Assignment 9: Solution to MATLAB code for BER generation of Spread Spectrum QPSK system over fading 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 ($E_b/N_0$) plot for Spread Spectrum Quadrature Phase Shift Keying (QPSK) system over Rayleigh fading channel (averaged over at least 1000 iterations). Assume system employs a Hadamard sequence of length 4 for spreading data. Fig. 1 depicts a QPSK Spread Spectrum modulator and demodulator system. Referring to the same, answer the following:

Important Note: Refer to the equalizer and channel generation portions of the code provided in tutorial 7 in Week 9 content, and use the same to solve this assignment.

Figure 1: QPSK spread spectrum modulator and demodulator system with fading channel.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mathematical notation</th>
<th>MATLAB variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-phase bipolar data</td>
<td>$i$</td>
<td>$I_{\text{data}}$</td>
</tr>
<tr>
<td>Quadrature bipolar data</td>
<td>$q$</td>
<td>$Q_{\text{data}}$</td>
</tr>
<tr>
<td>Baseband complex data</td>
<td>$s_s$</td>
<td>base_sig</td>
</tr>
<tr>
<td>Spreading code</td>
<td>$c$</td>
<td>spread_code</td>
</tr>
<tr>
<td>Transmit spread signal</td>
<td>$s$</td>
<td>tx_data</td>
</tr>
<tr>
<td>Number of multipath components</td>
<td>$l$</td>
<td>no_multipath</td>
</tr>
<tr>
<td>Multipath delay index vector</td>
<td>$\tau$</td>
<td>tau</td>
</tr>
<tr>
<td>Decay parameter for channel</td>
<td>$\alpha$</td>
<td>decay_para</td>
</tr>
<tr>
<td>Power delay profile vector</td>
<td>$P(\tau)$</td>
<td>PDP</td>
</tr>
<tr>
<td>Complex channel with Rayleigh amplitude and uniformly distributed phase</td>
<td>$c_{\text{amp}}$</td>
<td>Rayleigh_amp</td>
</tr>
<tr>
<td>Channel impulse response without normalization</td>
<td>$c_{\text{un}}$</td>
<td>imp_res_Unnorm</td>
</tr>
<tr>
<td>Channel impulse response with normalization</td>
<td>$c_{\text{n}}$</td>
<td>imp_res</td>
</tr>
<tr>
<td>Channel power</td>
<td>$P_{\text{ch}}$</td>
<td>ch_power</td>
</tr>
<tr>
<td>Transmit spread signal after channel</td>
<td>$s'$</td>
<td>tx_data_ch</td>
</tr>
<tr>
<td>AWGN channel noise</td>
<td>$n$</td>
<td>n_AWGN</td>
</tr>
<tr>
<td>Received signal</td>
<td>$r$</td>
<td>rec_data</td>
</tr>
<tr>
<td>Despread signal</td>
<td>$r_d$</td>
<td>despread_data</td>
</tr>
<tr>
<td>Decoded in-phase data</td>
<td>$\hat{i}$</td>
<td>decod_sig_I</td>
</tr>
<tr>
<td>Decoded quadrature data</td>
<td>$\hat{q}$</td>
<td>decod_sig_Q</td>
</tr>
</tbody>
</table>

Table 1: Table of notations.

1. The corresponding multipath delay index vector can be generated using the MATLAB command:

**Solution:** The multipath delay vector can be generated using the command in line number 18 of the appended MATLAB code.

*(correct option ii.)*

2. For a given decay parameter, the MATLAB command to generate exponential power delay profile of the channel is:

The exponential power delay profile of the channel can be generated using the command in line number 19 of the appended MATLAB code.

*(correct option iv.)*

3. The MATLAB command to generate multipath channel components with Rayleigh distributed amplitude and uniformly distributed phase is:

The multipath channel components with Rayleigh distributed amplitude and uniformly distributed phase is generated using the command in line number 40 of the appended MATLAB code.
4. The MATLAB command to generate channel impulse response (without normalization) is:
The channel impulse response without normalization can be generated using the command in line number 43 of the appended MATLAB code, where \( PDP_{\text{amp}} = \sqrt{PDP} \).
(correct option ii.)

5. The MATLAB command to calculate the net power of the channel multipath components (of the channel generated in question no. 4) is:
The net power of the channel multipath components can be calculated using the command in line number 44 of the appended MATLAB code.
(correct option iii.)

6. The effect of multipath channel on transmitted data \( s \) can be captured using MATLAB command:
The effect of multipath channel on transmitted data \( s \) can be captured using the command in line number 47 of the appended MATLAB code.
(correct option i.)

7. For a decay parameter, \( \alpha = 1.2 \) and number of multipath components, \( l = 4 \), the BER vs \( \text{EbNo} \) (averaged over atleast 10000 iterations) plot is approximately:
The BER vs \( \text{Eb/No} \) is plotted using the commands in line numbers 97-102 with \( \alpha = 1.2 \) in line number 16 of the appended MATLAB code.
(correct option iii.)

8. For a decay parameter, \( \alpha = 2.5 \) and number of multi-path \( l = 4 \), the BER vs \( \text{EbNo} \) (averaged over at least 10000 iterations) plot is approximately:
The BER vs \( \text{Eb/No} \) is plotted using the commands in line numbers 97-102 with \( \alpha = 2.5 \) in line number 16 of the appended MATLAB code.
(correct option i.)
Appendix

```matlab
% This code is to find the BER for BFSK spreading.
% Channel - frequency selective Rayleigh channel.
% Power delay profile (PDP)- exponential.
% Equalizer - MVSE equalizer.
% Spreading code - Walsh-Hadamard code (length=4)
% Institute: GSSSI, IIT Kharagpur.

clc; clear all; close all;

no_samples = 3e5; % no. of bits to be transmitted.
no_iter = 10000; % no. of iterations for averaging BER.
len_code = 4; % length of spreading code.
EbNodB = -2:2:12; % range of Eb/N0 values.
prob_err_avg = zeros(1,length(EbNodB)); % Initialization for avg. prob. error

%------------------ channel impulse response generation----------------------
decay_para = 1.2; % decay factor for FDP
no_multipath = 4; % no. of multipaths.
tau = 0:no_multipath-1; % delay corresponds to 4 multi-paths.
FDP = exp(-decay_para*tau);
PDP_amp = sqrt(FDP);

%-------------------------- code generation---------------------------------
K = no_multipath;
L = K;

code_mat = hadamard(len_code); % generates 3 Walsh-Hadamard codes (length=4)

I_data = 2*(rand(1,no_samples)>0.5)-1; % bipolar seq. for I-channel data.
Q_data = 2*(rand(1,no_samples)>0.5)-1; % bipolar seq. for Q-channel data.
base_data = I_data + 0j*Q_data; % complex baseband data.
```
tx_data = kron(base_data,code_mat(3,:)); % spread data
pow_txdata = sum(abs(tx_data).^2)/length(tx_data); % calc. signal power
N = length(tx_data); % length of the transmit data

for ii = 1:no_iter
  for kk = 1:length(EbNoDB) % for different values of Eb/No

    %********************** Implementing the channel***************************
    Rayleigh_amp = (1/sqrt(2))*(randn(1,no_multipath)+j*randn(1,no_multipath));
    % results in Rayleigh distributed amplitude
    % and uniformly distributed phase.
    imp_res_unnorm = Rayleigh_amp.*PDF_amp;
    ch_power = sum(abs(imp_res_unnorm).^2); % channel power
    imp_res = imp_res_unnorm/sqrt(ch_power); % power is normalized to unity
    EbNo_lin = 10^(EbNoDB(kk)/10); % to linear scale.
    tx_data_ch = conv(tx_data,imp_res); % convolution with channel impulse
    %response.
    % code length is included to account the relation Eb=len_code*Ec
    noise = sqrt(pow_txdata)*(sqrt(len_code)/sqrt(2*EbNo_lin))...% note
    (randn(1,length(tx_data_ch))+j*randn(1,length(tx_data_ch)));
    % noise power is adjusted
    % according to Eb/No and length of the code
    rec_data = tx_data_ch + noise; % received signal
    sigma_sq(kk) = var(noise); % this is for equalizer

    %****************** Receiver part of the system**************************
    % NMSE based equalization (assuming perfect channel state information
    % at receiver)
    Lc = length(imp_res); % length of impulse response
La = Lc+1; % order of the equalizer
L = Lc+Lc;
b = [imp_res(1) zeros(1,La)];
a = [imp_res zeros(1,La)];
Clow = toeplitz(a,b);
DEN = ctranspose(Clow)*Clow + sigma_sq(kk)*eye(La+1);
req = [zeros(1,(L-1)/2) 1 zeros(1,(L-1)/2)]; % required response looks
% like delta function
NUM = ctranspose(Clow)*req';
coeff_mmse = inv(DEN)*NUM; % MMSE equalizer coefficients
eq_sig_mmse = conv(rec_data,coeff_mmse); % equalized receive signal
ov_all_ch = conv(imp_res,coeff_mmse);
[C,I] = max(ov_all_ch); % to obtain synchronization with strong multipath
eq_sig_mmse = eq_sig_mmse(I:end)/C; % to compensate the delay introduced
% by channel and equalizer.
sig_mmse = eq_sig_mmse(1:length(tx_data));
rec_data = sig_mmse; % equalized signal for despreading
temp_sig = reshape(rec_data, len_code, length(rec_data)/len_code);
temp_sig = transpose(temp_sig);
[x,y] = size(temp_sig);
temp_sig1 = kron(ones(x,1),code_mat(3,:)); % generate periodic code seq.
temp_sig3 = transpose(temp_sig.*temp_sig1); % multiply with spread seq
despr = (1/len_code)*sum(temp_sig3); % despreading is completed
decod_sig_I = 2*(real(despr)>0)-1; % detection for real part.
decod_sig_Q = 2*(imag(despr)>0)-1; % detection for real part.
Err1 = sum(decod_sig_I == I_data); % Error detection for I-channel
Err2 = sum(decod_sig_Q == Q_data); % Error detection for Q-channel

prob_err(kk) = (Err1+Err2)/(2*no_samples); % probability of bit error
end
prob_err_avg = prob_err_avg + prob_err;
end

% prob_err_avg = prob_err_avg/no_iter; % average prob. of bit error.
figure
semilogy(EbNcdB, prob_err_avg); % semi-log plot
xlabel('Eb/Nc in dB');
ylabel('BER');
legend ('Prob. of Error')
grid on;