PERFORMANCE EVALUATION OF ADAPTIVE EQUALIZER IN A COMMUNICATION SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electrical Engineering

By

DEBASHREE MOHAPATRA & DURGESH NANDINI MOHANTY

Department of Electrical Engineering

National Institute of Technology

Rourkela-769008, Orissa

PERFORMANCE EVALUATION OF ADAPTIVE EQUALIZER IN A COMMUNICATION SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electrical Engineering

By

DEBASHREE MOHAPATRA & DURGESH NANDINI MOHANTY

Under the Guidance of

Dr. Susmita Das

Department of Electrical Engineering

National Institute of Technology

Rourkela-769008, Orissa

CERTIFICATE

This is to certify that the project entitled **“PERFORMANCE EVALUATION OF ADAPTIVE EQUALIZER IN A COMMUNICATION SYSTEM”** is a bona fide record work done by **Debashree Mohapatra** and **Durgesh Nandini Mohanty** during the partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology.

Date:

Rourkela

Dr.Susmita Das

Dept. of Electrical Engg

National Institute of Technology

Rourkela-769008

ACKNOWLEDGEMENT

We deem it a privilege to have been the students of Electrical Engineering in National Institute of Technology, Rourkela.

Our heartfelt regards to Dr. Susmita Das, our project guide who helped us bring out this project in a fruitful manner with her precious suggestions and rich experience.

We are grateful to Prof. Bidyadhar Subudhi, Head of the Department of Electrical Engineering for providing necessary facilities in the department.

Debashree Mohapatra	Durgesh Nandini Mohanty
10502018	10502061

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project deals with the study of the various kinds of interferences in a communication channel viz. Inter symbol Interference, Multipath Interference and Additive Interference. It deals with the design of an Adaptive Equalizer. The idea of the equalizer is to build (another) filter in the receiver that counteracts the effect of the channel. In essence, the equalizer must “unscatter” the impulse response. This can be stated as the goal of designing the equalizer E so that the impulse response of the combined channel and equalizer CE has a single spike. This can be solved using different techniques.

In this project, we have implemented an ‘Adaptive Equalizer’ using four different algorithms in Matlab. We have suggested different ways to decide the coefficients of the equalizer. The first procedure (LEAST SQUARE ALGORITHM) minimizes the square of the symbol recovery error over a block of data which can be done by using matrix pseudo inversion. The second method (LEAST MEAN SQUARE ALGORITHM) involves minimizing the square of the error between the received data values and the transmitted values which are achieved via an adaptive element. The third method (DECISION DIRECTED ALGORITHM) and the fourth method (DISPERSION MINIMIZING ALGORITHM) are used when there is no training sequence and other performance functions are appropriate.

In addition to this we have undertaken a study and realization of the Bit Error Rate of a communication system using VisSim Software.

Chapter 1

INTRODUCTION

When all is well in the receiver, there is no interaction between successive symbols; each symbol arrives and is decoded independently of all others. But when symbols interact, when the waveform of one symbol corrupts the value of a nearby symbol, then the received signal becomes distorted. It is difficult to decipher the message from such a received signal. This impairment is called “inter-symbol interference”.

When there is no inter-symbol interference (from a multipath channel, from imperfect pulse shaping, or from imperfect timing), the impulse response of the system from the source to the recovered message has a single nonzero term. The amplitude of this single “spike” depends on the transmission losses, and the delay is determined by the transmission time. When there is inter-symbol interference caused by the channel, this single spike is “scattered”, duplicated once for each path in the channel. The number of nonzero terms in the impulse response increases. The channel can be modeled as a finite-impulse-response, linear filter C, and the delay spread is the total time interval during which reflections with significant energy arrive. The idea of the equalizer is to build (another) filter in the receiver that counteracts the effect of the channel. In essence, the equalizer must “unscatter” the impulse response. This can be stated as the goal of designing the equalizer E so that the impulse response of the combined channel and equalizer CE has a single spike. This can be cast as an optimization problem, and can be solved using different techniques.

The transmission path may also be corrupted by additive interferences such as those caused by other users. These noise components are usually presumed to be uncorrelated with the source sequence and they may be broadband or narrowband, in-band or out-of-band relative to the band limited spectrum of the source signal. Like the multipath channel interference, they cannot be known to the system designer in advance. The second job of the equalizer is to reject such additive narrow band interferers by designing appropriate linear notch filters 'on-the-fly' based on the received signal. At the same time, it is important that the equalizer not unduly enhance the noise.

In this project initially we explain the basic concept of an equalizer, an adaptive equalizer and the structure of an adaptive equalizer that is an FIR filter. Next we describe the four algorithms of Adaptive Equalizer that we have implemented in Matlab. Finally we give an insight into the realization and study of the Bit Error Rate of Communication System using VisSim Software.

1. IMPLEMENTATION OF LINEAR EQUALIZER IN MATLAB

1.1 EQUALIZER

An equalization (EQ) filter is a filter, usually adjustable, chiefly meant to compensate for the unequal frequency response of some other signal processing circuit or system.

An EQ filter typically allows the user to adjust one or more parameters that determine the overall shape of the filter's transfer function. It is generally used to improve the fidelity of sound, to emphasize certain instruments, to remove undesired noises, or to create completely new and different sounds.

The design objective of the equalizer is to undo the effects of the channel and to remove the interference. Conceptually the equalizer attempts to build a system that is a “delayed inverse” of the digital model of the transmission channel, removing the inter symbol interference while simultaneously rejecting the additive interferers uncorrelated to the source.

1.2 ADAPTIVE EQUALIZER

An adaptive equalizer is an equalization filter that automatically adapts to time-varying properties of the communication channel. It can be implemented to perform tap-weight adjustments periodically or continually. Periodic adjustments are accomplished by periodically transmitting a preamble or short training sequence of digital data known by the receiver. Continual adjustment are accomplished by replacing the known training sequence with a sequence of data symbols estimated from the equalizer output and treated as known data. When performed continually and automatically in this way, the adaptive procedure is referred to as decision directed.

If the probability of error exceeds one percent, the decision directed equalizer might not converge. A common solution to this problem is to initialize the equalizer with an alternate process, such as a preamble to provide good channel-error performance, and then switch to decision-directed mode.

1.3 STRUCTURE OF AN EQUALIZER-FIR FILTER

A finite impulse response (FIR) filter is a type of a digital filter. The impulse response is finite because it settles to zero in a finite number of sample intervals. This is in contrast to infinite impulse response (IIR) filters, which have internal feedback and may continue to respond indefinitely. The impulse response of an Nth-order FIR filter lasts for N+1 samples, and then dies to zero.

We start the discussion by stating the difference equation which defines how the input signal is related to the output signal:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \tag{1.1}$$

where:

x[n] is the input signal,

y[n] is the output signal, and

bi are the filter coefficients.

N is known as the filter order; an Nth-order filter has (N + 1) terms on the right-hand side; these are commonly referred to as taps.

The previous equation can also be expressed as a convolution of filter coefficients and the input signal.

$$y[n] = \sum_{i=0}^{N} b_i x[n-i].$$

In general, an accurate digital model for a channel depends on many things: the underlying analog channel, the pulse shaping used, and the timing of the sampling process. At first glance, this seems like it might make designing an equalizer for such a channel almost impossible. But there is good news. No matter what timing instants are chosen, no matter what pulse shape is used, and no matter what the underlying analog channel may be (as long as it is linear), there is a FIR linear representation of the form (1.1) that closely models its behavior.

Advantages of FIR Filters (compared to IIR filters):

Compared to IIR filters, FIR filters offer the following advantages:

1. They can easily be designed to be "linear phase" (and usually are). Put simply, linear-phase filters delay the input signal, but don't distort its phase.

2. They are simple to implement. On most DSP microprocessors, the FIR calculation can be done by looping a single instruction.

3. They are suited to multi-rate applications. By multi-rate, we mean either "decimation" (reducing the sampling rate), "interpolation" (increasing the sampling rate), or both. Whether decimating or interpolating, the use of FIR filters allows some of the calculations to be omitted, thus providing an important computational efficiency. In contrast, if IIR filters are used, each output must be individually calculated, even if it that output will discarded (so the feedback will

be incorporated into the filter).

4. They have desirable numeric properties. In practice, all DSP filters must be implemented using "finite-precision" arithmetic, that is, a limited number of bits. The use of finite-precision arithmetic in IIR filters can cause significant problems due to the use of feedback, but FIR filters have no feedback, so they can usually be implemented using fewer bits, and the designer has fewer practical problems to solve related to non-ideal arithmetic.

5. They can be implemented using fractional arithmetic. Unlike IIR filters, it is always possible to implement a FIR filter using coefficients with magnitude of less than 1.0. (The overall gain of the FIR filter can be adjusted at its output, if desired.) This is an important consideration when using fixed-point DSP's, because it makes the implementation much simpler.

Disadvantages of FIR Filters (compared to IIR filters):

Compared to IIR filters, FIR filters sometimes have the disadvantage that they require more memory and/or calculation to achieve a given filter response characteristic. Also, certain responses are not practical to implement with FIR filters.

1.4 BASIC MODEL OF A COMMUNICATION SYSTEM

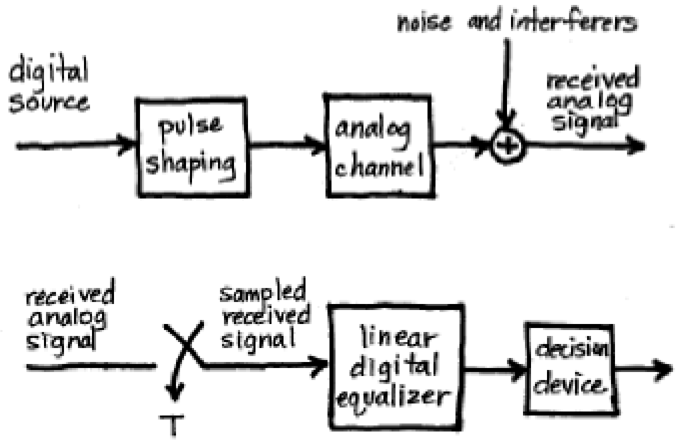


Fig 1.1 Signal Path of a baseband digital communication system

The signal path of a baseband digital communication system is shown in fig 1.1, which emphasizes the role of the equalizer in trying to counteract the effects of the multipath channel and the additive interference.

In the figure above, a digital input is passed thrrough an analog channel after pulse shaping. Pulse shaping is the process of changing the waveform of transmitted pulses. Its purpose is to make the transmitted signal suit better to the communication channel by limiting the effective bandwidth of the transmission. By filtering the transmitted pulses this way, the intersymbol interference caused by the channel can be kept in control. Then after passing through the analog channel noise and interferers are added to the signal in order to account for the actual noise and interferences which a signal encounters while passing through a channel.

The analog signal is sampled and then it is passed through a linear digital equalizer. The baseband linear (digital) equalizer is intended to (automatically) cancel unwanted effects of the channel and to cancel certain kinds of additive interferences.

All of the inner parts of the system are assumed to operate precisely. Thus the up and down conversion, the timing recovery and the carrier synchronization are assumed to be flawless and unchanging. Modeling the channel as a time invariant FIR filter, the next section focuses on the task of selecting the coefficients in the block labeled “linear digital equalizer”, with the goal of removing the inter symbol interference and attenuating the additive interferences. These coefficients are to be chosen based on the sampled received signal sequence and (possibly) knowledge of a prearranged “training sequence”. While the channel may actually be time-varying, the variations are often much slower than the data rate, and the channel can be viewed as (effectively) time-invariant over small time scales.

1.5 MULTIPATH INTERFERENCE

The villains of communication are multipath and other additive interferers. The distortion caused by an analog wireless channel can be thought of as a combination of scaled and delayed reflections of the original transmitted signal. These reflections occur when there are different paths from the transmitting antenna to the receiving antenna. The strength of the reflections depends on the physical properties of the reflecting objects, while the delay of the reflections is primarily determined by the length of the transmission path.

Let u(t) be the transmitted signal. If N delays are represented by Δ_1, Δ_2, Δ_3, $\Delta_4 \ldots \Delta_N$ and the strength of the reflections is α_1, α_2, α_3, $\alpha_4, \ldots \alpha_N$ then the received signal y(t) is

$$y(t) = \alpha_1 u(t-\Delta_1) + \alpha_2 u(t-\Delta_2) + \ldots + \alpha_N u(t-\Delta_N) + \eta(t) \qquad (1.2)$$

Where $\eta(t)$ represents additive interferences. This model of the transmission channel has the form of a finite impulse response filter, and the total length of time $\Delta_N - \Delta_1$ over which the impulse response is nonzero is called the delay spread of the physical medium.

This transmission channel is typically modeled digitally assuming a fixed sampling period T_s.Thus (1.2) is approximated by

$$y(kT_s)=\alpha_1 u(kT_s)+\alpha_2 u((k-1)T_s)+\ldots\alpha_n u((k-n)T_s)+\eta(kT_s) \qquad (1.3)$$

In order for the model (1.3) to closely represent the system (1.2), the total time over which the impulse response is nonzero(the time ηT_s) must be at least as large as the maximum delay Δ_N .Since the delay is not a function of the symbol period T_s, smaller T_s require more terms in the filter, i.e., larger n.

For example, consider a sampling interval of T_s = 40 nanoseconds (i.e. a transmission rate of 25 MHz). A delay spread of approximately 4 microseconds would correspond to one hundred taps in the model (1.3). Thus at ant time instant, the received signal would be a combination of (up to) one hundred data values. If T_s were increased to 0.4 microseconds (i.e 2.5 MHz), only ten terms would be needed, and there would only be interference with the ten nearest data values. If $T_s$were larger than 4 microseconds (i.e. 0.25 MHz),only one term would be needed in the discrete-time impulse response. In this case, adjacent sampled symbols would not interfere. Such finite duration response models can also be used to represent the frequency-selective dynamics that occur in the wired local end –loop in telephony, and other (approximately) linear, finite-delay-spread channels.

The design objective of the equalizer is to undo the effects of the channel and to remove the interference. Conceptually, the equalizer attempts to build a system that is a “delayed inverse” of (1.3), removing the inter symbol interference while simultaneously rejecting additive interferers uncorrelated to the source. If the interference $\eta(kT_s)$ is unstructured (for instance white noise) then there is little that a linear equalizer can do to improve it. But when the interference is highly structured (such as narrow band interference from another user) then the linear filter can often notch out the offending frequencies.

Chapter 2

2. ALGORITHMS FOR THE IMPLEMENTATION OF ADAPTIVE EQUALIZER IN MATLAB

2.1 TRAINED LEAST SQUARES LINEAR EQUALIZATION

When there is a timing sequence available (for instance, in the known frame information that is used in synchronization), then it can also be used to help build or 'train' an Equalizer. The basic strategy is to find a suitable function of the unknown equalizer parameters that can be used to define an optimization problem. Then applying various techniques the optimization can be solved in a variety of ways.

2.1.1 A MATRIX DESCRIPTION

The linear optimization problem is depicted in the figure. A prearranged training sequence s[k] is assumed known at the receiver. The goal is to find an FIR filter (called equalizer) so that the output of the equalizer is approximately equal to the known source, though possibly delayed in time. Thus the goal is to choose the impulse response f_i such that y[k] =s[k-δ] for some specific delay δ .

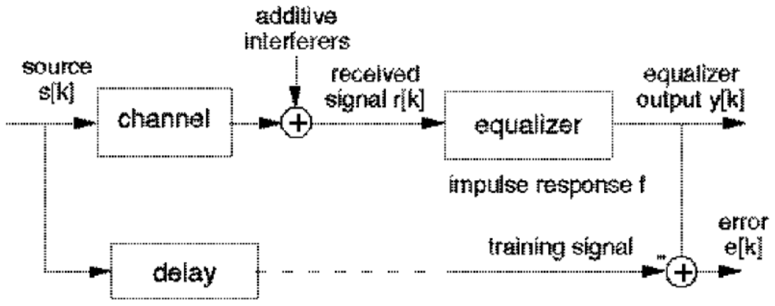


Fig 2.1 The problem of the linear equalization is to find a linear system f that undoes the effects of the channel while minimizing the effects of the interferences.

The input-output behavior of the FIR linear equalizer can be described as the convolution

$$y[k] = \sum_{j=0}^{n} f_j r[k-j] \qquad (1.4)$$

Where the lower index j can be no lower than zero (or else the equalizer is non-causal i.e. it can illogically respond to an input before the input is applied).

This convolution is illustrated in the figure as Direct Form FIR or Tapped Delay Line.

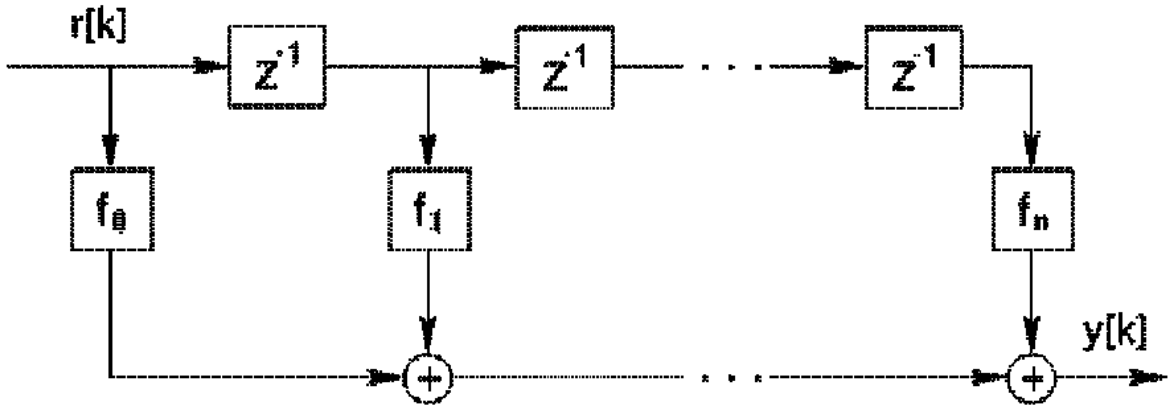


Fig 2.2 Direct Form FIR as Tapped Delay Line

The summation can also be written e.g. for k=n+1, as the inner product of two vectors

$$y[n+1] = [r[n+1],\ r[n], \dots,\ r[1]] \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \qquad (1.5)$$

Note that y[k+1] is the earliest output that can be formed given no knowledge of r[i] for i<1 . Incrementing the time index in the above equation gives

$$y[n+2] = [r[n+2],\ r[n+1], \ldots,\ r[2]] \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \qquad (1.6)$$

And

$$y[n+3] = [r[n+3],\ r[n+2], \ldots,\ r[3]] \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \qquad (1.7)$$

Observe that each of these uses the same equalization parameter vector. Concatenating p-n of these measurements into one matrix equation over the available data set for i=1 to p gives:

$$\begin{bmatrix} y[n+1] \\ y[n+2] \\ y[n+3] \\ \vdots \\ y[p] \end{bmatrix} = \begin{bmatrix} r[n+1] & r[n] & \ldots & r[1] \\ r[n+2] & r[n+1] & \ldots & r[2] \\ r[n+3] & r[n+2] & \ldots & r[3] \\ \vdots & \vdots & & \vdots \\ r[p] & r[p-1] & \ldots & r[p-n] \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \qquad (1.8)$$

Or with approximate matrix definitions

$$Y=RF \qquad (1.9)$$

R has a special structure that the entries along each diagonal are the same. R is known as a Toeplitz matrix and the Toeplitz command in Matlab makes it easy the build matrices with this structure.

2.1.2 SOURCE RECOVERY ERROR

The delayed source recovery error is

$$e[k] = s[k-\delta] - y[k] \tag{1.10}$$

for a particular δ. This section shows how the source recovery error can be used to define a performance function that depends on the unknown parameters f_i . Calculating the parameters that minimize this performance function provides a good solution for the equalization problem.

We define

$$S = \begin{bmatrix} s[n+1-\delta] \\ s[n+2-\delta] \\ s[n+3-\delta] \\ \vdots \\ s[p-\delta] \end{bmatrix} \tag{1.11}$$

And

$$E = \begin{bmatrix} e[n+1] \\ e[n+2] \\ e[n+3] \\ \vdots \\ e[p] \end{bmatrix} \tag{1.12}$$

So we can now write

$$E= S-Y=S-RF \tag{1.13}$$

As a measure of the performance of the f_i and F, consider

$$J_{LS} = \sum_{i=n+1}^{p} e^2[i]. \tag{1.14}$$

J_{LS} is non negative since it is the sum of squares. Minimizing such a summed, squared delayed source recovery error is common objective in equalizer design, since that f_i that minimize J_{LS} cause the output of the equalizer to become close the values of the delayed source. J_{LS} can be written as

$$\begin{aligned} J_{LS} &= E^T E = (S - RF)^T (S - RF) \\ &= S^T S - (RF)^T S - S^T RF + (RF)^T RF. \end{aligned} \tag{1.15}$$

Because J_{LS} is a scalar, $(RF)^T S$ and $S^T RF$ are also scalars. Since the transpose of a scalar is equal to itself $(RF)^T S = S^T RF$ and the above equation can be written as

$$J_{LS} = S^T S - 2S^T RF + (RF)^T RF. \tag{1.16}$$

The issue is now one of choosing the n+1 entries of F to make J_{LS} as small as possible.

2.1.3 THE LEAST SQUARES SOLUTION

Define the matrix

$$\begin{aligned} \Psi &= [F - (R^T R)^{-1} R^T S]^T (R^T R) [F - (R^T R)^{-1} R^T S] \\ &= F^T (R^T R) F - S^T RF - F^T R^T S + S^T R (R^T R)^{-1} R^T S. \end{aligned} \tag{1.17}$$

The purpose of the above definition is to rewrite in terms of ψ

$$\begin{aligned} J_{LS} &= \Psi + S^T S - S^T R(R^T R)^{-1} R^T S \\ &= \Psi + S^T [I - R(R^T R)^{-1} R^T] S. \end{aligned} \quad (1.18)$$

Since $S^T[I - R(R^T R)^{-1} R^T]S$ is not a function of F, the minimum of J_{LS} occurs at the F that minimizes the value of ψ. This occurs when

$$F^\dagger = (R^T R)^{-1} R^T S \quad (1.19)$$

assuming that $(R^T R)^{-1}$ exists.

The corresponding minimum achievable by J_{LS} at $F = F^\dagger$ is the summed squared delayed source recovery error. This is the remaining term in the equation, that is,

$$J_{LS}^{\min} = S^T [I - R(R^T R)^{-1} R^T] S. \quad (1.20)$$

The formulae for the optimum F and the associated minimum J_{LS} are specific for δ. To complete the task is to find the desired delay δ. The most straightforward approach is to set up a series of S=RF, one for each possible δ to compute the associated values of $J_{LS}^{\min}$ and pick the delay associated with the smallest one.

The program is straightforward to design in Matlab and is written below.

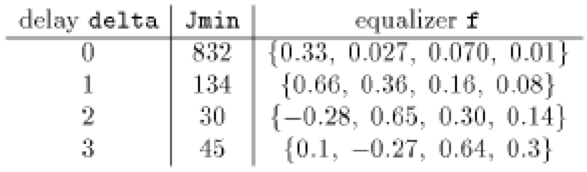
```
%ls equalizer
b=[0.5 1 -.6];                          %define channel
m=1000; s=sign(randn(1,m));             % binary source of length m
r=filter(b,1,s);                        % output of channel
n=3;                                    %length of equalizer -1
```

```
delta=3;                                        % use delay <=n
p=length(r)-delta;
R=toeplitz(r(n+1:p),r(n+1:-1:1));               % build matrix R
S=s(n+1-delta:p-delta)';                        % and vector S
f=inv(R'*R)*R'*S                                %calculate equalizer f
Jmin=S'*S-S'*R*inv(R'*R)*R'*S                   % Jmin for this f and delta
y=filter(f,1,r);                                % equalizer is a filter
dec=sign(y);                                    %quantize and find errors
err=.5*sum(abs(dec(delta+1:end)-s(1:end-delta)))
% output
delta,Jmin,f,err
```

```
f =    0.1013   -0.2635    0.6408    0.3048
Jmin =   46.9299
err =  0
delta = 3
```

The first three lines define a channel, create a binary source, and then transmit the source through the channel using the filter command. At the receiver, the data is put through a quantizer, and then the error is calculated for a range of delays. The new part is in the middle. The variable n defines the length of the equalizer, and delta defines the delay that will be used in constructing the vector S as defined before.(delta must be positive and less than or equal to n). The toeplitz matrix R is defined and the equalizer coefficients f are computed. The value of minimum achievable performance is Jmin. To demonstrate the effect of the equalizer, the received signal r is filtered by the equalizer coefficients, and the output is then quantized. If the equalizer has done its job (i.e. if the eye is open), then there should be some shift ‘sh’ at which no errors occur.

For example, using the default channel b=[0.5 1 -.6], and length 4 equalizer (n=3), four values of the delay delta give



delay **delta**	**Jmin**	equalizer **f**
0	832	$\{0.33,\ 0.027,\ 0.070,\ 0.01\}$
1	134	$\{0.66,\ 0.36,\ 0.16,\ 0.08\}$
2	30	$\{-0.28,\ 0.65,\ 0.30,\ 0.14\}$
3	45	$\{0.1,\ -0.27,\ 0.64,\ 0.3\}$

The best equalizer is the one corresponding to a delay of 2, since this $J_{\min}$ is the smallest. In this case, however, any of the last three open the eye. It is observed that the number of errors (as reported by err) is zero when the eye is open.

2.1.4 FRACTIONALLY SPACED EQUALIZATION

The preceding development assumed that the sampled input to the equalizer symbol spaced with the sampling interval equal to the symbol interval of T seconds. Thus the unit delay in realizing the tapped-delay line equalizer is T seconds. Sometimes the input to the equalizer is oversampled such that the sample interval is shorter than the symbol interval and the resulting equalizer is said to be fractionally spaced. The same kinds of algorithms and solutions can be used to calculate the coefficients in fractionally spaced equalizers as for T-spaced equalizers. Details of the construction of the matrices corresponding to $\overline{S}$ *and* $\overline{R}$ will necessarily differ due to the structural differences. The more rapid sampling allows greater latitude in the ordering of the blocks in the receiver.

2.2 AN ADAPTIVE APPROACH TO THE TRAINED EQUALIZATION

The block oriented design of the previous section requires substantial computation even when the system delay is known since it requires the calculating the inverse of an (n+1) X (n+1) matrix, when n is the largest delay in the FIR linear equalizer. This section considers using an adaptive element to minimize the average of the squared error.

$$J_{LMS} = \tfrac{1}{2}\,\mathrm{avg}\{e^2[k]\} \tag{1.21}$$

Observe that J_{LMS} is a function of all the equalizer coefficients f_i since

$$e[k] = s[k-\delta] - y[k] = s[k-\delta] - \sum_{j=0}^{n} f_j r[k-j] \tag{1.22}$$

which combines (1.10) with (1.4) and where r[k] is the received signal baseband after sampling. An algorithm for the minimization of J_{LMS} with respect to the i^{th} equalizer coefficient f_i is

$$f_i[k+1] = f_i[k] - \mu \left.\frac{dJ_{LMS}}{df_i}\right|_{f_i = f_i[k]} \tag{1.23}$$

To create an algorithm that can be easily implemented, it is necessary to evaluate this derivative with respect to the parameter of interest. This is

$$\frac{dJ_{LMS}}{df_i} = \frac{d\mathrm{avg}\{\frac{1}{2}e^2[k]\}}{df_i} \approx \mathrm{avg}\{\frac{\frac{1}{2}de^2[k]}{df_i}\} = \mathrm{avg}\{e[k]\frac{de[k]}{df_i}\} \tag{1.24}$$

where the final equality follows from the chain rule. Using the derivative of the source recovery error e[k] with respect to the i^{th} equalizer parameter f_i is

$$\frac{de[k]}{df_i} = \frac{ds[k-\delta]}{df_i} - \sum_{j=0}^{n} \frac{df_j r[k-j]}{df_i} = -r[k-i] \qquad (1.25)$$

since $\frac{ds[k-\delta]}{df_i} = 0$ and $\frac{df_j r[k-j]}{df_i} = 0$ for all $i \neq j$. Substituting the above equations, the update for the adaptive element is

$$f_i[k+1] = f_i[k] + \mu \text{avg}\{e[k]r[k-i]\} \qquad (1.26)$$

Typically, the averaging operation is suppressed since the iteration with small step-size μ itself has a low pass (averaging) behavior. This result is commonly called the Least Mean Squares (LMS) algorithm for direct linear equalizer impulse response coefficient adaptation

$$f_i[k+1] = f_i[k] + \mu e[k]r[k-i] \qquad (1.27)$$

The adaptive equalization scheme is illustrated in the Figure

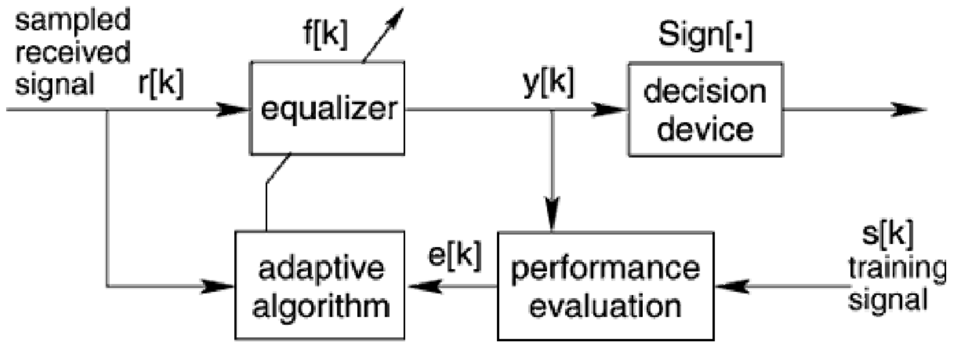


Fig 2.3 Trained Adaptive Linear Equalizer

When all goes well, the recursive algorithm converges to the vicinity of the block least squares answer for the particular δ used in forming the delayed recovery error. As long as μ is nonzero, if the underlying composition of the received signal changes so that the error increases and the desired equalizer changes, then f_i react accordingly. It is this tracking ability that earns it the label adaptive.

The following Matlab code implements an adaptive equalizer design. The opening and the closing of the program are similar to the previous program. The heart of the recursion lies in the for loop. For each new data point, a vector is built containing the new value and the past n values of the received signal. This is multiplied to f to make a prediction of the next source symbol, and the error is the difference between the prediction and the reality (this is the calculation of e[k]). The equalizer coefficients f are then updated as in (1.27).

```
b=[0.5 1 -0.6];                  % define channel
m=1000; s=sign(randn(1,m));      % binary source of length m
r=filter(b,1,s);                 % output of channel
n=4; f=zeros(n,1) ;              % step size and delay delta
mu=.1;delta=2;
for i=n+1:m                      % iterate
  rr=r(i:-1:i-n+1)';             % vector of the received signal
  e=s(i-delta)-f'*rr;            % calculate error
  f=f+mu*e*rr;                   % update equalizer coefficients
end
y=filter(f,1,r);                 % equalizer is a filter
dec= sign(y);                    %quantization
for sh=0:n                       % error at different delays
err(sh+1)=0.5*sum(abs(dec(sh+1:end)-s(1:end-sh)));
end
```

```
% output
rr,e,f
mu =
    0.1000
rr =
    0.9000
    2.1000
    0.1000
   -0.9000
e =
   -0.0953
f =   -0.2621
        0.6604
        0.2564
        0.1543
```

As with the matrix approach, the default b= [0.5 1 -0.6] can be equalized easily with a short equalizer (one with a small n). Observe that the convergent values of the f are very close to the final values of the matrix approach, that is, for a given channel, the value of f given by LMS equalizer is very close to LS equalizer. A design consideration in the adaptive approach to equalization involves selection of the step size. Smaller step sizes μ mean that the trajectory of the estimates is smoother (tends to reject noise better) but it also results in a slower convergence and slower tracking when the underlying solution is time varying. Similarly if the explicit averaging operation is retained, longer averages imply smoother estimates but slower convergence. Similar tradeoffs appear in the block approach in the choice of block size: larger blocks average the noise better but give no detail about changes in the underlying solution within the time span covered by a block.

2.3 DECISION – DIRECTED LINEAR EQUALIZATION

During the training period, the communication system does not transmit any message data. Commonly, a block of training data is followed by a block of message data. The fraction of time devoted to training should be small, but can be up to 20% in practice. If it were possible to adapt the equalizer parameters without using the training data, then the message bearing (and revenue generating) capacity of the channel would be enhanced.

Consider the situation in which some procedure has produced an equalizer setting that opens the eye of the channel. Thus all decisions are perfect, but the equalizer parameters may not yet be at their optimal values. In such a case, the output of the decision device is an exact replica of the delayed source, i.e. it is as good as a training signal. For a binary ± 1 source and decision device that is a sign operator, the delayed source recovery error can be computed as sign {y[k]} –y[k] where y[k] is the equalizer output and sign{y[k]} equals s[k-δ]. Thus, the trained adaptive equalizer of fig 2.3 can be replaced by the decision-directed error as shown in fig 2.4. This converts 1.27 to decision-directed LMS, which has the update

$$f_i[k+1] = f_i[k] + \mu(\text{sign}(y[k]) - y[k])r[k-i] \qquad (1.28)$$

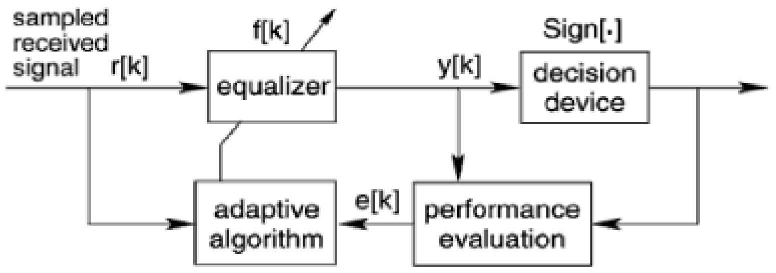


Figure 2.4: Decision – Directed Adaptive Linear Equalizer

It is observed that the source s[k] does not appear in 1.28. Thus, no training signal is required for its implementation and the decision-directed LMS equalizer adaptation law of 1.28 is called a "blind" equalizer. Given its genesis, one should expect decision-directed LMS to exhibit poor behavior when the assumption regarding perfect decisions is violated. The basic rule of thumb is that 5% (or so) decision errors can be tolerated before decision- directed LMS fails to converge properly.

The Matlab program DDequalizer.m has a familiar structure. The only code changed from LMSequalizer.m is the calculation of the error term which implements e[k] = sign{y[k]} – y[k] rather than the LMS error (1.23), and the initialization of the equalizer. Because the equalizer must begin with an open eye, f= 0 is a poor choice. The initialization used below starts all taps at zero except for one in the middle that begins at unity. This is called the "center-spike" initialization. If the channel eye is open, then the combination of the channel and equalizer will also have an open eye when initialized with the center spike.

MATLAB PROGRAM:

```
b=[ 0.5 1 -0.6];                        % define channel
m=1000; s= sign(randn(1,m));            % binary  source of length m
r= filter(b,1,s);                       % output of channel
```

```
n=4; f=[ 0 1 0 0]';                              % initialize equalizer
mu=.1;                                           %stepsize
for i=n+1:m                                      %iterate
  rr= r(i:-1:i-n+1)';                            %vector of received signal
  e=sign(f'*rr)-f'*rr;                           %calculate error
  f=f+mu*e*rr;                                   % update equalizer coefficients
end
y=filter(f,1,r);                                 % equalizer is a filter
dec=sign(y);                                     %quantization
for sh=0:n                                       % error at different delays
err(sh+1)=0.5*sum(abs(dec(sh+1:end)- s (1:end-sh)));
end
%output
f,e,rr
f =
  -0.2953
   0.5901
   0.3085
   0.1427

e =
  -0.0769
rr =
   2.1000
  -1.1000
```

-0.1000

2.1000

2.4 DISPERSION-MINIMISING LINEAR EQUALIZATION

This section considers an alternative performance function that leads to another kind of blind equalizer. Observe that for a binary ±1 source, the square of the source is known, even when the particular values of the source are not. Thus $s^2[k]=1$ for all k. This suggests creating a performance function that penalizes the deviation from this known squared value γ= 1. In particular, consider

$$J_{DMA} = \frac{1}{4} avg\left\{(\gamma - y^2[k])^2\right\} \tag{1.29}$$

which measures the dispersion of the equalizer output about its desired squared value γ.

The associated adaptive element for updating the equalizer coefficients is

$$f_i[k+1] = f_i[k] - \mu \left.\frac{dJ_{DMA}}{df_i}\right|_{f_i = f_i[k]} \tag{1.30}$$

Mimicking the previous equations the Dispersion Minimizing Algorithm (DMA) for blindly adapting the coefficients of a linear equalizer which is

$$f_i[k+1] = f_i[k] + \mu avg\left\{(\gamma - y^2[k])\, y[k]\, r[k-i]\right\} \tag{1.31}$$

Suppressing the averaging operation, this becomes

$$f_i[k+1] = f_i[k] + \mu(1 - y^2[k])\, y[k]\, r[k-i] \tag{1.32}$$

which is shown in the block diagram of fig 1.6.

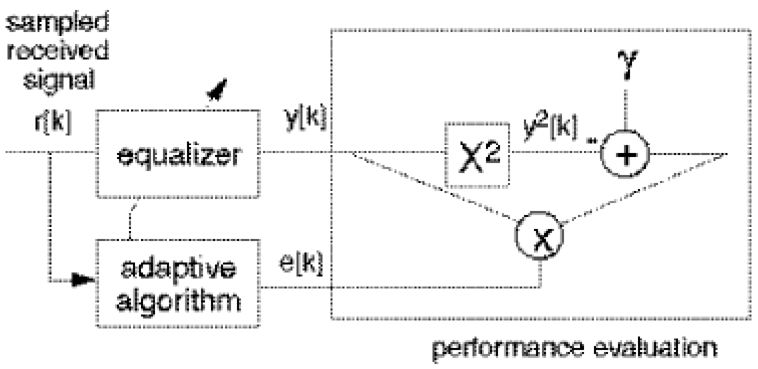


Figure 2.5: Dispersion Minimizing Adaptive Linear Equalizer

When the source alphabet is ±1, then γ=1. When the source is multilevel, it is still useful to minimize the dispersion, but the constant should change to $\gamma = \frac{\text{avg}\{s^4\}}{\text{avg}\{s^2\}}$.

While DMA typically may converge to the desired answer from a worse initialization than decision-directed LMS, it is not as robust as trained LMS. For a particular delay δ, the (average) squared recovery error surface being descended (approximately) along the gradient by trained LMS is unimodal, i.e. it has only one minimum. Therefore, no matter where the search is initialized, it finds the desired sole minimum, associated with the δ used in computing the source recovery error. The dispersion performance function is multimodal with separate minima corresponding to different achieved delays and polarities. To see this in the simplest case, observe that an answer n which all +1's are swapped with -1's has the same value at the optimal point. Thus, the convergent delay and polarity achieved depend on the initialization used. A typical initialization for DMA is a single nonzero spike located near the center of the equalizer.

A simple Matlab program that implements the DMA algorithm is given below in DMAequalizer.m. The first few lines define the channel, create the binary source, and pass the input through the channel. The last few lines implement the equalizer and calculate the error between the output of the equalizer and the source as a way of measuring the performance of the equalizer. These parts of the code are familiar from LSequalizer.m. The new part of the code is in the center, which defines the length n of the equalizer, the stepsize mu of the algorithm, and the initialization of the equalizer (which defaults to a “center spike” initialization). The coefficients of the equalizer are updated as in (1.29).

MATLAB PROGRAM:

```
b=[ 0.5 1 -0.6];                              %define channel
m=1000; s= sign(randn(1,m))                   %binary source of length m
r=filter(b,1,s);                              % output of channel
n=4; f= [0 1 0 0] ’;                          % center spike initialization
mu=.01;                                       % algorithm step size
for i=n+1:m                                   % iterate
   rr=r(i:-1:i-n+1)’;                         % vector of received signal
  e=(f’*rr)*(1-(f’*rr)^2);                    % calculate error
  f=f+mu*e*rr;                                % update equalizer coefficients
end
y=filter(f,1,r);                              % equalizer is a filter
dec=sign(y);                                  % quantization
for sh=0:n                                    % error at different delays
err(sh+1)=0.5*sum(abs(dec(sh+1:end)-s(1:end-sh)));
end
%output
f,e,rr
f =
  -0.2285
```

```
    0.6117

    0.2872

    0.1632

e =

   -0.5975

rr =

   -0.1000

    2.1000

    0.1000

   -0.9000
```

Running DMAequalizer.m results in an equalizer that is numerically similar to the equalizers of the previous two sections. Initializing with the “spike” at different locations results in equalizers with different delays.

Chapter 3

3. REALIZATION AND STUDY OF BIT ERROR RATE OF COMMUNICATION SYSTEM USING VisSim SOFTWARE

3.1 VisSim SOFTWARE

VisSim is a visual block diagram language for modeling, simulating and analyzing dynamic systems. It is developed by Visual Solutions.

VisSim is widely used in control system design and digital signal processing for simulation and design. It includes blocks for arithmetic, Boolean, and transcendental functions, as well as digital filters, transfer functions, numerical integration and interactive plotting.

We have used the following blocks in VisSim for getting the Bit Error Rate Curve control graph.

3.1.1 BIT ERROR RATIO

An error ratio is the ratio of the number of bits, elements, characters, or blocks incorrectly received to the total number of bits, elements, characters, or blocks sent during a specified time interval.

The most commonly encountered ratio is the bit error ratio (BER) - also sometimes referred to as bit error rate.

Bit error ratio is the number of erroneous bits received divided by the total number of bits transmitted; or the number of erroneous decoded (corrected) bits divided by the total number of decoded (corrected) bits.

A transmission might have a BER of 10 to the power minus 6, meaning that, out of 1,000,000 bits transmitted, one bit was in error.

The test time t can be calculated using Gaussian error distribution to:

$$t = -\frac{\ln(1-c)}{b * r}$$

where c is the degree of confidence level

b = upper bound of BER and

r = bit rate.

BER curves are usually plotted to describe the functionality of a digital communication system. In optical communication, BER (dB) vs. Received Power (dBm) is usually used; while in wireless communication, BER (dB) vs. SNR (dB) is used.

The BER is an indication of how often a packet or other data unit has to be retransmitted because of an error. Too high a BER may indicate that a slower data rate would actually improve overall transmission time for a given amount of transmitted data since the BER might be reduced, lowering the number of packets that had to be resent.

3.1.2 BIT ERROR RATE TEST

A BERT (bit error rate test or tester) is a procedure or device that measures the BER for a given transmission.

BERT or Bit Error Rate Test is a testing method for digital communication circuits that uses predetermined stress patterns comprising of a sequence of logical ones and zeros generated by a pseudorandom binary sequence.

A BERT typically consists of a test pattern generator and a receiver that can be set to the same pattern. They can be used in pairs, with one at either end of a transmission link, or singularly at one end with a loopback at the remote end. BERTs are typically stand-alone specialized instruments, but can be Personal Computer based. In use, the number of errors if any are counted and presented as a ratio such as 1 in 1,000,000, or 1 in 10e06.

Measurement of BER: Slam Dunk Approach

The definition of BER is the ratio of number of erroneous bits detected to the number of transmitted bits:

BER=number of erroneous bits ÷number of transmitted bits

Since BER depends on probability this number represents the actual BER of the link only if the number of transmitted bits tends to infinity.

If the link is set-up to transmit and receive a known data pattern it is very easy to measure the number of errors detected over the test duration by just using a simple error detector that compares the transmitted and received data. For example if one error was observed after transmitting 10^{12} bits it is not safe to assume that the BER of the link is $1/10^{12}$. Due to random nature of errors, there is no guarantee that there will be less than or equal to 1 error in the next 10^{12} bits. If there were 3 observed errors in the next 10^{12} bits it would not have been safe to assume that the BER of the link is $4/20^{12}$.

The theoretical way to obtain the accurate BER using this method is to transmit an infin ite number of bits- a practical impossibility. The confidence level of the BER number obtained increases with the measurement time. The only way to make a practical measurement using this method is to run the test for a long duration to guarantee BER with a certain confidence level.

For a standard like a Gigabit Ethernet (Data rate= 1.25 Gbit/s), that specifies a bit error rate of less than $1/10^{12}$ it takes around 13 mins to transmit 10^{12} bits. In one day the number of bits transmitted would be close to 100 times 10^{12} . The BER calculated after this will give a reasonable amount of confidence on the measurement made. On the other hand the OBSAI RP3 requires a BER of less than 1/10^15. Here the time required to transmit 10^15 bits at a data rate of 1.536 Gbit/s is around 7.5 days. The time required to make a measurement with the same amount of confidence as the Gigabit Ethernet case is much greater.

3.1.3 E_b/N_0 RATIO

E_b/N_0 (the energy per bit to noise power spectral density ratio) is an important parameter in digital communication or data transmission. It is a normalized signal-to-noise ratio (SNR) measure, also known as the "SNR per bit". It is especially useful when comparing the bit error rate (BER) performance of different digital modulation schemes without taking bandwidth into account.

E_b/N_0 is equal to the SNR divided by the "gross" link spectral efficiency in (bit/s)/Hz, where the bits in this context are transmitted data bits, inclusive of error correction information and other protocol overhead. It should be noted that when forward error correction is being discussed, E_b/N_0 is routinely used to refer to the energy per information bit (i.e. the energy per bit net of

FEC overhead bits).In this context, E_s/N_0 is generally used to relate actual transmitted power to noise.

The noise spectral density N_0, usually expressed in units of watts per hertz, can also be seen as having dimensions of energy, or units of joules, or joules per cycle. E_b/N_0 is therefore a non-dimensional ratio.

E_b/N_0 is commonly used with modulation and coding designed for noise-limited rather than interference-limited communication, and for power-limited rather than bandwidth-limited communications.[clarification needed] Examples of power-limited communications include deep-space and spread spectrum, and is optimized by using large bandwidths relative to the bit rate.

Relation to carrier-to-noise ratio:

E_b/N_0 is closely related to the carrier-to-noise ratio (CNR or C/N), i.e. the signal-to-noise ratio (SNR) of the received signal, after the receiver filter but before detection:

$$C/N = E_b/N_0 \cdot \frac{f_b}{B}$$

where, f_b is the channel data rate (net bitrate), and

B is the channel bandwidth

The equivalent expression in logarithmic form (dB):

$$CNR_{dB} = 10\log_{10}(E_b/N_0) + 10\log_{10}(\frac{f_b}{B}).$$

Relation to E_s/N_0:

Eb/N0 can be seen as a normalized measure of the energy per symbol per noise power spectral density (Es/N0), where Es is the Energy per symbol in Joules. This measure is also commonly used in the analysis of digital modulation schemes. The two quotients are related to each other according to the following:

$$\frac{E_s}{N_0} = \frac{E_b}{N_0} \log_2 M\,,$$

Where,

M is the number of alternative modulation symbols.

Es/N0 can further be expressed as:

$$\frac{E_s}{N_0} = \frac{C}{N}\frac{B}{f_s},$$

,

Where,

C/N is the carrier-to-noise ratio or signal-to-noise ratio.

B is the channel bandwidth in Hertz.

fs is the symbol rate in baud or symbols/second.

3.1.4 BER CURVE CONTROL USING VisSim

We have used the following blocks to implement BER Curve control in VisSim. The characteristics and parameters of the different blocks are described below.

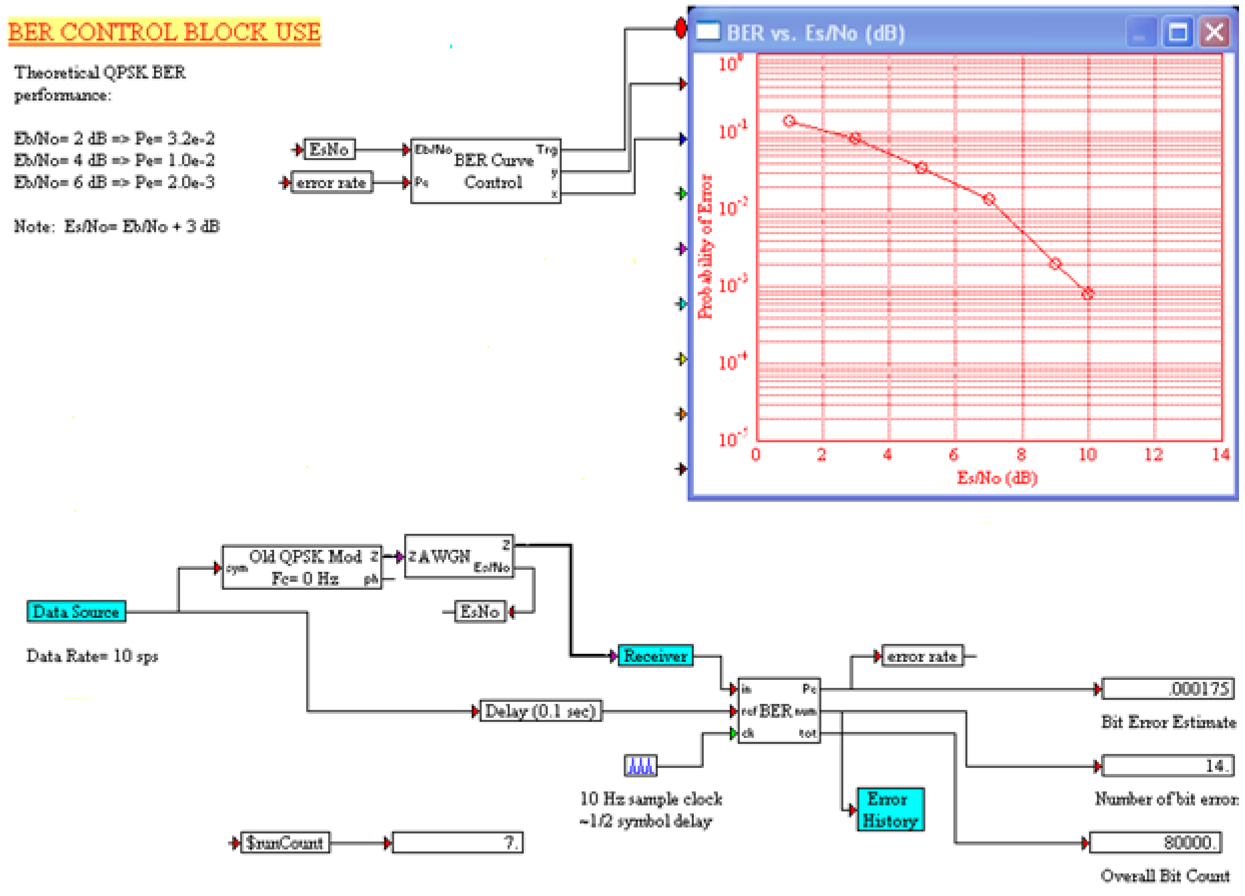
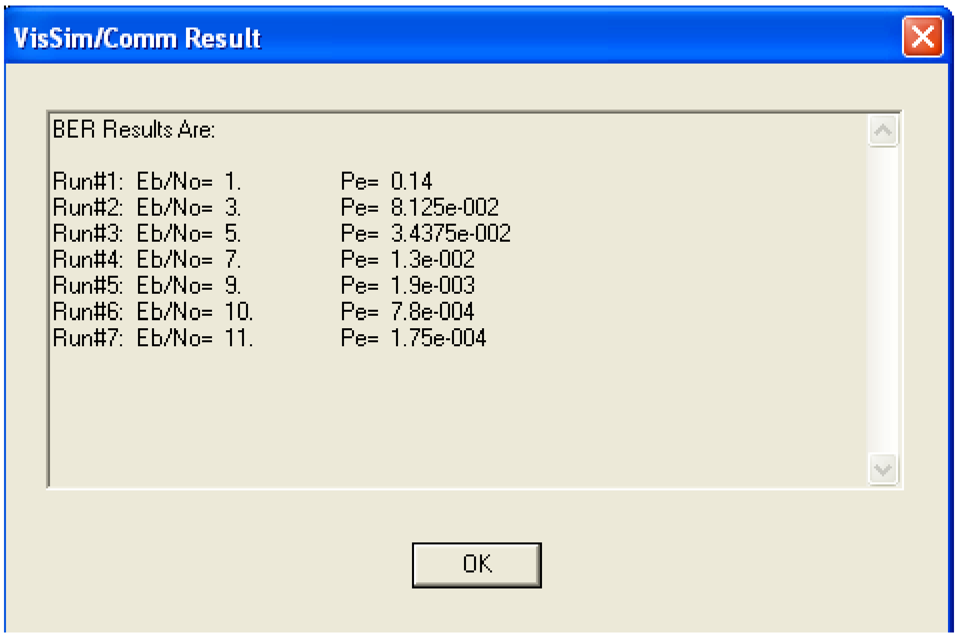


Fig 3.1: BER Control Block Use

3.1.5 OUTPUT

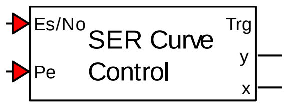


```
BER Results Are:

Run#1:  Eb/No=  1.        Pe=  0.14
Run#2:  Eb/No=  3.        Pe=  8.125e-002
Run#3:  Eb/No=  5.        Pe=  3.4375e-002
Run#4:  Eb/No=  7.        Pe=  1.3e-002
Run#5:  Eb/No=  9.        Pe=  1.9e-003
Run#6:  Eb/No=  10.       Pe=  7.8e-004
Run#7:  Eb/No=  11.       Pe=  1.75e-004
```

Results of the various Runs in BER control

3.1.6 BER Curve Control



This block is used to automatically control the generation of BER curves. It allows the user to specify individual run times for each of a BER simulation's multiple runs. In order for this block to function properly, it is necessary to activate the Auto Restart parameter in the Simulation Properties dialog box.

To control a BER simulation by specifying the number of desired errors the BER Control (# Errors) block is to be used.

The BER Curve Control block allows up to ten consecutive iterations of the simulation, each with its own time duration expressed in seconds. The BER Curve Control block accepts the current Es/No value (from an AWGN block or other custom source) and an output error rate from a Bit/Symbol Error Rate block. Care should be taken to match the number of runs in this block with those specified in the AWGN block (if used).

This block provides outputs to be used with a plot block configured for external trigger, XY plotting, and log Y scaling. The BER curve result is updated at the end of each run in a multi-run scenario (Auto Restart mode). At the end of the last run, an optional written BER summary message is provided, as shown in the figure below.

x1 = Current Es/No level

x2 = Error rate estimate (from Bit/Symbol Error Rate block)

y1 = Trigger for BER plot

y2 = Error rate results for BER plot (y-axis signal - use log scale)

y3 = SNR data for BER plot (x-axis signal)

Number of Runs

Specifies the number of simulation iterations. The valid range is from 1 to 10.

Bit Error Rate

Forces the use of the Eb/No label in the results summary.Use this setting when providing a reference Eb/No input to the block.

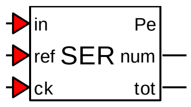
Symbol Error Rate

Forces the use of the Es/No label in the results summary. Use this setting when providing a reference Es/No input to the block.

Duration

Specifies each run's duration in seconds. As the SNR is made larger, a longer duration is usually necessary to obtain a reliable error rate estimate.

3.1.7 Bit/Symbol Error Rate



This block accepts either bits or symbols as input and can output either a Bit Error Rate (BER) or a Symbol Error Rate (SER) by comparing a recovered data stream to a reference data stream.

In order for this block to operate properly, the reference data stream must be delayed by the same amount as the recovered data stream. An external sampling clock must be provided to the Bit/Symbol Error Rate block. Sampling at approximately the half symbol point is recommended.

x1 = Recovered data stream

x2 = Reference data stream

x3 = External Clock (0, 1) (impulse train)

y1 = Error rate (symbol or bit)

y2 = Error count ((symbols or bits) (optional))

y3 = Total count ((symbol or bits) (optional))

Count Start Delay

Specifies the initial delay in symbol counts before starting the error counting process. A symbol count occurs each time the sampling clock goes high.

Bit Error Rate

Specifies the output error rate as a BER. In this mode, the total number of bits that are in error within a symbol is counted. The total count output shows the total number of symbols processed times the number of bits/symbol.

Symbol Error Rate

Specifies the output error rate as an SER. In this mode, regardless of how many bits within a symbol are in error, a single symbol error is recorded.

Bits per Symbol

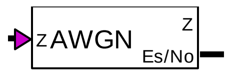
Specifies the number of bits per symbol. This parameter is only available when bit error rate mode is selected.

3.1.8 AWGN (Complex or Real)

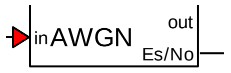
In communications, the additive white Gaussian noise (AWGN) channel model is one in which the information is given a single impairment: a linear addition of wideband or white noise with a flat spectral density. In other words, the signal contains equal power within a fixed bandwidth at any center frequency. White noise draws its name from white light in which the power spectral density of the light is distributed over the visible band in such a way that the eye's three color receptors (cones) are approximately equally stimulated. It is expressed as watts per hertz of bandwidth and a Gaussian distribution of noise samples. The model does not account for the phenomena of fading, frequency selectivity, interference, nonlinearity or dispersion. However, it produces simple and tractable mathematical models which are useful for gaining insight into the underlying behavior of a system before these other phenomena are considered.

Wideband Gaussian noise comes from many natural sources, such as the thermal vibrations of atoms in antennas (referred to as thermal noise or Johnson-Nyquist noise), shot noise, black body radiation from the earth and other warm objects, and from celestial sources such as the Sun.

In VisSim these blocks implement an AWGN channel in which Gaussian noise is added to the input signal. Two versions of this block exist: one block is complex and the other is real.



Complex AWGN block



Real AWGN block

The appropriate noise variance is automatically computed based on the desired SNR, simulation sampling frequency, symbol rate, and reference signal power. The block supports multiple run simulations by allowing up to ten different SNR values to be specified. The SNR is specified as Es/No, as opposed to Eb/No.

The AWGN blocks can be used in conjunction with the BER Curve Control block or the BER Control (# Errors) block to generate Bit Error Rate (BER) curves. The information signal's power is specified as a parameter, as is the single-sided noise bandwidth. The simulation step size is also taken into account in computing the noise variance. In the case of the complex version of the block, the two noise samples (real and imaginary) are independent.

The AWGN blocks can also be used as a source of Gaussian noise by simply leaving the inputs disconnected or using a 0 input.

x = Input signal ([Re, Im] for complex)

y1 = Output signal ([Re, Im] for complex)

y2 = Current run's Es/No value

Number of Runs

Specifies the number of simulation iterations. The maximum is ten.

Symbol Rate

Specifies the symbol rate R in hertz. This value is used internally to determine the energy per symbol as a fraction of the total specified signal power.

Real Average Complex Signal Power
Real Average Signal Power

Specifies the complex or average power of the information bearing signal and is used in calculating the appropriate noise variance.

3.1.9 QPSK MODULATOR:

Phase modulation is a version of frequency modulation where the phase of the carrier wave is modulated to encode bits of digital information in each phase change. The term "quadrature" implies that there are four possible phases (4-PSK) which the carrier can have at a given time, as shown at right on the characteristic constellation for this modulation type. The four phases are labeled {A, B, C, D} corresponding to one of {0, 90,180,270} degrees.

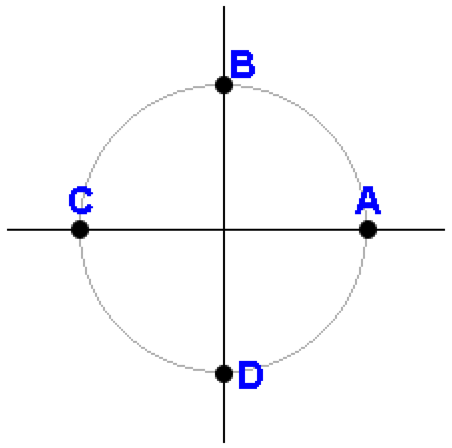
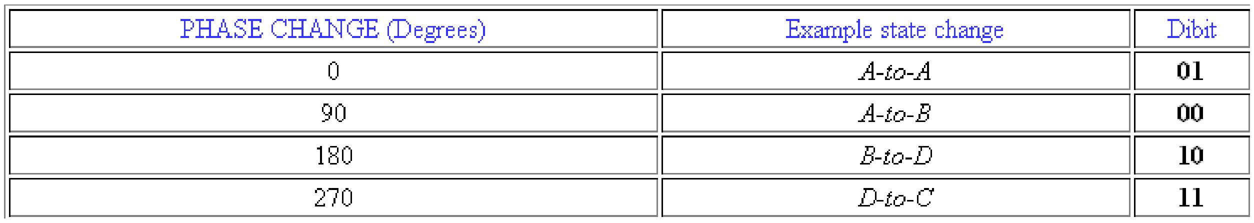


Fig 3.2: QPSK Modulator

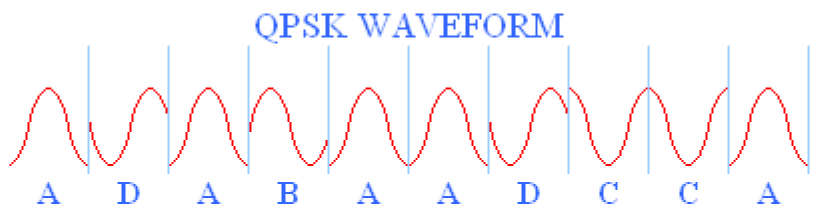
Transmitting Data Using QPSK:

Each of the four possible phase changes is assigned a specific two-bit value, or dibit. For example, in the V.22 modem, the relationship between phase changes and dibits is given by:



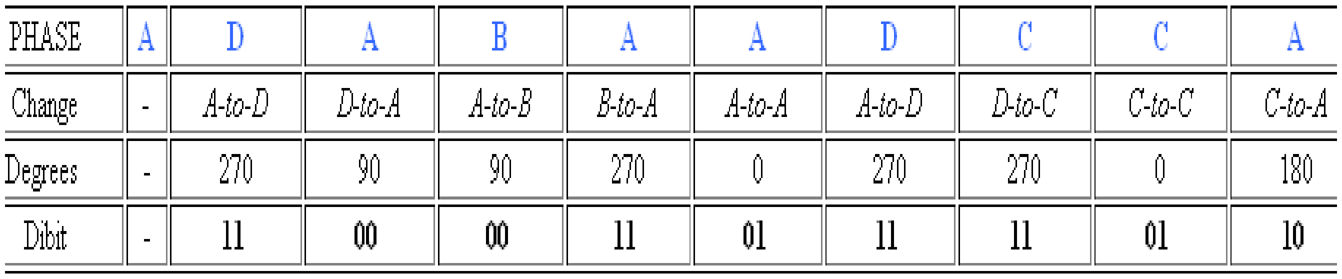
PHASE CHANGE (Degrees)	Example state change	Dibit
0	*A-to-A*	**01**
90	*A-to-B*	**00**
180	*B-to-D*	**10**
270	*D-to-C*	**11**

In the figure below, a carrier is shifted through the phases ADABAADCCA.



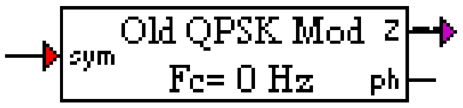
Two bits of information are conveyed in the transition between time slots.

The signal has undergone the following phase transmissions:



PHASE	A	D	A	B	A	A	D	C	C	A
Change	-	A-to-D	D-to-A	A-to-B	B-to-A	A-to-A	A-to-D	D-to-C	C-to-C	C-to-A
Degrees	-	270	90	90	270	0	270	270	0	180
Dibit	-	11	00	00	11	01	11	11	01	10

The VisSim block for QPSK Modulation is shown below:



CONCLUSION

In this paper the different components of a communication system were studied in details. We implemented an Adaptive Equalizer using four different algorithms in Matlab. We have suggested ways to decide the coefficients of the equalizer. The first procedure (LEAST SQUARE ALGORITHM) minimized the square of the symbol recovery error over a block of data which was done by using matrix pseudo inversion. The second method (LEAST MEAN SQUARE ALGORITHM) involved minimizing the square of the error between the received data values and the transmitted values which were achieved via an adaptive element. The third method (DECISION DIRECTED ALGORITHM) and the fourth method (DISPERSION MINIMIZING ALGORITHM) were used when there was no training sequence and other performance functions were appropriate. In addition to this we undertook the study and realization of Bit Error Rate of a communication system using VisSim software.

Therefore, with the design of the receiving filters, we can minimize the effects of Inter symbol Interference, Multipath Interference and Additive Interference, and thereby deliver the digital data to its destination with the smallest error rate possible. Hence we can improve the efficiency of a communication system.

REFERENCES

1. Analog and Digital Communications, S.Haykins ,Prentice hall,1996

2. Principles of Communication Systems, Herbert Taub, Donald L Schilling, Goutam Saha

3. Digital Communications, John G.Proakis, Prentice-Hall of India, 2003

4. S. U. H. Qureshi, “Adaptive equalization,” Proceedings of the IEEE, vol. 73, no. 9, pp. 1349-1387, 1985

5. W.A.Sethares, “The LMS Family,” in Efficient System Identification and Signal Processing Algorithms, Ed. N. Kalouptsidis and S.Theodorodis Prentice-Hall,1988

6. C.R.Johnson Jr., Lectures on Adaptive Parameter Estimation, Prentice-Hall,1988