PROJECT REPORT

ON

EE 5362- DIGITAL COMMUNICATION

Dr. Ioannis Schizas

The University of Texas at Arlington

FALL 2017

SUBMITTED BY

Name: LOKESH VELUSWAMY

Mav Id: 1001561782

Test Scenario 1

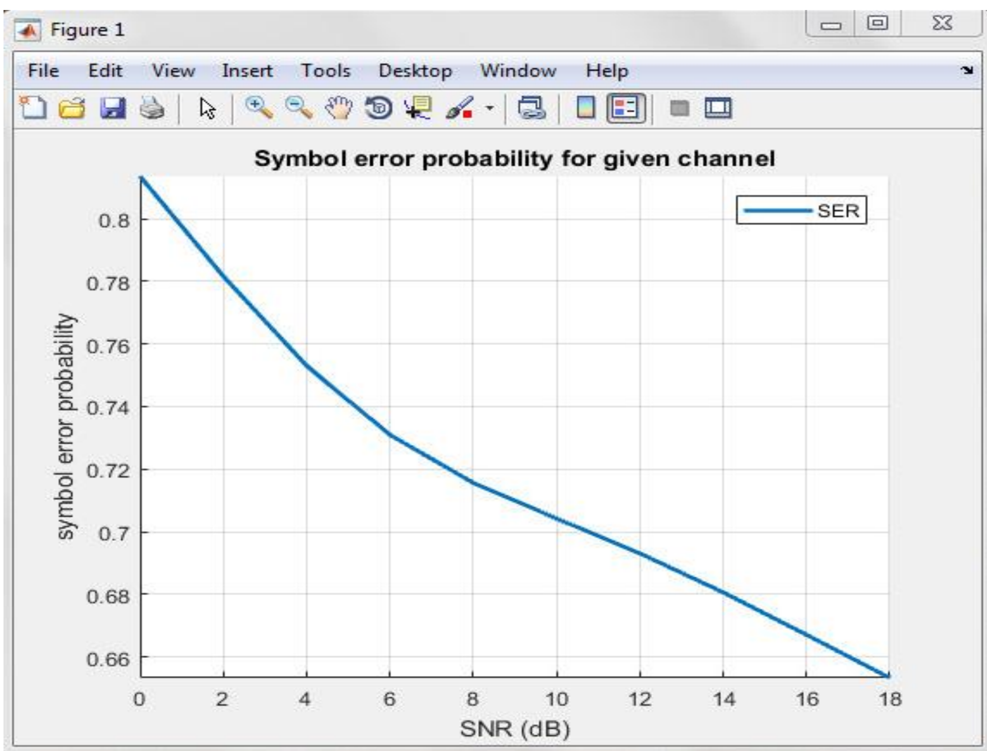
Implement the Maximum Likelihood (ML) decision rule and test its performance for the two channels. Plot the symbol error rate (SER) curve versus SNR (use semilog) for the SNR values $\gamma = 0 : 2 : 18$dB, by generating 10^7 symbols. To calculate SER count how many symbols are decided incorrectly and divide this number by the total number of transmitted symbols.

MATLAB CODE:
Channel 1

```
clear all
clc
SNR = 0:2:18;         %Signal to Noise Ratio
k = 10^7;             % Number of symbols
M = 2;
f = [0.407, 0.815, 0.407];     %Channel 1 Taps
m = randi(2,[1,k]);            %Random variables
Es = 1;
inPhase = sqrt(Es)*cos(2*pi*(m-1)/M);        %inphase component
quadrature = sqrt(Es)*sin(2*pi*(m-1)/M);     %Quadrature component
signal = inPhase+(i*(quadrature));           %Signal Generation
noisevar = 1/sqrt(2)*(randn(1,k) + i*randn(1,k));   %Noise Variance
r=conv(signal,f);                             %Convolution of signal
% Detection of Signals Calculations
for n = 1:length(SNR)
    rl = r(2:length(r)-1);
    rl = rl + 10^(-SNR(n)/20)*noisevar;
    r1 = real(rl);
    r2 = imag(rl);
    % recieved vectors Calculations

    s_cap = r1;
    s_cap1 = r2;
    for i = 1:k
        if s_cap(i) ~= 0 && s_cap(i) > 1/(sqrt(2))
            s_cap(i) = 1;
        else if s_cap(i)~= 0 && s_cap(i) < (-1/sqrt(2))
            s_cap(i) = -1;
            else
                s_cap(i) = 0;
            end
        end

         if s_cap1(i) ~= 0 && s_cap1(i) > 1/(sqrt(2))
            s_cap1(i) = 1;
        else if s_cap1(i)~= 0 && s_cap1(i) < (-1/sqrt(2))
             s_cap1(i) = -1;
            else
```

```
                s_cap1(i) = 0;
            end
         end
    end
    s_cap2 = s_cap + (i*s_cap1);
    error(1,n) = size(find([signal-s_cap2]),2);
end
SER = error/k;  %Symbol error ratio
close all;      % Plots of SNR and SER
hold all
semilogy(SNR,SER,'Linewidth',2);     % Semilog Graph
xlabel('SNR (dB)')
ylabel('symbol error probability')
legend('SER' , 'SER1');
grid on
axis tight;
title('Symbol error probability for given channel');
```

OUTPUT:



Conclusion:

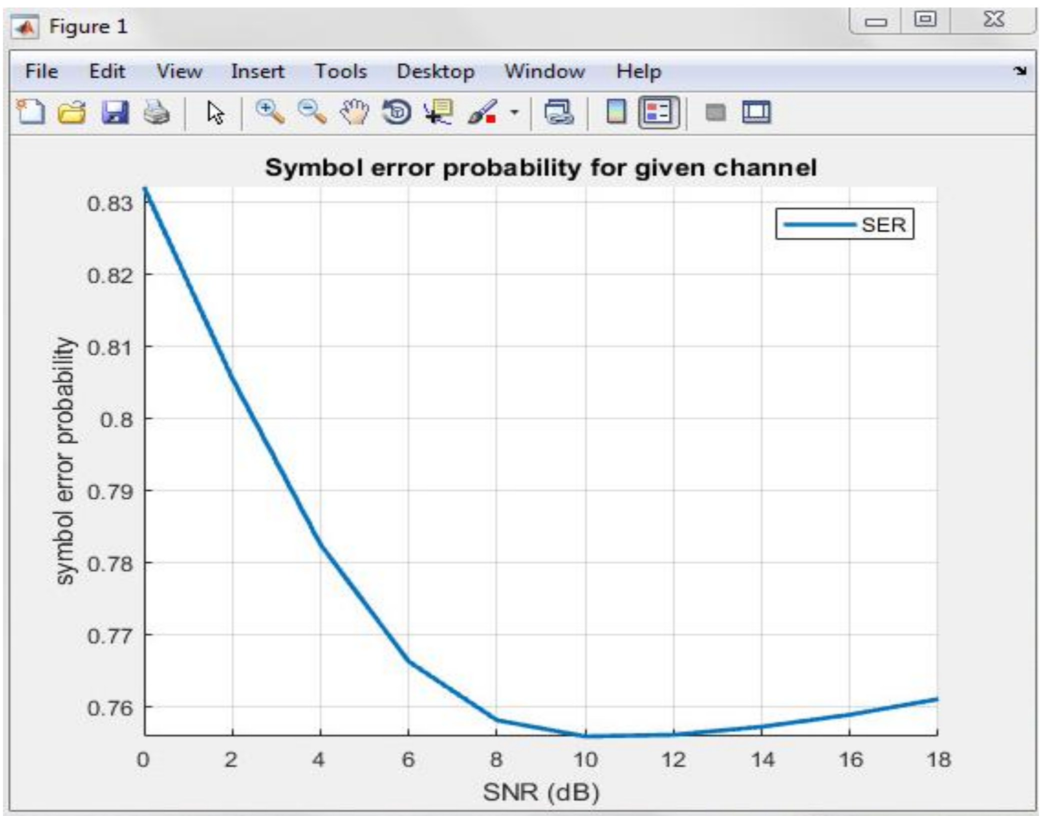
The SER tends to decrease with Gradual Slope with increase of SNR.

Channel 2

```
clear all
clc
SNR = 0:2:18;     %SNR Ratio
k = 10^7;         % Number of Symbols
M = 2;
f = [0.227, 0.46, 0.688, 0.46, 0.227];      %Channnel 2 Taps
m = randi(2,[1,k]);                          % Random Function
Generation
Es = 1;
inPhase = sqrt(Es)*cos(2*pi*(m-1)/M);        % In phase Component
Generation
quadrature = sqrt(Es)*sin(2*pi*(m-1)/M);     % Quadrature Component
Generation
signal = inPhase+(i*(quadrature));           % Signal Generation
noisevar = 1/sqrt(2)*(randn(1,k) + i*randn(1,k));  % Noise Variance
r=conv(signal,f);
% Detection Of Signals
for n = 1:length(SNR)
    rl = r(3:length(r)-2);
    rl = rl + 10^(-SNR(n)/20)*noisevar;
    r1 = real(rl);
    r2 = imag(rl);
    % recieved Signals
    s_cap = r1;
    s_cap1 = r2;
    for i = 1:k
        if s_cap(i) ~= 0 && s_cap(i) > 1/(sqrt(2))
            s_cap(i) = 1;
        else if s_cap(i)~= 0 && s_cap(i) < (-1/sqrt(2))
                s_cap(i) = -1;
            else
                s_cap(i) = 0;
            end
        end

         if s_cap1(i) ~= 0 && s_cap1(i) > 1/(sqrt(2))
            s_cap1(i) = 1;
        else if s_cap1(i)~= 0 && s_cap1(i) < (-1/sqrt(2))
                s_cap1(i) = -1;
            else
                s_cap1(i) = 0;
            end
         end
    end
    s_cap2 = s_cap + (i*s_cap1);
    error(1,n) = size(find([signal-s_cap2]),2);
end
SER = error/k;   %Signal Error Ratio
```

```
close all;          %Plot of SNR to SER
hold all
semilogy(SNR,SER,'Linewidth',2);    %Semilog Plot
xlabel('SNR (dB)')
ylabel('symbol error probability')
legend('SER')
grid on
axis tight;
title('Symbol error probability for given channel');
```

OUTPUT:



Conclusion:

The SER decreases up to a point where the SNR is 10 to 12 dB and then increases further. The system will have best performance at SNR near 10 to 15dB for the given Channel 2

Test Scenario 2

Implement the zero-forcing (ZF) linear equalizer for channel 1. Plot the SER curve versus SNR (use semilog) for the SNR values γ = 0 : 2 : 18dB, by generating 107 symbols. Obtain the SER curves for equalizer lengths 11, 21 and 31. Comment briefly on their relative performance.

MATLAB CODE

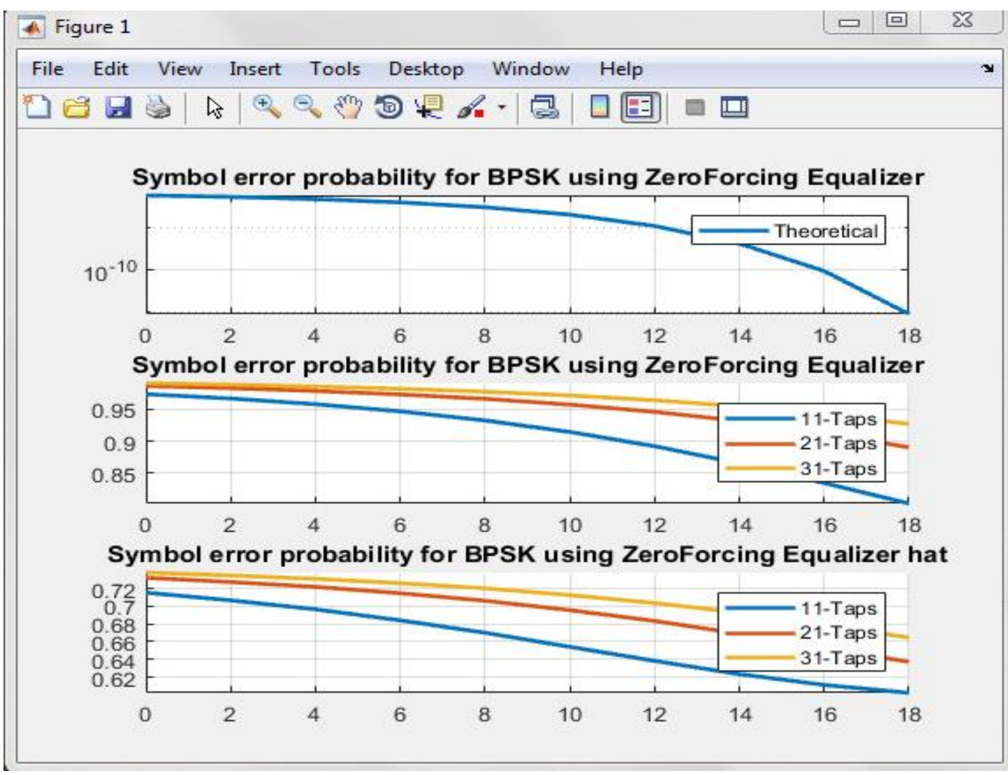
CHANNEL 1

```
clc;
clear all;
k = 10^7;         %Number of symbols
SNR = 0:2:18;     %Symbol to Noise Ratio
M = 2;
f = [0.407, 0.815, 0.407];    %Channel 1 Taps
no_taps = 3;

%  Transmitter Side Calculation

for n = 1:length(SNR)
    symbols = randi(2,[1,k]);                   %Random symbol generation.
    inphase = cos(2*pi*(symbols-1)/M);          %In Phase Component
Generation
    quadrature = sin(2*pi*(symbols-1)/M);       % Quadrature Phase
Component
    signal = inphase + 1i*quadrature;           % Signal Generation
    r = conv(signal,f);                         % Convolution Of Signal
with Channel Taps
    noise = 1/sqrt(2)*(randn(1,k+length(f)-1) +
1i*randn(1,k+length(f)-1));
    y = r + 10^(-SNR(n)/20)*(noise);            % Adding noise with 0dB
    l = length(f);
    for taps = 1:no_taps
     fm = toeplitz([f([2:end]) zeros(1,10*taps+1-l+1)], [ f([2:-1:1])
zeros(1,10*taps+1-l+1) ]);
     d  = zeros(1,10*taps+1);
     d(taps+1) = 1;
     c = [fm\d.'].';

     % Matched filter
     yfilt = conv(y,c);
     yfilt = yfilt(taps+2:end);
     yfilt = conv(yfilt,ones(1,1));           % Convolution
     filter_sampled = yfilt(1:1:k);           % Sampling at time T

        %Reciever
```

```
          s_cap = real(filter_sampled);
          s_cap1 = imag(filter_sampled);

          for i = 1:k
          if s_cap(i) ~= 0 && s_cap(i) > 1/(sqrt(2))
             s_cap(i) = 1;
          else if s_cap(i)~= 0 && s_cap(i) < (-1/sqrt(2))
                  s_cap(i) = -1;
              else
                  s_cap(i) = 0;
              end
          end

           if s_cap1(i) ~= 0 && s_cap1(i) > 1/(sqrt(2))
              s_cap1(i) = 1;
          else if s_cap1(i)~= 0 && s_cap1(i) < (-1/sqrt(2))
                  s_cap1(i) = -1;
              else
                  s_cap1(i) = 0;
              end
           end
          end
       %Reciever Hard Decision Coding
       ipHat = real(filter_sampled)>0;

       %Counting the Errors
       error1(taps,n) = size(find([signal - ipHat]),2);


      s_cap2 = s_cap + (i*s_cap1);
      error(taps,n) = size(find([signal-s_cap2]),2);
    end
end
SER = error/k;
SER1 = error1/k;

%Simulated Error Probability
SER_theoretical = 0.5*qfunc(sqrt(10.^(SNR/10)));

%Theoretical_SER

% Plot of SNR vs SER
close all
figure (1)
subplot(3,1,1);
semilogy(SNR,SER_theoretical,'Linewidth',2);
legend('Theoretical');
title('Symbol error probability for BPSK using ZeroForcing
Equalizer');
grid on
axis tight;
subplot(3,1,2);
```

```
hold all
semilogy(SNR,SER(1,:),'Linewidth',2);
semilogy(SNR,SER(2,:),'Linewidth',2);
semilogy(SNR,SER(3,:),'Linewidth',2);
legend('11-Taps','21-Taps','31-Taps');
title('Symbol error probability for BPSK using ZeroForcing
Equalizer');
grid on
axis tight;

subplot(3,1,3);
hold all
semilogy(SNR,SER1(1,:),'Linewidth',2);
semilogy(SNR,SER1(2,:),'Linewidth',2);
semilogy(SNR,SER1(3,:),'Linewidth',2);

legend('11-Taps','21-Taps','31-Taps');
title('Symbol error probability for BPSK using ZeroForcing Equalizer
hat');
grid on
axis tight;
```

OUTPUT:



Conclusion: The performance seems to decrease when we use the Zero Forcing equalizer Hat.

Test Scenario 3

Implement the linear minimum mean-square error (LMMSE) equalizer for channels 1 and 2. Plot the simulated SER semi-log curves as in Test Scenario 1 for lengths 11, 21 and 31. Comment on the difference in performance between the two channels.

MATLAB CODE

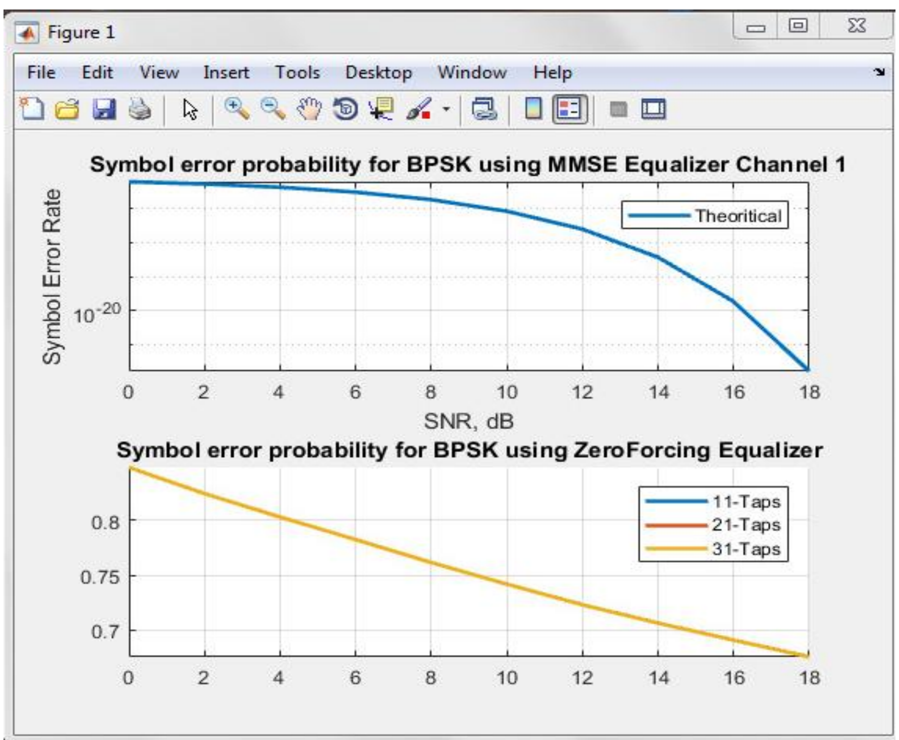
CHANNEL 1

```
clc;
clear all;
close all;
k = 10^7; %Number of symbols
no_taps = 3;
SNR = 0:2:18;
M=2;

for m=1:M   %Constellation
H(m,1) = cos(2*pi*(m-1)/M);
H(m,2) = sin(2*pi*(m-1)/M);
end

for n = 1:length(SNR)     %Transmitter
sym = randi(2,[1,k]);      %Generating N random symbols
inphase = cos(2*pi*(sym-1)/M);  %In phase Component
quadrature = sin(2*pi*(sym-1)/M);  %Quadrature Phase Component

signal = inphase+(1i*(quadrature));  %Composite Signal
f = [0.407,0.815,0.407];  %Channel 1 Taps

sig_with_sym1 = conv(inphase,f); %Convolution of random integers &
taps of channel 1
sig_with_sym2 = conv(quadrature,f);
sig_with_sym=sig_with_sym1+(1i*sig_with_sym2);
% Noise with 0db variance
noise = 1/sqrt(2)*(randn(1,k+length(f)-1) + 1i*randn(1,k+length(f)-
1));

% Addition of nosyme to channel
y = sig_with_sym + 10^(-SNR(n)/10)*(noise);
L = length(f);

% MMSE equalization
for taps = 1:no_taps
    tap = 10*taps+1;
hautocorr = conv(f,fliplr(f));
```

```
hM = toeplitz([hautocorr([3:end]) zeros(1,2*tap+1-L)], [
hautocorr([3:end]) zeros(1,2*tap+1-L) ]);
hM = hM + 1/2*10^(-SNR(n)/10)*eye(2*tap+1);
d = zeros(1,2*tap+1);
d([-1:1]+tap+1) = fliplr(f);
c_mmse  = [inv(hM)*d.'].';
% Matched filter
yfilt_mmse = conv(y,c_mmse);
yfilt_mmse = yfilt_mmse(tap+2:end);
yfilt_mmse = conv(yfilt_mmse,ones(1,1)); % convolution
filter_sampled_mmse = yfilt_mmse(1:1:k); % sampling at time T


s_cap=real(filter_sampled_mmse);
s_cap1=imag(filter_sampled_mmse);


for i = 1:k
if s_cap(i)~= 0 && s_cap(i) > (1/sqrt(2))
s_cap(i) = 1;
else if s_cap(i)~= 0 && s_cap(i) < (-1*(1/sqrt(2)))
s_cap(i) = -1;
else
s_cap(i) = 0;
end
end

if s_cap1(i)~= 0 && s_cap1(i) > (1/sqrt(2))
s_cap(i) = 1;
else if s_cap1(i)~= 0 && s_cap1(i) < (-1*(1/sqrt(2)))
s_cap1(i) = -1;
else
s_cap1(i) = 0;
    end
end
end
s_cap2=s_cap+(1i*(s_cap1)) ;
error(taps,n) = size(find([signal-s_cap2]),2);

end
end

SER = error/k; %Simulated Error Probability
SER_theoretical = 0.5*erfc(sqrt(10.^(SNR/10))); %Theoretical Symbol
Error Rate


% Plot of SNR and SER
close all;
figure (1)
subplot(2,1,1);
semilogy(SNR,SER_theoretical,'Linewidth',2);
```

```
grid on
legend('Theoritical');
xlabel('SNR, dB');
ylabel('Symbol Error Rate');
title('Symbol error probability for BPSK using MMSE Equalizer Channel
1');
axis tight;

subplot(2,1,2);
hold all
semilogy(SNR,SER(1,:),'Linewidth',2);
semilogy(SNR,SER(2,:),'Linewidth',2);
semilogy(SNR,SER(3,:),'Linewidth',2);
legend('11-Taps','21-Taps','31-Taps');
title('Symbol error probability for BPSK using ZeroForcing
Equalizer');
grid on
axis tight;
```

OUTPUT:



Conclusion: We can conclude from the above graph that the performance of the BPSK in Zero Forcing equalizer is better than that of a BPSK using a MMSE equalizer. The performance of the Zero Forcing Equalizer for different number of Taps remains more or less the same.

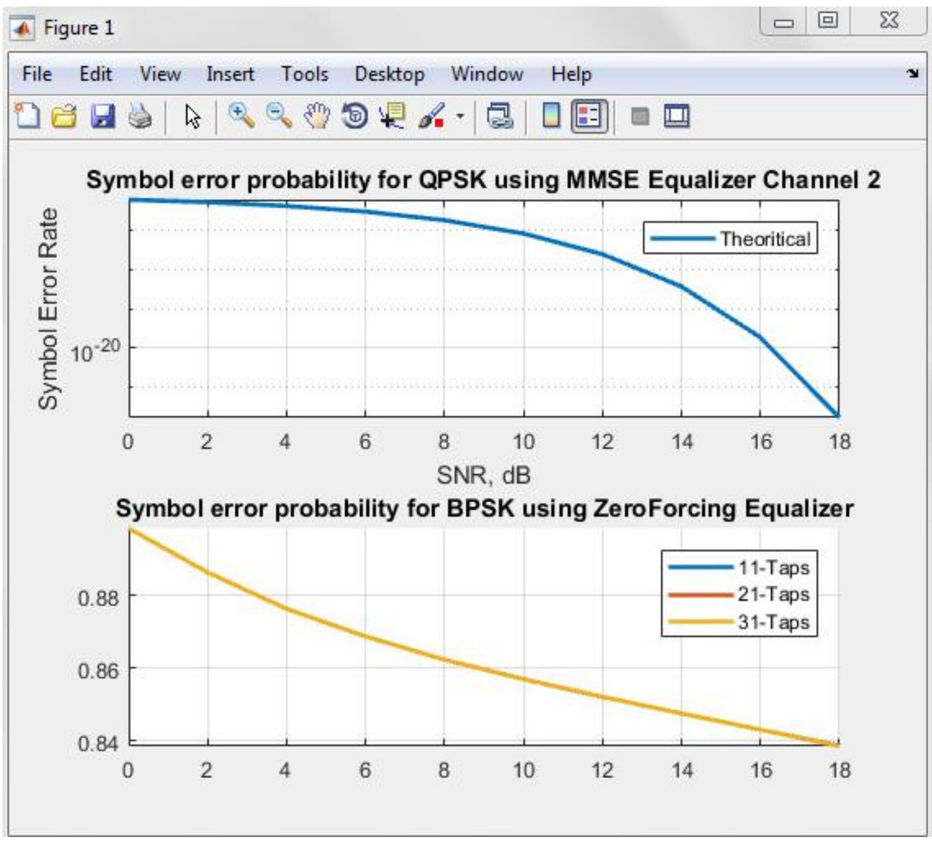
CHANNEL 2

```
clc;
clear all;
close all;
k = 10^7; %Number of symbols
no_taps = 3;
SNR = 0:2:18;
M=2;
 %Constellation
for m=1:M
    H(m,1)=cos(2*pi*(m-1)/M);
    H(m,2)=sin(2*pi*(m-1)/M);
end
 %Transmitter
for n = 1:length(SNR)
sym = randi(2,[1,k]);              %Generating N random symbols
inphase = cos(2*pi*(sym-1)/M);     %In phase Component
quadrature = sin(2*pi*(sym-1)/M); %Quadrature Component

signal = inphase+(1i*(quadrature)); %Composite Signal

f=[0.227, 0.46, 0.688, 0.46, 0.227]; %Channel 2 Taps
sig_with_sym1 = conv(inphase,f);     %Convolution of real part of
transmitted symbol & taps of channel
sig_with_sym2 = conv(quadrature,f); %Convolution of real part of
transmitted symbol & taps of channel
sig_with_sym = sig_with_sym1+(1i*sig_with_sym2);
% Nosyme with 0db variance
noise = 1/sqrt(2)*(randn(1,k+length(f)-1) + 1i*randn(1,k+length(f)-
1));

 % Addition of nosyme to channel
y = sig_with_sym + 10^(-SNR(n)/20)*(noise);
L = length(f);

%MMSE equalization
for taps = 1:no_taps
    tap = 10*taps+1;
hautocorr = conv(f,fliplr(f));
hM = toeplitz([hautocorr([5:end]) zeros(1,2*tap+1-L)], [
hautocorr([5:end]) zeros(1,2*tap+1-L) ]);
hM = hM + 1/2*10^(-SNR(n)/10)*eye(2*tap+1);
d = zeros(1,2*tap+1);
d([-2:2]+tap+1) = fliplr(f);
c_mmse  = [inv(hM)*d.'].';
%Matched filter
yfilt_mmse = conv(y,c_mmse);
yfilt_mmse = yfilt_mmse(tap+2:end);
yfilt_mmse = conv(yfilt_mmse,ones(1,1)); % convolution
```

```
filter_sampled_mmse = yfilt_mmse(1:1:k); % sampling at time T


s_cap=real(filter_sampled_mmse);
s_cap1=imag(filter_sampled_mmse);


for i = 1:k
if s_cap(i)~= 0 && s_cap(i) > (1/sqrt(2))
s_cap(i) = 1;
else if s_cap(i)~= 0 && s_cap(i) < (-1*(1/sqrt(2)))
s_cap(i) = -1;
else
s_cap(i) = 0;
end
end

if s_cap1(i)~= 0 && s_cap1(i) > (1/sqrt(2))
s_cap(i) = 1;
else if s_cap1(i)~= 0 && s_cap1(i) < (-1*(1/sqrt(2)))
s_cap1(i) = -1;
else
s_cap1(i) = 0;
    end
end
end
s_cap2=s_cap+(1i*(s_cap1)) ;
error(taps,n) = size(find([signal - s_cap2]),2);

end
end

SER = error/k;                                    %Symbol Error Rate
SER_theoretical = 0.5*erfc(sqrt(10.^(SNR/10))); %Theoretical Error
Rate

%Plot of SNR and SER
close all;
figure (1)
subplot(2,1,1);
semilogy(SNR,SER_theoretical,'Linewidth',2);
grid on
legend('Theoritical');
xlabel('SNR, dB');
ylabel('Symbol Error Rate');
title('Symbol error probability for QPSK using MMSE Equalizer Channel
2');
axis tight;

subplot(2,1,2);
hold all
semilogy(SNR,SER(1,:),'Linewidth',2);
```

```
semilogy(SNR,SER(2,:),'Linewidth',2);
semilogy(SNR,SER(3,:),'Linewidth',2);
legend('11-Taps','21-Taps','31-Taps');
title('Symbol error probability for BPSK using ZeroForcing
Equalizer');
grid on
axis tight;
```

OUTPUT:



Conclusion:

We can conclude from the above graph that the performance of the BPSK in Zero Forcing equalizer is better than that of a BPSK using a MMSE equalizer as previous but slope varies slightly and not as steep as for Channel 1. The performance of the Zero Forcing Equalizer for different number of Taps remains more or less the same.

REFERENCE:

Digital Communications by J. Proakis 5th Edition

www.wikipedia.com

www.mathworks.com