**Dated: 05/04/2017**

**EE5362**

**DIGITAL COMMUNICATION**

**Project Spring 2017**

**BY**

**DIPTIMAN HAZRA**

**ID: 1001155533**

**Test Scenario 1**: Implement the zero-forcing (ZF) linear equalizer for channel 1. Plot the bit error rate

(BER) curve versus SNR (use semilog) for the SNR values = 0 : 2 : 18dB, by generating 107 symbols.

(Count how many symbols are decided incorrectly and divide this number by the total number of transmitted symbols to estimate the BER). Compare the theoretical BER curves for equalizer lengths 11, 21 and 31. Comment briefly on their relative performance.

**Description**

A zero forcing refers to dragging down the inter symbol interference to zero in a noise free case in case the ISI is too high compared to noise. The combined response of the channel with the equalizer must satisfy Nyquist’s

Criterion.

**MATLAB CODE**

clear all

close all

clc

N = 10^7; % Number of bits or symbols

SNR = [0:2:18]; % Multiple Eb/N0 values

EL = [11 21 31]; % Length of the equaliser

for f = 1 : length(SNR)

%------------- Transmitter -----------------%

h = rand(1,N)>0.5; % Generating 0, 1 with equal probability

m = 2\*h-1; % BPSK modulation constellation(+/-1)

chan\_1 = [0.407, 0.815, 0.407];

length\_1 = length(chan\_1);

chanOut\_1 = conv(m,chan\_1);

awgn\_1 = 1/sqrt(2)\*[randn(1,N+length(chan\_1)-1)]; % AWGN for Channel 1

y\_1 = chanOut\_1 + 10^(-SNR(f)/20)\*awgn\_1;

% MMSE equalization for Channel 1

h\_1 = conv(chan\_1 , fliplr(chan\_1)); % Convolution of channel 1 with its flipped form

ell\_11 = toeplitz([h\_1([3:end]) zeros(1,EL(1)-length\_1)], [ h\_1([3:end]) zeros(1,EL(1)-length\_1) ]); % To elliptiz h1 forming a matrix

ell\_12 = toeplitz([h\_1([3:end]) zeros(1,EL(2)-length\_1)], [ h\_1([3:end]) zeros(1,EL(2)-length\_1) ]); % To elliptiz h1 forming a matrix

ell\_13 = toeplitz([h\_1([3:end]) zeros(1,EL(3)-length\_1)], [ h\_1([3:end]) zeros(1,EL(3)-length\_1) ]); % To elliptiz h1 forming a matrix

x1 = zeros(1,EL(1));

x2 = zeros(1,EL(2));

x3 = zeros(1,EL(3));

% -- -----setting half the length as 1 ----- %

x1((EL(1)+1)/2) = 1;

x2((EL(2)+1)/2) = 1;

x3((EL(3)+1)/2) = 1;

c\_mmse\_11 = [inv(ell\_11)\*x1.'].';

c\_mmse\_12 = [inv(ell\_12)\*x2.'].';

c\_mmse\_13 = [inv(ell\_13)\*x3.'].';

yFilt\_mmse\_11 = conv(y\_1,c\_mmse\_11);

yFilt\_mmse\_12 = conv(y\_1,c\_mmse\_12);

yFilt\_mmse\_13 = conv(y\_1,c\_mmse\_13);

yFilt\_mmse\_11 = yFilt\_mmse\_11(((EL(1)+3)/2):end);

yFilt\_mmse\_12 = yFilt\_mmse\_12(((EL(2)+3)/2):end);

yFilt\_mmse\_13 = yFilt\_mmse\_13(((EL(3)+3)/2):end);

yFilt\_mmse\_11 = conv(yFilt\_mmse\_11,ones(1,1)); % convolution

yFilt\_mmse\_12 = conv(yFilt\_mmse\_12,ones(1,1)); % convolution

yFilt\_mmse\_13 = conv(yFilt\_mmse\_13,ones(1,1)); % convolution

ySamp\_mmse\_11 = yFilt\_mmse\_11(1:1:N); % sampling at time T

ySamp\_mmse\_12 = yFilt\_mmse\_12(1:1:N); % sampling at time T

ySamp\_mmse\_13 = yFilt\_mmse\_13(1:1:N); % sampling at time T

ipHat\_1\_11 = real(ySamp\_mmse\_11)>0; %get the real sample

ipHat\_1\_12 = real(ySamp\_mmse\_12)>0; %get the real sample

ipHat\_1\_13 = real(ySamp\_mmse\_13)>0; %get the real sample

nErr\_2\_11(1,f) = size(find([h- ipHat\_1\_11]),2); % Number of Errors in Channel 1 for length 11

nErr\_2\_12(1,f) = size(find([h- ipHat\_1\_12]),2); % Number of Errors in Channel 1 for length 21

nErr\_2\_13(1,f) = size(find([h- ipHat\_1\_13]),2); % Number of Errors in Channel 1 for length 31

end

simser\_1\_11 = nErr\_2\_11/N; % Simulated BER for Channel 1 for length 11

simser\_1\_12 = nErr\_2\_12/N; % Simulated BER for Channel 1 for length 21

simser\_1\_13 = nErr\_2\_13/N; % Simulated BER for Channel 1 for length 31

BER = 0.5\*erfc(sqrt(10.^(SNR/10))); % Theoretical BER

% plot

figure

semilogy(SNR,BER,'b','Linewidth',1);

hold on

semilogy(SNR,simser\_1\_11(1,:),'r','Linewidth',2);

axis([0 9 10^-4 0.75]);

grid on

legend('theoretical','chn -1 Len -11');

xlabel('SNR(dB)');

ylabel('BIT ERROR PROBABILITY');

title('BER vs SNR for BPSK with equalizer length =11 in ISI with ZF equalizer');

figure

semilogy(SNR,BER,'b','Linewidth',1);

hold on

semilogy(SNR,simser\_1\_12(1,:),'m','Linewidth',2);

axis([0 9 10^-4 0.75]);

grid on

legend('theoretical','chn -1 Len -21');

xlabel('SNR(dB)');

ylabel('BIT ERROR PROBABILITY');

title('BER vs SNR for BPSK with equalizer length =21 in ISI with ZF equalizer');

figure

semilogy(SNR,BER,'b','Linewidth',1);

hold on

semilogy(SNR,simser\_1\_13(1,:),'k','Linewidth',2);

axis([0 9 10^-4 0.75]);

grid on

legend('theoretical','chn -1 Len -31');

xlabel('SNR(dB)');

ylabel('BIT ERROR PROBABILITY');

title('BER vs SNR for BPSK with equalizer length =31 in ISI with ZF equalizer');







**Conclusion:**

When the number of taps are increased, SER remains unchanged across different channels. But in zero forcing equalizer the symbol error rate or bit error rate decreases with the increase in taps.

**Test Scenario 2:** Implement the linear minimum mean-square error (LMMSE) equalizer for channels 1 and 2. Plot the simulated BER 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**

clear all;

close all;

clc;

N = 10^7; % number of bits or symbols

SNR = [0:2:18]; % multiple SNR values

EL = [11 21 31]; % Length of equalizer (2\*k+1)

for f = 1 : length(SNR)

% Transmitter

p = rand(1,N)>0.5; % Generating 0,1 with equal probability

signal = 2\*p-1; % BPSK modulation

% Channel model for channel (1)

nTap = 2 ; % channel 1

channel1 = [0.407, 0.815, 0.407];

Length\_1 = length(channel1);

%channel model for channel 2

nTap = 3; % channel 3

channel2 = [0.227, 0.46, 0.688, 0.46, 0.227];

Length\_2 = length(channel2);

% Convoluting the signal s with both the channels

channelOut1 = conv(signal,channel1);

channelOut2 = conv(signal,channel2);

% AWGN generation for both the channels

n1 = 1/sqrt(2)\*[randn(1,N+length(channel1)-1) + j\*randn(1,N+length(channel1)-1)]; % AWGN generation for channel (a)

n2 = 1/sqrt(2)\*[randn(1,N+length(channel2)-1) + j\*randn(1,N+length(channel2)-1)]; % AWGN generation for channel (b)

% Input to the Equaliser ie Noise Addition

Input1 = channelOut1 + 10^(-SNR(f)/20)\*n1; % Adding AWGN with signal for channel (1)

Input2 = channelOut2 + 10^(-SNR(f)/20)\*n2; % Adding AWGN with signal for channel (2)

% mmse equalization for Channel (1)

AutoCorr = conv(channel1 , fliplr(channel1)); % Convolution of channel (1) with its flipped form

Corr1 = toeplitz([AutoCorr([3:end]) zeros(1,EL(1)-Length\_1)], [ AutoCorr([3:end]) zeros(1,EL(1)-Length\_1) ]); % To elliptiz haCorr forming a matrix

Corr2 = toeplitz([AutoCorr([3:end]) zeros(1,EL(2)-Length\_1)], [ AutoCorr([3:end]) zeros(1,EL(2)-Length\_1) ]);

Corr3 = toeplitz([AutoCorr([3:end]) zeros(1,EL(3)-Length\_1)], [ AutoCorr([3:end]) zeros(1,EL(3)-Length\_1) ]);

Corr1 = Corr1 + 1/2\*10^(-SNR(f)/10)\*eye(EL(1)); % Adding SNR for length = 11

Corr2 = Corr2 + 1/2\*10^(-SNR(f)/10)\*eye(EL(2)); % Adding SNR for length = 21

Corr3 = Corr3 + 1/2\*10^(-SNR(f)/10)\*eye(EL(3)); % Adding SNR for length = 31

da1 = zeros(1,EL(1)); % Matrix of 0 1XP

da2 = zeros(1,EL(2));

da3 = zeros(1,EL(3));

da1([-1:1]+((EL(1)+1)/2)) = fliplr(channel1);

da2([-1:1]+((EL(2)+1)/2)) = fliplr(channel1);

da3([-1:1]+((EL(3)+1)/2)) = fliplr(channel1);

corr\_mmse\_a1 = [inv(Corr1)\*da1.'].'; % Inverse of channel matrix

corr\_mmse\_a2 = [inv(Corr2)\*da2.'].'; % Inverse of channel matrix

corr\_mmse\_a3 = [inv(Corr3)\*da3.'].'; % Inverse of channel matrix

Filter\_mmse\_a1 = conv(Input1,corr\_mmse\_a1);

Filter\_mmse\_a2 = conv(Input1,corr\_mmse\_a2);

Filter\_mmse\_a3 = conv(Input1,corr\_mmse\_a3);

Filter\_mmse\_a1 = Filter\_mmse\_a1(((EL(1)+3)/2):end);

Filter\_mmse\_a2 = Filter\_mmse\_a2(((EL(2)+3)/2):end);

Filter\_mmse\_a3 = Filter\_mmse\_a3(((EL(3)+3)/2):end);

Filter\_mmse\_a1 = conv(Filter\_mmse\_a1 ,ones(1,1)); % Convolution for length = 11

Filter\_mmse\_a2 = conv(Filter\_mmse\_a2 ,ones(1,1)); % Convolution for length = 21

Filter\_mmse\_a3 = conv(Filter\_mmse\_a3 ,ones(1,1)); % Convolution for length = 31

Sample\_mmse\_a1 = Filter\_mmse\_a1 (1:1:N); % Sampling at time T

Sample\_mmse\_a2 = Filter\_mmse\_a2 (1:1:N);

Sample\_mmse\_a3 = Filter\_mmse\_a3 (1:1:N);

% mmse equalization for Channel (2)

AutoCorr1 = conv(channel2 , fliplr(channel2)); % Convolution of channel (2) with its flipped form

Corrb1 = toeplitz([AutoCorr1([5:end]) zeros(1,EL(1)-Length\_2)], [ AutoCorr1([5:end]) zeros(1,EL(1)-Length\_2) ]); % To elliptiz hbCorr forming a matrix for length = 11

Corrb2 = toeplitz([AutoCorr1([5:end]) zeros(1,EL(2)-Length\_2)], [ AutoCorr1([5:end]) zeros(1,EL(2)-Length\_2) ]); % To elliptiz hbCorr forming a matrix for length = 21

Corrb3 = toeplitz([AutoCorr1([5:end]) zeros(1,EL(3)-Length\_2)], [ AutoCorr1([5:end]) zeros(1,EL(3)-Length\_2) ]); % To elliptiz hbCorr forming a matrix for length = 31

Corrb1 = Corrb1 + 1/2\*10^(-SNR(f)/10)\*eye(EL(1)); % Adding SNR for length = 11

Corrb2 = Corrb2 + 1/2\*10^(-SNR(f)/10)\*eye(EL(2)); % Adding SNR for length = 21

Corrb3 = Corrb3 + 1/2\*10^(-SNR(f)/10)\*eye(EL(3)); % Adding SNR for length = 31

db1 = zeros(1,EL(1)); % Matrix of 0 1XP

db2 = zeros(1,EL(2));

db3 = zeros(1,EL(3));

db1([-2:2]+((EL(1)+1)/2)) = fliplr(channel2);

db2([-2:2]+((EL(2)+1)/2)) = fliplr(channel2);

db3([-2:2]+((EL(3)+1)/2)) = fliplr(channel2);

corr\_mmse\_b1 = [inv(Corrb1)\*db1.'].';

corr\_mmse\_b2 = [inv(Corrb2)\*db2.'].';

corr\_mmse\_b3 = [inv(Corrb3)\*db3.'].';

Filter\_mmse\_b1 = conv(Input2,corr\_mmse\_b1);

Filter\_mmse\_b2 = conv(Input2,corr\_mmse\_b2);

Filter\_mmse\_b3 = conv(Input2,corr\_mmse\_b3);

Filter\_mmse\_b1 = Filter\_mmse\_b1(((EL(1)+3)/2):end);

Filter\_mmse\_b2 = Filter\_mmse\_b2(((EL(2)+3)/2):end);

Filter\_mmse\_b3 = Filter\_mmse\_b3(((EL(3)+3)/2):end);

Filter\_mmse\_b1 = conv(Filter\_mmse\_b1,ones(1,1)); % Convolution for length = 11

Filter\_mmse\_b2 = conv(Filter\_mmse\_b2,ones(1,1)); % Convolution for length = 21

Filter\_mmse\_b3 = conv(Filter\_mmse\_b3,ones(1,1)); % Convolution for length = 31

Sample\_mmse\_b1 = Filter\_mmse\_b1 (1:1:N); % Sampling at time T for length = 11

Sample\_mmse\_b2 = Filter\_mmse\_b2 (1:1:N); % Sampling at time T for length = 21

Sample\_mmse\_b3 = Filter\_mmse\_b3 (1:1:N); % Sampling at time T for length = 31

% Hard Decision decoding at Receiver

p\_mmse\_a1 = real(Sample\_mmse\_a1) > 0;

p\_mmse\_a2 = real(Sample\_mmse\_a2) > 0;

p\_mmse\_a3 = real(Sample\_mmse\_a3) > 0;

p\_mmse\_b1 = real(Sample\_mmse\_b1) > 0;

p\_mmse\_b2 = real(Sample\_mmse\_b2) > 0;

p\_mmse\_b3 = real(Sample\_mmse\_b3) > 0;

% Counting the errors

Error\_mmse\_a1(1,f) = size(find([p- p\_mmse\_a1]),2);

Error\_mmse\_a2(1,f) = size(find([p- p\_mmse\_a2]),2);

Error\_mmse\_a3(1,f) = size(find([p- p\_mmse\_a3]),2);

Error\_mmse\_b1(1,f) = size(find([p- p\_mmse\_b1]),2);

Error\_mmse\_b2(1,f) = size(find([p- p\_mmse\_b2]),2);

Error\_mmse\_b3(1,f) = size(find([p- p\_mmse\_b3]),2);

end

ber\_mmse\_a1 = Error\_mmse\_a1/N; % Simulated Bit Error Rate

ber\_mmse\_a2 = Error\_mmse\_a2/N; % Simulated Bit Error Rate

ber\_mmse\_a3 = Error\_mmse\_a3/N; % Simulated Bit Error Rate

ber\_mmse\_b1 = Error\_mmse\_b1/N; % Simulated Bit Error Rate

ber\_mmse\_b2 = Error\_mmse\_b2/N; % Simulated Bit Error Rate

ber\_mmse\_b3 = Error\_mmse\_b3/N; % Simulated Bit Error Rate

theory\_ber = 0.5\*erfc(sqrt(10.^(SNR/10))); % theoretical ber

% plot

figure

semilogy(SNR,theory\_ber,'k','Linewidth',2);

hold on

semilogy(SNR,ber\_mmse\_a1(1,:),'m','Linewidth',2);

semilogy(SNR,ber\_mmse\_b1(1,:),'r','Linewidth',2);

axis([0 18 10^-6 0.5]);

grid on

legend('theoretical', 'channel a','channel b');

xlabel('SNR(dB)');

ylabel('BIT ERROR RATE');

title('BER vs SNR for LMMSE with equalizer length = 11 for channel a and b');

figure

semilogy(SNR,theory\_ber,'k','Linewidth',2);

hold on

semilogy(SNR,ber\_mmse\_a2(1,:),'m','Linewidth',2);

semilogy(SNR,ber\_mmse\_b2(1,:),'r','Linewidth',2);

axis([0 18 10^-6 0.5]);

grid on

legend('theoretical', 'channel a','channel b');

xlabel('SNR(dB)');

ylabel('BIT ERROR RATE');

title('BER vs SNR for LMMSE with equalizer length = 21 for channel a and b');

figure

semilogy(SNR,theory\_ber,'k','Linewidth',2);

hold on

semilogy(SNR,ber\_mmse\_a3(1,:),'m','Linewidth',2);

semilogy(SNR,ber\_mmse\_b3(1,:),'r','Linewidth',2);

axis([0 18 10^-6 0.5]);

grid on

legend('theoretical', 'channel a','channel b');

xlabel('SNR(dB)');

ylabel('BIT ERROR RATE');

title('BER vs SNR for LMMSE with equalizer length = 31 for channel a and b');







**Conclusion**

Comparing the three graphs we can say that the performance of channel 1 is better than channel 2 for any equalizer lengths since the BER is increasing for channel 1 graph while for channel 2 it’s a dip.

**Test Scenario 3**: Implement the ML decision rule and test its performance for the two channels (bypassing Equalization ). Plot the symbol error rate (SER) curve versus SNR (use semilog) for the SNR values = 0 :2 : 18dB, by generating 107 symbols. Compare the BER curves with the ones you obtained in Test Scenario 1 and Test Scenario 2.

**MATLAB CODE**

close all;

clear all;

clc;

k = 10^7;

SNR\_db = 0:2:18; %SNR in db

M=2; %Constellation size

%H for detection

for m = 1:M

H(m,1) = cos(2\*pi\*m/M);

H(m,2) = sin(2\*pi\*m/M);

end

for i =1:length(SNR\_db)

m = randi(2,1,k); %generate random integer

s1 = cos(2\*pi\*m/M); %chosse corresponding constellation

s2 = sin(2\*pi\*m/M);

Es = 6\*10^(SNR\_db(i)/10);

r\_1 = (sqrt(Es)\*s1+randn(1,k))';

r\_2 = (sqrt(Es)\*s2+randn(1,k))';

r = [r\_1,r\_2]'; %detection

[y,mhat] = max(H\*r,[],1);

ps(i) = 1-length(find(mhat-m==0))/k;

end

semilogy(SNR\_db,ps,'\*k--','LineWidth',1);

hold on;

axis([0 6 10^-6 10^-1])

grid on;

legend('simulation')

xlabel('SNR in db')

ylabel('ps: sysmbol error prob')

