4.1 Adaptive Filter

Generally in most of the live applications and in the environment information of related incoming information statistic is not available at that juncture adaptive filter is a self regulating system that take help of recursive algorithm for processing. Moreover it is self regulating filter which uses some training vector that delivers various comprehensions of a desired response can be merged with reference to incoming signal. First input and training is compared accordingly error signal is generated and that is used to adjust some previously assumed filter parameters under effect of incoming signal. Filter parameter adjustment continue until steady state condition [19].

As far as application of noise reduction from speech is concern, adaptive filters can give best performance. Reason for that noise is somewhat similar with the randomly generates signal and every time its very difficult to measure its statistic. Design of fixed filter is completely failed phenomena for continuously changing noisy signal with the speech. Some of the signal changes with very fast rate in the context of information in the process of noise cancellation which requires the help of self regularized algorithms with the characteristics to converge rapidly. LMS and NLMS are generally used for signal enhancement as they are very simple and efficient. Because of very fast convergence rate and efficiency RLS algorithms are most popular in the specific kind of applications. Brief overview of functional characteristics for mentioned adaptive filter describes in following sections.
4.2 Least Mean Square Adaptive Filters

In the signal processing there is wide variety of stochastic gradient algorithm in that the LMS algorithm is an imperative component of the family. The LMS algorithm can be differentiated from the steepest descent method by term stop chiastic gradient for which deterministic gradient works usually in recursive computation of filter for inputs is used which is having the noteworthy feature of simplicity for which it is made standard over other linear adaptive filtering algorithms. Moreover, it does not require matrix inversion [20].

The LMS algorithm needs two fundamental process on the signal and it is of linear type of adaptive algorithm [19].

1. (1) An residue error can be predicted by comparing the output from the linear filtering phenomena and (2) Accordingly response to the input signal with the necessary response of the signal. Mainly above two parts are from filtering process.

2. Estimated error mainly takes part in generation of updating filter vector in the automatic adjustment of the parametric model.

Figure 4.1 Concept of adaptive transversal filter
Figure 4.2 Detailed structure of the transversal filter component

Figure 4.3 Detailed structure of the adaptive weight control mechanism
The mixture of these two developments working collectively creates close loop with reverse mechanism, as illustrated in the Figure 4.1. LMS algorithm believes in nature of transversal filter shown in Figure 4.2 [19]. This module is used for performing the adaptive control process on the tap weights vector of the transversal filter to enhance the designation adaptive weight control mechanism as shown in the Figure 4.3 [21].

In the adaptive filter most important part is the tap inputs form the fundamentals tap input vector \( u(n) \) is matrix length \( M \) and one row, where the number of delay elements is presented with \( M \) length vector and these inputs extent a multidimensional space denoted by \( \hat{U}_n \). Correspondingly, the tap weights are main elements. By taking base of wide sense stationary process the value computed for the LMS algorithm gives output and which is very nearer to wiener solution of the filter. This happens when the number of repetitions, \( n \), procedures tends to infinity.

In the filtering process the wanted reaction \( d(n) \) is supplied for processing and collectively with the tap input vector \( u(n) \). This fetched input is very important and it utilizes in the transversal filter which creates an output \( d(n \mid \hat{U}_n) \) used as an evaluation of the required response \( d(n) \). In the further process, an estimation error \( e(n) \) can be quoted and that representation used to take the modification between the actual needed response and the actual filter output, as shown in the output end of Figure 4.2 relationship of \( e(n) \) and \( u(n) \) can be shown. Obtained detailed values of the vector are helpful to manage closed path around feedback mechanism of system.

Figure 4.3 has given depth of adaptive weight control process. Purposely, a tap input vector \( u(n-k) \) and the inner product of the estimation error \( e(n) \) is calculated for various values of \( k \) starting with 0 to \( M-1 \). \( \mu \) is defined scaling factor in the process of calculation and which is non negative quantity that is also know as a step size of the process which can be clearly seen in the Figure 4.3.

Comparing the control mechanism of Figure 4.3 for the LMS algorithm with that of for the method of steepest descent it can be seen that the LMS algorithm in the process taking convolution of \( u(n-k) e^*(k) \) and it can be considered as prediction of element \( k \) in
the gradient vector $J(n)$ that follows rules of steepest descent concept in the mechanism. In other words, the expectation operator is removed from all the paths in Figure 4.3.

It is assumed that the from a jointly wide sense stationary environment the tap input and the desired response can be computed. In the adaptive filtering multiple regression model is taken into consideration in which its some characteristics and parametric vector is unknown hence the need for self adjusting filtering and linearly change of $d(n)$. For computing of tap vector $w(n)$ that changes and goes down at that time the ensemble sup up and its average error performance surface with a deterministic trajectory. Now that surface terminates on the vector of wiener solution. It is better and suitable for wiener solutions that $\hat{w}(n)$ different from $w(n)$ computed by the LMS algorithm follows a non predictable motion around the minimum point of the error performance surface and it can be observed that this motion is a form of Brownian motion for small $\mu$ [22].

Earlier, it is pointed out that the LMS filter involves feedback in its operation, which raises the related issue of stability. In this context, a meaningful criterion is to require that as $J(n)$ tends to $j(\infty)$ with $n$ tends to $\infty$ in general manner.

It can be recognized that $J(n)$ is outcome of LMS process and it is in terms of MSE at time $n$ and its final value $J(\infty)$ is a constant. By LMS algorithm if step size parameter is adjusted related to the spectral content of the tap inputs then it will satisfy following condition of the stability in the mean square.

The excess mean square error can be defined as the difference between the final value $J(\infty)$ and the minimum values $J_{\min}$ attained by the wiener solution. This difference indicates the price paid for using the adaptive (stochastic) method to cover and calculate the tap weights in the LMS filter instead of a deterministic approach, as in the method of steepest descent. The ratio of $J_{ex}(\infty)$ to $J_{min}$ is called the misadjustment, which gives difference of LMS and winner solutions. It is interesting here to note that the complete feedback mechanism acting around the tap weights acts in similarity to low pass filter, whose average time constant is inversely varies to step size parameter. As a consequence it is necessary to adjust small value to step size parameter and tends to adaptive process is
slowly in the convergence direction and because of that effects of gradient noise on tap weights are heavily filtered out. By this process in the cumulative manner results in misadjustment in the process.

Most advantageous feature of LMS adaptive algorithm is that it is very straightforward in the implementation and still very efficiently able to adjust with outer environment as per the requirement. Only limitation of the performance arises by choice of the step size parameters.

4.2.1 Least Mean Square Adaptation Algorithm

Using the steepest descent algorithm if it is mainly concentrated to make accurate measurement of the vector named gradient \( J(n) \) at every regular iteration. It is also possible to compute tap weight vector if step size parameter is suitably selected. Step size selection and tap weight vector optimally computed would be related to optimum wiener solution. As the advance knowledge of both mentioned matrix like correlation matrix \( R \) of the tap input and the cross correlation vector \( P \) between the tap inputs and the desired response.

To achieve an estimation of \( \nabla(n) \), very important method is to take another estimates of of the correlation matrix \( R \) and the cross correlation vector \( P \) in the formula, which is produced here for convenience [23].

\[
\nabla J (n) = -2p + 2Rw (n)
\]  

(4.1)

Very obvious choice of predictors is computation by using instantaneous estimates for \( R \) and \( P \) that are collaborated by the different discrete magnitude values of the tap input vector and necessary response, defined respectively by

\[
\tilde{R} (n) = u (n) u^H (n)
\]  

(4.2)

\[
P^\ast (n) = u (n) d^\ast (n)
\]  

(4.3)

Compatibly, the gradient vector instantaneous value can be defined as

\[
\nabla \hat{J} (n) = -2u (n) d^\ast(n) +2u(n)u^H(n) w ^\ast (n)
\]  

(4.4)
Note that the estimate $\mathcal{J}(n)$ may also be viewed as the gradient operator applied to the instantaneous squared error $|e(n)|^2$.

Substituting the estimate of for the gradient vector $J(n)$ in the steepest descent algorithm described, following relation can be taken to be into

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n) [d^*(n) - u^H(n) \hat{w}(n)]$$  \hspace{1cm} (4.5)

Here the tap weight vector has been used to distinguish it from the values obtained by using the steepest descent algorithm. Equivalently, it may be written that the result in the form of three basic relations as follows:

1. Filter output:
   $$Y(n) = \hat{w}^H(n) u(n)$$  \hspace{1cm} (4.6)

2. Estimation error or error signal:
   $$e(n) = d(n) - y(n)$$  \hspace{1cm} (4.7)

3. Tap weight adaptation:
   $$\hat{w}(n+1) = \hat{w}(n) + \mu u(n) e^*(n)$$  \hspace{1cm} (4.8)

Above equations show the estimation error $e(n)$, the calculation of which is decided on the present estimate of the tap weight vector, $\hat{w}(n)$. It is important to take into consideration that $\mu u(n) e^*(n)$ term, shows adjustment which is applied to the present estimate of the tap weight vector, $\hat{w}(n)$.

Mainly algorithm explained by mentioned equations is the complex form of the LMS algorithm. Inputs required by the algorithm should be most recent and fresh in terms of error vector, input vector etc. Here input are in the terms of stochastic range and the allowed set of direction along which it can go ahead from one iteration process to the next is non deterministic in nature and be thought of as consists of real gradient vector directions.
LMS algorithm is most popular because of this convergence speed but selection of step size is very important in the case of success of algorithm.

Figure 4.4 shows a LMS algorithm mechanism in the form of signal flow graph. This model bears a close resemblance to the feedback model of describing the steepest descent algorithm. The signal flow graph in the Figure 4.4 clearly demonstrates the simplicity of the LMS algorithm. In particular, it can be found that this Figure 4.4 that the per equation iteration LMS algorithm take requires only 2M+1 complex multiplications and 2M complex, where M is the number of tap weights in basic transverse filter. Comparatively large variance can be achieved by the instantaneous estimates of R and p. By the first step analysis it can be seen that LMS algorithm can not perform well because it uses present estimations. Still it is dynamic feature of LMS that it is recursive in nature, with the result that the algorithm itself effectively averages these estimates, in some sense, during the course of adaptation. LMS algorithm can be summarized as in following section.
### Table 4.1 Summary of the LMS Algorithm

<table>
<thead>
<tr>
<th>Parameter: M = number of taps (i.e. filter length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>µ = step size parameter</td>
</tr>
<tr>
<td>0 &lt; µ &lt; 2/MS max,</td>
</tr>
<tr>
<td>Where Smax is the maximum value of the power spectral density of the tap inputs u (n) and the filter length M is moderate to large.</td>
</tr>
<tr>
<td>Initialization: If prior knowledge of the tap weight vector ŵ (n) is available, use it to select an appropriate value for ŵ (n). Otherwise set ŵ (0) = 0.</td>
</tr>
<tr>
<td>• Given u (n) = M by 1 tap input vector at time n</td>
</tr>
<tr>
<td>= [u (n), u (n - 1),..., u (n – m + 1)]T</td>
</tr>
<tr>
<td>(n) = desired response at time n</td>
</tr>
<tr>
<td>• To be computed</td>
</tr>
<tr>
<td>ŵ (n + 1) = estimate of tap weight vector at time n + 1</td>
</tr>
<tr>
<td>Computation: For n= 0,1,2,......, compute</td>
</tr>
<tr>
<td>e (n) = d (n) - ŵH (n) u (n)</td>
</tr>
<tr>
<td>ŵ (n + 1) = ŵ (n) + µu(n) e* (n)</td>
</tr>
</tbody>
</table>

In Table 4.1 a summary of the LMS algorithm is represented in which equations incorporate. The Table 4.1 also includes a constraint on the allowed value of the acceptable step size parameters, which is needed to ensure that the algorithm converges. More is said on this necessary condition for convergence.
4.2.2 Statistical LMS Theory

Previously, it is referred that the LMS filter as a linear adaptive filter “linear” in the sense that its physical implementations is built around a linear combiner. In reality, however, the LMS filter is a highly complex nonlinear estimator that violates the principles of superposition and homogeneity [24]. Let $y_1(n)$ denote the response of a system to an input vector $u_1(n)$. Likewise, let $y_2(n)$ denote the response of the system to another input vector $u_2(n)$. For a system to be linear the composite input vector $u_1(n) + u_2(n)$ must result in a response equal to $y_1(n) + y_2(n)$; this result is called the principle of superposition. Furthermore, a linear system must satisfy the homogeneity property; that is, if $y(n)$ is the response of the system to an input vector $u(n)$, then the response of the system to the scaled input vector, where ‘$a$’ is a scaling factor. Consider now the LMS filter. Starting with the initial conditions $w(0) = 0$ and the frequent application of the weight update gives as under:

$$w(n) = \mu \sum_{i=0}^{n-1} e^*(i) u(i) \quad (4.9)$$

Below equations shows input output relation of LMS algorithm

$$y(n) = w^H(n) u(n) \quad (4.10)$$

$$= \mu \sum_{i=0}^{n-1} e(i) u^H(i) u(n) \quad (4.11)$$

Recognizing that the error signal $e(i)$ decided by the input vector $u(i)$, it can be defined from equation output of the filter is takes non linear nature and its function of input. The properties of superposition and homogeneity are thereby both violated by the LMS filter.

Thus, although the LMS filter is very simple in physical terms, its mathematical analysis is profoundly complicated because of its highly nonlinear nature. To proceed with a statistical analysis of the LMS filter, it is convenient to work with the weight error vector rather than the tap weight vector itself. Weight error vector in the LMS filter can be denoted by

$$\varepsilon(n) = w_o - \hat{w}(n) \quad (4.12)$$
Subtracting equation from the optimum tap weight vector \( w_o \) and using the definition of equation, to eliminate \( w(n) \) from the adjustment term on other side and it can be rearranged in the below form that the LMS algorithm in terms of the weight error vector \( \varepsilon(n) \) as

\[
\varepsilon(n+1) = [I - \mu u(n) u^H(n)] - \mu u(n) e^*_o(n) \tag{4.13}
\]

Where I is the identify matrix and

\[
e_0(n) = d(n) - w^H_o u(n) \tag{4.14}
\]

is the estimation error produced by the optimum wiener filter.

### 4.2.3 Direct Averaging Method

It is very critical to analyze convergence nature of such a stochastic algorithm in an average sense, the direct averaging method is useful. According to this method, the possible outcome of the stochastic difference equation is operating under the consideration of a very less valued step size parameter is by virtue of the low pass filtering action of the LMS algorithm near and similar to the answer of another stochastic difference equation with system matrix is equal to the ensemble average,

\[
E \left[ I - \mu u(n) u^H(n) \right] = I - \mu R \tag{4.15}
\]

R can be recognized as the correlation matrix of the tap input vector \( u(n) \) [25]. More specifically, it may be replaced the stochastic difference representation with another stochastic difference representation described by

\[
\varepsilon_0(n+1) = (I - \mu R) \varepsilon_0(n) - \mu u(n) e^*_0(n) \tag{4.16}
\]

where, for reasons that will be become apparent presently.

### 4.2.4 Small Step Size Statistical Theory

The development of statistical LMS theory to small step sizes should be restricted, embodied in the following assumptions:
Assumptions I. LMS algorithm can be acts as a low pass filter with a low with very less cut off because the step size parameter $\mu$ is small [25].

Under this assumption it might be used that the zero order terms $e_o(n)$ and $k_o(n)$ as approximations to the actual $e(n)$ and $K(n)$, respectively. To illustrate the validity of assumption I, consider the example of and LMS filter using a single weight. For this example, the stochastic difference equation simplifies to the scalar form

$$e_o(n+1) = (1 - \mu \sigma^2_u)e_o(n) + f_0(n)$$  \hspace{1cm} (4.17)

Where $\sigma^2_u$ is the variance $u(n)$. This difference equation represents a transfer function with single pole at given equation with in nature low pass filter

$$Z = (1 - \mu \sigma^2_u)$$  \hspace{1cm} (4.18)

For small $\mu$, the pole lies inside of, and very close to, $z$ plane unity circle, which implies a very low cutoff frequency.

Assumption II. The actual logic by which the observable data can be generated is that the desired response $d(n)$ is represented by a linear multiple regression model that is very similar to wiener filter and which is given by,

$$d(n) = w_0^H u(n) + e_0(n)$$  \hspace{1cm} (4.19)

Where the irreducible estimation error $e_0(n)$ is a process of compared to white noise which not dependent to the input vector values [26].

The characterization of $e_o(n)$ as white noise means that its successive samples are uncorrelated, as shown by

$$E [e_0(n) e_0^*(n-k) ] = \begin{cases} n J_{\min} & \text{for } k = 0 \\ 0 & \text{for } k \neq 0 \end{cases}$$  \hspace{1cm} (4.20)

The essence of the second assumption it can be showed that, provide that the use of a linear multiple regression model is justified and the no of co efficient in wiener filter is
nearly same to the level of the regression model. The statistical independence of \( e_o(n) \) from \( u(n) \) is stronger than the principle of orthogonality.

The choice of a small step size according to Assumption I is certainly under the designer’s control. To match the LMS filter’s length of the multiple regression model of with suitable order in Assumption II required the use of model selection criterion.

Assumption III. Desired response and the input vector are jointly Gaussian.

Thus, the small step size theory to be developed shortly for the statistical characterization of LMS filters applies to one of two possible scenarios: Assumption II holds, whereas in the other scenario, Assumption III holds. Between them, these two scenarios cover a wide range of environments in which the LMS filter operates. Most importantly, in deriving the small step size theory.

4.2.5 Natural Modes of the LMS Filter

Under assumption I, Butterweck’s interactive procedure reduces to the following pair of equations:

\[
e_{o}(n+1) = (I-\mu R)e_{o}(n) + f_{o}(n) \tag{4.21}
\]

\[
f_{o}(n) = -\mu u(n)e_{o}^{*}(n) \tag{4.22}
\]

Before proceeding further, it is informative to transform the difference equation into a simpler form by applying the unitary similarity transformation to the correlation matrix \( R \) [19]. It can be obtained that

\[
Q^{H} R Q = \Lambda \tag{4.23}
\]

Where \( Q \) is a unitary matrix whose columns constitute an orthogonal set of eigenvectors associated with eigen values of the correlation matrix \( R \) and \( \Lambda \) is a diagonal matrix in which it consist of the eigenvalues. To achieve the desired simplification, the definition can be introduced also as

\[
V(n) = Q^{H}e_{o}(n) \tag{4.24}
\]

Defining property of the unitary matrix \( Q \), namely
\[ \mathbf{Q} \mathbf{Q}^H = \mathbf{I} \quad (4.25) \]

I can be represented as the identity matrix,

\[ \mathbf{v}(n+1) = (\mathbf{I} - \mu \Lambda)\mathbf{v}(n) + \Phi(n) \quad (4.26) \]

Where the new vector \( \Phi(n) \) is defined in terms of \( f_0(n) \) by the transformation

\[ \Phi(n) = \mathbf{Q}^H f_0(n) \quad (4.27) \]

For a partial characterization of the stochastic force vector \( \Phi(n) \), its mean and correlation matrix over an ensemble of LMS filters may be expressed as follows:

1. First compute the mean value of the stochastic force vector \( \Phi(n) \). And deliberately it must be zero:

\[ E[\Phi(n)] = 0 \text{ for all } n \quad (4.28) \]

2. \( \Phi(n) \) is a diagonal matrix and is of the correlation matrix of the stochastic force directional quantity; that is,

\[ E[\Phi(n)\Phi^H(n)] = \mu^2 J_{\min} \Lambda \quad (4.29) \]

\( J_{\min} \) shows the minimum mean square error which is generated by the wiener filter and \( \Lambda \) is the diagonal matrix of eigenvalues of the correlation matrix.

### 4.2.6 Learning Curves for Adaptive Algorithms

Statistical work of adaptive filters can be observed by ensemble average learning curves. Identical two types of learning curves are as under [19].

1. First type is the mean square error (MSE) learning curve. MSE curve produces ensemble averaging of squared estimation error. Means plot of mean values in the learning curve is

\[ J(n) = E \left[ \left| e(n) \right|^2 \right] \quad (4.30) \]

versus the iteration \( n \).
2. Second most important is the mean square deviation (MSD) learning curve, which is processed by taking ensemble averaging of the squared error deviation $\| \varepsilon(n) \|^2$. The mean square deviation versus the iteration $n$ is plotted in the second learning curve.

$$D(n) = \mathbb{E}[\| \varepsilon(n) \|^2]$$  \hspace{1cm} (4.31)

The estimation error generated by the LMS filter is expressed as

$$e(n) = d(n) - \hat{w}^H(n) u(n) \hspace{1cm} (4.32)$$

$$= d(n) - w_0^H u(n) + \varepsilon H(n) u(n)$$

$$= e_0(n) + \varepsilon H(n) u(n)$$

$$= e_0(n) + \varepsilon_0^H(n) u(n) \hspace{0.5cm} \text{for } \mu \text{ small.}$$

$e_0(n)$ is the estimation error and $\varepsilon_0(n)$ is the zero order weight error vector of the LMS filter. Hence, the mean square error produced and it is shown by following iterations

$$J(n) = \mathbb{E}[\| e(n) \|^2] \hspace{1cm} (4.33)$$

$$\approx \mathbb{E}[(e_0(n) + \varepsilon_0^H(n) u(n)) (e_0(n) + u^H(n) e_0(n))]$$

$$= J_{min} + 2 \text{Re} \{\mathbb{E}[\varepsilon_0^* n(n) u(n)]\} + \mathbb{E}[\varepsilon_0^H(n) u(n) u^H(n) e_0(n)]$$

$J_{min}$ is the minimum mean square error. Denotes the real part of the quality enclosed between the braces. Following reasons depending on which scenario applies and so that right hand side of equation gets null value: Under Assumption II, the irreducible estimation error $e_0(n)$ produced by the wiener filter is statistically independent. At $n$ iteration, the zero order weight error vector $\varepsilon_0(n)$ depends on past values of $e_0(n)$, a relationship that follows from the iterated use [27]. Hence, here it can be written

$$\mathbb{E}[(e_0^*(n) \varepsilon_0^H(n) u(n))] = \mathbb{E}[e_0^*(n)] \mathbb{E}[\varepsilon_0^H(n) u(n)]$$

$$= 0$$

The null result of above equation also holds under Assumption III. For the kth components of $\varepsilon_0(n)$ and $u(n)$, it can be he expected,

$$\mathbb{E}[(e_0^*(n) \varepsilon_0^k(n) u(n-k))], \hspace{0.5cm} k = 0, 1, \ldots, M - 1$$  \hspace{1cm} (4.35)
Assuming that the are Jointly Gaussians are input vector and desired response and the estimation error $e_0(n)$ is therefore also Gaussian, then applying the identity described, it can be obtained immediately that

$$E [e_0^*(n) e_0^*, k (n) u (n-k)] = 0 \quad \text{for all } k$$

(4.36)

### 4.2.7 Comparison of the LMS Algorithm with the Steepest Descent Algorithm

When the coefficient set value of the transversal filter approaches the optimum value and it is defined by wiener equation then the minimum mean square error $J_{\text{min}}$ is realized. Mentioned condition is recognized as ideal condition when number of iteration reaches to infinity by the steepest descent algorithm. The steepest descent algorithm measures gradient vector at each of the step in the iterations of the algorithm [19]. But in the case of LMS, it depends on a noisy momentary estimation with gradient vector, also with that the tap weight vector estimate $\hat{w}(n)$ for large $n$ and it can only fluctuate. Thus, after too many loop execution in the form of iteration the LMS algorithm results in a mean square error $J(\infty)$ that is greater than the minimum mean square error $J_{\text{min}}$. The amount by which the actual value of $J(\infty)$ is greater than $J_{\text{min}}$ is the excess mean square error.

A well-defined learning curve has been shown by steepest descent algorithm, gained by plotting the number of iterations versus mean square error. The learning involves of sum of descending exponentials, which equates the number of tap coefficients while in individual applications of the LMS algorithm, the noisy decaying exponentials representation is contained by the learning curve. The noise amplitude usually generates small values as the step size parameter $\mu$ is reduced in the limit the learning curve of the LMS filter assumes a deterministic character.

Adaptive transversal filter is in form of ensemble component and each of which is assumed to use the LMS algorithm with the same step size $\mu$ and the same initial tap weight vector $\hat{w}(0)$. In the case of adaptive filter it can be considered to give stationary ergodic inputs which are selected at random for the same statistical population. The learning curves which is noisy are calculated for this ensemble of adaptive filters.

Thus, two entirely different ensemble averaging operations are used in the steepest descent and LMS algorithms for determining their learning curves. In the steepest descent algorithm, the correlation matrix $R$ and the cross correlation vector $p$ are initially
computed using ensemble averaging operations which is useful to the populations of the
tap inputs and the wanted response calculation. These values are then used to calculate
the learning curve of the algorithm. In the LMS algorithm noisy learning curves are
computed for an ensemble of adaptive LMS filters with identical parameters. The
learning curve is then smoothed by averaging over the ensemble of noisy learning curves.

4.3 Normalized Least Mean Square Adaptive Filters

In the standard form of a least mean square filter, the tap weight vector of the filter at
iteration n+1 gets the necessary adjustment and gives the product of three terms:

- The step size parameter µ, which subject to design concept.
- The tap input vector u (n), which is actual input information to be processed.
- The estimation error e (n) for real valued data, or its complex conjugate e*(n) for complex valued data, which is calculated at iteration n.

The adjustment is directly proportional to the tap input vector u (n). As a result LMS
filter suffers from a gradient noise amplification problem in the case when u(n) is very
large. As a solution normalized LMS filter can be used. The term normalized can be
considered because the adjustment given to the tap weight vector at iteration n + 1 is
“normalized” with respect to the squared Euclidean norm of the tap input vector u(n)
[19].

4.3.1 Structure and Operation of NLMS

In the form of constructional view, the normalized LMS filter is exactly the same as the
standard LMS filter, as shown in the Figure 4.4. Fundamental concept of both the filter is
transversal filter.
Highlighted contrast in both type of algorithm is in weight upgradation mechanism. One vector which is long with values M in one row known as a tap input vector generates an output which is generally deducted from the desired response to generate the estimation error $e(n)$ [19]. Very natural modification in the vector modification directs new algorithm which is know as a normalized LMS algorithm.

The normalized LMS filter gives minimal disturbance and may be stated as follows: gradually by different iterations weight vector will change in straight weight will change step by step, it is controlled by updated filter output and its proposed values.

To cast this principle in mathematical terms, assume $\hat{\mathbf{w}}(n)$ denote the previous weight vector of the filter at iteration n and $\hat{\mathbf{w}}(n+1)$ denote its modified weight vector at next iteration. Selected conditions for implementing normalized LMS filter may be articulated in the category of constrained optimization which follows: Determination of updated tap weight vector $\hat{\mathbf{w}}(n+1)$ is possible from given the tap input vector $\mathbf{u}(n)$ and desired response $d(n)$.

Change can be highlighted in Euclidean norm,

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)$$  \hspace{1cm} (4.37)
Subject to the constraint
\[ \tilde{W}(n+1)^H u(n) = d(n) \]  

(4.38)

Described constrained can be analyzed in form of optimization problem and in that the method of Lagrange multipliers can be used.

\[ J(n) = ||\delta \hat{w}(n+1)||^2 + Re[\lambda^*(d(n) - \hat{w}(n+1)^H u(n))] \]  

(4.39)

Where \( \lambda \) is the complex valued Lagrange multiplier and the asterisk denotes complex conjugation. The squared Euclidean norm \( ||\delta \hat{w}(n+1)||^2 \) is, naturally, real valued. The real part operator, denoted by \( Re[\cdot] \) and applied to the second term, ensures that the contribution of the constraint to the cost function is likewise real valued. Most important cost function \( J(n) \) which is a quadratic function in \( \hat{w}(n+1) \), as is shown by expanding into

\[ J(n) = (\hat{w}(n+1) - \hat{w}(n))^H (\hat{w}(n+1) - \hat{w}(n)) + Re[\lambda^*(d(n) - \hat{w}(n+1)u(n))] \]  

(4.40)

To find the optimum value of the updated weight vector that minimizes the cost function \( J(n) \), procedure is as follows [20].

By differentiating the cost function \( J(n) \) with respect to \( \hat{w}(n+1) \). Then, following the rule for differentiating a real valued function with respect to a complex valued weight vector as shown,

\[ 1. \quad \frac{\partial J(n)}{\partial \hat{w}^*(n+1)} = 2(\hat{w}(n+1) - \hat{w}(n)) - \lambda^* u(n) \]  

(4.41)

Setting this result equal to zero and solving for the optimum value \( \hat{w}(n+1) \),

\[ \hat{w}(n+1) = \hat{w}(n) + 1/2\lambda^* u(n) \]  

(4.42)

Solve for the unknown multiplier \( \lambda \) by substituting the result of step 1 [i.e., the weight vector \( \hat{w}(n+1) \)] into the constraint of formula. Doing the substitution, first it can be written,

\[ d(n) = \hat{w}^H(n+1) u(n) \]  

(4.43)

\[ = (\hat{w}(n) + 1/2\lambda^* u(n))^H u(n) \]
\[
= \hat{w}(n)u(n) + \frac{1}{2} \lambda u^H(n)u(n)
\]

= \hat{w}(n)u(n) + \frac{1}{2} \lambda \|u(n)\|^2

Then, solving for \(\lambda\), it can be obtained that,

\[
\lambda = \frac{2e(n)}{\|u(n)\|^2}
\] (4.44)

Where

\[
e(n) = d(n) - \hat{w}(n)^Hu(n)
\] (4.45)

is the error signal.

2. Combine the results of steps 1 and 2 to prepare the optimal value of the incremental change, \(\delta \hat{w}(n + 1)\).

\[
\delta \hat{w}(n+1) = \hat{w}(n + 1) - \hat{w}(n)
\] (4.46)

\[
= 1/\|u(n)\|^2 u(n)e^*(n)
\]

In order to work out control over the change in the tap weight vector in gradual iteration process by keeping direction constant for the vector. By introducing a positive real scaling factor denoted by \(\mu\). That is, it can be redefined the change simply as

\[
\delta \hat{w}(n + 1) = \hat{w}(n + 1) - \hat{w}(n)
\] (4.47)

\[
= \left[\frac{\mu}{\|u(n)\|^2}\right] u(n)e^*(n)
\] (4.48)

Equivalently, it can be written that,

\[
\hat{w}(n+1) = \hat{w}(n) + \left[\frac{\mu}{\|u(n)\|^2}\right] u(n)e^*(n)
\] (4.49)

Indeed, this is the necessary recursion for calculation of the M by 1 tap weight vector in the normalized LMS algorithm. Above equation justifies why term normalized is used in this case.: Production of different vectors like \(u(n)\) and \(e^*(n)\) is achieved. That product is normalized with respect to the squared Euclidean norm of the tap input vector \(u(n)\).

Comparing the recursion of equation for the normalized LMS filter with that of the conventional LMS filter, the following observations might be taken [20].
The adaptation constant is different in both of the algorithm. For LMS it is dimensionless and for NLMS it is with dimensions of inverse power.

Setting $\mu(n) = \frac{\hat{\mu}}{||u(n)||^2}$ It can be viewed that the normalized LMS filter as an LMS filter with a time varying step size parameter.

Most prominently, the NLMS algorithm exhibits potentially faster rate of convergence than that of the standard LMS algorithm for uncorrelated as well correlated input data.

**Table 4.2 Summary of normalized LMS filter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>number of taps (i.e. filter length)</td>
</tr>
<tr>
<td>$\hat{\mu}$</td>
<td>adaptation constant</td>
</tr>
<tr>
<td>$0 &lt; \hat{\mu} &lt; 2$</td>
<td></td>
</tr>
<tr>
<td>$E[</td>
<td>u(n)^2</td>
</tr>
</tbody>
</table>

Where

$E[|e(n)^2|] = \text{error signal power}$

$E[|u(n)^2|] = \text{input signal power}$

$D(n) = \text{mean square deviation}$

Initialization if prior knowledge about the tap weight vector $\text{w}(n)$ is available, that knowledge can be used to select an appropriate value for $\text{w}(0) = 0$.

(a) Given: $u(n) = M$ by 1 tap input vector at time $n$

$\text{d}(n) = \text{desired response at time step } n$

(b) To be computed: $\text{w}(n+1) = \text{estimate of tap weight vector at time step } n+1$

Computation: for $n = 0,1,2,…$ compute

$e(n) = d(n) - \text{w}^H(n)u(n)$

$\text{w}(n+1) = \text{w}(n) + \frac{\hat{\mu} u(n)e^*(n)}{||u(n)||^2}$
Conventional LMS suffers from problem of gradient noise removal and its increase value damage the quality of system while the normalized LMS filter arises issue when the tap-input vector $u(n)$ is small, numerical calculation difficulties may arise because then and it can categorized with a small value for the square norm $\| u(n) \|^2$. As a solution modified version of the calculation is mentioned here.

$$\hat{w}(n+1) = \hat{w}(n) + \left[ \frac{\mu}{\delta + \| u(n) \|^2} \right] u(n) e^*(n) \quad (4.50)$$

Where $\delta > 0$ for $\delta = 0$, equation reduces to the form equation The normalized LMS filter is summarized in Table 4.2

### 4.3.2 Stability of the Normalized LMS Filter

Basically mechanism describe is responsible in the generation of wanted which is reproduced here for convenience of presentation [21].

$$d(n) = w^H u(n) + v(n) \quad (4.51)$$

In this equation, $w$ is the model’s unknown parameter vector and $v(n)$ is the additive disturbance. The tap-weight vector $\hat{w}(n)$ computed by the normalized LMS filter is an estimate of $w$. The uneveness among to vector named $w$ and $\hat{w}(n)$ which is accounted and considered by the weight-error vector

$$\varepsilon(n) = w - \hat{w}(n) \quad (4.52)$$

In further process subtracting iteration from $w$,

$$\varepsilon(n+1) = \varepsilon(n) - \mu / \left[ \| u(n) \|^2 \right] u(n) e^*(n) \quad (4.53)$$

As already stated, the main concept of a normalized LMS filter is that of reducing the increased change $\delta \hat{w}(n+1)$ in the tap weight vector of the filter to another next computation $n+1$, subject to a constraints imposed on the updated tap weight vector $\hat{w}(n+1)$. In light of this idea, it is logical that the stability analysis of the normalized LMS filter on the basis of mean square deviation as follows,

$$D(n) = E \left[ \| \varepsilon(n) \|^2 \right] \quad (4.54)$$
Mathematical process described below uses Euclidean norms of both sides and then application of adjustment as well as taking expectations, it is possible to write as under,

\[
\mathcal{D}(n+1) - \mathcal{D}(n) = \frac{\hat{\mu}^2}{\mathbb{E}} \left[ \frac{|e(n)|^2}{\|u(n)\|^2} \right] - 2\hat{\mu} \mathbb{E} \left\{ \text{Re} \left[ \frac{\xi_u(n) e^*(n)}{\|u(n)\|^2} \right] \right\}
\]  

(4.55)

Where \( \xi_u(n) \) is considered as the undisturbed error signal and can be cleared by

\[
\xi_u(n) = (w - \hat{w}(n))^H u(n)
\]

(4.56)

\[
= \mathcal{E}^H(n) u(n)
\]

It can be observed that the mean square deviation \( \mathcal{D}(n) \) decreases in exponential manner with higher number of iterations \( n \), and the NLMS filter gets stability in the mean square error and it gives that the normalized step size parameter \( \hat{\mu} \) is bounded as follows:

\[
0 < \hat{\mu} < \frac{\text{Re} \left[ \mathbb{E} \left[ \xi_u(n) e^*/\|u(n)\|^2 \right] \right]}{\mathbb{E} \left[ |e^2|/\|u(n)\|^2 \right]}
\]  

(4.57)

\[
\hat{\mu}_{opt} = \frac{\text{Re} \left[ \mathbb{E} \left[ \xi_u(n) e^*/\|u(n)\|^2 \right] \right]}{\mathbb{E} \left[ |e^2|/\|u(n)\|^2 \right]}
\]  

(4.58)

From above equation, it can also be concluded that highest value of the mean square deviation \( \mathcal{D}(n) \) is found at the center of the interval defined therein. After process optimal step size is as under.

### 4.3.3 Special Environment of Real Valued Data

For the case of real valued date the normalized LMS algorithm takes the form [26]

\[
\hat{w}(n+1) = \hat{w}(n) + \frac{\hat{\mu}}{\|u(n)\|^2} u(n) e(n).
\]

(4.59)

Likewise, the optimal step size parameter of equations reduces to \( \hat{\mu}_{opt} \) tractable, three assumptions can be introduced.
To make the computation of $\mu_{\text{opt}}$ tractable, three assumptions can be introduced:

**Assumption I** The noticed variation in the given input signal energy $\|u(n)\|^2$ in successive iteration process are less enough to validate the approximations

$$E \left[ \frac{\xi_u(n)e(n)}{\|u(n)\|^2} \right] \approx \frac{E[\xi_u(n)e(n)]}{E[\|u(n)\|^2]}$$  \hfill (4.61)

and

$$E \left[ \frac{e^2(n)}{\|u(n)\|^2} \right] \approx \frac{E[e^2(n)]}{E[\|u(n)\|^2]}$$  \hfill (4.62)

Correspondingly, the formula of equation approximates to

$$\mu_{\text{opt}} \approx \frac{E[\xi_u(n)e(n)]}{E[e^2(n)]}$$  \hfill (4.63)

**Assumption II** In the multiple regression the undisturbed error signal $\xi_u(n)$ is uncorrelated with the disturbance (noise) $v(n)$ for the descried response $d(n)$.

The disturbed error signal $e(n)$ is related to the undisturbed error signal $\xi_u(n)$,

$$e(n) = \xi_u(n) + v(n)$$  \hfill (4.64)

Using above equation and then invoking Assumption II,

$$E [\xi_u(n)e(n)] = E [\xi_u(n)(\xi_u(n) + v(n))]$$  \hfill (4.65)

$$= E [\xi^2_u(n)]$$

By simplifying equation, the formula for the optimal step size to

$$\mu_{\text{opt}} \approx \frac{E[\xi^2_u(n)]}{E[e^2(n)]}$$  \hfill (4.66)
Unlike the disturbed error signal \( e(n) \), the undisturbed error signal \( \xi_u(n) \) is inaccessible and, therefore, not directly measurable. To overcome this computational difficulty, last assumption can be introduced.

Assumption III It mandatory to note that the input signal spectral content is basically flat over a frequency band larger than that engaged by each element of the weight error vector \( \varepsilon(n) \), as a consequence by justifying the approximation

\[
E[\xi_u^2(n)] = E\left[\|\varepsilon(n)^T u(n)\|^2\right]
\]

\[
= E\left[\|\varepsilon(n)\|^2\right] E[u^2(n)]
\]

\[
= \mathcal{D}(n) E[u^2(n)]
\]

where \( \mathcal{D}(n) \) is the mean square deviation. Note that the approximate formula of equation involves the input signal \( u(n) \) rather than the tap input vector \( u(n) \).

Assumption III is a statement of the low pass filtering action of the LMS filter. Thus, using above equations the approximation can be as under

\[
\hat{\mu}_{opt} \approx \frac{\mathcal{D}(n) E[u^2(n)]}{E[e^2(n)]}
\]

(4.68)

The practical virtue of the approximate formula of \( \hat{\mu}_{opt} \) defined in above equation is borne out in the fact that simulations as well as real time implementations have shown that it provides a good approximation for \( \hat{\mu}_{opt} \) for the case of large filter lengths and speech inputs.

### 4.4 Recursive Least Squares Adaptive Filters

In feedback mechanism implementation of the method of the least squares, one can start the computation with previously enumerated initial condition and by using the information contained in new data samples to update the old estimates. Length of observed data is variable. Accordingly, it can be expressed that the cost function to be minimized as \( \mathcal{E}(n) \) [19]. Thus it can be written that as shown:
Here is can be observed that $e(i)$ is the difference. This difference is calculated between the necessary reaction $d(i)$ and the outcome $y(i)$ which generated by a transversal filter and its tap weights are at time $i$ equal $u(i)$, $u(i-1)$, … $u(i-M+1)$, as in Figure 4.6 that is,

\[
\begin{align*}
\widehat{w}(n) &= d(i) - y(i) \\
 &= d(i) - w^H(n) u(i)
\end{align*}
\]

where $u(i)$ is the tap input vector at time defined by

\[
u(i) = [u(i), u(i-1), ..., u(i-m)]^T
\]

And $w(n)$ is the tap weight vector at time $n$, defined by

\[
w(n) = [w_0(n), w_1(n), ..., w_{M-1}(n)]^T
\]

Consider that the tap weights of the transversal filter is basically fixed during the observation interval

\[1 \leq i \leq n\]

for which the cost function $\mathcal{E}(n)$ is defined.

The weighting factor $\beta(n, i)$ in above equation it has the property that
\[ 0 < \beta(n,i) \leq 1, i = 1, 2, \ldots, n. \] 

(4.73)

The use of the weighting factor \( \beta(n,i) \), in general, is manly required to verify that the data in the old past are omitted or forgotten in order to undergo the chances of the statistical variations of the observable data. This condition arises mainly when the filter works in nonstationary environment. Generally usage of a special form of weighting is the exponential weighting factor, or forgetting factor, which can be narrated by

\[ \beta(n,i) = \lambda^{n-i}, i = 1, 2, \ldots, n, \] 

(4.74)

Mainly \( \lambda \) is a positive constant. Here three cases are possible with that. When \( \lambda = 1 \), it is the usual mechanism of least squares. The inverse of \( 1 - \lambda \) is, roughly speaking, a memory of algorithm and it is measured in that form of the algorithm [28]. The special case \( \lambda = 1 \) corresponds to infinite memory.

4.4.1 Regularization

Least square estimation, input data are given and it is contained a tap input vector \( u(n) \) and the according desired wanted response \( d(n) \) for changing [19].

The ill posed nature of least squares estimation is due to the following reasons:

- To renovate the input output mapping uniquely the available data is in form of insufficient information as a input data.
- The occurrence of noise or imprecision in the input data which can not be avoided adds uncertainty to the reconstructed input output mapping.

For generating estimation problem “well posed,” some form of prior information about the input output mapping is needed. This, in turn, means that the formulation of the cost function must be expanded to take the prior information into account.

To satisfy that objective, it can be expanded the cost function to be minimized as the sum of two components:

\[ E(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 + \delta \lambda^n \| w(n) \|^2 \] 

(4.75)
Here, the use of pre windowing is assumed. Cost function can be defined as follows:

The weighted vectors and its squares,

\[ \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 = \sum_{i=1}^{n} \lambda^{n-i} |d(i) - w^H(n) u(i)|^2 \]  

(4.76)

Error is data which is not independent. Consideration of this data is on exponential based and on that weighted error between the desired response \( d(i) \) and the actual response of the filter, \( y(i) \), because of this tap weight vector can be correlated.

\[ Y(i) = w^H(n) u(i) \]  

(4.77)

1. A regularizing term,

\[ \delta \lambda^n \| w(n) \|^2 = \delta \lambda^n w^H(n) w(n) \]  

(4.78)

Where \( \delta \) is a positive real number and it is known as regularization parameter. Excluding the factor \( \delta \lambda^n \), the regularizing term based on solely on the tap weight vector \( w(n) \). The term is comprised in the cost function to stabilize the solution.

In a strict sense, the term \( \delta \lambda^n \| w(n) \|^2 \) is a “rough” form of regularization for two reasons. First, the exponential weighting factor \( \lambda \) lies in the interval \( 0 < \lambda \leq 1 \); hence, for \( \lambda \) less than unity, \( \lambda^2 \) tends to zero for large \( n \), which means that the beneficial effect of adding \( \delta \lambda^n \| \hat{w}(n) \|^2 \) to the cost function is forgotten with time. Second, and more important, the regularizing term should be of the form \( \delta \|DF(\hat{w})\|^2 \), where \( F(\hat{w}) \) is the input output map realized by the RLS filter and \( D \) is the differential operator.

### 4.4.2 Reformulation of the Normal Equations

Expanding above equation and collecting terms, it can be found that when in the cost function the impact of including the regularizing term, \( \mathcal{E}(n) \) is equivalent to a reformulation of the \( M \) by \( M \) time average correlation matrix of the tap input vector:

\[ \Phi(n) = \sum_{i=1}^{n} \lambda^{n-i} u(i) u^H(i) + \delta \lambda^n I \]  

(4.79)

I can be defined as identity matrix with length \( M \).
The M by 1 time average cross correlation vector \( z(n) \) between the tap inputs of the transversal filter and the desired response is unaffected by the use of regularization [20].

\[
Z(n) = \sum_{i=1}^{n} \lambda^{n-i} u(i) d^*(i) \tag{4.80}
\]

Where, again, the use of pre windowing is assumed.

The optimum value of the M by 1 tap weight vector, for which the cost function attains its minimum value is as under as per the method of least squares.

\[
\Phi(n) \hat{w}(n) = z(n) \tag{4.81}
\]

### 4.4.3 Recursive Computations of \( \Phi(n) \) and \( z(n) \)

Isolating the term corresponding to \( i = n \) from the remaining of the accumulation and on the other side of the equality it can be written as

\[
\Phi(n) = \lambda \sum_{i=1}^{n-1} \left( \lambda^{n-1-i} u(i) u^H(i) + \delta \lambda^{n-1} I \right) + u(n) u^H(n) \tag{4.82}
\]

Hence, the following recursion for updating the value of the correlation matrix of the tap inputs may have [20]:

\[
\Phi(n) = \lambda \Phi(n-1) + u(n)u^H(n) \tag{4.83}
\]

Here, \( \Phi(n-1) \) is the “old” value of the correlation matrix, and the matrix product \( u(n)u^H(n) \) plays the role of a “correlation” term in the updating operation. Note that the recursion of above equation holds, irrespective of the initializing condition.

Similarly, above equation may be used to derive the following recursion for updating the cross correlation vector between the tap inputs and the desired response:

\[
z(n) = \lambda z(n-1) + u(n)d^*(n) \tag{4.84}
\]

It is necessary to determine the inverse of the correlation matrix \( \Phi(n) \) to compute the least square estimate for tap weight vector. In practice, however, it can be tried usually to avoid performing such an operation, as it can be quite time consuming. Also, it is preferable to compute the least squares estimate \( \hat{w}(n) \) for the tap weight vector...
recursively for n=1, 2… ∞. It can be realized both of these objectives by using a basic result in matrix algebra known as the matrix inversion lemma.

4.4.4 The Matrix Inversion Lemma

Let A and B be two positive define M by M matrices related by

\[ A = B^{-1} + CD^{-1}CH \] (4.85)

Where D is a positive-definite N-by-M matrix and C is an M-by-N matrix. According to the matrix inversion lemma, the inverse of the matrix A as may be expressed as,

\[ A^{-1} = B - BC(D + C^HBC)^{-1}C^HB \] (4.86)

The proof of this lemma is established by multiplying above equations and recognizing that the product of a square matrix and its inverse is equal to the identity matrix. The matrix inversion lemma states that if a matrix A is given, as defined in above equations, it can be determined its inverse \( A^{-1} \) by using the relation expressed in above equation. In effect, the lemma is described by that pair of equation [21].

It is very important to quote that, equation shown below describes the operation of the algorithm, whereby priori estimation error would be computed in the transversal filter. Further step shows the adaptive operation of the algorithm, in which the tap weight vector is changed by incrementing its previous value by an amount equal to the product of the complex conjugate of the priori estimation error \( \xi(n) \) and the time varying gain vector \( k(n) \), hence the name “gain vector”. Step coated in the next line helps to update the value of the gain vector itself. Most important characteristics of RLS algorithm is that the inversion of the correlation matrix \( \Phi(n) \) is replaced at each step by a simple scalar division. RLS algorithm is summarized as shown in Table 4.3.
Table 4.3 Summary of RLS algorithm

<table>
<thead>
<tr>
<th>Initialize the algorithm by setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{w}(0) = 0, )</td>
</tr>
<tr>
<td>( P(0) = \delta^{-1}I, )</td>
</tr>
<tr>
<td>and</td>
</tr>
</tbody>
</table>
| \( \delta = \begin{cases} 
\text{small positive constant for high SNR} \\
\text{Large positive constant for low SNR}
\end{cases} \) |
| For each instant of time, n = 1, 2, \ldots, compute |
| \( k(n) = \frac{\Pi(n)}{\lambda + u^H(n) \Pi(n)} \) |
| \( \Pi(n) = P(n-1) u(n), \) |

\( \xi(n) = d(n) - \hat{w}^H(n-1) u(n), \)

\( \hat{w}(n) = \hat{w}(n-1) + k(n) \xi^*(n), \)

and

\( p(n) = \lambda^{-1}p(n-1) - \lambda^{-1}k(n) u^H(n) p(n-1). \)

Table 4.3 describes summary of RLS algorithm. The summary shows, the calculation of the gain vector \( k(n) \) takings in two stages. RLS algorithm concept and signal flow graph is shown in Figure 4.7 and Figure 4.8 in detail.

- Initially, an intermediate quality which can be shown with \( \Pi(n) \), is computed.
- In next step, \( \Pi(n) \) is utilized to compute \( k(n) \).
This two stage computation of $k(n)$ is favored over the direct calculation of $k(n)$ using above equation from a finite precision arithmetic point of view.

To initialize the RLS filter,:  
- The initial weight vector $\hat{w}(0)$ for which it is necessary to set $\hat{w}(0) = 0$.  
- The initial correlation matrix $\Phi(0)$. Setting $n = 0$ in above equation it can be found with the use of pre windowing, following can be obtained,
\[
\Phi(0) = \delta I \tag{4.87}
\]

where $\delta$ is the regularization. The parameter $\delta$ should be assigned a small value for high signal to noise ratio (SNR) and a large value for low SNR, which may be justified on regularization grounds [27].
4.4.5 Selection of the Regularization Parameter

The convergence behavior of the RLS algorithm was evaluated for a stationary environment, with particular reference to two variable parameters:

- The signal to noise ratio (SNR) of the tap input data, which is determined by the prevalent operation conditions.
- The regularization parameter $\delta$, which is under the designer’s control.

Let $F(x)$ denote a matrix function of $x$, and let $f(x)$ denote a nonnegative scalar function of $x$, where the variable $x$ assumes values in some set $x$. Following definitions might be introduced where there exist constants $c_1$ and $c_2$ that are independent of the variable $x$, such that

$$F(x) = \theta (f)$$

and where $\| F(x) \|$ is the matrix norm of $F(x)$, which is itself defined by

$$c_1 f(x) \leq \| F(x) \| \leq c_2 f(x) \quad \text{for all} \ x \in x$$

(4.89)

The significance of the definition introduced in above derivation, it will become apparent presently.

$$\| F(x) \| = (\text{tr} [F^H (x) F(x)])^{1/2}$$

(4.90)

The initialization of the RLS filter includes setting the initial value of the time average correlation matrix

$$\Phi (0) = \delta I$$

(4.91)

The dependence of the regularization parameter $\delta$ on SNR is given in detailed. In particular, $\Phi(0)$ is reformulated as

$$\Phi (0) = \mu^2 R_0$$

(4.92)
Where \( \mu = 1 - \lambda \) \hspace{1cm} (4.93)

and \( R_0 \) is a deterministic positive definite matrix defined by

\[
R_0 = \sigma_u^2 I
\]  \hspace{1cm} (4.94)

in which \( \sigma_u^2 \) is the variance of a date sample \( u(n) \). Thus, according to above equation), the regularization parameter \( \delta \) is defined by

\[
\delta = \sigma_u^2 \mu^\alpha
\]  \hspace{1cm} (4.95)

The parameter \( \alpha \) provides the mathematical basis for distinguishing the initial value of the correlation matrix \( \Phi(n) \) as small, medium, or large. In particular, for situations in which

\[
\mu \in [0, \mu_0] \quad \text{with} \quad \mu_0 \ll 1
\]  \hspace{1cm} (4.96)

It may distinguish three scenarios in light of the definition introduced [28].

1. \( \alpha > 0 \), which corresponds to a small initial value \( \Phi(0) \).
2. \( 0 > \alpha \geq -1 \), which corresponds to a medium initial value \( \Phi(0) \).
3. \( -1 \geq \alpha \), which corresponds to a large value \( \Phi(0) \).

With these definitions and the three distinct initial conditions at hand, it can be summarized on the selection of the regularization parameter \( \delta \) in initializing the RLS algorithm for situations [28].

1. High SNR

When the noise level in tap inputs is low i.e., the input SNR is on the order of 30dB, the RLS algorithm exhibits an exceptionally fast rate of convergence, provided that the correlation matrix is initialized with a small enough norm. Typically, this requirement is satisfied by setting \( \alpha = 1 \). As \( \alpha \) is reduced toward zero, the convergence behavior of the RLS algorithm deteriorates.
2. Medium SNR

In a medium SNR environment, i.e., the input SNR is on the order of 10dB, the rate of convergence of the RLS algorithm is worse than the optimal rate for the high SNR case, but the convergence behavior of the RLS algorithm is essentially insensitive to variations in the matrix norm of $\Phi(0)$ for $-1 \leq \alpha < 0$.

3. Low SNR

Finally, when the noise level in the tap inputs is high i.e., the input SNR is on the order of -10 dB or less, it is preferable to initialize the RLS algorithm with a correlation matrix $\Phi(0)$ with a large matrix norm (i.e., $\alpha \leq -1$), since this condition may yield the best overall performance.

These remarks hold for a stationary environment or a slowly time varying one. If, however, there is an abrupt change in the state of the environment and the change takes changes as renewed initialization with a “large” initial $\Phi(0)$ wherein $n=0$ corresponds to the instant at which the environment switched to a new state. In such a situation, the recommended practice is to stop the operation of the RLS filter and restart a new by initializing in with a small $\Phi(0)$.

4.4.6 Convergence Analysis of RLS Algorithm

The convergence behavior of the RLS algorithm in a stationary environment, assuming that the exponential weighting factor $\lambda$ is unity. To pave the way for the discussion, three assumptions can be make, all of which are reasonable in their own ways [19].

Assumption I The desired response $d(n)$ an the tap input vector $u(n)$ are related by the multiple linear regression model

$$d(n) = w_0^H u(n) + e_0(n)$$

(4.97)

here $w_0$ can be known as regression vector and $e_0(n)$ is noise measurement vector. The noise $e_0(n)$ is white with zero mean and variance $\sigma_0^2$, which makes it independent of the
repressor \( u(n) \). The relationship expressed in above equation is depicted in above figure, which is a reproduction basic concept.

Assumption II The input signal \( u(n) \) is drawn from a stochastic process, which is ergodic in the autocorrelation function. The implication of Assumption II is that time averages for ensemble averages may be substituted. In particular, it may be expressed the ensemble average correlation matrix of the input vector \( u(n) \) as

\[
R \approx \frac{1}{n} \Phi(n) \quad \text{for } n > M \quad (4.98)
\]

Where \( \Phi(n) \) is the time average correlation matrix of \( u(n) \) and the requirement \( n > M \) ensure that the input signal spreads across all the taps of the transversal filter. The approximation of above equation improves with an increasing number of time steps \( n \).

Assumption III The functions in the weight error vector \( \varepsilon(n) \) are slow compared with those of the input signal vector \( u(n) \). The justification for Assumption III is to recognize that the weight error vector \( \varepsilon(n) \) is the accumulation of a series of changes extending over \( n \) iterations of the RLS algorithm. This property is shown by

\[
e(n) = w_0(n) - \hat{w}(n)
\]

\[
= \varepsilon_0 - \sum_{i=1}^{n-1} k(i) \xi^*(i) \quad (4.100)
\]

Algorithm both \( k(i) \) and \( \xi(i) \) depend on \( u(i) \), the summation in equation has a “smoothing” effect on \( \varepsilon(n) \). The effect, the RLS filter acts as a time varying low pass filter. No further assumption on the statistical characterization of \( u(n) \) and \( d(n) \) are made in what follows.

4.4.7 Convergence of the RLS algorithm in the mean value

Solving the normal equation for \( \hat{w}(n) \), it can be written,

\[
\hat{w}(n) = \Phi^{-1}(n) z(n) \quad n > M \quad (4.101)
\]

Where, for \( \lambda = 1 \),
\[ \Phi(n) = \sum_{i=0}^{n} u(i) u^H(i) + \Phi(0) \]  
\[ (4.102) \]

and

\[ z(n) = \sum_{i=0}^{n} u(i) d^*(i) \]  
\[ (4.103) \]

Finally after simplification,

\[ z(n) = u(i) u^H(i) w_0 + \sum_{i=0}^{n} u(i) e^0(i) \]  
\[ (4.104) \]

\[ = \Phi(n) \hat{w}_0 + \sum_{i=0}^{n} u(i) e^0(i) \]

\[ \hat{w}(n) = \Phi^{-1}(n) \Phi(n) w_0 - \Phi^{-1}(n) \Phi(n) w_0 + \Phi^{-1}(n) \sum_{i=0}^{n} u(i) e^0(i) \]  
\[ (4.105) \]

\[ = w_0 - \Phi^{-1}(n) \Phi(n) w_0 + \Phi^{-1}(n) \sum_{i=0}^{n} u(i) e^0(i) \]  
\[ (4.106) \]

Taking the expectation of both sides of above equation and invoking Assumption I and II, it can be written

\[ E[\hat{w}(n)] \approx w_0 - 1/n R^{-1} w_0 \]  
\[ = w_0 - 1/n R^{-1} w_0 \]  
\[ = w_0 - 1/n P, n > M, \]

Where p is the ensemble average cross correlation vector between the desired response \( d(n) \) and input vector \( u(n) \) [20]. Equations state that the RLS algorithm is convergent in the mean value. For finite \( n \) greater than the filter length \( M \), the estimate \( \hat{w}(n) \) is biased, due to the initialization of the algorithm by setting \( \Phi(0) = \delta I \), but the bias decreases zero as \( n \) approaches infinity.

**4.4.8 Mean Square Deviation of the RLS Algorithm**

The weight error correlation matrix is defined by

\[ K(n) = E[\varepsilon(n) \varepsilon^H(n)] \]  
\[ (4.108) \]

\[ = E[(w_0 - \hat{w})(n)(w_0 - \hat{w})(n)^H] \]

The following two important observations for \( n > M \):

\[ \]
1. The mean square deviation $D(n)$ is magnified by the inverse of the smallest eigenvalue $\lambda_{\text{min}}$. Hence, to a first order of approximation, the sensitivity of the RLS algorithm to eigenvalues spread is determined initially in proportion to the inverse of the smallest eigenvalues. Therefore, ill conditioned least squares problems may lead to poor convergence properties.

2. The mean square deviation $D(n)$ decays almost linearly with the number of iterations, $n$. Hence the estimate $\hat{w}(n)$ produced by the RLS algorithm converges in the norm (i.e., mean square) to the parameter vector $w_0$ of the multiple linear regression model almost linearly with time.

4.4.9 Ensemble Average Learning Curve of the RLS Algorithm

In the RLS algorithm, there are two types of error: the a priori estimation error $\xi(n)$ and the a posteriori estimation error $e(n)$. It can be found that the mean square values of these two errors vary differently with time $n$. At time $n = 1$, the mean square value of $\xi(n)$ becomes large equal to the mean square value of the desired response $d(n)$ and then decays with increasing $n$. The mean square value of $e(n)$, on the other hand, becomes small at $n = 1$ and then rises with increasing $n$, until a point is reached for large $n$ for which $e(n)$ is equal to $\xi(n)$ [21]. Accordingly, the choice of $\xi(n)$ as the error of interest yields a learning curve for the RLS algorithm which has the same general shape as that for the LMS algorithm. By choosing $\xi(n)$ thus, a direct graphical comparison can be made between the learning curves of the RLS and LMS algorithms. A computation of the ensemble average learning curve of the RLS algorithm on the a priori estimation error $\xi(n)$. The convergence analysis of the RLS algorithm presented here assumes that the exponential weighting factor equals unity.

The following observations can be concluded:

1. The ensemble average learning curve of the RLS algorithm converges in about $2M$ iterations, where $M$ is the filter length. This means that the rate of convergence of the RLS algorithm is typically an order of magnitude faster than that of the LMS algorithm.
2. As the number of iterations, \( n \), approaches infinity, the mean square error \( J'(n) \) approaches a final value equal to the variance \( \sigma^2_0 \) of the measurement error \( e_0(n) \). In other words, the RLS algorithm produces zero excess mean square error or zero misadjustment.

3. Convergence of the RLS algorithm in the mean square is independent of the eigenvalues of the ensemble average correlation matrix \( R \) of the input vector \( u(n) \).