EED5069 Adaptive Filter Theory

Adaptive Channel Equalization

Term Project

Zekeriya Sarı

January 20, 2017

Contents

Chapter 1

Problem Statement

Data transmission rate of a bandwidth-limited communication channel characterized by high signal-to-noise ratios is determined primarily by the *intersymbol interference*(ISI) of the channel. Caused by the dispersive nature of the channel, ISI manifests itself as the "spreading" of the pulses transmitted through the channel, and hence results in overlapping of the adjacent pulses, which in return, if not taken care of properly, reduces the detection performance of the receiver. Since one needs to have undisturbed transmission which occurs when the replica of the transmitted signals are obtained in the receiver, possibly with a suitable time delay, an *equalizer* is employed in cascade with the channel to overcome the ISI of the channel. Figure 1.1 shows a typical block diagram of such a system. Here, random number generator 1 signifies the message to be transmitted while random number generator 2 is responsible for the modeling the additive noise present in the channel. The task of the equalizer to compensate the ISI of the channel. Note from the figure that the adaptive transversal equalizer is provided with the desired response which is the properly-delayed replica of the transmitted message by means of the delay block.

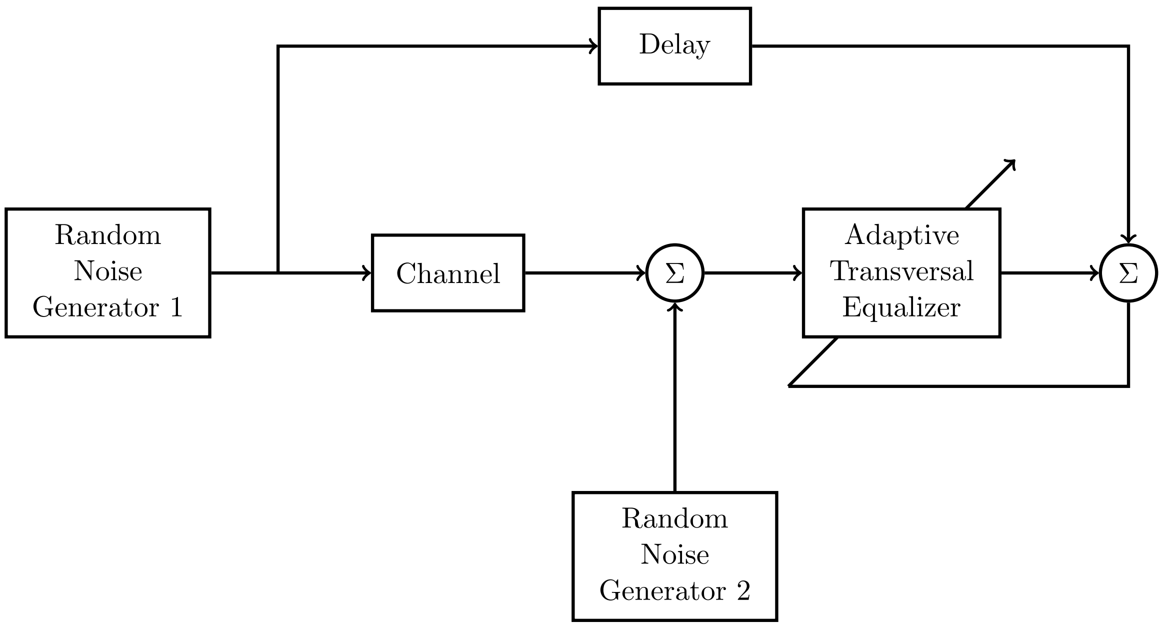


Figure 1.1: Block diagram of the channel equalizer.

For the system given in Figure 1.1, consider a tapped-delay-line equalizer implemented as an adaptive transversal filter with a block diagram given in Figure 1.2 and an impulse response

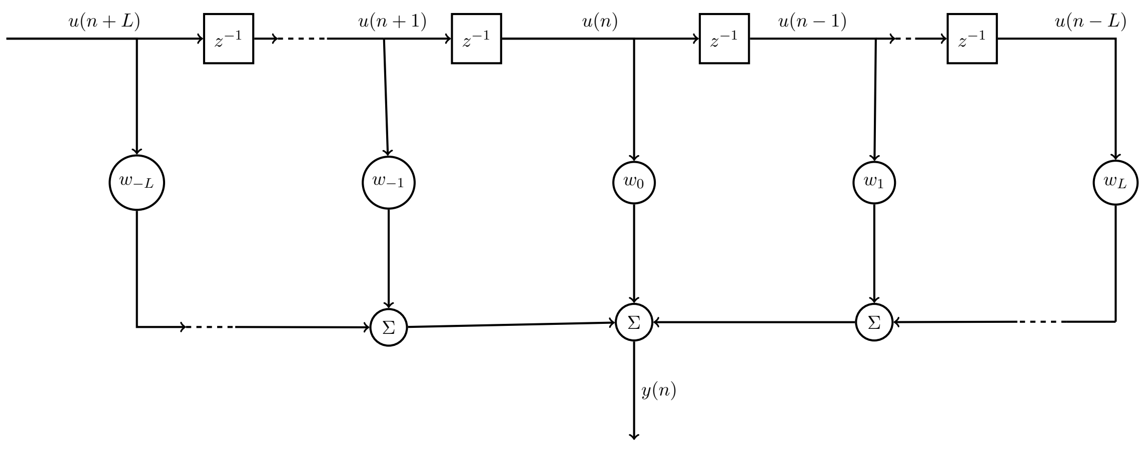


Figure 1.2: Block diagram of the symmetric delay-line-filter equalizer implementation.

given as,

$$h(n) = \sum_{k=-N}^{N} h_k \delta(n-k) \tag{1.1}$$

where $\delta(n)$ is the Dirac delta function. Note from Figure 1.2 that the equalizer is symmetric about its center tap. Consider also a linear communication system whose channel transfer function may take one of the following three possible forms,

(i) $H(z) = 0.25 + z^{-1} + 0.25z^{-2}$

(ii) $H(z) = 0.25 + z^{-1} - 0.25z^{-2}$

(iii) $H(z) = -0.25 + z^{-1} + 0.25z^{-2}$

The channel output in response to x_n is defined by,

$$u(n) = \sum_{k} h_k x_{n-k} + \nu(n) \tag{1.2}$$

where $h(n)$ is the impulse response of the channel and $\nu(n)$ is additive white Gaussian noise with zero mean and variance $\sigma_\nu^2 = 0.01$. The sequence x_n consists of the a Bernoulli sequence with $x_n = \pm 1$. Having determined the optimum delay for the channel models given above, this study aims to design an adaptive equalizer trained first by using the LMS algorithm with a step size $\mu = 0.001$ and then by using the RLS algorithm with a regularization parameter $\delta = 0.005$. The study also compares the performance of the LMS algorithm to that of the RLS algorithm by plotting the learning curves of the equalizer by averaging the squared value of the error signal over an ensemble of 100 independent trials.

Chapter 2

Least Mean Square Algorithm

The least mean squares(LMS) algorithm, being an important family of the stochastic gradient algorithm, favors itself for its simplicity. As opposed to the case encountered in the steepest descent algorithm which uses deterministic gradient for the Wiener filtering problem in a recursive fashion, it does not require any correlation function evaluation or matrix inversion. It is this very simplicity of the LMS algorithm that makes it the top rated among all other linear adaptive filters.

2.1 Least Mean Square Adaptation Algorithm

Given a transversal filter with tap weights $w_0(n), \dots, w_{M-1}(n)$, tap inputs $u(n), \dots, u(n-M+1)$ and a desired response $d(n)$, the SD algorithm adapts the tap weights of the filter as,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu(\boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}(n)), \quad n = 0, 1, \dots \tag{2.1}$$

where μ is the step size, $\boldsymbol{w}(n) = \begin{bmatrix} w_0(n) & \dots & w_{M-1}(n) \end{bmatrix}$ the tap-weight vector, $\boldsymbol{R}$ is the correlation matrix of the tap input vector $\boldsymbol{u}(n) = \begin{bmatrix} u(n) & \dots & u(n-M+1) \end{bmatrix}$ and $\boldsymbol{p}$ is the cross correlation vector between the tap input vector $\boldsymbol{u}(n)$ and the desired response $d(n)$. Note that SD algorithm requires the exact knowledge of $\boldsymbol{R}$ and $\boldsymbol{p}$ which may not be possible all the time. Hence, $\boldsymbol{R}$ and $\boldsymbol{p}$ must be estimated from the data at hand, and this what the LMS algorithm does. It uses the instantaneous estimates $\hat{\boldsymbol{R}}(n)$ of $\boldsymbol{R}$ and $\hat{\boldsymbol{p}}(n)$ of $\boldsymbol{p}$, i.e.,

$$\hat{\boldsymbol{R}}(n) = \boldsymbol{u}(n)\boldsymbol{u}^H(n) \tag{2.2}$$

and

$$\hat{\boldsymbol{p}}(n) = \boldsymbol{u}(n)d^*(n) \tag{2.3}$$

Substituting (2.2) and (2.3) into (2.1), the adaptation algorithm of the LMS algorithm is,

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \mu\boldsymbol{u}(n)\left(d^*(n) - \boldsymbol{u}^H(n)\hat{\boldsymbol{w}}(n)\right) \tag{2.4}$$

where $\hat{w}(n)$ is used to distinguish the adaptation of the LMS from that of the SD.The algorithm starts with an initial guess $\hat{\boldsymbol{w}}(0)$.

2.2 Stability and Performance Analysis of the LMS ALgortihm

Instead of working with the tap weights directly, we define the weight error vector,

$$\boldsymbol{\epsilon}(n) = \hat{\boldsymbol{w}}(n) - \boldsymbol{w}_0 \tag{2.5}$$

where $\boldsymbol{w}_0$ is the optimum Wiener filter solution and $\hat{\boldsymbol{w}}(n)$ is the estimate of the tap weights at iteration n. Then, the LMS algorithm can be rewritten in terms of $\boldsymbol{\epsilon}(n)$,

$$\boldsymbol{\epsilon}(n) = \left(\boldsymbol{I} - \mu\boldsymbol{u}(n)\boldsymbol{u}^H(n)\right)\boldsymbol{\epsilon}(n) + \mu\boldsymbol{u}(n)e_o^*(n) \tag{2.6}$$

where $\boldsymbol{I}$ is the identity matrix and $e_o(n)$ is the estimation of the error produced by the optimum Wiener solution, i.e.,

$$e_o(n) = d(n) - \boldsymbol{w}_o^H(n)\boldsymbol{u}(n) \tag{2.7}$$

The correlation matrix of $\boldsymbol{\epsilon}(n)$ is,

$$\boldsymbol{K}(n) = E\left[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\right] \tag{2.8}$$

Using the independence assumption [Haykin, 2008] and direct-averaging method, we can write from (2.6),

$$\boldsymbol{K}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{R})\boldsymbol{K}(n)(\boldsymbol{I} - \mu\boldsymbol{R}) + \mu^2 J_{min}\boldsymbol{R} \tag{2.9}$$

Defining the estimation error $\boldsymbol{e}(n)$ at iteration n,

$$\boldsymbol{e}(n) = d(n) - \hat{\boldsymbol{w}}^H(n)\boldsymbol{u}(n) = d(n) - \boldsymbol{w}_o^H\boldsymbol{u}(n) - \boldsymbol{\epsilon}^H(n)\boldsymbol{u}(n) = \boldsymbol{e}_o(n) - \boldsymbol{\epsilon}^H(n)\boldsymbol{u}(n) \tag{2.10}$$

then, using the independence assumption [Haykin, 2008], we can write for the cost function $J(n) = E\left[|\boldsymbol{e}(n)|^2\right]$,

$$J(n) = E\left[|\boldsymbol{e}(n)|^2\right] = E\left[\boldsymbol{e}(n)\boldsymbol{e}^*(n)\right] = J_{min} + E\left[\boldsymbol{\epsilon}^H(n)\boldsymbol{u}(n)\boldsymbol{u}^H(n)\boldsymbol{\epsilon}(n)\right] \tag{2.11}$$

Since,

$$\begin{aligned} E\left[\boldsymbol{\epsilon}^H(n)\boldsymbol{u}(n)\boldsymbol{u}^H(n)\boldsymbol{\epsilon}(n)\right] &= E\left[tr\left[\boldsymbol{\epsilon}^H(n)\boldsymbol{u}(n)\boldsymbol{u}^H(n)\boldsymbol{\epsilon}(n)\right]\right] \\ &= E\left[tr\left[\boldsymbol{u}(n)\boldsymbol{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\right]\right] \\ &= tr\left[E\left[\boldsymbol{u}(n)\boldsymbol{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\right]\right] \\ &= tr\left[\boldsymbol{R}\boldsymbol{K}(n)\right] \end{aligned} \tag{2.12}$$

where we have used the fact that $tr\left[\boldsymbol{A}\boldsymbol{B}\right] = tr\left[\boldsymbol{B}\boldsymbol{A}\right]$ for any two matrix $\boldsymbol{A}$ and $\boldsymbol{B}$ with proper dimensions. Hence, (2.11) is equal to,

$$J(n) = J_{min} + tr\left[\boldsymbol{R}\boldsymbol{K}(n)\right] \tag{2.13}$$

For the LMS algorithm the excess mean square error $J_{ex}(n) = J(n) - J_{min}$. From (2.13),

$$J_{ex}(n) = tr\left[\boldsymbol{R}\boldsymbol{K}(n)\right] \tag{2.14}$$

From the unitary transformation of $\boldsymbol{R}$, i.e.,

$$\boldsymbol{Q}^H\boldsymbol{R}\boldsymbol{Q} = \boldsymbol{\Lambda} \tag{2.15}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix consisting of the eigenvalues of $\boldsymbol{R}$ and $\boldsymbol{Q}$ is the unitary matrix consisting of the eigenvectors corresponding to these eigenvalues. Using the transformation,

$$\boldsymbol{Q}^H\boldsymbol{K}(n)\boldsymbol{Q} = \boldsymbol{X}(n) \tag{2.16}$$

it can be shown that,

$$tr\left[\boldsymbol{R}\boldsymbol{K}(n)\right] = tr\left[\boldsymbol{\Lambda}\boldsymbol{X}(n)\right] \tag{2.17}$$

which implies,

$$J_{ex}(n) = \boldsymbol{\Lambda X}(n) = \sum_{i=1}^{M} \lambda_i x_i(n) \tag{2.18}$$

Using (2.15) and (2.16), from (2.9) we can write,

$$\boldsymbol{X}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{\Lambda})\,\boldsymbol{X}(n)\,(\boldsymbol{I} - \mu\boldsymbol{\Lambda}) + \mu^2 J_{min}\boldsymbol{\Lambda} \tag{2.19}$$

from which we have,

$$x_i(n) = (1 - \mu^2\lambda_i)^2 x_i(n) + \mu^2 J_{min}\lambda_i \quad i = 1, \ldots, M \tag{2.20}$$

Using the notation,

$$\begin{aligned} \boldsymbol{x}(n) &= \begin{bmatrix} x_1(n) & \ldots & x_M(n) \end{bmatrix}^T \\ \boldsymbol{\lambda} &= \begin{bmatrix} \lambda_1 & \ldots & \lambda_M \end{bmatrix}^T \\ b_{ij} &= \begin{cases} (1-\lambda_i)^1 & i = j \\ \mu^2\lambda_i\lambda_j & i \neq j \end{cases} \end{aligned} \tag{2.21}$$

we have,

$$\boldsymbol{x}(n+1) = \boldsymbol{B}\boldsymbol{x}(n) + \mu^2 J_{min}\boldsymbol{\Lambda} \tag{2.22}$$

where $\boldsymbol{B} = [b_{ij}]$. The solution to the difference equation in (2.22) is [Haykin, 2008],

$$\boldsymbol{x}(n) = \sum_{i=1}^{M} c_i^n \boldsymbol{g}_i \boldsymbol{g}_i^T \left[\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\right] + \boldsymbol{x}(\infty) \tag{2.23}$$

where $(c_i, \boldsymbol{g}_i)$ is the eigenpairs of $\boldsymbol{B}$, and $x(0)$ and $\boldsymbol{x}(\infty)$ is the initial and final values of $\boldsymbol{x}(n)$, respectively. Thus we have,

$$\begin{aligned} J(n) = \boldsymbol{\lambda}^T\boldsymbol{x}(n) &= \sum_{i=1}^{M} c_i^n \boldsymbol{g}_i \boldsymbol{g}_i^T \left[\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\right] + \boldsymbol{\lambda}^T\boldsymbol{x}(\infty) \\ &= \sum_{i=1}^{M} c_i^n \gamma_i + J_{ex}(n) \end{aligned} \tag{2.24}$$

where, $\gamma_i = \boldsymbol{\lambda}^T \boldsymbol{g}_i \boldsymbol{g}_i^T \left[\boldsymbol{x}(0) - \boldsymbol{x}(\infty)\right]$ and $J_{ex}(\infty) = \boldsymbol{\lambda}^T\boldsymbol{x}(\infty)$.

The LMS algorithm can be shown to be stable if the step size μ satisfies,

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.25}$$

where λ_{max} is the largest eigenvalue of $\boldsymbol{R}$ [Haykin, 2008], which may not always be available. In that case $tr\,[\boldsymbol{R}]$ is good estimate [Haykin, 2008]. Since,

$$tr\,[\boldsymbol{R}] = Mr(0) = \sum_{k=0}^{M-1} E\left[|u(n-k)|^2\right] = \text{tap input power} \tag{2.26}$$

for the stability of the LMS algorithm, the step size μ must satisfy,

$$0 < \mu < \frac{2}{\text{tap input power}} \tag{2.27}$$

With the discussion presented in this chapter in mind, the summary of the LMS algorithm is given in Table 2.1.

Table 2.1: Summary of the LMS algortihm.

Parameters:	M = the number of taps μ = step size parameter $0 < \mu < \dfrac{2}{\text{tap input power}}$ tap input power $= \sum_{k=0}^{M-1} E\left[\lvert u(n-k)\rvert^2\right]$
Initialization:	If prior knowledge on the tap-weight vector $\hat{\boldsymbol{w}}(n)$ is available, use it to select an appropriate value for $\hat{\boldsymbol{w}}(0)$. Otherwise, set $\hat{\boldsymbol{w}}(0) = \mathbf{0}$.
Given:	$\boldsymbol{u}(n)$ = M-by-1 tap input vector at time n. $d(n)$ = desire response at time n.
To be computed:	$\hat{\boldsymbol{w}}(n+1)$ = estimate of the tap-weight vector at time $n+1$
Computation:	for $n = 0, 1, \ldots$ compute $e(n) = d(n) - \hat{\boldsymbol{w}}^H(n)\boldsymbol{u}(n)$ $\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \mu\boldsymbol{u}(n)e^*(n)$

Chapter 3

Recursive Least Squares Algorithm

The recursive least squares(RLS) algorithm is the extension of the method of least squares such that given the least squares estimate of the tap weight vector at iteration $n-1$, updated estimate of this vector is computed at iteration n, hence the algorithm is called as recursive least squares. The rate of convergence of the RLS algorithm faster than that of the LMS algorithm at the expense of the increased computational complexity.

3.1 Recursive Least Squares Algorithm Adaptation

Given a transversal filter with tap-weights $w_0(n), \ldots, w_M(n)$, tap-inputs $u(n), \ldots, u(n-M+1)$ and the desired response $d(n)$, the RLS algorithm minimizes the exponentially-weighted least squares, i.e.,

$$\mathfrak{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \tag{3.1}$$

where $0 < \lambda < 1$. The optimum value of the tap-weights $\hat{\boldsymbol{w}}(n)$ is found from the *normal equations*,

$$\boldsymbol{\Phi}(n)\hat{\boldsymbol{w}}(n) = \boldsymbol{z}(n) \tag{3.2}$$

where $\boldsymbol{\Phi}(n)$ is the exponentially -weighted time average of the correlation matrix of the tap-input vector $\boldsymbol{u}(n) = \begin{bmatrix} u(n) & \ldots & u(n-M+1) \end{bmatrix}$,

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \boldsymbol{u}(i)\boldsymbol{u}^H(i) \tag{3.3}$$

and $\boldsymbol{z}(n)$ is the exponentially-weighted time averaged cross correlation vector between the tap-input vector $\boldsymbol{u}(n)$ and the desired response $d(n)$,

$$\boldsymbol{z}(n) = \sum_{i=1}^{n} \lambda^{n-i} u(i) d^*(i) \tag{3.4}$$

From (3.3) and (3.4), the following update equations,

$$\boldsymbol{\Phi}(n) = \lambda \boldsymbol{\Phi}(n-1) + \boldsymbol{u}(n)\boldsymbol{u}^H(n) \tag{3.5}$$

and

$$\boldsymbol{z}(n) = \lambda \boldsymbol{z}(n-1) + \boldsymbol{u}(n) d^*(n) \tag{3.6}$$

can be written.

In order to derive the update equation for the tap-weight vector we will use the matrix inversion lemma which states that given two positive definite matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ related by,

$$\boldsymbol{A} = \boldsymbol{B}^{-1} + \boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{C}^{H} \tag{3.7}$$

then the inverse $\boldsymbol{A}^{-1}$ of $\boldsymbol{A}$ is,

$$\boldsymbol{A}^{-1} = \boldsymbol{B} - \boldsymbol{B}\boldsymbol{C}\left(\boldsymbol{D} + \boldsymbol{C}^{H}\boldsymbol{B}\boldsymbol{C}^{H}\right)^{-1}\boldsymbol{C}^{H}\boldsymbol{B} \tag{3.8}$$

By substituting,

$$\boldsymbol{A} = \boldsymbol{\Phi} \qquad \boldsymbol{B}^{-1} = \lambda\boldsymbol{\Phi}(n-1) \qquad \boldsymbol{C} = \boldsymbol{u}(n) \qquad \boldsymbol{D} = 1 \tag{3.9}$$

in (3.8) we have,

$$\boldsymbol{\Phi}^{-1}(n) = \lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1) - \frac{\lambda^{-2}\boldsymbol{\Phi}^{-1}(n-1)\boldsymbol{u}(n)\boldsymbol{u}^{H}(n)\boldsymbol{\Phi}^{-1}(n-1)}{1 + \lambda^{-1}\boldsymbol{u}^{H}(n)\boldsymbol{\Phi}^{-1}(n-1)\boldsymbol{u}(n)} \tag{3.10}$$

Using the notation,

$$\boldsymbol{P}(n) = \boldsymbol{\Phi}^{-1}(n) \tag{3.11}$$

and

$$\boldsymbol{k}(n) = \frac{\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{1 + \lambda^{-1}\boldsymbol{u}(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)} \tag{3.12}$$

(3.10) can be rewritten as,

$$\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{u}^{H}(n)\boldsymbol{P}(n-1) \tag{3.13}$$

From (2.9),

$$\boldsymbol{k}(n) = \left(\lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{u}^{H}(n)\boldsymbol{P}(n-1)\right)\boldsymbol{u}(n) = \boldsymbol{P}(n)\boldsymbol{u}(n) \tag{3.14}$$

From (3.2) we have,

$$\begin{aligned}\hat{\boldsymbol{w}}(n) &= \boldsymbol{\Phi}^{-1}(n)\boldsymbol{z}(n) = \boldsymbol{P}(n)\boldsymbol{z}(n) \\ &= \lambda\boldsymbol{P}(n)\boldsymbol{z}(n-1) + \boldsymbol{P}(n-1)\boldsymbol{u}(n)d^{*}(n)\end{aligned} \tag{3.15}$$

Substituting (3.13) for $\boldsymbol{P}(n)$ in the first on the right hand side of (3.15),

$$\begin{aligned}\hat{\boldsymbol{w}}(n) &= \boldsymbol{P}(n-1)\boldsymbol{z}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^{H}(n)\boldsymbol{P}(n-1)\boldsymbol{z}(n-1) + \boldsymbol{P}(n)\boldsymbol{u}(n)d^{*}(n) \\ &= \hat{\boldsymbol{w}}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^{H}(n)\hat{\boldsymbol{w}}(n-1) + \boldsymbol{P}(n)\boldsymbol{u}(n)d^{*}(n) \\ &= \hat{\boldsymbol{w}}(n-1) + \boldsymbol{k}(n)\left(d^{*}(n) - \boldsymbol{u}^{H}(n)\hat{\boldsymbol{w}}(n-1)\right) \\ &= \hat{\boldsymbol{w}}(n-1) + \boldsymbol{k}(n)\xi(n)\end{aligned} \tag{3.16}$$

where $\xi(n)$ is the *a priory* estimation error which is different from *a posteriori* estimation error $\boldsymbol{e}(n) = d(n) - \hat{\boldsymbol{w}}^{H}(n)\boldsymbol{u}(n)$

The RLS algorithm requires the initial value $\boldsymbol{P}(0)$ that assures the non-singularity of $\boldsymbol{\Phi}(n)$. For this purpose, $\Phi(n)$ is modified slightly,

$$\boldsymbol{\Phi}(n) = \sum_{i=1}^{n}\lambda^{n-i}\boldsymbol{u}(i)\boldsymbol{u}^{H}(i) + \delta\lambda^{i}\boldsymbol{I} \tag{3.17}$$

where $\boldsymbol{I}$ is the identity matrix and δ is small positive constant. Evaluating (3.17) at $n = 0$,

$$\boldsymbol{\Phi}(0) = \delta\boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{\Phi}^{-1}(0) = \boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I} \tag{3.18}$$

With the discussion presented in this chapter in mind, the summary of the RLS algorithm is given in Table 3.1.

Table 3.1: Summary of the RLS algortihm.

Parameters:	M = the number of taps δ = step size parameter(small positive constant)
Initialization:	If prior knowledge on the tap-weight vector $\hat{\boldsymbol{w}}(n)$ is available, use it to select an appropriate value for $\hat{\boldsymbol{w}}(0)$. Otherwise, set $\hat{\boldsymbol{w}}(0) = \mathbf{0}$. $\boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I}$
Given:	$\boldsymbol{u}(n) = M$-by-1 tap input vector at time n. $d(n)$ = desire response at time n.
To be computed:	$\hat{\boldsymbol{w}}(n+1)$ = estimate of the tap-weight vector at time $n+1$
Computation:	for $n = 0, 1, \ldots$ compute $\boldsymbol{k}(n) = \dfrac{\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{1+\lambda^{-1}\boldsymbol{u}^H(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)}$ $\xi(n) = d(n) - \hat{\boldsymbol{w}}(n-1)\boldsymbol{u}(n)$ $\hat{\boldsymbol{w}}(n) = \hat{\boldsymbol{w}}(n-1) + \boldsymbol{k}(n)\xi^*(n)$ $\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{u}^H(n)\boldsymbol{P}(n-1)$

Chapter 4

Simulation Results

This chapter is devoted to the simulation results of the communication system whose block diagram is redrawn in Figure 4.1. Here, the channel transfer function may take one of the

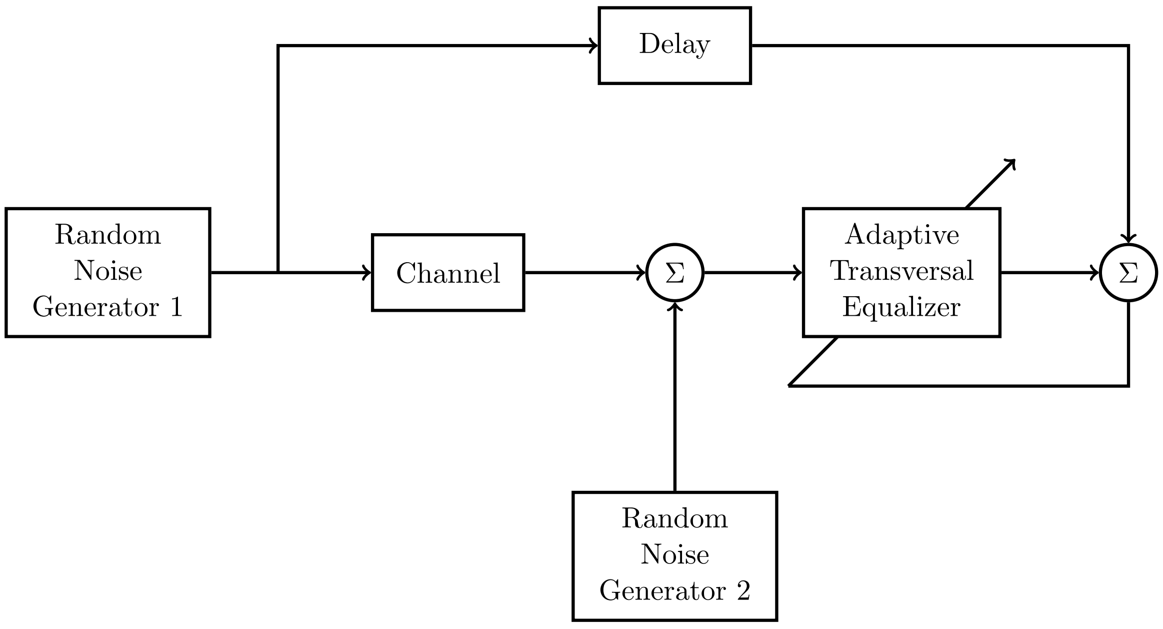


Figure 4.1: Block diagram of the channel equalizer.

following three possible forms,

(i) $H(z) = 0.25 + z^{-1} + 0.25z^{-2}$

(ii) $H(z) = 0.25 + z^{-1} - 0.25z^{-2}$

(iii) $H(z) = -0.25 + z^{-1} + 0.25z^{-2}$

whose impulse responses are plotted in Figure 4.2. The channel output in response to x_n is defined by,

$$u(n) = \sum_k h_k x_{n-k} + \nu(n) \tag{4.1}$$

where $h(n)$ is the impulse response of the channel and $\nu(n)$ is additive white Gaussian noise with zero mean and variance $\sigma_\nu^2 = 0.01$. The sequence x_n consists of the a Bernoulli sequence with $x_n = \pm 1$.

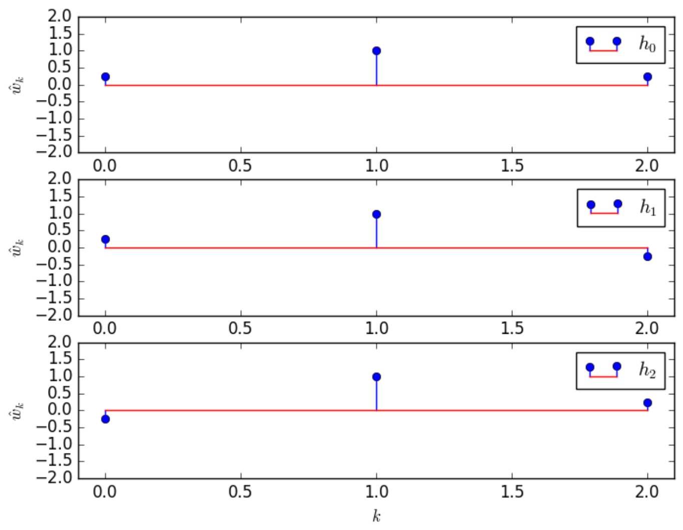


Figure 4.2: Impulse responses of the channels.

The simulations are carried out for an adaptive equalizer implemented with a transversal filter with 21 taps being symmetric about its center tap $n = 10$ will be used. Since the channel impulse responses are symmetric about it center tap $n = 1$, the amount of optimum delay $\Delta = 10 + 1 = 11$. Hence, the adaptive equalizer is provided with the desired response $d(n) = x(n - \Delta)$. The amount of delay Δ is determined to be equal to the midpoint of the impulse response of the channel and that of the adaptive filter so that the it is optimum in the sense that the LMS algorithm is enabled to provide an inversion of the minimum and non-minimum phase components of the channel respons [Haykin, 2008].

In the sequel, the performance of the system implemented with an equalizer trained by LMS algorithm and RLS algorithm are presented for different values values SNR levels and algorithm parameters. SNR level for the system can be defined as,

$$SNR = 10 \log \frac{\sigma_x^2}{\sigma_\nu^2} = 10 \log \frac{1}{\sigma_\nu^2} \tag{4.2}$$

where we have used the fact the variance $\sigma_x^2 = 1$ since $x(n) = \pm 1$ is a Bernoulli sequence.

4.1 LMS Adaptive Equalizer

In order to apply the LMS algorithm, initial filter taps $\hat{\boldsymbol{w}}(0)$ and the step size parameter μ must be determined first. Generally, the LMS algorithm is initialized with $\hat{\boldsymbol{w}}(0) = \mathbf{0}$ unless specific knowledge about $\hat{\boldsymbol{w}}(0)$ is available, which is the case here. The step size parameter μ is determined such that,

$$0 < \mu < \frac{2}{\lambda_{max}} \approx \frac{2}{Mr(0)} \tag{4.3}$$

where M is the number of taps of the filter M and $r(0)$ is the autocorrelation function $r(k)$ evaluated at lag $k = 0$. We have,

$$u(n) = \sum_{k=0}^{2} h_k x(n-k) + \nu(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + \nu(n) \tag{4.4}$$

Recalling that,

$$E(x(n-k)x(n-l)) = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases} \tag{4.5}$$

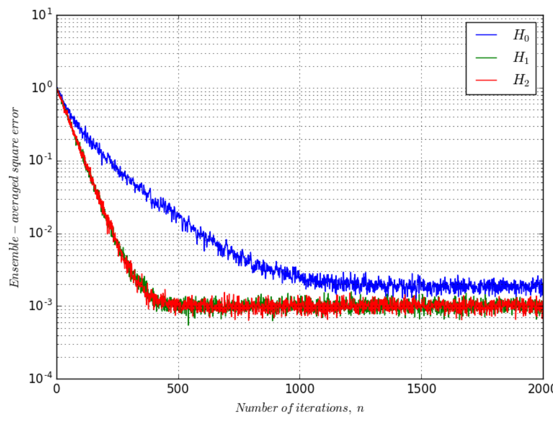
from (4.4),

$$r(0) = h_0^2 + h_1^2 + h_2^2 + \sigma_\nu^2 \tag{4.6}$$

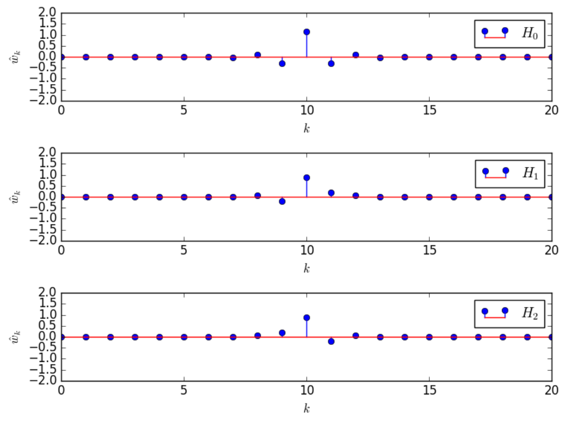
Thus, from (4.3) for the problem at hand for a given SNR, the step size parameter μ must satisfy,

$$0 < \mu < \mu_{crit} = \frac{2}{21(0.25^2 + 1^2 + 0.25^2 + 10^{-SNR/10})} \tag{4.7}$$

Figure 4.3 illustrates the ensemble averaged learning curves obtained by averaging the instantaneous squared error $e^2(n)$ and ensemble averaged the filter taps obtained by averaging the filter taps after the filter has converged. Ensemble averaging has been performed over 100 independent trials. As illustrated in Figure 4.3a, the equalizer has converged after approximately 1000 iterations with an averaged instantaneous error of 0.002 when the channel has transfer function H_0 while it has converged after approximately 400 iterations with an averaged instantaneous error of 0.001 when the channel has transfer function H_1 or H_2. Note from Figure 4.2 that the impulse of the channel modeled by H_0 is even symmetric while the channel modeled by H_1 or H_2 are odd symmetric about its center tap $n = 1$. Learning curves in Figure 4.3a depict that the equalizer converges faster when the channel impulse response is odd symmetric and the type of odd symmetry does not effect the convergence rate. Figure 4.3b shows that the equalizer impulse response is symmetric about its center tap $n = 10$, as expected. It worths noting that the type of symmetry that the equalizer impulse response has corresponds to the type of symmetry that the channel impulse response has, i.e. if the channel impulse response is even symmetric, then the equalizer impulse response is also even symmetric and if the channel impulse response is odd symmetric, then the equalizer impulse response is also odd symmetric.



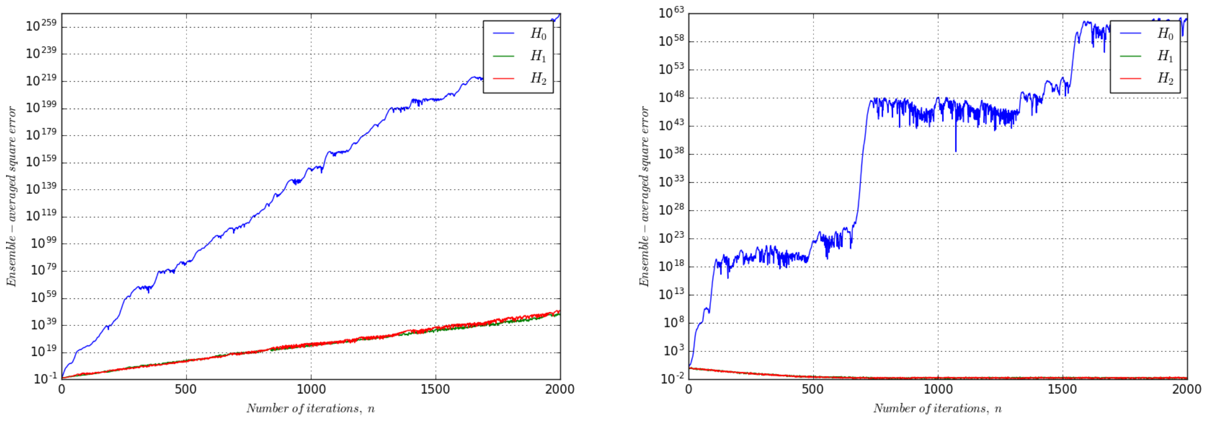
(a) Learning curves



(b) Filter tap weights

Figure 4.3: Equalizer trained by the LMS algorithm for SNR = 30 dB, $\mu = 0.01$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.

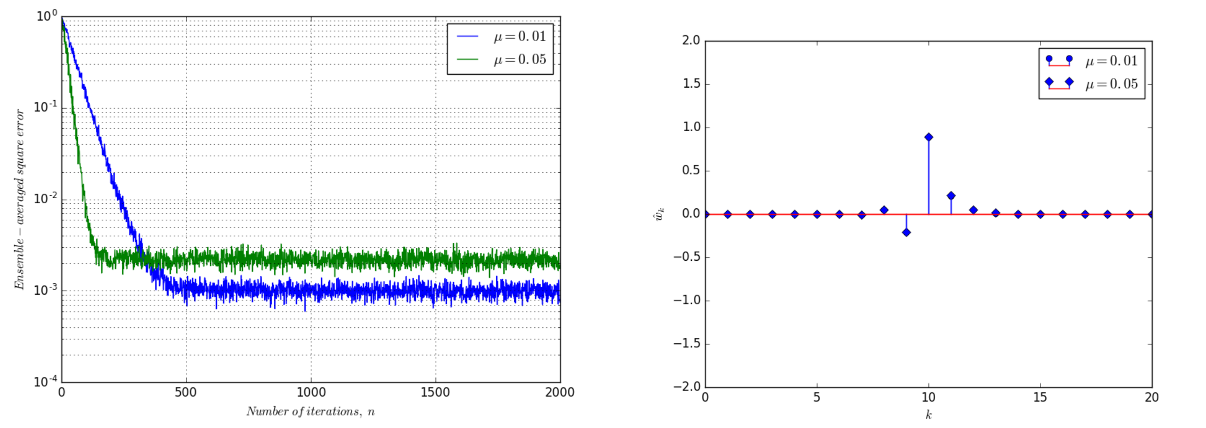
From (4.7) for SNR = 30 dB, $\mu_{\text{crit}} = 0.0846$. Figure 4.4 shows the cases when the step size does not satisfy the condition given in (4.7), i.e. for $\mu = 0.1$ and $\mu = 0.08$ which is greater than and close to μ_{crit}, respectively. Note from the Figure 4.4a that when μ does not satisfy the stability condition, the learning curves diverge and when μ is at verge of stability condition the learning curves converge when the channel impulse response is odd symmetric and the learning curve diverges when the impulse response is even symmetric.



(a) Learning curves for $\mu = 0.1$ (b) Learning curves for $\mu = 0.08$

Figure 4.4: Equalizer trained by the LMS algorithm for SNR = 30 dB for (a) $\mu = 0.1$ and (b) $\mu = 0.08$. Ensemble averaging has been performed over 100 independent trials.

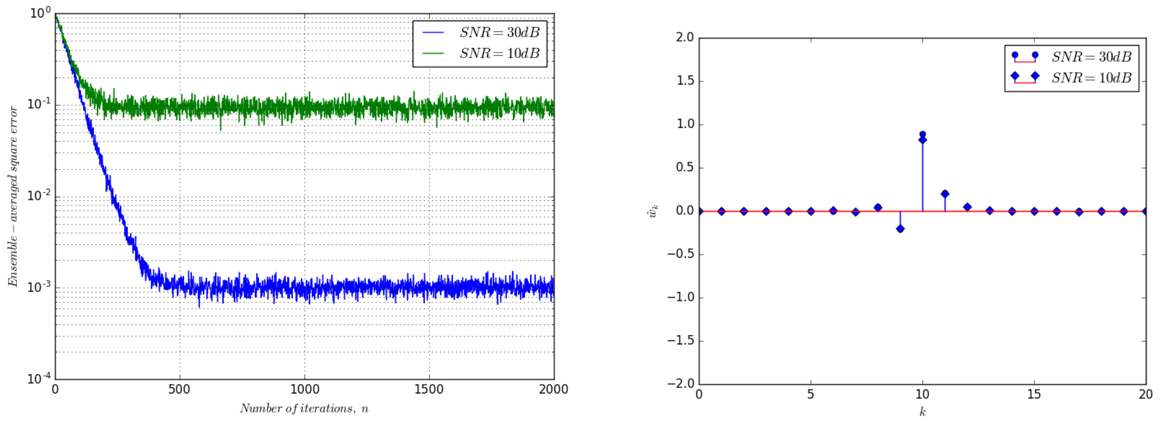
Figure 4.5 compares the convergence rate of the LMS algorithm for the step size $\mu = 0.01$ and $\mu = 0.05$ for SNR = 30 dB when the channel transfer function is H_1. From the Figure 4.5a, the LMS algorithm with a step size $\mu = 0.01$ has converged after approximately 400 iterations with an averaged instantaneous error 0.001 while the LMS algorithm with a step size $\mu = 0.05$ has converged after approximately 200 iterations with an averaged instantaneous error 0.002. Hence, increased step size (of course satisfying the stability condition) implies increased convergence rate in the expense of increased instantaneous error.



(a) Learning curves (b) Filter tap weights

Figure 4.5: Equalizer trained by the LMS algorithm when the channel transfer function is H_1 for SNR = 30 dB, $\mu = 0.01$ and $\mu = 0.05$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.

Figure 4.6 compares the performance of the LMS algorithm for SNR = 30 dB and SNR = 10 dB for a step size $\mu = 0.01$. From Figure 4.6a, it seen that the equalizer has converged after approximately 500 iterations with averaged error 0.001 with an SNR = 30 while it has converged after approximately 200 iterations with averaged error 0.1 with an SNR = 10 dB. Hence, the equalizer converges much faster with larger average instantaneous error in response to SNR increase.



(a) Learning curves (b) Filter tap weights

Figure 4.6: Equalizer trained by the LMS algorithm when the channel transfer function is H_1 for SNR = 30 dB and SNR = 10 dB and $\mu = 0.01$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.

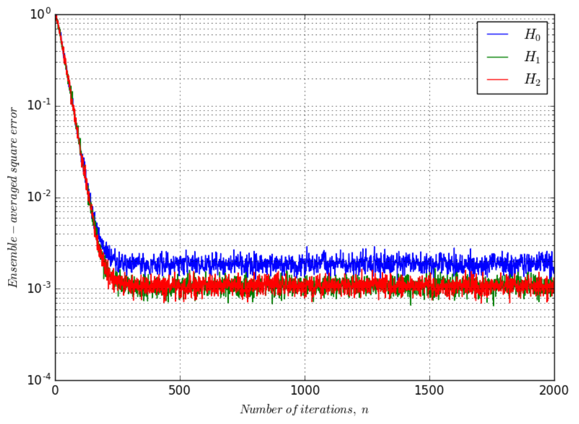
4.2 RLS Adaptive Equalizer

As in the case of LMS algorithm, RLS algorithm is generally initialized with $\hat{\boldsymbol{w}}(0) = \mathbf{0}$ unless a prior knowledge of $\hat{\boldsymbol{w}}(0)$ is available. The forgetting factor λ must satisfy $0 < \lambda < 1$ and is generally chosen close to but close to 1. The regularization parameter δ is a small positive number. One final remark about RLS algorithm is that its learning curves are obtained by ensemble averaging the instantaneous squared a priori error $\xi^2(n)$ for the purpose of comparison of its performance to that of LMS algorithm.

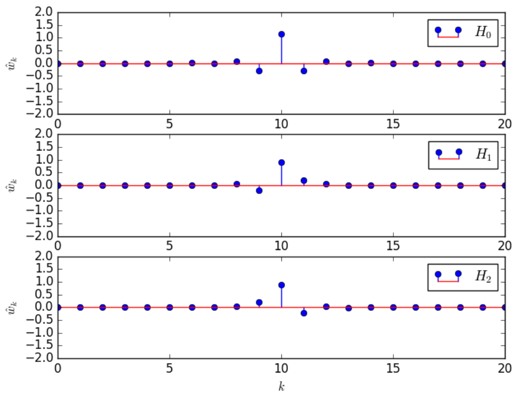
Figure 4.7 shows the learning curves and converged tap weights obtained by ensemble averaging the instantaneous squared a priori error $\xi^2(n)$ over 100 independent trials. From Figure 4.7a it is seen that the equalizer converges after approximately 250 iterations for all three of the channel transfer functions. However, the averaged instantaneous squared a priori errors differs depending on the symmetry of the channel impulse response: the averaged instantaneous squared a priori error is 0.002 if the channel transfer function is H_0 which is even symmetric and is 0.001 if the channel transfer function is H_0 or H_1 which are odd symmetric. Note from Figure 4.7b that the impulse response of the equalizer is even symmetric if the channel response is even symmetric and is odd symmetric if the channel response is odd symmetric, being symmetric about its center tap $n = 10$.

Figure 4.8 investigate the case when the forgetting factor does not satisfy the condition $0 < \lambda < 1$. Figure 4.8a and Figure 4.8b shows the learning curve when the forgetting factor $\lambda = 1.0$ and $\lambda = 1.5$, respectively, for SNR = 30 dB, $\delta = 0.005$ and illustrates that when magnitude of the forgetting factor λ is not less then 1, the RLS algorithm is not asymptotically stable.

Figure 4.9 compares the performance of the RLS algorithm for $\lambda = 0.98$ and $\lambda = 0.85$ and for $\delta = 0.005$, SNR = 30 dB for the channel with transfer function H_1. Note from Figure 4.9a that the equalizer converges after approximately 50 iterations with an averaged instantaneous

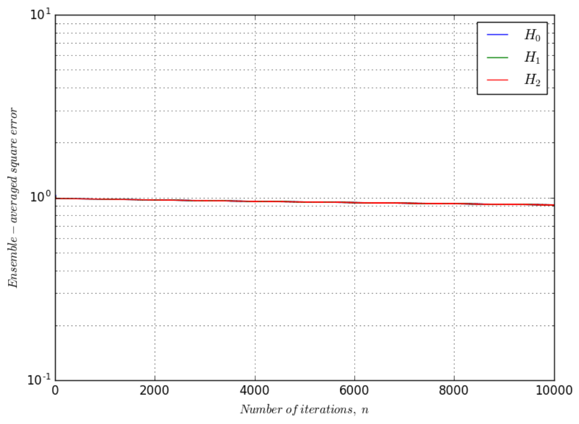


(a) Learning curves

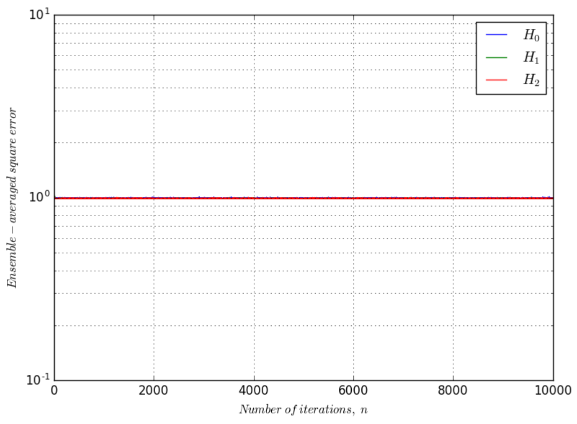


(b) Filter tap weights

Figure 4.7: Equalizer trained by the RLS algorithm for SNR = 30 dB, $\delta = 0.005$, $\lambda = 0.98$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.



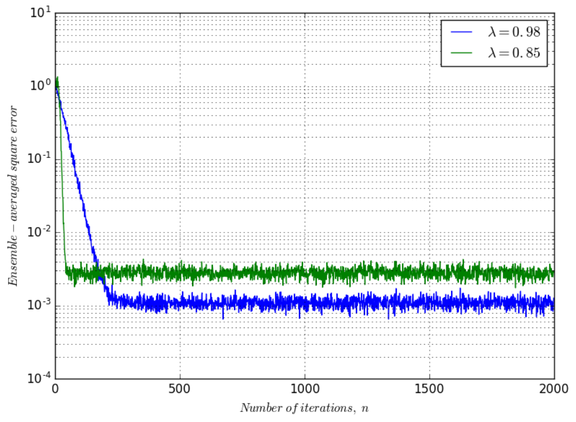
(a) Learning curves for $\lambda = 1.0$



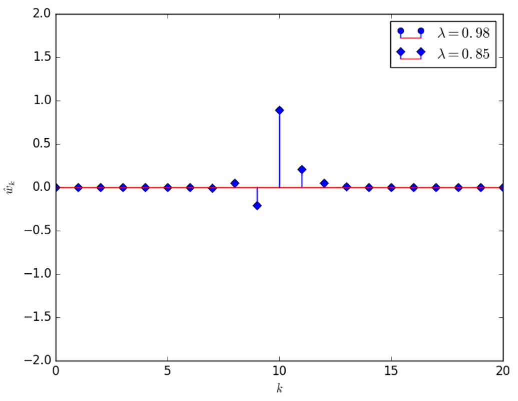
(b) Learning curves for $\lambda = 1.5$

Figure 4.8: Equalizer trained by the RLS algorithm for SNR = 30 dB for (a) $\lambda = 1.0$ and (b) $\lambda = 1.5$. Ensemble averaging has been performed over 100 independent trials.

squared a priori error 0.003 if the forgetting factor $\lambda = 0.85$ while it converges after 250 iterations with an averaged instantaneous squared error 0.001 if the forgetting factor $\lambda = 0.89$. The results in Figure 4.9 depict that increased forgetting factor reduces the averaged instantaneous squared a priori error in the expense of reduction in the convergence rate, as the name *forgetting factor* implies.



(a) Learning curves

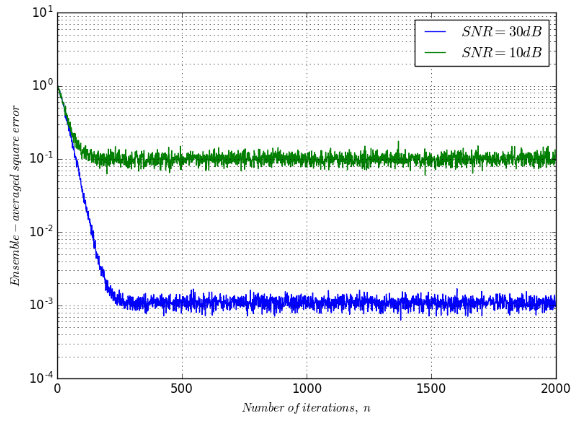


(b) Filter tap weights

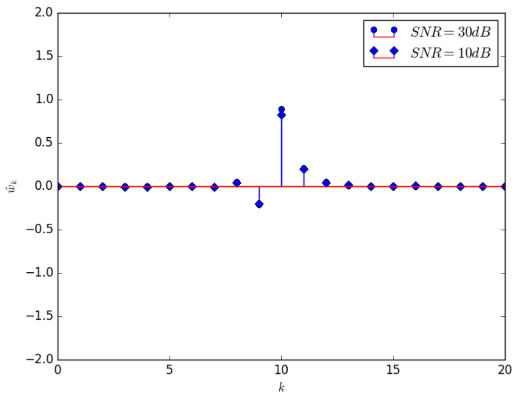
Figure 4.9: Equalizer trained by the RLS algorithm when the channel transfer function is H_1 for SNR = 30 dB, $\lambda = 0.98$ and $\mu = 0.85$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.

Figure 4.10 investigates the performance of the RLS algorithm under different SNR values by illustrating the results for SNR = 30 dB and SNR = 10 dB, $\lambda = 0.98$, $\delta = 0.005$ when the channel transfer function is H_1. The learning curves in Figure 4.10a show that the equalizer converges after approximately 250 iterations. However, the equalizer converges with an averaged instantaneous squared a priori error of 0.1 for SNR = 10 dB and of 0.001 for SNR = 30 dB. This implies that increased error ambient noise deteriorates the filter convergence, the larger the amount of noise is the larger the averaged instantaneous squared a priori error is.

Figure 4.11 compares the performance of RLS and LMS algorithms for SNR = 30 dB, $\delta = 0.005$ and $\mu = 0.01$ when the channel transfer function is H_1. Since performance of RLS algorithm for λ close to 1 approaches to the performance of the LMS algorithm, $\lambda = 0.98$ has been chosen for the purpose of the comparison of the RLS and LMS algorithms. Figure 4.11a shows that for the given parameter values RLS and LMS algorithms converges to the same averaged instantaneous error 0.001. However, the convergence rate of RLS algorithm is superior than that of LMS algorithm, since RLS converged after approximately 250 iterations while LMS converged after approximately 500 iterations.

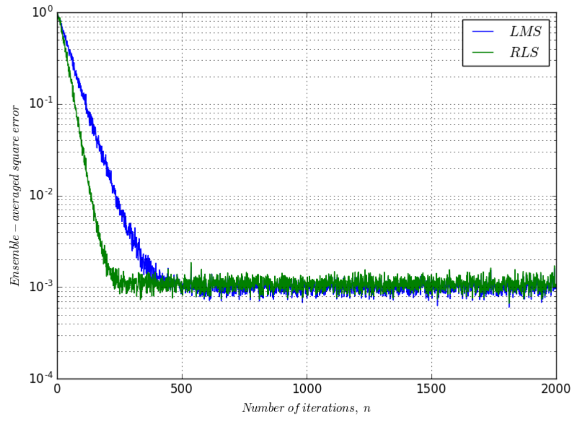


(a) Learning curves

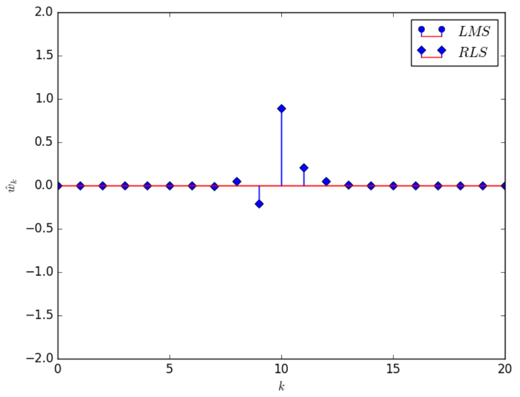


(b) Filter tap weights

Figure 4.10: Equalizer trained by the RLS algorithm when the channel transfer function is H_1 for SNR = 30 dB and SNR = 10 dB, $\lambda = 0.98$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.



(a) Learning curves



(b) Filter tap weights

Figure 4.11: Equalizer trained by the LMS algorithm and RLS algorithm when the channel transfer function is H_1 for SNR = 30 dB, $\mu = 0.01$, $\lambda = 0.98$. (a) Ensemble averaged learning curves and (b) ensemble averaged filter tap weights over 100 independent trials.

Chapter 5

Conclusion

To conclude, adaptive channel equalization has been investigated in the context of this study. In order to eliminate the intersymbol interference caused by the dispersive nature of the communication channel, an equalizer has been implemented by means of tapped-delay-line transversal filter. The performance of the equalizer trained by LMS and RLS algorithms have been queried and compared to each other.

Throughout the simulations, it has seen that the equalizer impulse response is symmetric about its center tap, as expected. The symmetry of the impulse response of the equalizer corresponds to the symmetry of the channel impulse response, since cascaded connection of the channel with the equalizer must have an impulse response that forces the receiver output to be zero at all the sampling instances, except for the time instant that corresponds to the transmitted pulse.

Also worths noting from the simulations is that channel impulse symmetry matters in case of LMS adaptive filtering in that the equalizer converges faster if the channel impulse response is odd symmetric than the case if the channel impulse response is even symmetric.

One also thinks that the ambient noise, being another inherent property of the channel apart from the intersymbol interference, affects the convergence property of the adaptive filtering. Numerical simulations have confirmed this thought by illustrating that the average instantaneous squared error has been increased in response to an increase in the ambient noise.

One last remark about the study is that the LMS algorithm is simpler in terms of in terms of computational complexity but slower in terms of rate of convergence when compared to the RLS algorithm.

Bibliography

[Haykin, 2008] Haykin, S. S. (2008). *Adaptive filter theory.* Pearson Education India.