University of Windsor

WIRELESS COMMUNICATION SYSTEM

PROJECT REPORT ON
"DESIGN OF ADAPTIVE CHANNEL EQUALIZER"

Prepared By
NARAYANAN SRINIVASAN – 105192312
MANISHCHANDAR SUNDARAVEL – 110023774
Master of Electrical and Computer Engineering
University of Windsor

Submitted To
DR. ESAM ABDEL-RAHEEM
Ed Lumley Centre for Engineering Innovation (CEI)
University of Windsor
Ontario N9B 1K3

Introduction:

The main aim of the project is to design an equalizer for the two modes one with training mode and another with blind mode. To design this, first we should have some knowledge about equalizer and their needs in communication. Generally, Equalizer is a device that removes distortion incurred by signal when transmitted through channel [1]. When a signal is passed through multipath fading scattering environment, the receiver receives a signal which are all constantly changing and delayed versions of the original signal. These symbols with delayed versions and overlapped version of original signal leads to Intersymbol Interference. In communication Intersymbol Interference is a serious problem as when the signals are distorted it will be difficult for the receiver to create original signal. The main purpose of the Equalizer is to reduce Intersymbol Interference allowing fast recovery of the transmitted signals or symbols in communication and it also improves receiver's performance.

Adaptive Filter:

Adaptive filter is like an ordinary linear filter with variable parameters controlling transfer function where the parameters are adjusted according to optimization algorithm [2]. Due to the complexity of the algorithms used mostly all adaptive filters are digital filters. The main purpose of the adaptive filter is to get the coefficients of the unknown received signal to identify the system and reduce the error. The most common version of adaptive filter used is closed loop adaptive filter. The closed loop adaptive filter uses error signal to adjust the parameters. Figure 1 shows the block diagram of adaptive filter.

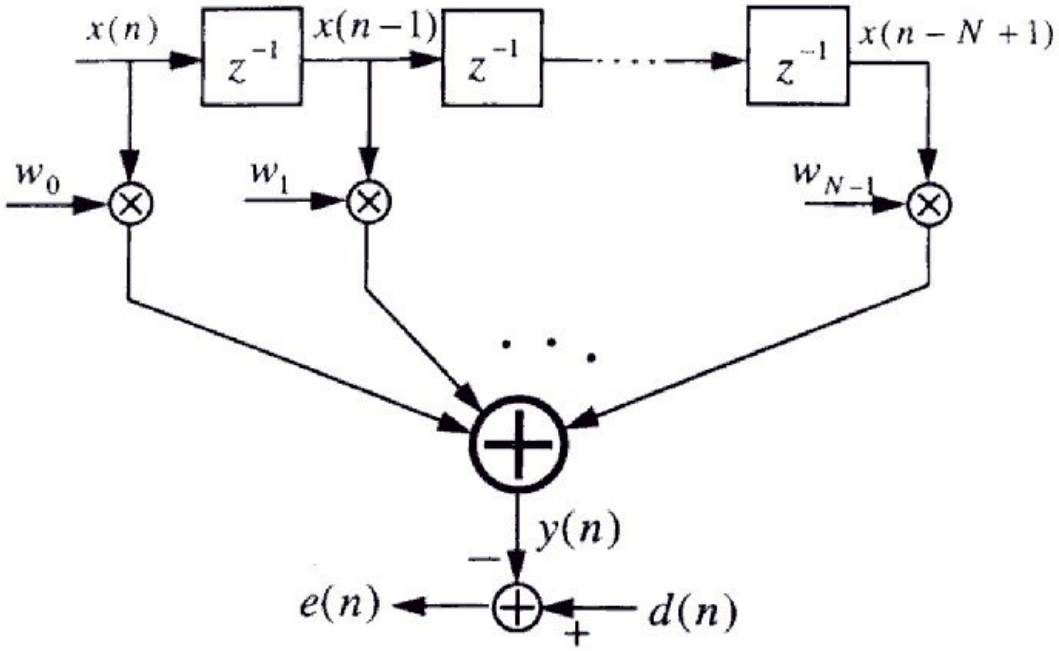

Figure 1: Block Diagram of Adaptive filter

Here w represents the FIR filter tap weight vector coefficients, x(n) is the samples of the input vector, z1 i s a delay of one sample period, y(n) is the adaptive filter output, d(n) is the desired echo signal, and e(n) is the error of estimation at time n [3].

The aim of an adaptive filter is to calculate the error signal which can be calculated as the difference between desired signal and adaptive filter output. This error signal is once again fed back into the adaptive filter where their coefficients are changed to minimize a function which is namely cost function. When both signals i.e. adaptive signal output and desired signal are matched the error signal will be zero. In this case the echoed signal would be completely cancelled leads to proper original signal.

LMS Algorithm:

LMS stands for least mean square algorithm originally it was first developed by Widrow and Hoff. From there it became one of the most used adaptive filtering algorithms. The LMS algorithm is a type of adaptive filter known as stochastic gradient-based algorithms, as it uses the filter tap weights' gradient vector to converge on the optimal wiener solution. Owing to its computational simplicity, it is well known and commonly used. With each iteration the filter tap weights of adaptive filter are periodically updated according to the below equation:

$$\mathbf{W}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

Here x(n) is the input vector of time delayed input values, x(n) = [x(n) x(n-1) x(n-2) .. x(n-N+1)]T. The vector w(n) = [w0(n) w1(n) w2(n) .. wN-1(n)] T represents the coefficients of the adaptive FIR filter tap weight vector at time n. m is called step size parameter where this parameter controls the updating factor. I f the step size value is too small the output converges and will yield optimal solution, but it will take some time on the other hand when the step size is too large then the output diverges, and the signal will be unstable.

Implementation of the LMS Algorithm:

Every iteration of LMS algorithm follows these steps:

1) The adaptive filter output y(n) is calculated by the below expression:

$$y(n)=\mathbf{w}^{T}(n)\mathbf{x}(n)$$

2) The error signal e(n) can be calculated from the below expression:

$$e(n)=d(n)-y(n)$$

3) The tap weight of the FIR filter is calculated from the below expression:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

Because of the computational simplicity compared to all other algorithms it is most preferred one and the popular one.

Procedure:

1. The input bipolar sequence which is to transmitted can be obtained from the expression **a= (randn(ni,1)>0) *2-1;**
2. The transmitted is passed through the channel **h**, where the channel is given **h= [0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];**
3. The transmitted signal **a** and **h** multiplexed to yield the output **ah.**
4. The output **ah** is added with the additive white gaussian noise signal given by the expression **v= sqrt(0.001)*randn(ni,1);**
5. Now the output will be **u=ah+v;**
6. This output signal is passed through adaptive filter H(Z) we have designed with certain specifications.
7. The signal from the filter **y** is fed into LMS algorithm for the training purpose along with considerable delay function from our filter.
8. This process is continuous till it yields desired signal which is exact replica of the transmitted signal and for every iteration error signal is calculated **e(n)**.
9. At the final, a Decision device is used to select a perfect signal which closely resembles the transmitted signal.
10. In the meanwhile, learning curve and BER are calculated for the study purpose of the LMS algorithm.

Part 1:

Training Mode equalizer:

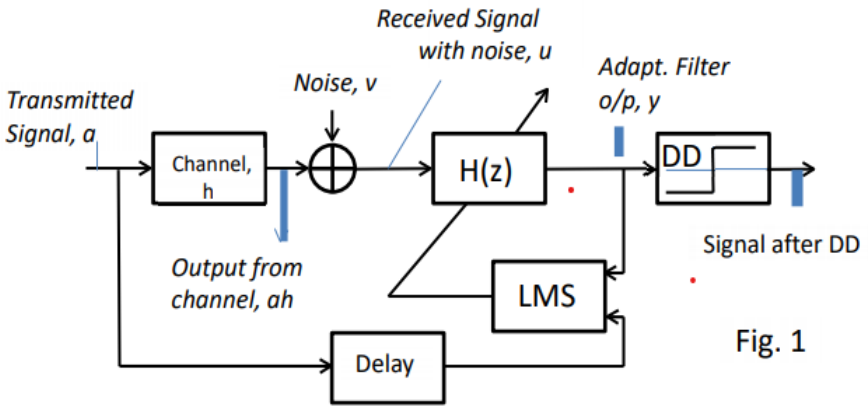

Fig 1: Training-Mode Adaptive Channel Equalization

Given:

```
a= (randn(ni,1)>0) *2-1;
ni=10000;
h= [ 0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];
v= sqrt(0.001)*randn(ni,1);
```

Matlab Code :

First Run : (Step Size mu=0.05)

```
% LMS Algorithm
%Given
No_of_Seq=10000;    % number of data samples
h=[0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];
Ch_length=length(h);  % length of the channel
Trans_signal=(randn(1,No_of_Seq)>0)*2-1;  % Bipolar sequence
length_Trans_signal= length(Trans_signal); % length of bipolar sequence
x=filter(h,1, Trans_signal);  %channel adaptive filter delay
mu=0.007;  % Random selected the step size
N = 11;
W = zeros(1,N);
v=sqrt(0.001)*randn(1,No_of_Seq);
% Additive white gaussian noise
u = x(1: No_of_Seq)+v;
d = N-1;
c= zeros(1,N);
b=[zeros(1,d) Trans_signal (1: No_of_Seq -d)];
[out,W,err]= fundef_LMS(No_of_Seq,u,c,W,b,mu);
for i=1:length(out)
if out(i)>=0.5,f(i)=1;
```

```
else f(i)=-1;
end
end
% decision Device
Dec_Dev=Trans_signal(1:length_Trans_signal-(N-1));
g=f(N:length(f));
l=0;
for j=1:length(Dec_Dev)
if Dec_Dev(j)~=g(j), l=l+1;
end
end
berc=l./(length(g));  %Bit-Error-Rate calculation
disp (berc);
plot(Trans_signal,'.'), grid, title('Input')
xlabel('Samples')
ylabel('Range')
plot(out), grid, title('Output')
xlabel('Samples')
ylabel('Range')
plot(abs(err),'.'), grid, title('Convergence')
xlabel('Samples')
ylabel('Range')
fvtool(W)
fvtool(err)
fvtool(in)fvtool(err)
fvtool(in)
```

Function code for the above LMS Algorithm MATLAB Code:

```
function [out,W,err] = fundef_LMS(No_of_Seq,Noise,c,W,b,mu)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
err=[];
for i=1:No_of_Seq
c=[Noise(i) c(1:length(c)-1)];
out(i)=W*c';
e=b(i)-out(i);
err(i)=e;
W=W+mu*e*c;
end
end
```

Test 1 Graphs Obtained:

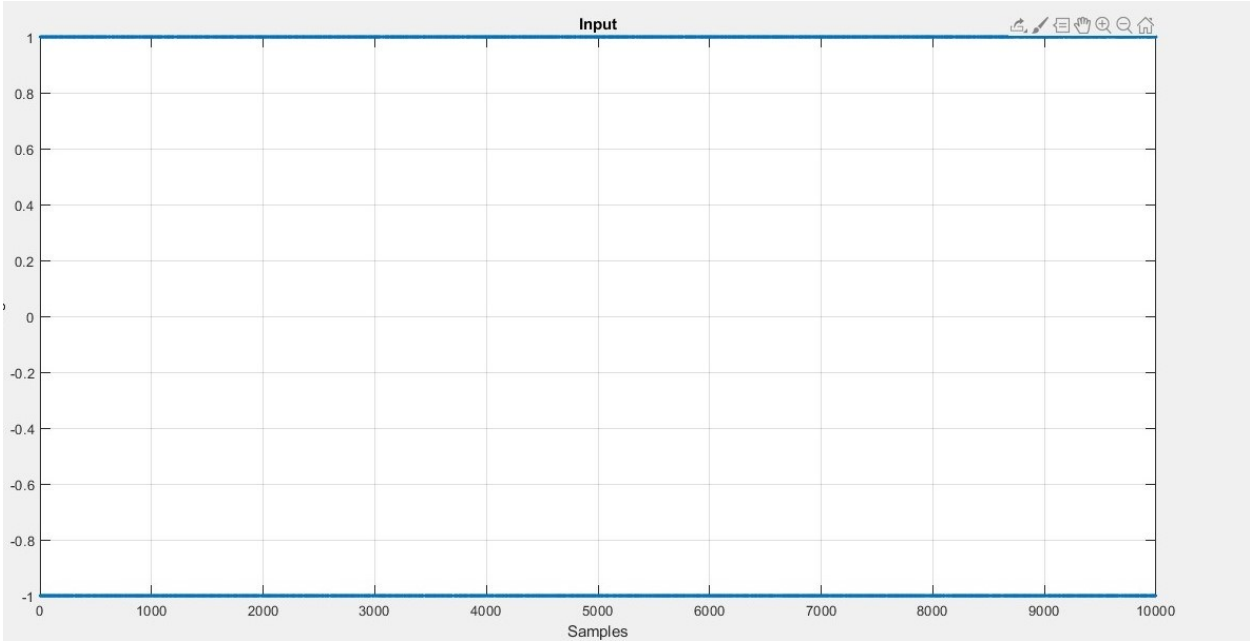

Fig 2: Input Signal Graph plot

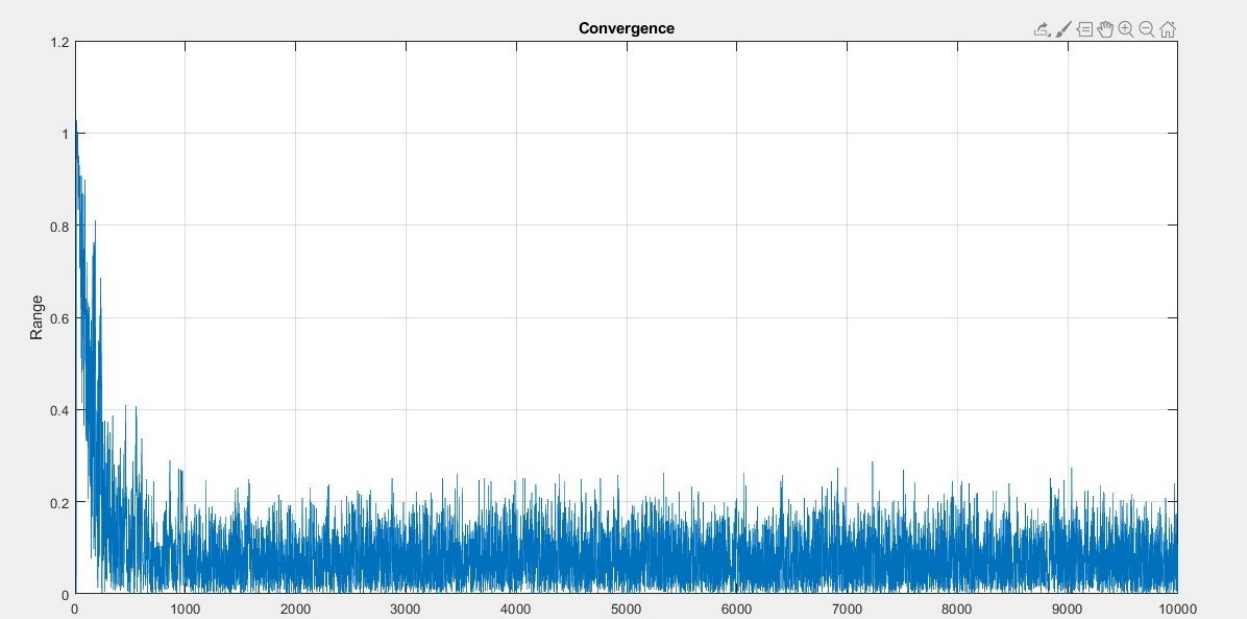

Fig 3: Convergence Graph Plot

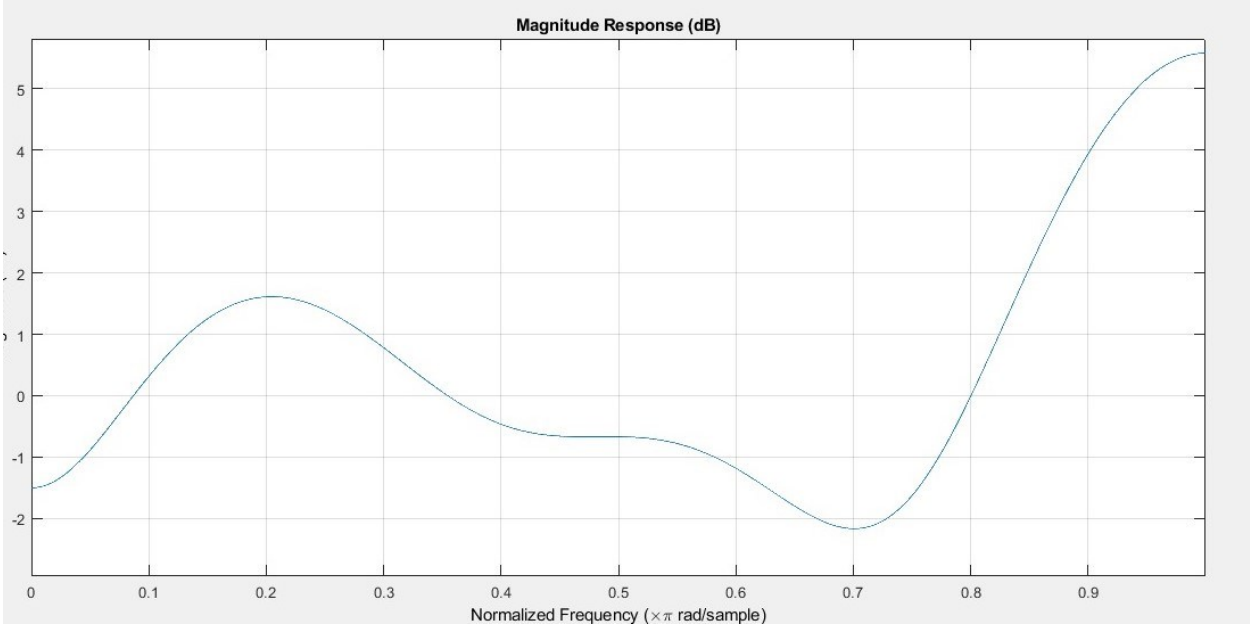

Fig 4: Weight Graph Plot

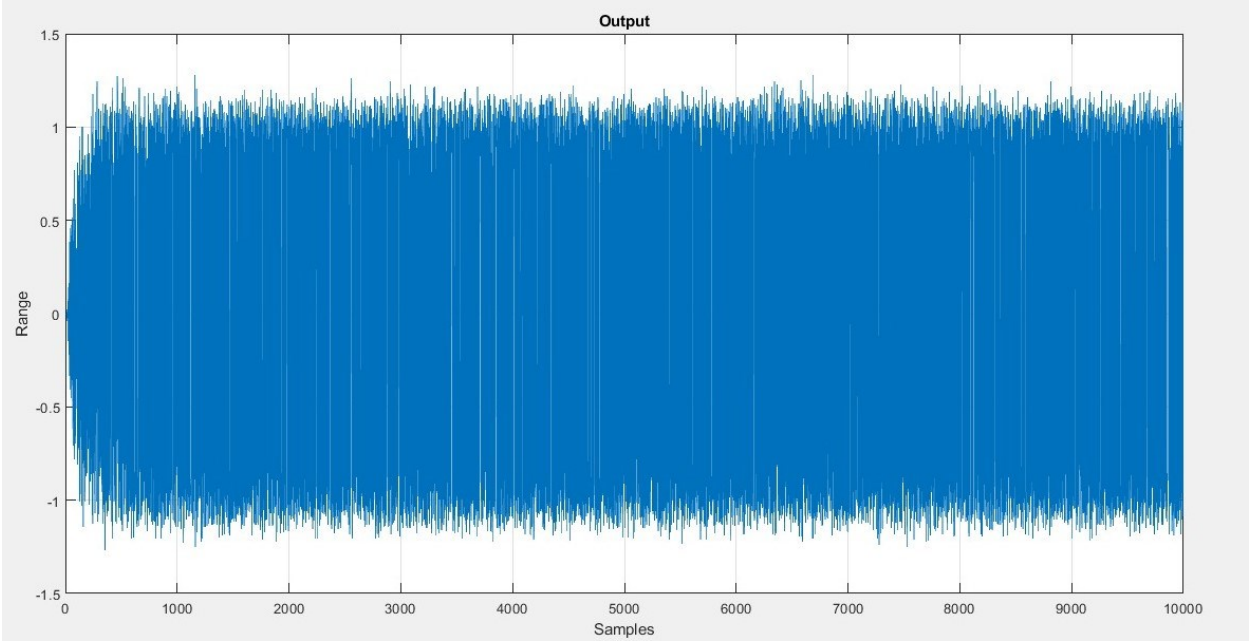

Fig 5: Output Signal Graph Plot

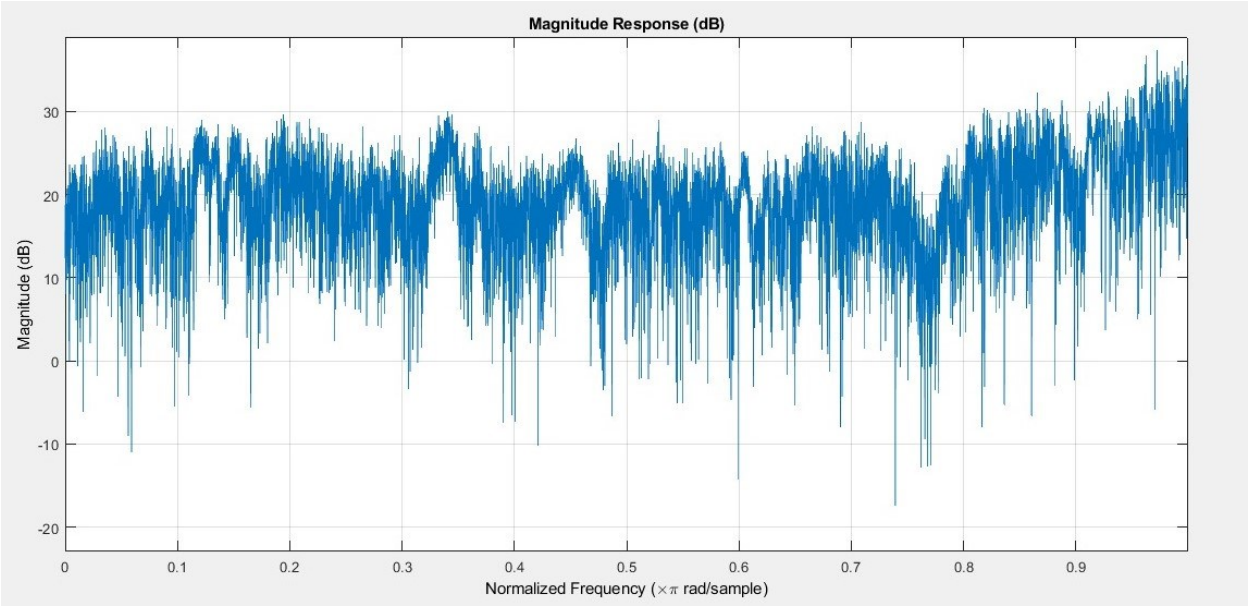

Fig 6: Error Graph Plot

Second Run: (Step Size mu=0.07)

```
% LMS Algorithm
%Given
No_of_Seq=10000;    % number of data samples
h=[0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];
Ch_length=length(h);  % length of the channel
Trans_signal=(randn(1,No_of_Seq)>0)*2-1;  % Bipolar sequence
length_Trans_signal= length(Trans_signal); % length of bipolar sequence
x=filter(h,1, Trans_signal);  %channel adaptive filter delay
mu=0.009;  % Random selected the step size
N = 11;
W = zeros(1,N);
v=sqrt(0.001)*randn(1,No_of_Seq);
% Additive white gaussian noise
u = x(1: No_of_Seq)+v;
d = N-1;
c= zeros(1,N);
b=[zeros(1,d) Trans_signal (1: No_of_Seq -d)];
[out,W,err]= fundef_LMS(No_of_Seq,u,c,W,b,mu);
for i=1:length(out)
if out(i)>=0.5,f(i)=1;
else f(i)=-1;
end
end
% decision Device
Dec_Dev=Trans_signal(1:length_Trans_signal-(N-1));
```

```
g=f(N:length(f));
l=0;
for j=1:length(Dec_Dev)
if Dec_Dev(j)~=g(j), l=l+1;
end
end
berc=l./(length(g));  %Bit-Error-Rate calculation
disp (berc);
plot(Trans_signal,'.'), grid, title('Input')
xlabel('Samples')
ylabel('Range')
plot(out), grid, title('Output')
xlabel('Samples')
ylabel('Range')
plot(abs(err),'.'), grid, title('Convergence')
xlabel('Samples')
ylabel('Range')
fvtool(W)
fvtool(err)
fvtool(in)
```

Bit Rate Error (BER) Values

The BER Value for LMS Test 1 is 0.0049

The BER Value for LMS Test 1 is 0.0044

Test 2 Graphs Obtained:

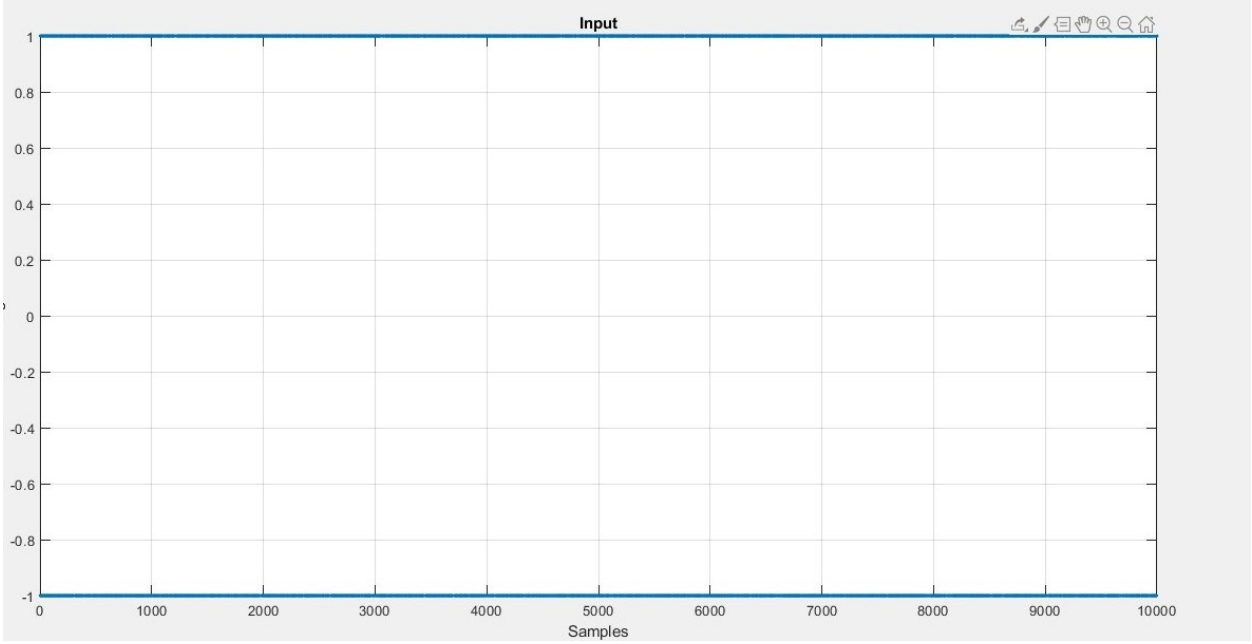

Fig 7: Input Signal Graph

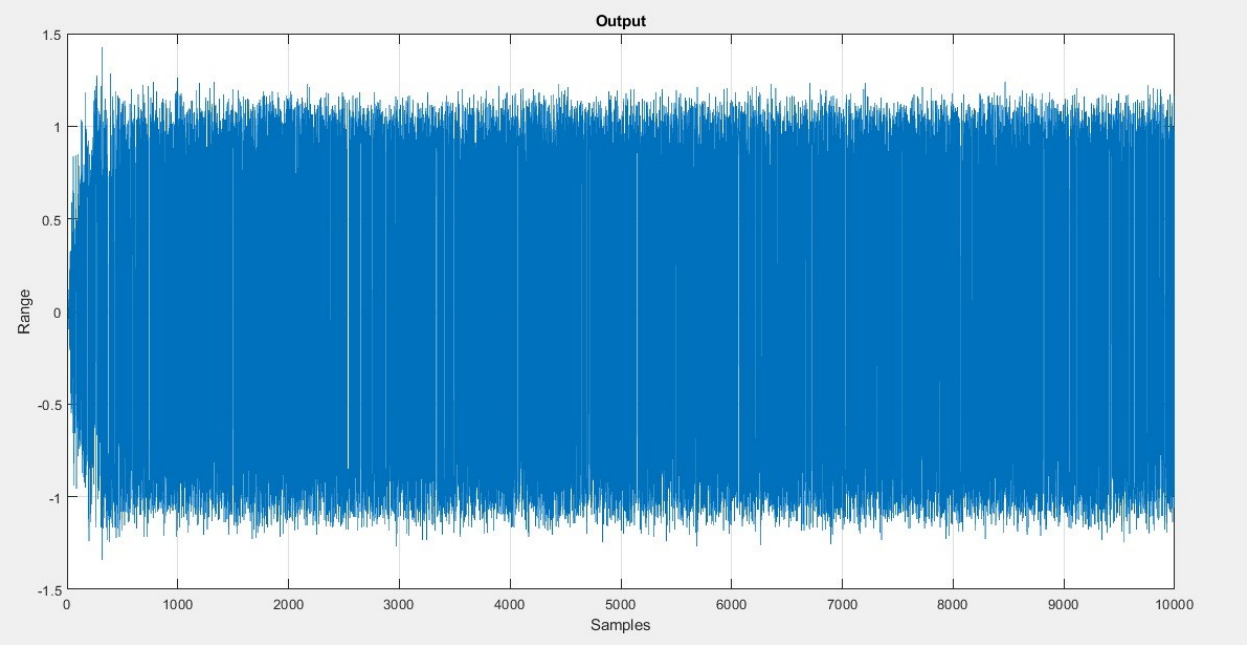

Fig 8: Output Signal Graph

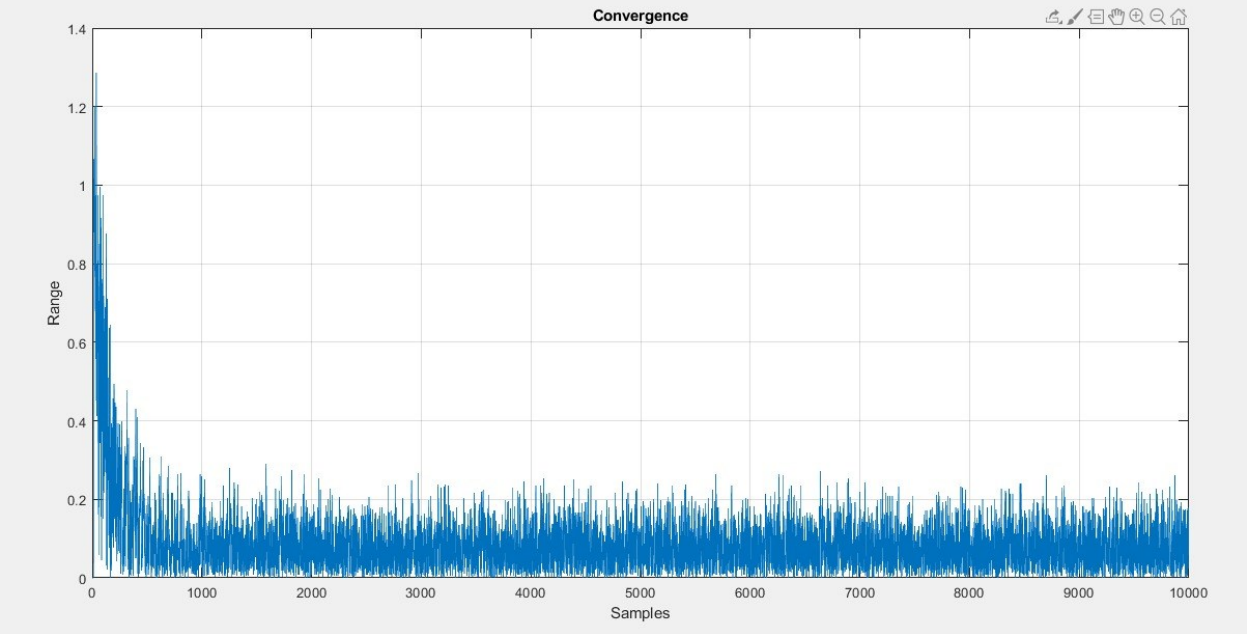

Fig 9: Convergence Graph

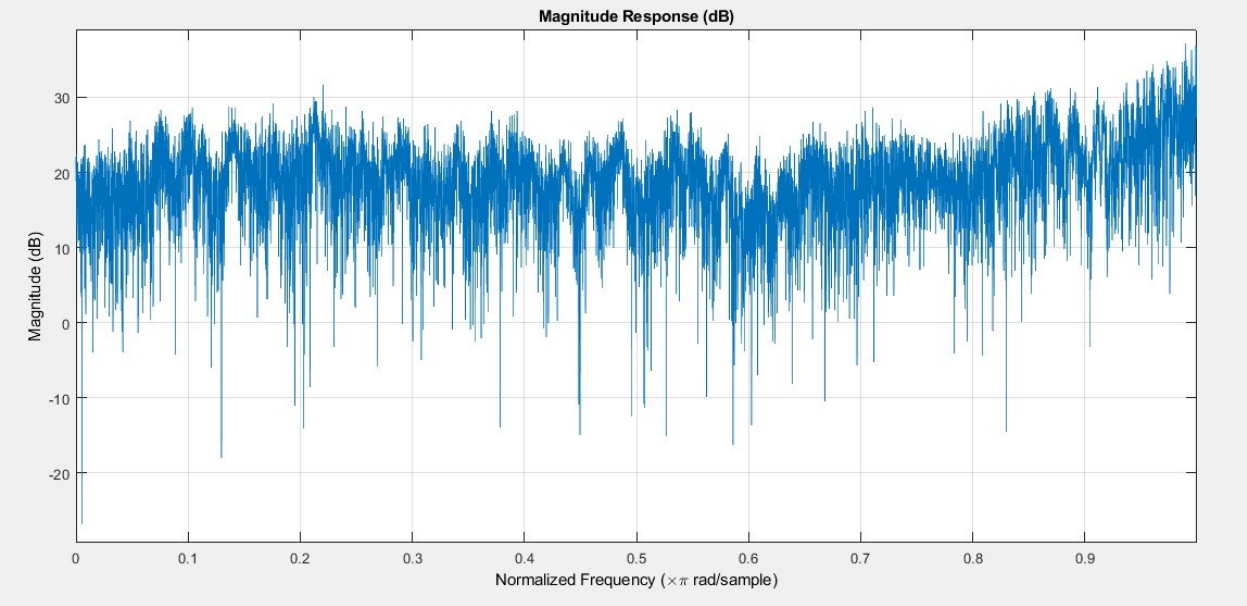

Fig 10: Error Graph

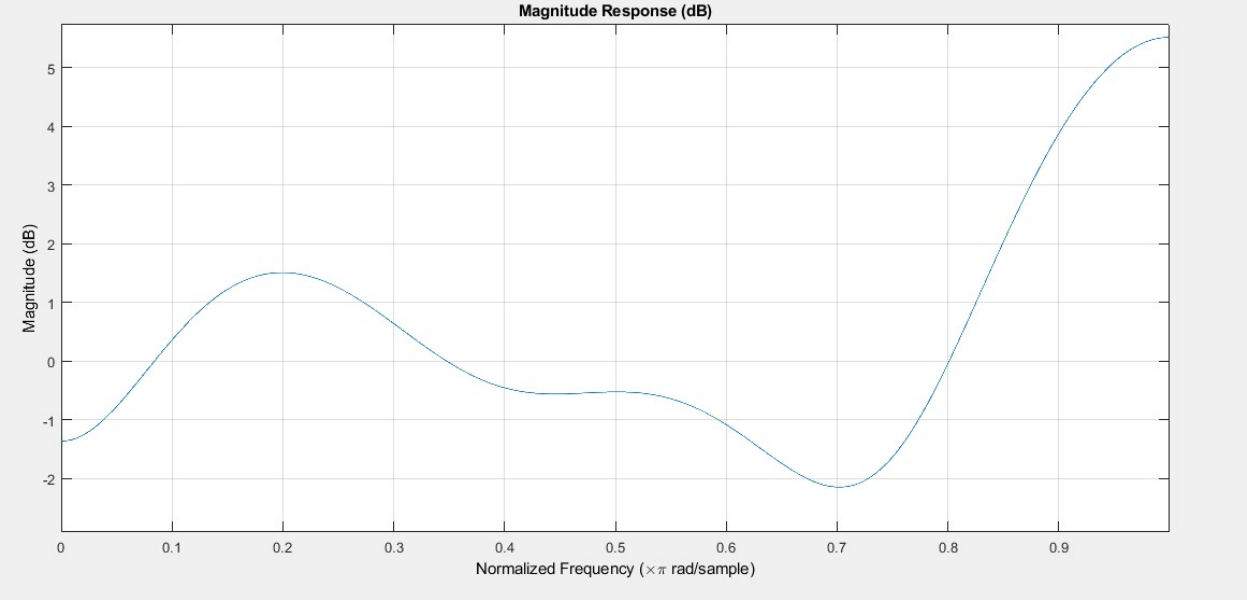

Fig 11: Weight Graph

Training-Mode Adaptive Equalizer Comment :

As we analyze the above graphs, we have tested 2 different conditions with different mu values in the Training-mode adaptive equalizer. The graphs obtained are input signal, convergence, error and output signals. By comparing the Bit Rate Error (BER) values of the two test runs of LMS we can understand that by increasing the step size value the BER value decreases. Hence, we can reproduce the original transmitted signal with higher accuracy. However, there is one condition where when a higher step size is chosen we can exactly reproduce the input transmitted signal hence we increase the step size little by little at a gradual rate.

Channel Equalizer:

Channel equalizer is designed to perform the distortion of signal that is transmitted through a channel h. In a high data rate digital communication system, the equalizers are crucial to restore the information transmitted, i.e., decrease or elimination of channel interference at the input. The primary function of the channel equalizer in a dispersive channel h with a slow time-variation is reducing the Inter-Symbol-Interference (ISI) formed by the multiple channels more destructive to the signal than the comparative channel noise. The ISI occurs if the modulation bandwidth exceeds the coherence bandwidth of the channel where the modulation pulses are spread in time.

The efforts of channel equalization focus on reducing or eliminating the ISI distortion caused by the multipath filter [5]. The quality of the filtered signal is based on the performance of the equalizer. A channel equalizer is designed based on the expected channel amplitude and delay characteristics. The equalizer converges on the three factors which span over time such as equalizer algorithm, equalizer structure and time rate of change of the multipath radio channel. Furthermore, these equalizers are designed to be adaptive since the multi-channel bandwidth is generally unknown and time varying.

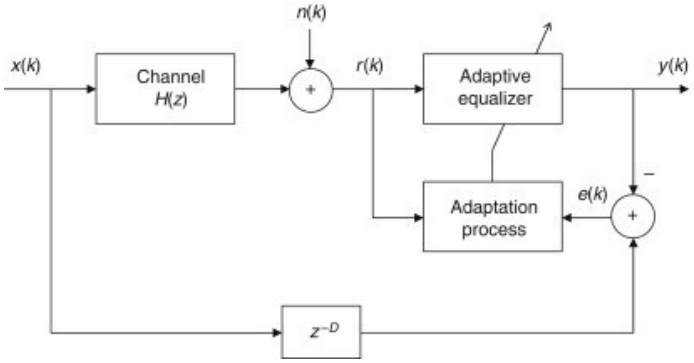

Fig 12: General schematic of an Adaptive Channel Equalization

In channel equalization, various techniques and methods are used to perform equalization. However, several equalizers use adapt the co-efficient of equalizer such as LMS, RLS and CMA algorithms. There are different types of channel equalizers ranging from a simple linear filter to a complex equalizer algorithm. Some of the equalizers in digital communications are linear equalizer, MSME equalizer, Zero forcing equalizer, Decision feedback equalizer, Blind equalizer, Adaptive equalizer, Viterbi equalizer, BCJR equalizer, and Turbo equalizer [4].

Learning Curve in Equalizer

The learning curve in an equalizer is the representational graph which represents the rate at which the equalizer algorithm estimates output using the input sequence and reference sequence. The performance in training and learning capacity of an equalizer is analyzed based on the learning curve graph.

Part 2:

Blind-Mode Adaptive Equalizer:

In a blind mode adaptive filter, the digital signal transmitted is equalized from the received signal, utilizing the signal statistics which is unknown to the receiver. The blind equalization of a digital signal is basically a blind deconvolution applied to the input signal. In this mode, the focus of blind equalization is the equalizer filter estimation, which is the inverse of the channel impulse response other than the estimation of channel impulse response. In the blind mode adaptive there are two modes of operation, noiseless mode, and noisy channel. In a conventional LMS adaptive equalizer, the filters depend on the use of training sequence hence the blind-mode adaptive channel equalizer was designed with algorithms that do not rely on training signals [6]. With these blind algorithms, the individual receivers will have the ability to self-recover from system breakdowns. Such self-recovery ability is critical in multi-cast and broadcast systems where the channel variation occurs commonly [6]. The performance of the blind-mode adaptive filter can be evaluated using transient and steady-state analysis.

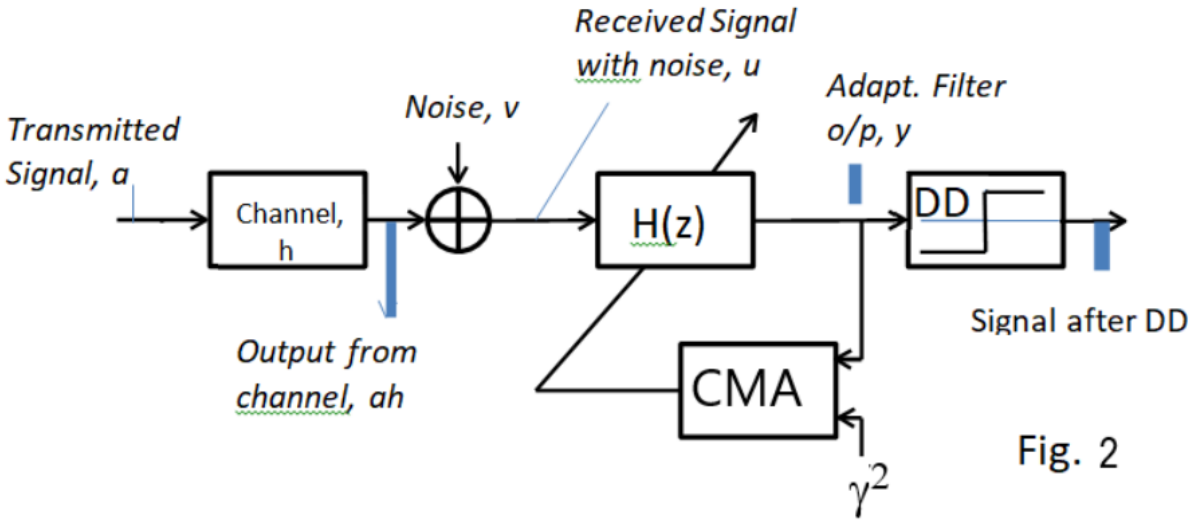

Fig 13: Blind-Mode Adaptive Channel Equalization

Methodology

Given:

a= (randn(ni,1)>0) *2-1;
ni=10000;
SNR=35dB (i.e., SNR = Signal to Noise ratio)
h= [0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];
v=radn(length(ah),1) ;
v= v*sqrt(1/SNR)*(sqrt(0.5));
snrdb=35;

Procedure

1. The bipolar input signal sequence ‘a’ is obtained from the given expression **a=(radn(ni,1)>0)*2-1;**
2. The transmitted input signal passes through a channel h, where the values of channel is given as given **h= [0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];**
3. The transmitted input signal **a,** and channel **h** is multiplexed where the output **ah** is obtained

4. Along with this the transmitted signal **ah** the additive gaussian noise signal **v** is multiplexed. Where the noise signal v is given by the expression **v= sqrt(0.001)*randn(ni,1);**
5. At this stage, the output signal is represented by **u=ah+v;**
6. Next the output signal **u** is passed through the adaptive filter **H(z)** which is designed with certain design characteristics.
7. The output from the adaptive filter H(z) which is **y** is fed into the Constant Modulus Algorithm (CMA) in a feedback loop.
8. The CMA here is fed with two input of which one is **y** the output of the H(z), and dispersion factor γ^2 . Then the output of the CMA is fed into the H(z). Comparing with the LMS mode, here no delay is used.
9. At the final output, the decision device is used to select a perfect signal which closely resembles the unknown transmitted signal.
10. In the meantime, the learning curve and BER are calculated for the analysis of the CMA Algorithm.

Matlab Code :
First Code: (Step Size mu=0.07)

```
% CMA Algorithm
% Given
No_of_Seq=10000;    % number of data samples
snrdb=35;     % Signal to noise ratio(dB)
snr = 10.^(snrdb/10);
h=[0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];     % The given channel
Ch_length=length(h);  % length of the channel
Trans_signal=(randn(1,No_of_Seq)>0)*2-1;  % Bipolar sequence
Length_Trans_signal= length(Trans_signal); % lenght of bipolar sequence
x=filter(Ch_length,1,Trans_signal);  %channel adaptive filter delay
mu=0.007;  % Random selected the step size
N = 11;  % size of filter given in problem
W= zeros(1,N);  %Filter coefficient
v = randn(length(x),1);
v1 = sqrt(1/snr)*(sqrt(0.5)); % Additive white gaussian noise
u = x(1:No_of_Seq)+v1;
d = N-1;
b = zeros(1,N);
c=[zeros(1,d) Trans_signal(1:No_of_Seq-d)];
[out,err,W]= fundef_CMA(No_of_Seq,u,b,W,c,mu);
for i=1:length(out)
if out(i)>=0.5,f(i)=1;
else f(i)=-1;
end
end
Dec_Dev=Trans_signal(1:Length_Trans_signal-(N-1));
g=f(N:length(f));
l=0;
for j=1:length(Dec_Dev)
if Dec_Dev(j)~=g(j), l=l+1;
end
```

```
end
berc=l./(length(g));  %Bit-Error-Rate calculation
disp (berc);
plot(No_of_Seq,'.'), grid, title('Input Signals');
xlabel('Samples')
ylabel('Range')
plot(out), grid, title('Output Signal');
xlabel('Samples')
ylabel('Range')
plot(abs(err)), grid, title('Convergence');
xlabel('Samples')
ylabel('Range')
fvtool(W)
fvtool(err)
fvtool(in)
```

Function code for the above LMS Algorithm MATLAB Code:

```
function [out,err,W] = fundef_CMA(No_of_Seq,u,b,W,c,mu)
err = [];
for i=1:No_of_Seq
b=[u(i) b(1:length(b)-1)];
out(i)=W*b';
e = c(i)*(0.3-abs(out(i)^2));  %factor = 0.3
err(i) = e;
W=W+mu*e*b;
end
```

Test 1 Graphs Obtained:

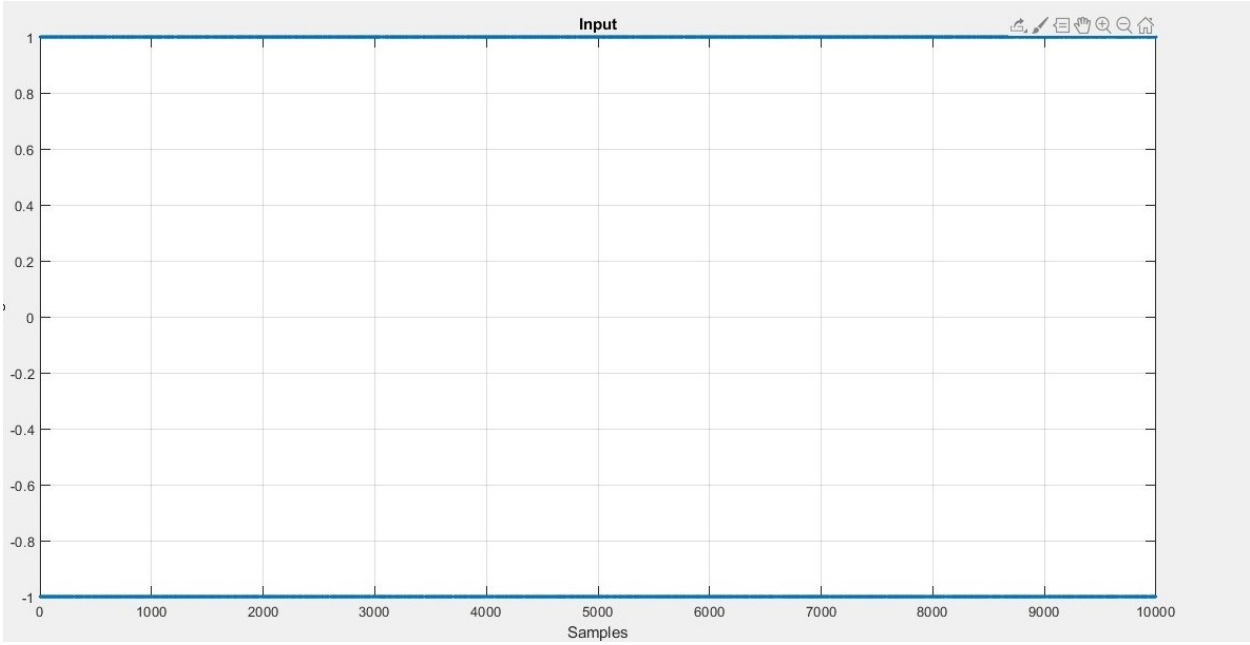

Fig 14: Input Signal Graph

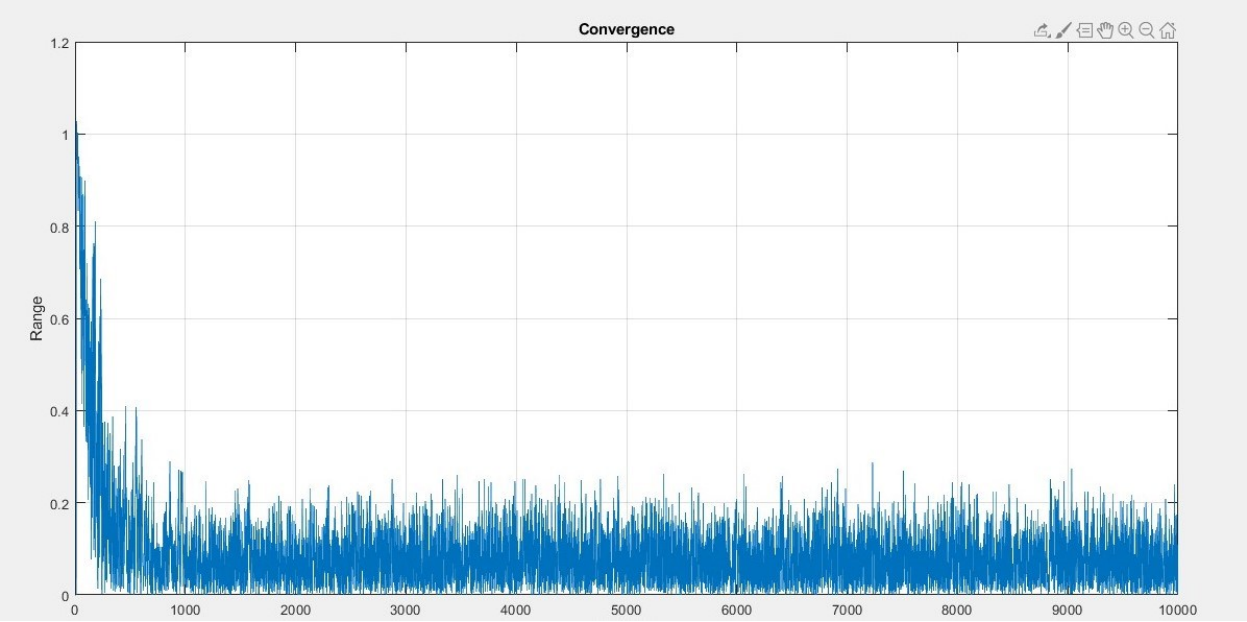

Fig 15: Convergence Graph

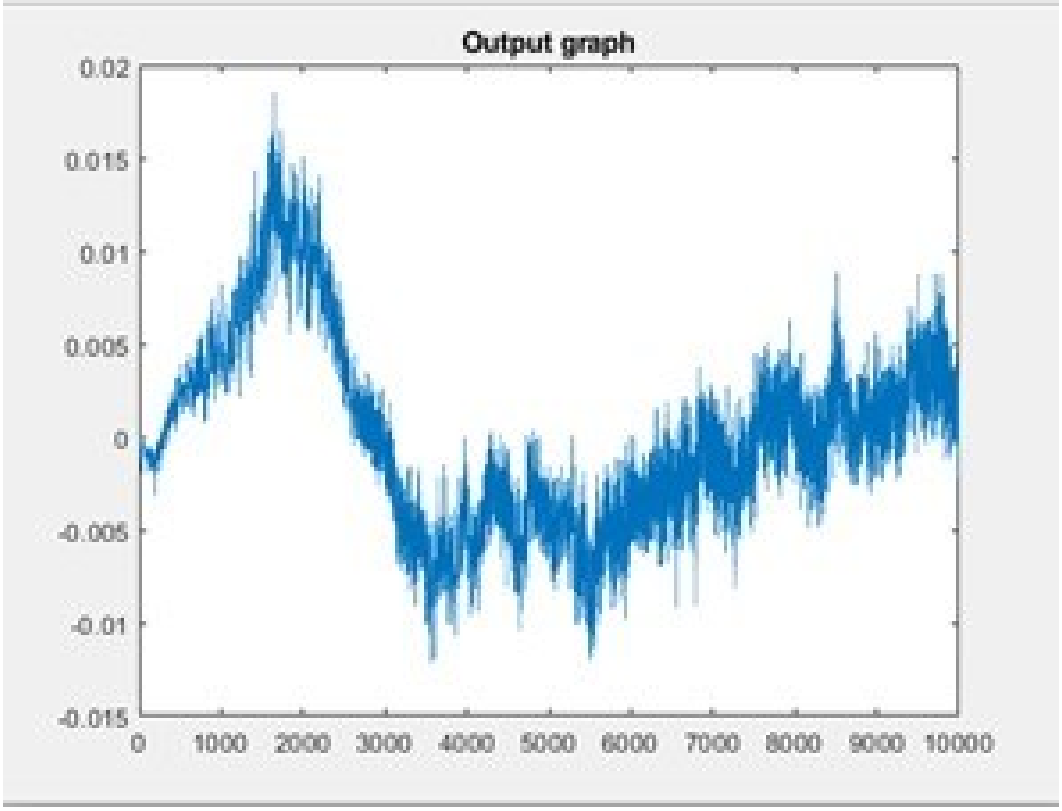

Fig 16: Output Signal Graph

Second Run: (Step Size mu=0.09)

```
% CMA Algorithm
% Given
No_of_Seq=10000;    % number of data samples
snrdb=35;     % Signal to noise ratio(dB)
snr = 10.^(snrdb/10);
h=[0.05 -0.063 0.088 -0.126 -0.25 0.9047 0.25 0 0.126 0.038 0.088];     % The given channel
Ch_length=length(h);  % length of the channel
Trans_signal=(randn(1,No_of_Seq)>0)*2-1;  % Bipolar sequence
Length_Trans_signal= length(Trans_signal); % lenght of bipolar sequence
x=filter(Ch_length,1,Trans_signal);  %channel adaptive filter delay
mu=0.009;  % Random selected the step size
N = 11;  % size of filter given in problem
W= zeros(1,N);  %Filter coefficient
v = randn(length(x),1);
v1 = sqrt(1/snr)*(sqrt(0.5)); % Additive white gaussian noise
u = x(1:No_of_Seq)+v1;
d = N-1;
b = zeros(1,N);
c=[zeros(1,d) Trans_signal(1:No_of_Seq-d)];
[out,err,W]= fundef_CMA(No_of_Seq,u,b,W,c,mu);
for i=1:length(out)
```

```
if out(i)>=0.5,f(i)=1;
else f(i)=-1;
end
end
Dec_Dev=Trans_signal(1:Length_Trans_signal-(N-1));
g=f(N:length(f));
l=0;
for j=1:length(Dec_Dev)
if Dec_Dev(j)~=g(j), l=l+1;
end
end
berc=l./(length(g));  %Bit-Error-Rate calculation
disp (berc);
plot(No_of_Seq,'.'), grid, title('Input Signals');
xlabel(‘Samples’)
ylabel(‘Range’)
plot(out), grid, title('Output Signal');
xlabel(‘Samples’)
ylabel(‘Range’)
plot(abs(err)), grid, title('Convergence');
xlabel(‘Samples’)
ylabel(‘Range’)
fvtool(W)
fvtool(err)
fvtool(in)
```

Bit Rate Error of the CMA Algorithm:

Test 1: BER=0.4993

Test 2: BER=0.5020

Test 2 Graphs Obtained:

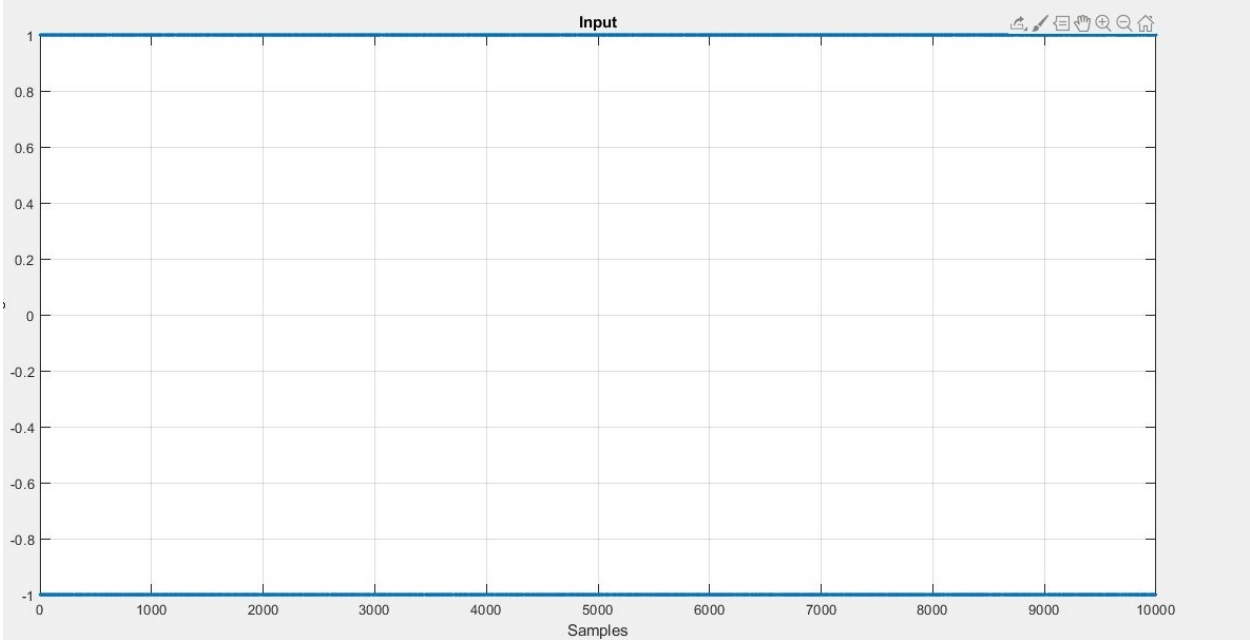

Fig 17: Input Signal Graph

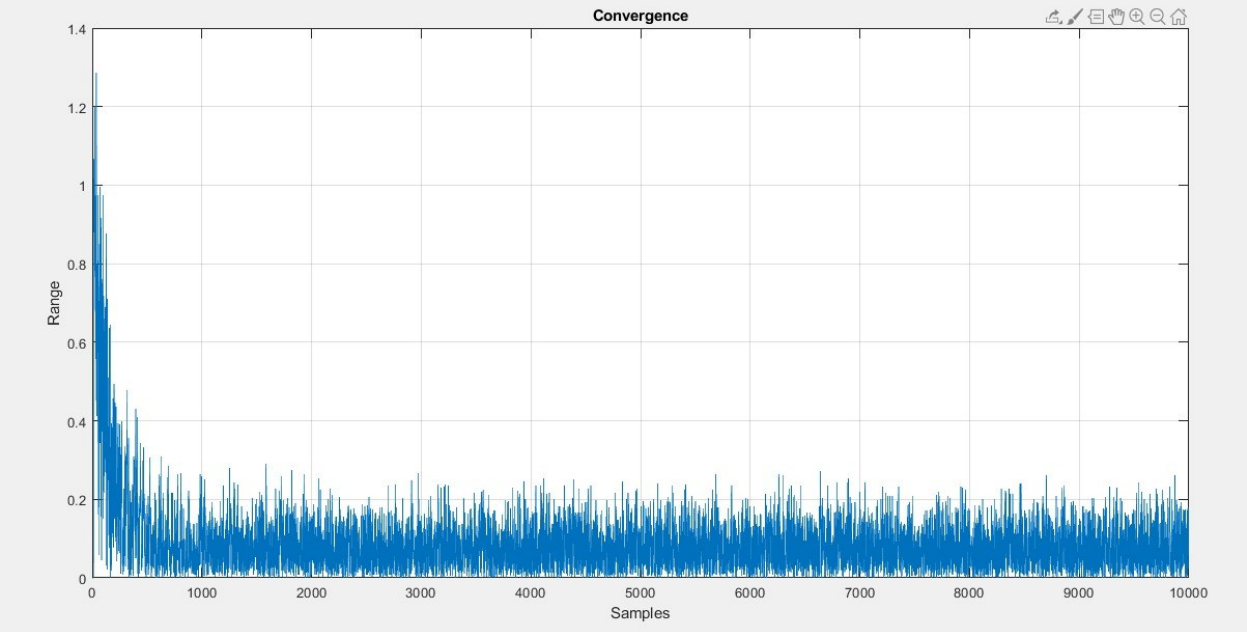

Fig 18: Convergence Graph

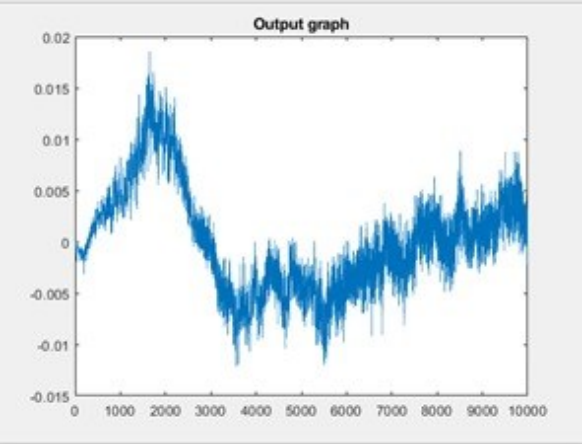

Fig 19: Output Signal Graph

Blind-Mode Adaptive Equalizer Comment :

As we analyze the above graphs, we have tested 2 different conditions with different mu values in the Blind-mode adaptive equalizer. Here no delay is used, instead of delay we calculate the disposition factor and feed it into the adaptive filter algorithm. The graphs obtained are input signal, convergence, error and output signals. By comparing the Bit Rate Error (BER) values of the two test runs of LMS we can understand that by increasing the step size value the BER value decreases. Hence, we can reproduce the original transmitted signal with higher accuracy. However, there is one condition where when a higher step size is chosen, we can exactly reproduce the input transmitted signal hence we increase the step size little by little at a gradual rate.

Conclusion:

With this project now we are able to design and test run adaptive channel equalizer using matlab for different modes of operation.

References:

[1] Farhang-Boroujeny B. 2013 Adaptive filters: theory and applications, 2nd Edition, Wiley.
[2] https://en.wikipedia.org/wiki/Adaptive_filter
[3] Malik, G. and Sappal, A. S. 2011 Adaptive Equalization Algorithms:An Overview. (IJACSA) Vol. 2, No.3.
[4] https://blog.oureducation.in/equalizer/
[5] Jalali, Sammuel, "Wireless Channel Equalization in Digital Communication Systems" (2012). *CGU Theses & Dissertations.* Paper 42. http://scholarship.claremont.edu/cgu_etd/42
[6] Zhi Ding. “Adaptive Filters for Blind Equalization.” 2000 CRC Press LLC. <http://www.engnetbase.com>