ADAPTIVE DIGITAL FILTERS: PART II

PERFORMANCE ANALYSIS OF LMS METHOD

Excess Mean Square Error

To reduce the mean square error time constant, one must choose a step size μ as large as possible, consistent with the constraints of convergence. However, a higher choice of μ causes higher excess mean square error. Hence there is a trade off between convergence speed and steady-state accuracy, while selecting μ.

In LMS method, $\hat{\nabla}_n = -2E_nR_n$, which differs from the true value, $\nabla_n = 2(RH_n - P)$. Error in the estimate of the gradient of the mse can be modeled as an additive noise term: $\nabla[\xi_n(H)] = \nabla_n = \hat{\nabla}_n + v_n$. The noise term v_n causes the steady-state value of the mse to be larger than the theoretical minimum value and the difference is referred to as the *excess mse*.

Excess mean square error: $\xi_{n_{excess}} = \xi_n(H_n) - \xi_{\min}$

This can be shown, in terms of weight variations as,

$$\xi_{n_{excess}} = V_n^T R V_n \qquad (V = H_n - H_w)$$

It can be approximated as:

$$\xi_{n_{excess}} \approx \mu \xi_{\min} M P_R$$

To avoid substantial computational effort required to determine the ξ_{min}, a normalized version of the *excess mse* is often used.

The misadjustment factor of the LMS method:

$$M_f = \frac{\xi_{n_{excess}}}{\xi_{\min}} \approx \mu M P_R$$

The normalized excess mean square error increases with μ, the filter order and average power of the input. Hence, to reduce M_f, we must decrease μ. But to reduce τ_{mse}, we must increase μ. Hence there will be a trade off.

Table: Performance Characteristics of an Adaptive Filter.

Property	**Value**
Convergence range	$0 < \mu < \frac{1}{MP_R}$
Learning-curve time constant	$\tau_{mse} \approx \frac{1}{4\mu\lambda_{av}}$
Excess mean square error	$\xi_{n_{excess}} \approx \mu M \xi_{\min} P_R$

MODIFIED LMS METHODS

Several modifications to the basic LMS algorithm have been proposed to improve the performance:

1. Normalized LMS Method:

The *Normalized LMS method* was proposed to develop a version of the LMS method that has a step size (α) that does not depend on average power P_R or filter order M. Here the learning rule is:

$$H_{n+1} = H_n + 2\mu_n E_n R_n, \qquad \left(R_n = \left[R_n, R_{n-1}, \cdots, R_{n-M+1}\right]\right), \quad n \geq 0$$

Here the difference with the basic LMS method is that the step size μ_n is no longer constant, but varies with time.

Here $\mu_n = \dfrac{\alpha}{M\hat{P}_{R(n)}}$

where $\hat{P}_{R(n)}$ = running estimate of the average input power.

If $\hat{P}_{R(n)}$ is replaced by the true average power P_R, then, according to the table given before, the range of ***constant*** step sizes needed to ensure convergence is $0 < \alpha < 1$. In this sense the step size is normalized. Here a single value of α can be used, independent of the filter size and the input power.

Usually a **rectangular window** or **running-average filter** is used to estimate the average input power in a simple manner. For an *N*th order running –average filter:

$$\hat{P}_{R(n)} = \frac{1}{N}\sum_{i=0}^{N-1} R_{n-i}^2$$

2. **Correlation LMS Method:**

This method was proposed to avoid the trade-off required while selecting μ. Here one can choose a large step size while striving for convergence, and this can be followed by the choice of a small step size, once convergence has been achieved. Then the **main objective** is to detect *when* convergence has been achieved. Theoretically it is possible to use H_w or ξ_{min} to detect convergence, but, because of the computational burden involved, it is always preferable to use a less direct means.

Let us assume that H_n has converged to H_w.

Then, we have:

$$E_n R_n = [P_n - R_n^T H] R_n$$

$$\therefore E[E_n R_n] = E[P_n R_n] - E[R_n R_n^T] H_w = P - RH_w = 0$$

(at the optimal weight,

$H_w = R^{-1}P$)

Now, $E[E_n R_n]$ gives the cross-correlation of the error with the reference input.

$\therefore$ Cross correlation of error with reference input,

$$P_{ER}(i) = 0, \qquad 0 \le i \le (M-1)$$

Hence, when the weight vector is optimal, the error is uncorrelated with the reference input. For the special case $i = 0$, the scalar relationship that holds when the LMS method has converged is: $E[E_n R_n] = 0$.

The **basic idea behind the correlation LMS method** is to **choose a step size that is directly proportional to the magnitude of** $E[E_n R_n]$, where both E_n and R_n present the scalar quantities at the current nth instant. This way the step size will become small when the LMS method has converged, but will be larger during the convergence process. One can estimate the expected value by using a running-average filter with input $E_n R_n$. However this is a computation heavy procedure because it requires storage of previous samples of E_n.

3. Leaky LMS Method:

When an input with poor spectral content is used, the LMS method can diverge with one or more elements of the weight vector growing without bound. An elegant way **to guard**

against this possibility is to **introduce a second term in the objective function**. Here we calculate an **augmented mean square error** as:

Augmented mse: $\xi_\gamma(H_n) = E[E_n^2] + \underbrace{\gamma H_n^T H_n}_{\text{penalty function term}}$ **,** $\gamma > 0$

The penalty function term tends to penalize the minimization process if we select a H_n for which $H_n^T H_n$ is large. A **simplified formulation of the learning rule for this method** is:

$$H_{n+1} = \nu H_n + 2\mu E_n R_n$$

where $\nu = 1 - 2\mu\gamma$ is called the ***leakage factor***. **When η = 1, leaky LMS method reduces to the basic LMS method**. If R_n = 0, then H_n "**leaks**" to zero at the rate $H_n = \gamma^n H_0$. Typically, the ***leakage factor*** is $0 < \nu < 1$, with $\nu \approx 1$. The inclusion of the ***leakage factor*** enhances the stability of the algorithm for a variety of inputs. However, it also increases the ***excess mse*** due to the inclusion of the penalty term.

THE RECURSIVE LEAST SQUARES (RLS) METHOD

RLS is an efficient alternative to the LMS method, which typically converges much faster than the LMS method, but at a cost of more computational effort per iteration. This method uses the **more general time-varying performance criterion**,

$$\xi_n(H) = \sum_{i=1}^{n} \gamma^{n-i} E_i^2 + \underbrace{\delta\gamma^n H^T H}_{\text{regularization term}}, \quad k \geq 1$$

γ = forgetting factor (an exponential weighting factor) , $0 < \gamma \leq 1$

δ = regularization parameter ($\delta > 0$)

The regularization term is very similar to the penalty function term of leaky LMS method. When $\gamma < 1$, it has the effect of reducing the contributions from errors in the remote past. When $\gamma = 1$ and $\delta = 0$, the performance criterion becomes proportional to ***mse*** at iteration n. The **solution for the optimal weight at iteration n** can be obtained as:

$$H_n = R_{(n)}^{-1} P_{(n)}$$

Unlike the LMS method, which asymptotically approaches the optimal weight vector using a gradient based search, the RLS method attempts to find the optimal weight at each iteration.

The Recursive Least Squares (RLS) Algorithm

Although the LMS algorithm has its strength in its computational simplicity, its weakness lies in slow convergence. The basic LMS algorithm has only a single adjustable parameter, the step size parameter μ. To obtain faster convergence, it is necessary to devise more complex algorithms, which involve additional parameters. Typically, we can attempt to **make use of an algorithm that involves M parameters, one for each of the eigenvalues** $\lambda_0, \lambda_1, \lambda_2, ..., \lambda_{M-1}$ **of the correlation matrix R**. An efficient alternative to the LMS method is called the **recursive least squares or RLS method**. The **RLS method** typically *converges much faster* than the LMS method, but at the cost of *more computational effort per iteration*. RLS algorithm is used in adaptive filters to find the filter coefficients that recursively produce the least squares (minimum of the sum of the absolute squared) of the error signal (difference between the desired and the actual signal). This is in contrast to the LMS and many other steepest descent based algorithms that operate on the statistical approach based on the ***mse*** criterion. In **RLS method**, we deal directly with the data sequence and obtain estimates of correlations from the data.

For a recursive algorithm, it is necessary to introduce a time index in the filter-coefficients or weight vector and the error sequence.

In matrix notation, $H_n^T = \left[h_{o(n)}, h_{1(n)}, h_{2(n)}, \ldots\ldots, h_{(M-1)(n)}\right]$

(M = order of the filter).

Similarly, the reference vector, i.e. the signal input to the filter is $R_n^T = \left[R_n, R_{n-1}, R_{n-2}, \ldots\ldots, R_{(n-M+1)}\right]$. Now the **RLS problem can be formulated on the basis of the performance criterion**:

$$\xi_n\left(H_n\right) = \sum_{k=1}^{n} \gamma^{n-k} E_k^2, \quad n \geq 1 \tag{1}$$

or on the basis of the **more general time-varying performance criterion**:

$$\xi_n\left(H_n\right) = \sum_{k=1}^{n} \gamma^{n-k} E_k^2 + \delta\gamma^n H_n^T H_n, \quad n \geq 1 \tag{2}$$

where $E_k = P_k - \hat{N}_k = P_k - H_k^T R_k$ (3)

is the difference between the desired i.e., the primary signal and the filter output signal, at the kth instant. The objective of the RLS filter is to minimize the performance function (also termed as a "cost function") by appropriately selecting the filter coefficients, i.e., weights H_n, updating the filter coefficients as new data arrive.

The ***exponential weighting factor***, $0 < \gamma \leq 1$, is called the ***forgetting factor***. The purpose of this factor is to weight most recent data samples more heavily and, when $\gamma < 1$, it reduces the contributions from errors in remote past. This allows the filter coefficients to adapt to time-varying statistical characteristics of the data. This is accomplished by the concept of exponential weighting. In equation (2), δ is called the ***regularization parameter*** ($\delta > 0$). The second term in equation (2) is similar to the penalty function term in the leaky LMS method. This tends to prevent solutions for which $H_n^T H_n$ grows arbitrarily large. This has the effect of improving the stability of the RLS algorithm. When $\gamma = 1$ (no exponential weighting) and $\delta = 0$ (no regularization), the performance criterion in equation (2) is proportional to the mean square error at time n. Now, considering **the generalized performance criterion**,

$$\begin{aligned}
\xi_n(H) &= \sum_{k=1}^{n} \gamma^{n-k} \left[P_k - H^T R_k\right]^2 + \delta\gamma^n H^T H \\
&= \sum_{k=1}^{n} \gamma^{n-k} \left[P_k^2 - 2P_k H^T R_k + \left(H^T R_k\right)^2\right] + \delta\gamma^n H^T H \\
&= \sum_{k=1}^{n} \gamma^{n-k} \left[P_k^2 - 2H^T P_k R_k\right] + \sum_{k=1}^{n} \gamma^{n-k} H^T R_k \left[R_k^T H\right] + \delta\gamma^n H^T H \\
&= \sum_{k=1}^{n} \gamma^{n-k} \left[P_k^2 - 2H^T P_k R_k\right] + H^T \left[\sum_{k=1}^{n} \gamma^{n-k} R_k R_k^T + \delta\gamma^n I\right] H
\end{aligned}$$

Let us introduce the following expressions for generalized versions of the auto-correlation matrix and cross-correlation vector, at time instant n.

$$R_{R(n)} \triangleq \sum_{k=1}^{n} \gamma^{n-k} R_k R_k^T + \delta \gamma^n I$$

$$P_{PR(n)} \triangleq \sum_{k=1}^{n} \gamma^{n-k} P_k R_k$$

Substitution of these quantities gives,

$$\xi_n(H) = \sum_{k=1}^{n} \gamma^{n-k} P_k^2 - 2H^T P_{PR(n)} + H^T R_{R(n)} H$$

Hence by obtaining the gradient vector of partial derivatives of $\xi_n(H)$ with respect to the elements of H,

$$\nabla[\xi_n(H)] = 2R_{R(n)} H - 2P_{PR(n)}$$

Hence, the optimal weight at time instant n is obtained from:

$$\nabla[\xi_n(H)] = 0 \text{ or } R_{R(n)} H = P_{PR(n)} \text{ or } H_n = R_{R(n)}^{-1} P_{PR(n)}$$

Choice of γ

Smaller the value of γ, smaller the contribution of the previous samples. This makes the filter more sensitive to recent samples, which causes more fluctuations in the filter coefficients. When γ = 1, the case is referred to as the ***growing window RLS algorithm***.

Recursive Formulation

Although we can obtain the optimal weight vector H_n by minimizing $\xi_n(H_n)$, the computational burden required to find H_n is enormous. This computational effort increases with increasing n. **The required computation can be made more economical if the solution can be reformulated in a *recursive manner***. Here **we start with the solution at (n–1)th iteration and add a correction factor to obtain the solution at nth iteration**, i.e.

$$H_n = H_{n-1} + \Delta H_{n-1}$$

To achieve this recursive formulation, we first need to express $R_{R(n)}$ in terms on $R_{R(n-1)}$.

$$\begin{aligned} R_{R(n)} &= \gamma\left[\sum_{k=1}^{n}\gamma^{n-k-1}R_k R_k^T + \delta\gamma^{n-1}I\right] \\ &= \gamma\left[\sum_{k=1}^{n-1}\gamma^{n-k-1}R_k R_k^T + \delta\gamma^{n-1}I\right] + R_n R_n^T \\ &= \gamma R_{R(n-1)} + R_n R_n^T \qquad (n \ge 1), \end{aligned}$$

Hence the exponentially weighted and regularized auto-correlation matrix can be computed recursively.

An initial value of $R_{R(0)}$ is required to start the recursion process. One can set $R_{R(0)} = \delta I$ (assuming R_n as causal). Similarly, the generalized cross-correlation vector can be obtained in a recursive manner.

$$\begin{aligned} P_{PR(n)} &= \sum_{k=1}^{n}\gamma^{n-k}P_k R_k = \sum_{k=1}^{n-1}\gamma^{n-k}P_k R_k + \gamma^0 P_n R_n \\ &= \gamma P_{PR(n-1)} + P_n R_n, \qquad n \ge 1 \end{aligned}$$

These two equations are called the time-update equations for $\mathbf{R}_{\mathbf{R(n)}}$ and $\mathbf{P}_{\mathbf{PR(n)}}$. The recursive formulation for $P_{PR(n)}$ requires an initial value. This initial value is set as $P_{PR(0)} = 0$.

The time update equations greatly simplify computations of $R_{R(n)}$ and $P_{PR(n)}$. However **we still need to compute the inverse of $\mathbf{R}_{\mathbf{R(n)}}$ to solve the equation:** $R_{R(n)}H_n = P_{PR(n)}$**.** This is another computation heavy procedure. However, **it is also possible to compute $R_{R(n)}^{-1}$ recursively by utilizing the *matrix inversion lemma*** given as:

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(DA^{-1}B + C^{-1}\right)^{-1}DA^{-1}$$

We utilize this result to obtain $R_{R(n)}^{-1}$ from $R_{R(n-1)}^{-1}$.

Let $A = \gamma R_{R(n-1)}, B = R_n, C = 1,$ and $D = R_n^T$.

$$\begin{aligned}
\therefore R_{R(n)}^{-1} &= \left(\gamma R_{R(n-1)} + R_n.1.R_n^T\right)^{-1} \\
&= \frac{1}{\gamma}\left[R_{R(n-1)}^{-1}\right] - \left[\frac{\frac{1}{\gamma}R_{R(n-1)}^{-1}.R_n.R_n^T\frac{1}{\gamma}R_{R(n-1)}^{-1}}{R_n^T.\frac{1}{\gamma}.R_{R(n-1)}^{-1}.R_n + 1}\right] \\
&= \frac{1}{\gamma}\left[R_{R(n-1)}^{-1} - \frac{R_{R(n-1)}^{-1}R_nR_n^TR_{R(n-1)}^{-1}}{\gamma + R_n^TR_{R(n-1)}^{-1}R_n}\right]
\end{aligned}$$

Hence **it is possible to compute $R_{R(n)}^{-1}$ from the knowledge of** $R_{R(n-1)}^{-1}$.

Now let us denote:

$$r_n \triangleq R^{-1}_{R(n-1)} R_n$$
$$c_n \triangleq \gamma + R_n^T r_n$$

From the original definite of $R_{R\ (n)}$, it is a symmetric matrix.

Now,

$$\begin{aligned} r_n^T &= \left(R^{-1}_{R(n-1)} R_n\right)^T = R_n^T \left(R^{-1}_{R(n-1)}\right)^T \\ &= R_n^T \left(R^T_{R(n-1)}\right)^{-1} \qquad \left(\because \left(X^{-1}\right)^T = \left(X^T\right)^{-1}\right) \\ &= R_n^T R^{-1}_{R(n-1)} \qquad \left(\because R_R \text{ is a symmetric matrix}\right) \end{aligned}$$
$$R^{-1}_{R(n)} = \frac{1}{\gamma}\left[R^{-1}_{R(n-1)} - \frac{r_n r_n^T}{c_n}\right] \qquad (n \geq 1)$$

This equation shows that we need not compute any matrix inversion explicitly. To start the procedure, an initial value for the inverse of the auto-correlation matrix is required. It can be set as, $R^{-1}_{R(0)} = \delta^{-1} I$ (assuming R_n as causal).

The initial estimate of the inverse of the auto-correlation matrix is not likely to be accurate. However the exponential weighting associated with $\gamma < 1$ tends to minimize the effects of any initial error in the estimate after a sufficient number of iterations. Let us introduce the notation Q for R_R^{-1}.

Then the **RLS algorithm** can be given as:

ALGORITHM 1: RLS Algorithm

```
1. BEGIN
2. Specify the following parameters:
     $M$ = filter order ( $M \geq 0$)
     $\gamma$ = exponential weighting factor ($0 < \gamma \leq 1$)
     $\delta$ = regularization parameter ($\delta > 0$)
     $N$ = number of iterations ( $N \geq 1$)
3. Initialization:
     $H_0 = 0, P_{PR(0)} = 0, Q_{(0)} = \frac{I}{\delta}$
     $\left(H_0 \text{ and } P_{PR(0)} \text{ are } (M \times 1) \text{ and } Q_{(0)} \text{ is } (M \times M)\right)$
4. Computation: FOR $n = 1$ to $N$
     $R_n = \left[R_n, R_{n-1}, R_{n-2}, \ldots..R_{(n-M+1)}\right]^T;$
     $r_n = Q_{(n-1)} R_n;$
     $c_n = \gamma + R_n^T r_n;$
     $P_{PR(n)} = \gamma P_{PR(n-1)} + P_n R_n;$
     $Q_{(n)} = \frac{1}{\gamma}\left[Q_{(n-1)} - \frac{r_n r_n^T}{c_n}\right];$
     $H_n = Q_{(n)} P_{PR(n)};$
   ENDFOR
5. END
```

The RLS method in Algorithm – 1 is an algorithm of order $O(M^2)$. The much simpler LMS algorithm is an algorithm of order $O(M)$ with the computational effort proportional to m. There are faster variations of the RLS algorithm later proposed, which exploit recursive formulations of r_n and H_n.

Now we demonstrate the development of the **RLS algorithm** in another form, where the adaptation in weights is shown in the $H_n = H_{n-1} + \Delta H_{n-1}$ form. Let us go back to the relation,

$$R_{R(n)}^{-1} = \frac{1}{\gamma}\left[R_{R(n-1)}^{-1} - \frac{R_{R(n-1)}^{-1} R_n R_n^T R_{R(n-1)}^{-1}}{\gamma + R_n^T R_{R(n-1)}^{-1} R_n} \right]$$

We have already defined earlier $Q_n = H_{R(n)}^{-1}$. $\therefore$ We can write,

$$Q_{(n)} = \frac{1}{\gamma}\left[Q_{(n-1)} - K_{(n)} R_n^T Q_{(n-1)} \right]$$

where $K_{(n)}$ is a ***gain vector***, defined as

$$K_{(n)} = \frac{Q_{(n-1)} R_n}{\gamma + R_n^T Q_{(n-1)} R_n} = \frac{Q_{(n-1)} R_n}{\gamma + \mu_{(n)}}$$

$K_{(n)}$ is called the **Kalman gain vector**. $\mu_{(n)}$ is a scalar quantity and is given as

$$\mu_{(n)} = R_n^T Q_{(n-1)} R_n$$

Now, postmultiplying by R_n,

$$\begin{aligned} Q_{(n)}R_n &= \frac{1}{\gamma}\left[Q_{(n-1)}R_n - K_{(n)}R_n^T Q_{(n-1)}R_n\right] \\ &= \frac{1}{\gamma}\left[\left\{\gamma+\mu_{(n)}\right\}K_{(n)} - K_{(n)}\mu_{(n)}\right] \\ &= \frac{K_{(n)}}{\gamma}\left[\gamma+\mu_{(n)}-\mu_{(n)}\right] = K_{(n)} \end{aligned}$$

Hence, the Kalman gain vector may also be defined as,

$$K_{(n)} = Q_{(n)}R_n$$

Now,

$$H_n = Q_{(n)}P_{PR(n)} \text{ and } P_{PR(n)} = \gamma P_{PR(n-1)} + P_n R_n$$

$$\begin{aligned} \therefore H_n &= Q_{(n)}P_{PR(n)} \\ &= \gamma Q_{(n)}P_{PR(n-1)} + P_n Q_{(n)}R_n \\ &= \gamma\left[\frac{1}{\gamma}\left\{Q_{(n-1)} - K_{(n)}R_n^T Q_{(n-1)}\right\}\right]P_{PR(n-1)} + P_n K_{(n)} \\ &= Q_{(n-1)}P_{PR(n-1)} + K_{(n)}\left[P_n - R_n^T Q_{(n-1)}P_{PR(n-1)}\right] \\ &= H_{n-1} + K_{(n)}\left[P_n - R_n^T H_{(n-1)}\right] \\ &= H_{n-1} + K_{(n)}\alpha_n \end{aligned}$$

Here $R_n^T H_{(n-1)}$ is the **output of the adaptive filter at time instant n utilizing the filter coefficients at time instant (n – 1)**.

Here $\alpha_n = P_n - R_n^T H_{n-1}$ is called the ***a priori* error**. The error calculated after updating the filter is

$$e_n = P_n - R_n^T H_n$$

This is called ***a posteriori* error**.

Hence the correction factor in each iteration is

$$\Delta H_{n-1} = K_{(n)} \alpha_n$$

This correction factor is directly proportional to both the ***a priori* error** and the **Kalman gain vector**.

ALGORITHM 2: RLS Algorithm

```
1. BEGIN
2. Specify the following parameters:
    $M$ = filter order ($M \geq 0$)
    $\gamma$ = exponential weighting factor ( $0 < \gamma \leq 1$)
    $\delta$ = regularization parameter ($\delta > 0$)
    $N$ = number of iterations ($N \geq 1$)
3. Initiatization:
    $H_0 = 0, P_{PR(0)} = 0, Q_{(0)} = \frac{I}{\delta}$
    $\left(H_0 \text{ and } P_{PR(0)} \text{ are } (M \times 1) \text{ and } Q_{(0)} \text{ is } (M \times M)\right)$
4. Computation: FOR $n = 1$ to $N$
    $R_n = \left[R_n, R_{n-1}, R_{n-2}, \ldots\ldots, R_{(n-M+1)}\right]^T;$
    $\alpha_n = P_n - R_n^T H_{n-1};$
    $K_n = \frac{Q_{(n-1)} R_n}{\gamma + R_n^T Q_{(n-1)} R_n};$
    $Q_{(n)} = \frac{1}{\gamma}\left[Q_{(n-1)} - K_{(n)} R_n^T Q_{(n-1)}\right];$
    $H_n = H_{n-1} + K_{(n)} \alpha_n;$
    ENDFOR
5. END
```

The recursion used for calculating $Q_{(n)}$ follows a ***Riccati equation*** and thus draws parallels to the ***Kalman filter***.

Choice of RLS filter parameters

There are three important filter parameters associated with the RLS method: γ, δ and M. The suitable filter order M varies with the application and is often found empirically. It has also been shown that $\Delta = (1 - \gamma)$ plays a role similar to the step size in the LMS algorithm. Hence γ should be chosen close to unity so that Δ is kept small.

Comparison of LMS and RLS algorithms

A major advantage of the direct-form RLS algorithms over the LMS algorithm is their faster convergence rate. The figure below shows a comparison of their convergence rates for an adaptive FIR channel equalizer problem.

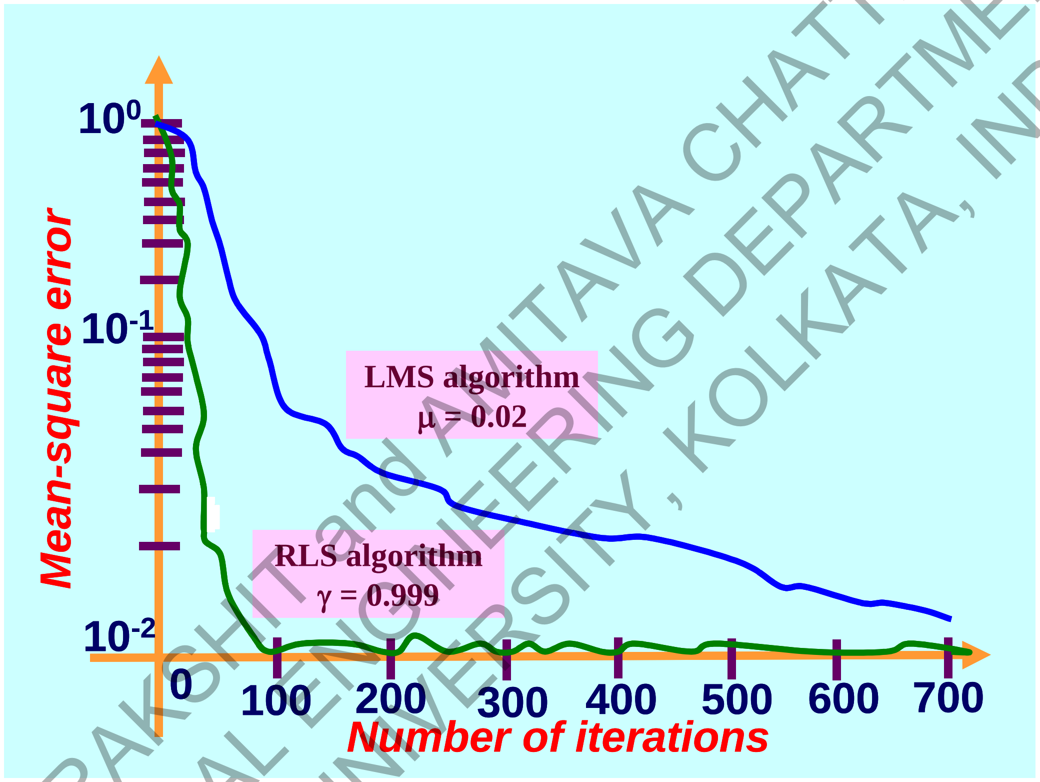

Fig. Learning curves for RLS and LMS algorithms for adaptive equalizer of length M = 11. The eigenvalue spread of the channel is $\frac{\lambda_{\max}}{\lambda_{\min}} = 11$.

For its superior convergence rate, RLS algorithm is more suitable for those applications where the signal statistics vary rapidly with time. For example, for the channel equalization problem, LMS algorithm is slow to adapt to the new channel characteristics. But RLS algorithm can adapt sufficiently fast to track such rapid variations.

However, the RLS algorithms for FIR adaptive filtering have two important disadvantages. One is their increased computational complexity per iteration. Secondly, RLS algorithms are sensitive to round-off errors that accumulate as a result of the recursive computations. In some cases, the round-off errors cause these algorithms to become unstable.

REFERENCES

[1] J.G. Proakis and D.G. Manolakis. Digital Signal Processing: Principles, Algorithms, and Applications. Fourth Edition, Pearson Education, 2007.

[2] R.J. Schilling and S.L. Harris. Fundamentals of Digital Signal Processing using MATLAB. Thomson India Edition, 2007.

[3] Recursive Least Squares Filter. http://en.wikipedia.org/wiki/Recursive_least_squares_filter.