DESIGN AND IMPLEMENTATION OF NOISE FREE AUDIO SPEECH SIGNAL USING FAST BLOCK LEAST MEAN SQUARE ALGORITHM

J. Jebastine [1], Dr. B. Sheela Rani [2]

[1]Department of Electronics and Communication, Research Scholar,
Sathyabama University, Chennai-600 119.
[2]Vice-Chancellor & Dean, PG Studies and Research,
Sathyabama University, Chennai-600 119.
enochjeba@gmail.com

ABSTRACT

This paper describes the development of an adaptive noise cancellation algorithm for effective recognition of speech signal and also to improve SNR for an adaptive step size input. An adaptive filter with Fast Block Least Mean square Algorithm is designed for noise free audio (speech/music) signals. The signal input used is a audio speech signal which could be in the form of a recorded voice. The filter used is adaptive filter and the algorithm used is Fast Block LMS algorithm. A Gaussian noise is added to this input signal and given as a input to the Fast Block LMS. The algorithm is implemented in Matlab and was tested for noise cancellation in speech signals. A Simulink model is designed which results in a noise free audio speech signal at the output. The FBLMS algorithm is computationally efficient in noise cancellation. The noise level in speech signal can be 1) mild, 2) moderate, 3) severe. The SNR is estimated by varying the adaptive step size.

KEY WORDS
Noise cancellation, FBLMS Adaptive Algorithm, Simulink Model, SNR.

1. INTRODUCTION

The purpose of filtering, which is a signal processing operation, is to manipulate the information present in the signal. Filter is a device which extracts desired information from the input signal by mapping input signal to another output signal. The digital format of discrete time signal can be processed by digital filter. With these filters desired spectral characteristics of signal can be achieved by which unwanted signal can be rejected, like noise or interference and bit rate can be reduced in transmission. For time-invariant filters the internal parameters and the structure of the filter are fixed and if the filter is linear the output signal is a linear function of the input signal. The problem which is not known in advance can be tackled by making the filter adaptive i.e. changing the parameters of filter according to some algorithm.e.g. The characteristics of the signal, or the unwanted signal, or systems influence on the signal that one would like to compensate. The Adaptive filters are capable of adjusting to unknown environment and even track signal or system characteristics varies in time. three steps are in designing time invariant linear filter, namely specification are approximated using rational transfer function, choice of defining the algorithm and choice of implementation form for the algorithm. An adaptive filter is required when either the fixed specifications are unknown or the specifications cannot be satisfied by time-invariant filters. In actual fact an adaptive filter is a nonlinear filter because its

DOI : 10.5121/sipij.2012.3304

characteristics dependents on the input signal, consequently the homogeneity and additive conditions are not satisfied. In general, adaptive filters are one that varies through time because the characteristics of its inputs may be varying. That is why it separates itself from classical digital signal processing (ie) the digital system itself changes through time. When there is a need to process signal in an environment of unknown statistics adaptive filters will not perform better than fixed filter. This paper describes adaptive filtering and its simulation in MATLAB with Fast Block LMS algorithm for noise cancellation in speech signals.

2. ADAPTIVE FILTER

An adaptive filter is a filter that self-adjusts its transfer function according to an optimizing algorithm. Because of the complexity of the optimizing algorithms, most adaptive filters are digital filters that perform digital signal processing and adapt their performance based on the input signal. In contrast, a non-adaptive filter has static filter coefficients, which together form the transfer function .Since for the desired processing operation some parameters (for instance, the properties of some noise signal) are not known in advance for some application adaptive coefficients are required. In this case it is common to use an adaptive filter, which uses feedback to modify the values of the filter coefficients and thus its frequency response. The adaptive process involves the use of a cost function, which is a criterion for optimum performance of the filter (for example, minimizing the noise component of the input), to feed an algorithm, which determines how to modify the filter coefficients to minimize the cost on the next iteration.

2.1 Specification of an Adaptive System

The complete specification of an adaptive system consists of three items:

2.1.1. Application

The choice of the signals acquired from the environment to be the input and desired-output signals defines the type of application. During the last two decades the applications which successfully use adaptive techniques are increased enormously. Some examples are echo cancellation, equalization of dispersive channels, system identification, signal enhancement, adaptive beam forming, noise cancelling, and control. This paper explores about noise cancellation.

2.1.2. Adaptive-Filter Structure

The computational complexity of the process and also the necessary number of iterations to achieve a desired performance level are being influenced by structure choice. Basically, there are two major classes of adaptive digital filter realizations, distinguished by the form of the impulse response, namely the finite-duration impulse response (FIR) filter and the infinite-duration impulse response (IIR) filters. FIR filters are usually implemented with non-recursive structures, whereas IIR filters utilize recursive realizations.

2.1.2.1. Adaptive FIR filters realizations

The adaptive FIR filter structure which is most widely used is the transversal filter, also called tapped delay line, that implements an all-zero transfer function with a canonical direct form realization without feedback. For this realization, the output signal y(k) is a linear combination of the filter coefficients, that yields a quadratic mean-square error (MSE = E[|e(k)|2]) function with a unique optimal solution. Other alternative adaptive FIR realizations are also used in order to obtain improvements as compared to the transversal filter structure, in terms of computational complexity, speed of convergence, and finite word length properties.

2.1.2.2. Adaptive IIR filter realizations

The most widely used realization of adaptive IIR filters is the canonical direct form realization, due to its simple implementation and analysis. However, there are some inherent problems related to recursive adaptive filters which are structure dependent, such as pole-stability monitoring requirement and slow speed of convergence. To represent these problem varies realizations were proposed attempting to overcome the limitations of the direct form structure.

Among these alternative structures, the cascade, the lattice, and the parallel realizations are considered because of their unique features

2.1.2.3. Algorithm

The procedure for the algorithm is to adjust the filter coefficients for the adaptive filter in order to reduce a prescribed criterion. The algorithm is determined by minimization algorithm, the objective function, and the nature of error signal. Various type algorithms determine several crucial aspects of the overall adaptive process, such as existence of sub-optimal solutions, biased optimal solution, and computational complexity.

2.2. A Review of LMS Algorithm

The LMS algorithm which uses an instantaneous estimate of the gradient vector of a cost function is an approximation of the steepest descent algorithm. Based on sample values of the tap-input vector and an error signal the gradient is estimated. The algorithm iterates each coefficient in the filter, moving it in the direction of the approximated gradient. For the LMS algorithm it is necessary to have a reference signal d[n] representing the desired filter output. The difference between the reference signal and the actual output of the transversal filter is the error signal which is given in the equation (1)

$$e(n) = d(n) - c^{H}[n]x[n] \tag{1}$$

The objective of the LMS algorithm is to find a set of filter coefficients c(n) to achieve the least mean squared error that minimizes the expected value of the quadratic error signal. The squared finite impulse response error and its expected value are (for simplicity of notation and perception) drop the dependence of all variables on time n .A schematic of the learning setup is depicted in figure (1)

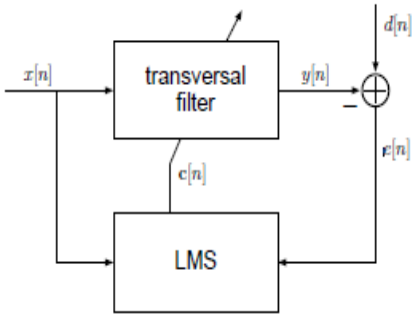

Figure (1) General Block diagram for Adaptive Filter with LMS

From Figure,

$$e^2 = (d - c^H x)^2 = d^2 - 2dc^H x + c^H x x^H c \tag{2}$$

$$E[e^2] = E(d)^2 - E(2dc^H x) + E(c^H xx^H c) \tag{3}$$

$$E[e^2] = E(d)^2 - c^H 2E(dx) + c^H E(xx^H)c \tag{4}$$

Note, that the squared error e2 is a quadratic function of the coefficient vector c, and thus has only one (global) minimum (and no other (local) minima, ref equation (4).

The gradient descent approach demands that the position on the error surface , ref equation(5) according to the current coefficients should be moved into the direction of the 'steepest descent', i.e., in the direction of the negative gradient of the cost function J = E(e2) with respect to the coefficient vector

$$-\nabla j = E[dx] - 2E(xx^H)c \tag{5}$$

The expected values in this equation, E(d x) = p, the cross-correlation vector between the desired output signal and the tap-input vector, and E(x xH) = R, the auto-correlation matrix of the tap-input vector, would usually be estimated using a large number of samples from d and x. In the LMS algorithm, however, a very short-term estimate is used by only taking into account the current samples: E(d x) ≈ d x, and E(x xH) ≈ x xH , leading to an update equation for the filter coefficients

$$C^{new} = C^{old} + \mu / 2(-\nabla_c j(c)) \tag{6}$$

$$C^{new} = C^{old} + \mu x(d - x^H c) \tag{7}$$

$$C^{new} = C^{old} + \mu x e^* \tag{8}$$

Here, we introduced the 'step-size' parameter μ, ref equation (6) & (7) which controls the distance we move along the error surface. In the LMS algorithm the update of the coefficients, is performed at every time instant n,

$$c[n+1] = c[n] + \mu e^*[n]x[n] \tag{9}$$

2.2.1.Choice of step-size

The error function surface at each update step that moves along is controlled by' step-size' parameter μ.μ certainly has to be chosen such that μ > 0 (otherwise we would move the coefficient vector in a direction towards larger squared error). Also, μ should not be too large, since in the LMS algorithm we use a local approximation of p and R in the computation of the gradient of the cost function, and thus the cost function at each time instant may differ from an accurate global cost function. Moreover, larger the step-size causes the LMS algorithm to be instable, i.e., the coefficients do oscillate instead of converge to fixed values. Analyzing closely, for stable behavior of the LMS algorithm depends on the largest eigenvalue λmax of the tap-input auto-correlation matrix R and thus on the input signal. For stable adaptation behavior the step-size has to be

$$\mu < \frac{2}{\lambda_{\max}} \tag{10}$$

Since we still do not want to compute an estimate of R and its eigen values, we first approximate $\lambda_{\max} \approx tr(R)tr(R)$ is the trace of matrix R, i.e., the sum of the elements on its diagonal), and then – in the same way as we approximated the expected values in the cost function , $-tr(R) \approx \|x(n)\|^2$ the tap-input power at the current time n. Hence, the upper bound for μ for stable behavior depends on the signal power.

3. FAST BLOCK LMS ALGORITHM

For some application like adaptive echo cancellation and adaptive noise cancellation needs adaptive filters with a large filter length. LMS algorithm applied to the adaptive filter might take a long time to complete the filtering and coefficients updating process. This may cause problems in these applications because the adaptive filter must work in real time to filter the input signals. In such situation, one can use the fast block LMS algorithm.

The Fast block LMS algorithm transform the input signal x(n) to the frequency domain using the fast Fourier transform (FFT). It also updates the filter coefficients in the frequency domain. This updates can save computational resources. The fast block LMS algorithm differs from the standard LMS algorithm in the following ways: The size of block is exactly the same as the filter length. The filter coefficients are updated by sample by sample basis in the standard LMS algorithm.

The multiplication operation needed for the fast block LMS algorithm is much lesser than the standard LMS algorithm. If both the filter length and block size are N, the standard LMS algorithm requires N(2N+1) multiplications, whereas the fast block LMS algorithm requires only (10Nlog2N+26N) multiplications. The fast block LMS algorithm can execute 16 times faster than the standard LMS algorithm, when N=1024.

3.1 Circular Matrices

Consider the MxM circular matrix Ac as shown in equation (11)

$$A_c = \begin{bmatrix} a_0 & a_{M-1} & \cdots & a_1 \\ a_1 & a_0 & \cdots & a_2 \\ \cdot & \cdot & & \cdot \\ & & \cdots & \\ \cdot & \cdot & & \cdot \\ a_{M-1} & a_{M-2} & \cdots & a_0 \end{bmatrix} \tag{11}$$

Each row (column) in Ac is determined by shifting the previous row (column) circularly by one element. The circular matrix has an important property according to which such matrices are diagonalised by DFT matrices. That is, if F is the M xM DFT matrix defined as F given in equation (12)

$$F=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{-j2\pi}{M}} & \cdots & e^{\frac{-j2\pi(M-1)}{M}} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 1 & e^{\frac{-j2\pi(M-1)}{M}} & \cdots & e^{\frac{-j2\pi(M-1)^2}{M}} \end{bmatrix} \quad (12)$$

Then $A_F = FA_cF^{-1}$ is a diagonal matrix. Furthermore, the diagonal element of AF correspond to the DFT of the first column of Ac, i.e. $A_F = diag[\overline{a_F}]$, $[\overline{a_F}] = F\overline{a}$ Where, $\overline{a} = [a_0 \quad a_1 \quad .... \quad a_{M-1}]^T$.

3.2. Window Matrices and Matrix Formulation of the Overlap-Save Method

Define the N'xN' circular matrix, for N'=L+N-1, as Xc(k)

$$X_{c(K)=}\begin{bmatrix} x(kL-N+1) & x(kL-N-1) & \cdots & x(kL-N+2) \\ x(kL-N+2) & x(kL-N+1) & \cdots & x(kL-N+3) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ x(kL-N-1) & x(kL-N-2) & \cdots & x(kL-N+1) \end{bmatrix} \quad (13)$$

$$\tilde{y}(k) = \begin{bmatrix} 0 \\ \hline y(k) \end{bmatrix}$$

Also, define the length N' column vector

Where, $y(k) = [y(kL) \quad y(kL+1) \quad ..... \quad y(kL-1)]^T$

$\overline{0}$ is the length of N-1 zero vector. Let denote by $\mathbf{y_c(k)}$ the column vector that appears on the left-hand side of equation (14). That is

$$y_c(k) = [** \quad .... \quad ** y(kL) \quad y(kL+1) \quad ..... \quad y(kL-1)]^T \quad (14)$$

We can see that $\tilde{\mathbf{y}}(\mathbf{k})$ can be obtained from $\overline{\mathbf{y}}_\mathbf{c}(\mathbf{k})$ with zero. This substitution can be written in the form of a matrix-vector product as

$$\tilde{y}(k) = P_{o,L}\tilde{y}_c(k) \quad (15)$$

where $\mathbf{P_{0,L}}$ is the N'xN' windowing matrix defined as

$$P_{o,L} = \begin{bmatrix} \overline{0} & \overline{0} \\ \hline \overline{0} & I_L \end{bmatrix} \quad (16)$$

with $\overline{\mathbf{I}}_\mathbf{L}$being the LxL identity matrix and $\overline{\mathbf{0}}\mathbf{s}$ are zero matrix with appropriate dimensions. Thus, we obtain

$$\tilde{y}(k) = P_{o,L} X_c(k) \overline{W}(k) \tag{17}$$

The above equation (17) may be written as

$$\tilde{y}(k) = P_{o,L} F^{-1} F X_c(k) F^{-1} F \overline{W}(k) \tag{18}$$

where F is the N'xN' DFT matrix .Next, define

$$\overline{W}(k) = F\tilde{W}(K) \tag{19}$$

$$X_F(k) = F X_c F^{-1} \tag{20}$$

Since $X_C(k)$ is a circular matrix, $\mathbf{X_F(k)}$ is the diagonal matrix having elements of DFT of the first column of $X_C(k)$).The first column of XC(k) is the input vector $\mathbf{\tilde{x}(k)}$.Thus obtain equation (21)

$$\tilde{y}(k) = P_{o,L} F^{-1} X_F(k) W_F(k) \tag{21}$$

3.3. FBLMS Algorithm

The Fast BLMS (FBLMS) algorithm is a computationally efficient implementation of the BLMS algorithm in the frequency domain, referring to equation (21), we write

$$\tilde{y}(k) = P_{o,L} F^{-1} X_F(k) W_F(k)$$

With respect to the filtering part of the FBLMS algorithm

The output vector $\tilde{y}(k)$, in extended form is defined as

$$\tilde{y}(k) = \begin{bmatrix} 0 \\ \_ \\ y(k) \end{bmatrix} \tag{22}$$

The extended error vector is $\tilde{e}(k) = \tilde{d}(k) - \tilde{y}(k)$ .To obtain the frequency domain equivalent

$$\overline{W}(k+1) = \overline{W}(k) + 2\frac{\mu_B}{L} X^T(k) \overline{e(k)} \tag{23}$$

Then , replace $\mathbf{W(k)}$ and $\mathbf{\bar{e}(k)}$ by their extended version, where Xc(k) is the circular matrix of samples of the filter input and, $\mu = \frac{\mu_B}{L}$ and $P_{N.O} = \begin{bmatrix} I_N & 0 \\ 0 & 0 \end{bmatrix}$

From $P_{N.O}$ which is an N'xN' windowing matrix it shows that even after each iteration the last L-1 elements of the updated weight vector $\tilde{W}(k+1) P_{N.O} \tilde{W}(k+1)$ remain equal to zero. The frequency domain equivalent can be done by premultiplying it on both sides by the DFT matrix F and using the identity $\mathbf{F^{-1}F = I}$ to obtain

$$\tilde{W}(k+1) = \tilde{W}(k) + 2\mu P_{N,O} X_c^T(k) \tilde{e}(k) \tag{24}$$

and then we can write the weight-vector updating equation as

$$\tilde{W}(k+1) = \tilde{W}(k) + 2\mu P_{N,O} X_F^{*T}(k) e_F(k) \quad (25)$$

Where $e_F(k) = F\tilde{e}(k)$

and $P_{N.O} = FP_{N,O}F^{-1}$

The block diagram of the FBLMS algorithm is shown in figure (2).

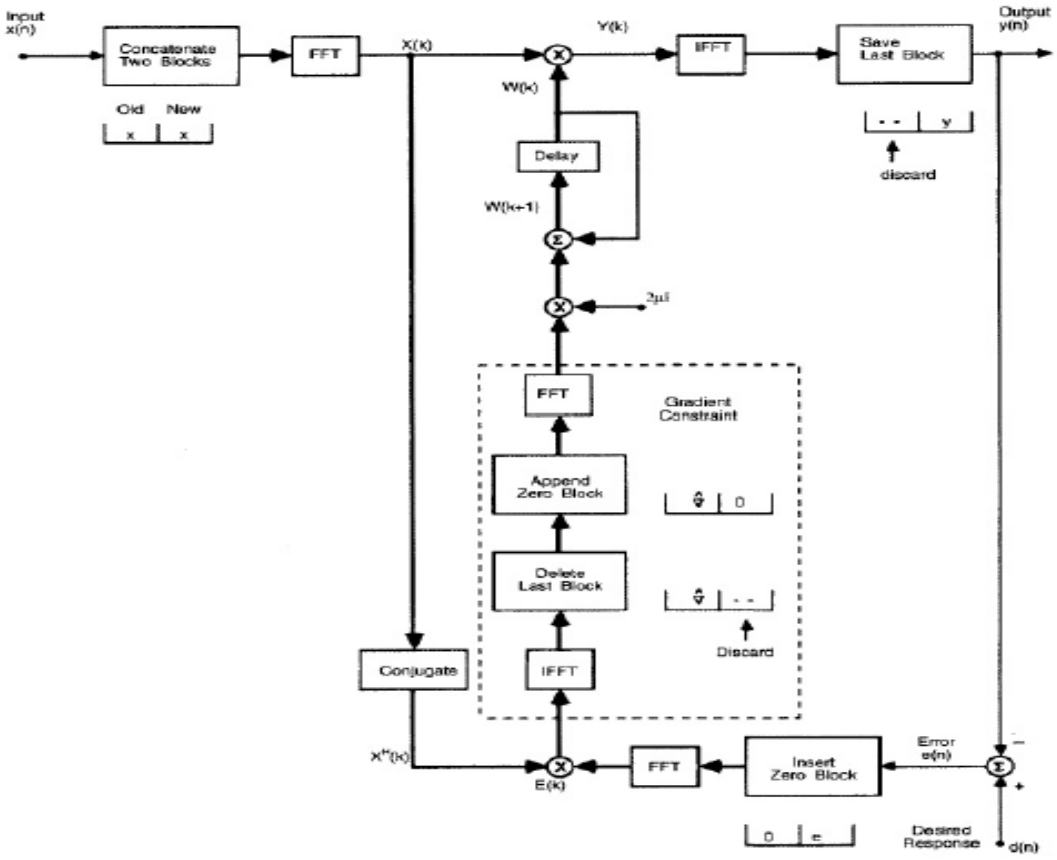

Figure (2) Implementation of FBLMS Algorithm

3.3.1. Constrained and Unconstrained FBLMS Algorithms

Already shown that for fairly mild conditions the FBLMS algorithm can work well even when the tap-weight constraining matrix PN,0 is dropped from equation (24). That is, the recursion

$$\tilde{W}(k+1) = \tilde{W}(k) + 2\mu X_F^{*} \bar{e}_F(k) \quad (26)$$

When N is chosen sufficiently large that converges the same set of tap-weights and the input process, x(n), does not satisfy some specific (unlikely to happen in practice) conditions. If the gradient constraining operation, enclosed by the dotted-line box, is dropped constrained algorithm is easily converted to the unconstrained FBLMS algorithm.

3.3.2. Convergence Behaviour of the FBLMS Algorithm

Consider the unconstrained recursion,

$$\tilde{e}(k) = \tilde{d}(k) - P_{O,L} F^{-1} X_F(k) \vec{W}(k) \tag{27}$$

Since the first N-1 elements of $\vec{d}(k)$ are all zero, thus

$$\tilde{W}(k+1) = \tilde{W}(k) + 2\mu X_F^* \bar{e}_F(k) \tag{28}$$

$$\overline{W}_F(k+1) = \overline{W}_F(k) + 2\mu X_F^*(k)[P_{O,L}(\bar{d}_F(k) - X_F(k))W_F(k)] \tag{29}$$

Next, we define the tap-weight error vector

$$\bar{v}_F(k) = \overline{W}_F(k) - \overline{W}_{O,F} \tag{30}$$

Where $\overline{W}_{0,F}$, is the optimum value of the filter tap-weight vector in the frequency domain. Using the equation (28), (29) & (30) we can obtain equation (31)

$$\bar{v}_F(k+1) = (1 - 2\mu X_F^*(k) P_{O,L} X_F(k)) v_F(k) + 2\mu X_F^*(k) P_{O,L} \bar{e}_{O,F}(k) \tag{31}$$

Where $\bar{e}_{O,F}(k)$ is the optimum error vector obtained when $\overline{W}_F(k)$ is replaced by $\overline{W}_{O,F}(k)$.It can be shown (omitted) that, for coloured inputs, as happens with the conventional LMS algorithm, the unconstrained FBLMS algorithm will also perform poorly. The same is true for the constrained FBLMS algorithm.

3.3.3. Step-Normalization

The convergence behaviour of the FBLMS algorithm can be greatly improved by using individually normalized step-size parameters for each element of the tap-weight vector $\overline{W}_F(k)$, rather than a common step-size parameter. The above technique is called “normalizing the step size” , that is same as one used for improving the convergence of the transform-domain LMS algorithm. The step-normalization is implemented by replacing the scalar step-size parameter μ by the diagonal matrix

$$\mu(k) = diag[\mu_0(k) \quad \mu_1(k) \quad ..... \quad \mu_{N-1}] \tag{32}$$

where μi(k) is the normalized step-size parameter for the i-th tap. These are obtained by the equations.

$\mu_i(k) = \dfrac{\mu_0}{\partial^2_{x_F,i}(k)}$ Where μ0 is a constant, $\partial^2_{x_F,i}(k)$ are the power estimates of the samples of the filter input in the frequency domain, $x_{F,i}(k)$ these estimates may be obtained in equation (33) using the following recursion

$$\partial^2_{x_{F,i}}(k) = \beta\partial^2_{x_{F,i}}(k-1) + (1-\beta)\left|x_{F,i}(k)\right|^2 \quad (33)$$

where, 0<ß<1, ß is close to 1.

3.3.4. Steps involved in FBLMS Algorithm

Step: 1.Input Tap-weight vector, $\overline{W}_F(k)$

Signal power estimates, $\partial^2_{x_{F,i}}(k)^2$

Extended input vector, $\tilde{x}(k) = [x(kL-N+1)y(kL-N+2)....x(kL+L-1)]^T$

Step: 2. Output vectors

Desired output vector, $d(k) = \left[d(kL) \quad d(kL+1) \quad ..... \quad d(kL+L-1)\right]^T$

Output: Filter output, $y(k) = \left[y(kL) \quad y(kL+1) \quad ..... \quad y(kL+L-1)\right]^T$

Tap-weight vector update, $\overline{W}_F(k+1)$

where N: filter length, L: block length

Step: 3. Filtering $\tilde{x}_F(k) = FFT(\tilde{x}(k))$

$\overline{y}(k)$ = the last elements of IFFT of $\tilde{x}_F(k) \otimes \tilde{W}_F(k)$

Here $\otimes$ denotes element - by element multiplication of vectors.

Step: 4. Error estimation: $\overline{e}(k) = \overline{d}(k) - \overline{y}(k)$

Step: 5. Step-Normalization for i=0 to N-1 , $\partial^2_{x_{F,i}}(k) = \beta\partial^2_{x_{F,i}}(k-1) + (1-\beta)\left|x_{F,i}(k)\right|^2$

$$\mu_i(k) = \frac{\mu_0}{\partial^2_{x_{F,i}}(k)} \ , \ \overline{\mu}(k) = \left[\mu_0(k) \quad \mu_1(k) \quad ..... \quad \mu_{N-1}(k)\right]^T$$

Step:6. Tap-weight adaptation: $\overline{e}_F(k) = FFT\left(\begin{bmatrix} 0 \\ \overline{e}(k) \end{bmatrix}\right)$, $\overline{W}_F(k+1) = \overline{W}_F(k) + 2\mu(k) \otimes X_F^*(k) \otimes \overline{e}_F(k)$

Step: 7. Tap-weight constraint: $\overline{W}_F(k+1) = FFT\left(\begin{bmatrix} thefirstNe & lementsof & (\overline{W}_F(K+1)) \\ & \overline{0} & \end{bmatrix}\right)$

This last step is applicable only for the constrained FBLMS algorithm The algorithm is applicable to both real- and complex-valued signals.

3.3.5. Selection of the Block Length

In general the block processing of signals, results in a certain time delay at the system output In many applications this processing delay may be intolerable and hence it has to be minimized. In this application where the processing delay is not an issue, L is usually chosen to N. The exact

value of L depends on N. For a given N, one should choose L so that N'=N+L-1 is an appropriate composite number so that efficient FFT and IFFT algorithm can be used in the realization of the FBLMS algorithm. On the other hand, in applications where it is important to keep the processing delay small, one may need to strike a compromise between system complexity and processing delay.

4. SIMULATION AND RESULTS

For the purpose of noise cancellation in speech signal corrupted by White Gaussian noise, the Fast Block LMS algorithm was simulated and tested using Matlab as shown in figure (3&4).The model is designed in the situation under which the adaptive filter needed to remove a white noise from a desired speech signal. The original speech signal is recorded audio signal as shown in figure (5). White Gaussian noise is added to the original speech signal once it is being generated. The primary input used for the filter is the linear combination of the generated noise and the original signal. Figure (6) shows the original speech corrupted by white noise with frequency response as shown below, the filtered speech signal using the Fast Block LMS algorithm is shown in figure (7).

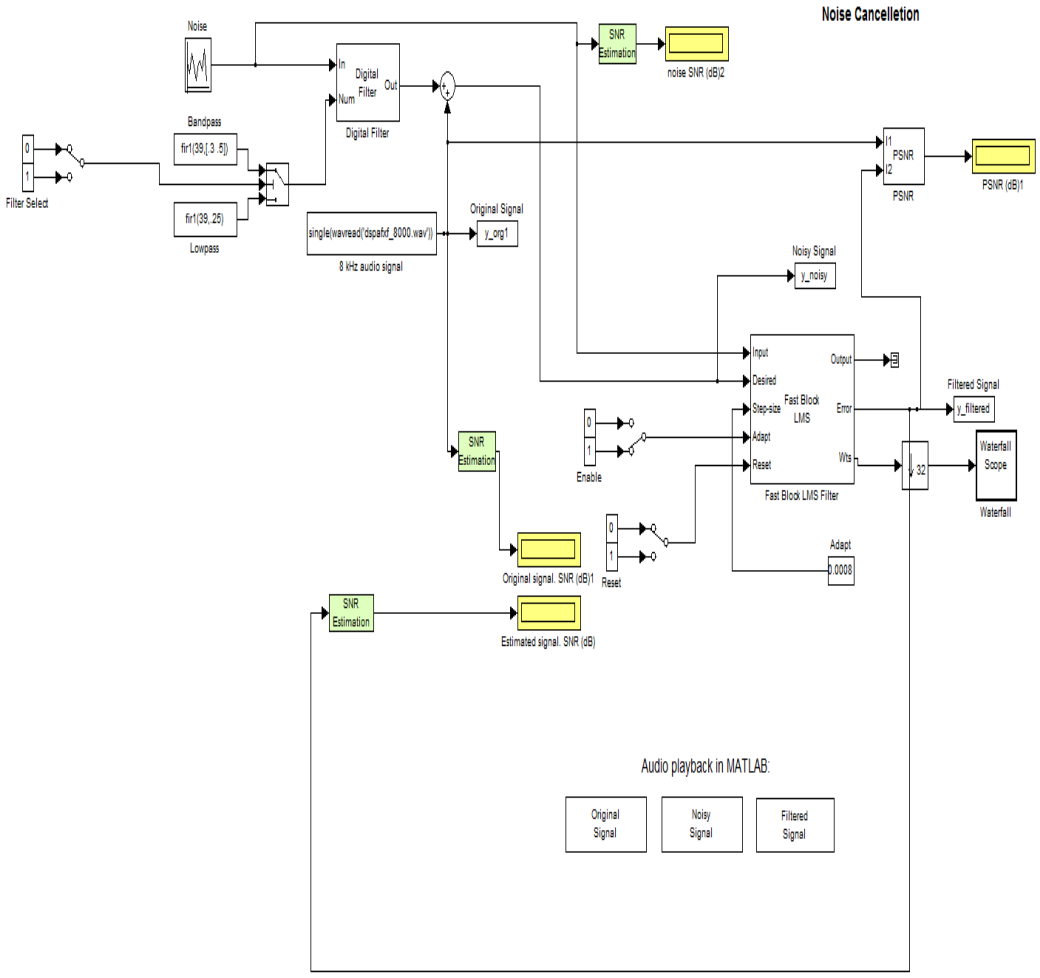

Figure (3) Simulink Model of Fast Block LMS for speech signal Processing

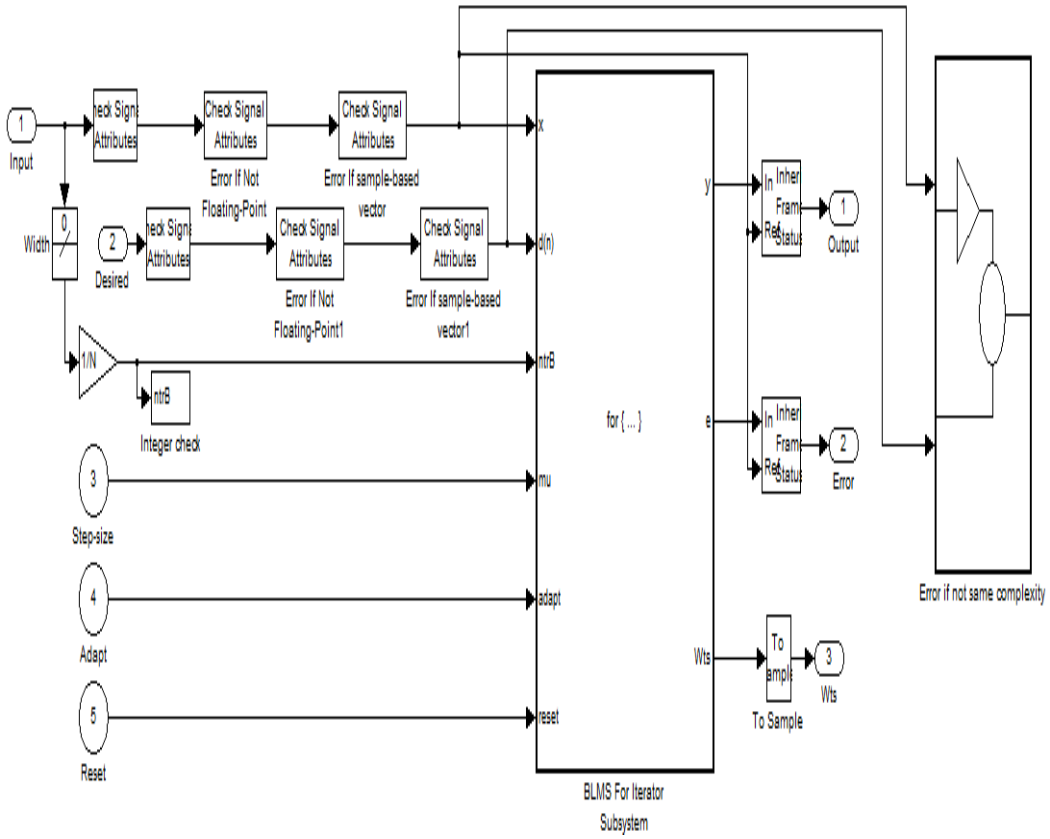

Figure (4) Internal structure of Fast Block LMS for speech signal Processing

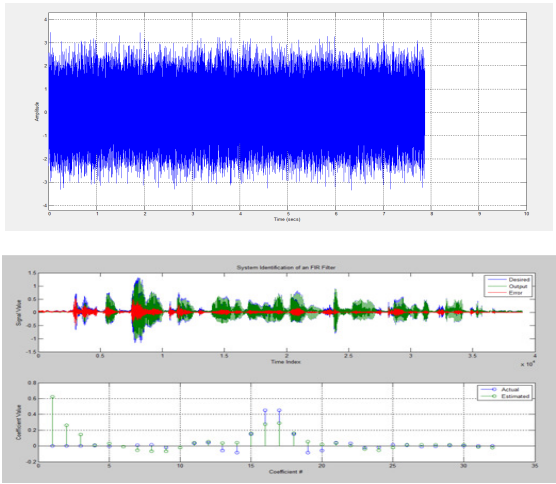

Figure (5): Original Speech signal (Recorded audio signal) and its Spectrum

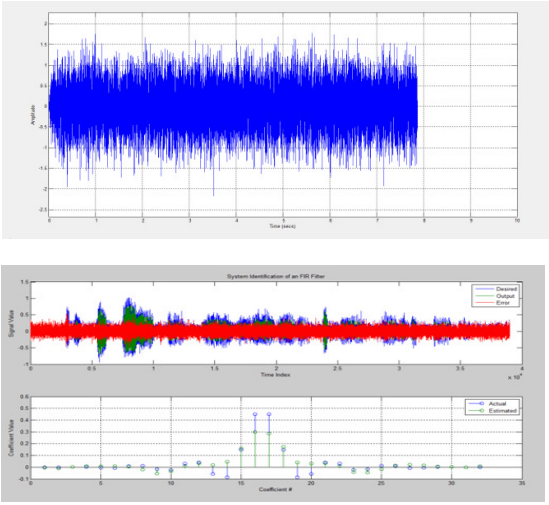

Figure (6): White Gaussian Noise Signal and its Spectrum

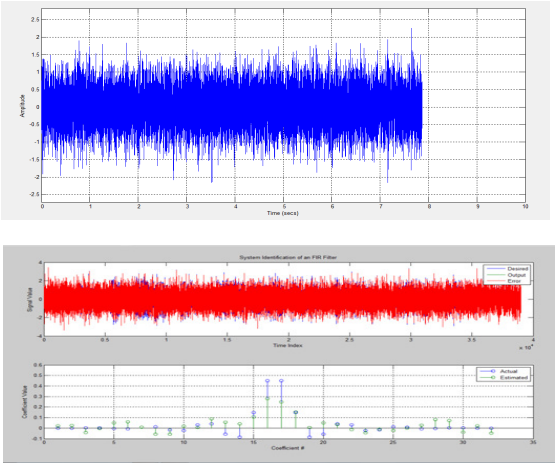

Figure (7): Filtered output Speech signal after FBLMS and its Spectrum

The step size can be varied using the adapt block. It can be enabled or disabled using boolean. The filter can also be reset using a suitable enable or disable. The signal to noise ratio output is calculated for the various step size input signal.. The peak signal to noise ratio is also calculated for the output signal. The results obtained for the various values of step size are tabulated below, Refer Table (1).

The Signal to Noise ratio for FBLMS is also estimated and tabulated, Refer Table (2).

Table (1) PSNR ratio of different input step size using FBLMS

Step Size	White Gaussian Noise (dB)	Original Input Speech Signal (dB)	Filtered Output Signal (dB)	PSNR (dB)
0.00001	0.2059	43.84	14.83	14.9
0.0001	0.2059	43.84	40.15	42.01
0.001	0.2059	43.84	40.15	42.19
0.0015	0.2059	43.84	40.33	42.52
0.0019	0.2059	43.84	40.56	42.82
0.01	0.2059	43.84	40.86	41.94
0.015	0.2059	43.84	40.66	41.1
0.017	0.2059	43.84	40.61	40.81
0.02	0.2059	43.84	40.4	40.19
0.03	0.2059	43.84	36.8	36.49

Table (2) Comparison of SNR ratio of different Noise Level

Different types of Noise Level Adaptive Algorithms	Mild Noise Level SNR (dB)	Moderate Noise Level SNR (dB)	Severe Noise Level SNR (dB)
NLMS	13	13	11
BLMS	15.5	15	9
FB-LMS	23.45	20.5	15.75

5. CONCLUSION

In this paper Fast Block LMS algorithm for speech signal is simulated and tested by using Matlab. The results from the FBLMS shows that it can remove the different levels of noise more efficiently and effectively and which may cause faster response. It has a low computational complexity property than LMS algorithm. The Peak signal to noise ratio for different step size are found and the optimum value of step size is found to be 0.0019.In Low power noise the system showed SNR improvement up to 23.15dB, for Medium noise level the system showed SNR improvement up to 20.5dBand in severe noise level the system showed SNR improvement up to 15.75dB.This techniques can be used for noise cancellation in speech signal and Bio medical signals. The future work is to be implemented in DSP processor with real time applications.

REFERENCES

[1] Yaghoub Mollaei (2009), Hardware Implementation of Adaptive Filters: Publishing: Proceedings of 2009 Student Conference on Research and Development (SCOReD 2009), 16-18 .

[2] T. S. Qiu, D. X. Wei,(2005), and Adaptive Signal Processing in Communication, Beijing: Publishing House of Electronics Industry, pp. 23-47.

[3] S. Haykin (2003), Adaptive Filter Theory, 4rd ed.,B. Y. Zheng. Trans. Publishing House of Electronics Industry, Beijing, pp. 183-230.

[4] R. H. Kwong, E. W. Johnston,(1992), "A variable step size LMS algorithm," IEEE Transactions on Signal Processing, vol. 40, No.7, pp. 1636-1642.

[5] Special Issue on Adaptive Filtering (1984), IEEE Trans. Inform. Theory, vol. IT-30.

[6] Special Issue on Adaptive Systems and Applications (1987), IEEE Trans. Circuirs Sysf., vol. CAS-34.

[7] T. Wang and C.-L. Wang,(1992), "Comments on 'A fast block FIR adaptive digital filtering algorithm with individual adaptation of parameters,' " IEEE Trans. Circuits Syst.-II: Analog and Digital Signal Processing, vol. 39, pp. 254-256.

[8] Robouts, G., and Moonen, MP (2002), 'A sparse block exact affine projection algorithm', IEEE Trans.Speech Audio Process. 2002, 10, (2), pp. 100–108.

[9] Albu, F., and Kwan, H.K, (2004), 'Combined echo and noise cancellation based on Gauss-Seidel Pseudo affine projection algorithm', Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS 2004, Vancouver, Canada, pp. 505–508.

[10] Kun Shi, Member, IEEE, and Xiaoli Ma, Senior Member, IEEE (2010), A Frequency Domain Step-Size Control Method for LMS Algorithms,IEEE SIGNAL PROCESSING LETTERS, VOL. 17, NO. 2.

[11] Y. Zhou, S. C. Chan and K. L. Ho (2004), A New Block Exact Fast LMS/ Newton Adaptive filtering Algorithm, The 47h IEEE International Midwest Symposium on Circuits and Systems, Pg.No.29-32.

[12] K. Berberidis and S. Theodoridis, (1999), "A New Fast Block Adaptive Algorithm," IEEE Trans.Signal Processing, vol. 47, No.I , pp. 75-87.

[13] G. A. Clark, S. K. Mitra, and S. R. Parker, (1981),"Block Implementation of Adaptive Digital Filters," IEEE Trans. Acoustic, Speech, Signal Processing, vol. ASSP-29, No. 3, pp. 744-752.

[14] J. G. Proakis, C. M. Rader, F. Ling, C. L. Nikias, M. Moonen, and I. K. Proudler (2002),Algorithms for Statistical signal Processing Prentice Hall Press, New Jersey.

[15] Alfred0 C. Tan,(1997), One-Dimensional Block Adaptive IIR Filters, IEEE ASSP Magazine Pg.No:742 -744.

Author

J.JEBASTINE graduated in Madras University, Chennai at 2002 in Electrical & Electronics discipline and pursues his Master's degree in Applied Electronics in Sathyabama University, Chennai and awarded at 2004. He is doing his research work in the field of Signal processing. Now he is an Associate Professor in the department of Electronics & Communication Engineering of Jeppiaar Engineering College, Chennai. He presented 10 papers in National and International Conferences.