1st assignment for 3rd topic of project

281405

MOHAMMED ALABADSA
Student ID: 28G21130

QPSK:

In QPSK 2 bits are transmitted per symbol, the error occurs if one of these bits is incorrect or both of them are incorrect.
if we assume that P_e is error probability of the first bit so $(1 - P_e)$ is the probability of the correct case of the first bit. Similarly, the probability of the correct case of the second bet will be $(1 - P_e)$.

The probability of correct first symbol is $(1 - P_e)^2$

The probability of symbol error is $1 - (1 - P_e)^2 = 1 - \left(1 - {P_e}^2 - 2P_e\right) = 2P_e - {P_e}^2$

Where $P_e = Q\left(\sqrt{\frac{E_s}{N_0}}\right)$

The probability of symbol error = $2Q\left(\sqrt{\frac{E_s}{N_0}}\right) - \left[Q\left(\sqrt{\frac{E_s}{N_0}}\right)\right]^2$

In QPSK the SNR is high, thus $Q\left(\sqrt{\frac{E_s}{N_0}}\right) \ll 1$

$\left[Q\left(\sqrt{\frac{E_s}{N_0}}\right)\right]^2 \ll Q\left(\sqrt{\frac{E_s}{N_0}}\right)$

Hence, $P_s = 2Q\left(\sqrt{\frac{E_s}{N_0}}\right)$, since 2 bits are transmitted in a symbol, when we write bit-error-rate (BER) we divide by 2. Note that $E_s = 2E_b$

$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$

$$\boldsymbol{P_b = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)}$$

16-QAM:

The same as in QPSK, the probability of symbol error is $1 - (1 - P_e)^2$

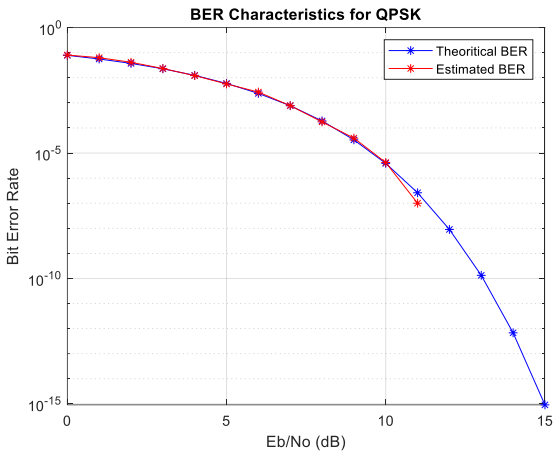
Where $P_e = \frac{6}{4} Q\left(\sqrt{\frac{E_s}{5N_0}}\right)$

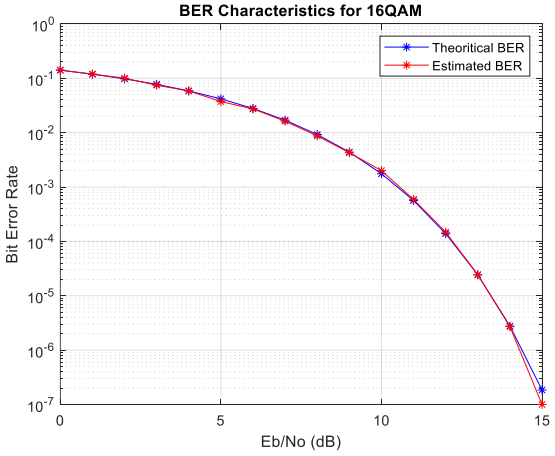
$P_s = 3Q\left(\sqrt{\frac{E_s}{5N_0}}\right)$, where $E_s = 4E_b$, since 4 bits are transmitted in a symbol, we divide by 4.

$P_b = \frac{3}{4} Q\left(\sqrt{\frac{4E_b}{5N_0}}\right)$

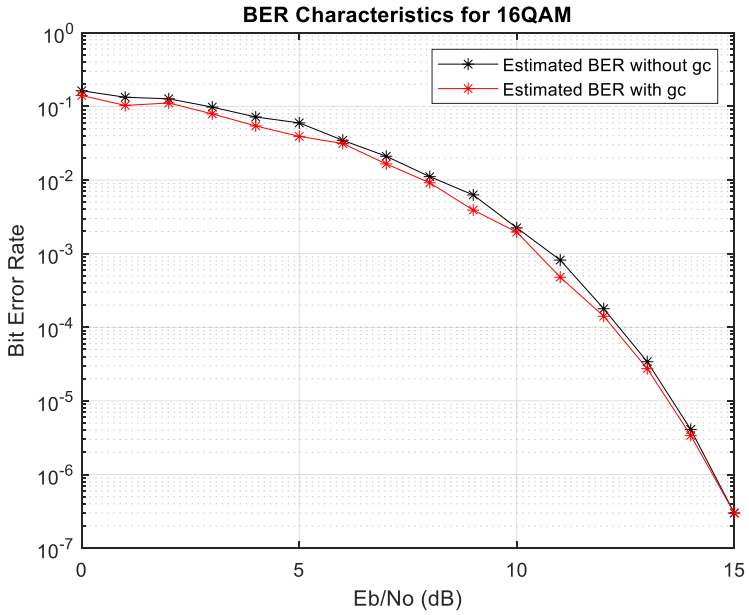
$$\boldsymbol{P_b = \frac{3}{8} erfc\left(\sqrt{\frac{4E_b}{10N_0}}\right)}$$

- **Using the reference code:**





- **Gray coding**



It can be noticed that with gray coding the performance is better compared with in case without gray coding, the reason is that gray coding permits the change in one bit instead of two bits when transmitted signal is detected in the other side, for example, if 00 was sent and passed through some noise, it may be detected as 01 or 10 on the receiver side, it can be seen that there is just one bit change between transmitted and received signal. On the other hand, in case of binary there will be two bits change, that is why with gray coding, it has a better BER performance.

- **SNR**

It refers to signal to noise ratio or average signal power divided by average noise power, it is measured in dB and it can be mathematically represented as:

$$SNR = \frac{Signal\ Power\ (S)}{Noise\ Power\ (N)}$$

- $\frac{E_b}{N_0}$

It is normalized signal to noise ratio or standardized SNR by bit, it can be written as

$$= \frac{Energy\ per\ bit\ (Eb)}{Noise\ power\ spectral\ density\ (N0)}$$

$S = E_b/T_b$
$N = N_0 \cdot BW$

$$SNR = \frac{S}{N}$$
$$= \frac{E_b/T_b}{N_0 \cdot BW} = \frac{E_b}{N_0}\frac{1}{T_b}\frac{1}{BW}$$
$$= \frac{E_b}{N_0}R_b\frac{1}{BW} = \frac{E_b}{N_0}\frac{R_b}{BW}$$
$$SNR = \frac{E_b}{N_0} \cdot \eta = \frac{E_b}{N_0} \cdot \log_2 M$$

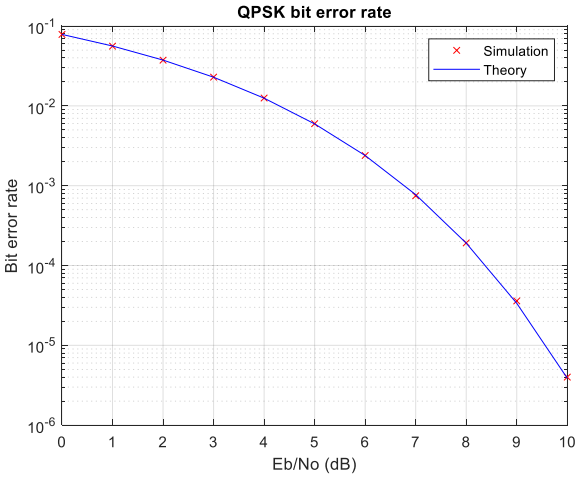
η is the spectral efficiency

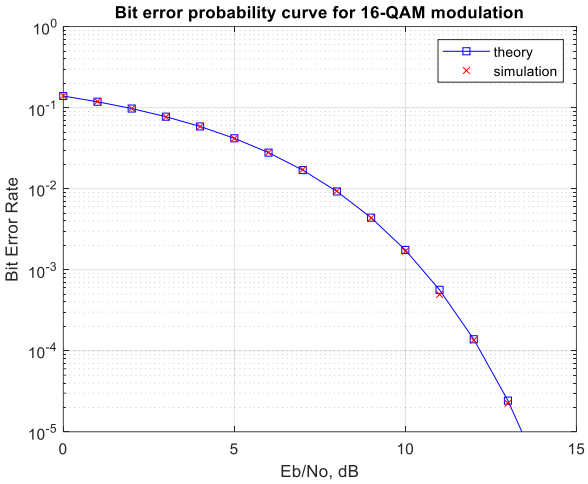
- $\frac{E_s}{N_0}$

It is symbol energy to noise ratio $= \frac{Energy\ per\ symbol}{Noise\ power\ spectral\ density}$

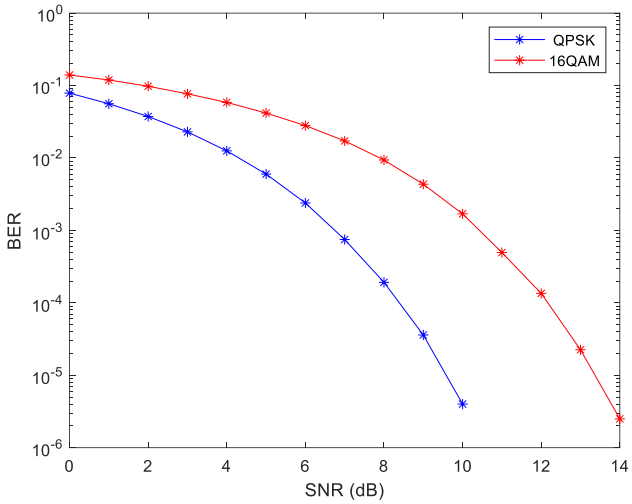
$E_s = E_b * \log_2 M$, M is modulation order

- **Without toolbox**





Comparison between QPSK and 16QAM



Reference code:

```
clear all;clc;close all;

M = 16;                 % Modulation order
k = log2(M);            % Bits per symbol
EbNoVec = (0:15)';      % Eb/No values (dB)
numSymPerFrame = 100;   % Number of QAM symbols per frame

berEst = zeros(size(EbNoVec));
berEst_gr = 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;
    numErrs_gr = 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 'binary' and 'Gray' symbol mapping
        txSig = qammod(dataSym,M,'bin');
        txSig_gr = qammod(dataSym,M);

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

        % Demodulate the noisy signal
        rxSym = qamdemod(rxSig,M,'bin');
        rxSym_gr = qamdemod(rxSig_gr,M);

        % Convert received symbols to bits
        dataOut = de2bi(rxSym,k);
        dataOut_gr = de2bi(rxSym_gr,k);

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

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

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

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

semilogy(EbNoVec,berTheory,'b-*')
hold on
semilogy(EbNoVec,berEst,'k-*')
hold on
semilogy(EbNoVec,berEst_gr,'r-*')
grid
title('BER Characteristics for 16QAM')
legend('Theoritical BER','Estimated BER without gc','Estimated BER with gc')
xlabel('Eb/No (dB)')
ylabel('Bit Error Rate')
```

Without toolbox:

```
clear all;clc;close all;
%%---------------------------------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 = 0.5*erfc(sqrt(2*EbNo_lin)/sqrt(2));
figure;
semilogy(EbNo_dB, bit_err, 'rx', EbNo_dB, bit_err_theo, 'b-');
xlabel('Eb/No (dB)');
ylabel('Bit error rate');
title('QPSK bit error rate');
legend('Simulation','Theory');
grid on;

%%-------------------------------16QAM-------------------------------------------------
------
% Bit Error Rate for 16-QAM modulation using Gray modulation mapping
N = 10^5; % number of symbols
M = 16;   % constellation size
k = log2(M); % bits per symbol

% defining the real and imaginary PAM constellation
% 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);

% Mapping for binary <--> Gray code conversion
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; % random 1's and 0's
    ipBitReshape = reshape(ipBit,k,N).';
    bin2DecMatrix = ones(N,1)*(2.^[(k/2-1):-1:0]) ; % conversion from 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));
    % mapping the Gray coded symbols into constellation
    modRe = alphaRe(ipGrayDecRe+1);
    modIm = alphaIm(ipGrayDecIm+1);
    % complex constellation
    mod = modRe + j*modIm;
    s = k_16QAM*mod; % normalization of transmit power to one

    % noise
    % -----
    n = 1/sqrt(2)*[randn(1,N) + j*randn(1,N)]; % white guassian noise, 0dB variance

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

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

    % rounding to the nearest 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);

    % Constellation to Decimal conversion
    ipDecHatRe = ind(floor((ipHatRe+4)/2+1))-1; % LUT based
    ipDecHatIm = ind(floor((ipHatIm+4)/2+1))-1; % LUT based

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

```
    ipBinHatIm = dec2bin(ipDecHatIm,k/2);

    % converting binary string to number
    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 errors for real and imaginary
    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('Bit error probability curve for 16-QAM modulation')

%%
figure();
semilogy(EbNo_dB,bit_err,'b-*',Eb_N0_dB,simBer,'r-*');
hold on;
xlabel('SNR (dB)');
ylabel('BER');
legend(' QPSK','16QAM');
```