Name: Indu Ilanchezian
Roll No: IMT2014024

DIGITAL COMMUNICATIONS LAB
LAB 5.2
GENERATE BER vs. SNR PLOTS FOR (15,7), (15,11) CODER OVER AWGN CHANNEL

FREQUENCY SHIFT KEYING:

The tones used are: f1 = 1000Hz and f2 = 2000Hz
The bit rate is 1000 bps. Therefore, the duration of each bit is 1/1000 = **1 millisecond**
Number of samples = Sampling frequency * duration of each bit = **8 samples**
The bandwidth is 2 * (bit rate) + (f2-f1) = 3000 Hz

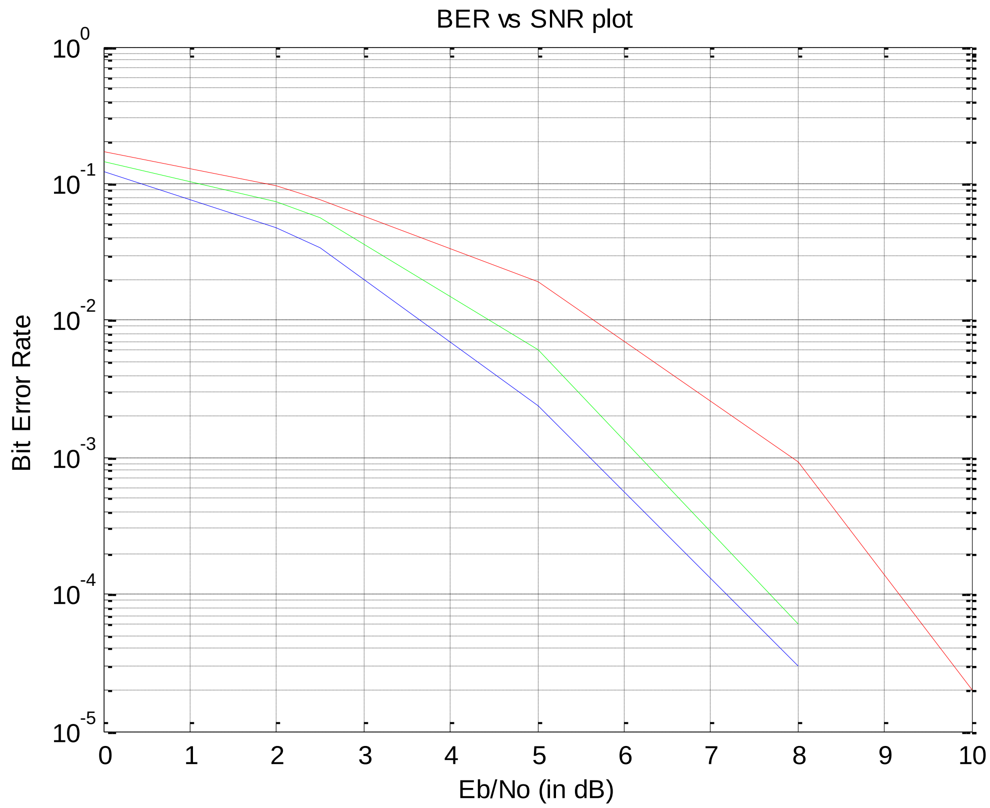
The two waveforms used are:

0: $\sqrt{\frac{2}{T}}\cos(2000\pi t)$

1: $\sqrt{\frac{2}{T}}\cos(4000\pi t)$

The BER vs. SNR plot for the following three cases are plotted in the graph shown below:

1. No Coder
2. (15,11) coder
3. (15,7) coder



Legend:
Red - No Coder
Green - (15,11) Coder
Blue - (15,7) Coder

Name: Indu Ilanchezian
Roll No: IMT2014024

Note: The graph is plotted for Eb/No values : [-5 -2.5 0 2.5 5 10] dB

We can observe that the (15, 7) coder performs better than the (15, 11) coder and the no coder case as the BER values for the (15,7) coder is lower than those of the (15, 11) coder and the no coder case for the same Eb/No ratio. Similarly, the one-bit error correcting code shows a lower BER as compared to the no coder case.

Name: Indu Ilanchezian
Roll No: IMT2014024

APPENDIX-1

1. Function to modulate a given sequence of bits using Frequency Shift Keying

```
function symbols = bits2symbolsfsk(bits_vector)
%This function returns a vector of symbols corresponding to the given
%vector of bits
%The number of samples taken for each time interval T(0.001s) is 8 samples
%The waveforms are orthogonal
%Bit 0 is mapped to Acos(2*pi*1000*t) and bit 1 is mapped to
%Acos(2*pi*2000*t). Here, A is (2/T)^0.5
num_samples = 8;
T = 0.001;
A = (2/T)^(0.5);
t = 0:T/num_samples:T-0.00001;
x = A.*cos(2000*pi*t);
y = A.*cos(4000*pi*t);
output_matrix = zeros(length(bits_vector), num_samples);
for i = 1:1:length(bits_vector)
    if bits_vector(i) == 0
        output_matrix(i,:) = x;
    else
        output_matrix(i,:) = y;
    end
end
symbols = reshape(output_matrix', 1, length(bits_vector)*num_samples);
end
```

2. Function to add white Gaussian noise to the waveform

```
function output_vector = addwhitegaussiannoisefsk(input_vector, SNR)
%The function takes a signal as an input
%The output is a noisy signal, with white gaussian noise added to the input
signal
%The input SNR is in dB
%The standard deviation(sigma) is computed using the given SNR value
SNR = db2pow(SNR);
%Eb = power * time
Tb = 0.001;
Eb = (sum(input_vector.^2)/length(input_vector))*Tb;
No = Eb/SNR;
%noise_power = (No/2) * Bandwidth
bandwidth = 3000;
noise_power = (No/2)*bandwidth;
sigma = noise_power^0.5;
s = size(input_vector);
noise = sigma.*randn(s(1),s(2));
output_vector = input_vector + noise;
end
```

Name: Indu Ilanchezian
Roll No: IMT2014024

3. Function to convert the signal to corresponding bit sequence using a matched filter

```
function bit_vector = symbols2bitsfsk(symbol_vector)
num_samples = 8;
T = 0.001;
A = (2/T)^(0.5);
t = 0:T/num_samples:T-0.00001;
%The matched filer corresponding to bit 1
matchedfilters1 = A.*cos(2000*pi*(T-t));
%The matched filter corresponding to bit 0
matchedfilters2 = A.*cos(4000*pi*(T-t));
%The number of samples in each time interval T is 8
%The number of symbols is equal to (length of signal)/(number of samples)
numsymbols = length(symbol_vector)/num_samples;
bit_vector = zeros(1, length(symbol_vector)/num_samples);
symbol_matrix = vec2mat(symbol_vector,num_samples);
%Convert each symbol to the corresponding bit
for i = 1:1:numsymbols
    current_symbol = symbol_matrix(i,:);
    %Convolve with matched filter corresponding to symbol 1 and get the
peak value
    z1 = max(conv(current_symbol, matchedfilters1));
    %Convolve with matched filter corresponding to symbol 2 and get the
peak value
    z2 = max(conv(current_symbol, matchedfilters2));
    %If z1>z2, the symbol is s1(i.e. 1), otherwise symbol is s2(i.e. 0)
    if z1 > z2
        bit_vector(:,i) = 0;
    else
        bit_vector(:,i) = 1;
    end
end
end
```

4. Script to plot BER vs. SNR when no coder is used

```
message = randi(2,1,10^5)-1;
SNR = [0,2,2.5,5,8,10,15];
ber = zeros(1,length(SNR));
block_length = 7;
for j = 1: 1:length(SNR)
    x_vector = [];
    x_final = [];
    disp(SNR(j));
    x = message;
    signal = bits2symbolsfsk(x);
    noisy_signal = addwhitegaussiannoisefsk(signal,SNR(j));
    recv_signal = symbols2bitsfsk(noisy_signal);
    x_final = [x_final, recv_signal];
    disp(size(message));
    disp(size(x_final));
    error_matrix = mod((message-x_final),2);
    disp(size(error_matrix));
    count = numel(find (error_matrix == 1));
    bit_error_rate = count/length(message);
    ber(:,j) = bit_error_rate;
end
semilogy(SNR, ber, 'r');
grid;
xlabel('Eb/No (in dB)');
```

Name: Indu Ilanchezian
Roll No: IMT2014024

```
ylabel('Bit Error Rate');
title('BER vs SNR plot');
```

5. Function to plot BER vs. SNR graph using (15, 11) and (15, 7) coder

```
function ber = simulatechannelfsk(encoderType,color)
message = randi(2,1,10^5)-1;
if encoderType == 1
    g = [1, 0, 0, 1, 1];
    block_length = 11;
elseif encoderType == 2
    g = [1, 1, 1, 0, 1, 0, 0, 0, 1];
    block_length = 7;
end
message = vec2mat(message, block_length);
SNR = [0,2,2.5,5,8,10,15];
ber = zeros(1,length(SNR));
for j = 1: 1:length(SNR)
    message_final = zeros(size(message,1),block_length);
    disp(SNR(j));
    for i = 1 : 1: size(message,1)
        x = message(i,:);
        u = BCHEncoder(x,encoderType);
        signal = bits2symbolsfsk(u);
        noisy_signal = addwhitegaussiannoisefsk(signal,SNR(j));
        recv_signal = symbols2bitsfsk(noisy_signal);
        msg = BCHDecoder(recv_signal,encoderType);
        msg = deconv(msg, g);
        msg = mod(msg, 2);
        message_final(i,:) = msg;
    end
error_matrix = mod((message-message_final),2);
disp(size(error_matrix));
count = numel(find (error_matrix == 1));
bit_error_rate = count/(2*10^5);
ber(:,j) = bit_error_rate;
end
semilogy(SNR, ber, color);
end
```