Objective-

1. Develop an equalizer using the Least Mean Square algorithm
2. Plot Bit Error Rate (BER) vs Signal to Noise Ratio (SNR) curve for BPSK, QPSK, and 64 QAM with and without equalization.
3. Study the variation of convergence time with the order of equalizer
4. Plot the frequency domain spectrum of the channel and compare it with the spectrum of the equalizer

Theory

When the modulation bandwidth exceeds the coherence bandwidth of the radio channels, ISI occurs, and the modulation pulses are spread in time into adjacent symbols. An equalizer in a receiver compensates for the average range of expected channel amplitude and delay characteristics The equalizer must be adaptive because the channel is not known and varies in time.

Adaptive Equalizer

- $\boldsymbol{y_k} = y_k\ y_{k-1}\ y_{k-2} \ldots y_{k-N}\ ^T$
- $\boldsymbol{d} = [d_1\ d_2 \ldots d_L] = training\ signal$
- $d_k = \Sigma\ w_{nk}\ y_{k-n}$
- $error\ signal = e_k = d - d_K$
- $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} + \eta.\ y_k e_k$
- k represents the time index & η denotes the rate parameter

After convergence of the algorithm, the receiver freezes the weights and use it to equalize the data

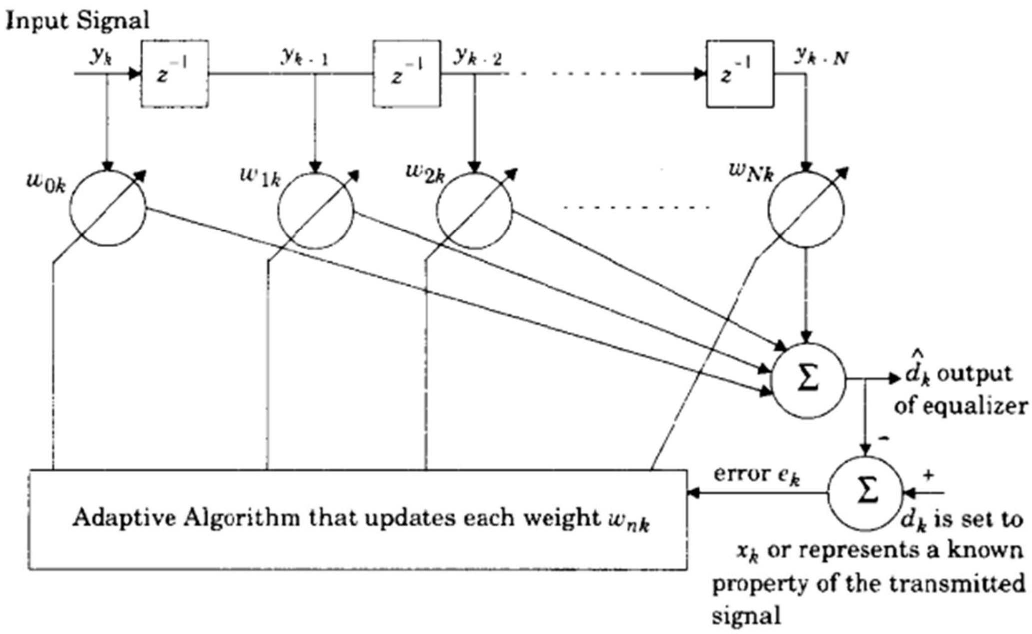


Fig 7 Basic linear equalizer during training[1]

Source Code

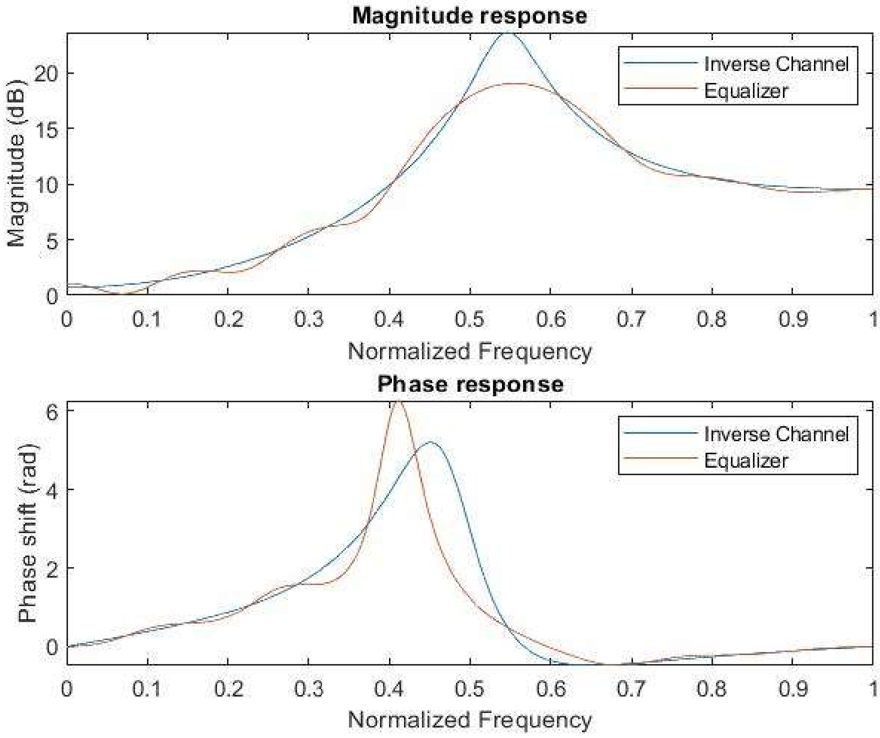
1. Develop an equalizer using the Least Mean Square algorithm
2. Plot the frequency domain spectrum of the channel and compare it with the spectrum of the equalizer

```
clc;
clear all;
close all;
% Rayleigh Distributed Channel discrete impulse response
transfer function
h=(raylrnd(3,1,4))/10;
% h = [0.9 0.3 0.5 -0.1];                          %Channel
impulse response
SNRr = 30;                                       %Noise
Signal to noise ratio
eta = 0.045;                                     %step size
order=15;                                        % Order of
the equalizer
```

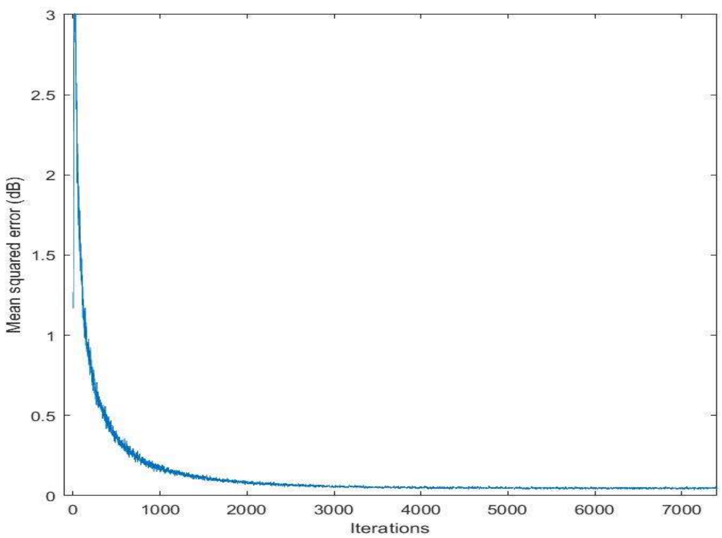
```
run = 1;
j=1;
N = 10000;                               % Number of samples
Bits = 2;                               % 2-bit for Binary
modulation
while j>4e-3
    U = zeros(1,order);                 % Input frame
    W = randn(1,order);                 % Initial Weigths
    data = randi([0 1],1,N);            % Random signal
    d = real(pskmod(data,Bits));        % BPSK Modulated signal
    r = filter(h,1,d);                  % Signal after passing
through channel
    x = awgn(r, SNRr);                  % Noisy Signal after
channel (given/input)

    for n = 1 : N
        U(1,2:end) = U(1,1:end-1);      % Sliding window
        U(1,1) = x(n);                  % Present Input

        y = (W)*U';                 % Calculating output of LMS
        e = d(n) - y;               % Instantaneous error
        W = W +  eta * e * U ;      % Weight update rule of LMS
        J(run,n) = e*e';            % Instantaneous square
error
    end
    MJ = mean(J,1);
    j = MJ(N)
    run=run+1
    if(run>500)
        break;
    end
end

CS=freqz(h);                                        % Channel
Spectrum
NF=(0:length(CS)-1)./(length(CS));                  % Normalized
Frequencies
IMR=-10*log10(real(CS).^2 + imag(CS).^2);           % Inverse
channel magnitude response (desired)
IPR=-imag(CS)./real(CS);                            % Inverse
channel phase response (desired)
ES=freqz(W);                                        % Equalizer
Spectrum
```

```
EMR=10*log10(real(ES).^2 + imag(ES).^2);     % Equalizer
magnitude response
EPR=imag(ES)./real(ES);                      % Equalizer
phase response
plot((MJ));

xlabel('Iterations');
ylabel('Mean squared error (dB)');
figure % Magnitude reponse
subplot(2,1,1)
plot(NF,IMR)
hold on
plot(NF,EMR)
hg=legend('Inverse Channel','Equalizer','Location','Best');
xlabel('Normalized Frequency');
ylabel('Magnitude (dB)');
title('Magnitude response');
subplot(2,1,2)
plot(NF, IPR)
hold on
plot(NF, EPR)
hg=legend('Inverse Channel','Equalizer','Location','Best');
xlabel('Normalized Frequency');
ylabel('Phase shift (rad)');
title('Phase response');
```

RESULT



Magnitude response
Magnitude (dB)
Normalized Frequency
Inverse Channel
Equalizer
Phase response
Phase shift (rad)
Normalized Frequency
Inverse Channel
Equalizer



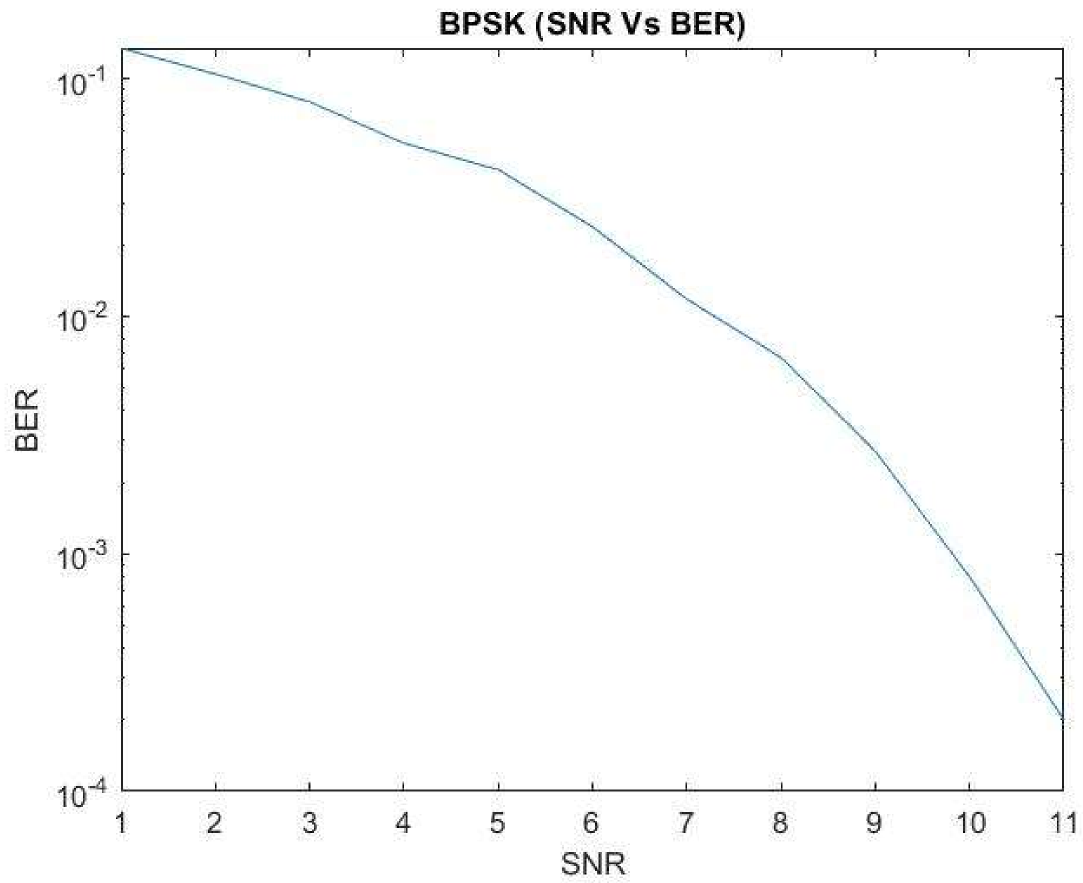
Mean squared error (dB)
Iterations

Source Code

- Plot Bit Error Rate (BER) vs Signal to Noise Ratio (SNR) curve for BPSK, QPSK, and 64 QAM with equalization.
- Study the variation of convergence time with the order of equalizer.

BPSK

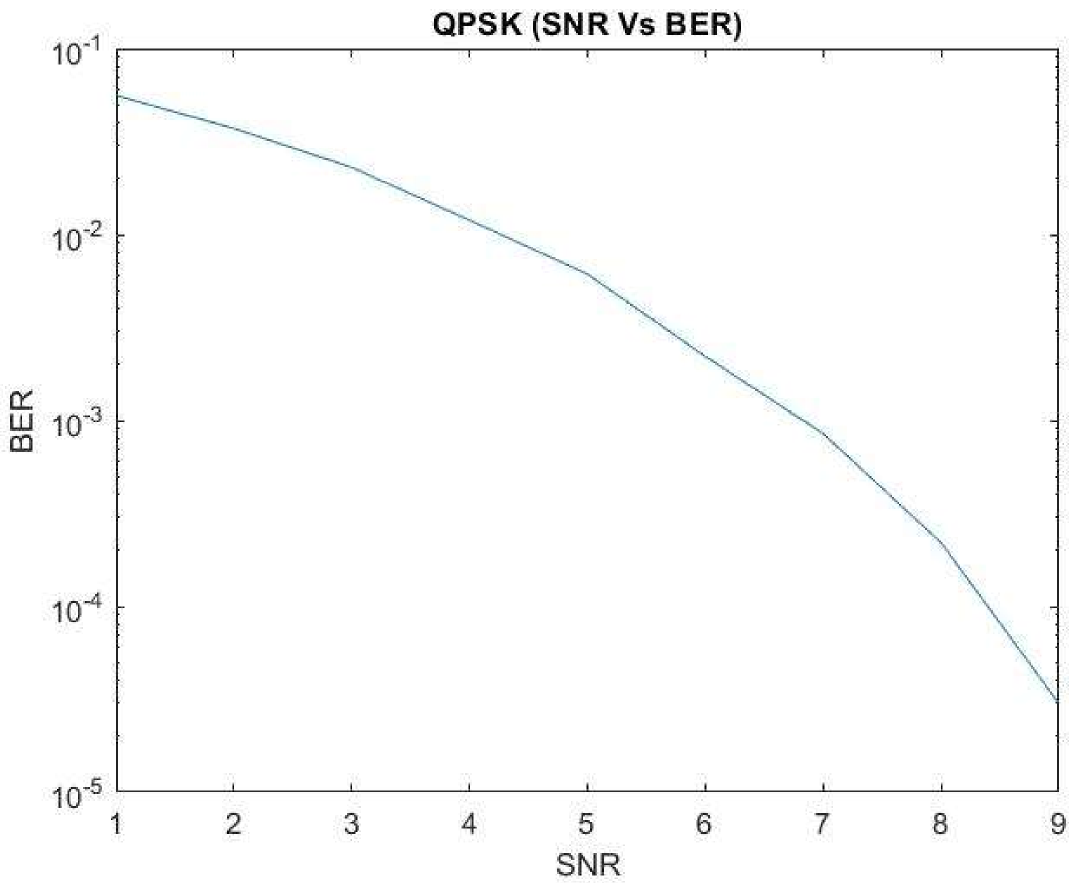
```
clc;
clear all;
close all;
h = [0.9 0.3 0.5 -0.1]; % Channel
k=1;
for SNRr=1:1:30
    N = 10000;                      % Number of samples
    Bits = 2;                      % Number of bits for modulation
(2-bit for Binary modulation)
    data = randi([0 1],N,1);           % Random signal
%    d = real(pskmod(data,Bits));       % BPSK Modulated signal
(desired/output)
    H = comm.BPSKModulator;
    s1 = real(H(data));
    s=s1';
    %r = (filter(h,1,s));                % Signal after
passing through channel
    x = awgn(s, SNRr);                  % Noisy Signal after
channel (given/input)
      y=x;
   for j=1:length(y)
    if y(j)>0
        z(j)= 1;
    else
        z(j)= -1;
    end
   end

error=length(find(z~=s));
ber(k)=error/length(y);
k=k+1;

end
    semilogy(1:1:30,ber)
```



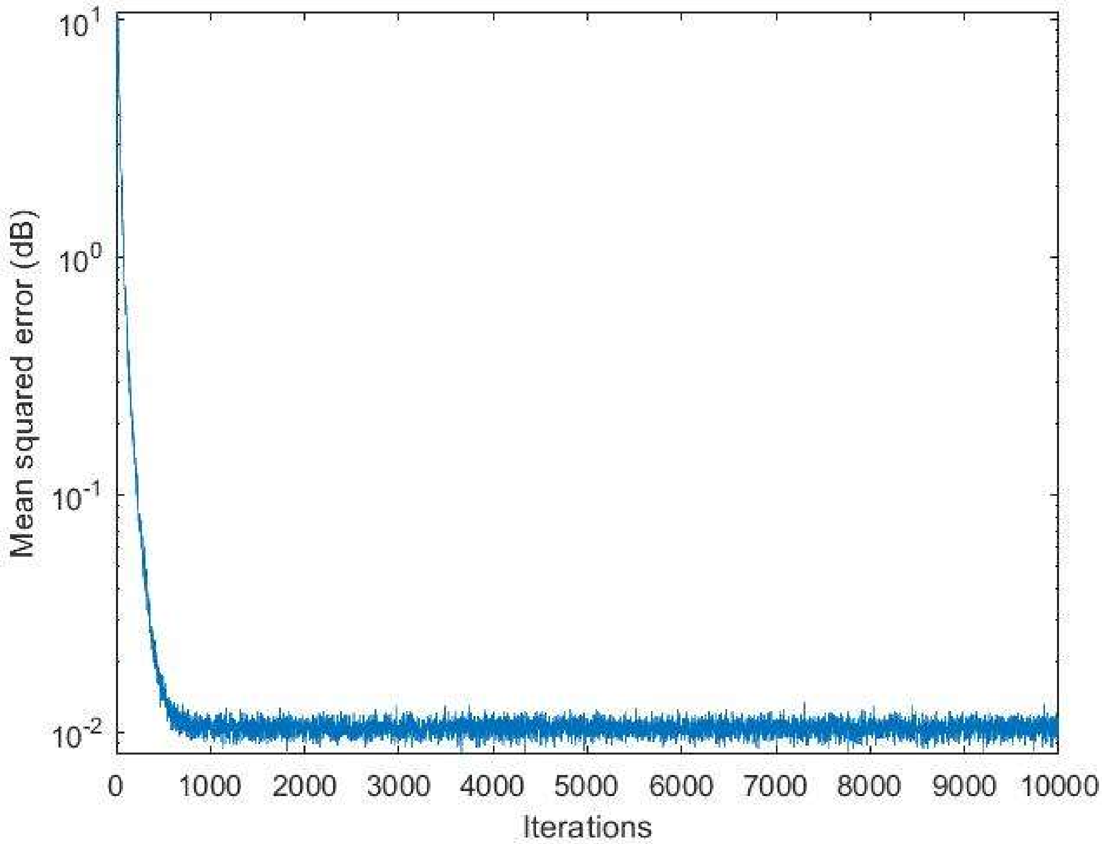
QPSK

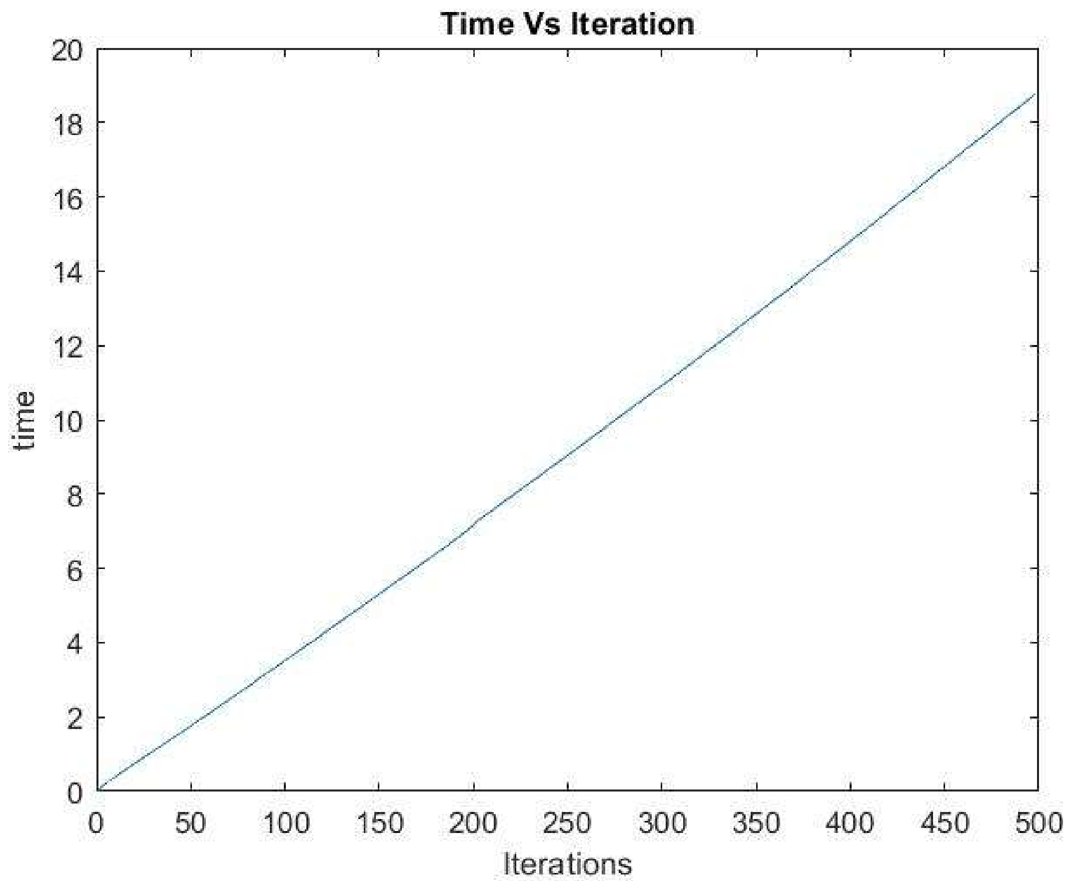
```
clear all;
close all;

%h = [0.9 0.3 0.5 -0.1];
for snrdb=1:1:30
    N = 50000;                    % Number of samples
    %Bits = 4;                    % Number of bits for
modulation (2-bit for Binary modulation)
    data = randi([0 3],N,1);           % Random signal
   % s1 = (pskmod(data,Bits));     % BPSK Modulated signal
(desired/output)
     qpskModulator = comm.QPSKModulator;
     s1 = qpskModulator(data);
     s1=(s1')*sqrt(2);
```

```
    si=sign(real(s1));
    sq=sign(imag(s1));
    %r = filter(h,1,s1);
    w=awgn(s1,snrdb);
    r1=w;
    si_=sign(real(r1));
    sq_=sign(imag(r1));
    ber1=(length(find(si~=si_)))/N;
    ber2=(length(find(sq~=sq_)))/N;
    ber(snrdb)=mean([ber1 ber2]);
end

semilogy(1:1:30, ber)
```



Convergence Time and Order Relation





Order Vs Time Comparision

- FOR ORDER 10 & ITERATION 500 TIME TAKEN TO CONVERGE IS **18.0213**
- FOR ORDER 12 & ITERATION 500 TIME TAKEN TO CONVERGE IS **18.7484**
- FOR ORDER 15 & ITERATION 500 TIME TAKEN TO CONVERGE IS **19.0374**
- FOR ORDER 20 & ITERATION 500 TIME TAKEN TO CONVERGE IS **19.1134**
- FOR ORDER 25 & ITERATION 500 TIME TAKEN TO CONVERGE IS **18.82520**
- FOR ORDER 50 & ITERATION 2 TIME TAKEN TO CONVERGE IS **0.084041**

Conclusion

- Adaptive Equalizer developed using Least Mean Algorithm trains the weights and reduces the mean square error exponentially and after some time get saturated.
- Channel follows Rayleigh distribution and therefore equalizer try to follow channel to the possible limit so that error is reduced to minimum and its effect get vanished.
- Bit Error Rate Vs Signal to noise Ratio Curve shows that with increase with SNR, BER is decreases which is obvious and this is applicable to both BPSK and QPSK difference is rate of decrease in QPSK is slow as compared to BPSK.
- With Increase of Order time taken to convergence decreases.
- As the number of iteration is increased time increases linearly

Reference

- T. S. Rappaport, "Wireless communication principles and practice," New Jersey: Prentice Hall, 1996
- Proakis, John G. Digital communications / John G. Proakis McGraw-Hill New York 1989
- Haykin, S.: Adaptive Filter Theory, 3rd edn. Prentice Hall (1996)