Recursive Least-Squares Adaptive Filters

Dr. Yogananda Isukapalli

Contents

- Introduction
- Least-Squares problem
- Derivation of RLS algorithm
  -- The Matrix Inversion Problem
- Convergence analysis of the RLS algorithm
- Application of RLS Algorithm
  --Adaptive Equalization

Introduction

- Linear least-squares problem was probably first developed and solved by Gauss (1795) in his work on mechanics
- L-S solutions have attractive properties;
  - can be explicitly evaluated in closed forms
  - can be recursively updated as more input data is made available
  - are maximum likelihood estimators in the presence of Gaussian measurement noise

- Over the last several years, a wide variety of adaptive algorithms based on least squares criterion has been derived
  - RLS (**R**ecursive **L**east **S**quares) algorithms and corresponding fast versions
- FTF (**F**ast **T**ransversal **F**ilter)
- FAEST (**F**ast **A**posteriori **E**rror **S**equential **T**echnique)
  - QR and inverse QR algorithms
  - LSL (**L**east **S**quares **L**attice) and QR decomposition

Least-Squares Problem

Consider a standard observation model in additive noise.

$$\mathbf{d}(i) = \mathbf{U}_i^H \mathbf{W} + \mathbf{n}(i)$$

d(*i*)…noisy measurement linearly related to **W**
W…Is the unknown vector to be estimated
$\mathbf{U}_i$…Given column vector
n(*i*)…the noise vector

In a practical scenario, the **W** can be the weight vector, $\mathbf{U}_i$ The data vector, **d**(*i*) the observed output using a Series of Arrays and **n(*i*)** is the noise vector added at each sensor.

If we have N+1 measurements then they can be grouped together into a single matrix expression;

$$\begin{bmatrix} \mathbf{d}(0) \\ \mathbf{d}(1) \\ . \\ . \\ \mathbf{d}(N) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_0^H \\ \mathbf{u}_1^H \\ . \\ . \\ \mathbf{u}_N^H \end{bmatrix} \mathbf{W} + \begin{bmatrix} \mathbf{n}(0) \\ \mathbf{n}(1) \\ . \\ . \\ \mathbf{n}(N) \end{bmatrix}$$

$$\Leftrightarrow \mathbf{d} = \mathbf{U}\mathbf{W} + \mathbf{n}$$

The problem is to minimize the distance between $\mathbf{d}$ and $\mathbf{U}\hat{\mathbf{W}}$
Where $\hat{\mathbf{W}}$ is the estimated weight vector.

$$\min_{\hat{\mathbf{W}}} \left\| \mathbf{d} - \mathbf{U}\hat{\mathbf{W}} \right\|^2$$

A least squares solution to the above problem is,

$$\hat{\mathbf{W}} = \left(\mathbf{U}^{\mathbf{H}}\mathbf{U}\right)^{-1}\mathbf{U}^{H}\mathbf{d}$$

Let Z be the cross correlation vector and Φ be the covariance matrix.

$$\mathbf{Z} = \mathbf{U}^{H}\mathbf{d}$$

$$\Phi = \mathbf{U}^{H}\mathbf{U}$$

$$\hat{\mathbf{W}} = \Phi^{-1}\mathbf{Z}$$

The above equation could be solved block by block basis but we are interested in recursive determination of tap weight estimates **w.**

RLS algorithm

- The RLS algorithm solves the least squares problem recursively
- At each iteration when new data sample is available the filter tap weights are updated
- This leads to savings in computations
- More rapid convergence is also achieved

The derivation of RLS algorithm

The attempt is to find a recursive solution to the following minimization problem,

$$\varepsilon(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| e(i) \right|^2$$

$$e(i) = d(i) - y(i) = d(i) - \mathbf{W}^H(n)\mathbf{U}(i)$$

$$\mathbf{U}(i) = \left[\mathbf{U}(i), \mathbf{U}(i-1), ..., \mathbf{U}(i-M+1)\right]^T$$

$$\mathbf{W}(n) = \left[\mathbf{W}_0(n), \mathbf{W}_1(n), ...\mathbf{W}_{M-1}(n)\right], \quad \textit{the length of filter is } M.$$

– The weighting factor has the property that

$$0 << \lambda \leq 1.$$

– weighting factor is used to “forget” data samples in distant past, usual value is 0.99.

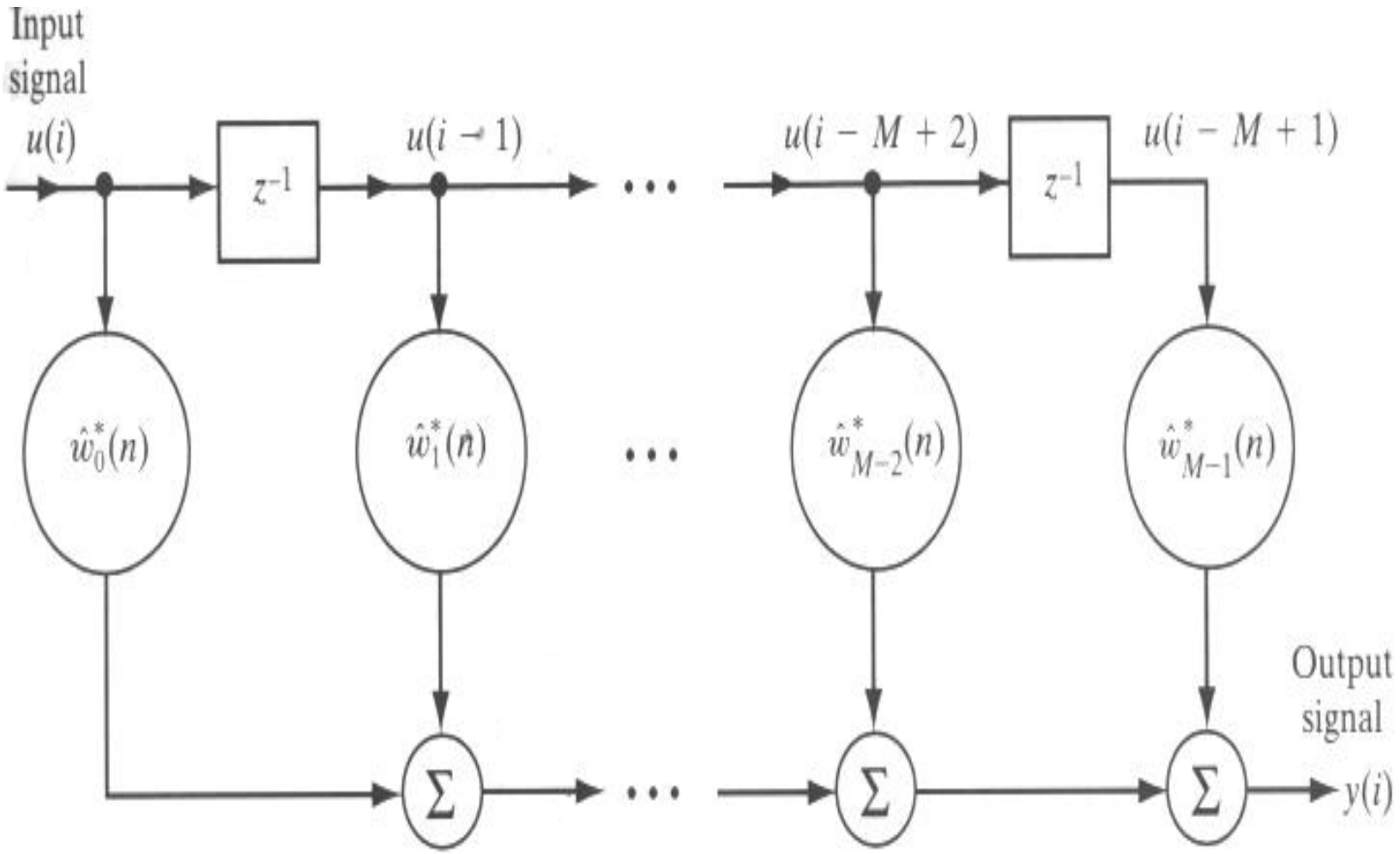


FIGURE 9.1 Transversal filter with time-varying tap weights.

– The optimum value for the tap-weight vector is defined by normal equations.

$$\Phi(n)\hat{\mathbf{w}}(n) = \mathbf{Z}(n) \Leftrightarrow \hat{\mathbf{w}}(n) = \Phi^{-1}(n)\mathbf{Z}(n)$$

$$\Phi(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{U}(i)\mathbf{U}^{H}(i) + \delta\lambda^{n}\mathbf{I}$$

$$\Phi(n) = \left[\lambda \sum_{i=1}^{n-1} \lambda^{n-i-1}\mathbf{U}(i)\mathbf{U}^{H}(i) + \delta\lambda^{n}\mathbf{I}\right] + \mathbf{U}(n)\mathbf{U}^{H}(n)$$

$$\Phi(n) = \lambda\Phi(n-1) + \mathbf{U}(n)\mathbf{U}^{H}(n)$$

$$\mathbf{Z}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{U}(i)d^{*}(i), \textit{the cross correlation term.}$$

$$\mathbf{Z}(n) = \lambda\mathbf{Z}(n-1) + \mathbf{U}(n)d^{*}(n)$$

– Then we use the matrix inversion lemma to the recursive model of correlation matrix to make it possible to invert correlation matrix recursively

The matrix inversion lemma

– **A** and **B** are two positive-definite M-by-M matrices
– **D** is another positive-definite N-by-M matrix
– **C** is an M-by-N matrix.

Let **A**,**B**,**C** and **D** be related as,

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{\mathrm{H}}.$$

Then the inverse of **A** is given by,

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^{\mathrm{H}}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{H}}\mathbf{B}.$$

Application of matrix inversion lemma to the present Problem is based on the following definitions.

$$\mathbf{A} = \mathbf{\Phi}(n)$$

$$\mathbf{B}^{-1} = \lambda\mathbf{\Phi}(n-1)$$

$$\mathbf{C} = \mathbf{U}(n)$$

$$\mathbf{D} = 1.$$

– These definitions are substituted in the matrix inversion lemma
– After some calculations we get following equations

$$\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n)$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{K}(n)\mathbf{U}^{H}(n)\mathbf{P}(n-1)\,(\textit{Riccati equation})$$

$$\mathbf{K}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{U}(n)}{1+\lambda^{-1}\mathbf{U}^{H}(n)\mathbf{P}(n-1)\mathbf{U}(n)} = \mathbf{P}(n)\mathbf{U}(n)$$

– Now we have recursive solution to the inverse of correlation matrix
– Next we need update method for the tap-weight vector

- Time update for the tap-weight vector

$$\hat{\mathbf{W}}(n) = \mathbf{\Phi}^{-1}(n)\mathbf{Z}(n)$$
$$= \mathbf{P}(n)\mathbf{Z}(n)$$
$$= \lambda\mathbf{P}(n)\mathbf{Z}(n-1) - \mathbf{P}(n)\mathbf{U}(n)d^{*}(n)$$

By substituting the Riccati equation to the first term in the right side of the equation,

$$\hat{\mathbf{W}}(n) = \mathbf{P}(n-1)\mathbf{Z}(n-1) - \mathbf{K}(n)\mathbf{U}^{H}(n)\mathbf{P}(n-1)\mathbf{Z}(n-1)$$
$$+ \mathbf{P}(n)\mathbf{U}(n)d^{*}(n)$$
$$= \hat{\mathbf{W}}(n-1) - \mathbf{K}(n)\mathbf{U}^{H}(n)\hat{\mathbf{W}}(n-1) + \mathbf{P}(n)\mathbf{U}(n)d^{*}(n)$$

Then using the fact that,

$$\mathbf{K}(n) = \mathbf{P}(n)\mathbf{U}(n)$$

The desired recursion equation for updating the tap-weight Vector.

$$\hat{\mathbf{W}}(n) = \hat{\mathbf{W}}(n-1) + \mathbf{K}(n)\left[d^*(n) - \mathbf{U}^H(n)\hat{\mathbf{W}}(n-1)\right]$$

$$= \hat{\mathbf{W}}(n-1) + \mathbf{K}(n)\xi^*(n)$$

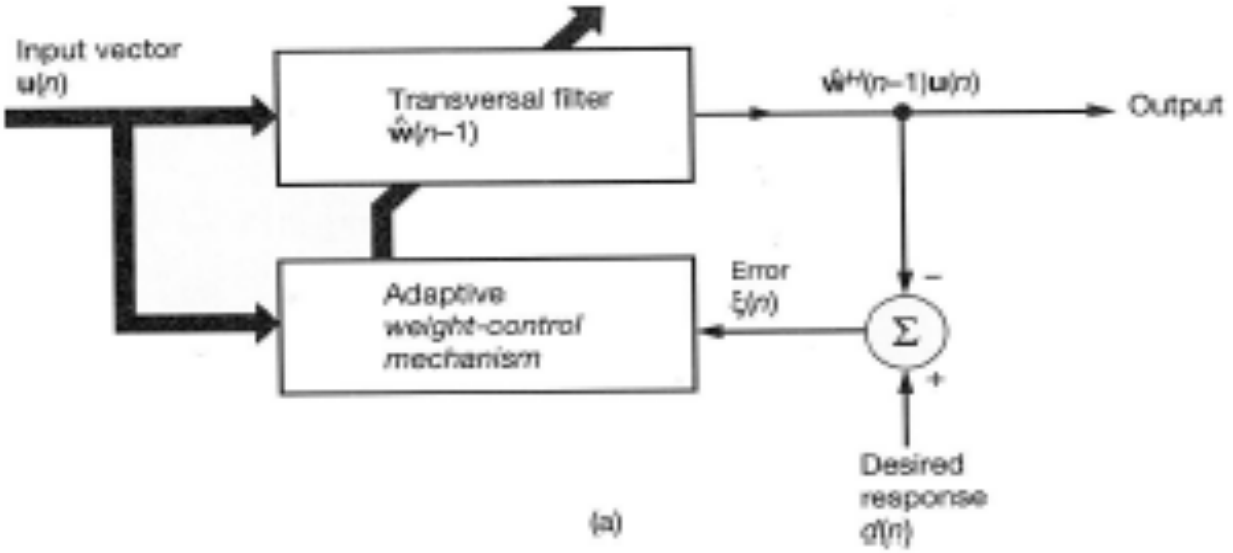
$$\xi(n) = d(n) - \mathbf{U}^T(n)\hat{\mathbf{W}}^*(n-1)$$

$$= d(n) - \hat{\mathbf{W}}^H(n-1)\mathbf{U}(n)$$

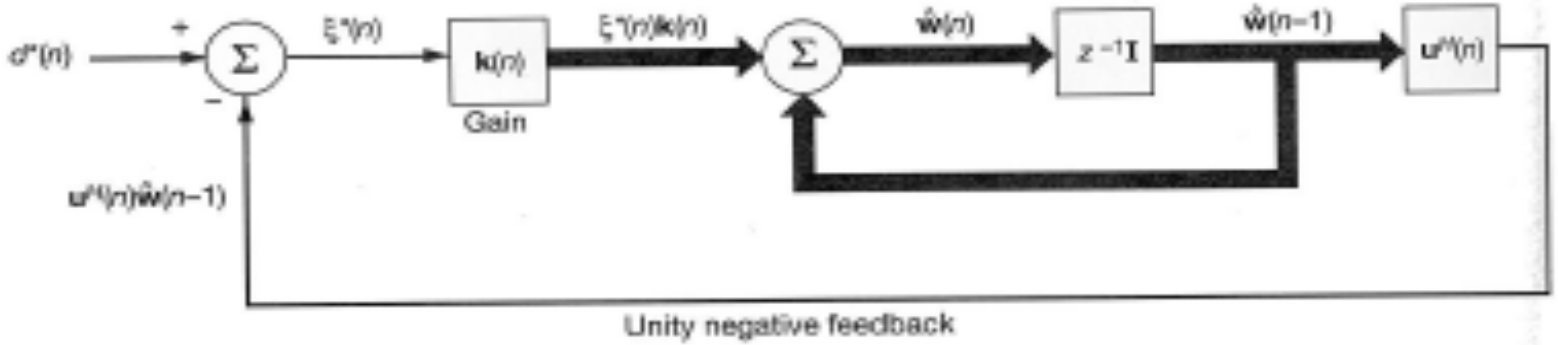
The block diagram shown in the next page illustrates the Use of *a priori* estimation error in RLS algorithm.

a priori estimation error is in general different from *a posteriori* estimation error $e(n)$

$$e(n) = d(n) - \hat{\mathbf{W}}^H(n)\mathbf{U}(n)$$



(a)



(b)

- Block diagram of RLS algorithm

Summary of RLS algorithm

Initialize the algorithm by setting

$$\hat{\mathbf{W}}(0) = 0, \qquad \mathbf{P}(0) = \delta^{-1}\mathbf{I}.$$

$$\delta = \begin{cases} \text{small positive constant for high SNR} \\ \text{large positive constant for low SNR} \end{cases}$$

For each time instant of time, n = 1,2,...compute

$$\pi(n) = \mathbf{P}(n-1)\mathbf{U}(n). \ \mathbf{K}(\mathrm{n}) = \frac{\pi(n)}{\lambda + \mathbf{U}^H(n)\pi(n)},$$

$$\xi(\mathrm{n}) = d(n) - \hat{\mathbf{W}}^H(n-1)\mathbf{U}(n),$$

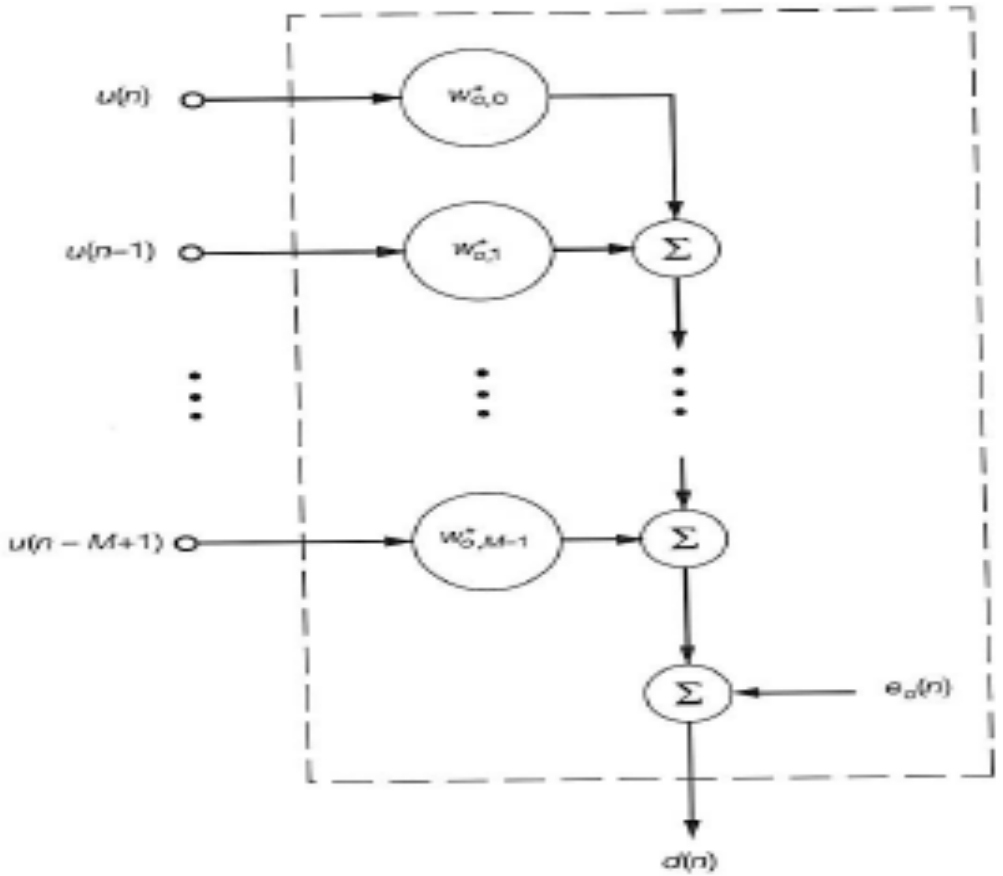
$$\hat{\mathbf{W}}^H(n) = \hat{\mathbf{W}}^H(n-1) + \mathbf{K}(n)\xi^*(\mathrm{n}), \quad \text{and}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{K}(n)\mathbf{U}^H(n)\mathbf{P}(n-1).$$

Convergence Analysis of the RLS algorithm

The desired response and the tap input vector are assumed to be related by the multiple linear regression model.

Block Diagram of Linear Regression Model.



Assumption **I**:

$$d(n) = \mathbf{w}_0^H \mathbf{u}(n) + e_0(n),$$

Where,

$\mathbf{w}_0$ *is the regression parameter vector.*

$e_0(n)$ *is the measurment noise.*

$e_0(n)$ *is white with zero mean and variance* σ_0^2.

$e_0(n)$ *is independent of the regressor* $\mathbf{u}(n)$.

The exponential weighting factor λ *is unity.*

Assumption **II**:

The input signal vector **u**(n) is drawn from a stochastic Process, which is ergodic in the autocorrelation function.

Assumption **III**:

The Fluctuations in the weight-error vector must be slower Compared with those of the input signal vector **u**(n).

Convergence in the Mean Value

With the help of above assumptions it can be shown that, RLS algorithm is convergent in the mean sense for n ≥ M, Where 'M' is the number of taps in the additive transversal filter.

$$E[\hat{\mathbf{w}}(n)] = \mathbf{w}_0 - \frac{\delta}{n}\mathbf{p}.$$

- Unlike the LMS algorithm, the RLS algorithm does not have to wait for n to be infinitely large for convergence.

- The mean-squared error in the weight vector

The weight error vector is defined as,

$$\boldsymbol{\varepsilon}(n) = \hat{\mathbf{w}}(n) - \mathbf{w}_0$$

Expression for the mean-squared error in the weight vector,

$$E\left[\boldsymbol{\varepsilon}^H(n)\boldsymbol{\varepsilon}(n)\right] = \frac{\sigma_0^2}{n}\sum_{i=1}^{M}\frac{1}{\lambda_i}$$

- Ill conditioned least-squares problems may lead to poor convergence properties.
- The estimate $\hat{\mathbf{w}}(n)$ produced converges in the norm to the parameter vector $\mathbf{w}_0$ of the multiple linear regression model almost linearly with time.

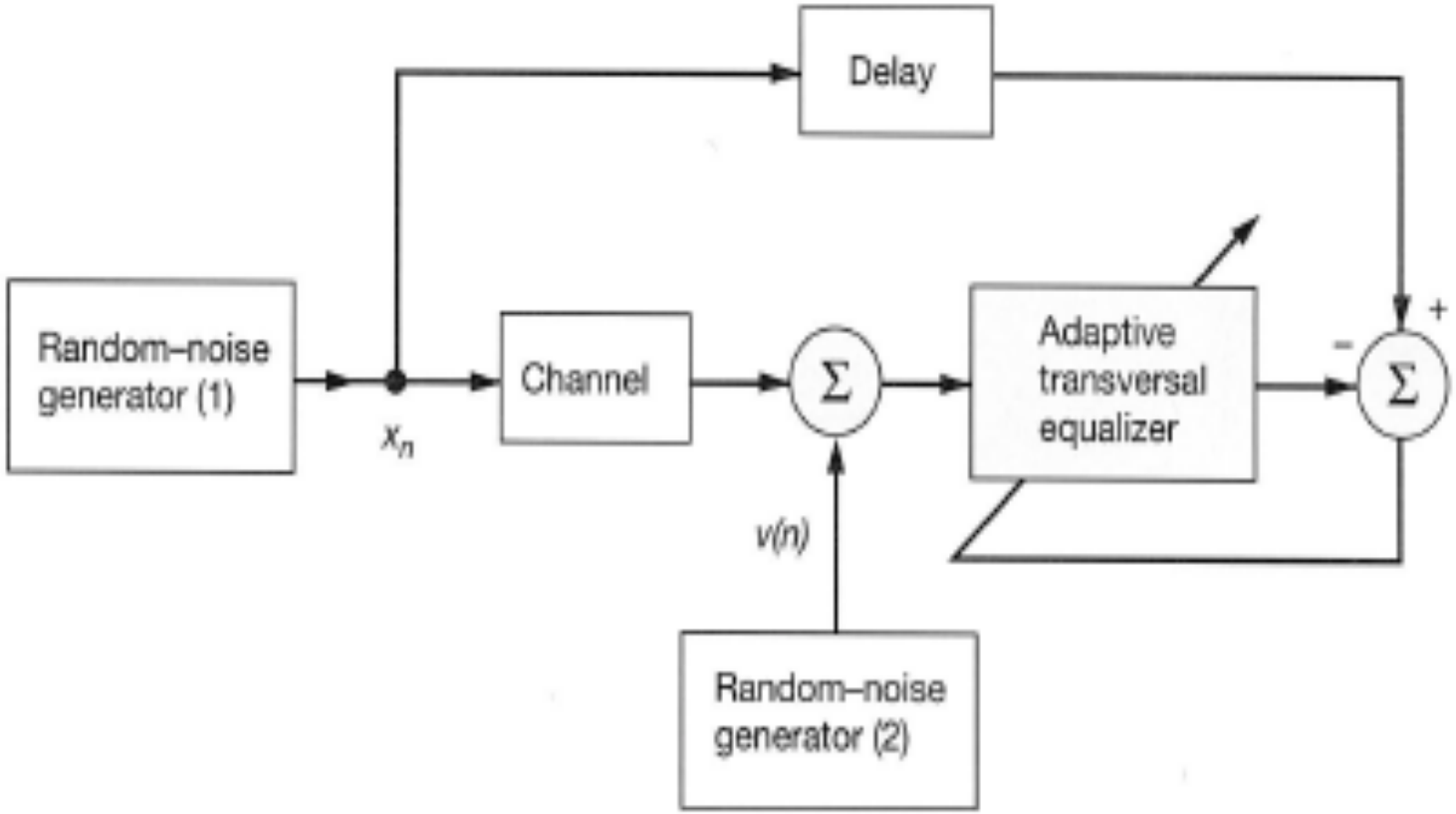
Learning curve of the RLS algorithm

Considerations on convergence:

- RLS algorithm converges in about 2M iterations, where M is the length of transversal filter
- RLS converges typically order of magnitude faster than LMS algorithm
- RLS produces zero misadjustment when operating in stationary environment (when n goes to infinity only measurement error is affecting to the precision
- convergence is independent of the eigenvalue spread

Example of RLS algorithm: Adaptive equalization I

- Block diagram of adaptive equalizer



Impulse response of the channel is,

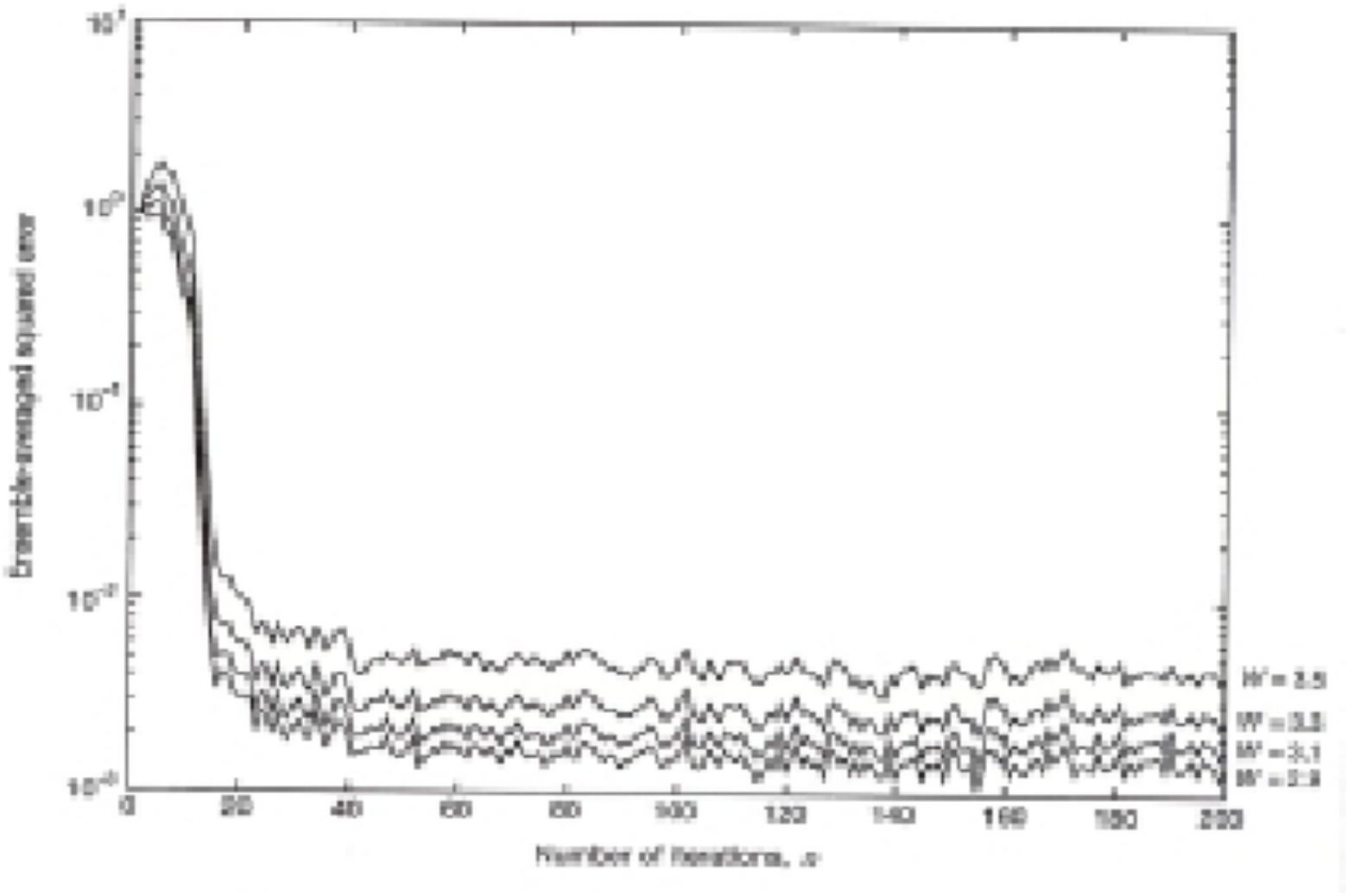
$$h_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right], & n=1,2,3 \\ 0 & \text{otherwise} \end{cases}$$

- where W controls the amount of amplitude distortion and therefore the eigenvalue spread produced by the channel.
- 11 taps, forgetting factor = 1

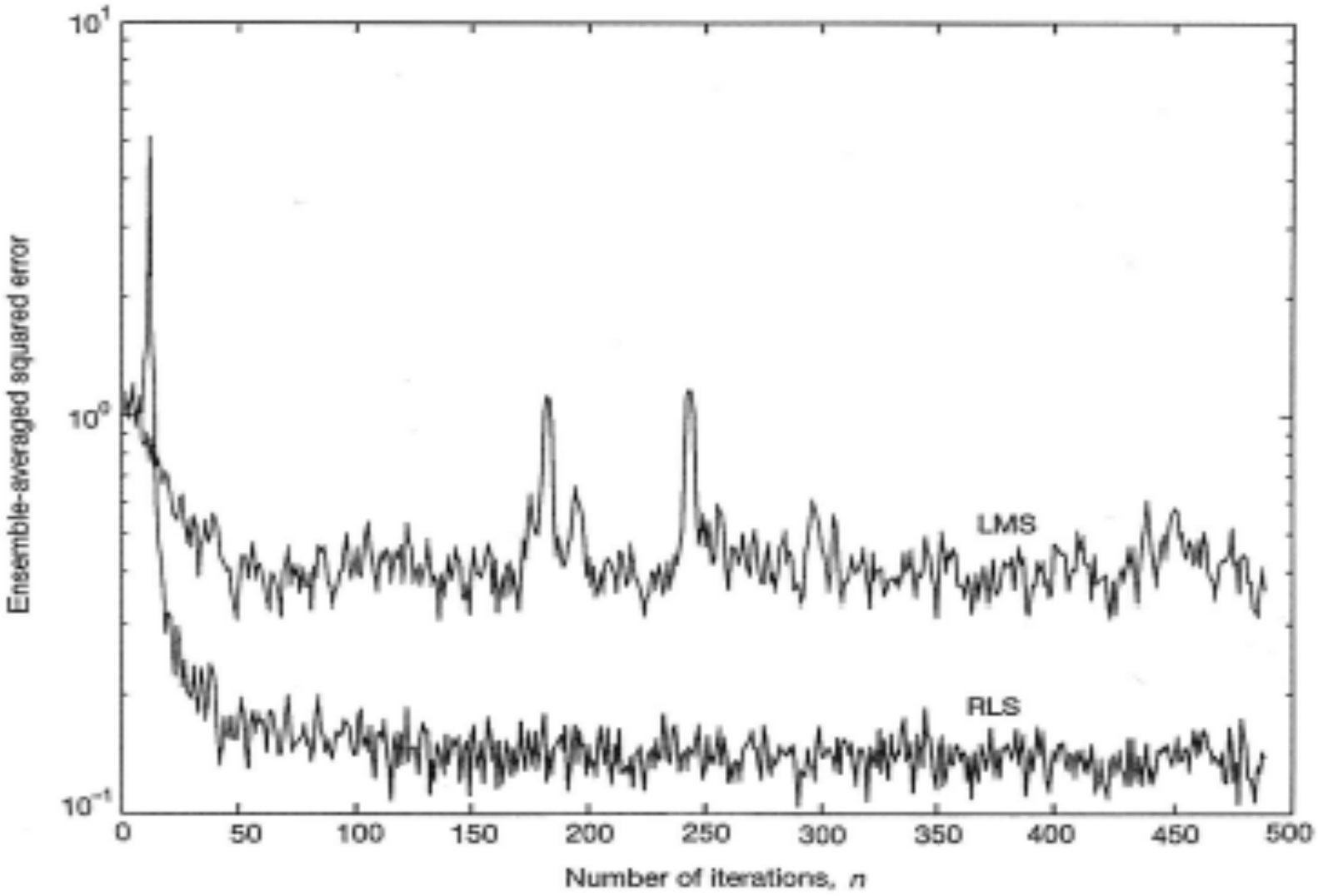
Experiment is done in to parts

- part 1: signal to noise ratio is high = 30 dB
- part 2: signal to noise ratio is low = 10 dB

Results of Part1



Results of Part2



– Part 1 summary

- Convergence of RLS algorithm is attained in about 20 iterations (twice the number of taps).
- Rate of convergence of RLS algorithm is relatively insensitive to variations in the eigenvalue spread.
- Steady-state value of the averaged squared error produced by the RLS algorithm is small, confirming that the RLS algorithm produces zero misadjustment.

– Part 2 summary

- The rate of convergence is nearly same for the LMS and RLS algorithm in noisy environment.