Introduction

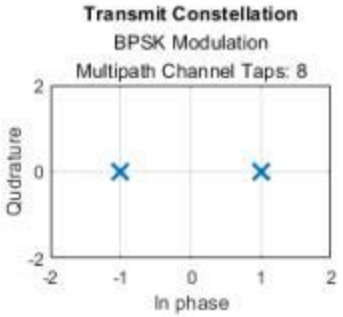
In this final project, I've done 3 part. The first part is a channel estimation using a graph. I will compare the difference resulted by different methods and different modulation method. The second and the third part is basic learning about the TDL and CDL channel.

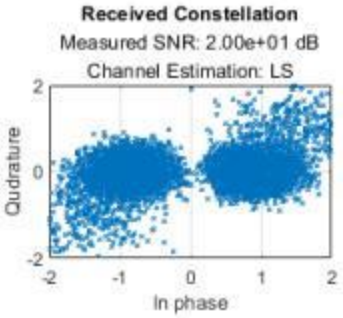
1.1

In this part, we transmit a photo through the channel. The code is based on "Jordan Street's" OFDM simulation presentation. I make a few changes and combine them to compare with each other.

As we can see from the graph that modulation method plays an important role in the recovered image. Since it is using the knn method, the Ber performs better when the modulation requires less bits.

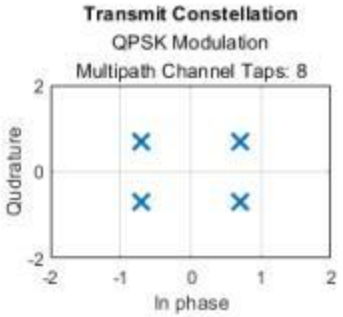
When choosing the same method, using LS as channel estimation will largely reduce the bit error rate and thus realize the recovery.

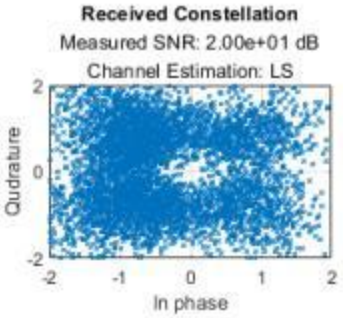

Transmit Constellation
BPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase

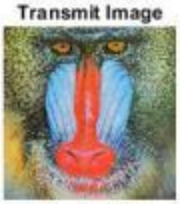

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image

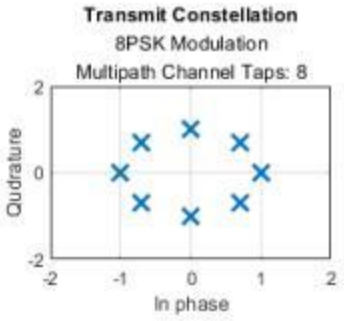

Recovered Image
BER: 0.00041

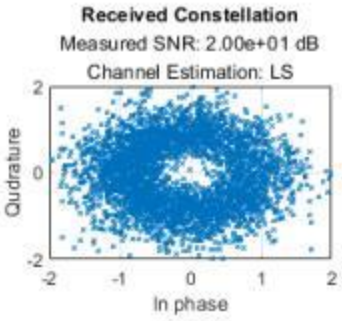

Transmit Constellation
QPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image

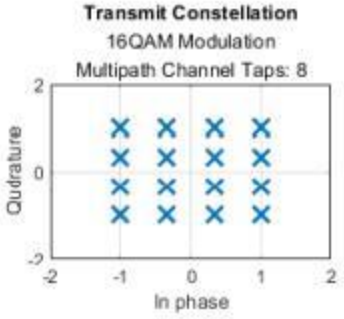

Recovered Image
BER: 0.031

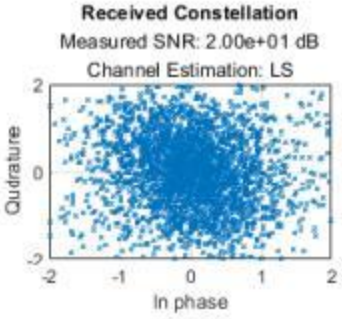

Transmit Constellation
8PSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image

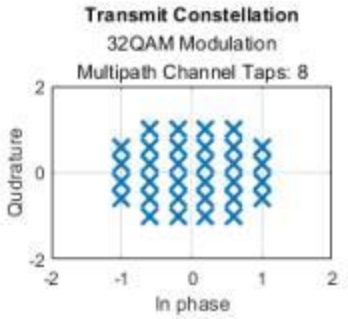

Recovered Image
BER: 0.15

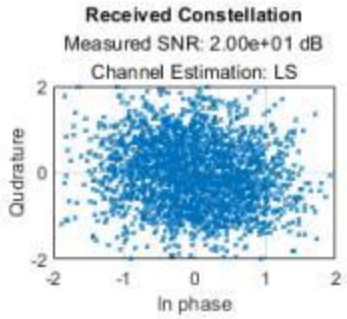

Transmit Constellation
16QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase

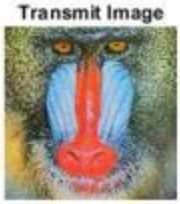

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase

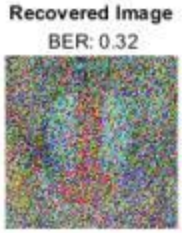

Transmit Image

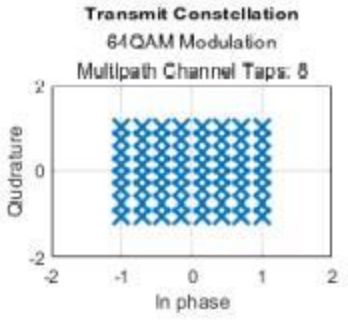

Recovered Image
BER: 0.26

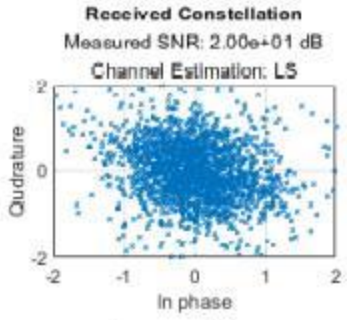

Transmit Constellation
32QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase

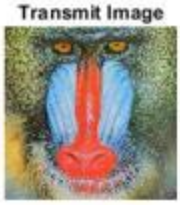

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image

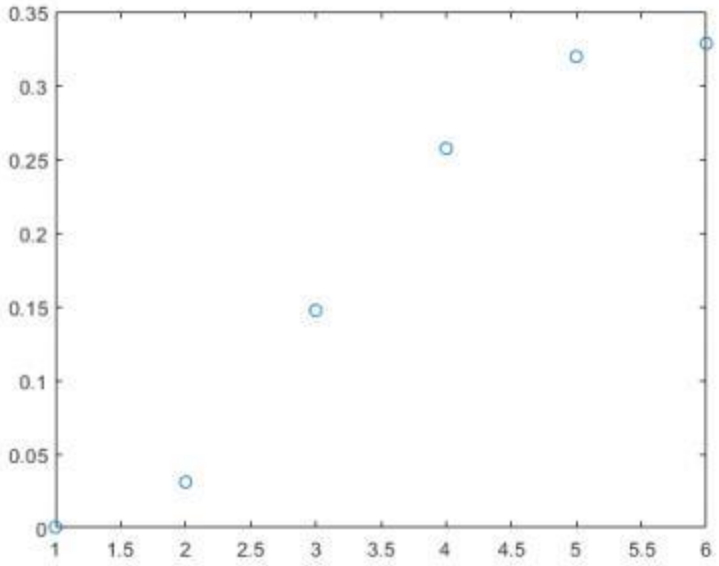

Recovered Image
BER: 0.32


Transmit Constellation
64QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image


Recovered Image
BER: 0.33



1.2

In this part, the code shows how to display the waveform spectrum received through a tapped delay line (TDL) multi-input/multi-output (MIMO) channel model from TR 38.901 Section 7.7.2 using an nrTDLChannel System object. The code is from the official site of the 5G tool box

1.3

In this part, the code shows how to plot channel output and path gain snapshots for various sample density values while using an nrCDLChannel System object. The final graph shows the path gain relationship with the sample density

Reference

https://www.mathworks.com/help/5g/ref/nrcdlchannel-system-object.html

https://www.mathworks.com/matlabcentral/fileexchange/67156-ofdm-channel-estimation-in-matlab

https://www.mathworks.com/help/5g/ref/nrtdlchannel-system-object.html

```
clear all
close all
clc

berarr = zeros(6,1);
for i = 1:6


nfft = 64;
n_fft = 64;


n_cpe = 16;

snr = 20; % in dB



n_taps = 8;

ch_est_method = 'LS'
%ch_est_method = 'none';

mod_methods = {'BPSK', 'QPSK', '8PSK', '16QAM', '32QAM', '64QAM'};
switch(i)
    case 1
       mod_method = 'BPSK';
    case 2
       mod_method = 'QPSK';
    case 3
       mod_method = '8PSK';
    case 4
       mod_method = '16QAM';
    case 5
       mod_method = '32QAM';
    case 6
       mod_method =  '64QAM';
end

mod_order = find(ismember(mod_methods,mod_method));

im = imread('baboon.png');
im_bin = dec2bin(im(:))';
im_bin = im_bin(:);
sym_rem = mod(mod_order-mod(length(im_bin),mod_order),mod_order);
padding = repmat('0',sym_rem,1);
im_bin_padded = [im_bin;padding];
cons_data = reshape(im_bin_padded,mod_order,length(im_bin_padded)/
mod_order)';
cons_sym_id = bin2dec(cons_data);

if mod_order == 1
```

```
    mod_ind = 2^(mod_order-1);
    n = 0:pi/mod_ind:2*pi-pi/mod_ind;
    in_phase = cos(n);
    quadrature = sin(n);
    symbol_book = (in_phase + quadrature*1i);
end

if mod_order == 2 || mod_order == 3
    mod_ind = 2^(mod_order-1);
    n = 0:pi/mod_ind:2*pi-pi/mod_ind;
    in_phase = cos(n+pi/4);
    quadrature = sin(n+pi/4);
    symbol_book = (in_phase + quadrature*1i);
end

if mod_order == 4 || mod_order == 6
    mod_ind = sqrt(2^mod_order);
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
end

if mod_order == 5
    mod_ind = 6;
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
    symbol_book = symbol_book([2:5 7:30 32:35]);
end

X = symbol_book(cons_sym_id+1);

fft_rem = mod(n_fft-mod(length(X),n_fft),n_fft);
X_padded = [X;zeros(fft_rem,1)];
X_blocks = reshape(X_padded,nfft,length(X_padded)/nfft);
x = ifft(X_blocks);

x_cpe = [x(end-n_cpe+1:end,:);x];
x_s   = x_cpe(:);

data_pwr = mean(abs(x_s.^2));

noise_pwr = data_pwr/10^(snr/10);
noise =
 normrnd(0,sqrt(noise_pwr/2),size(x_s))+normrnd(0,sqrt(noise_pwr/2),size(x_s))*1i;
x_s_noise = x_s + noise;

snr_meas = 10*log10(mean(abs(x_s.^2))/mean(abs(noise.^2)));

g = exp(-(0:n_taps-1));
g = g/norm(g);
x_s_noise_fading = conv(x_s_noise,g,'same');
```

```
x_p = reshape(x_s_noise_fading,nfft+n_cpe,length(x_s_noise_fading)/
(nfft+n_cpe));
x_p_cpr = x_p(n_cpe+1:end,:);

X_hat_blocks = fft(x_p_cpr);

if n_taps > 1
    switch(ch_est_method)
        case 'none'
        case 'LS'
            G = X_hat_blocks(:,1)./X_blocks(:,1);
            X_hat_blocks = X_hat_blocks./
repmat(G,1,size(X_hat_blocks,2));
        case 'normalized Ls'
            G = X_hat_blocks(:,1)./X_blocks(:,1);
            G = G/norm(G);
            X_hat_blocks = X_hat_blocks./
repmat(G,1,size(X_hat_blocks,2));
    end
end


X_hat = X_hat_blocks(:);
X_hat = X_hat(1:end-fft_rem);


A=[real(symbol_book) imag(symbol_book)];
if (size(A,2)>2)
    A=[real(symbol_book)' imag(symbol_book)'];
end
rec_syms = knnsearch(A,[real(X_hat) imag(X_hat)])-1;

rec_syms_cons = dec2bin(rec_syms);
rec_im_bin = reshape(rec_syms_cons',numel(rec_syms_cons),1);
rec_im_bin = rec_im_bin(1:end-sym_rem);
ber = sum(abs(rec_im_bin-im_bin))/length(im_bin);

rec_im = reshape(rec_im_bin,8,numel(rec_im_bin)/8);
rec_im = uint8(bin2dec(rec_im'));
rec_im = reshape(rec_im,size(im));

figure(i);
subplot(2,2,1);
plot(X,'x','linewidth',2,'markersize',10);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')

if n_taps > 1
    title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation
\nMultipath Channel Taps: %d',mod_method,n_taps));
else
```

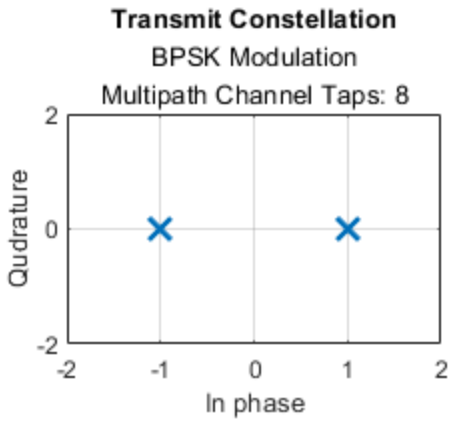
```
    title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation
\nMultipath Channel Taps: %d',mod_method));
end
grid on

subplot(2,2,2);
plot(X_hat(1:500:end),'x','markersize',3);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')

if n_taps > 1
    title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d
 dB\nChannel Estimation: %s',snr_meas,ch_est_method));
else
    title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d
 dB',snr_meas));
end
grid on

subplot(2,2,3);
imshow(im);
title('\bfTransmit Image');

subplot(2,2,4);
imshow(rec_im);
title(sprintf('\\bfRecovered Image\n \\rmBER: %.2g',ber));

berarr(i) =ber;
end
figure(7)
plot(berarr,'o');

for i = 1:3


nfft = 64;
n_fft = 64;


n_cpe = 16;

snr = 20; % in dB



n_taps = 8;

switch i
    case 1
        ch_est_method = 'LS';
    case 2
        ch_est_method = 'none';
```

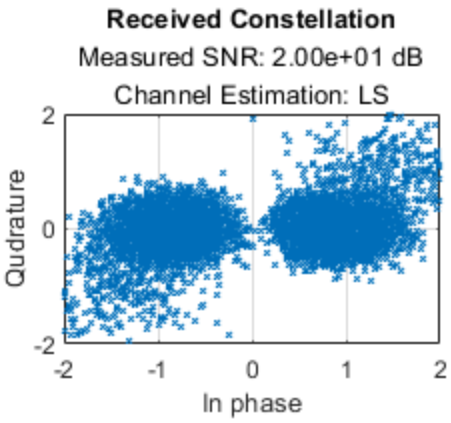
```
    case 3
        ch_est_method = 'normalized Ls'
end

mod_methods = {'BPSK', 'QPSK', '8PSK', '16QAM', '32QAM', '64QAM'};
mod_method = 'BPSK';

mod_order = find(ismember(mod_methods,mod_method));

im = imread('baboon.png');
im_bin = dec2bin(im(:))';
im_bin = im_bin(:);
sym_rem = mod(mod_order-mod(length(im_bin),mod_order),mod_order);
padding = repmat('0',sym_rem,1);
im_bin_padded = [im_bin;padding];
cons_data = reshape(im_bin_padded,mod_order,length(im_bin_padded)/
mod_order)';
cons_sym_id = bin2dec(cons_data);

if mod_order == 1
    mod_ind = 2^(mod_order-1);
    n = 0:pi/mod_ind:2*pi-pi/mod_ind;
    in_phase = cos(n);
    quadrature = sin(n);
    symbol_book = (in_phase + quadrature*1i);
end

if mod_order == 2 || mod_order == 3
    mod_ind = 2^(mod_order-1);
    n = 0:pi/mod_ind:2*pi-pi/mod_ind;
    in_phase = cos(n+pi/4);
    quadrature = sin(n+pi/4);
    symbol_book = (in_phase + quadrature*1i);
end

if mod_order == 4 || mod_order == 6
    mod_ind = sqrt(2^mod_order);
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
end

if mod_order == 5
    mod_ind = 6;
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
    symbol_book = symbol_book([2:5 7:30 32:35]);
end

X = symbol_book(cons_sym_id+1);

fft_rem = mod(n_fft-mod(length(X),n_fft),n_fft);
X_padded = [X;zeros(fft_rem,1)];
```

```
X_blocks = reshape(X_padded,nfft,length(X_padded)/nfft);
x = ifft(X_blocks);

x_cpe = [x(end-n_cpe+1:end,:);x];
x_s   = x_cpe(:);

data_pwr = mean(abs(x_s.^2));

noise_pwr = data_pwr/10^(snr/10);
noise =
 normrnd(0,sqrt(noise_pwr/2),size(x_s))+normrnd(0,sqrt(noise_pwr/2),size(x_s))*1i;
x_s_noise = x_s + noise;

snr_meas = 10*log10(mean(abs(x_s.^2))/mean(abs(noise.^2)));

g = exp(-(0:n_taps-1));
g = g/norm(g);
x_s_noise_fading = conv(x_s_noise,g,'same');


x_p = reshape(x_s_noise_fading,nfft+n_cpe,length(x_s_noise_fading)/
(nfft+n_cpe));
x_p_cpr = x_p(n_cpe+1:end,:);

X_hat_blocks = fft(x_p_cpr);

if n_taps > 1
    switch(ch_est_method)
        case 'none'
        case 'LS'
            G = X_hat_blocks(:,1)./X_blocks(:,1);
            X_hat_blocks = X_hat_blocks./
repmat(G,1,size(X_hat_blocks,2));
        case 'normalized Ls'
            G = X_hat_blocks(:,1)./X_blocks(:,1);
            G = G/norm(G);
            X_hat_blocks = X_hat_blocks./
repmat(G,1,size(X_hat_blocks,2));
    end
end


X_hat = X_hat_blocks(:);
X_hat = X_hat(1:end-fft_rem);


A=[real(symbol_book) imag(symbol_book)];
if (size(A,2)>2)
    A=[real(symbol_book)' imag(symbol_book)'];
end
rec_syms = knnsearch(A,[real(X_hat) imag(X_hat)])-1;

rec_syms_cons = dec2bin(rec_syms);
rec_im_bin = reshape(rec_syms_cons',numel(rec_syms_cons),1);
```

```
rec_im_bin = rec_im_bin(1:end-sym_rem);
ber = sum(abs(rec_im_bin-im_bin))/length(im_bin);

rec_im = reshape(rec_im_bin,8,numel(rec_im_bin)/8);
rec_im = uint8(bin2dec(rec_im'));
rec_im = reshape(rec_im,size(im));

figure(i+7);
subplot(2,2,1);
plot(X,'x','linewidth',2,'markersize',10);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')

if n_taps > 1
    title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation
\nMultipath Channel Taps: %d',mod_method,n_taps));
else
    title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation
\nMultipath Channel Taps: %d',mod_method));
end
grid on

subplot(2,2,2);
plot(X_hat(1:500:end),'x','markersize',3);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')

if n_taps > 1
    title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d
 dB\nChannel Estimation: %s',snr_meas,ch_est_method));
else
    title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d
 dB',snr_meas));
end
grid on

subplot(2,2,3);
imshow(im);
title('\bfTransmit Image');

subplot(2,2,4);
imshow(rec_im);
title(sprintf('\\bfRecovered Image\n \\rmBER: %.2g',ber));

end


ch_est_method =

    'LS'
```

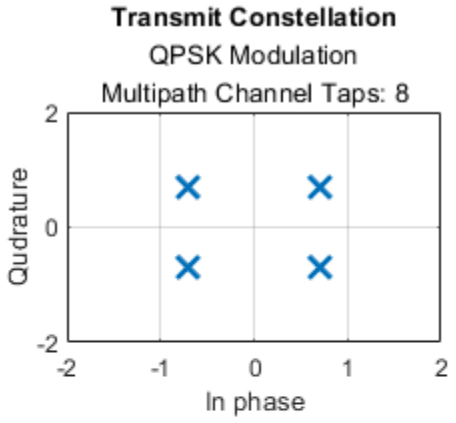
```
ch_est_method =

    'LS'

ch_est_method =

    'LS'

ch_est_method =

    'LS'

ch_est_method =

    'LS'

ch_est_method =

    'LS'

ch_est_method =

    'normalized Ls'
```

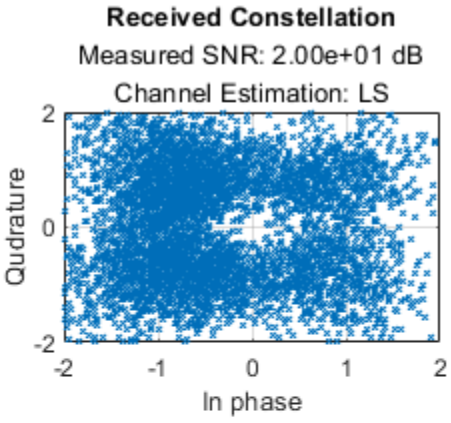

Transmit Constellation
BPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-1
1


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
2
0
-2
-1
1


Transmit Image

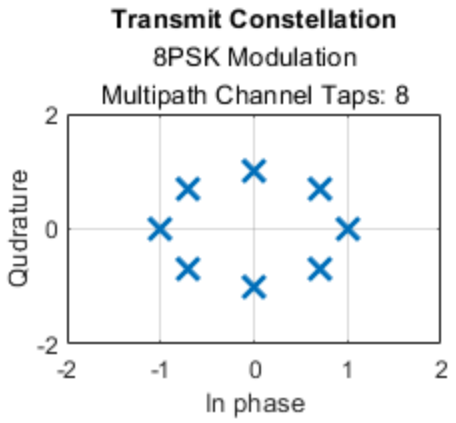

Recovered Image
BER: 0.00041

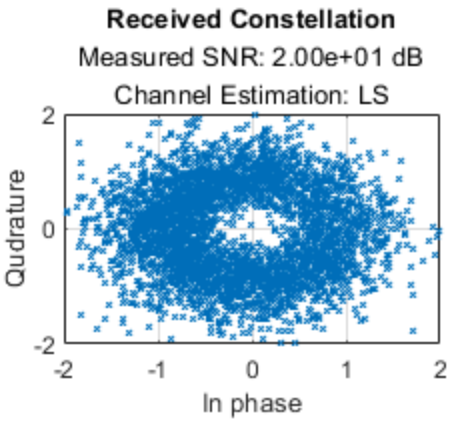

Transmit Constellation
QPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-1
1


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
2
0
-2
-1
1


Transmit Image

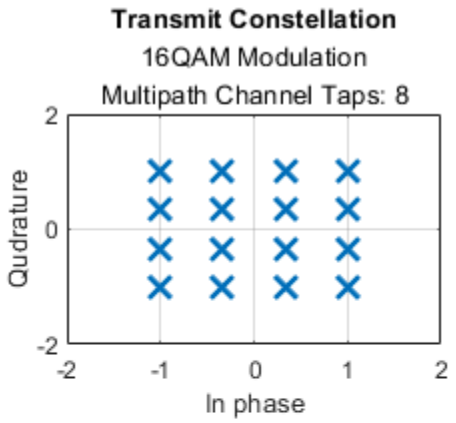

Recovered Image
BER: 0.031

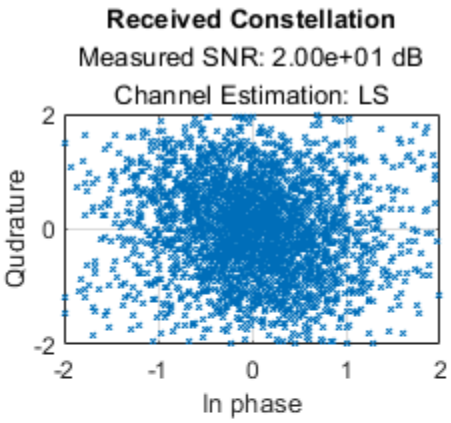

Transmit Constellation
8PSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-2
-1
0
1
2


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
2
0
-2
-2
-1
0
1
2


Transmit Image

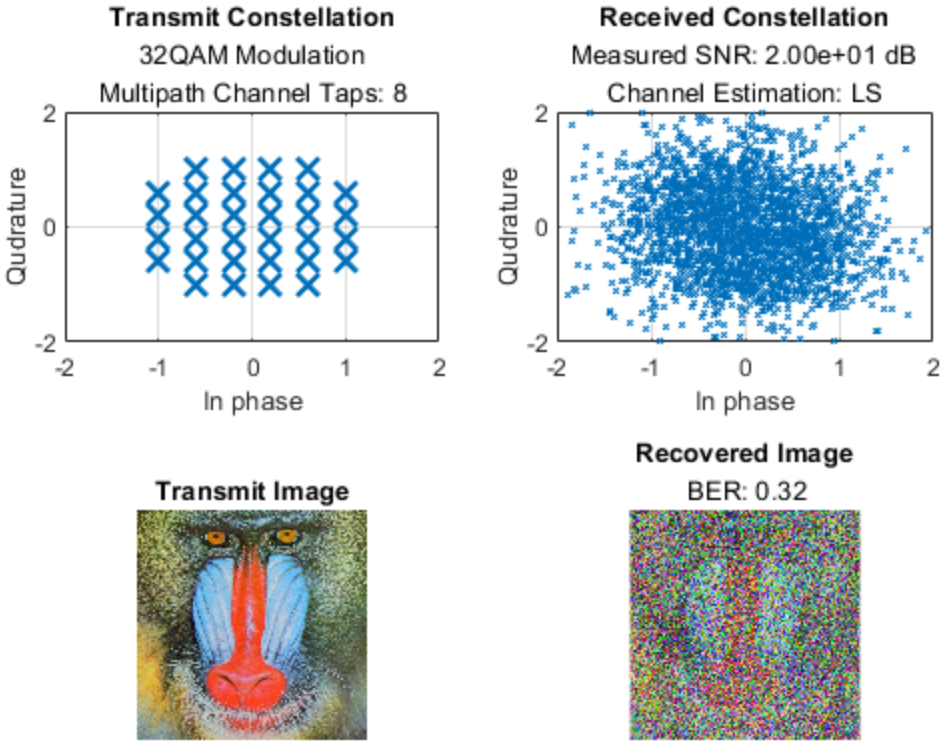

Recovered Image
BER: 0.15

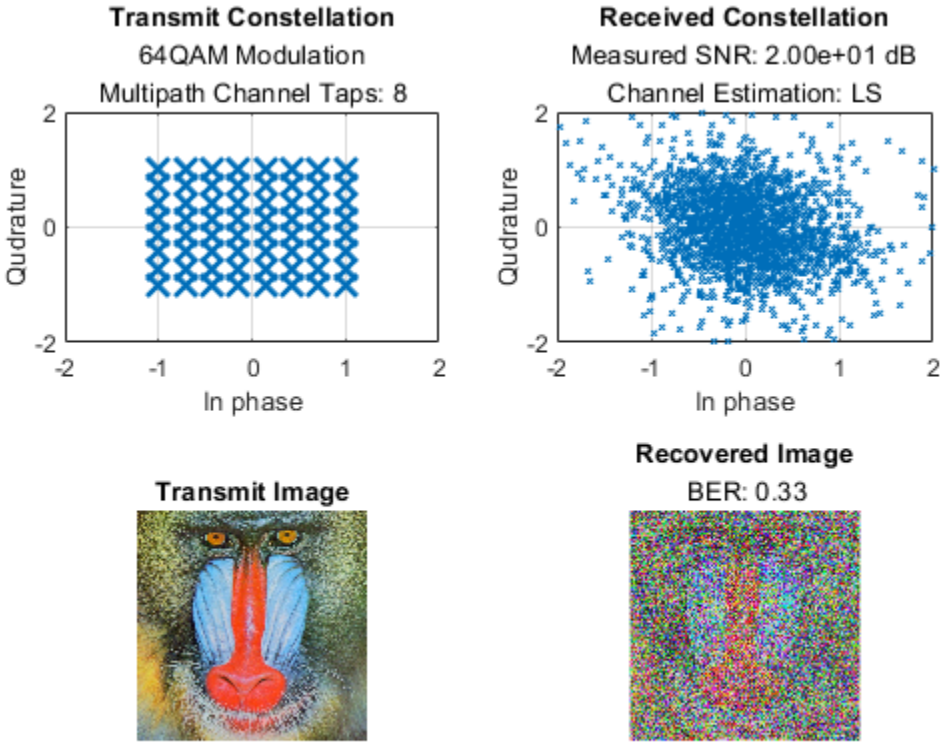

Transmit Constellation
16QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-2
-1
0
1
2

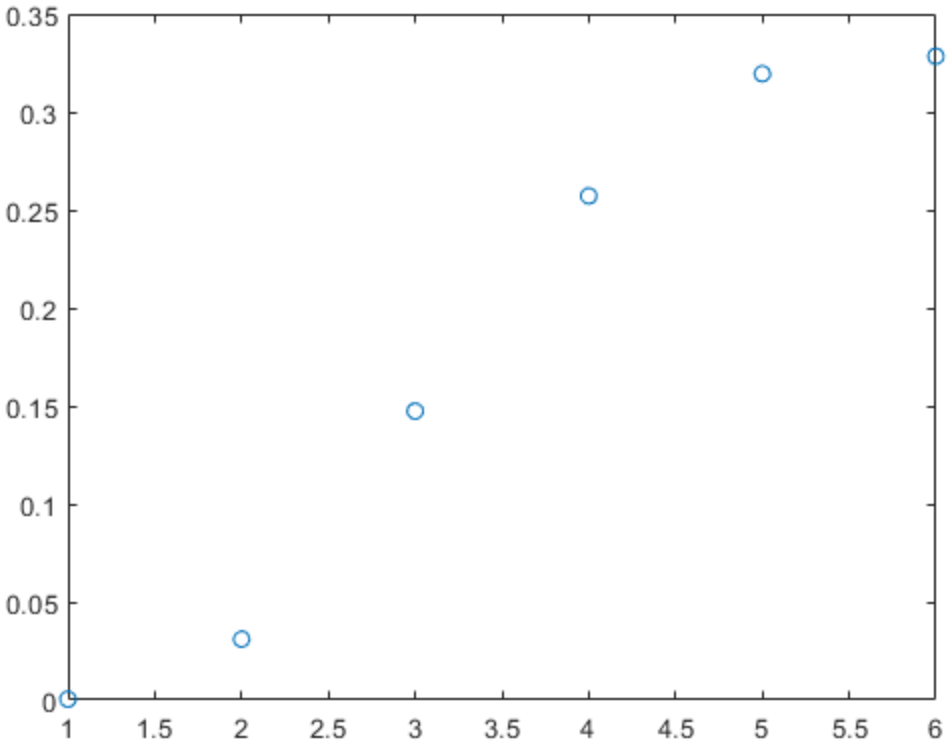

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
2
0
-2
-2
-1
0
1
2

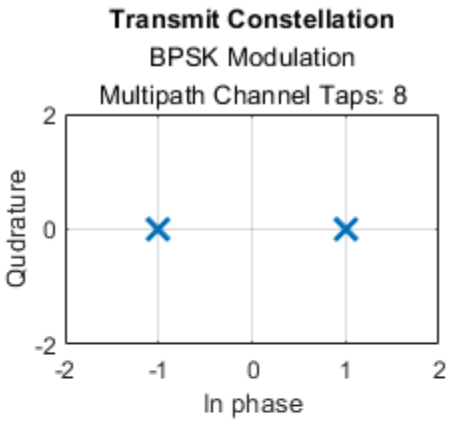

Transmit Image

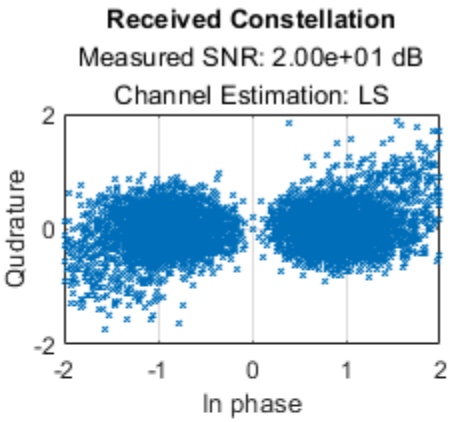

Recovered Image
BER: 0.26


Transmit Constellation
32QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase
Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
Transmit Image
Recovered Image
BER: 0.32


Transmit Constellation
64QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase
Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase
Transmit Image
Recovered Image
BER: 0.33



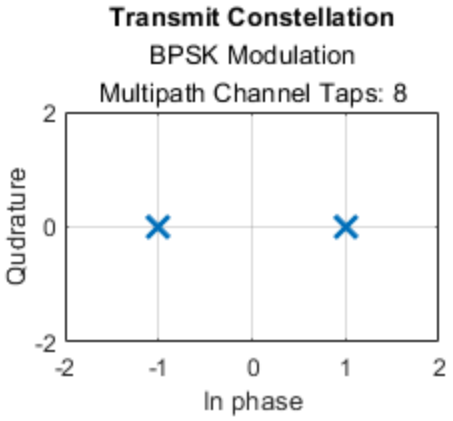


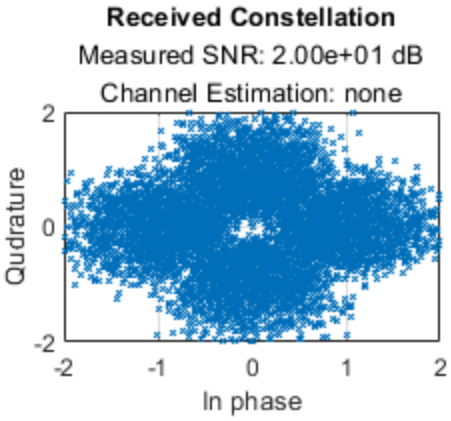


Transmit Image

Recovered Image

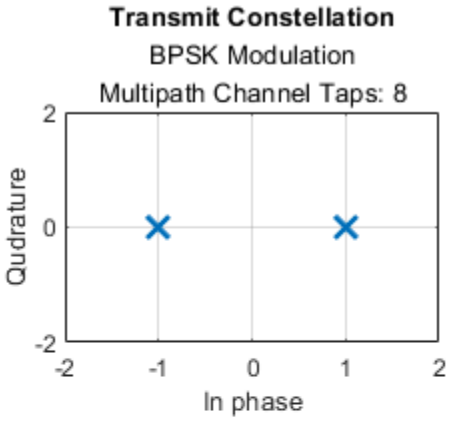
BER: 0.00043

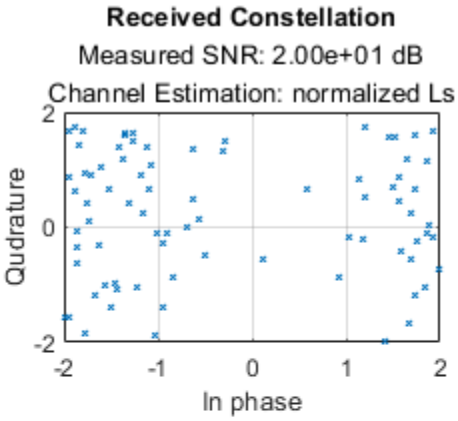

Transmit Constellation
BPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-2
-1
0
1
2


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: none
Qudrature
In phase
2
0
-2
-2
-1
0
1
2


Transmit Image


Recovered Image
BER: 0.5


Transmit Constellation
BPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase
2
0
-2
-2
-1
0
1
2

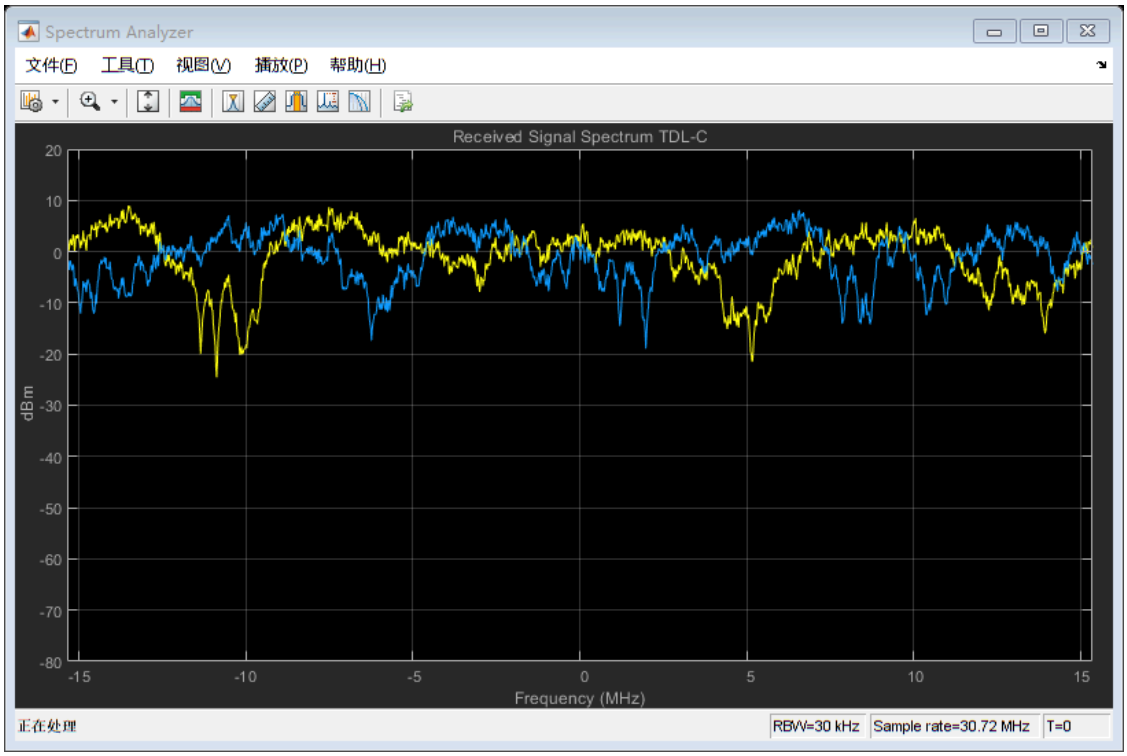

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: normalized Ls
Qudrature
In phase
2
0
-2
-2
-1
0
1
2


Transmit Image


Recovered Image
BER: 0.00043

Published with MATLAB® R2020b

```
v = 30.0;                       % UE velocity in km/h
fc = 4e9;                       % carrier frequency in Hz
c = physconst('lightspeed'); % speed of light in m/s
fd = (v*1000/3600)/c*fc;     % UE max Doppler frequency in Hz

tdl = nrTDLChannel;
tdl.DelayProfile = 'TDL-C';
tdl.DelaySpread = 300e-9;
tdl.MaximumDopplerShift = fd;

SR = 30.72e6;
T = SR * 1e-3;
tdl.SampleRate = SR;
tdlinfo = info(tdl);
Nt = tdlinfo.NumTransmitAntennas;

txWaveform = complex(randn(T,Nt),randn(T,Nt));

rxWaveform = tdl(txWaveform);

analyzer = dsp.SpectrumAnalyzer('SampleRate',tdl.SampleRate,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99 );
analyzer.Title = ['Received Signal Spectrum ' tdl.DelayProfile];
analyzer(rxWaveform);
```

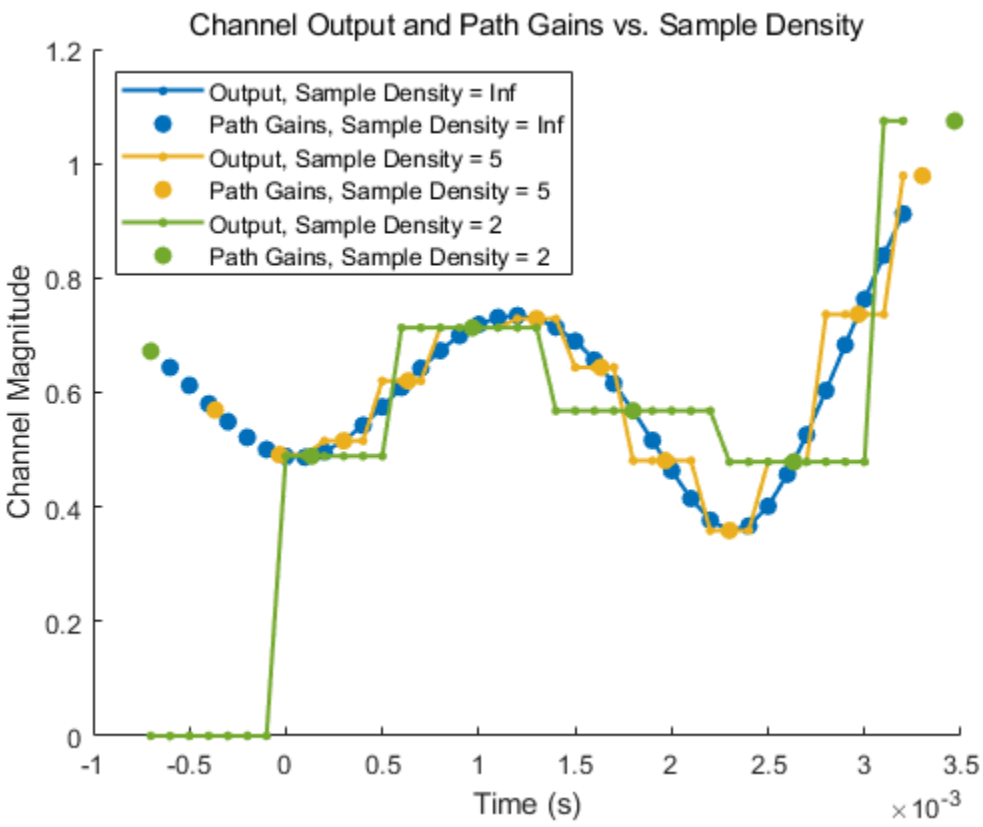


Published with MATLAB® R2020b

```
v = 15.0;                    % UE velocity in km/h
fc = 4e9;                    % carrier frequency in Hz
c = physconst('lightspeed'); % speed of light in m/s
fd = (v*1000/3600)/c*fc;     % UE max Doppler frequency in Hz


cdl = nrCDLChannel;
cdl.DelayProfile = 'CDL-B';
cdl.MaximumDopplerShift = 300.0;
cdl.SampleRate = 10e3;
cdl.Seed = 19;

cdl.TransmitAntennaArray.Size = [1 1 1 1 1];
cdl.ReceiveAntennaArray.Size = [1 1 1 1 1];

T = 40;
in = ones(T,1);

s = [Inf 5 2]; % sample densities

legends = {};
figure; hold on;
SR = cdl.SampleRate;
for i = 1:length(s)

    % call channel with chosen sample density
    release(cdl); cdl.SampleDensity = s(i);
    [out,pathgains,sampletimes] = cdl(in);
    chInfo = info(cdl); tau = chInfo.ChannelFilterDelay;

    % plot channel output against time
    t = cdl.InitialTime + ((0:(T-1)) - tau).' / SR;
    h = plot(t,abs(out),'o-');
    h.MarkerSize = 2;
    h.LineWidth = 1.5;
    desc = ['Sample Density = ' num2str(s(i))];
    legends = [legends ['Output, ' desc]];
    disp([desc ', Ncs = ' num2str(length(sampletimes))]);

    % plot path gains against sample times
    h2 = plot(sampletimes-tau/SR,abs(sum(pathgains,2)),'o');
    h2.Color = h.Color;
    h2.MarkerFaceColor = h.Color;
    legends = [legends ['Path Gains, ' desc]];
end

xlabel('Time (s)');
title('Channel Output and Path Gains vs. Sample Density');
ylabel('Channel Magnitude');
legend(legends,'Location','NorthWest');
```

```
Sample Density = Inf, Ncs = 40
```

```
Sample Density = 5, Ncs = 13
Sample Density = 2, Ncs = 6
```



Published with MATLAB® R2020b