```
clc;
clear all;
close all;

M = 16; %Number of points in constellation is 16
%%x1 is an array of symbols to which mapping is to be done%%
%As we have b symbols 2 to the left of 0 and 2 to the right of zero.
x1 = [-3,-1,1,3];
const = x1 + 1i*x1.';
k=double(1.0)/double(sqrt(10)); %normalizing factor
const=k*const; %Normalising the constellation to make its power unity.

%This array is used to map between non-gray and gray constellation points.
maparr=[0 1 3 2 4 5 7 6 12 13 15 14 8 9 11 10];

inputs = 10000; %number of bits
input=zeros(1,inputs);
%generating 4bit random input between 0 and 15
for k=1:inputs
 input(k)= randi([0, (M-1)]);
end
input_withoutgray=const(input(:)+1); %constellation symbols for non gray
input_gray=maparr(input(:)+1);%corresponding gray input for the same constellation input.

snr = -10:10; %SNR from -10 to 10 dB in steps of 1.
decisions_bin = zeros(1,inputs);
number_snrs = length(snr);
%To estimate BER error for each SNR value and add it to estimate for both
%gray and non gray labelling
berr_estimate_withoutgray = zeros(number_snrs,1);
berr_estimate_gray = zeros(number_snrs,1);

%looping through each SNR value
for k=1:number_snrs
    snr_now = snr(k);
    ebno=10^(snr_now/10);
    sigma=sqrt(1/(ebno));
    % add 2d Gaussian noise to our symbols.
    received_withoutgray = input_withoutgray+ (sigma*randn(inputs,1)+1j*sigma*randn(inputs,1))/sqrt(10);% add complex AWGN noise to our input signal
    decisions=zeros(inputs,1);
    %calculating absolute distance of every signal point from each point of constellation.
    %The minimum distance constellation point is the signal.
    for n=1:inputs
        distancesbin = abs(received_withoutgray(n)-const);
        [min_dist_bin,decisions_bin(n)] = min(distancesbin(:));
    end
    %map the decoded signal back to gray code input to compare error for gray
    %decisions_bin are index values while they correspond to some
    %decisions_gray value.
    decisions_gray=maparr(decisions_bin);
```

```
    decisions_bin=decisions_bin-1;

    %To calculate bit error for both gray and non gray labelling
    num=zeros(1,inputs);

    %%calculating BER for 16QAM without gray labelling%%
    for s=1:length(input)
        d_bin=de2bi(decisions_bin(s),4); %4bit binary string for ease of comparing
        i_bin=de2bi(input(s),4);
        biterror=0;
        for t=1:4
            if d_bin(t)~=i_bin(t)
                biterror=biterror+1;  %%adding the error
            end
            num(s)=biterror; %%storing the total error
        end
    end
    error=num;

    %%calculating BER for 16QAM with gray labelling
    for s=1:length(input_gray)
        d_bin=de2bi(decisions_gray(s),4); %4bit binary string for ease of comparing
        i_bin=de2bi(input_gray(s),4);
        biterror=0;
        for t=1:4
            if d_bin(t)~=i_bin(t)
                biterror=biterror+1;  %%adding the error
            end
            num(s)=biterror; %%storing the total error
        end
    end
    error_gray=num;
    berr_estimate_withoutgray(k) =berr_estimate_withoutgray(k)+ sum(error)/inputs;
    berr_estimate_gray(k) =berr_estimate_gray(k)+ sum(error_gray)/inputs;

end


%Plotting the Bit Error Rate
figure;
semilogy(snr,berr_estimate_withoutgray,'o'); %plotting the BER without gray label.
hold on;
semilogy(snr,berr_estimate_gray,'*'); %plotting the BER with gray label.
hold on;
semilogy(snr,3*qfunc(sqrt((10.^(snr/10))))); % plotting BER theoretical using Q-↙
function .
legend("Experimental BER without gray code","Experimental BER with gray code","↙
Theoretical using Q function");
xlabel("SNR (dB)");
ylabel("BER (Bit Error Rate .)");
title("BER plot for 16 QAM with and without gray-labelling");
```