A neural architecture for nonlinear adaptive filtering of time series

Hoffmann, Nils; Larsen, Jan

Published in:
Proceedings of the IEEE Workshop Neural Networks for Signal Processing

Link to article, DOI:
10.1109/NNSP.1991.239488

Publication date:
1991

Document Version
Publisher's PDF, also known as Version of record

A NEURAL ARCHITECTURE FOR NONLINEAR ADAPTIVE FILTERING OF TIME SERIES

Nils Hoffmann and Jan Larsen
The Computational Neural Network Center
Electronics Institute, Building 349
Technical University of Denmark
DK-2800 Lyngby, Denmark

INTRODUCTION

The need for nonlinear adaptive filtering may arise in different types of filtering tasks such as prediction, system identification and inverse modeling [17]. The problem of predicting chaotic time series has been addressed by several authors [6],[11]. In the latter case a feed-forward neural network was used both for prediction and for identification of a simple, nonlinear transfer function (system identification). These filtering tasks can be solved using a filtering configuration shown in Fig. 1 [17].

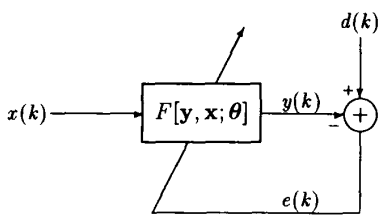

Figure 1: Nonlinear adaptive filtering configuration. $x(k)$ is the input, $y(k)$ the output and $d(k)$ the desired signal. The filter is adapted in order to minimize the cost function: $\sum_{i=0}^{k} \lambda^{k-i} e^2(i)$, where $k = 1, 2, \cdots, N$ and $0 < \lambda \leq 1$ is the forgetting factor [7].

The nonlinear filter may be designed to realize

$$y(k) = F[y(k-1), \cdots, y(k-M), x(k), \cdots, x(k-L+1); \boldsymbol{\theta}] \qquad (1)$$

where $F[\cdot]$ is an unknown nonlinear function parameterized by $\boldsymbol{\theta}$, k is the discrete time index and L, M are filter orders.

0-7803-0118-8/91/0009-0533$01.00

The general structure of equation (1) enables one to model any nonlinear, discrete system. θ is assumed to be slowly time varying and consequently $y(k)$ is quasi stationary. The use of a recursive, nonlinear filter does, however, pose serious difficulties regarding stability. The filter may display limit cycles, chaotic behaviour and unboundedness. The scope of this paper is to implement only the nonrecursive part of (1).

We propose a modularized architecture for the nonlinear filter in Fig. 1 including algorithms for adapting the filter. Further we develop simple guidelines for selecting a specific filter design within the proposed architecture given a priori knowledge of the distribution and origin (type of modeling problem) of the input signal $x(k)$ and the desired response $d(k)$. Finally we present simulations in order to further investigate the nature of the relations between filter design and the statistics of x and d.

NONLINEAR FILTER ARCHITECTURE

The proposed filter architecture is shown in Fig. 2. The filter, which may be viewed as a generalization of the Wiener Model [15], is divided into three partially independent (depending on specific design) sections: A preprocessing unit containing the filter memory, a memoryless, multidimensional nonlinearity (MMNL) and a linear combiner. The structure is selected in order to modularize the modeling problem which ensures a proper and sparse parameterization, and facilitates incorporation of a priori knowledge in contrast to the limited possibilities when using an ordinary feed-forward neural network. The main objective of the preprocessor is to extract the essential

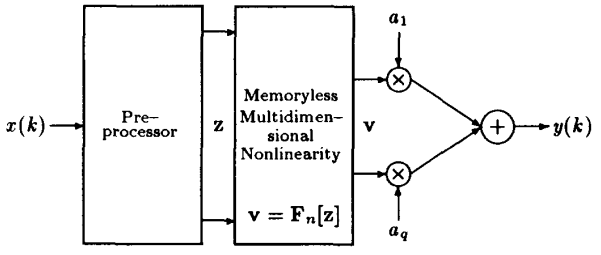

Figure 2: Nonlinear filter architecture.

information contained in $\mathbf{x}_k = [x(k), x(k-1), \cdots, x(k-L+1)]'$ ensuring that $\mathbf{z}$ has a dimension, $p \leq L$. The nonlinearity is memoryless and transforms the vector $\mathbf{z}$ into the vector $\mathbf{v}$, and finally the linear combiner forms a weighted sum $y(k)$ of the terms in $\mathbf{v}$. This corresponds to rewriting (1) as:

$$y(k) = \mathbf{a}'\mathbf{F}_n[\mathbf{F}_p(x(k))] \tag{2}$$

where $\mathbf{F}_p(\cdot)$ is the preprocessor, $\mathbf{F}_n[\cdot]$ is the MMNL and $\mathbf{a}$ the weigths of the linear combiner. The filter could be viewed as a heterogeneous multilayer neural network. All sections could be adapted, but this is not always necessary, as we shall see in the next section.

FILTER DESIGN

Signal Dependence

It seems reasonable that the specific design of the filter depends on the origin and distribution of the signals $x(k)$ and $d(k)$, and we will summarize some guidelines for choosing an appropriate design as follows:

Case	x	d	Model
1	Gaussian	Gaussian	$d(k) = \mathbf{a}'\mathbf{x}_k + \epsilon(k)$
2	Gaussian	Non-Gaussian	$d(k) = F[\mathbf{x}_k] + \epsilon(k)$
3	Non-Gaussian	Gaussian	$d(k) = F[\mathbf{x}_k] + \epsilon(k)$
4	Non-Gaussian	Non-Gaussian	$d(k) = F[\mathbf{x}_k] + \epsilon(k)$

1. The linear filter is optimal in this case, so $\mathbf{v} = [1\ \mathbf{z}']'$, see e.g. [13, Theorem 14.3].
2. The Wiener model, i.e. a bank of orthogonal linear filters in the preprocessing unit followed by a fixed (non-adaptive) polynomial nonlinearity, provides a filter which can be adapted in a very simple and fast way [15]. The linear filters may be determined using principal component analysis (PCA) on $\mathbf{x}$. This case is associated with the problem of nonlinear system identification where $d(k)$ is the output of an unknown system and $x(k)$ the input.
3. In this case there is no obvious choice of filter design. The case arises e.g. in inverse modeling where $d(k)$ is the driving signal of the unknown system and $x(k)$ the resulting output.
4. This case relates to prediction of nonlinear time series where $x(k) = d(k-\tau)$ is, in fact, a delayed version of $d(k)$. Previous simulation studies [11] indicate that the nonlinearity should be constructed from bounded functions (e.g. the commonly used $\tanh(\cdot)$) rather than polynomials, which have the inconvenient property of growing fast towards infinity.

Preprocessing Methods

We present two possible methods for dimensionality determination. If the unknown system being modeled can be described in the form of a nonlinear differential equation it is possible, in some cases, to determine dimensionality by letting $\mathbf{z} = [x(k), Dx(k), \cdots, D^{p-1}x(k)]'$ where D is a discrete derivative operator. We denote this preprocesor: The **derivative preprocessor** (DPP).

The following example (the pendulum) illustrates that we often have $p < L$ without loosing information:

$$\frac{d^2x(t)}{dt^2} + \beta\frac{dx(t)}{dt} + \alpha\sin[x(t)] = d(t) \tag{3}$$

$$D^2x(k) + \beta\ Dx(k) + \alpha\sin[x(k)] = F[D^2x(k), Dx(k), x(k)] = d(k) \tag{4}$$

Equation (4) which is a discrete approximation of (3) clearly shows, that $d(k)$ may be expressed as a function of only $p = 3$ variables i.e. the derivatives of $x(k)$. To ensure that the approximation of the derivative operator D is accurate (i.e. approximates the continuous time derivative operator) over a broad range of frequencies it must be implemented using a linear filter with high order. A tapped delay line used without a preprocessing element would necessarily need the same length L to hold the same information so L is obviously greater than p in this example. In practice there exist two major problems with this kind of preprocessing: 1. Derivatives amplify noise (SNR is often low at high frequencies which are amplified the most) and 2. The optimal differentiating filter is non-causal. The first problem may be dealt with by noise reducing lowpass-filtering (see [3] for an optimal approach). The second obstacle may be circumvented by delaying $d(k)$ thus allowing for non-causal filtering i.e. estimating $d(k-r)$ $(r \geq 0)$ using $x(k), x(k-1), \cdots$.

Another method is principal component analysis (PCA) which serves two purposes: 1. PCA makes the components in $\mathbf{z}$ mutually uncorrelated (convergence speed-up for certain weight estimation algorithms, e.g. Backpropagation [8]) and 2. It determines the dimensionality which is done by removing the last $L - p$ principal components (PC's) with eigenvalues close to zero. The amount of lost information can be estimated as the total variance of the removed PC's divided by the total variance of $\mathbf{x}$, i.e. $L \cdot V\{x(k)\}$. The remaining PC's constitute the optimal *linear* projection (in the mean square sense) of $\mathbf{x}$ on the space spanned by the first p eigenvectors of the covariance matrix of $\mathbf{x}$. A theoretical wellfounded procedure for on–line estimation of the PC's has recently been described [18]. Other schemes are given in [8, Chap. 8.3].

Further support for the use of PCA can be found in an information theoretical interpretation: Maximize the mutual information $I(\mathbf{x};\mathbf{z})$ between $\mathbf{x}$ and $\mathbf{z}$. It can be shown that if $\mathbf{z}$ is Gaussian (also valid for certain similar probability density functions): $\max I(\mathbf{x};\mathbf{z}) = \max H(\mathbf{z}) \Leftrightarrow \max\det V\{\mathbf{z}\}$ where $V\{\cdot\}$ is the variance and $H(\cdot)$ is the entropy. PCA is in fact done by maximizing $\det V\{\mathbf{z}\}$ [10, p. 682] which implicates that dimensionality determination by means of PCA is equivalent to maximizing $I(\mathbf{x};\mathbf{z})$. When dealing with signals corrupted by noise PCA is not always preferable (especially if the signal to noise ratio is low) because the PC's then will reflect the noise. Furthermore using PCA when the spectral overlap of the signals $x(k)$ and $d(k)$ is small is not reasonable. This is due to the fact that the spectrum of the PC's corresponding to large eigenvalues mainly contains the

dominating frequencies in $x(k)$ thus neglecting the frequencies that dominate the spectrum of $d(k)$.

Memoryless Multidimensional Nonlinearities

When approximating the MMNL, $\mathbf{F}_n[\mathbf{z}]$, $\mathbf{z} \in \mathcal{I}$ where $\mathcal{I}$ is the input space, we distinguish between *local* and *global* approximation methods [6]. In a local approximation context $\mathcal{I}$ is divided into smaller domains. F is now approximated in each domain by separate nonlinearities. This results in a modularization of the MMNL which ensures a sparse parameterization. By global approximation is meant, that no dividing of $\mathcal{I}$ is done at all. In general there is a trade off between the number of domains and the complexity of the subsequent nonlinearities.

Global Approximation Methods. In this case we deal with only one nonlinearity which must have the ability of approximating F arbitraryly accurate. We will discriminate between *fixed* and *adaptive* nonlinearities.

A natural choice of a **f**ixed **n**onlinearity (FNL) is to let $\mathbf{v}$ contain all possible products of z_i, e.g. terms of the form $\prod_{i=1}^{p} z_i^{s_i}$ up to some order $s = \sum_{i=1}^{p} s_i$. When these terms are added by the linear combiner it all amounts to a multidimensional Taylor expansion which combined with the linear filters in the preprocessor defines a discrete Volterra filter. Fréchet showed [15] that any continuous $F(x(t))$ can be represented by a Volterra filter with uniform convergence when $s \to \infty$ for $x(t) \in \mathcal{J}$, $\mathcal{J} \subseteq \mathcal{I}$. A convenient representation can be obtained by using a complete set of orthogonal polynomials P_i, where i is the order of the polynomial. If $z_i \in N(0,1)$, P_i are identical to the Hermite polynomials. With these polynomials convergence in mean is assured over a suitable interval $[a;b]$. The generalization to the multidimensional case is done by forming all products of polynomials in different variables e.g. $\prod_{i=1}^{p} P_{s_i}(z_i)$ (see [15] for details). In general the probability density $f_{\mathbf{Z}}(\mathbf{z})$ is unknown which makes it impossible to find the orthogonal polynomials. Instead we propose the use of Chebychev polynomials preceeded by squashing functions that limits the z_i to the interval $]-1;1[$ thereby limiting the v_i to $]-1;1[$.

An obvious choice for an **a**daptive **n**onlinearity (ANL) is a layered feed-forward neural network composed of sigmoidal neurons. It is well-known (see e.g. [5]) that a two layer feed-forward network with a linear output neuron (under rather mild conditions on the activation function, g) can uniformly approximate any function as the number of neurons in the hidden layer reaches infinity. We suggest that the nonlinearity is composed of to layers. The first layer consists of p neurons which maps z_i, $1 \leq i \leq p$ into $g(z_i w_i^1 + w_i^0)$ (w_i^1 ensures a proper scaling, see below). The second layer consists of q neurons ($q \to \infty$ for an arbitrarily accurate approximation) and the outputs then form the nonlinear terms $[v_1, \cdots, v_{q-p}]'$. It is further suggested to explicitly model the linear part by letting $[v_{q-p+1}, \cdots, v_q]' = [z_1, \cdots, z_p]'$.

Local Approximation Methods. A possible way to divide the input space is to use Localized Receptive Fields [14]. The output from each re-

ceptive field is then fed into separate nonlinearities. As above they could be either fixed or adaptive. Note that there is a trade-off between the number of domains in the input space and the complexity of the succeeding nonlineraities. Other local approximation schemes can be found in [6], [16], [15, Chap. 21].

Scaling of z. Scaling of $\mathbf{z}$ serves two purposes. First, we have to restrict z_i to an interval where the nonlinearity is slowly varying; i.e. neither growing towards infinity (as polynomials for large arguments) nor being constant (like $\tanh(\cdot)$ for large arguments). Secondly, we have to ensure that only the significant amplitude range of z_i (i.e. the interval where $f_{z_i}(z_i) > \varepsilon$, $0 < \varepsilon \ll 1$) is fed into the filter. Otherwise very unlikely values of z_i will be weighted too much in the cost function thus resulting in a poor performance. Scaling with a suitable measure of z_i, e.g. 2–3 standard deviations, serves this purpose.

Weight Estimation Algorithms

The task is to estimate the weights $\boldsymbol{\theta}$ so that the cost function $\sum_{i=0}^{k} \lambda^{k-i} e^2(i)$, $k = 1, 2, \cdots, N$ is minimized [7] where e is the difference between the desired and the actual response and $0 < \lambda \leq 1$ is the forgetting factor.

Fixed Nonlinearity. In designs with a FNL it is only necessary to adapt the linear combiner. This is especially simple if $\mathbf{x}$ is Gaussian and PCA is used as preprocessing making $\mathbf{z}$ white (independent) and Gaussian. Now if z_i is scaled to unity variance and Hermite polynomials are used in the nonlinearity then the v_j will be uncorrelated and the weights may thus be updated using the crosscorrelation method proposed in [15]: $a_j = C\{v_j d\}/V\{v_j\}$, $1 \leq j \leq q$ where $C\{v_j d\}$ is the covariance between v_j and d and $V\{v_j\}$ is the variance of v_j. In most cases, however, $\mathbf{v}$ is non-white but owing to the fact that $y(k)$ is linear in the weights, adaptive algorithms known from linear adaptive filtering such as the recursive least squares (RLS) [7, p. 385] or the least mean squares (LMS) [17, p. 99] are usable. The latter is perhaps the best choice for large values of q because it needs less computations and memory capacity while the major advantage of the RLS is the much faster convergence for highly correlated inputs (v_j).

Adaptive Nonlinearity. Designs with ANL implicate that estimation of the weights is a nonlinear optimization task which in general is hard to solve (global optimization) but local optimization schemes have been given, e.g. Backpropagation (BP). BP is known to have very slow convergence [4]. There is therefore a need for development of algorithms with faster convergence. Several second-order algorihms (SOA) have been proposed, see e.g. [4]. A SOA incorporates the information contained in the Hessian ($\mathbf{H}$) of the cost function and the weights are updated according to the Newton-Raphson algorithm. In contrast to BP the SOA parameters are given a natural interpretation. $0 < \lambda \leq 1$ is the exponential forgetting factor, $0 < \mu \leq 1$ is the stepsize which normally is non-critical, and δ ($\mathbf{H}^{-1} = \delta \mathbf{I}$) is initially chosen large. We suggest a further development that takes the problems of nearly

singular Hessian matrices into account. This problem arises in "flats" parts of the cost function. It is proposed to use the U-D factorization of $\mathbf{H}^{-1}$ due to Bierman [2].

SIMULATIONS

Simulated Systems

In order to compare filter designs when filtering signals with different origin and distribution we study three systems covering the cases 2–4 on p. 3.

Example	x	d	Model
System Identification (SI)	Band-limited Gaussian noise	Non-Gaussian	Equation (5)
Inverse Modelling (IM)	Non-Gaussian	Gaussian lowpass filtered noise	Equation (6)
Prediction (P)	Non-gaussian	Non-gaussian	Equation (7)

System Identification. A simple system describing a wave force problem is given by Morison's equation [1, p. 234]. $x(t)$ is the wave velocity and $d(t)$ the wave force . The desired signal $d(k)$ is a discrete version of $d(t)$ sampled with the sampling period $\Delta\, T = 1$ and the same applies to $x(k)$.

$$d(t) = 0.2\frac{dx(t)}{dt} + 0.8x(t)|x(t)| \tag{5}$$

Inverse Modelling. We consider the pendulum where $x(t)$ is the angle deflection and $d(t)$ the force. The desired signal $d(k)$ is a discrete version of $d(t-\tau)$ (sampled with $\Delta\, T = 0.05$) where τ is a delay aiming to cancel both the delay between $d(t)$ and $x(t)$ and the delay in the preprocessing unit of the nonlinear filter. $x(k)$ corresponds to the angle $x(t)$.

$$\frac{d^2x(t)}{dt^2} + 0.2\frac{dx(t)}{dt} + 4\pi^2 \sin[x(t)] = d(t) \tag{6}$$

Prediction. The signal $x(t)$ is generated by the chaotic Mackey-Glass equation which often is used in a benchmark test for nonlinear predictors [6]. $d(k)$ is a discrete version of $x(t)$ and $x(k)$ a delayed, discrete version of $x(t-\tau)$ where τ signifies how far ahead we predict. Sampling the signal with $\Delta\, T = 1$ τ equals 100 timesteps like in [6],[11].

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-17)}{1+x(t-17)^{10}} \tag{7}$$

The systems mentioned above have all been simulated using discrete approximations of the derivatives. These discrete filters are all non-recursive which means that a non-recursive nonlinear adaptive filter is adequate. The actual training and cross validation signals have been obtained by decimating the input and output signals in order to avoid oversampling.

Numerical Results

In the table below are listed the main results. A measure of the filter performance is given by the error index: $E = \sigma_e/\sigma_d$, where σ_e, σ_d denote the standard deviation of the error and the desired signals respectively (cross validation). The number of parameters W gives an indication of the complexity.

Ex.	Prep.	L	p	Nonlinearity					
				Fixed		Adaptive		None	
				E	W	E	W	E	W
SI	PCA	14	4	0.263	94	0.215	158	0.417	5
SI	DPP	19	2	0.200	46	0.107	128	0.389	3
SI	None	14	-	-	-	-	-	0.382	15
IM	DPP	19	3	0.075	152	0.116	131	0.448	4
IM	None	19	-	-	-	-	-	0.402	20
P	PCA	20	4	0.152	170	-	-	0.539	5
P	None	20	-	-	-	-	-	0.539	21

In all simulations we have used 9000 samples for training and 8000 for cross validation. During training an algorithm based on a statistical test [12], [9] was used to eliminate non-significant weights which accounts for the variations in W. The FNL consisted of bounded Chebychev polynomials and the ANL was implemented using af multilayer neural net. In both cases we used 2. order algorithms for adapting the weights. In general the simulations indicate that the nonlinear filters are clearly superior to the linear with respect to E. This improvement is, however, gained at the expense of an increased complexity. The FNL and ANL's seems to show roughly equal perfomance on the selected examples, with the ANL having a better parameterization in the SI example (more significant weights and lower E) and vice versa in the IM example. The two preprocessing methods seem to complement each other. In the examples SI,IM, where the equations can be closely approximated with discrete derivative operators of low order, the use of discrete differentiating filters in the preprocessing unit yields a better performance with lower complexity than the use of PCA. This shows that the discrete derivatives are more informative than the PC's in the chosen examples. Using PCA in the example with the pendulum is in fact extremely bad because the PC's mainly reflect the low frequency components in $x(k)$ while the high frequencies carry most of the information about $d(k)$. In contrast, PCA works very well for the example of prediction whereas it is not easy to approximate the Mackey-Glass equation using only low-order derivatives (i.e. the use of a DPP is a bad choice). Finding an appropriate preprocessing method seems thus to require knowledge of an approximate mathematical model for the unknown system. Alternatively a rule of thumb saying that the preprocessor should make the spectrums of the $z_i(k)$ "close" to the spectrum of $d(k)$ could be used. In the prediction example we have used a PCA with $L = 20$ and $p = 4$ which allows us to compare our results with the ones obtained in [11] where

a performance of $E = 0.054$ was found using an ANL and a total of 171 weights and $E = 0.28$ using a 6th order fixed polynomial nonlinearity and 210 weights. This indicates, that allthough the ANL still performs better on this example an increase in performance of the FNL has been gained by using bounded polynomials.

CONCLUSION

In this paper a neural architecture for adaptive filtering which incorporates a modularization principle is proposed. It facilitates a sparse parameterization, i.e. fewer parameters have to be estimated in a supervised training procedure. The main idea is to use a preprocesssor which determine the dimension of the input space and further can be designed independent of the subsequent nonlinearity. Two suggestions for the preprocessor are presented: The derivative preprocessor and the principal component analysis. A novel implementation of fixed Volterra nonlinearities is given. It forces the boundedness of the polynominals by scaling and limiting the inputs signals. The nonlinearity is constructed from Chebychev polynominals. We apply a second-order algorithm for opdating the weights for adaptive nonlinearities based on previous work of Chen et al. [4] and the U-D factorization of Bierman [2]. Finally the simulations indicate that the two kinds of preprocessing tend to complement each other while there is no obvious difference between the performance of the ANL and FNL.

ACKNOWLEDGEMENTS

We would like to thank Lars Kai Hansen, Peter Koefoed Møller, Klaus Bolding Rasmussen and John E. Aasted Sørensen for helpfull comments on this paper.

REFERENCES

[1] J.S. Bendat, Nonlinear Systems Analysis and Identification, New York: JOHN WILEY & SONS, 1990.

[2] G.J. Bierman, "Measurement Updating using the U-D Factorization," in Proceedings of the IEEE Int. Conf. on Decision and Control, 1975, pp. 337–346.

[3] B. Carlsson, A. Ahlén & M. Sternad, "Optimal Differentiation Based on Stochastic Signal Models," IEEE Transactions on Signal Processing, vol. 39, no. 2, 341–353, 1991.

[4] S. Chen. , C.F.N. Cowan, S.A. Billings & P.M. Grant, "Parallel Recursive Prediction Error Algorithm for Training Layered Neural Networks," Int. Journal of Control, vol. 51, no. 6, pp. 1215–1228, 1990.

[5] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," Math. Control Signal Systems, no. 2, pp. 303–314, 1989.

[6] J.D. Farmer & J.J. Sidorowich, "Exploiting Chaos to Predict the Future and Reduce Noise," Technical Report LA-UR-88, Los Alamos National Laboratory, 1988.

[7] S. Haykin, Adaptive filter theory, Englewood Cliffs, New Jersey: PRENTICE-HALL, 1986.

[8] J. Hertz, A. Krogh & R.G. Palmer, Introduction to the Theory of Neural Computation, Redwood City, California: ADDISON-WESLEY PUBLISHING COMPANY, 1991.

[9] N. Hoffmann & J. Larsen, "An Algorithm for Parameter Reduction in Nonlinear Adaptive Filters," in preparation for submission to IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco Marriott, California, March 23–26 1992.

[10] P.R. Krishnaiah (ed.), Multivariate Analysis 2, New York: ACADEMIC PRESS, 1969.

[11] A.S. Lapedes & R. Farber, "Nonlinear Signal Processing Using Neural Networks, Prediction and System Modeling," Technical Report LA-UR-87, Los Alamos National Laboratory, 1987.

[12] J. Larsen & L.K. Hansen, "Reduction of Neural Network Complexity," submitted to Neural Information Processing Systems, Denver, Colorado, Dec. 2–5, 1991.

[13] J.M. Mendel, Lessons in Digital Estimation Theory Englewood Cliffs, New Jersey: PRENTICE-HALL, 1987.

[14] J. Moody & C.J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," Neural Computation, no. 1, pp. 281–294, 1989.

[15] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, Malabar, Florida: ROBERT E. KRIEGER PUBLISHING COMPANY, 1989.

[16] H. Tong & K.S. Lim, "Threshold Autoregression, Limit Cycles and Cyclical Data," Journal of the Royal Statistical Society, vol. 42, no. 3, pp. 245-292, 1980.

[17] B. Widrow & S.D. Stearns, Adaptive Signal Processing, Englewood Cliffs, New Jersey: PRENTICE-HALL, 1985.

[18] Kai-Bor Yu, "Recursive Updating the Eigenvalue Decomposition of a Covariance Matrix," IEEE Transactions on Signal Processing, vol. 39, no. 5, pp. 1136–1145, 1991.