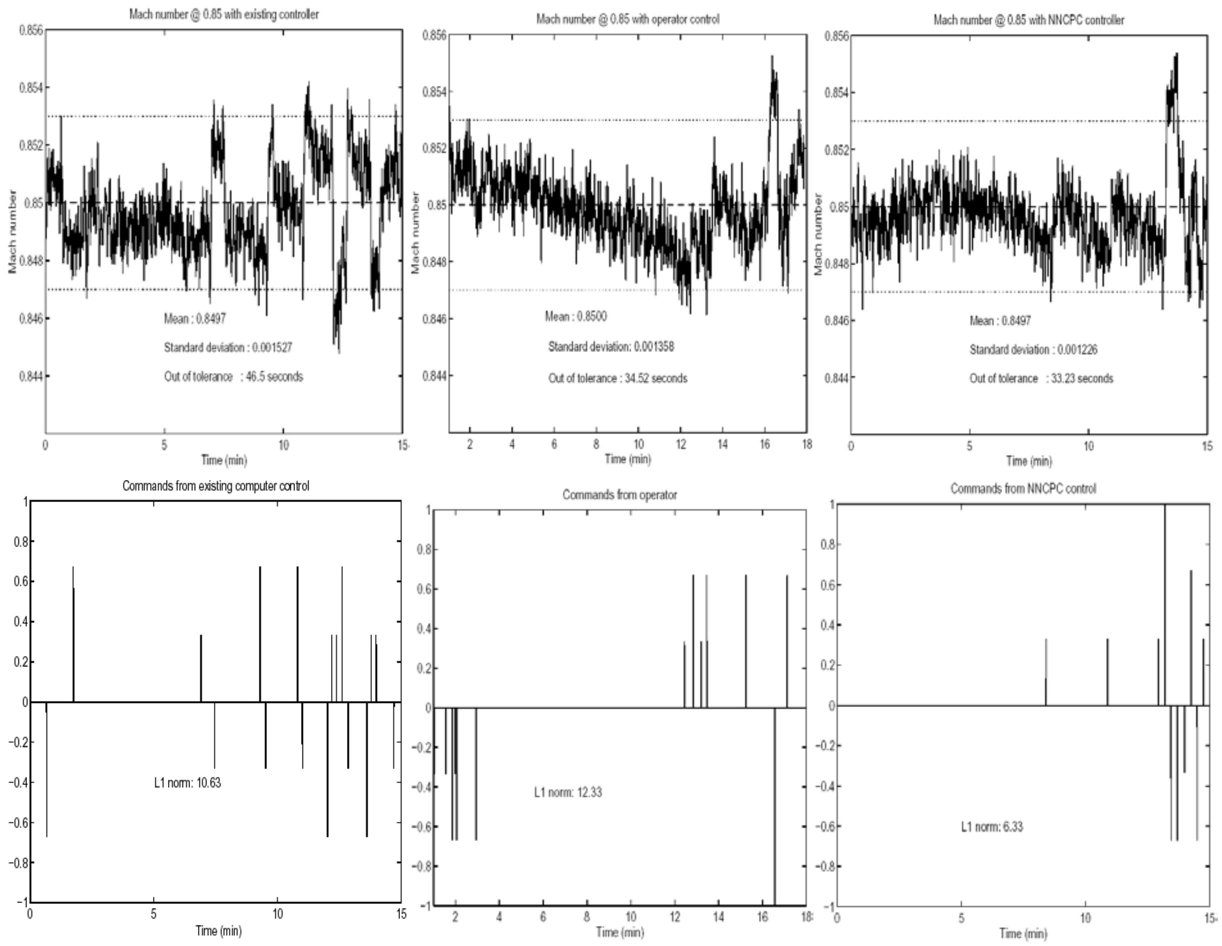

Figure 4.16. Comparison of the existing controller (left), expert human operator (middle), and the SOM-LLM controller (right). The achieved Mach numbers for 15 minutes superimposed on the set point and the tolerance bounds (top), and control inputs produced by the controllers (bottom).

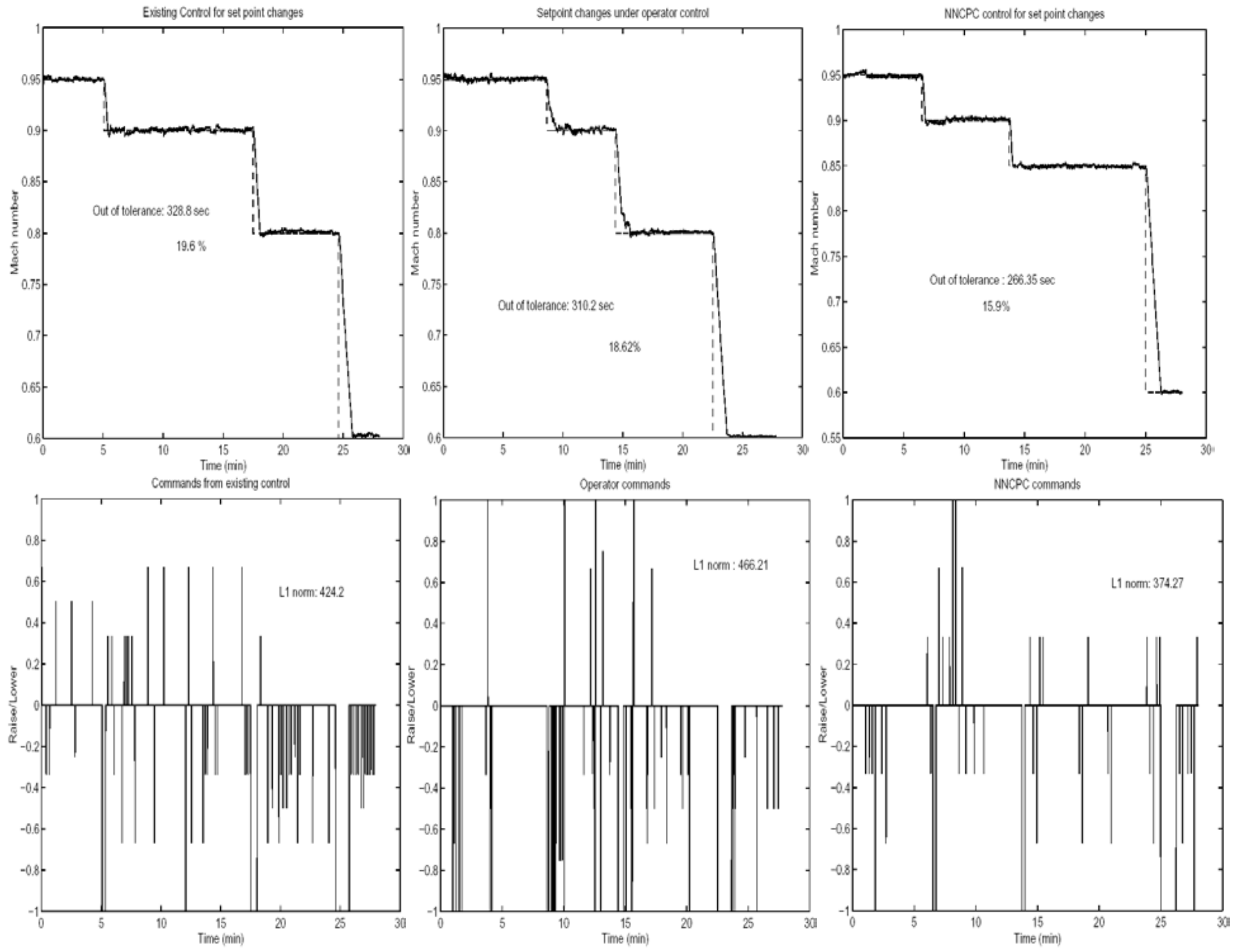

Figure 4.17. Comparison of the existing controller (left), expert human operator (middle), and the SOM-LLM controller (right). Mach number tracking step changes in set point for 28 minutes (top), and control inputs produced by the controllers (bottom).