Table of Contents

QPSK Modulator/Demodulator Example

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

```
            Prepared for ELEC 301

           by Alper T. Erdogan

                  *16.03.2020*
```

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

QPSK Modulated Signal

From the single bit sequence generate a vector sequence

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

QPSK Constellation Mapper [0;0]-> -1-i [0;1]-> -1+i [1;0]-> 1-i [1;1]-> 1+i

```
for k=1:size(sv,2)
    switch num2str(sv(:,k)')
        case '0  0'
            c(k)=-1-i;
        case '1  0'
            c(k)=1-i;
        case '0  1'
            c(k)=-1+i;
        otherwise
            c(k)=1+i;
        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;
```

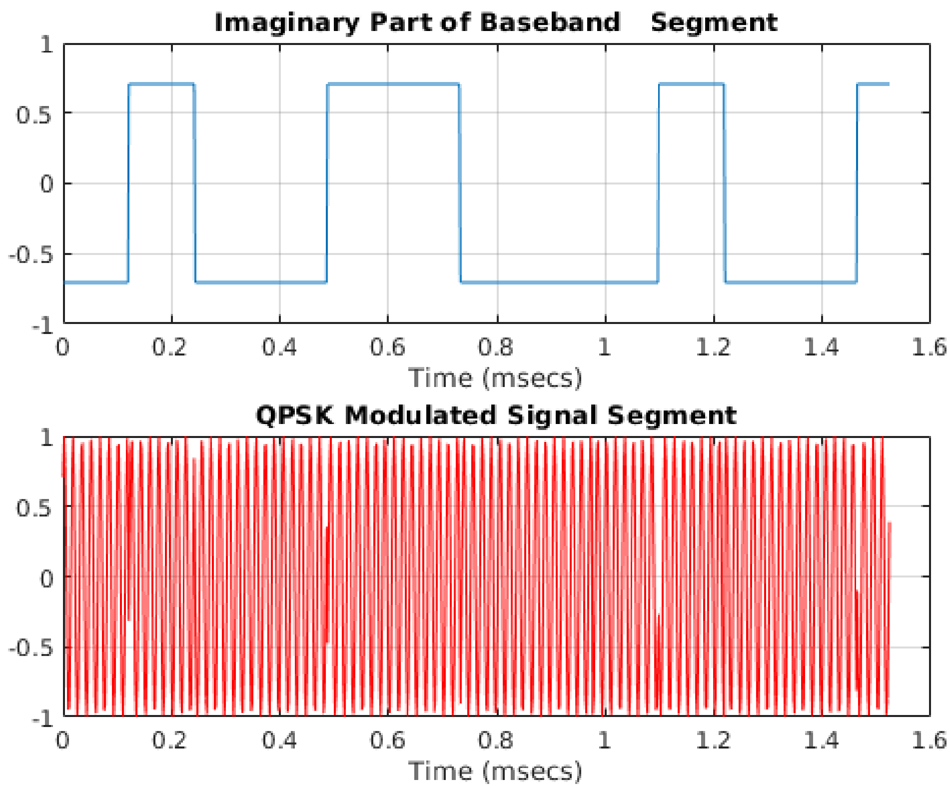
Display the Segments of Baseband Signal and Modulated Signal

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 QPSK Receiver Processing

Coherent QPSK Receiver operation

First extract real component baseband signal

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

```
ur=2*y.*cost;
```

Then low pass filter this signal

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

```
zr = lowpass(ur,30e3,Fsampling);
```

Then extract the imaginary component baseband signal

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

```
ui=-2*y.*sint;
```

Then low pass filter this signal

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

```
zi = lowpass(ui,30e3,Fsampling);
```

Basband signal

```
z=zr+i*zi;
```

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

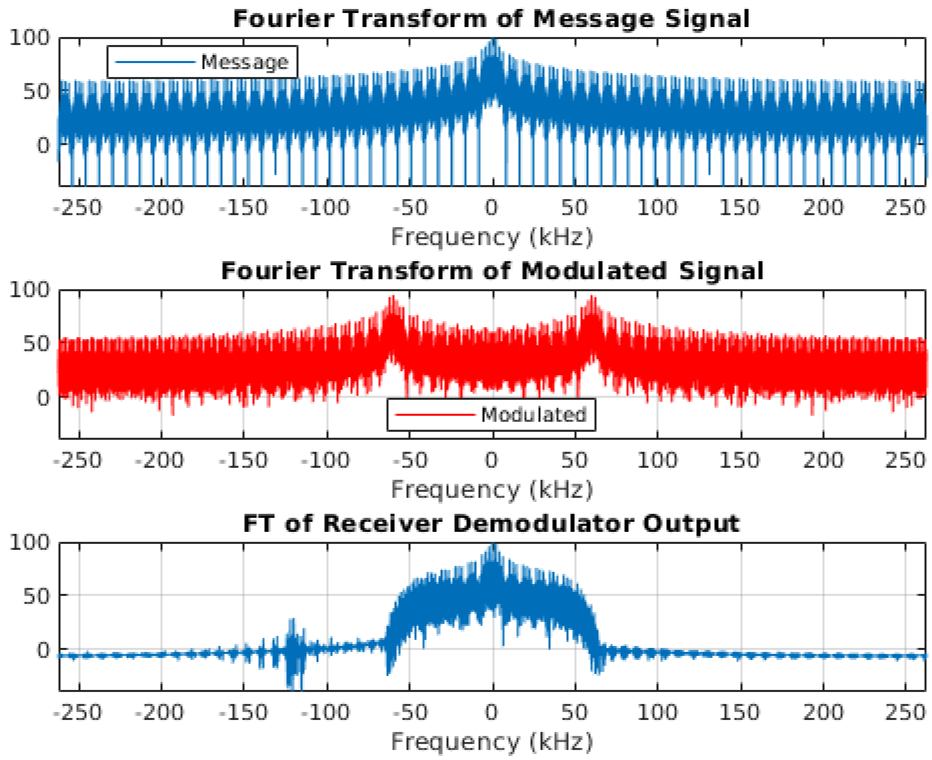
```
[ftx,freqs]=fouriertransform(x,Fsampling);
```

Calculate Fourier Transform of the receiver baseband

```
[ftz,freqs]=fouriertransform(z,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(ftz)));
axis([-Fsampling/2000 Fsampling/2000 -40 100])
grid
xlabel('Frequency (kHz)')
title('FT of Receiver Demodulator Output')
```

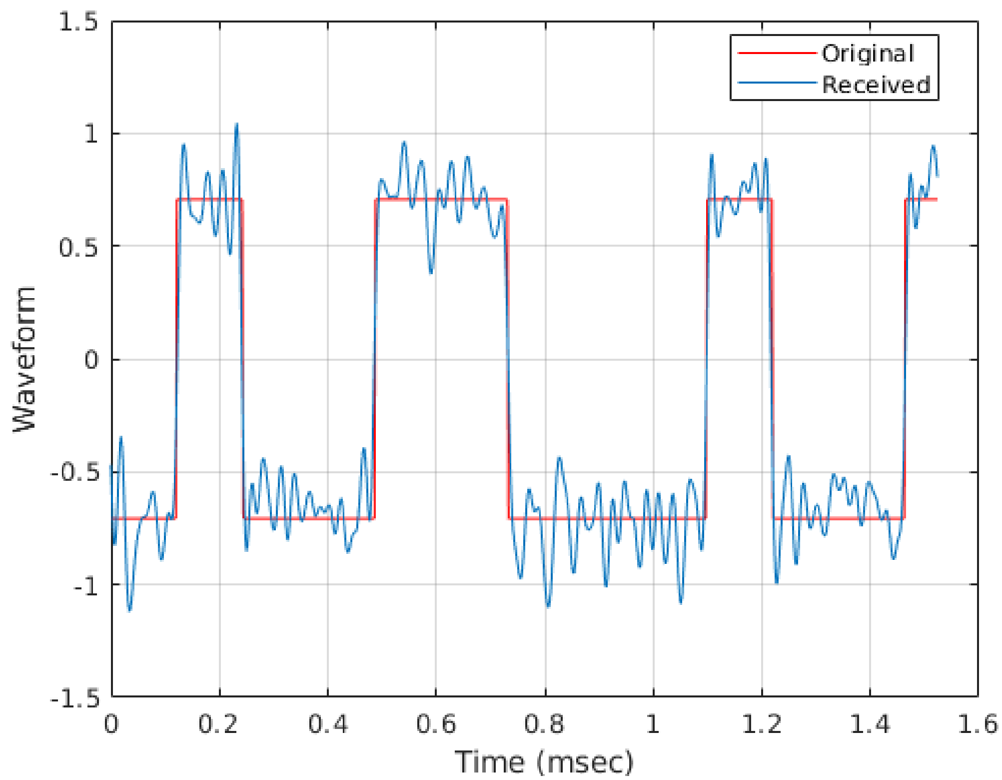


Display the Original Song and the Receiver Output Segments

Can you feel the noise?

Comparing the imaginary components of transmitted and received baseband signal segments

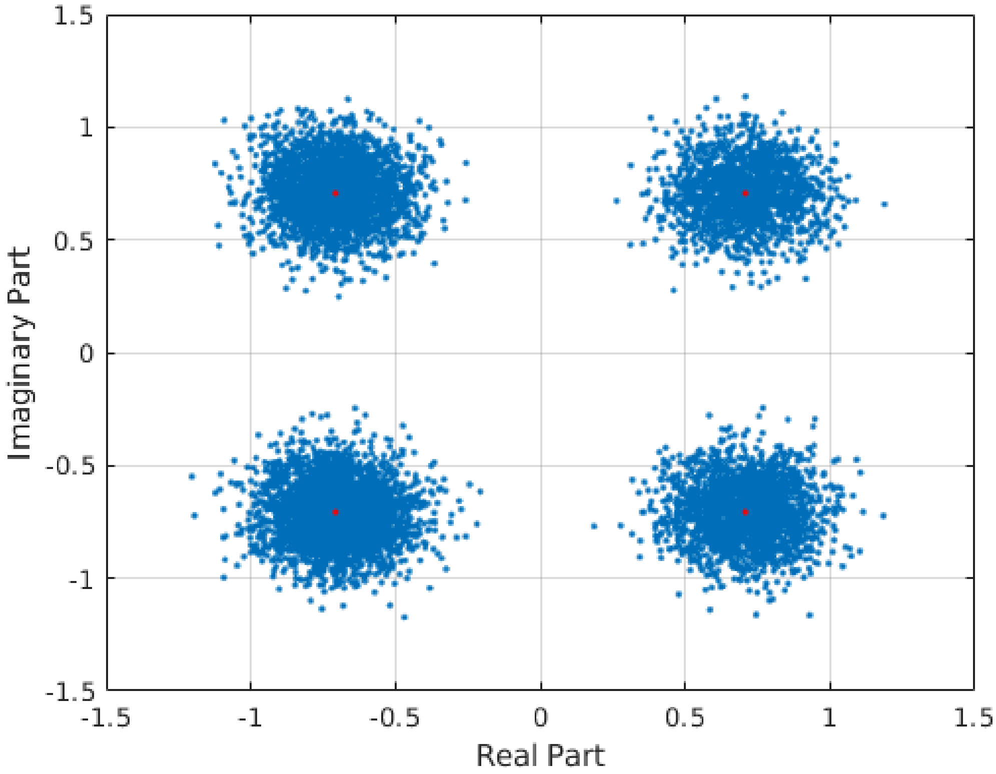
```
figure(4)
plot(t(1:SL)*1000,imag(xb(1:SL)),'r')
hold on
plot(t(1:SL)*1000,imag(z(1:SL)))
grid
xlabel('Time (msec)');
ylabel('Waveform');
legend('Original','Received','Location','Best');
```



Constellation Estimates

We sample the baseband received signal to get noisy estimates of transmitted constellation point. This is not the best way though. Any other suggestions for improvement?

```
ce=z(ceil(Ns/2):Ns:length(z));

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



Bit Estimates

We implement QPSK Demapper to extract bits from constellation estimates

Check which quadrant ce lies in

```
ser=real(ce)>0;
sei=imag(ce)>0;
se(1:2:(2*length(ser)))=ser;
se(2:2:(2*length(ser)))=sei;
```

Calculate Bit Error Rate

```
BER=sum(se~=s)/length(s)
```

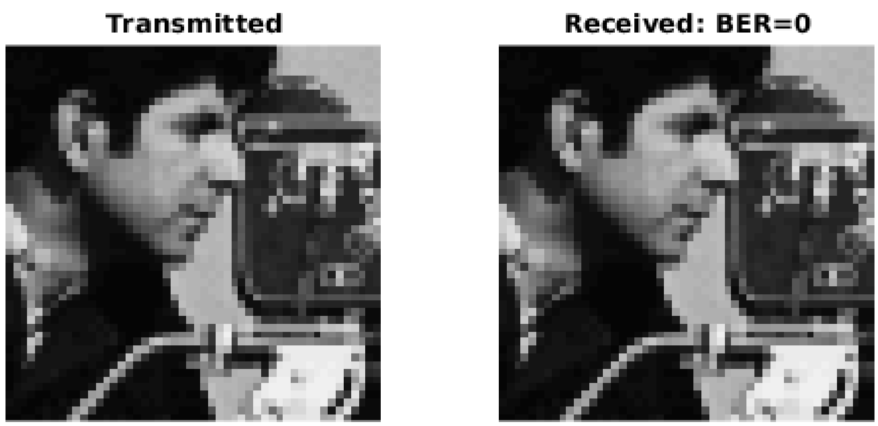
```
BER =

     0
```

Reconstruct Image

From the bits we estimated, we reconstruct 8-bit gray level image

```
Imvbe=reshape(se,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(BER)])
```



LOW SNR CASE

Define new SNR level in (dB)

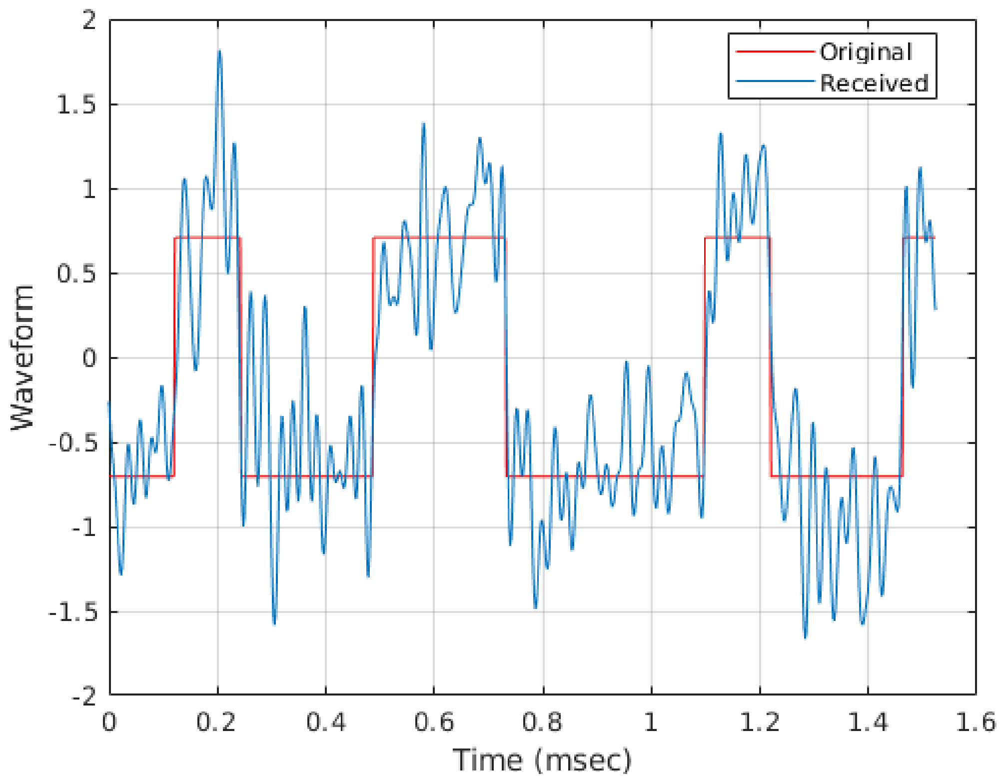
```
SNR=1;

% 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 QPSK Receiver Processing
```

```
% First extract real component baseband signal
% $u_r(t)=2x(t)cos(2\pi f_c t)$
ur=2*y.*cost;
% Then low pass filter this signal
zr = lowpass(ur,30e3,Fsampling);
% Then extract the imaginary component baseband signal
ui=-2*y.*sint;
% Then low pass filter this signal
zi = lowpass(ui,30e3,Fsampling);
% Basband signal
z=zr+i*zi;
```

Display the Original Song and the Receiver Output Segments

Comparing the imaginary components of transmitted and received baseband signal segments

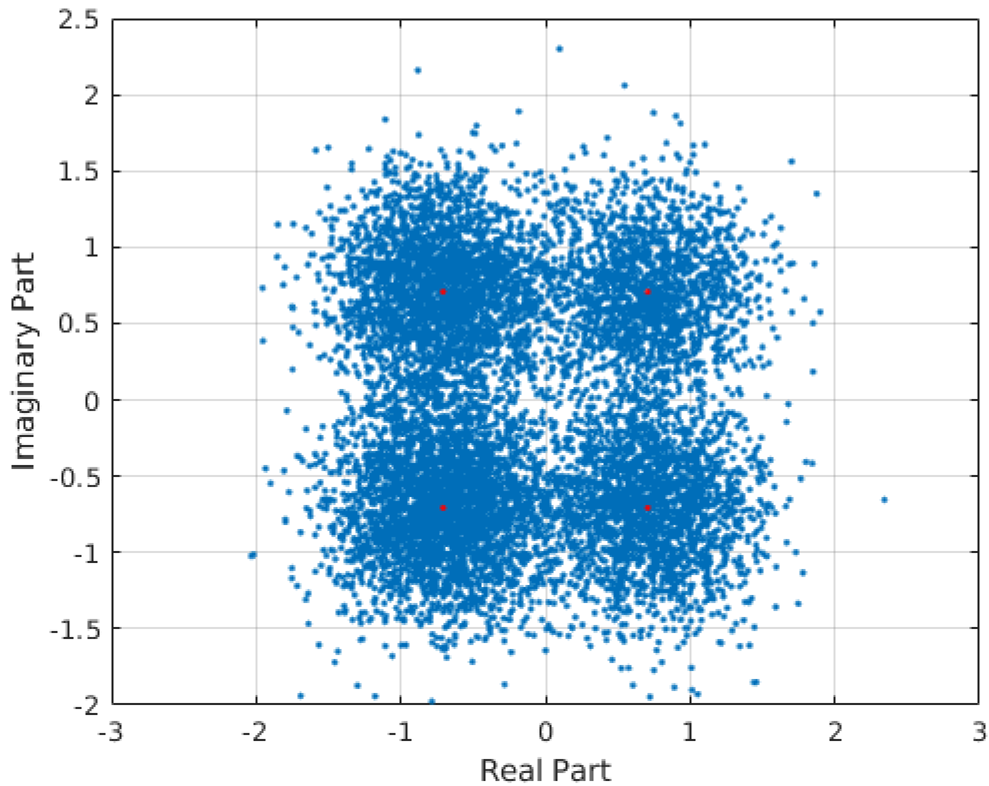
```
figure(7)
plot(t(1:SL)*1000,imag(xb(1:SL)),'r')
hold on
plot(t(1:SL)*1000,imag(z(1:SL)))
grid
xlabel('Time (msec)');
ylabel('Waveform');
legend('Original','Received','Location','Best');
```



Constellation Estimates

```
ce=z(ceil(Ns/2):Ns:length(z));
```

```
figure(8)
% Plot constellation estimates
plot(real(ce),imag(ce),'.');
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 lies in

```
ser=real(ce)>0;
sei=imag(ce)>0;
se(1:2:(2*length(ser)))=ser;
se(2:2:(2*length(ser)))=sei;
```

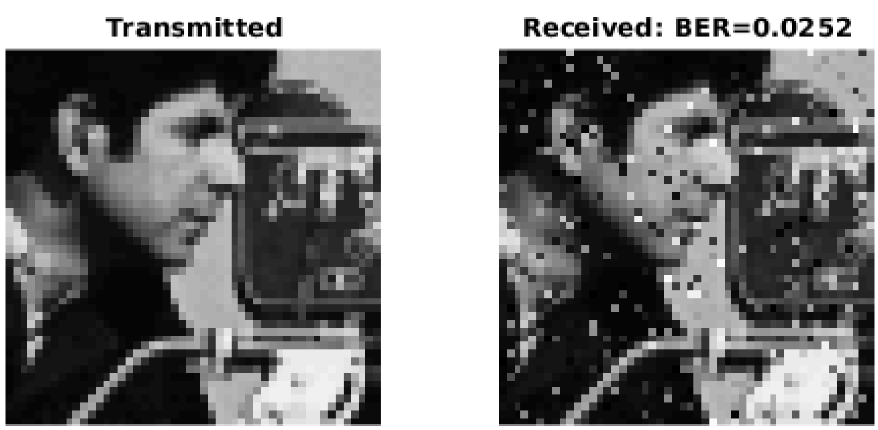
Calculate Bit Error Rate

```
BER=sum(se~=s)/length(s)
```

```
BER =

    0.0252
```

Reconstruct Image From the bits we estimated, we reconstruct 8-bit gray level image

```
Imvbe=reshape(se,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(BER)])
```



Published with MATLAB® R2018b