CONFIGURING STACK FILTERS BY THE LMS ALGORITHM

Nirwan Ansari and Yuchou Huang
Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
University Heights
Newark, New Jersey 07102

Jean-hsang Lin
Department of Electrical Engineering
University of Delaware
Newark, Delaware 19716

Abstract – Stack filters are a class of sliding–window nonlinear digital filters that possess the weak superposition property (threshold decomposition) and the ordering property known as the stacking property. They have been demonstrated to be robust in suppressing noise. In this paper, a new method based on the Least Means Squares (LMS) algorithm is developed to adaptively configure a stack filter. Experimental results are presented to demonstrate the effectiveness of the proposed method to noise suppression.

INTRODUCTION

The subject of adaptive filters [1] has matured to the point where it now constitutes an important part of statistical signal processing. Whenever there is a requirement to process signals that result from operation in an environment of unknown statistics, the use of adaptive filters offers an attractive solution to the problem as it usually provides a significant improvement in performance over the use of a fixed filter designed by conventional methods. Furthermore, it provides new signal processing capabilities that would not be possible otherwise.

All stack filters [2]–[5] obey the threshold decomposition property and the stacking property. The difference between two stack filters lies solely in the Boolean operation performed on each level. A necessary and sufficient condition for a Boolean operation to preserve the stacking property [6] is that the operation must be positive, in which case it has a minimum sum of products representation which is free of complements of any of the variables. Thus, only the logical AND and OR operations are permitted. A brief discussion on stack filters will be presented in Section 2.

Adaptive approaches to configuring stack filters have recently been proposed [7],[8]. In this paper, a new method based on LMS algorithm is developed to configure stack filters. The LMS algorithm [9] along with its properties will be discussed. The incorporation of the LMS learning to configure stack filters will then be developed.

STACK FILTERS

Median filters [2],[3] and other rank-order operators [4] possess two properties called threshold decomposition property and the stacking property. The first is a limited superposition property; the second is an ordering property which allows an efficient VLSI implementation of the threshold decomposition architecture.

0-7803-0118-8/91/0009-0570$01.00

Any filter which possesses the threshold decomposition property and the stacking property is known as a stack filter [5]. Based on the threshold decomposition property and the stacking property, stack filters can be constructed as a "stack" of Positive Boolean functions [6]. Thus, stack filters form a very large class of easily implemented nonlinear filters which include the rank order operators as well as all compositions of morphological operators.

Passing an M-valued discrete time signal through a rank-order filter is equivalent to the following procedure:

(i) Decomposing the M-valued input signal into a set of M-1 binary signals. The kth binary signal, where k is an integer in {1,2,...,M-1}, is obtained by thresholding the input signal at value k. That is, it takes a value of 1 whenever the input signal is greater than or equal to k, but it is 0 otherwise. Note that summing these M-1 binary signals always provides the original input signal.

(ii) Filtering each binary signal independently with its own rank-order filter. Note that each threshold level is performed in parallel. During the process of filtering, each rank-order filter simply adds the number of 1's in the window and compares the result to an integer r, the desired rank of the filter. The output is 1 when the summation is greater than or equal to r, and 0 when the summation is less than r. For example, if the filter's window is b=2r+1, the filter is a median filter.

(iii) Adding the output of each binary rank-order filter one sample at a time. It is found that the output of the rank-order filter possesses the stacking property. This means that the binary output signals are piled on top of each other according to their threshold levels. It can be seen that a column of 1's always has a column of 0's on top. The desired output value is simply the value of the threshold level where the transition from 1 to 0 takes place.

Consider an M-valued sequence s. The threshold signals $\vec{T}^1, \vec{T}^2, \cdots, \vec{T}^{M-1}$ of the sequence are defined by

$$\vec{T}^j(i) = \begin{cases} 1 & \text{if } s(i) \geq j \\ 0 & \text{if } s(i) < j, \end{cases} \tag{1}$$

where i stands for the ith element of the appropriate vector and each element of a threshold vector is binary. Note that these threshold vectors possess the stacking property

$$\vec{T}^1 \geq \vec{T}^2 \geq \cdots \geq \vec{T}^{M-2} \geq \vec{T}^{M-1}, \tag{2}$$

which implies

$$\vec{T}^1(i) \geq \vec{T}^2(i) \geq \cdots \geq \vec{T}^{M-2}(i) \geq \vec{T}^{M-1}(i), \tag{3}$$

where i is the ith element of the appropriate vector.

Let s_1 and s_2 be two binary sequences. A filter defined by a function $F(\cdot)$ is said to have the stacking property if

$$F(s_1) \geq F(s_2) \quad \text{whenever} \quad s_1 \geq s_2. \tag{4}$$

Based on Equations (2), (3) and (4), the output of this filter should have the following relationship

$$F(\vec{T}^1) \geq F(\vec{T}^2) \geq \cdots \geq F(\vec{T}^{M-2}) \geq F(\vec{T}^{M-1}). \tag{5}$$

The function $F(\cdot)$ of a rank-order filter, in fact, is a Boolean function. It means that if a filter defined by the function $F(\cdot)$ has the stacking property, then $F(\cdot)$ must be a positive Boolean function. The operation of these positive Boolean functions is simply the "max" and "min" operations. For example, $F(x_1, x_2, x_3) = x_1 + x_3$, which is a stack filter, is equivalent to $max(x_1, x_3)$.

Stack filters are a generalization of median filters. The structure of a median filter root signal [3] is well known, and the structure and analysis of stack filters are similar. Properties and analysis of stack filters have been covered in great details [5].

LINEAR DISCRIMINANT FUNCTIONS AND THE LMS ALGORITHM

The problem of finding a linear discriminant function will be formulated as a problem of minimizing a criterion function. A linear discriminant function [10] is defined by

$$g(\mathbf{x}) = \mathbf{a}^t\mathbf{x} + w_0, \quad \text{or} \tag{6}$$

$$g(\mathbf{u}) = \mathbf{w}^t\mathbf{u}, \tag{7}$$

where

$$\mathbf{u} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ \mathbf{a} \end{bmatrix}.$$

$\mathbf{x}$ is an input pattern vector, $\mathbf{a}$ is the weight vector, and w_0 is the threshold weight. The linear discriminant function is used to classify input pattern $\mathbf{x}$ as one of two possible categories, and the decision rule is: Decide class **A** if $g(\mathbf{x}) > 0$, and class **B** if $g(\mathbf{x}) < 0$. Thus $\mathbf{x}$ is assigned to class **A** if the inner product $\mathbf{a}^t\mathbf{x}$ exceeds the threshold, $-w_0$, and $\mathbf{x}$ is assigned to class **B** if the inner product $\mathbf{a}^t\mathbf{x}$ is less than the threshold, $-w_0$. When $\mathbf{a}^t\mathbf{x}$ is equal to $-w_0$, $\mathbf{x}$ can be assigned to either class **A** or class **B**.

If $g(\mathbf{x}) = 0$, it defines the decision surface that separates points assigned to class **A** from points assigned to class **B**. When $g(\mathbf{x})$ is linear as shown in Equation (6), this decision surface is a *hyperplane*. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on the decision surface, then

$$\begin{aligned} \mathbf{a}^t\mathbf{x}_1 + w_0 &= \mathbf{a}^t\mathbf{x}_2 + w_0 \\ \mathbf{a}^t(\mathbf{x}_1 - \mathbf{x}_2) &= 0. \end{aligned} \tag{8}$$

Therefore, $\mathbf{a}$ is normal to the hyperplane. The orientation of the decision surface is decided by $\mathbf{a}$, and the location of surface is determined by w_0.

Consider the structure of an adaptive threshold logic element in Fig. 1. This structure has two parts: (1) A transversal filter with adjustable tap weights whose values at time n are denoted $w_1(n), w_2(n), \cdots, w_N(n)$, and (2) a mechanism for adjusting these tap weights in an adaptive manner.

During the filtering process, an additional signal $d(n)$, called the *desired response*, is supplied along with the usual tap input. The desired signal response provides a reference for adjusting the tap weights of the filter. Denote $e_L(n)$ as the estimation error produced during LMS learning. Thus as shown in Fig. 1,

$$e_L(n) = d(n) - \mathbf{w}^t(n)\mathbf{u}(n), \tag{9}$$

where the term $\mathbf{w}^t(n)\mathbf{u}(n)$ is the inner product of the tap weight vector $\mathbf{w}(n)$ and the tap input vector $\mathbf{u}(n)$, and the superscript t stands for vector or matrix

transpose. Since there are $N+1$-input which correspond to the N inputs for a filter width of N and a threshold determined by w_0, thus the weight vector is $\mathbf{w}^t(n) = [w_0(n), w_1(n), w_2(n), \cdots, w_N(n)]$, and the input vector is $\mathbf{u}^t(n) = [1, u(n), u(n-1), \cdots, u(n-N+1)]$.

Here, only real input data, real weights and real desired output data are considered, and the criterion function is based on the mean squared error,

$$J(n) = E[(d(n) - \mathbf{w}^t(n)\mathbf{u}(n))^2]. \tag{10}$$

If the tap input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are jointly stationary, then the mean squared error $J(n)$ at time n can be written as

$$J(n) = \sigma_d^2 - \mathbf{w}^t(n)\mathbf{p} - \mathbf{p}^t\mathbf{w}(n) + \mathbf{w}^t(n)\mathbf{R}\mathbf{w}(n), \tag{11}$$

where σ_d^2 is the variance of the desired response $d(n)$, $\mathbf{p}$ is the cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$, $\mathbf{R}$ is the autocorrelation matrix of the tap-input vector $\mathbf{u}(n)$,

$$\begin{aligned} \mathbf{p} &= E[\mathbf{u}(n)d(n)], \\ \mathbf{R} &= E[\mathbf{u}(n)\mathbf{u}^t(n)]. \end{aligned} \tag{12}$$

The gradient of the criterion function denoted by ∇ is simply the derivative of the mean-squared error J with respect to the tap-weight vector $\mathbf{w}$:

$$\nabla = \frac{dJ(n)}{d\mathbf{w}} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n). \tag{13}$$

An optimal weight vector such that $J(n)$ is minimized can be obtained by letting $J = 0$.

The simplest choice of estimators $\mathbf{R}$ and $\mathbf{p}$ is to use the instantaneous estimates that are based on sample values of the tap-input and desired response, as defined by

$$\begin{aligned} \mathbf{R} &= \mathbf{u}(n)\mathbf{u}^t(n), \\ \mathbf{p} &= \mathbf{u}(n)d(n), \end{aligned} \tag{14}$$

respectively. Correspondingly, the instantaneous estimate of the gradient vector is

$$\nabla(n) = -2\mathbf{u}(n)d(n) + 2\mathbf{u}(n)\mathbf{u}^t(n)\mathbf{w}(n). \tag{15}$$

According to the method of steepest descent [1], the updated value of the tap-weight vector at time $n+1$ is computed by using the simple recursive relation

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla(n)], \tag{16}$$

where μ is a positive real-valued constant.

Substituting Equation (13) into Equation (16), the following updating rule is obtained,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)], \tag{17}$$

and substituting Equation (14) into Equation (17), the following LMS learning rule is obtained:

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\mathbf{u}(n)[d(n) - \mathbf{u}^t(n)\mathbf{w}(n)] \\ &= \mathbf{w}(n) + \mu\mathbf{u}(n)e_L(n), \end{aligned} \tag{18}$$

where

$$e_L(n) = d(n) - y(n),$$
$$y(n) = \mathbf{u}^t(n)\mathbf{w}(n). \tag{19}$$

Based on the concept of linear discriminant function, the hardlimiting threshold level should be chosen as follows:

$$y_o = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{if } y \leq 0. \end{cases} \tag{20}$$

The single layer neuron can be used with both continuous valued and binary inputs. This simple net generates much interest when initially developed because of its ability to learn to recognize simple patterns. For a linear separable case, LMS learning is good enough to classify the input samples. To classify non-linearly separable samples, it is necessary to use multi-layer networks such as multi-layer neurons [11]. On the other hand, the weight vector obtained by the LMS rule [12] cannot be unreasonable if the samples are not separable. LMS learning rule can be generalized to a more general rule by replacing the hardlimiting function by a sigmoid function [9].

CONFIGURING STACK FILTERS BY THE LMS LEARNING

Denote $s(n)$ as the original signal sequence, $\eta(n)$ as the noise process, and $r(n)$ as the resulting sequence. It is assumed that $s(n)$ is corrupted by additive noise process, $\eta(n)$, and thus the resulting sequence $r(n) = s(n) + \eta(n)$, as shown in Fig. 2.

The problem addressed here is to configure a stack filter S in order to recover the original signal sequence from the corrupted sequence. Since stack filters possess the threshold decomposition and stacking properties, configuring a stack filter is equivalent to first converting the input signal sequence into sequence of binary signals by threshold decomposition, and then finding the appropriate positive Boolean function used for all levels. Now, the input signal sequence is $r(n)$. Assume $r(n)$ is an M-valued sequence, by threshold decomposition the thresholded binary sequences denoted by $\vec{T}^{M-1}, \vec{T}^{M-2}, \cdots, \vec{T}^2, \vec{T}^1$, are obtained where

$$\vec{T}^1 \geq \vec{T}^2 \geq \cdots \geq \vec{T}^{M-2} \geq \vec{T}^{M-1}, \quad \text{and} \tag{21}$$

$$\vec{T}^j(n) = \begin{cases} 1 & \text{if } r(n) \geq j, \\ -1 & \text{if } r(n) < j. \end{cases} \tag{22}$$

Note that 1 and -1 rather than 1 and 0 are adopted here.

Let N be the window width of the stack filter. At each threshold level, the input sequence is a binary sequence, and the output is a binary number. Thus, the input-output relationship can be realized by a Boolean function. Recalled from the previous section, some binary Boolean functions can be realized by a linear discriminant function, and thus can be trained by the LMS learning rule discussed in the previous section. However, the Boolean function obtained by training the single neuron may not be a positive Boolean function. Heuristics which will be discussed later are introduced to ensure that the resulting Boolean function is a positive Boolean function.

The general single-neuron structure for configuring stack filters is shown in Fig. 3. The input sequence $r(n)$ is first converted to thresholded binary sequence $\vec{T}^1, \vec{T}^2, \cdots, \vec{T}^{M-1}$. For each window sample of width N of the input sequence $r(n)$,

there are (M-1) window samples of width N of the thresholded binary sequences; that is, (M-1) binary input patterns are presented to the single-neuron. Thus, the weights of the neuron are updated by the (M-1) binary input patterns (M-1) times for each window sample of $r(n)$. The serial binary outputs of the neuron are then stacked back into (M-1) levels. Finally, the M-valued filtered output signal is reconstructed, by the stacking property, from the binary outputs by a search for level at which a transition from 1 to -1 occurs.

Denote $\vec{r}_i$ as the ith window sample of width N of the input sequence $r(n)$, and $\vec{T}_i^1, \vec{T}_i^2, \cdots, \vec{T}_i^{M-1}$ as the (M-1) thresholded binary input patterns that result from the ith input sample $\vec{r}_i$. These parallel (M-1) thresholded binary input patterns are transformed into a sequence of binary patterns as follows:

$$\vec{u}_j = \vec{T}_i^k \quad \text{where } j = (M-1)(i-1) + k. \tag{23}$$

The weights of the neuron are then updated by a learning rule. Using Equation (18) and (19), we have

$$\mathbf{w}_{j+1} = max\{\mathbf{w}_j + \mu \vec{u}_j [d_j - \mathbf{w}_j^t \vec{u}_j]/(N+1), 0\}, \tag{24}$$

where

$$j = (M-1)(i-1) + k.$$

The max operator is to ensure nonnegative weights. Since the elements of $\vec{u}_j$ are either 1 or -1, $\frac{\vec{u}_j}{(N+1)} = \frac{\vec{u}_j}{|\vec{u}_j|^2}$ becomes the normalized input pattern.

Note that during training, negative weights except w_0 are set to zero. These heuristics are introduced in order to preserve the stacking property of a stack filter. After training, the final weight vector is used for the remaining inputs. It is easy to show that the above heuristics preserve the stacking property.

Denote $\mathbf{w}^k$ as the weight vector used for the kth thresholded level signal. The kth level output is:

$$y_o^k = f_H(y^k),$$

where $y^k = (\vec{T}^k)^t \mathbf{w}^k$, and $k = M-1, M-2, \cdots, 1$. Since

$$\vec{T}^{M-1} \leq \vec{T}^{M-2} \leq \cdots \leq \vec{T}^1,$$

and

$$\mathbf{w}^{(M-1)} = \mathbf{w}^{(M-2)} = \cdots = \mathbf{w}^{(1)} \geq \mathbf{0},$$

thus,

$$y_o^{(M-1)} \leq y_o^{(M-2)} \leq \cdots \leq y_o^{(1)}.$$

Hence, the stacking property is preserved.

The LMS method to configure stack filters has been developed and discussed in details above. The effectiveness of the proposed algorithm for noise suppression shall be demonstrated by experimental results. Various types of signals and noise have been considered, but, in this paper, only results for a signal, "Mexican hat," and the ϵ-mixture of Gaussian noise will be presented.

The ϵ-mixture of Gaussian noise has the following probability density function:

$$P(y) = (1-\epsilon)\Phi(\frac{y}{\sigma_1}) + \epsilon\Phi(\frac{y}{\sigma_2}),$$

where $\Phi(y)$ is the probability density function of a Gaussian random variable with zero mean and unit variance. The ϵ-mixture of Gaussian noise is suitable for representing deviations of sample values from a single distribution. With probability

$(1-\epsilon)$, the added noise for a sample, $\eta(n)$, is a Gaussian random variable with standard deviation σ_1, which represents the background thermal noise; with probability ϵ, $\eta(n)$ is a Gaussian random variable with standard deviation σ_2, which represents the impulsive noise. In the simulations, the ϵ-mixture of Gaussian noise is generated with $\epsilon = 0.8$, $\sigma_1 = 1$ and $\sigma_2 = 10$.

The "Mexican hat" signal and the ϵ-mixture of Gaussian noise are shown separately in Fig. 4. The filtered output signals obtained by the LMS learning rule with various window widths are shown in Fig. 5-7. In these figures, the first five hundred samples are the results obtained during the training, and the remaining samples are the results after having the trained weights fixed. Fig. 8 shows the mean absolute error between the desired output and the filtered output by the LMS learning rule using various window widths.

From these results, we can draw the following conclusions:

(1) The noise suppression depends on the signal, noise and filter window width.

(2) Filters with larger window widths perform better than those with smaller window widths.

CONCLUSIONS

A framework for configuring stack filters using the LMS learning rule was established and tested. We have demonstrated through experimental results that the proposed algorithm performs the noise suppression task reasonably well. Other learning rules can be used instead of the LMS algorithm. The current design only makes use of a simple single neuron. Further improvement is expected if a multi-layer network (multi-layer neurons) is employed.

Future research efforts include:
(1) Extend a single-neuron structure to a multi-layer neural network.
(2) To speed up the training by employing a new recently proposed stack filtering architecture [13].

ACKNOWLEDGEMENTS

The authors thank C. H. Chu for the "Mexican Hat" signal data.

References

[1] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1986.

[2] A. C. Bovik, T. S. Huang and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1342-1349, Dec. 1983.

[3] J. P. Fitch, E. J. Coyle and N. C. Gallagher, "Root properties and convergence rates of median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 230-239, Feb. 1985.

[4] J. P. Fitch, E. J. Coyle and N. C. Gallagher, "Threshold decomposition of multidimensional ranked-order Operations," *IEEE Trans. Circuits Syst.*, vol CAS-32, pp. 445-450, May 1985.

[5] P. D. Wendt, E. J. Coyle and N. C. Gallagher, "Stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 898-911, Aug. 1986.

[6] E. N. Gilbert, "Lattice-theoretic properties of frontal switching functions," *J. Math. Phys.*, vol. 33, pp. 57-67, Apr. 1954.

[7] C. H. Chu, "A genetic algorithm approach to the configuration of stack filters," in *Proc. Intl. Conf. on Genetic Algorithms*, George Mason University, June 4-7, 1989, pp. 218-224.

[8] J. H. Lin, T. M. Selike and E. J. Coyle, "Adaptive stack filtering under the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 938-954, Jun. 1990.

[9] S. C. Douglas and T. H. Y. Meng, "Optimum error nonlinearities for LMS adaptation," in *Proc. ICASSP 90*, Albuquerque, New Mexico, April 3-6, 1990.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Menlo Park, CA: John Wiley & Sons. Inc., 1973.

[11] B. Widrow, R. G. Winter and R. A. Baxter, "Layered neural nets for pattern recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1109-1118, Jul. 1988.

[12] R. Rosenblatt, *Principles of Neurodynamics*, New York: Spartan Books, 1959.

[13] J. H. Lin, "A new architecture and fast algorithm for stack filters," presented at *1991 CISS Conference at John Hopkins University*, March 20, 1991.

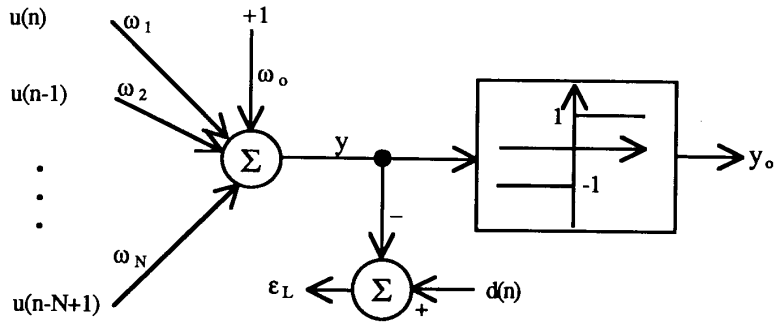

Fig. 1. A neuron: An adaptive threshold logic element — the adaptive filter structure used by the LMS rule.

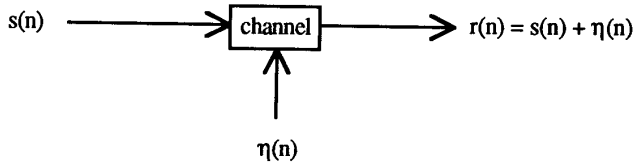

Fig. 2. Additive noise channel.

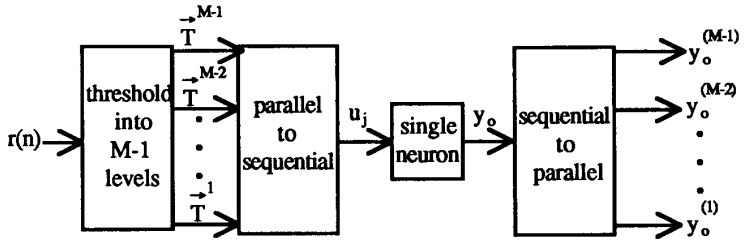

Fig. 3. Single neuron structure for training.

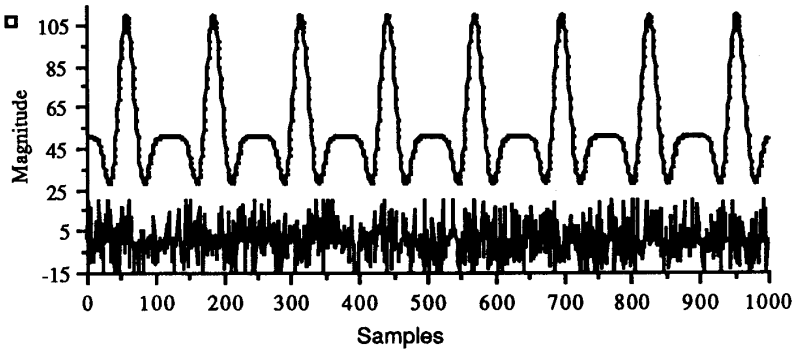

Fig. 4. Original signal ("Mexican hat") and ε-mixture of Gaussian noise shown separately.

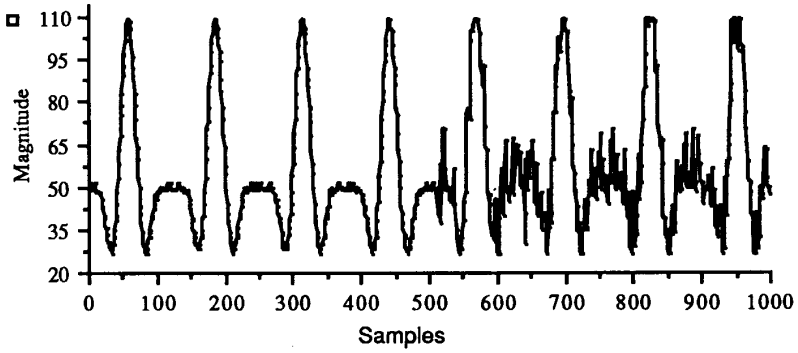

Fig. 5. Output signal obtained by filtering the corrupted signal using the LMS rule with window width=3.

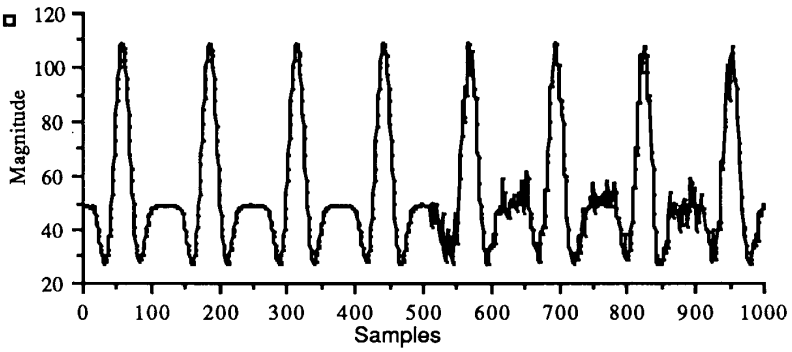

Fig. 6. Output signal obtained by filtering the corrupted signal using the LMS rule with window width=9.

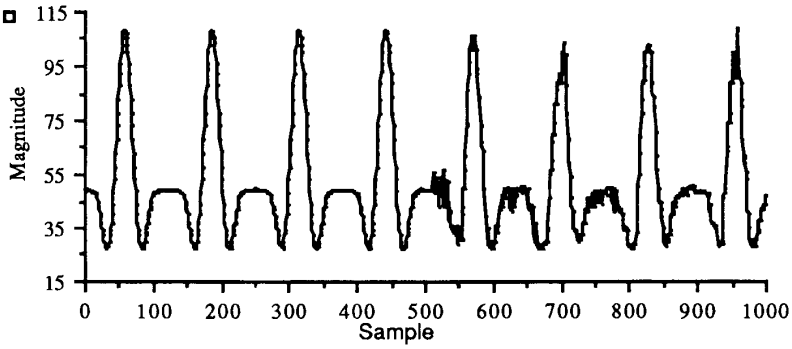

Fig. 7. Output signal obtained by filtering the corrupted signal using the LMS rule with window width=15.

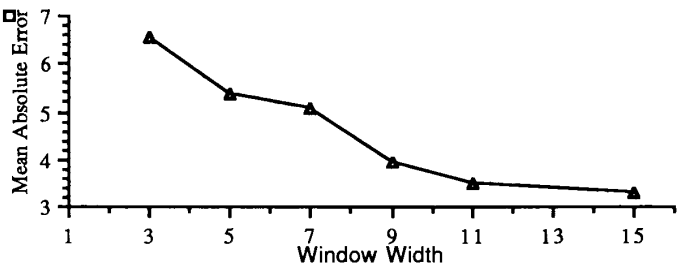

Fig. 8. Mean absolute errors between the original "Mexican hat" signal and the output signals obtained by the LMS rule using various window widths..