CHAPTER 6

LEAST MEAN SQUARE ALGORITHM

6.1 Introduction

The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959 [12] is an adaptive algorithm, which uses a gradient-based *method of steepest decent* [10]. LMS algorithm uses the estimates of the gradient vector from the available data. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to the minimum mean square error. Compared to other algorithms LMS algorithm is relatively simple; it does not require correlation function calculation nor does it require matrix inversions.

6.2 LMS Algorithm and Adaptive Arrays

Consider a Uniform Linear Array (ULA) with N isotropic elements, which forms the integral part of the adaptive beamforming system as shown in the figure below.

The output of the antenna array $x(t)$ is given by,

$$x(t) = s(t)a(\theta_0) + \sum_{i=1}^{N_u} u_i(t)a(\theta_i) + n(t) \tag{6.1}$$

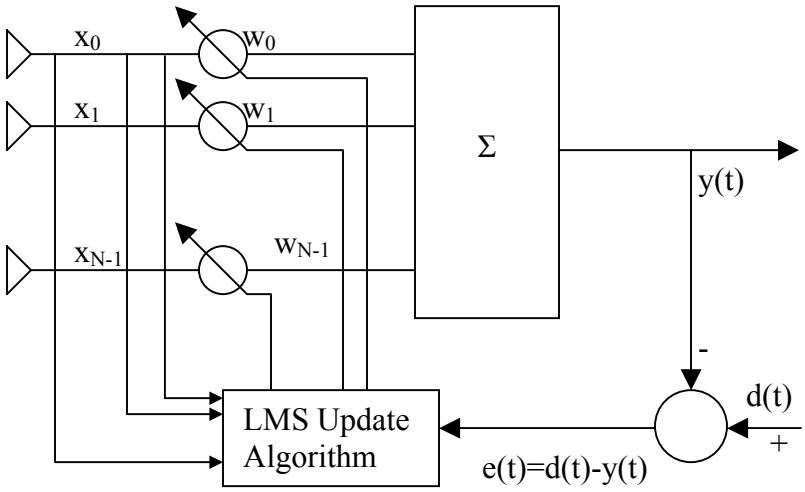

Figure 6.1 LMS adaptive beamforming network

$s(t)$ denotes the desired signal arriving at angle θ_0 and $u_i(t)$ denotes interfering signals arriving at angle of incidences θ_i respectively. $a(\theta_0)$ and $a(\theta_i)$ represents the steering vectors for the desired signal and interfering signals respectively. Therefore it is required to construct the desired signal from the received signal amid the interfering signal and additional noise $n(t)$.

As shown above the outputs of the individual sensors are linearly combined after being scaled using corresponding weights such that the antenna array pattern is optimized to have maximum possible gain in the direction of the desired signal and nulls in the direction of the interferers. The weights here will be computed using LMS algorithm based on Minimum Squared Error (MSE) criterion. Therefore the spatial filtering problem involves estimation of signal $s(t)$ from the received signal $x(t)$ (i.e. the array output) by minimizing the error between the reference signal $d(t)$, which closely matches or has some extent of correlation with the desired signal estimate and the beamformer output *y(t)* (equal to *wx(t)*). This is a classical Weiner filtering problem for which the solution can be iteratively found using the LMS algorithm.

6.3 LMS algorithm formulation (All signals are represented by their sample values)

From the *method of steepest descent*, the weight vector equation is given by [10],

$$w(n+1) = w(n) + \frac{1}{2}\mu[-\nabla(E\{e^2(n)\})] \tag{6.2}$$

Where μ is the step-size parameter and controls the convergence chachteristics of the LMS algorithm; $e^2(n)$ is the mean square error between the beamformer output $y(n)$ and the reference signal which is given by,

$$e^2(n) = [d^*(n) - w^h x(n)]^2 \tag{6.3}$$

The gradient vector in the above weight update equation can be computed as

$$\nabla_w(E\{e^2(n)\}) = -2r + 2Rw(n) \tag{6.4}$$

In the *method of steepest descent* the biggest problem is the computation involved in finding the values r and R matrices in real time. The LMS algorithm on the other hand simplifies this by using the instantaneous values of covariance matrices r and R instead of their actual values i.e.

$$R(n) = x(n)x^h(n) \tag{6.5}$$

$$r(n) = d^*(n)x(n) \tag{6.6}$$

Therefore the weight update can be given by the following equation,

$$\begin{aligned} w(n+1) &= w(n) + \mu x(n)[d^*(n) - x^h(n)w(n)] \\ &= w(n) + \mu x(n)e^*(n) \end{aligned} \tag{6.7}$$

The LMS algorithm is initiated with an arbitrary value $w(0)$ for the weight vector at n=0. *The* successive corrections of the weight vector eventually leads to the minimum value of the mean squared error.

Therefore the LMS algorithm can be summarized in following equations;

$$Output,\ y(n) = w^h x(n) \tag{6.8}$$

$$Error,\ e(n) = d^*(n) - y(n) \tag{6.9}$$

$$Weight,\ w(n+1) = w(n) + \mu x(n) e^*(n) \tag{6.10}$$

6.4 Convergence and Stability of the LMS algorithm

The LMS algorithm initiated with some arbitrary value for the weight vector is seen to converge and stay stable for

$$0 < \mu < 1/\lambda_{max} \tag{6.11}$$

Where λ_{max} is the largest eigenvalue of the correlation matrix *R*. The convergence of the algorithm is inversely proportional to the eigenvalue spread of the correlation matrix *R*. When the eigenvalues of R are widespread, convergence may be slow. The eigenvalue spread of the correlation matrix is estimated by computing the ratio of the largest eigenvalue to the smallest eigenvalue of the matrix. If μ is chosen to be very small then the algorithm converges very slowly. A large value of μ may lead to a faster convergence but may be less stable around the minimum value. One of the literatures [will provide reference number here] also provides an upper bound for μ based on several approximations as $\mu <= 1/(3trace(R))$.

6.5 Simulation results for the LMS algorithm

For simulation purposes a 4-element linear array is used with its individual elements spaced at half-wavelength distance. The desired signal $s(t)$ arriving at θ_0 is a simple complex sinusoidal-phase modulated signal of the following form,

$$s(t) = e^{j\sin(\omega t)} \tag{6.12}$$

The interfering signals $u_i(t)$ arriving at angles θ_i are also of the above form. By doing so it can be shown in the simulations how interfering signals of the same frequency as the desired

signal can be separated to achieve rejection of co-channel interference. However, Rayleigh fading is added to the incoming interfering signals. The Rayleigh faded interfering signal consists of random phase and amplitude. Illustrations are provided to give a better understanding of different aspects of the LMS algorithm with respect to adaptive beamforming. For simplicity purpose the reference signal $d(t)$ is considered to be the same as the desired signal $s(t)$.

6.5.1 Beamforming examples

Two examples are provided to show the beamforming abilities of the LMS algorithm. Each example has a normalized array factor plot and corresponding LMS error plot

Case 1:

In the first case the desired angle is arriving at 30 degrees and there are three interfering signals arriving at angles –20, 0 and 60 degrees respectively. The array factor plot in Figure 6.2a shows that the LMS algorithm is able to iteratively update the weights to force deep nulls at the direction of the interferers and achieve maximum in the direction of the desired signal. It can be seen that nulls are deep at around 40dB level below the maximum. The LMS error plot in figure 6.2b shows that the LMS algorithm converges. In this case the LMS error is almost 0.025 at around 3000 samples.

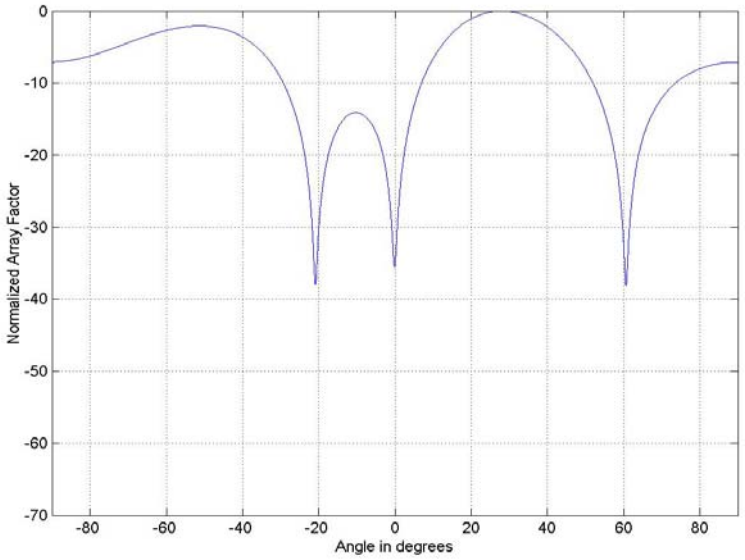

Figure 6.2a Normalized Array Factor plot for case1

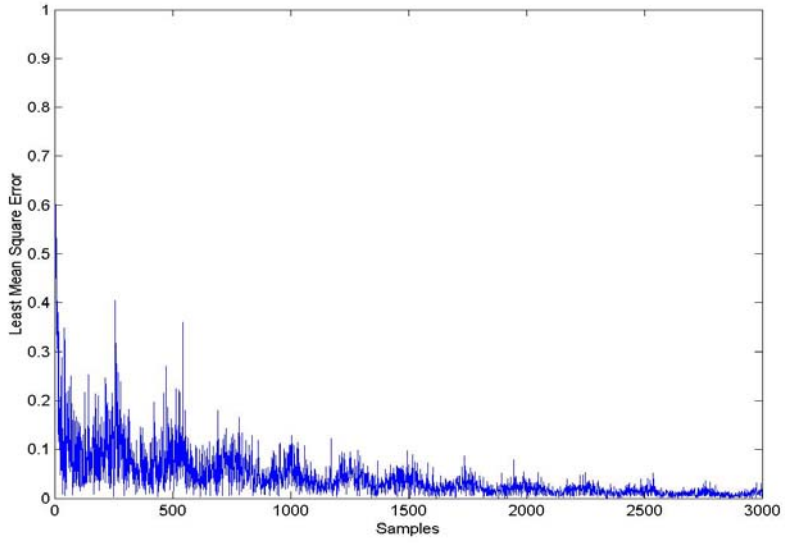

Figure 6.2b LMS error plot for case1

The optimum complex weights in the case considered for which the algorithm converges is found to be $W_1 = 0.3279 + 0.0051i$, $W_2 = 0.1599 - 0.1654i$; $W_3 = -0.1678 - 0.1652i$ and $W_4 = 0.0063 + 0.3125i$. The magnitude of the complex weights plotted against the number of samples in figure 6.2c shows the convergence of the weights to its optimum values.

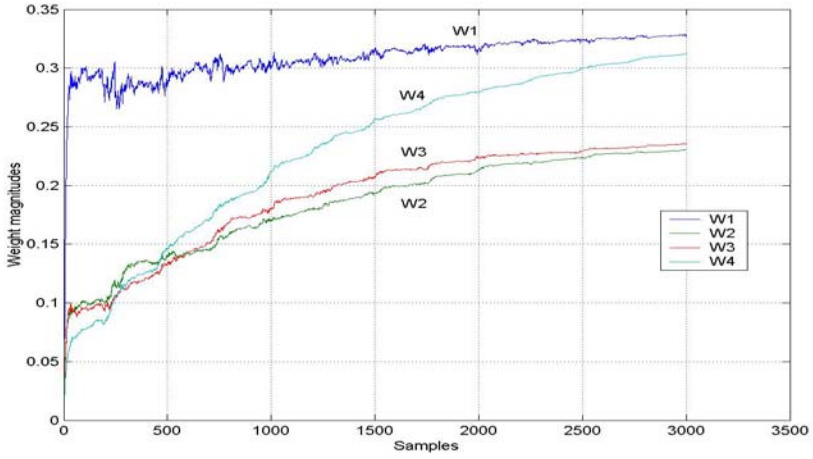

Figure 6.2c Weight convergence plot for case1

Case 2:

Another example shown in figure.6.3a is provided for the case where the desired signal is arriving at an angle of 0^0 and there are four interferers (compared to three interferers in previous case) arriving at angles –40, -10, 30 and 60 degrees respectively. It is seen that nulls are deeper when the number of interferers are larger at around the 50dB level from the maximum value. However, from the LMS error plot in figure 6.3b, it is quite evident that it takes longer to converge, it takes about 6000 samples to converge where the error is less than 2%.

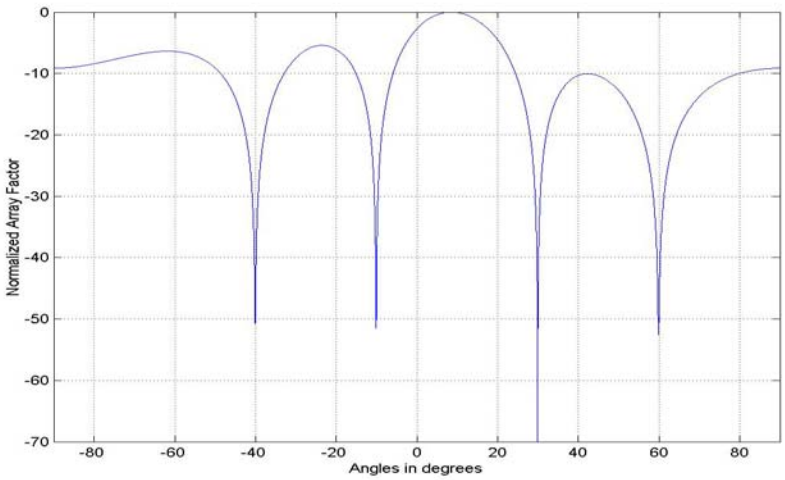

Figure 6.3a Normalized Array Factor plot for case2

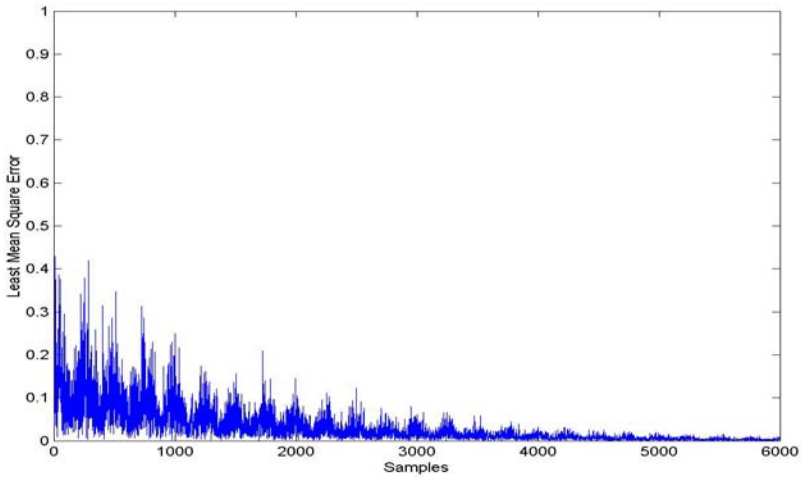

Figure 6.3b LMS error plot for case2

Figure 6.4, shown below, gives a comparison of the normalized array factor plots (for case1) to understand the effect of under convergence when there is not enough time for the algorithm to converge. It can be seen clearly that after first few iterations (in this case 300 iterations) the algorithm hasn't converged at all. The nulls are not at the interfering direction, also not deep enough indicating that it will take longer than 300 iterations to converge.

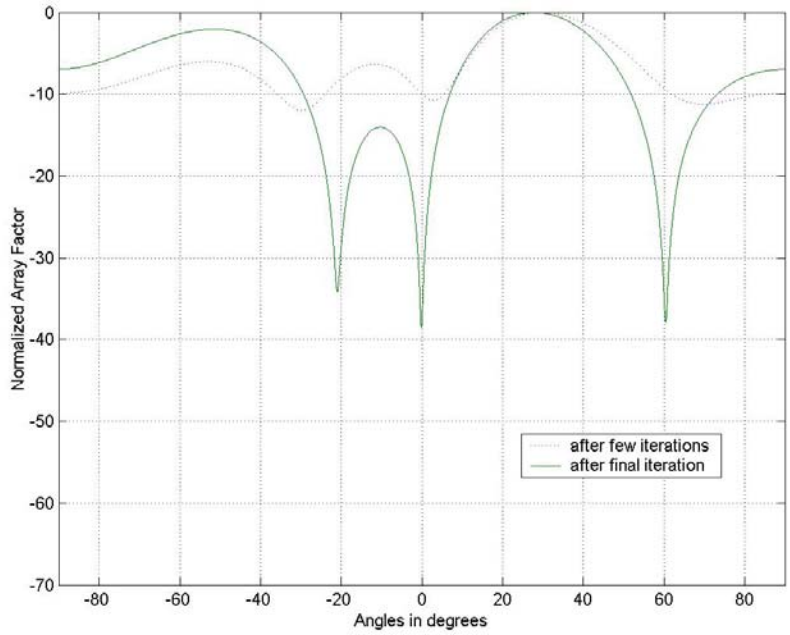

Figure 6.4 Normalized array factor plots at different iteration values

6.5.2 Dependency of the step-size parameter μ

The step-size parameter or the convergence factor μ is the basis for the convergence speed of the LMS algorithm. For the LMS algorithm to converge and be stable equation 6.11 repeated below gives the allowable range of μ.

$$0 < \mu < \frac{1}{\lambda_{\max}}$$ Where λ_{max} is the largest eigenvalue of the correlation matrix R

The LMS error for case 1 is shown in figure 6.5a with a convergence factor of $\mu = 0.013$. This satisfies the above condition that leads to a decent convergence where the error is within 2%. The effect of small and large values of μ is discussed subsequently using plots in figures 6.5b and figure 6.6c.

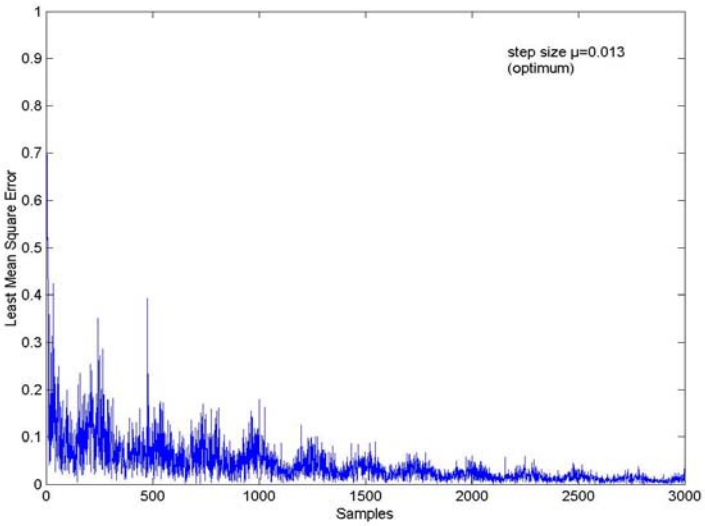

Figure 6.5a LMS error for optimum value of μ

The LMS error plot in Figure 6.5b shows the effect of the convergence factor on the convergence when μ is small. For case1 when $\mu = 0.0013$, i.e when μ is small the convergence rate is slow the error is still quite large. It has not converged even after 3000 samples with the error at 30%.

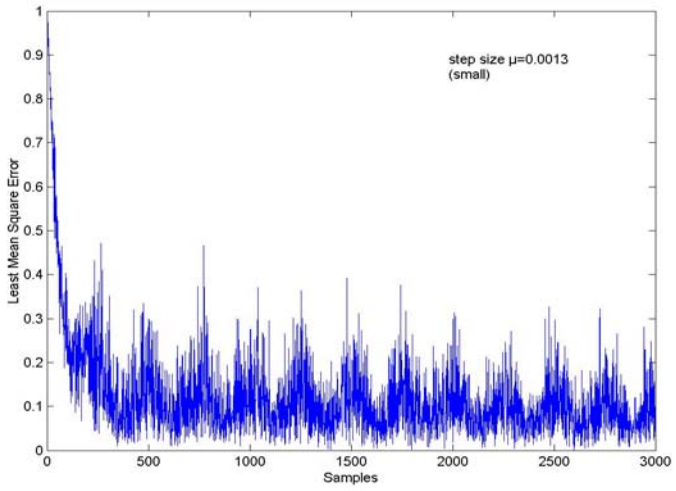

Figure 6.5b LMS error for small value of μ

The LMS error plot in Figure 6.5c shows the effect of the convergence factor on the convergence when μ is large. A moderately large value of μ leads to faster convergence. However, when the value of μ is too high it leads to the instability of the algorithm and leads to an erroneous result. For case 1 when $\mu = 0.2$ the LMS error is seen to be erratic as shown in figure 6.5c.

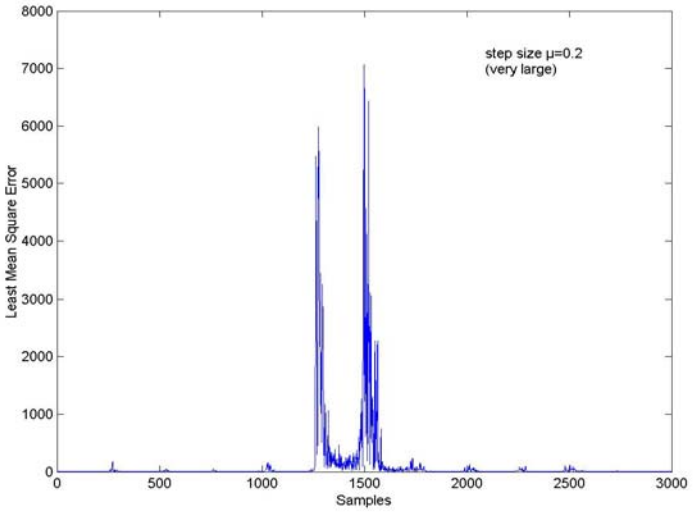

Figure 6.5c LMS error for large value of μ

6.5.3 Weighted Signal

The plot shown in figure 6.6 shows the tracking of the desired signal by using the LMS algorithm. At the beginning of the adaptation process there is a significant amount of error between the weighted signal $y(t)$ and the desired signal. This is because the algorithm is initiated with an arbitrary value for weight, which is nowhere close to the optimum weight. However, as the adaptation process continues, based on the error computed at every iteration, the weight converges towards its optimum and the weighted signal $y(t)$ follows the desired signal $s(t)$ more closely with a small error.

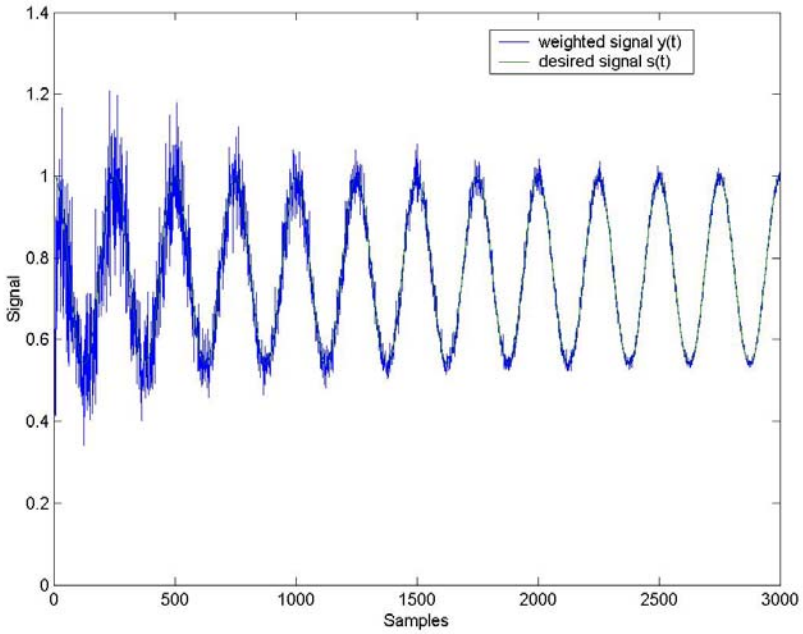

Figure 6.6 Weighted signal vs. desired signal response

The LMS algorithm is most commonly used adaptive algorithm because of its simplicity and a reasonable performance. Since it is an iterative algorithm it can be used in a highly time-varying signal environment. It has a stable and robust performance against different signal conditions. However it may not have a really fast convergence speed compared other complicated algorithms like the *Recursive Least Square* (RLS). It converges with slow speeds when the environment yields a correlation matrix R possessing a large eigenspread. Usually traffic conditions are not static, the user and interferer locations and the signal environment are varying with time, in which case the weights will not have enough time to converge when

adapted at an identical rate. That is, μ the step-size needs to be varied in accordance with the varying traffic conditions. There are several variants of the LMS algorithm that deal with the shortcomings of its basic form. The *Normalized LMS* (NLMS) introduces a variable adaptation rate. It improves the convergence speed in a non-static environment. In another version, the *Newton LMS*, the weight update equation includes whitening in order to achieve a single mode of convergence. For long adaptation processes the *Block LMS* is used to make the LMS faster. In block LMS, the input signal is divided into blocks and weights are updated blockwise. A simple version of LMS is called the *Sign LMS.* It uses the sign of the error to update the weights. Also, LMS is not a blind algorithm i.e. it requires *a priori* information for the reference signal.