G. S. Sanyal School of Telecommunications
Indian Institute of Technology Kharagpur

MOOC: Spread Spectrum Communications & Jamming

Assignment 6: Solution to MATLAB code for BER generation of QPSK system over AWGN channel.

Due date: Max. marks: 20

Write a MATLAB code to generate Bit Error Rate (BER) vs Bit-Energy-to-Noise-Power-Spectral-Density ratio (E_b/N_0) and Signal-Power-to-Noise-Power ratio (SNR) for Quadrature Phase Shift Keying (QPSK) system over Additive White Guassian Noise (AWGN) channel based on the QPSK modulator and demodulator system depicted in Fig. 1. Referring to the same, and following the table of notations (Table 1), answer the following questions:

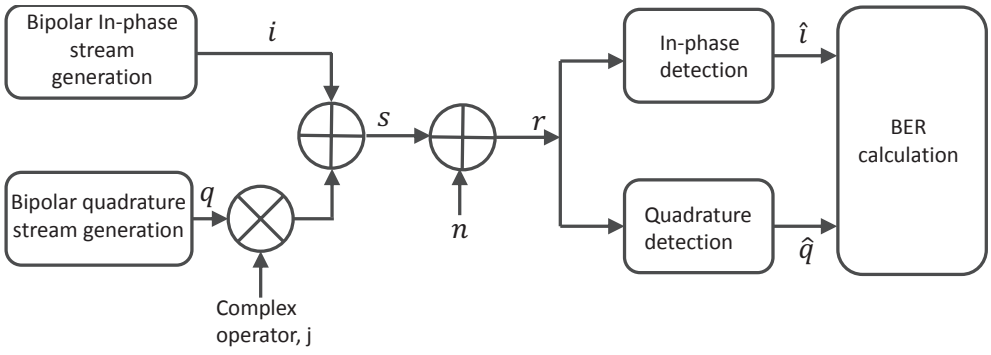

Figure 1: QPSK modulator and demodulator system

Parameter	**Mathematical notation**	**MATLAB variable**
In-phase bipolar data	i	I_data
Quadrature bipolar data	q	Q_data
Transmitted complex data	s	tx_data
AWGN channel noise	n	n_AWGN
Received signal	r	rec_data
Decoded in-phase data	$\hat{i}$	decod_sig_I
Decoded quadrature data	$\hat{q}$	decod_sig_Q
Number of error in in-phase data	E_i	Err1
Number of error in quadrature data	E_q	Err2

Table 1: Table of notations.

1. The random bipolar data samples (3×10^5 samples each) can be generated using the following MATLAB command:

   Solution: The random bipolar sequences I_data and Q_data are generated using commands in line numbers 13-14 of the appended MATLAB code. The MATLAB variable no_samples specifies number of samples to be generated using *rand* function. i.e., no_samples = 3*10^5.
   (correct option iii.)

2. The baseband QPSK modulated signal s can be generated using the MATLAB command:

   Solution: Since baseband QPSK signal is complex, it can be generated using command given in line number 16 of the appended MATLAB code.
   (correct option iv.)

3. The complex AWGN signal, n with zero mean and variance equal to 0.5 can be generated using the following MATLAB command: (Hint: The MATLAB function *randn* generates a zero mean Gaussian random variable with unit variance. AWGN noise for QPSK signal is complex Gaussian (in-phase and quadrature components)).

   Solution: Refer to line number 26 of the appended MATLAB code. Since *randn* function generates real valued AWGN noise with zero mean and unit variance, the sequence generated by *randn* function is scaled by a factor of $1/\sqrt{2}$. As a result, the resulting noise sequence possesses a variance of 0.5 for both real and imaginary components. It should be noted that the number of noise samples should be equal to $length(tx\_data)$=3*10^5. **(correct option i.)**

4. The zero crossing detector at the receiver can be implemented using the following MATLAB command:

   Solution: Refer to line numbers 38-39 of the appended MATLAB code. The respective commands in line numbers 38 and 39 convert the received in-pahase and quadrature components into bipolar streams based on zero crossing detection.
   (correct option iii.)

5. The number of bit errors for in-phase transmission is calculated using the MATLAB command:

   Solution: Refer to line number 40 of the appended MATLAB code.
   (correct option iii.)

6. The BER for QPSK transmission is calculated using the MATLAB command:

   Solution: Refer to line number 42 of the appended MATLAB code. BER for QPSK transmission scheme is calculated by taking ratio of total number of errors in both in-phase and quadrature streams to the total number of bits transmitted.
   (correct option iv.)

7. The BER plot versus E_b/N_0 for the QPSK system is:

   Solution: To obtain BER value with respect to a given Eb/No, a proper magnitude scaling is required for both noise and the transmitted signal. The command in line number 29 ensures required Eb/No for the received signal rec_data.
   (correct option iv.)

8. The BER plot versus SNR for the QPSK system is: (Hint: For QPSK, 2*Eb/No = SNR)

   Solution: Refer to line number 31 of the appended MATLAB code. The scaling factor, 1/sqrt(pow_txdata), to tx_data ensures the condition 2*Eb/No = SNR.
   (correct option iv.)

Appendix

```
%**************************************************************************
        % This code is to find the BER for QPSK signalling over AWGN channel.
        % Institute: GSSST,IIT Kharagpur.
        % Written by: Rakesh R T, Research Scholar, GSSST.
%**************************************************************************
clc;clear all;close all;
no_samples = 3e5; % no. of bits to be transmitted.
EbNodB = -2:2:14; % Range of Eb/No values.
SNRdB = -2:2:14; % Range of SNR values.
sel_var = 2; %1-for BER vs Eb/No, 2-for BER vs SNR
%--------------------------------------------------------------------------
%%
I_data = 2*(rand(1,no_samples)>0.5)-1;%inphase bipolar seq. for txn.
Q_data = 2*(rand(1,no_samples)>0.5)-1;%quadrature bipolar seq. for txn.

tx_data = I_data+j*Q_data; % Tx baseband signal.
pow_txdata = sum(abs(tx_data).^2)/length(tx_data); % calc. signal power
N = length(tx_data); % length of the transmit data
%%
for kk = 1:length(EbNodB) % for different values of Eb/No

    %***************** Implementing the channel*************************
    EbNo_lin = 10^((EbNodB(kk))/10);% to linear scale.


    n_AWGN = (1/sqrt(2))*(randn(1,length(tx_data))+j*randn(1,length(tx_data)));
                                % variance = 0.5
    if sel_var == 1;
        rec_data = tx_data + (1/sqrt(EbNo_lin))*n_AWGN; % received signal
    else
```

```
        rec_data = (tx_data/sqrt(pow_txdata)) + (1/sqrt(SNR_lin))*n_AWGN;
        % received signal
    end


    %% ******************Receiver part of the system**********************

    decod_sig_I = 2*(real(rec_data)>0)-1; %I-phase decoded bits
    decod_sig_Q = 2*(imag(rec_data)>0)-1; %Q-phase decoded bits
    Err1 = sum(decod_sig_I ~= I_data);% Error detection
    Err2 = sum(decod_sig_Q ~= Q_data);% Error detection
    BER(kk) = (Err1+Err2)/(2*no_samples); % probability of bit error
end
%%
figure
if sel_var == 1;
    semilogy(EbNodB, BER); %semi-log plot
    xlabel('Eb/No in dB');
    ylabel('BER');
else
    semilogy(SNRdB, BER); %semi-log plot
    xlabel('SNR in dB');
    ylabel('BER');
end
legend ('Prob. of Error')
grid on;
```