```
% 19ucc023
% Mohit Akhouri
% Observation 2 - Performance analysis of convolutional encoded system

% In this code , we will do the performance analysis of convolutional
% encoded system and we plot the BER vs. SNR(dB)

clc;
clear all;
close all;

n = 1000000; % Size of random sequence of 0's and 1's

rate_1 = 1/2; % Rate for 1/2-Convolutional Encoder
rate_2 = 1/3; % Rate for 1/3-Convolutional Encoder

data = randi([0,1],1,n); % Generating Random Sequence of 0's and 1's

trellis_1_by_2 = poly2trellis(3,[7 5]);  % Trellis Structure of 1/2
 Convolutional Encoder
trellis_1_by_3 = poly2trellis(3,[4 5 7]); % Trellis Structure of 1/3
 Convolutional Encoder

codeword_1_by_2 = convenc(data,trellis_1_by_2); % Codeword generated
 for 1/2 Convolutional Encoder
codeword_1_by_3 = convenc(data,trellis_1_by_3); % Codeword generated
 for 1/3 Convolutional Encoder

max_SNR = 10; % Range for Maximum SNR value
BER_coded_using_1_by_2 = zeros(1,max_SNR); % BER in case of 1/2
 Convolutional Encoder
BER_coded_using_1_by_3 = zeros(1,max_SNR); % BER in case of 1/3
 Convolutional Encoder

BER_uncoded = zeros(1,max_SNR); % BER in case of uncoded system
SNR_dB = zeros(1,max_SNR); % Array to store the SNR values ( in dB )

% Main loop algorithm for calculation of BPSK Modulated waveform and
 Bit
% error rates for two types of convolutional encoders
for i = 1:max_SNR
    SNR_dB(i) = i; % SNR value (without any units)
    SNR = 10^(i/10); % SNR value (in dB)

    % Calculation of sigma factor for calculation of Noise
    Sigma_1 = sqrt(1/(2*rate_1*SNR));
    Sigma_2 = sqrt(1/(2*rate_2*SNR));
    % Calculation of Noise in case of BPSK Modulation
    Noise_1 = Sigma_1*randn(1,n/rate_1);
    Noise_2 = Sigma_2*randn(1,n/rate_2);

    % Computing Transmitted and Received codewords in case of BPSK
```

```
    % Modulation for two types of encoder - 1/2 convolutional encoder
 and
    % 1/3 convolutional encoder
    transmitted_1 = 2*codeword_1_by_2 -1;
    received_1 = transmitted_1 + Noise_1;

    transmitted_2 = 2*codeword_1_by_3 -1;
    received_2 = transmitted_2 + Noise_2;

    % Detection of received codeword via HARD DECISION DECODER

    recovered_codeword_1_by_2 = (received_1>0); % Received codeword
 from 1/2 convolutional encoder
    recovered_codeword_1_by_3 = (received_2>0); % Received codeword
 from 1/3 convolutional encoder

    % Using 'vitdec' function to convolutionally decode binary data
    recovered_data_1_by_2 =
 vitdec(recovered_codeword_1_by_2,trellis_1_by_2,20,'trunc','hard');
    recovered_data_1_by_3 =
 vitdec(recovered_codeword_1_by_3,trellis_1_by_3,20,'trunc','hard');

    % Calculation of Net error rate in case of two types of encoders
    NER_hard_1_by_2 = sum(data~=recovered_data_1_by_2);
    NER_hard_1_by_3 = sum(data~=recovered_data_1_by_3);

    % Calculation of Bit error rates ( BER ) for both the encoders
    BER_coded_using_1_by_2(i) = NER_hard_1_by_2/n;
    BER_coded_using_1_by_3(i) = NER_hard_1_by_3/n;

    % Calculation of Bit error rate ( BER ) in case of uncoded system
    BER_uncoded(i) = qfunc(sqrt(SNR));
end

% Plots of BER vs. SNR (dB) for both types of encoders
figure;
semilogy(SNR_dB,BER_coded_using_1_by_2);
hold on;
semilogy(SNR_dB,BER_coded_using_1_by_3);
hold on;
semilogy(SNR_dB,BER_uncoded);
xlabel('SNR (dB) ->');
ylabel('BER ->');
title('19ucc023 - Mohit Akhouri','Plots of the BER vs. SNR(dB) for 1/2
 Convolutional Encoder and 1/3 Convolutional Encoder');
legend('BER in case of 1/2 Convolutional Encoder','BER in case of 1/3
 Convolutional Encoder','BER in case of uncoded system');
grid on;
hold off;
```

Published with MATLAB® R2020b