1st assignment for 3rd topic of project

Name: Samaun Jahan Rodro

Date: 14/01/2022

Version: 1

Course Name: Nonlinear Analysis for Optical Fiber Communications

Student ID: 28G21131

Word Count: 1409

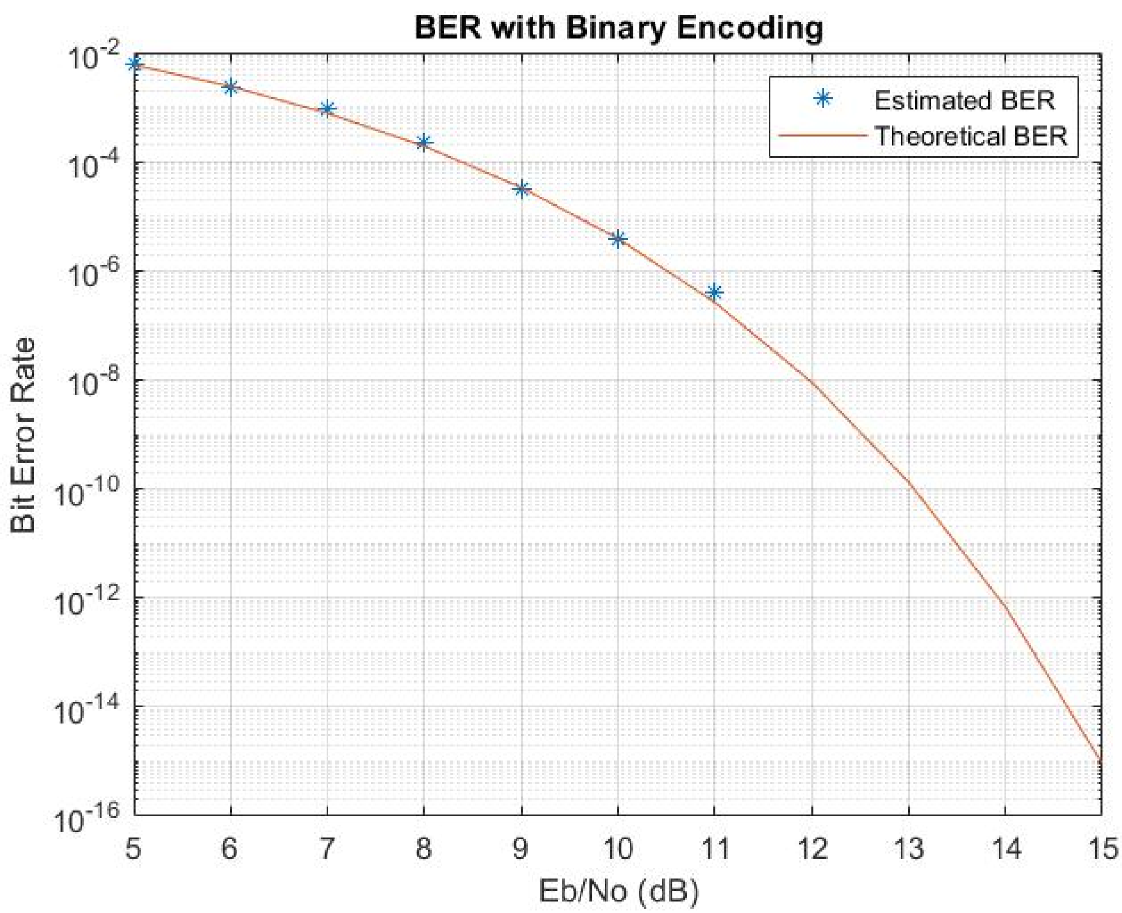
(i) See the following references and plot the bit error rate (BER) curves of QPSK and 16QAM with simulation and theoretical value.
https://www.mathworks.com/help/comm/ref/biterr.html

Solution:

16 QAM:

```
M = 4;                 % Modulation order
k = log2(M);               % Bits per symbol
EbNoVec = (5:15)';         % Eb/No values (dB)
%EbNoVec=0:1:10;
numSymPerFrame = 100;      % Number of QAM symbols per frame

berEst = zeros(size(EbNoVec));

for n = 1:length(EbNoVec)
    % Convert Eb/No to SNR
    snrdB = EbNoVec(n) + 10*log10(k);
    % Reset the error and bit counters
    numErrs = 0;
    numBits = 0;

    while numErrs < 200 && numBits < 1e7
        % Generate binary data and convert to symbols
        dataIn = randi([0 1],numSymPerFrame,k);
        dataSym = bi2de(dataIn);


        % QAM modulate using 'Gray' symbol mapping
        %txSig = qammod(dataSym,M);
        txSig = qammod(dataSym,M,'bin');


        % Pass through AWGN channel
        rxSig = awgn(txSig,snrdB,'measured');

        % Demodulate the noisy signal
        %rxSym = qamdemod(rxSig,M);
        rxSym = qamdemod(rxSig,M,'bin');
        % Convert received symbols to bits
        dataOut = de2bi(rxSym,k);


        % Calculate the number of bit errors
        nErrors = biterr(dataIn,dataOut);

        % Increment the error and bit counters
        numErrs = numErrs + nErrors;
        numBits = numBits + numSymPerFrame*k;
    end

    % Estimate the BER
    berEst(n) = numErrs/numBits;
end

berTheory = berawgn(EbNoVec,'qam',M);
```

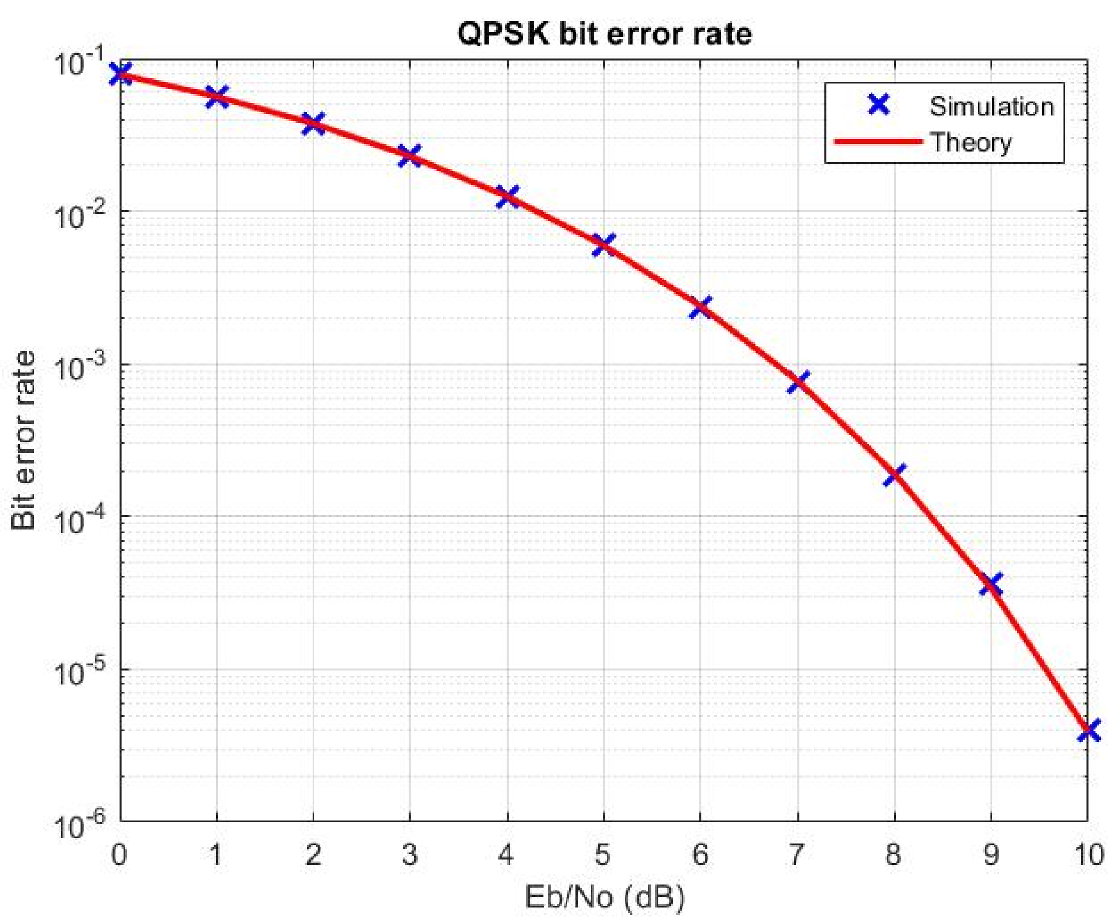
```
figure(1);
semilogy(EbNoVec,berEst,'*')
hold on
semilogy(EbNoVec,berTheory)
grid
legend('Estimated BER','Theoretical BER')
xlabel('Eb/No (dB)')
ylabel('Bit Error Rate')
title ('BER with Binary Encoding')
```



QPSK:

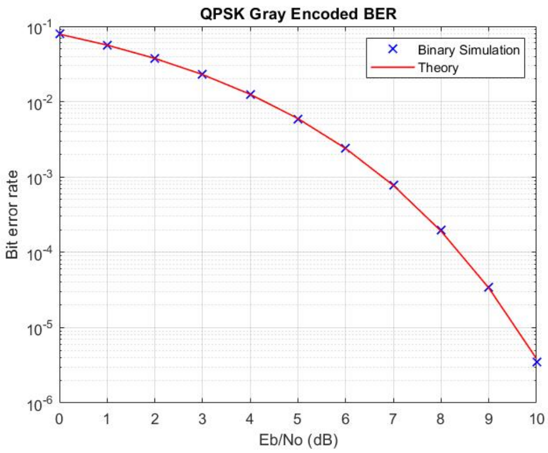
```
%number of symbols in simulation
Nsyms = 1e6;
% energy per symbol
Es = 1;
% energy per bit (2 bits/symbol for QPSK)
Eb = Es / 2;
% Eb/No values to simulate at, in dB
EbNo_dB = linspace(0, 10, 11);


% Eb/No values in linear scale
EbNo_lin = 10.^(EbNo_dB / 10);
% keep track of bit errors for each Eb/No point
bit_err = zeros(size(EbNo_lin));
for i=1:length(EbNo_lin)
```

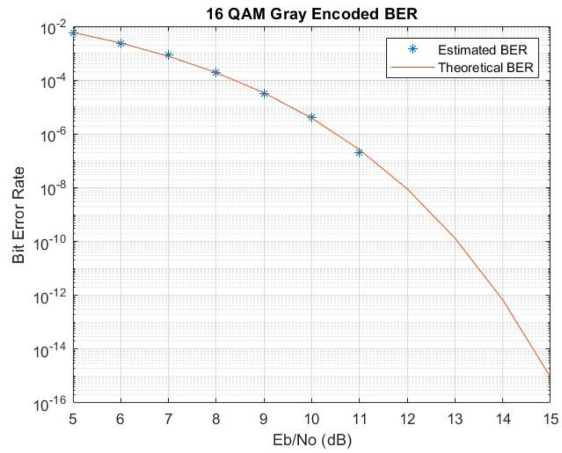
```
    % generate source symbols
    syms = (1 - 2 * (randn(Nsyms,1) > 0)) + j * (1 - 2 * (randn(Nsyms, 1) > 0));
    % add noise
    syms_noisy = sqrt(Es/2) * syms + sqrt(Eb/(2*EbNo_lin(i))) * (randn(size(syms)) + j
* randn(size(syms)));
    % recover symbols from each component (real and imaginary)
    syms_rec_r = sign(real(syms_noisy));
    syms_rec_i = sign(imag(syms_noisy));
    % count bit errors
    bit_err(i) = sum((syms_rec_r ~= real(syms)) + (syms_rec_i ~= imag(syms)));
end
% convert to bit error rate
bit_err = bit_err / (2 * Nsyms);

% calculate theoretical bit error rate, functionally equivalent to:
% bit_err_theo = qfunc(sqrt(2*EbNo_lin));
bit_err_theo = 0.5*erfc(sqrt(2*EbNo_lin)/sqrt(2));
figure;
semilogy(EbNo_dB, bit_err, 'bx', EbNo_dB, bit_err_theo, 'r', 'MarkerSize', 10,
'LineWidth', 2);
xlabel('Eb/No (dB)');
ylabel('Bit error rate');
title('QPSK bit error rate');
legend('Simulation','Theory');
grid on;
```



(ii) Compare the results with and without Gray encoding. And discuss why the results are different.

Solution:





Gray coding improves performance when compared to no gray coding. This is because gray coding allows for a one-bit change rather than two-bit change when a transmitted signal is detected on the other side. As can be seen from the question (i) and (ii) graphs, BER with Gray encoding is identical to the theoretical value, however BER without Gray displays additional errors. It's because one symbol error might result in one or more bits faults when we examine bit-to-symbol mapping. Symbol mistakes that involve a symbol being mistaken with one of its nearest neighbors are the most common. Gray coding ensures that the most common symbol mistakes only result in a one-bit error. Bit mistakes per symbol error will be higher in other forms of coding. Gray coding reduces the bit error rate for a given symbol error rate because of this. As comparison, when one examines binary encoding, the likelihood of flipping the signal is higher, — in other words, for 3 (011) and 4 (100), all bit's inversion, causing delay and more mistakes, whereas in gray, just a single bit is altered, causing less error. Estimated BER with Gray is also better than without gray encoding, as seen in the figures.

(iii) Discuss the difference among SNR, EsN0, and EbN0.

Solution:

The normalized signal to noise ratio, or signal to noise per bit, is defined as E_b/N_o. When evaluating the Bit error rate (BER) performance of various modulation schemes, E_b/N_o is very relevant.

SNR: average signal power divided by average noise power. Let's assume S the average signal power and N the average noise power **S/N.**

E_b / N_o is another signal-to-noise ratio that is standardized or normalized by bit. The energy per bit divided by the power spectral density. However, it is possible to convert from one representation to another.

Let's compare these two measures of the signal quality

E_b vs S: Average energy per bit = average signal power * time for a bit

$E_b = ST_{\mathbf{b}}$
$S = E_b / T_{\mathbf{b}}$

Now, let’s compare for noise

N_o vs N: Noise power density = average noise power / bandwidth

N_o = N/ BW
N = N_o * BW

SNR vs E_b / N_0: SNR = S / N

SNR = $\frac{E_b/T_b}{N_0 \cdot BW}$

SNR = $\frac{E_b}{N_0}\frac{1}{T_b}\frac{1}{BW}$

SNR = $\frac{E_b}{N_0}R_b\frac{1}{BW}$ [we know $1/T_b$ is bit rate]

SNR = $\frac{E_b}{N_0}\frac{R_b}{BW}$ [we know that R_b / BW is the definition of Spectral efficiency]

SNR = $\frac{E_b}{N_0}\eta$

E_b/N_o vs E_s/N_o: Es= energy per symbol (one symbol may represent multiple bits)

$\frac{E_S}{N_0} = \frac{E_B}{N_0}\log_2 M$ [M= number of alternative modulation symbols]

So, $SNR = \frac{E_b}{N_0} \cdot \eta = \frac{E_b}{N_0}\log_2 M$

$$
\begin{aligned}
\left.\frac{E_b}{N_0}\right|_{db} &= SNR|_{db} - \log_2 M|_{db} \\
&= SNR|_{db} - 10\log_{10} k|_{db}
\end{aligned}
$$

(iv) Plot the bit error rate (BER) curves of QPSK and 16QAM with simulation and theoretical value without any ToolBox.

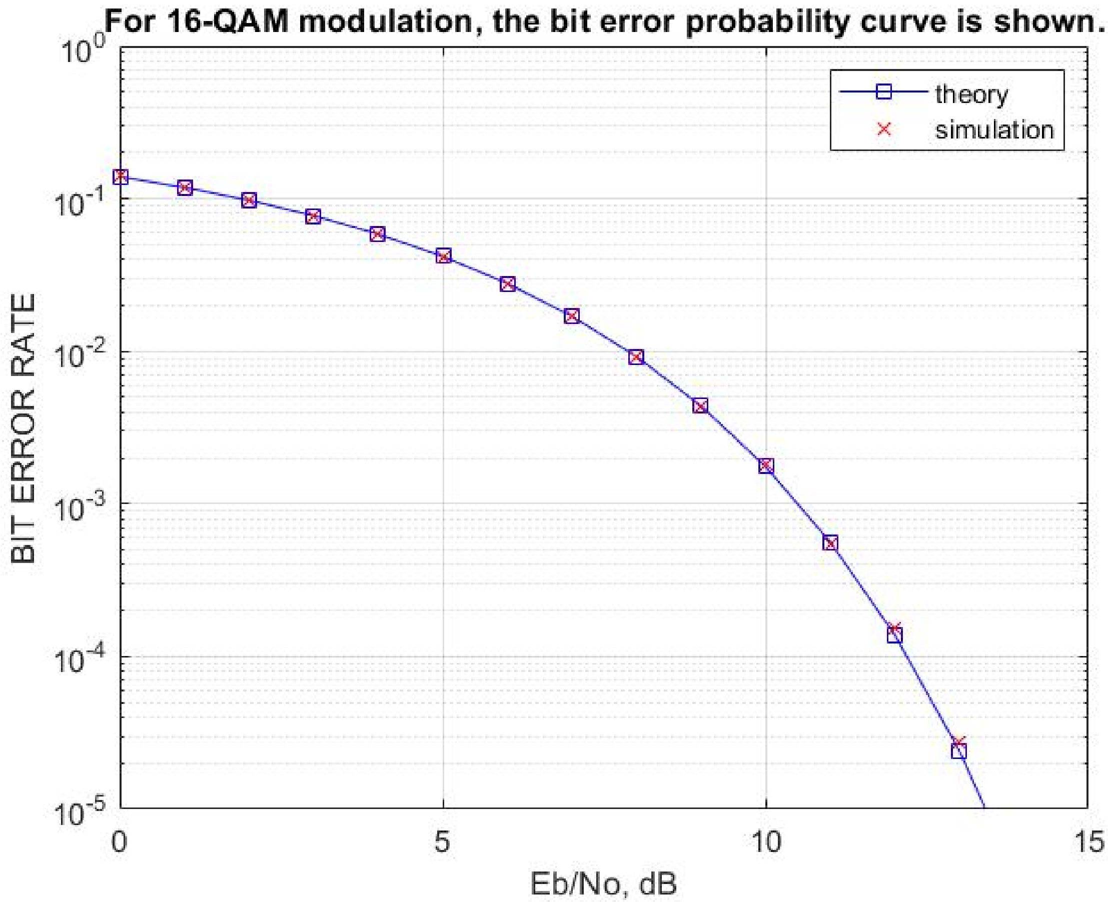
Solution:

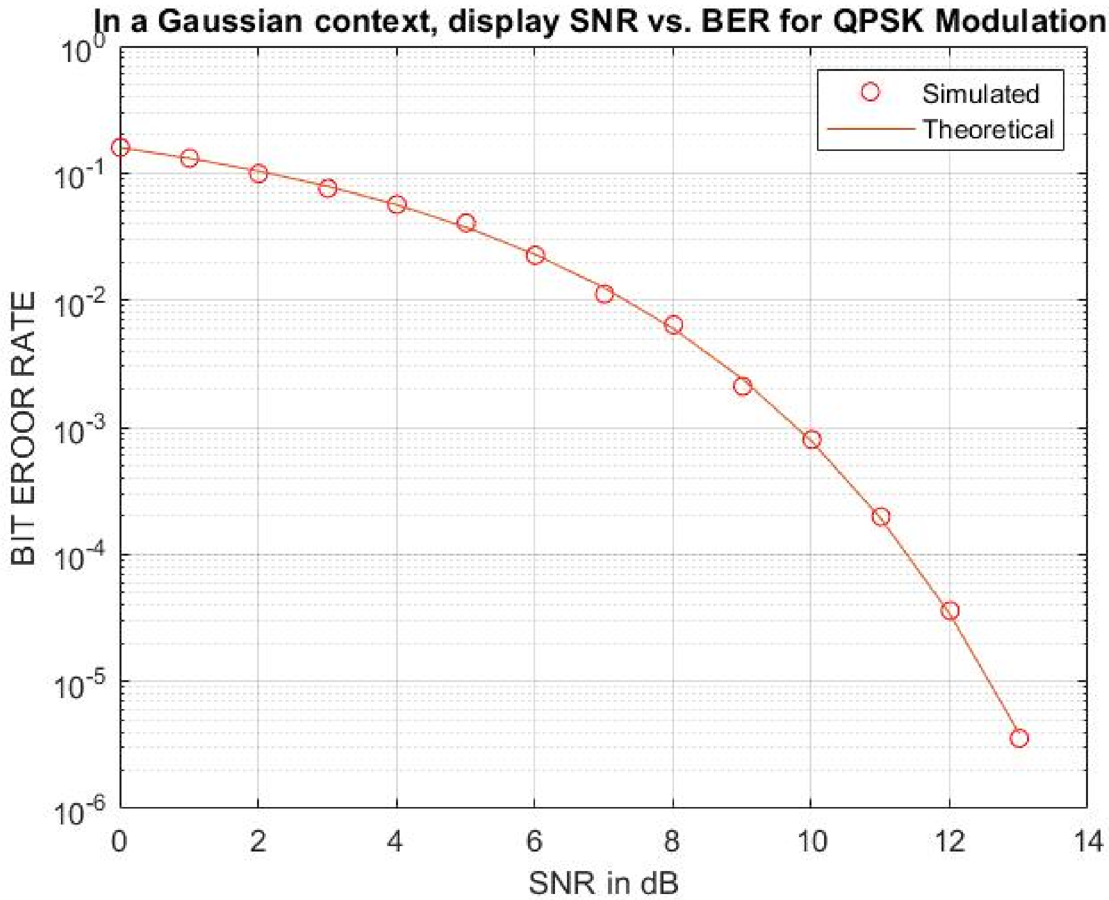
```
% Simulating across a range of SNR
Eb_No = -3: 1: 10;
% Frame Length
bit_num = 10000;

% Main loop
for aa = 1: 1: length(Eb_No)
    % To convert Eb/No numbers to channel SNR, use the formula below.
    SNR = Eb_No + 10*log10(2);
    T_Errors = 0;
    T_bits = 0;


    % Continue until 100 errors.
    while T_Errors < 100
```

```
        % Make a few data bits
        uncoded_bits  = round(rand(1,bit_num));

        % For Quadrature Carriers, split the stream into two streams.
        B1 = uncoded_bits(1:2:end);
        B2 = uncoded_bits(2:2:end);

        % QPSK modulator set to pi/4 radians constellation
        % If you want to change the constellation angles
        % just change the angles.
        qpsk_sig = ((B1==0).*(B2==0)*(exp(i*pi/4))+(B1==0).*(B2==1)...
            *(exp(3*i*pi/4))+(B1==1).*(B2==1)*(exp(5*i*pi/4))...
            +(B1==1).*(B2==0)*(exp(7*i*pi/4)));

        % Noise variance
        N0 = 1/10^(SNR(aa)/10);

        % To the recipient, send a Gaussian Link message.
        rx = qpsk_sig +
sqrt(N0/2)*(randn(1,length(qpsk_sig))+i*randn(1,length(qpsk_sig)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % At the receiver, there is a QPSK demodulator.
        B4 = (real(rx)<0);
        B3 = (imag(rx)<0);

        uncoded_bits_rx = zeros(1,2*length(rx));
        uncoded_bits_rx(1:2:end) = B3;
        uncoded_bits_rx(2:2:end) = B4;

        % Bit Errors Calculation
        diff = uncoded_bits - uncoded_bits_rx;
        T_Errors = T_Errors + sum(abs(diff));
        T_bits = T_bits + length(uncoded_bits);

    end


    BER(aa) = T_Errors / T_bits;

end

% BER through Simulation
figure(1);
semilogy(SNR,BER,'or');
hold on;
xlabel('SNR in dB');
ylabel('BIT EROOR RATE');
title('In a Gaussian context, display SNR vs. BER for QPSK Modulation.');
% Theoretical BER
figure(1);
theoryBer = 0.5*erfc(sqrt(10.^(Eb_No/10)));
semilogy(SNR,theoryBer);
grid on;
legend('Simulated', 'Theoretical');


%%%%%%%%%%%%%%%% Bit Error Rate for 16-QAM modulation %%%%%%%%%%%%%%%%%%

N = 10^5; % the total amount of symbols
```

```matlab
M = 16;   % the size of the constellation
k = log2(M);

% for 16-QAM
alphaRe = [-(2*sqrt(M)/2-1):2:-1 1:2:2*sqrt(M)/2-1];
alphaIm = [-(2*sqrt(M)/2-1):2:-1 1:2:2*sqrt(M)/2-1];
k_16QAM = 1/sqrt(10);

Eb_N0_dB  = [0:15]; % multiple Es/N0 values
Es_N0_dB  = Eb_N0_dB + 10*log10(k);

% Conversion of gray codes
ref = [0:k-1];
map = bitxor(ref,floor(ref/2));
[tt ind] = sort(map);

for ii = 1:length(Eb_N0_dB)

    % symbol generation
    ipBit = rand(1,N*k,1)>0.5;
    ipBitReshape = reshape(ipBit,k,N).';
    bin2DecMatrix = ones(N,1)*(2.^[(k/2-1):-1:0]) ; % Binary to decimal

    % real
    ipBitRe =  ipBitReshape(:,[1:k/2]);
    ipDecRe = sum(ipBitRe.*bin2DecMatrix,2);
    ipGrayDecRe = bitxor(ipDecRe,floor(ipDecRe/2));

    % imaginary
    ipBitIm =  ipBitReshape(:,[k/2+1:k]);
    ipDecIm = sum(ipBitIm.*bin2DecMatrix,2);
    ipGrayDecIm = bitxor(ipDecIm,floor(ipDecIm/2));

    % constellations based on Gray coded symbols
    modRe = alphaRe(ipGrayDecRe+1);
    modIm = alphaIm(ipGrayDecIm+1);


    mod = modRe + j*modIm;
    s = k_16QAM*mod;

    % noise
    n = 1/sqrt(2)*[randn(1,N) + j*randn(1,N)];

    y = s + 10^(-Es_N0_dB(ii)/20)*n; % additive white gaussian noise

    % demodulation

    % real
    y_re = real(y)/k_16QAM;
    % imaginary
    y_im = imag(y)/k_16QAM;

    % rounding up to the next letter of the alphabet
    ipHatRe = 2*floor(y_re/2)+1;
    ipHatRe(find(ipHatRe>max(alphaRe))) = max(alphaRe);
    ipHatRe(find(ipHatRe<min(alphaRe))) = min(alphaRe);
    ipHatIm = 2*floor(y_im/2)+1;
    ipHatIm(find(ipHatIm>max(alphaIm))) = max(alphaIm);
    ipHatIm(find(ipHatIm<min(alphaIm))) = min(alphaIm);

    % Converting from Constellation to Decimal
    ipDecHatRe = ind(floor((ipHatRe+4)/2+1))-1; % LUT based
```

```
    ipDecHatIm = ind(floor((ipHatIm+4)/2+1))-1; % LUT based

    % converting a string to binary
    ipBinHatRe = dec2bin(ipDecHatRe,k/2);
    ipBinHatIm = dec2bin(ipDecHatIm,k/2);

    % translating a binary text to a numerical value
    ipBinHatRe = ipBinHatRe.';
    ipBinHatRe = ipBinHatRe(1:end).';
    ipBinHatRe = reshape(str2num(ipBinHatRe).',k/2,N).' ;

    ipBinHatIm = ipBinHatIm.';
    ipBinHatIm = ipBinHatIm(1:end).';
    ipBinHatIm = reshape(str2num(ipBinHatIm).',k/2,N).' ;

    % Counting both real and imaginary errors
    nBitErr(ii) = size(find([ipBitRe- ipBinHatRe]),1) + size(find([ipBitIm -
ipBinHatIm]),1) ;

end
simBer = nBitErr/(N*k);
theoryBer = (1/k)*3/2*erfc(sqrt(k*0.1*(10.^(Eb_N0_dB/10))));

figure
semilogy(Eb_N0_dB,theoryBer,'bs-',Eb_N0_dB,simBer,'rx');
axis([0 15 10^-5 1])
grid on
legend('theory', 'simulation');
xlabel('Eb/No, dB')
ylabel('BIT ERROR RATE')
title('For 16-QAM modulation, the bit error probability curve is shown.')
```


For 16-QAM modulation, the bit error probability curve is shown.
theory
simulation
BIT ERROR RATE
Eb/No, dB
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
0
5
10
15



In a Gaussian context, display SNR vs. BER for QPSK Modulation.
Simulated
Theoretical
BIT EROOR RATE
SNR in dB
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
0
2
4
6
8
10
12
14