Eye diagram

```
close all;
clear all;
fc = 5000; %cut-off frequency
fs=1e6; % sampling frequency
bits = 5000; %no. of bits in sequence
time = (1:bits)*10e-9;
a = randi([0 1],1,bits);
for j=1:bits
 for i=((j-1)*100+1):(j*100)
 if(a(1,j)==0)
 polar_signal(1,i)=-1;
 else
 polar_signal(1,i)=1;
 end
 end
end
subplot(2,1,1)
plot(time,polar_signal(1:bits));
title('POLAR SIGNAL')
ylim([-1.5 1.5]);
[N,D] = butter(2,fc/(fs/2)); %low pass BUTTERWORTH filter
filtered_signal = filter(N,D,polar_signal);
subplot(2,1,2)
plot(time,filtered_signal(1:bits));
title('FILTERED SIGNAL');
ylim([-1.5 1.5]);
figure(2)
k=1;
for(s=1:1:bits/4)
 for p=1:1:300
 r(:,p) = filtered_signal(:,k);
 k=k+1;
 end
 subplot(2,1,1)
 plot(time(1:300),r)
 xlim([time(1) time(200)]);
 title('EYE DIAGRAM FOR FILTERED SIGNAL WITHOUT NOISE');
 hold on
 grid on
end
received_signal = awgn(filtered_signal,25); %AWGN added corresponding to
SNR=25dB
k=1;
for(s=1:1:bits/4)
 for p=1:1:300
 r(:,p) = received_signal(:,k);
 k=k+1;
 end
 subplot(2,1,2)
 plot(time(1:300),r)
 xlim([time(1) time(200)]);
 title('EYE DIAGRAM FOR FILTERED SIGNAL WITH NOISE');
 hold on
 grid on
end
```

QAM

```
M = 16;      % No of levels
k = log2(M);    % No of bits required to represent one symbol
n = 3000;       % Total number of bits in sequence
sps = 1;        % symbols per sample
rng default;              % seed to generate random numbers
data = randi([0, 1], n, 1);      % generate random bits 0s and 1s of
dimension n * 1
stem(data(1:40), 'filled');
title('Input Data');
xlabel('Bit');
ylabel('Binary Levels');
symbols = bit2int(data, k);   % generates symbols in group of k and converts
them to integer
stem(symbols(1:40));
title('Input Symbols');
xlabel('Symbol');
ylabel('Symbol Level');
modulated = qammod(symbols,M,'bin');   % binary modulation
scatterplot(modulated,1,0,'w.');
fs = 500000;      % sampling frequency
fc = 2000;        % carrier frequency
qamMod = [];
t_prev = 0;
for i=1:1:length(modulated)
    Ai = real(modulated(i));
    Aq = imag(modulated(i));
    t = t_prev:1/fs:t_prev+(1/fs)*399;
    t_prev = t_prev + (1/fs)*400;
    qamMod = [qamMod Ai*cos(2*pi*fc*t)+Aq*sin(2*pi*fc*t)];
end
t_qam = 0:1/fs:1/fs*(300000-1);

plot(t_qam(1:3600), qamMod(1:3600));
xlabel('Time');
ylabel('QAM Modulated Signal');
EbNo = 10;
snr = EbNo+10*log10(k)-10*log10(sps);
receivedSignal = awgn(modulated,snr,'measured');
recievedConstelletion = scatterplot(receivedSignal,1,0,'r.');
hold on;
scatterplot(modulated,1,0,'w*',recievedConstelletion);
% Convert binary data to symbol data
symbols = bi2de(reshape(data, 4, length(data)/4).', 'left-msb');

% Generate the QAM signal
constellation = [-3+3i, -3+1i, -3-3i, -3-1i, -1+3i, -1+1i, -1-3i, -1-1i,
3+3i, 3+1i, 3-3i, 3-1i, 1+3i, 1+1i, 1-3i, 1-1i];
qam_signal = constellation(symbols+1);
scatterplot(constellation)
title('Signal Constellation for 16 QAM');
function symbols = bit2int(data, k)
    n = numel(data);
    symbols = zeros(1, floor(n/k));
    for i = 1:floor(n/k)
```

```
        binary_vector = data((i-1)*k+1:i*k);
        decimal_value = 0;
        for j = 1:k
            decimal_value = decimal_value + binary_vector(j) * 2^(k-j);
        end
        symbols(i) = decimal_value;
    end
end
```

Error_Controle coding

```
% Extension of (7,4) block to (8,4) block
% Given H Matrix
H = [1 0 1 1 1 0 0 0;
     1 1 0 1 0 1 0 0;
     0 1 1 1 0 0 1 0;
     1 1 1 0 0 0 0 1];
disp("H Matrix is")
disp(H)
k = 4; % Represent information bits
n = 8 % Represent total no of bits
% Redundant or parity bit P=n-k
% Parity=4 bits
% We need to generate a G Matrix
P = H';
L = P;
L((5:8), : ) = [];%removing the rows from 5 - 8
I = eye(k); %Generate identity matrix of order 4*4
G = [I L];%concatenate identity matrix with L
disp("G Matrix is");
disp(G)
% Generating u matrix of all 4 bit possible sequences
u=[0 0 0 0
 0 0 0 1
 0 0 1 0
 0 0 1 1
 0 1 0 0
 0 1 0 1
 0 1 1 0
 0 1 1 1
 1 0 0 0
 1 0 0 1
 1 0 1 0
 1 0 1 1
 1 1 0 0
 1 1 0 1
 1 1 1 0
 1 1 1 1];
% CodeWords
c = mod(u * G, 2); % Gives the reminder when divisible by 2
disp("Code Words");
disp(c);
%we want to check this recieved codeword
r = [0 0 1 0 0 0 0 0]
% Generate Syndrome
ht = H';
```

```
s = mod(r * ht, 2);
syn = s';
disp("Syndrome");
disp(syn);
correct = r;
for i = 1 : 1 : size(ht) % Size(ht) gives the max length of row or column
if(ht(i,1:4)==s)
correct(i) = 1-correct(i);
break;
end
end
fprintf('The Recieved Codeword is:')
disp(r)
fprintf('The Error is in bit:')
disp(i)
fprintf('The Corrected Codeword is :')
disp(correct)
```