Table of Contents

D-QPSK Modulator/Demodulator Example

This documents describes/implements the QPSK modulation and demodulation of a song signal.

```
Prepared for ELEC 301

by Beyzanur Çoban 64763

*01.04.2020*
```

Program Initialization

```
%Clear Variables and Close All Figure Windows

% Clear all previous variables
clear
% Close all previous figure windows
close all
```

Read and Display an Example Image

cameraman.tif is an example gray-level image provided my matlab

Load the Cameraman Image

```
Im = imread('cameraman.tif');
% Extract part of the image
Im=Im(51:100,101:150);
```

Display the image

```
imshow(Im);
```

Convert Image to a Binary Vector

We need to convert the image to a binary bit sequence

Convert 256x256 image matrix to an image (column) vector (of size 256^2x1) by concatenating columns

```
Imv=Im(:);
```

Convert each the number in each row to a binary vector

```
Imvb=de2bi(Imv);
```

Note that **Imvb** has size 256^2x8

Now generate a row vector containing all bits

```
Imvbt=Imvb';
s=Imvbt(:)';
```

Generate Modulated Signal

D-QPSK Modulated Signal

From the single bit sequence generate a vector sequence

```
sv=[s(1:2:end);
    s(2:2:end)];
```

D-QPSK Constellation Mapper (Question 5)

[0;0]-> c(n-1)*exp(i*pi/2) [0;1]-> -c(n-1) [1;0]-> c(n-1) [1;1]-> c(n-1)*exp(-i*pi/2)

```
% I initialized the first value because in every loop, the phase
 should
% change based on the first value
c(1) = -1+1i;
for k=2:size(sv,2)
    switch num2str(sv(:,k)')
        case '0  0'
            c(k)=c(k-1)*exp(1i*pi/2);
        case '1  0'
```

```
            c(k)=c(k-1);
        case '0  1'
            c(k)=-c(k-1);
        otherwise
            c(k)=c(k-1)*exp(-1i*pi/2);
    end
end
% Normalize the power to 1
c=c/sqrt(2);
```

Rectangle Modulation

```
% Sample Rate
Fsampling=2^19;
% Sample Intervale
Tsampling=1/Fsampling;
% Symbol Rate
Fsymbol=2^13;
% Symbol Period
Tsymbol=1/Fsymbol;
% Number of Samples per Symbol Period
Ns=Tsymbol/Tsampling;
```

Baseband Signal (samples)

```
xb=kron(c,ones(1,Ns));
```

Carrier frequency:

$f_c = 60kHz$

```
fc=60e3; % 60 kHz;
```

Carrier signal: _

$c(t) = cos(2\pi f_c t)$

```
t=(0:1:(length(xb)-1))*Tsampling;
cost=cos(2*pi*fc*t);
sint=sin(2*pi*fc*t);
```

Transmitter output

$x(t) = Re(xb(t))cos(2\pi f_c t) - Im(xb(t))sin(2\pi f_c t)$

```
x=real(xb).*cost-imag(xb).*sint;
```

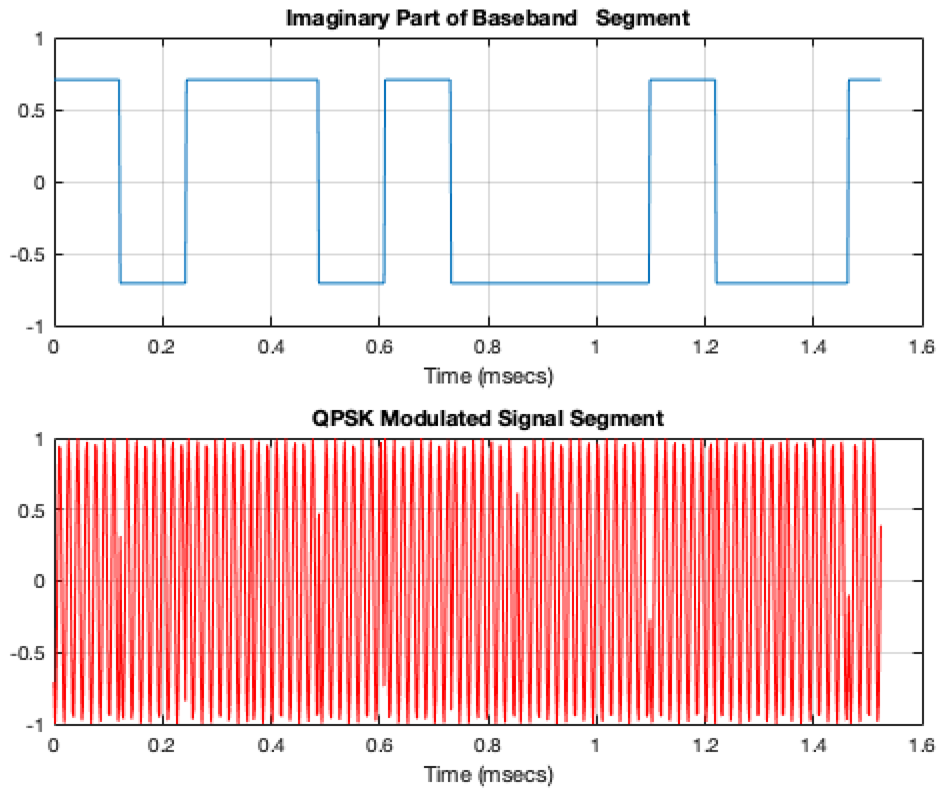
Display the Segments of Baseband Signal and Modulated Signal

Display small section of the original signal and then the DSB-SC modulated version

```
figure(2)
% Segment Length
```

```
SL=800
% plot the  segment of imaginary component (for SL samples)
subplot(2,1,1)
plot(t(1:SL)*1000,  imag(xb(1:SL)));
xlabel('Time (msecs)')
title('Imaginary Part of Baseband   Segment')
grid

subplot(2,1,2)
% plot the modulated signal
plot(t(1:SL)*1000,x(1:SL),'r');
hold on

xlabel('Time (msecs)')
grid
title('QPSK Modulated Signal Segment')
```

```
SL =

   800
```



Channel Effect

We add some noise

First calculate average signal energy (per sample)

```
sigpow=mean(x.^2);
```

Define SNR level in (dB)

```
SNR=10;
```

Noise Level

```
NoiseAmp=sqrt(10^(-SNR/10)*sigpow);
```

Generate Noise signal as Gaussian Noise

```
noise=NoiseAmp*randn(1,length(x));
```

Noisy received signal

$$y(t) = x(t) + n(t)$$

```
y=x+noise;
```

The D-QPSK Receiver Processing (Question 5 for SNR=10)

D-QPSK Receiver operation

First extract real component baseband signals for each receiver oscillator phase error

$$u_r(t) = 2x(t)cos(2\pi f_c t)$$

```
ur1=2*y.*cos(2*pi*fc*t);
ur2=2*y.*cos((2*pi*fc*t) + (pi/10));
ur3=2*y.*cos((2*pi*fc*t) + (pi/5));
ur4=2*y.*cos((2*pi*fc*t) + (3*pi/10));
ur5=2*y.*cos((2*pi*fc*t) + (2*pi/5));
ur6=2*y.*cos((2*pi*fc*t) + (pi/2));
```

Then low pass filter these signals

$$z_r(t) = u_r(t) * h_{LP}(t)$$

```
zr1 = lowpass(ur1,30e3,Fsampling);
zr2 = lowpass(ur2,30e3,Fsampling);
zr3 = lowpass(ur3,30e3,Fsampling);
zr4 = lowpass(ur4,30e3,Fsampling);
zr5 = lowpass(ur5,30e3,Fsampling);
zr6 = lowpass(ur6,30e3,Fsampling);
```

Then extract the imaginary component baseband signals for each different receiver oscillator phase error

$$u_i(t) = 2x(t)sin(2\pi f_c t)$$

```
ui1=-2*y.*sin(2*pi*fc*t);
ui2=-2*y.*sin((2*pi*fc*t) + (pi/10));
ui3=-2*y.*sin((2*pi*fc*t) + (pi/5));
ui4=-2*y.*sin((2*pi*fc*t) + (3*pi/10));
ui5=-2*y.*sin((2*pi*fc*t) + (2*pi/5));
ui6=-2*y.*sin((2*pi*fc*t) + (pi/2));
```

Then low pass filter these signals

$z_i(t) = u_i(t) * h_{LP}(t)$

```
zi1 = lowpass(ui1,30e3,Fsampling);
zi2 = lowpass(ui2,30e3,Fsampling);
zi3 = lowpass(ui3,30e3,Fsampling);
zi4 = lowpass(ui4,30e3,Fsampling);
zi5 = lowpass(ui5,30e3,Fsampling);
zi6 = lowpass(ui6,30e3,Fsampling);
```

Basband signals

```
z1=zr1+1i*zi1;
z2=zr2+1i*zi2;
z3=zr3+1i*zi3;
z4=zr4+1i*zi4;
z5=zr5+1i*zi5;
z6=zr6+1i*zi6;
```

Fourier Transforms of Baseband, Modulated and Demodulated Signals

Calculate and Display the Fourier Transforms of the Baseband,modulated and demodulated signals

Calculate the Fourier Transform of the baseband signal

```
[ftxb,freqs]=fouriertransform(xb, Fsampling);
```

Calculate the Fourier Transform of the passband signal

```
[ftx,freqs]=fouriertransform(x,Fsampling);
```

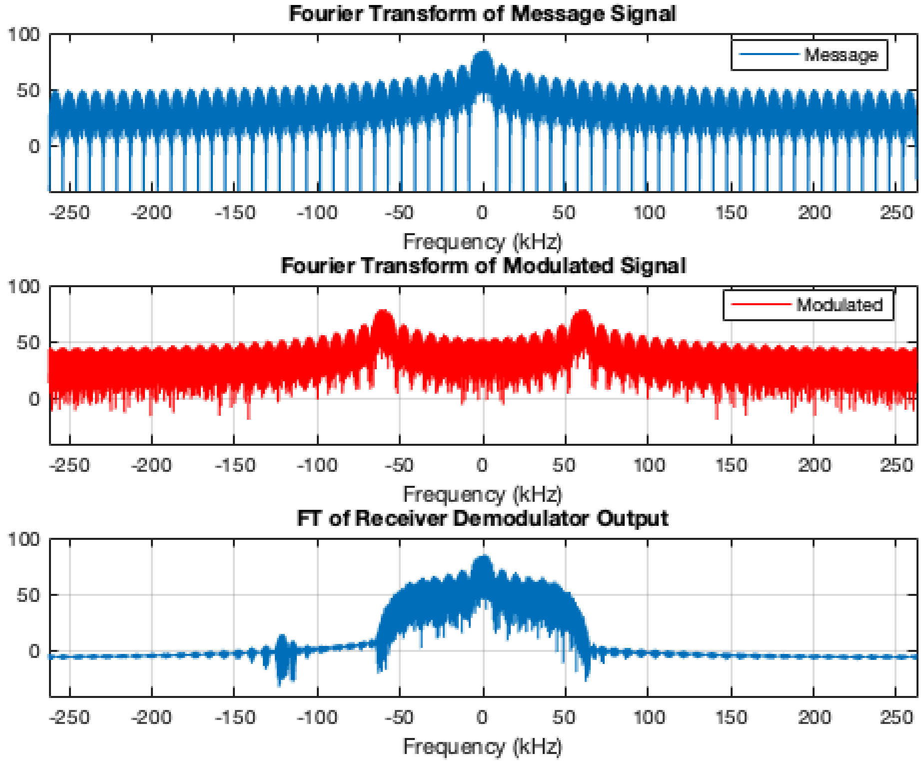
Calculate Fourier Transform of the receiver baseband

```
[ftz1,freqs]=fouriertransform(z1,Fsampling);
[ftz2,freqs]=fouriertransform(z2,Fsampling);
[ftz3,freqs]=fouriertransform(z3,Fsampling);
[ftz4,freqs]=fouriertransform(z4,Fsampling);
[ftz5,freqs]=fouriertransform(z5,Fsampling);
[ftz6,freqs]=fouriertransform(z6,Fsampling);
```

Display these Fourier Transforms

```
figure(3)
subplot(3,1,1);
```

```
plot(freqs/1000, 20*log10(abs(ftxb)));
axis([-Fsampling/2000 Fsampling/2000 -40 100])
legend('Message','Location','Best')
xlabel('Frequency (kHz)');
title('Fourier Transform of Message Signal')
subplot(3,1,2)
plot(freqs/1000, 20*log10(abs(ftx)),'r');
grid
legend('Modulated','Location','Best')
xlabel('Frequency (kHz)');
title('Fourier Transform of Modulated Signal')
axis([-Fsampling/2000 Fsampling/2000 -40 100])
subplot(3,1,3)
plot(freqs/1000, 20*log10(abs(ftz1)));
axis([-Fsampling/2000 Fsampling/2000 -40 100])
grid
xlabel('Frequency (kHz)')
title('FT of Receiver Demodulator Output')
```

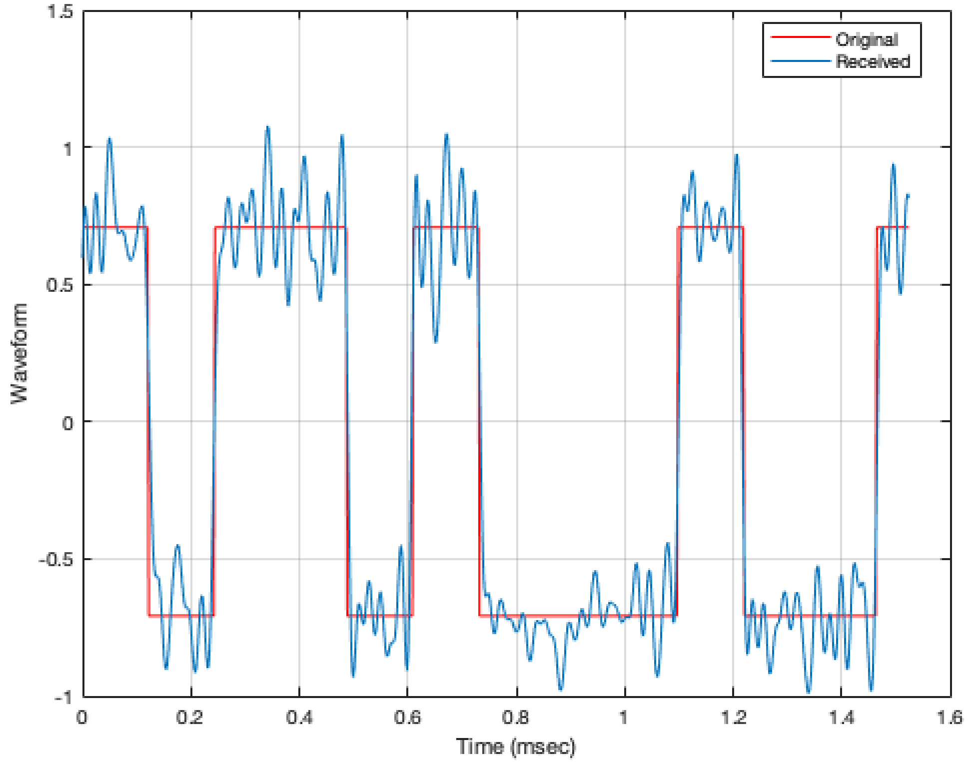


Display the Original Song and the Receiver Output Segments

Can you feel the noise?

Comparing the imaginary components of transmitted and received baseband signal segments

```
figure(4)
plot(t(1:SL)*1000,imag(xb(1:SL)),'r')
hold on
plot(t(1:SL)*1000,imag(z1(1:SL)))
grid
xlabel('Time (msec)');
ylabel('Waveform');
legend('Original','Received','Location','Best');
```

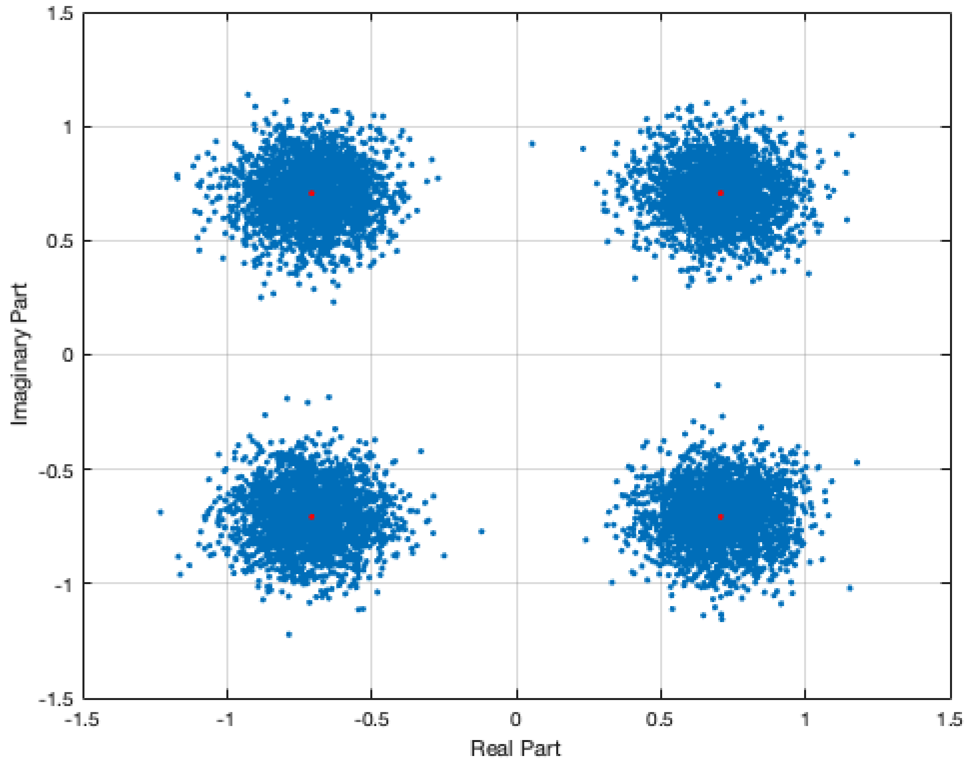


Constellation Estimates

We sample the baseband received signal to get noisy estimates of transmitted constellation point.

```
ce1=z1(ceil(Ns/2):Ns:length(z1));
ce2=z2(ceil(Ns/2):Ns:length(z2));
ce3=z3(ceil(Ns/2):Ns:length(z3));
ce4=z4(ceil(Ns/2):Ns:length(z4));
ce5=z5(ceil(Ns/2):Ns:length(z5));
ce6=z6(ceil(Ns/2):Ns:length(z6));

figure(5)
% Plot constellation estimates
plot(real(ce1),imag(ce1),'.');
hold on
p=plot(real(c),imag(c),'r.');
set(p,'MarkerSize',5)
```

```
xlabel('Real Part');
ylabel('Imaginary Part');
grid
```



Bit Estimates

We implement D-QPSK Demapper to extract bits from constellation estimates

Check which quadrant ce's lies in

```
ser1(1) = 0;
sei1(1) = 0;
for n=2:length(ce1)
    theta = angle(ce1(n)*conj(ce1(n-1)));
    if ((theta <= pi/4) && (theta >-pi/4))
        ser1(n) = 1;
        sei1(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser1(n) = 0;
        sei1(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser1(n) = 0;
        sei1(n) = 1;
    elseif ((theta <=-pi/4) && (theta > -3*pi/4))
        ser1(n) = 1;
        sei1(n) = 1;
```

```
    end

end
se1(1:2:(2*length(ser1)))=ser1;
se1(2:2:(2*length(ser1)))=sei1;

se1l=0~=se1;


for n=2:length(ce2)
    theta = angle(ce2(n)*conj(ce2(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser2(n) = 1;
        sei2(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser2(n) = 0;
        sei2(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser2(n) = 0;
        sei2(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser2(n) = 1;
        sei2(n) = 1;
    end

end
se2(1:2:(2*length(ser2)))=ser2;
se2(2:2:(2*length(ser2)))=sei2;

se2l=0~=se2;

for n=2:length(ce3)
    theta = angle(ce3(n)*conj(ce3(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser3(n) = 1;
        sei3(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser3(n) = 0;
        sei3(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser3(n) = 0;
        sei3(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser3(n) = 1;
        sei3(n) = 1;
    end

end
se3(1:2:(2*length(ser3)))=ser3;
se3(2:2:(2*length(ser3)))=sei3;

se3l=0~=se3;

for n=2:length(ce4)
```

```
    theta = angle(ce4(n)*conj(ce4(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser4(n) = 1;
        sei4(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser4(n) = 0;
        sei4(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser4(n) = 0;
        sei4(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser4(n) = 1;
        sei4(n) = 1;
    end

end
se4(1:2:(2*length(ser4)))=ser4;
se4(2:2:(2*length(ser4)))=sei4;

se4l=0~=se4;

for n=2:length(ce5)
    theta = angle(ce5(n)*conj(ce5(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser5(n) = 1;
        sei5(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser5(n) = 0;
        sei5(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser5(n) = 0;
        sei5(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser5(n) = 1;
        sei5(n) = 1;
    end

end
se5(1:2:(2*length(ser5)))=ser5;
se5(2:2:(2*length(ser5)))=sei5;

se5l=0~=se5;

for n=2:length(ce6)
    theta = angle(ce6(n)*conj(ce6(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser6(n) = 1;
        sei6(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser6(n) = 0;
        sei6(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser6(n) = 0;
        sei6(n) = 1;
```

```
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser6(n) = 1;
        sei6(n) = 1;
    end

end
se6(1:2:(2*length(ser6)))=ser6;
se6(2:2:(2*length(ser6)))=sei6;

se6l=0~=se6;
```

Calculate Bit Error Rates

```
BER1=sum(se1~=s)/length(s)
BER2=sum(se2~=s)/length(s)
BER3=sum(se3~=s)/length(s)
BER4=sum(se4~=s)/length(s)
BER5=sum(se5~=s)/length(s)
BER6=sum(se6~=s)/length(s)

% I don't know why all of the BER values for different phase errors
 are the
% same
```

```
BER1 =

    0.1372


BER2 =

    0.1372


BER3 =

    0.1372


BER4 =

    0.1372


BER5 =

    0.1372


BER6 =

    0.1372
```

Array of receiver oscillator phase error

```
phase_error = [0 pi/10 pi/5 3*pi/10 2*pi/5 pi/2];
```

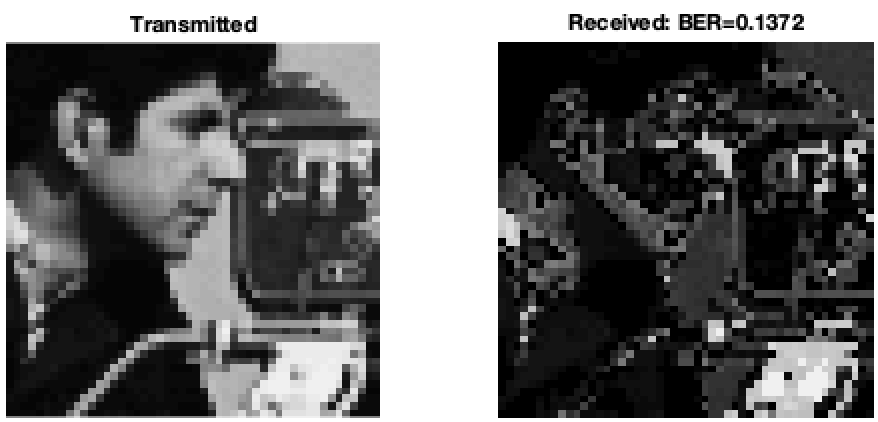
Array of bit error rates for SNR=10

```
BERs10 = [BER1 BER2 BER3 BER4 BER5 BER6];
```

Reconstruct Image

From the bits we estimated, we reconstruct 8-bit gray level image I could not figure out why my reconstructed image is not properly reconstructed

```
Imvbe=reshape(se1l,8,length(s)/8)';
% Vectorized image estimate in decimals
Imve=bi2de(Imvbe);
% Image estimate in matrix form
Ime=reshape(Imve,50,50);
figure(6)
subplot(1,2,1)
imshow(Im)
title('Transmitted')
subplot(1,2,2)

imshow(uint8(Ime))
title(['Received: BER=' num2str(BER1)])
```



LOW SNR CASE (Question 5 for SNR=1)

Define new SNR level in (dB)

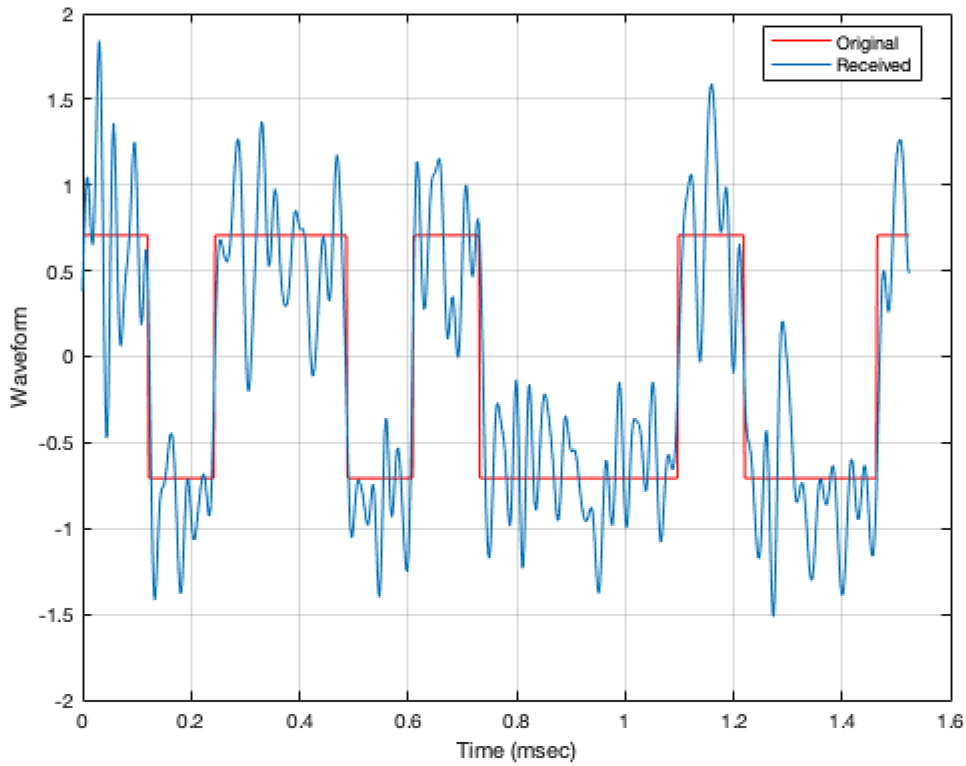
```
SNR=1;
```

```
% Noise Level
NoiseAmp=sqrt(10^(-SNR/10)*sigpow);
% Generate Noise signal as Gaussian Noise
noise=NoiseAmp*randn(1,length(x));
% Noisy received signal
y=x+noise;
% The D-QPSK Receiver Processing
% First extract real component baseband signals
% $u_r(t)=2x(t)cos(2\pi f_c t)$
ur1=2*y.*cos(2*pi*fc*t);
ur2=2*y.*cos((2*pi*fc*t) + (pi/10));
ur3=2*y.*cos((2*pi*fc*t) + (pi/5));
ur4=2*y.*cos((2*pi*fc*t) + (3*pi/10));
ur5=2*y.*cos((2*pi*fc*t) + (2*pi/5));
ur6=2*y.*cos((2*pi*fc*t) + (pi/2));
% Then low pass filter these signals
zr1 = lowpass(ur1,30e3,Fsampling);
zr2 = lowpass(ur2,30e3,Fsampling);
zr3 = lowpass(ur3,30e3,Fsampling);
zr4 = lowpass(ur4,30e3,Fsampling);
zr5 = lowpass(ur5,30e3,Fsampling);
zr6 = lowpass(ur6,30e3,Fsampling);
% Then extract the imaginary component baseband signals
ui1=-2*y.*sin(2*pi*fc*t);
ui2=-2*y.*sin((2*pi*fc*t) + (pi/10));
ui3=-2*y.*sin((2*pi*fc*t) + (pi/5));
ui4=-2*y.*sin((2*pi*fc*t) + (3*pi/10));
ui5=-2*y.*sin((2*pi*fc*t) + (2*pi/5));
ui6=-2*y.*sin((2*pi*fc*t) + (pi/2));
% Then low pass filter these signals
zi1 = lowpass(ui1,30e3,Fsampling);
zi2 = lowpass(ui2,30e3,Fsampling);
zi3 = lowpass(ui3,30e3,Fsampling);
zi4 = lowpass(ui4,30e3,Fsampling);
zi5 = lowpass(ui5,30e3,Fsampling);
zi6 = lowpass(ui6,30e3,Fsampling);
% Basband signals
z1=zr1+1i*zi1;
z2=zr2+1i*zi2;
z3=zr3+1i*zi3;
z4=zr4+1i*zi4;
z5=zr5+1i*zi5;
z6=zr6+1i*zi6;
```

Display the Original Song and the Receiver Output Segments

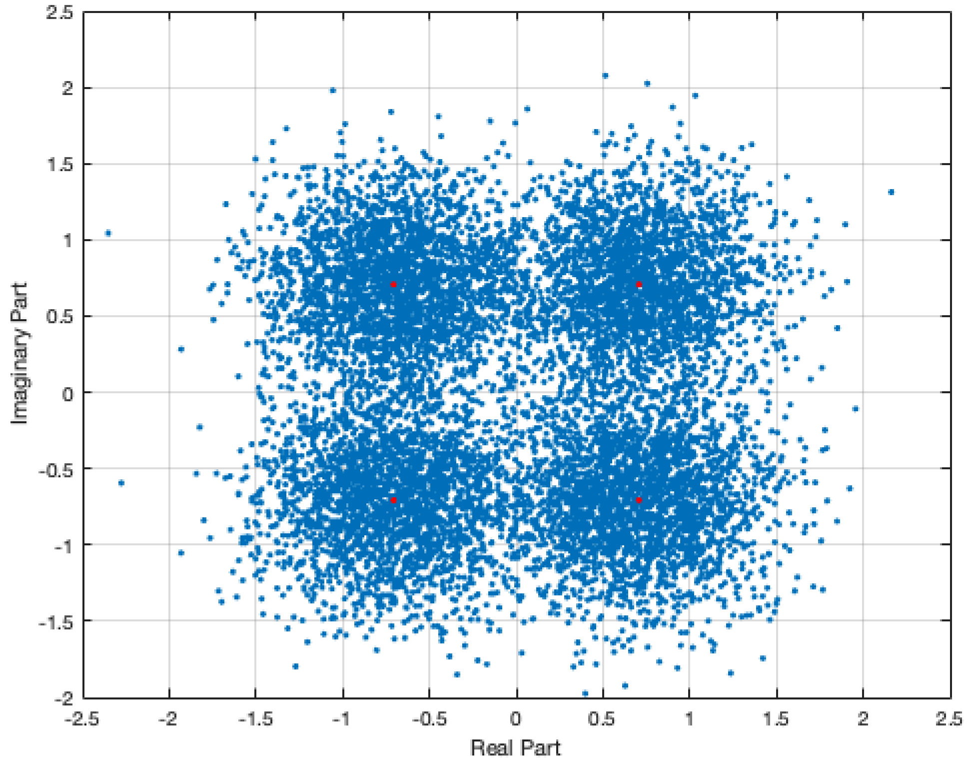
Comparing the imaginary components of transmitted and received baseband signal segments

```
figure(7)
plot(t(1:SL)*1000,imag(xb(1:SL)),'r')
hold on
plot(t(1:SL)*1000,imag(z1(1:SL)))
grid
xlabel('Time (msec)');
ylabel('Waveform');
legend('Original','Received','Location','Best');
```



Constellation Estimates

```
ce1=z1(ceil(Ns/2):Ns:length(z1));
ce2=z2(ceil(Ns/2):Ns:length(z2));
ce3=z3(ceil(Ns/2):Ns:length(z3));
ce4=z4(ceil(Ns/2):Ns:length(z4));
ce5=z5(ceil(Ns/2):Ns:length(z5));
ce6=z6(ceil(Ns/2):Ns:length(z6));

figure(8)
% Plot constellation estimates
plot(real(ce1),imag(ce1),'.');
hold on
p=plot(real(c),imag(c),'r.');
set(p,'MarkerSize',5)
xlabel('Real Part');
ylabel('Imaginary Part');
grid
```



Bit Estimates

Check which quadrant ce's lies in

```
for n=2:length(ce1)
    theta = angle(ce1(n)*conj(ce1(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser1(n) = 1;
        sei1(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser1(n) = 0;
        sei1(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser1(n) = 0;
        sei1(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser1(n) = 1;
        sei1(n) = 1;
    end

end
se1(1:2:(2*length(ser1)))=ser1;
se1(2:2:(2*length(ser1)))=sei1;

se1l=0~=se1;
```

```
for n=2:length(ce2)
    theta = angle(ce2(n)*conj(ce2(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser2(n) = 1;
        sei2(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser2(n) = 0;
        sei2(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser2(n) = 0;
        sei2(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser2(n) = 1;
        sei2(n) = 1;
    end

end
se2(1:2:(2*length(ser2)))=ser2;
se2(2:2:(2*length(ser2)))=sei2;

se2l=0~=se2;


for n=2:length(ce3)
    theta = angle(ce3(n)*conj(ce3(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser3(n) = 1;
        sei3(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser3(n) = 0;
        sei3(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser3(n) = 0;
        sei3(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser3(n) = 1;
        sei3(n) = 1;
    end

end
se3(1:2:(2*length(ser3)))=ser3;
se3(2:2:(2*length(ser3)))=sei3;

se3l=0~=se3;


for n=2:length(ce4)
    theta = angle(ce4(n)*conj(ce4(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser4(n) = 1;
        sei4(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser4(n) = 0;
        sei4(n) = 0;
```

```
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser4(n) = 0;
        sei4(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser4(n) = 1;
        sei4(n) = 1;
    end

end
se4(1:2:(2*length(ser4)))=ser4;
se4(2:2:(2*length(ser4)))=sei4;

se4l=0~=se4;

for n=2:length(ce5)
    theta = angle(ce5(n)*conj(ce5(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser5(n) = 1;
        sei5(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser5(n) = 0;
        sei5(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser5(n) = 0;
        sei5(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser5(n) = 1;
        sei5(n) = 1;
    end

end
se5(1:2:(2*length(ser5)))=ser5;
se5(2:2:(2*length(ser5)))=sei5;

se5l=0~=se5;

for n=2:length(ce6)
    theta = angle(ce6(n)*conj(ce6(n-1)));
    if ((theta <= pi/4) && (theta > -pi/4))
        ser6(n) = 1;
        sei6(n) = 0;
    elseif ((theta <= 3*pi/4) && (theta > pi/4))
        ser6(n) = 0;
        sei6(n) = 0;
    elseif ((theta <= -3*pi/4) && (theta > 3*pi/4))
        ser6(n) = 0;
        sei6(n) = 1;
    elseif ((theta <= -pi/4) && (theta > -3*pi/4))
        ser6(n) = 1;
        sei6(n) = 1;
    end

end
se6(1:2:(2*length(ser6)))=ser6;
```

```
se6(2:2:(2*length(ser6)))=sei6;

se6l=0~=se6;
```

Calculate Bit Error Rates

```
BER1=sum(se1~=s)/length(s)
BER2=sum(se2~=s)/length(s)
BER3=sum(se3~=s)/length(s)
BER4=sum(se4~=s)/length(s)
BER5=sum(se5~=s)/length(s)
BER6=sum(se6~=s)/length(s)

% I don't know why all of the BER values for different phase errors
 are the
% same
```

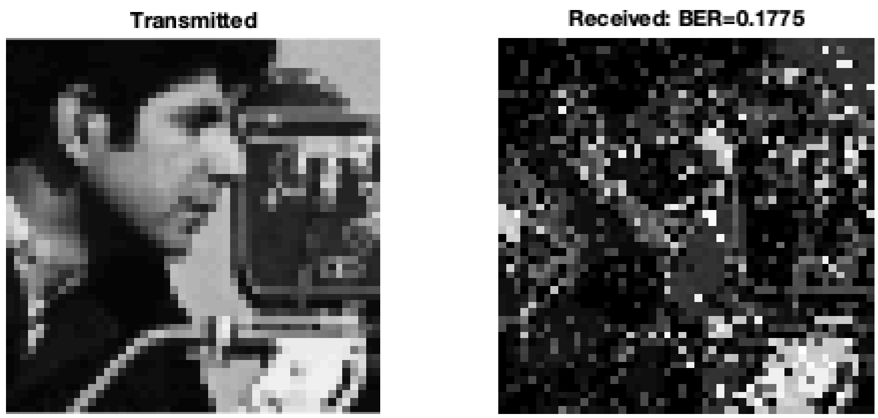
```
BER1 =

    0.1775


BER2 =

    0.1775


BER3 =

    0.1775


BER4 =

    0.1775


BER5 =

    0.1775


BER6 =

    0.1775
```

Array of bit error rates for SNR=1

```
BERs1 = [BER1 BER2 BER3 BER4 BER5 BER6];
```

Reconstruct Image From the bits we estimated, we reconstruct 8-bit gray level image I could not figure out why my reconstructed image is not properly reconstructed

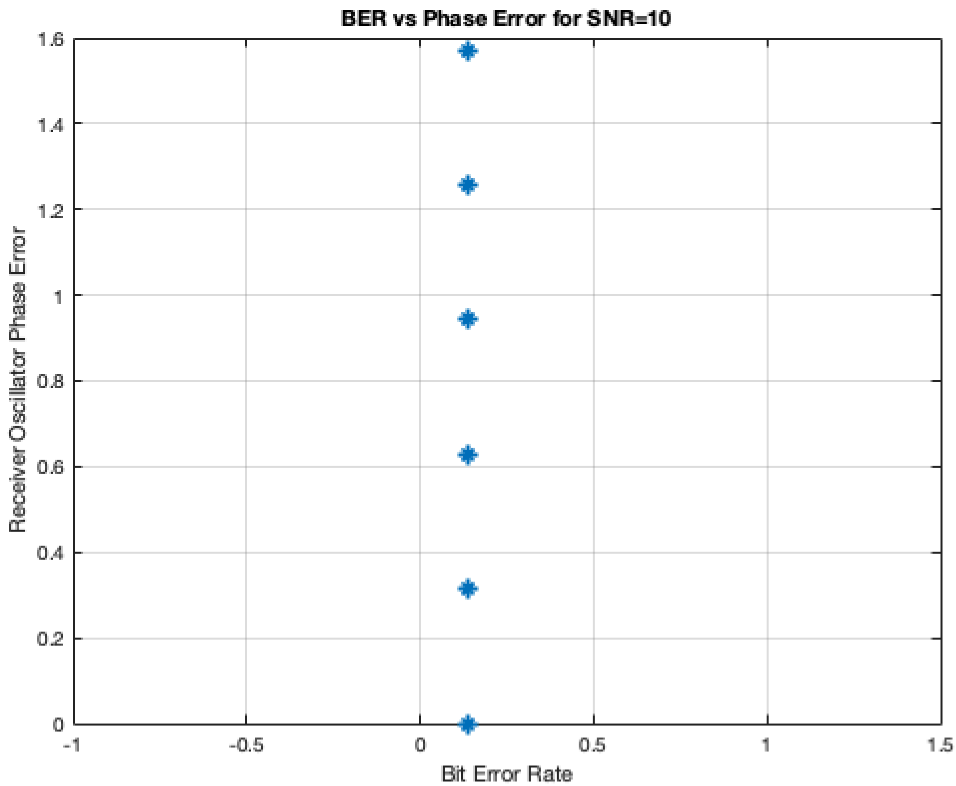
```
Imvbe=reshape(se1,8,length(s)/8)';
% Vectorized image estimate in decimals
Imve=bi2de(Imvbe);
% Image estimate in matrix form
Ime=reshape(Imve,50,50);
figure(9)
subplot(1,2,1)
imshow(Im)
title('Transmitted')
subplot(1,2,2)

imshow(uint8(Ime))
title(['Received: BER=' num2str(BER1)])
```



Question 5 for SNR=10 (Graph)

BER vs. Phase Error Plot for SNR=10

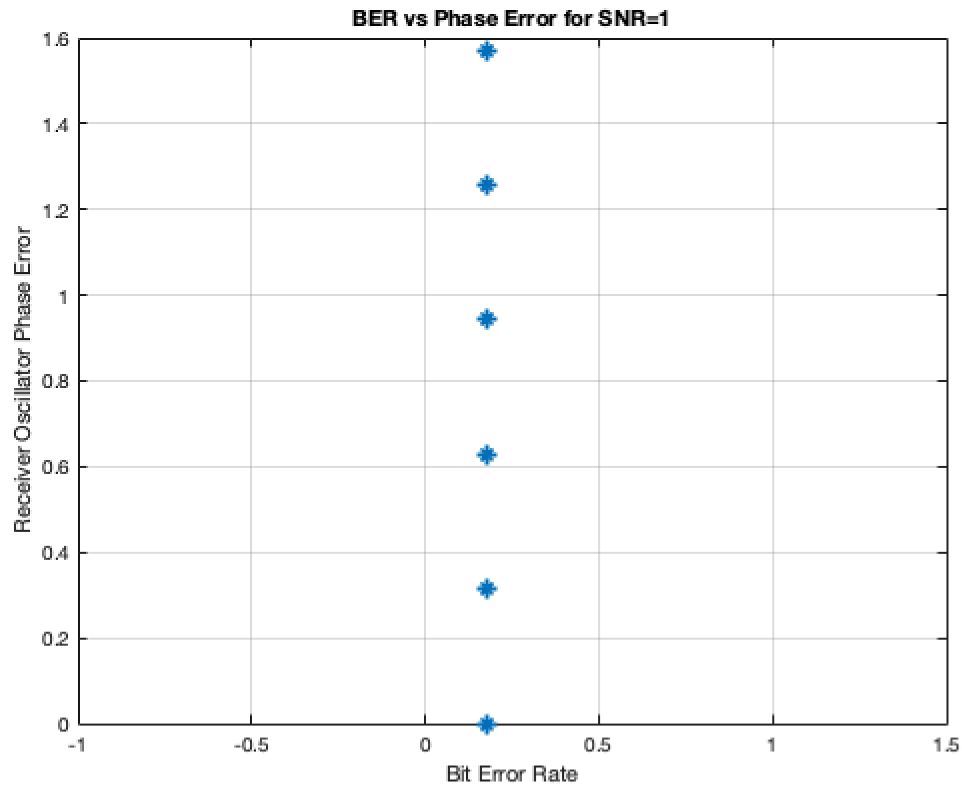
```
figure(10)
plot(BERs10, phase_error, '*')
grid
xlabel('Bit Error Rate')
ylabel('Receiver Oscillator Phase Error')
title('BER vs Phase Error for SNR=10')
```



Question 5 for SNR=1 (Graph)

BER vs. Phase Error Plot for SNR=1

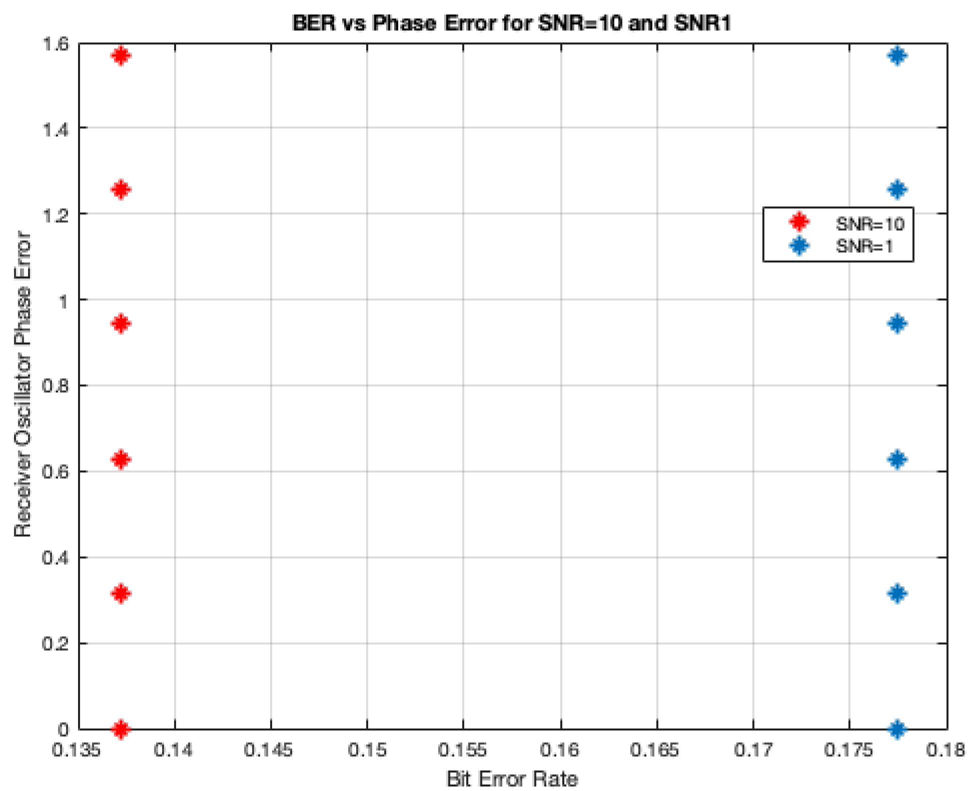
```
figure(11)
plot(BERs1, phase_error, '*')
grid
xlabel('Bit Error Rate')
ylabel('Receiver Oscillator Phase Error')
title('BER vs Phase Error for SNR=1')
```



Question 5 Comparison Graph

Comparison on the same graph

```
figure(12)
plot(BERs10, phase_error, 'r*')
hold on
plot(BERs1, phase_error, '*')
grid
xlabel('Bit Error Rate')
ylabel('Receiver Oscillator Phase Error')
title('BER vs Phase Error for SNR=10 and SNR1')
legend('SNR=10','SNR=1','Location','Best');
```



Published with MATLAB® R2018b