EECS 862 PROJECT II - EQUALIZER SIMULATION

VIJAYA CHANDRAN RAMASAMI (KUID 698659)

Contents

List of Figures

1. Linear Equalization

An Equalizer is a *compensator* for Channel Distortion. For communication channels in which the channel characterestics are unknown or time-varying, optimum transmit and receive filters cannot be designed directly. For such channels, an equalizer is needed to compensate for the ISI created by the distortion in the channel. There are three types of equalization methods commonly used :

- Maximum Likelihood (ML) Sequence Detection - Optimal, but Impractical.
- Linear Equalization - Suboptimal, but simple.
- Non-Linear Equalization (DFE)- for severe ISI channels.

Linear Equalizers are simple to implement and are highly effective in channels where is the ISI is not severe (like the wireline telephone channel). Most linear equalizers are implemented as a linear transversal filter, shown in figure(1).

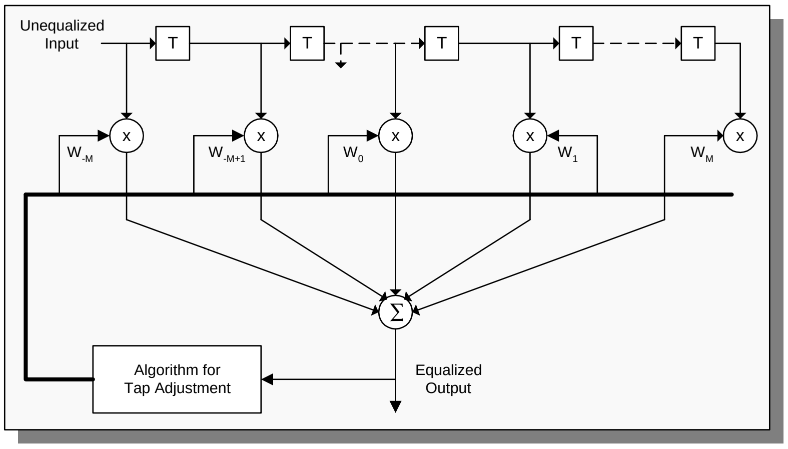

Figure 1. Linear Transversal Equalizer

Where, the number of equalizer taps is $2M+1$ and T is the symbol duration. If $Y(t)$ is the input to the equalizer, then the output of the equalizer is given by,

$$Y_{eq}(t) = \sum_{-M}^{i=M} w_i Y(t - iT) \tag{1}$$

Where, w_i are the complex equalizer tap weights selected based on some optimization criterion.

1.1. **Criterion for Optimization.**

Two criteria are commonly used for optimizing the equalizer tap weights :

- Peak Distortion Criterion - leading to the Zero-Forcing Equalizer.
- Mean Square Error (MSE) Criterion - leading to the LMMSE equalizer and the Gradient (LMS) algorithm.

The aim of this project is to setup and simulate both the Zero-Forcing Equalizer and the LMS algorithm.

1.2. **Weight Adaptation.** Linear Equalizers are further classified into two types based on weight adaptation :

- *Preset Equalizers*, used for channels in which the frequency response charaterestics are unknown, but invariant (such as the telephone channel). The Weights are calculated only once (in the beginning of the session) are not varied during the session.
- *Adaptive Equalizers*, used for channels in which the frequency response in time-variant. These equalizers are capable of tracking a slowly time-varying channel by updating their parameters on a periodic basis.

2. Zero Forcing Equalizer

The Zero-Forcing Equalizer belongs to the class of *preset* linear equalizers and it uses the *Peak Distortion Criterion* to evaluate the equalizer tap weights.

Consider the communication system block diagram (with an equalizer) given in figure(2).

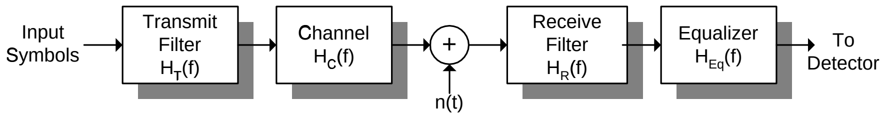

Figure 2. Communication System Block Diagram (with Equalizer)

Assuming that a *raised-cosine* pulse shape is used, the condition for zero-ISI is given by,

$$H_T f H_C(f) H_R(f) H_{Eq}(f) = H_{rc}(f) \tag{2}$$

We have by definition, $H_T(f)H_R(f) = H_{rc}(f)$. So the value of $H_{Eq}(f)$ that compensates for the channel distortion $H_C(f)$ is given by,

$$H_{Eq}(f) = \frac{1}{H_C(f)} \tag{3}$$

This equalizer is also called the *inverse channel* equalizer. The discrete time version of the above equation (3), with a sampling rate equal to the symbol-duration T, is given by,

$$1 = H_{Eq}(z)H_C(z) \tag{4}$$

Or, in the discrete-time domain, we have,

$$\sum_{i=-\infty}^{\infty} w_j h(n-j) = p_{eq}(n) = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{if } n \neq 0 \end{cases} \tag{5}$$

Where, h(n) is the (discrete-time) impulse response of the channel and $p_{Eq}(t)$ is the response after equalization. Since, the filter *forces* due the ISI to zero, it is also called the *Zero-Forcing Equalizer.* In practice, since the ISI caused by the channel is limited to a *finite* number of symbols on either side of the desired symbol, it required to force the ISI to zero only at those sampling instants. The result is a finite duration (FIR) traversal filter as shown in fig(1).

2.1. Derivation of the Filter Coeffcients.

Let $p_r(t)$ represents the unequalized pulse corresponding to the cascade of $H_T(f)$, $H_C(f)$, $H_R(f)$ and the input (test) pulse $p(t)$ into the channel. Let the length of the transversal filter be $2M+1$.The equalized output pulse is given by,

$$p_{eq}(t) = \sum_{i=-M}^{M} w_i p_r(t - iT) \tag{6}$$

The Zero-Forcing condition is is now applied to the samples of $p_{Eq}(t)$ taken at sampling times $t = mT$.

$$p_{eq}(mT) = \sum_{i=-M}^{M} w_i p_r((m-i)T) = \begin{cases} 1, \text{if } m = 0 \\ 0, \text{if } m = 0, \pm 1, \pm 2, \ldots, \pm M \end{cases} \tag{7}$$

Which results in a set of $2M+1$ simultaneous equations, whose solution is given by,

$$\begin{pmatrix} w_{-M} \\ . \\ . \\ w_0 \\ . \\ . \\ w_M \end{pmatrix} = \begin{pmatrix} p_r(0) & p_r(-1) & \dots & p_r(-2M) \\ p_r(1) & p_r(0) & \dots & p_r(-2M+1) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ p_r(2M-1) & \dots & p_r(0) & p_r(-1) \\ p_r(2M) & \dots & p_r(1) & p_r(0) \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ . \\ . \\ 1 \\ . \\ . \\ 0 \end{pmatrix} \tag{8}$$

It is important to note that the equalizer designed using the above equation (8) does not completely eliminate ISI, as it has a finite length. The remaining or residual ISI can be reduced by increasing M, but a larger M will increase the complexity of both the equalizer design and implementation. As $M \to \infty$, ISI is completely eliminated.

2.2. **Diadvantage.** The Zero-Forcing Equalizer design does not take into account the effect of additive noise. Since the equalizer frequency response is approximately the inverse of the channel's frequency response (which is usually low-pass), it will cause a significant *enhancement in the noise power* at *high frequencies.* So, an equalizer designed with both Noise and ISI taken into account will offer a much better performance than the Zero-Forcing Equalizer.

2.3. **Design and Results.**

2.3.1. *Channel Sounding.* The Impulse response of the given channel (4th order Butterworth LPF) was evaluated using channel sounding as :

m	$p_r(mTb)$
-6	0
-5	0
-4	0
-3	0
-2	0
-1	0.1234
0	0.8255
1	0.1131
2	-0.0912
3	0.0358
4	-0.0057
5	-0.0023
6	0.0021

The input pulse in the channel $p(t)$ and the output pulse $p_r(t)$ are plotted in figure(3). The ISI due to the channel and the channel delay are evident from the plots. It fact, the optimal sampling point of the pulse was computed to be 27 samples with respect to a common time-origin. Using the

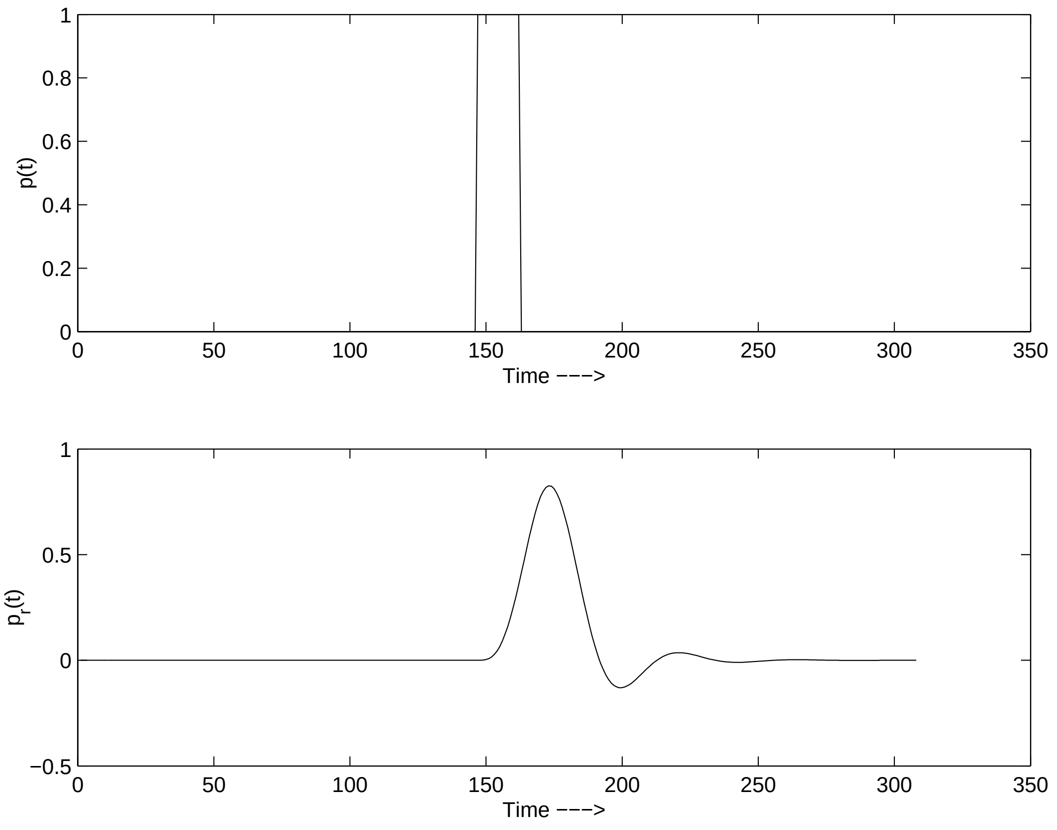

FIGURE 3. Channel Sounding

tabulated values in the above equation(8), we get,

$$\begin{pmatrix} w_{-3} \\ w_{-2} \\ w_{-1} \\ w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 0.8255 & 0.1234 & 0 & 0 & 0 & 0 & 0 \\ 0.1131 & 0.8255 & 0.1234 & 0 & 0 & 0 & 0 \\ -0.0912 & 0.1131 & 0.8255 & 0.1234 & 0 & 0 & 0 \\ 0.0358 & -0.0912 & 0.1131 & 0.8255 & 0.1234 & 0 & 0 \\ -0.0057 & 0.0358 & -0.0912 & 0.1131 & 0.8255 & 0.1234 & 0 \\ -0.0023 & -0.0057 & 0.0358 & -0.0912 & 0.1131 & 0.8255 & 0.1234 \\ 0.0021 & -0.0023 & -0.0057 & 0.0358 & -0.0912 & 0.1131 & 0.8255 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{9}$$

The Zero-Forcing Equalizer Weights are computed as,

$$\begin{pmatrix} w_{-3} \\ w_{-2} \\ w_{-1} \\ w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -0.0045 \\ 0.0298 \\ -0.1952 \\ 1.2755 \\ -0.2271 \\ 0.1970 \\ -0.1087 \end{pmatrix} \tag{10}$$

The eye-diagrams at the equalizer input and output are plotted in figures (4) and (5).

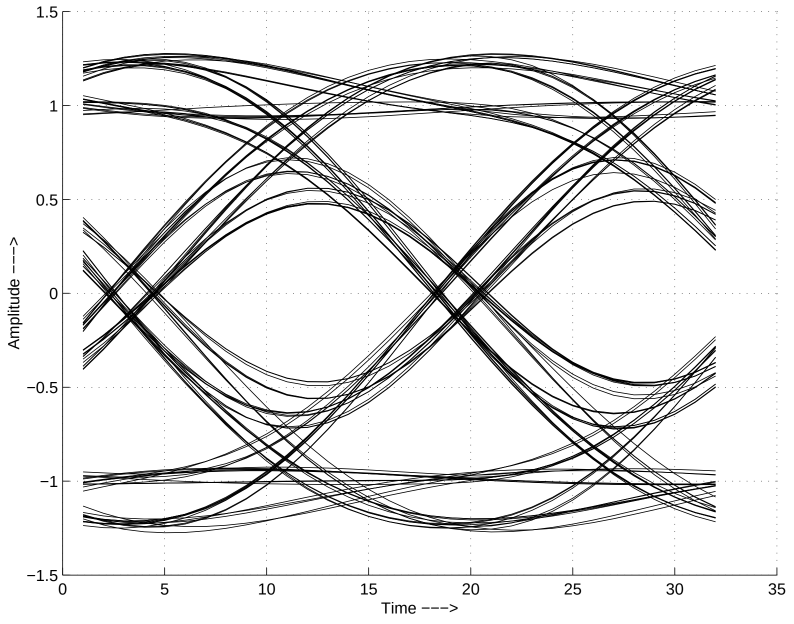

Figure 4. Eye Diagram at the Equalizer Input

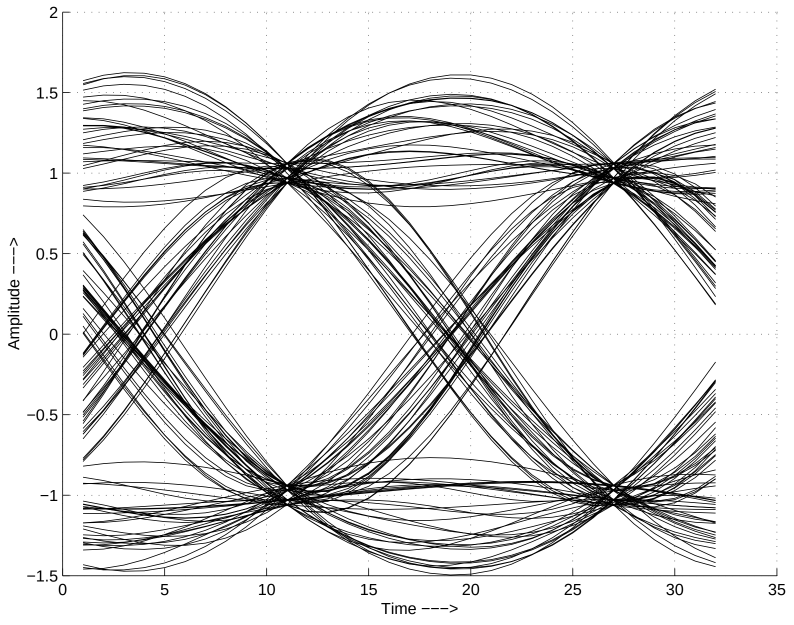

Figure 5. Eye Diagram at the Equalizer Output

It is clear from the eye-diagram plots that the equalizer has indeed reduced the ISI by a considerable amount and the *eye-opening* has increased. A comparision of the original and the equalized pulses is shown in figure(6) to illustrate the reduction of ISI.

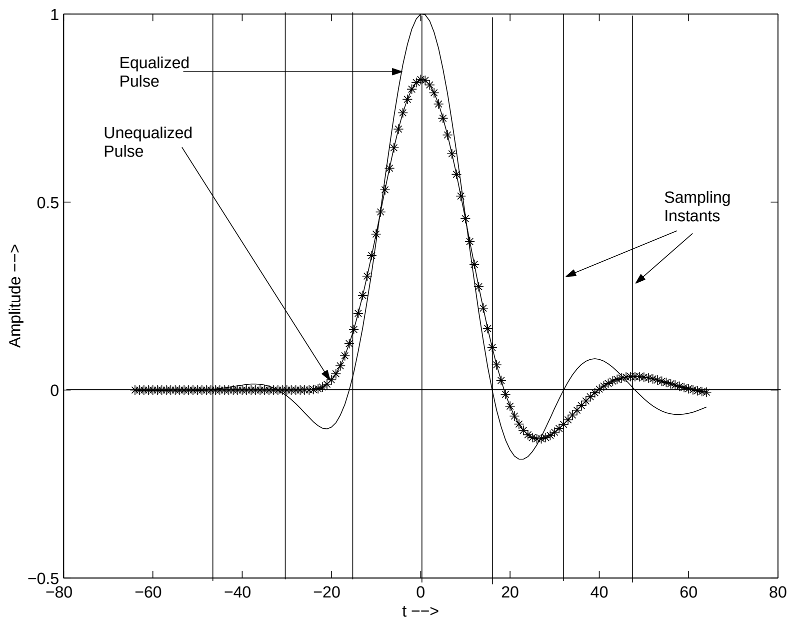

FIGURE 6. Comparision of the Unequalized and Equalized Pulses

2.4. **MSE at Sampling Times.** Ignoring the first 32 bits from the equalizer input and output, the MSE was estimated as :

$$\text{MSE (@ input)} = 0.0680 \tag{11}$$

$$\text{MSE (@ output)} = 0.0023 \tag{12}$$

3. Linear Minimum Mean Squared Error (LMMSE) Filter

The Zero-Forcing Equalizer has a severe drawback due to its noise performance. The LMMSE filter overcomes this drawback by *relaxing* the zero-ISI condition and selecting the equalizer characterestic such that the *combined* power in the ISI and the additive noise at the equalizer output is minimized.

3.1. Basic Assumptions.

The following are assumed in the derivation of the LMMSE Equalizer.

- The input symbols are temporally uncorrelated, (i.e), $E\left\{A_n A_{n+j}\right\} = \delta(j)$
- The input symbols are uncorrelated with noise.

3.2. Derivation of the Equalizer Tap Weights.

Consider the input to the equalizer,

$$Y(n) = \sum_{k=-\infty}^{\infty} A_k p_r(n-k) + N(n) \tag{13}$$

And the equalizer output,

$$Y_{eq}(n) = \sum_{i=-M}^{M} w_i Y(n-i) \tag{14}$$

The Equalizer output at the sampling times are given by,

$$Y_{eq}(n) = \sum_{i=-M}^{M} w_i \left[\sum_{k=-\infty}^{\infty} A_k p_r(n-i-k) + n(n-i) \right] \tag{15}$$

The quantity to be minimized the mean squared error (MSE) given by,

$$MSE = E\left\{(A_n - Y_{eq}(n))^2\right\} \tag{16}$$

Using the above equations, we can re-write the expression for the MSE as,

$$MSE = E\left\{A_n^2\right\} - 2\sum_{i=-M}^{M} w_i E\left\{A_n Y(n-i)\right\} + E\left\{\left(\sum_{i=-M}^{M} w_i Y(n-i)\right)^2\right\} \tag{17}$$

Further simplification to the above equation (17) is possible by using the following facts :

- $E\left\{A_n^2\right\} = \sigma_A^2$ (by definition).
- $E\left\{A_n Y(n-i)\right\} = \sigma_A^2 p_r(-i)$. (using the basic assumptions).
- $E\left\{Y(n-i)Y(n-j)\right\} = \sigma_A^2 \gamma(i,j) + \sigma_N^2 \rho(i-j)$. (using the basic assumptions).
  Where,
  - $\gamma(i,j) = \sum_{l=-\infty}^{\infty} p_r(l-i)p_r(l-j)$ represents the auto-correlation of the unequalized pulse values.
  - $\rho(i,j) = E\left\{n(n-i)n(n-j)\right\}$ represents the auto-correlation of the receiver input noise.
  - σ_A^2 represents the Average Signal Power.
  - σ_N^2 represents the Average Noise Power.

The equation (17) can be converted into a compact matrix notation using these following definitions :

- $\mathbf{P}_r = [p_r(M), \ldots, p_r(0), \ldots, p_r(-M)]^T$ are the received pulse values.
- $\mathbf{\Gamma}$ is a matrix whose $(i,j)^{th}$ element is $\gamma(i,j) + (\sigma_N^2/\sigma_A^2)\rho(i-j)$.
- $\mathbf{w} = [w_{-M}, \ldots, w_0, \ldots, w_M]^T$ represents the equalizer tap-weights.

The final expression for the MSE is,

$$MSE = \sigma_A^2 \left(1 - 2\mathbf{P}_r^T \mathbf{w} + \mathbf{w}^T \mathbf{\Gamma} \mathbf{w}\right) \tag{18}$$

Differentiating the above equation (18) with respect to the equalizer tap weights w_i and setting the resulting expressions to zero, we arrive at a set of simultaneous equations given by,

$$\mathbf{P}_r = \mathbf{\Gamma}\mathbf{w} \tag{19}$$

Note that if the input noise is uncorrelated (or white) the above equation (19) becomes,

$$\mathbf{P}_r = (\mathbf{R} + \frac{\sigma_N^2}{\sigma_A^2}\mathbf{I})\mathbf{w} \tag{20}$$

Where, $\mathbf{R}$ is the auto-correlation matrix of the unequalized pulse values and I is a identity matrix. The equalizer tap-weights can be computed form the above equation (20) as :

$$\mathbf{w} = (\mathbf{R} + \frac{\sigma_N^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{P}_r \tag{21}$$

Recognizing the fact that $\sigma_A^2/\sigma_N^2 = SNR$, we can rewrite the above equation (21) as :

$$\mathbf{w} = (\mathbf{R} + \frac{1}{SNR}\mathbf{I})^{-1}\mathbf{P}_r \tag{22}$$

The above equation can be used to visualize the *trade-off* between Noise and ISI in the design of the LMMSE equalizer. If the receiver is operating in an almost noise-free condition ($SNR \to \infty$), the second term in the $\mathbf{\Gamma}$ matrix goes to zero and the solution approaches the zero-forcing equalizer solution. In all other cases, the noise term is weighed appropriately (based on the SNR).

3.3. **Existence of an optimal solution.** The existence of an optimal solutions basically rests with the ability to invert the Covariance ($\mathbf{\Gamma}$) matrix. In general, the Coavariance matrix is symmetric non-negative definite. The definiteness of the matrix is strengthened by the the additive white noise component that provides a strong diagnol component, thus preventing singularity.

3.4. **Residual MSE.** The Residual MSE after the equalization can be computed using the expression,

$$MSE = \sigma_A^2(1 - \mathbf{P}_r^T \mathbf{\Gamma} \mathbf{P}_r) \tag{23}$$

3.5. **Calculation and Results.** Using the data collected by *Channel sounding*, the R matrix was estimated as,

$$\mathbf{R} = \begin{pmatrix} 0.7192 & 0.1815 & -0.0569 & 0.0180 & -0.0008 & -0.0023 & 0.0013 \\ 0.1815 & 0.7192 & 0.1815 & -0.0569 & 0.0180 & -0.0008 & -0.0023 \\ -0.0569 & 0.1815 & 0.7192 & 0.1815 & -0.0569 & 0.0180 & -0.0008 \\ 0.0180 & -0.0569 & 0.1815 & 0.7192 & 0.1815 & -0.0569 & 0.0180 \\ -0.0008 & 0.0180 & -0.0569 & 0.1815 & 0.7192 & 0.1815 & -0.0569 \\ -0.0023 & -0.0008 & 0.0180 & -0.0569 & 0.1815 & 0.7192 & 0.1815 \\ 0.0013 & -0.0023 & -0.0008 & 0.0180 & -0.0569 & 0.1815 & 0.7192 \end{pmatrix} \tag{24}$$

The value of the signal-to-noise ratio, was computed from the value of E_b/N_o as,

$$SNR = \frac{\sigma_A^2}{\sigma_N^2} = \frac{1}{8}\left(\frac{E_b}{N_o}\right) \tag{25}$$

The tap-weights were computed directly from equation (22) and the results are tabulated in (1): The MSE was computed using equation (23) and the fact that $\sigma_A^2 = 16$ as,

m	$w_m(E_b/N_o = \infty)$	$w_m(E_b/N_o = 10)$	$w_m(E_b/N_o = 2)$	$w_m(E_b/N_o = 1)$
-3	-0.0039	0.0232	0.0079	0.0042
-2	0.0288	-0.0449	-0.0183	-0.0102
-1	-0.1932	0.0167	0.0183	0.0113
0	1.2714	0.5384	0.1732	0.0941
1	-0.2194	0.0155	0.0197	0.0123
2	0.1850	0.0191	0.0013	0.0003
3	-0.0959	-0.0081	-0.0005	-0.0001

TABLE 1. Equalizer Weights

$$MSE = \begin{cases} 0.9184, & \frac{E_b}{N_o} = 1 \\ 0.8505, & \frac{E_b}{N_o} = 2 \\ 0.5468, & \frac{E_b}{N_o} = 10 \end{cases} \tag{26}$$

4. The Gradient (LMS) Algorithm

The LMS or Gradient algorithm is used to implement *adaptive* equalization. It is a *stochastic gradient optimization algorithm* based on a traditional optimization technique called the *Method of Steepest Descent*. This algorithm takes into advantage the following facts :

- The Mean square error (MSE) surface when plotted against the filter coeffcients is a *quadratic*, bowl-shaped one with an *unique minimum*.
- The Gradient of a function always points "uphill", i.e, towards the maximum of the function. (Conversely, the negative of the gradient is a vector quantity always pointing towards the minimum).

4.1. The Method of Steepest Descent.

In the Steepest Descent Optimization method, the weight vector is made to "evolve" in the direction of the negative gradient of the MSE :

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \frac{\Delta}{2}\left[(-\nabla(E\{\epsilon^2[n]\})\right] \tag{27}$$

Where n reprsents the iteration number, $\epsilon[n]$ is the error between the actual and the desired outputs in the n^{th} iteration and Δ is the step-size. The main disadvantage of this method is the calculation of the actual gradient, which involves *ensemble* averages not readily available in real time.

4.2. The LMS Simplification.

The LMS algorithm is bascially a simplification of the Method of Steepest Descent, where *instantaneous* values are used instead of actual (ensemble averaged) values :

$$\mathbf{w}[n+1] = \mathbf{w} + \Delta\epsilon[n] * \mathbf{Y}_i[n] \tag{28}$$

Where $\mathbf{Y}_i[n] = [Y((n-M)T), \ldots, Y(0), \ldots, Y((n+M)T)]^T$ represents the tap-inputs at the time instant (or iteration) n. The error $\epsilon[n]$ is computed from the equalized output using either a training sequence or the decoded output as reference.

$$\epsilon[n] = A_n - y_{eq}[n] \tag{29}$$

Where $y_{eq}[n]$ is the equalized output.

4.3. The Algorithm.

Using the Expressions for the error (29) and the output in the above equation (28), we get steps the LMS algorithm as :

$$y_{eq}[n] = \mathbf{w}[n]^H \mathbf{Y}_i[n] \tag{30}$$

$$\epsilon[n] = A_n - y_{eq}[n] \tag{31}$$

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \Delta\epsilon^*[n]\mathbf{Y}_i[n] \tag{32}$$

The block diagram for implementing this algorithm is shown in fig(7).

4.4. Stability.

It has been shown that starting with an *arbitrary* initial weight vector, the LMS algorithm will converge and stay stable as long as the value of Δ is chosen as per the following rule:

$$0 < \Delta < \frac{1}{\lambda_{max}} \tag{33}$$

Where λ_{max} is the maximum eigenvalue (or *trace*) of the Covariance Matrix, $\mathbf{\Gamma}$.

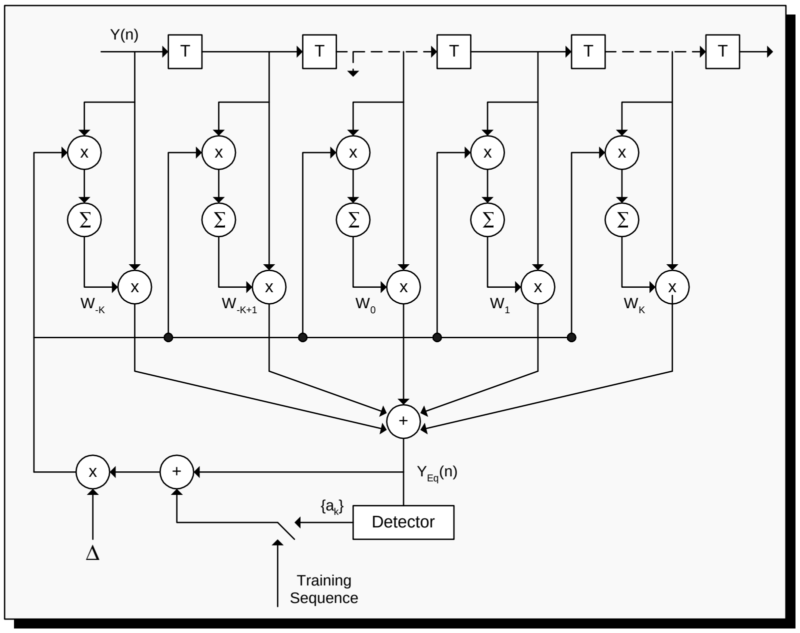

FIGURE 7. LMS Adaptive Equalizer

4.5. **Convergence.** Within the margin given by equation (33), the larger the value of Δ, the faster the convergence but the lesser the stability around the minimum value. On the other hand, the smaller the value of Δ, the slower the convergence but the algorithm will be more stable around the optimum value. An good choice for Δ in the case in which the SNR is known is :

$$\Delta = \frac{0.2}{(2M+1)(S+N)} \tag{34}$$

4.6. **Results.** The Gradient algorithm was setup as shown in the figure (7) and the following results were obtained. The number of samples simulated were 5120 samples and the MSE was calculated for the last 2560 samples. (If only 512 samples were used, the results obtained were not accurate, i.e, either the algorithm did not converge or the residual MSE was too large for the result to be resonably accurate). The value of Δ was set based on equation (34).

4.6.1. *Final Weight Vectors.* The final equalizer weight vectors are tabulated in Table 2. It can

m	$w_m(E_b/N_o = \infty)$	$w_m(E_b/N_o = 10)$	$w_m(E_b/N_o = 2)$	$w_m(E_b/N_o = 1)$
-3	-0.0035	0.0424	0.0138	0.0189
-2	0.0260	-0.0409	-0.0364	-0.0155
-1	-0.1939	0.0148	0.0658	-0.0314
0	1.2736	0.5279	0.1741	0.1010
1	-0.2200	0.0163	0.0123	-0.0115
2	0.1865	0.0310	-0.0202	0.0134
3	-0.0991	-0.0163	0.0152	-0.0025

TABLE 2. Equalizer Weights

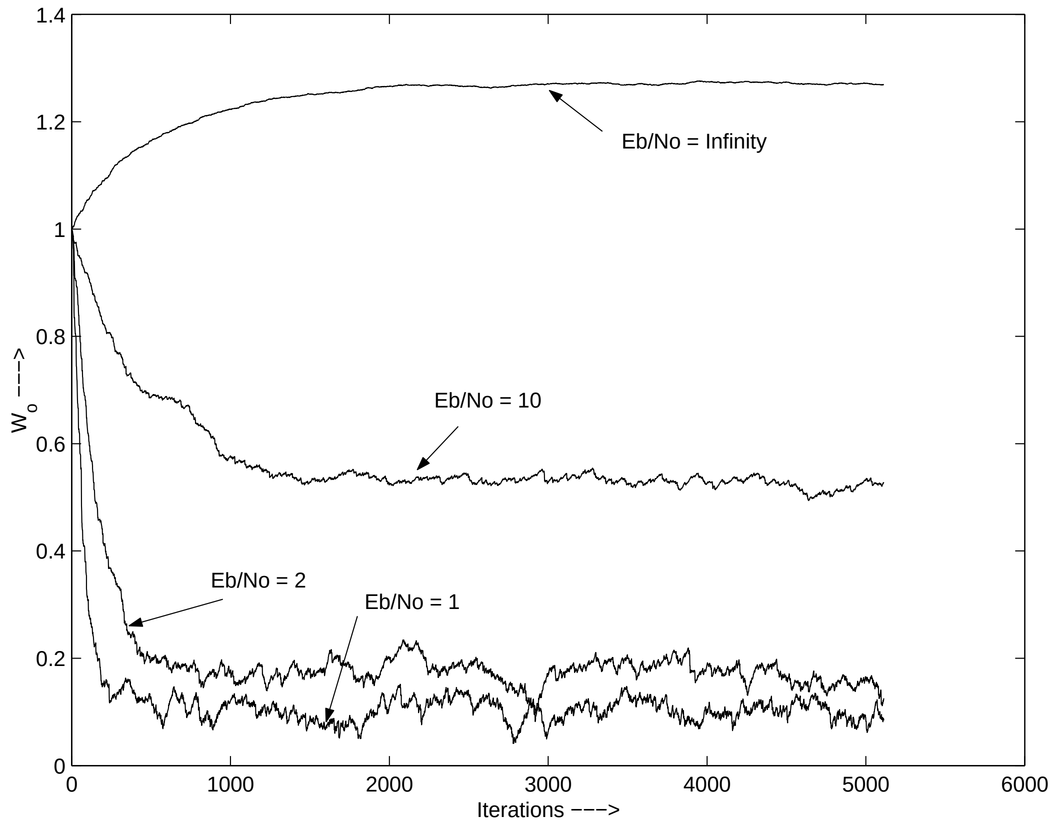

FIGURE 8. Weight Vector Convergence

be noted that the equalizer weights after the gradient algorithm has converged are similar to the weights computed for the LMMSE equalizer.

4.6.2. *MSE.* The MSE was computed for the last 2560 samples as:

$$MSE = \begin{cases} 0.9441, & \frac{E_b}{N_o} = 1 \\ 0.8620, & \frac{E_b}{N_o} = 2 \\ 0.5518, & \frac{E_b}{N_o} = 10 \end{cases} \tag{35}$$

The MSE after the gradient algorithm has converged is approximately the same as the MSE found for the LMMSE equalizer.

4.6.3. *Weight Vector Convergence.* The figure (8) illustrates the weight vector convergence.

5. Appendix - MATLAB Code

5.1. Channel Sounding.

```
function [Pr] = ChannelSound(nChannel, dChannel, nPerSymbol, Instants)

padRange = max(abs(Instants));

% Take an extra hundred to account for flter delays.
nPad = padRange*nPerSymbol + 50;

% Generate a Test Pulse.
PulseIn = [zeros(1,nPad), ones(1,nPerSymbol), zeros(1,nPad)];

% Pass the test pulse thorough the channel.
PulseOut = filter(nChannel,dChannel,PulseIn);

% Obtain the Pulse Response at the sampling instants.
[Pr_0,t_0] = max(PulseOut);
t_Index = t_0 + (Instants)*nPerSymbol;
Pr = PulseOut(t_Index);
```

5.2. Estimating Channel Delay.

```
function [delay] = EvaluateChannelDelay(nChannel, dChannel, nPerSymbol)

nPad = 100; % To account for filter delays.

% Generate a Test Pulse.
PulseIn = [zeros(1,nPad), ones(1,nPerSymbol), zeros(1,nPad)];

% Pass the test pulse thorough the channel.
PulseOut = filter(nChannel,dChannel,PulseIn);

% Obtain the Pulse Response at the sampling instants.
[Pr_0,t_0] = max(PulseOut);

delay = t_0 - (nPad+1) + 1;
```

5.3. Zero-Forcing Equalizer Design.

```
function [EqW] = ZeroForcingEqualizer(M, nChannel, dChannel,nPerSymbol)

Range = -2*M:2*M;

% Perform Channel Sounding.
PulseResp = ChannelSound(nChannel,dChannel,nPerSymbol,Range);

% Form the Equalizer Simulataneous equations..
% .. and Solve them.
prMatrix = zeros(2*M+1,2*M+1);
for index = 0:2*M,
   prMatrix(index+1,:) = PulseResp(2*M+1-index:2*M+1-index+2*M);
end
prMatrix = prMatrix';
Y = [zeros(1,M), 1, zeros(1,M)]';
EqW = inv(prMatrix)*Y
```

5.4. **LMMSE Equalizer Design.**

```
function [EqW,ResidualMSE] = LMMSEFilter(M, nChannel,dChannel,EbNo,nPerSymbol)

NSR = 1/(2*EbNo);

sampleRange = 6; % in terms of M.

Range = -sampleRange*M:sampleRange*M;

Pr = ChannelSound(nChannel,dChannel,nPerSymbol,Range);

PrCorr = xcorr(Pr,Pr,2*M);
PrCorr = PrCorr(7:end);
prMatrix = toeplitz(PrCorr);
prMatrix = prMatrix + NSR*eye(2*M+1,2*M+1);

mid = sampleRange*M+1;
Y = fliplr(Pr(mid+(-M:M)))';
EqW = inv(prMatrix)*Y;

Sav = nPerSymbol;
ResidualMSE = Sav*(1-Y'*inv(prMatrix)*Y);

EqW = inv(prMatrix)*Y;
```

5.5. **Eye Diagram Function.**

```
function EyeDiagram(RxSignal, nSymbolsInEye, nSymbols, nSamplesPerSymbol, nFigure)

nIters = floor(length(RxSignal)/(nSymbolsInEye*nSamplesPerSymbol));
figure(nFigure);
hold on;
for index = 1:nIters,
   plot(1:nSamplesPerSymbol*nSymbolsInEye, RxSignal((index-1)*nSymbolsInEye ...
                     *nSamplesPerSymbol+1:index*nSymbolsInEye*nSamplesPerSymbol));
end
hold off;
xlabel('Time --->');
ylabel('Amplitude --->');
grid on;
```

5.6. **LMS Algorithm Implementation.**

```
clear;
nSymbols = 2000;
```

```
nPerSymbol = 16;
M = 3;
EbNo = 10;

NoiseVar = 8/EbNo;

Mu = 0.003;
%Mu =0.003; ebno = 2
%Mu = 0.001;%0 ebno = 1

Symbols = rand(1,nSymbols) > 0.5;
Symbols = 2*Symbols - 1;
Waveform = zeros(1,nSymbols*nPerSymbol);
Waveform(1:nPerSymbol:end) = Symbols;
WaveIn = filter(ones(1,nPerSymbol),1,Waveform);
[nFilter, dFilter] = butter(4,0.4/(nPerSymbol/2));
WaveOut = filter(nFilter,dFilter,WaveIn);
WaveOut = WaveOut + sqrt(NoiseVar)*randn(1,length(WaveOut));
[delay] = EvaluateChannelDelay(nFilter,dFilter,nPerSymbol);
Y_In = WaveOut(delay:nPerSymbol:end);
nS =length(Y_In);
nIterations = length((M+1):nS-(M+1));

W = zeros(nIterations,2*M+1);

W(1,:) = [zeros(1,M), 1, zeros(1,M)];
for index = (M+1):nS-(M+1),
   X = Y_In(index-M:index+M);
   Y_Out(index) = W(index-(M+1)+1,:)*X';
   Error(index-(M+1)+1) = Symbols(index) - Y_Out(index);
   W(index+1-(M+1)+1,:) = W(index-(M+1)+1,:) + Mu*Error(index-(M+1)+1)*X;
end

figure(1);
plot(abs(Error));
figure(2);
plot(W(:,M+1));
```

References

[1] John.G.Proakis, “Digital Communications”,*McGraw-Hill Series in Electrical Engineering and Computer Science*, 3^{rd} Edition.