Table of Contents


```
clear all
clc
close all
```

Read Audio

```
% Reading the audioo file, finding sampling frequency, and the type of
 channel
filename= "test.wav";
[Audio, fs]= audioread(filename);  %Read the audio file
info=audioinfo(filename);
NumChannels = info.NumChannels;

%Check if the audio file is mono or stereo
if(NumChannels == 1)
    Audio_Signal = Audio(:,1);
else
    Audio_Signal = [Audio(:,1)', Audio(:,2)'];
end
```

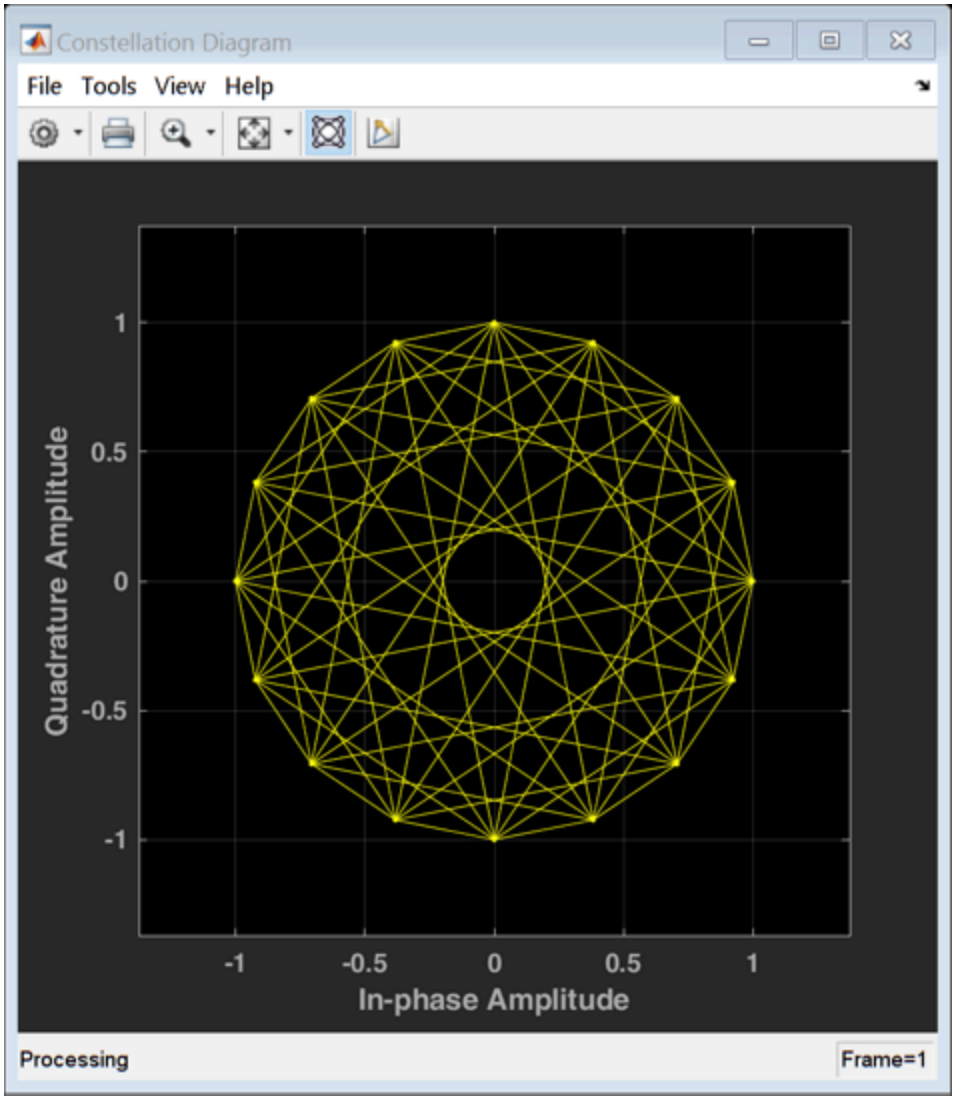
Quantization and Encoding

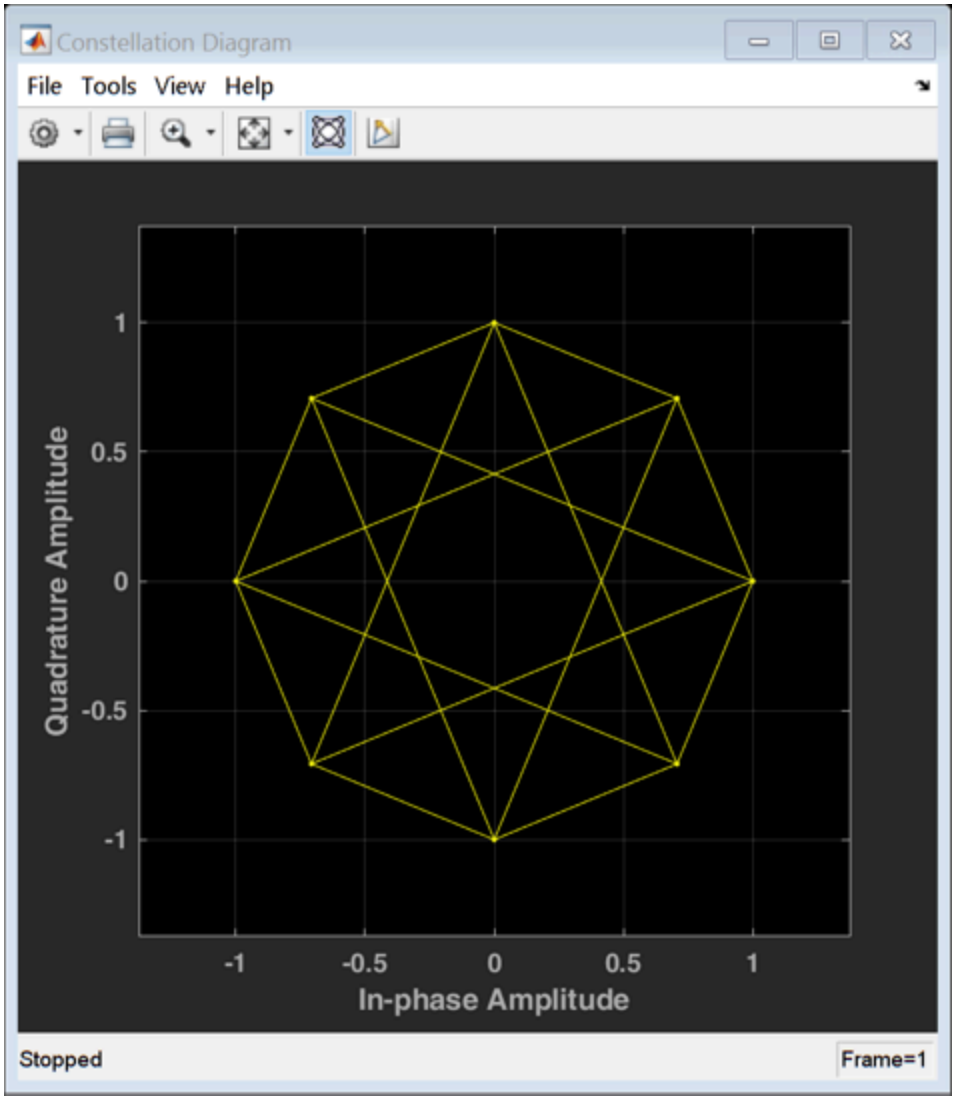
```
% define number of bits for quantization, quantize the signal and
 encode it
n_bits = 8;
n_levels = 2^n_bits;

[Quantized_Samples, Levels] = Quantizer(Audio_Signal, n_bits);
Bit_Stream = Encoder(Quantized_Samples, n_levels);
```

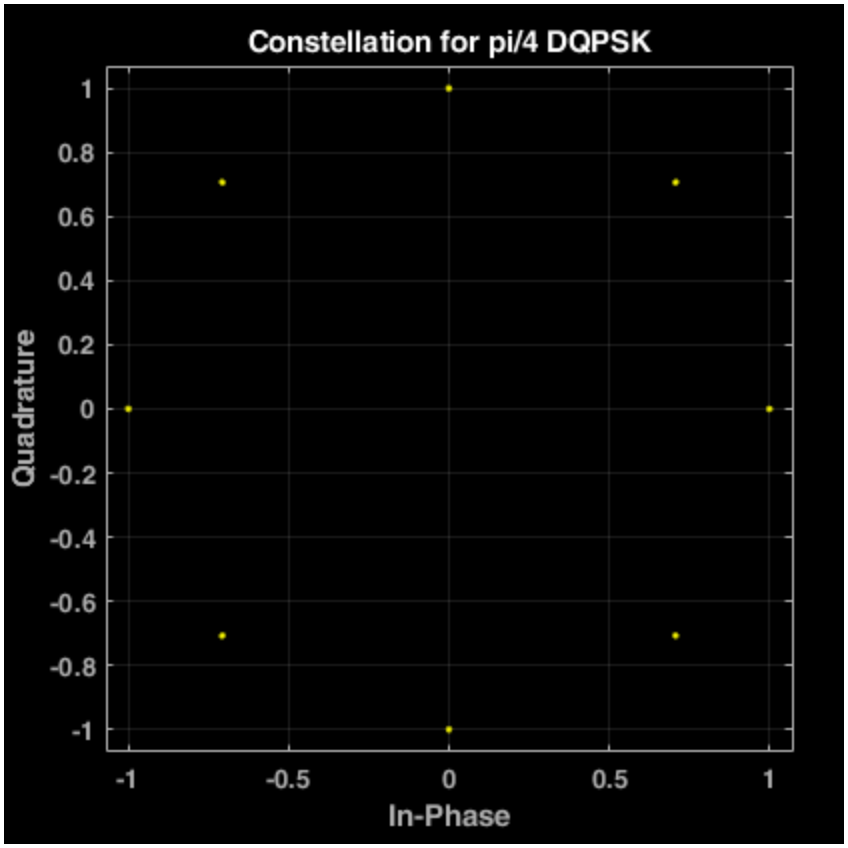
Modulator

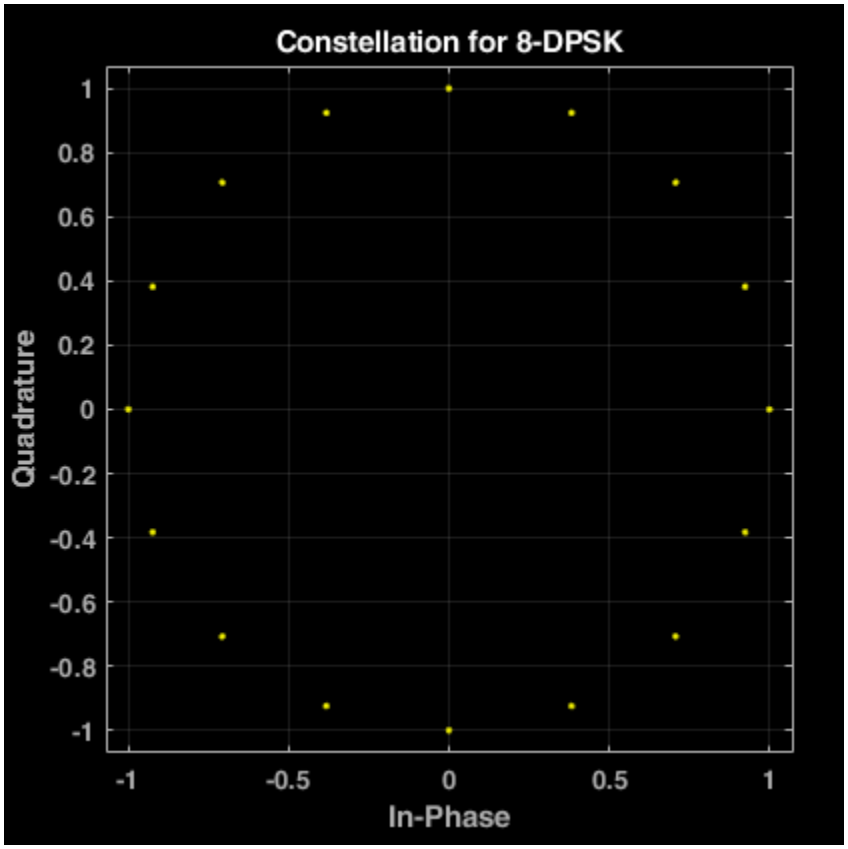
```
%Using pi/4 DQPSK Modulation
```

```
modulator = comm.DQPSKModulator('BitInput',true);
mod_bits = modulator(Bit_Stream');
constDiagram =
 comm.ConstellationDiagram('ShowTrajectory',true,'ShowReferenceConstellation',fals
constDiagram(mod_bits);
release(constDiagram);
scatterplot(mod_bits);
title('Constellation for pi/4 DQPSK'); grid on;

%Using 8-DPSK Modulation
mod = comm.DPSKModulator(8, pi/8, 'BitInput',true);
mod_bits2 = mod(Bit_Stream');
constDiagram1 =
 comm.ConstellationDiagram('ShowTrajectory',true,'ShowReferenceConstellation',fals
constDiagram1(mod_bits2);
scatterplot(mod_bits2);
title('Constellation for 8-DPSK'); grid on;
```




Constellation Diagram
File Tools View Help
Quadrature Amplitude
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
In-phase Amplitude
Stopped
Frame=1

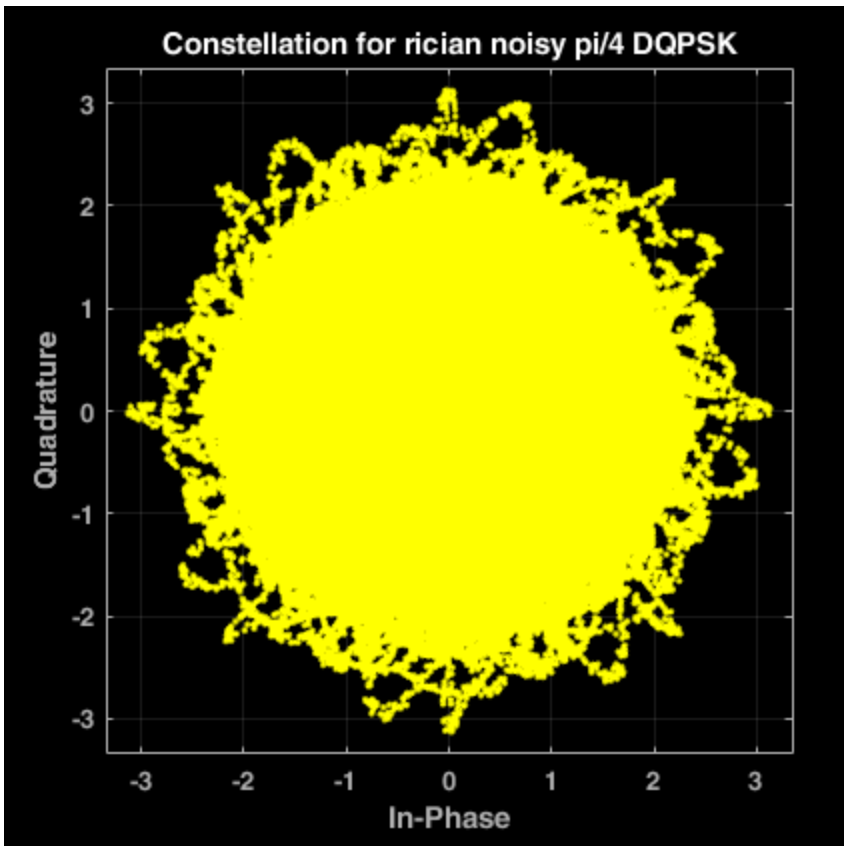

Constellation for pi/4 DQPSK
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

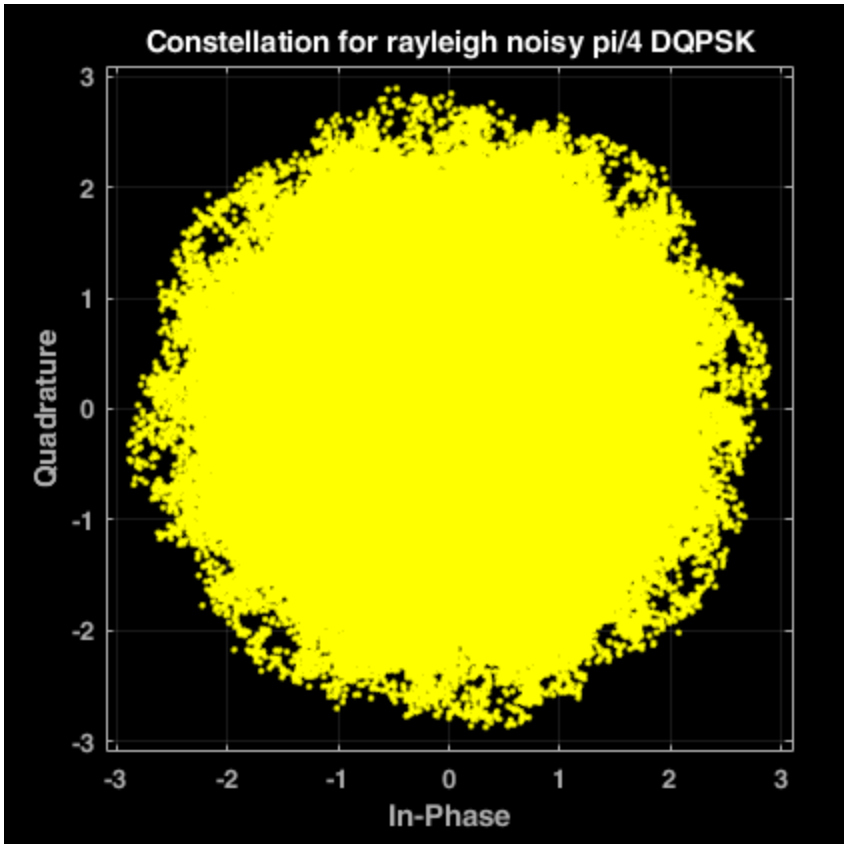

Constellation for 8-DPSK
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

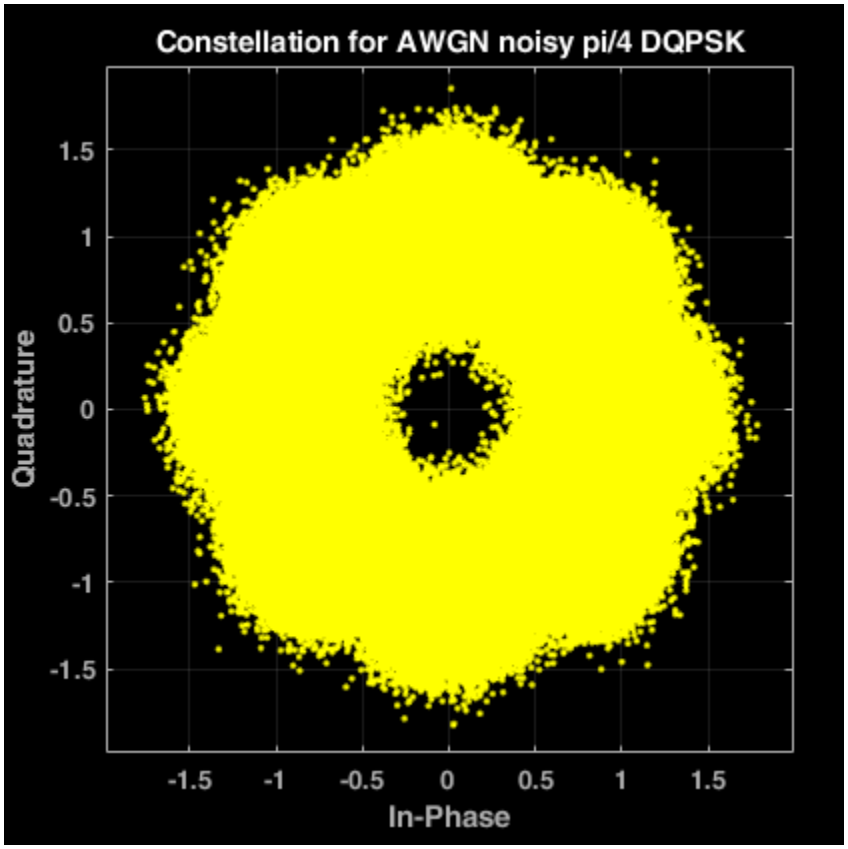
Channel

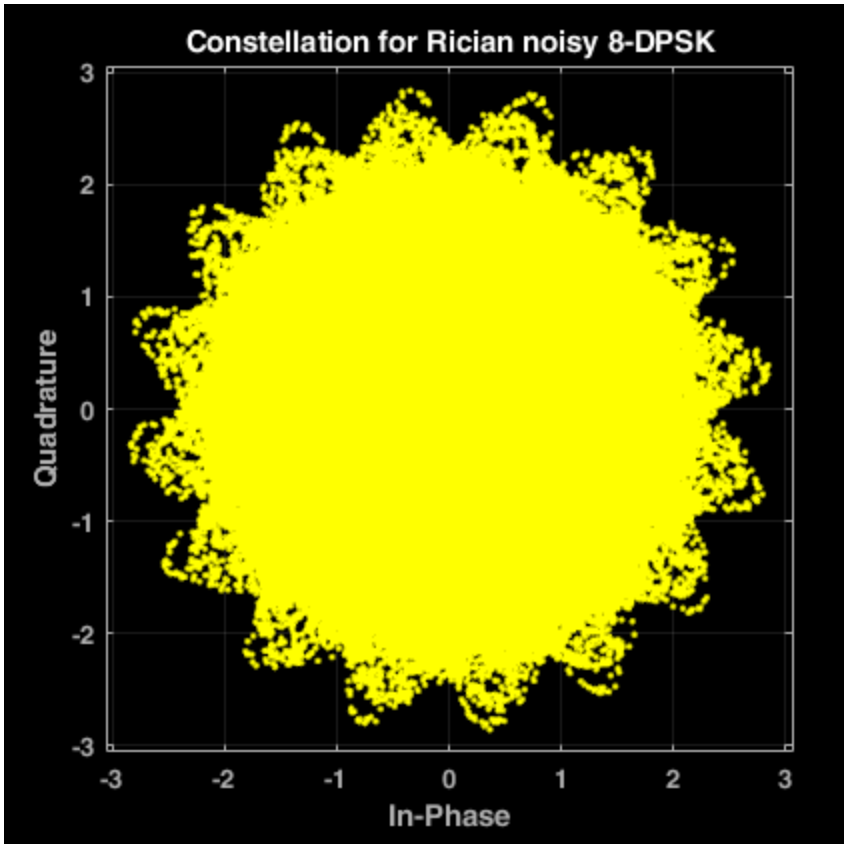
```
%simulation for fading channels and specifying specs according to
%(A. Soltanian and R. E. Van Dyck) Performance of the Bluetooth System
%in Fading Dispersive Channels and Interference paper.
%where max path time delay = 200 nsec, the path gain ranges from 3 to
 9 db
%MaximumDopplerShift = 30 while FadingTechnique is Filtered Gaussian
 noise

rayleighchan = comm.RayleighChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'PathDelays',[550 700 600]*1e-9, ...
    'AveragePathGains',[1.5 0.8 0.5], ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                 22, ...
    'FadingTechnique','Filtered Gaussian noise');

ricianchan = comm.RicianChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                22,...
    'PathDelays',[0.0 600 500]*1e-9, ...
    'AveragePathGains',[1 1.5 0.8], ...
    'KFactor',5, ...
    'FadingTechnique','Filtered Gaussian noise');

%The AWGN channel simulates the interfrence between more than 2
 bluetooth
%devices or with IEEE 802.11 WLAN.
AWGchan = comm.AWGNChannel('EbNo',9,'BitsPerSymbol',2);


AWGchan2 = comm.AWGNChannel('EbNo',9,'BitsPerSymbol',3);

%Channel effect applied on pi/4 DQPSK modulated signal
mod_bits_RG = rayleighchan(mod_bits);
mod_bits_rician = ricianchan(mod_bits);
mod_bits_AWGN = AWGchan(mod_bits);
scatterplot(mod_bits_rician);
title('Constellation for rician noisy pi/4 DQPSK'); grid on;
scatterplot(mod_bits_RG);
title('Constellation for rayleigh noisy pi/4 DQPSK'); grid on;
scatterplot(mod_bits_AWGN);
title('Constellation for AWGN noisy pi/4 DQPSK'); grid on;

%Channel effect applied on 8-DPSK modulated signal
mod_bits_RG2 = rayleighchan(mod_bits2);
mod_bits_rician2 = ricianchan(mod_bits2);
mod_bits_AWGN2 = AWGchan2(mod_bits2);
scatterplot(mod_bits_rician2);
title('Constellation for Rician noisy 8-DPSK'); grid on;
scatterplot(mod_bits_RG2);
```

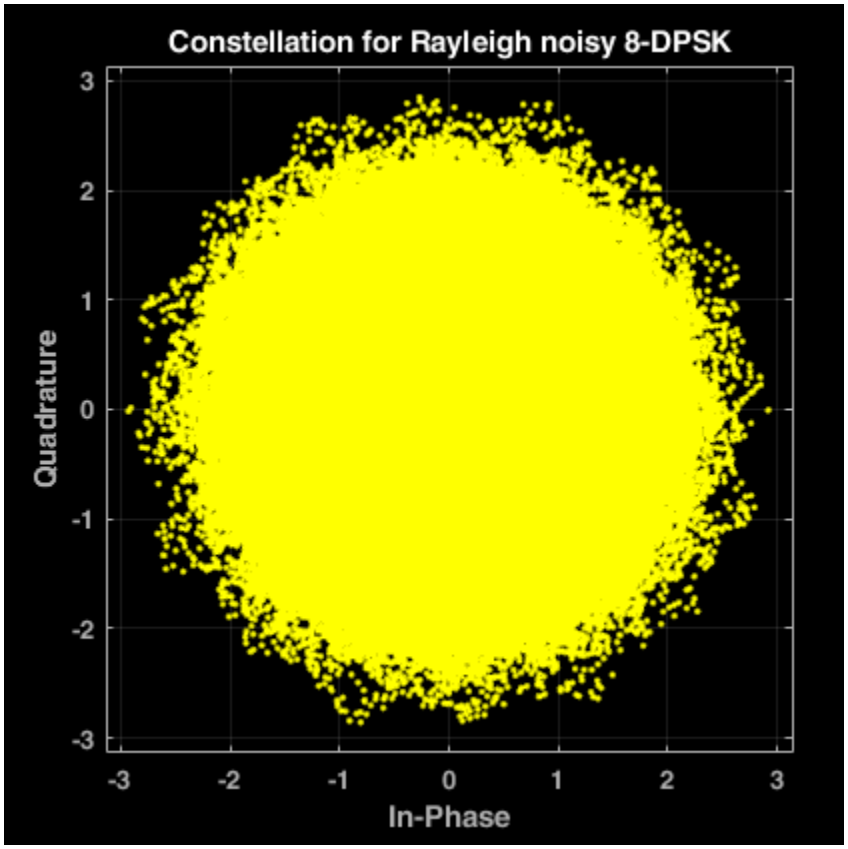
```
title('Constellation for Rayleigh noisy 8-DPSK'); grid on;
scatterplot(mod_bits_AWGN2);
title('Constellation for AWGN noisy 8-DPSK'); grid on;
```

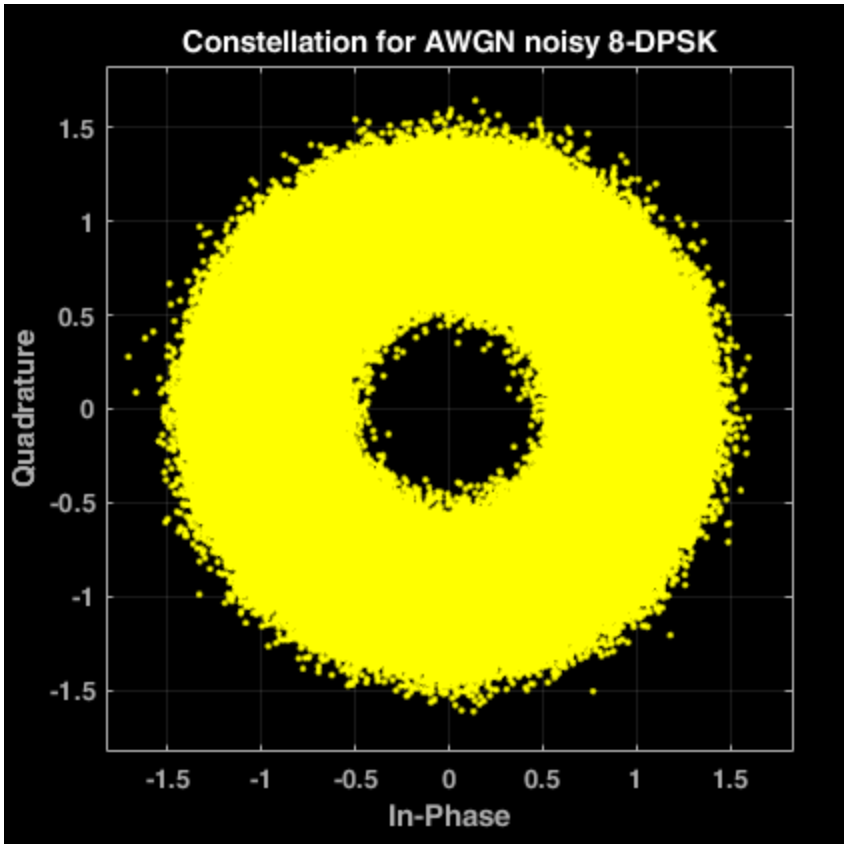



Constellation for rayleigh noisy pi/4 DQPSK
Quadrature
In-Phase
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


Constellation for AWGN noisy pi/4 DQPSK
Quadrature
In-Phase
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5


Constellation for Rician noisy 8-DPSK
Quadrature
In-Phase
3
2
1
0
-1
-2
-3


Constellation for Rayleigh noisy 8-DPSK
Quadrature
In-Phase
3
2
1
0
-1
-2
-3



Demodulator

```
%demodulate the noisy signal using pi/4 DQPSK
demodulator = comm.DQPSKDemodulator('BitOutput',true);
demod_RG = demodulator(mod_bits_RG);
demod_rician = demodulator(mod_bits_rician);
demod_AWGN = demodulator(mod_bits_AWGN);

%demodulate the noisy signal using 8-DPSK
demodulator2 = comm.DPSKDemodulator(8, pi/8, 'BitOutput',true);
demod_RG2 = demodulator2(mod_bits_RG2);
demod_rician2 = demodulator2(mod_bits_rician2);
demod_AWGN2 = demodulator2(mod_bits_AWGN2);
```

Decoder

```
% Decode the recived bits after demodulation, to be able to retrive
 the
% frequencies corrosponds to number of levels where dequantizing takes
% place in the decoder function

%pi/4 QPSK
Decoded_Signal_RG = Decoder(demod_RG', Levels);
if(NumChannels == 1)
    Output_AudioSignal_RG = Decoded_Signal_RG;
```

```
else
    Output_AudioSignal_RG =
 [Decoded_Signal_RG(1:length(Decoded_Signal_RG)/2)',
 Decoded_Signal_RG(length(Decoded_Signal_RG)/2 +1:end)'];
end
Decoded_Signal_Ri = Decoder(demod_rician', Levels);
if(NumChannels == 1)
    Output_AudioSignal_Ri = Decoded_Signal_Ri;
else
    Output_AudioSignal_Ri =
 [Decoded_Signal_Ri(1:length(Decoded_Signal_Ri)/2)',
 Decoded_Signal_Ri(length(Decoded_Signal_Ri)/2 +1:end)'];
end
Decoded_Signal_AW = Decoder(demod_AWGN', Levels);
if(NumChannels == 1)
    Output_AudioSignal_AW = Decoded_Signal_AW;
else
    Output_AudioSignal_AW =
 [Decoded_Signal_AW(1:length(Decoded_Signal_AW)/2)',
 Decoded_Signal_AW(length(Decoded_Signal_AW)/2 +1:end)'];
end


%8-DPSK
Decoded_Signal_RG2 = Decoder(demod_RG2', Levels);
if(NumChannels == 1)
    Output_AudioSignal_RG2 = Decoded_Signal_RG2;
else
    Output_AudioSignal_RG2 =
 [Decoded_Signal_RG2(1:length(Decoded_Signal_RG2)/2)',
 Decoded_Signal_RG2(length(Decoded_Signal_RG2)/2 +1:end)'];
end
Decoded_Signal_Ri2 = Decoder(demod_rician2', Levels);
if(NumChannels == 1)
    Output_AudioSignal_Ri2 = Decoded_Signal_Ri2;
else
    Output_AudioSignal_Ri2 =
 [Decoded_Signal_Ri2(1:length(Decoded_Signal_Ri2)/2)',
 Decoded_Signal_Ri2(length(Decoded_Signal_Ri2)/2 +1:end)'];
end
Decoded_Signal_AW2 = Decoder(demod_AWGN2', Levels);
if(NumChannels == 1)
    Output_AudioSignal_AW2 = Decoded_Signal_AW2;
else
    Output_AudioSignal_AW2 =
 [Decoded_Signal_AW2(1:length(Decoded_Signal_AW2)/2)',
 Decoded_Signal_AW2(length(Decoded_Signal_AW2)/2 +1:end)'];
end
```

Writing Demodulated Signals into Audio Files

```
audiowrite('Output Audio\RG_QPSK.wav',Output_AudioSignal_RG, fs);
audiowrite('Output Audio\Ri_QPSK.wav',Output_AudioSignal_Ri, fs);
```

```
audiowrite('Output Audio\AWGN_QPSK.wav',Output_AudioSignal_AW, fs);
audiowrite('Output Audio\RG_8DPSK.wav',Output_AudioSignal_RG2, fs);
audiowrite('Output Audio\Ri_8DPSK.wav',Output_AudioSignal_Ri2, fs);
audiowrite('Output Audio\AWGN_8DPSK.wav',Output_AudioSignal_AW2, fs);
```

BER comparision

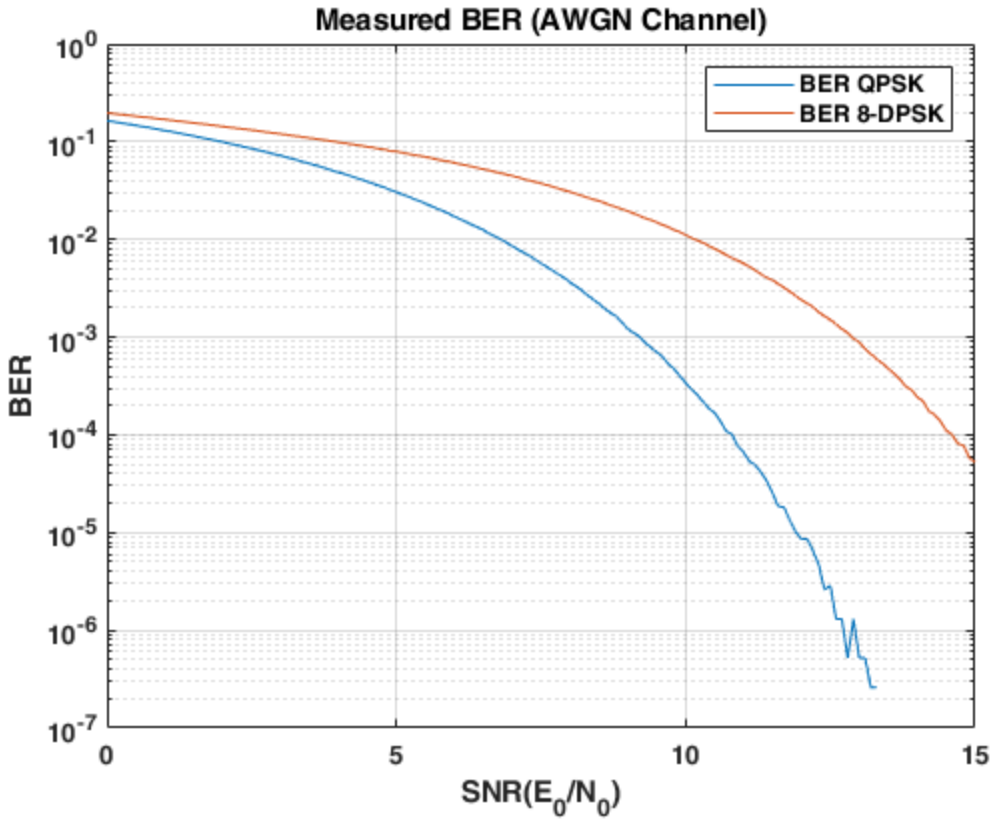
```
%finding the BER for each type of modulation used 1) pi/4 DQPSK 2) 8
 DPSK
%through each channel 1) Rayleigh Channel 2) Rician Channel 3) AWGN
 Channel

BER_RG_QPSK = biterr(Bit_Stream, demod_RG')
BER_Rician_QPSK = biterr(Bit_Stream, demod_rician')
BER_AWGN_QPSK = biterr(Bit_Stream, demod_AWGN')
BER_RG_8DPSK = biterr(Bit_Stream, demod_RG2')
BER_Rician_8DPSK = biterr(Bit_Stream, demod_rician2')
BER_AWGN_8DPSK = biterr(Bit_Stream, demod_AWGN2')
```

```
BER_RG_QPSK =

     1990851


BER_Rician_QPSK =

     1991055


BER_AWGN_QPSK =

        4892


BER_RG_8DPSK =

     1966174


BER_Rician_8DPSK =

     1966394


BER_AWGN_8DPSK =

       74421
```

BER Figures

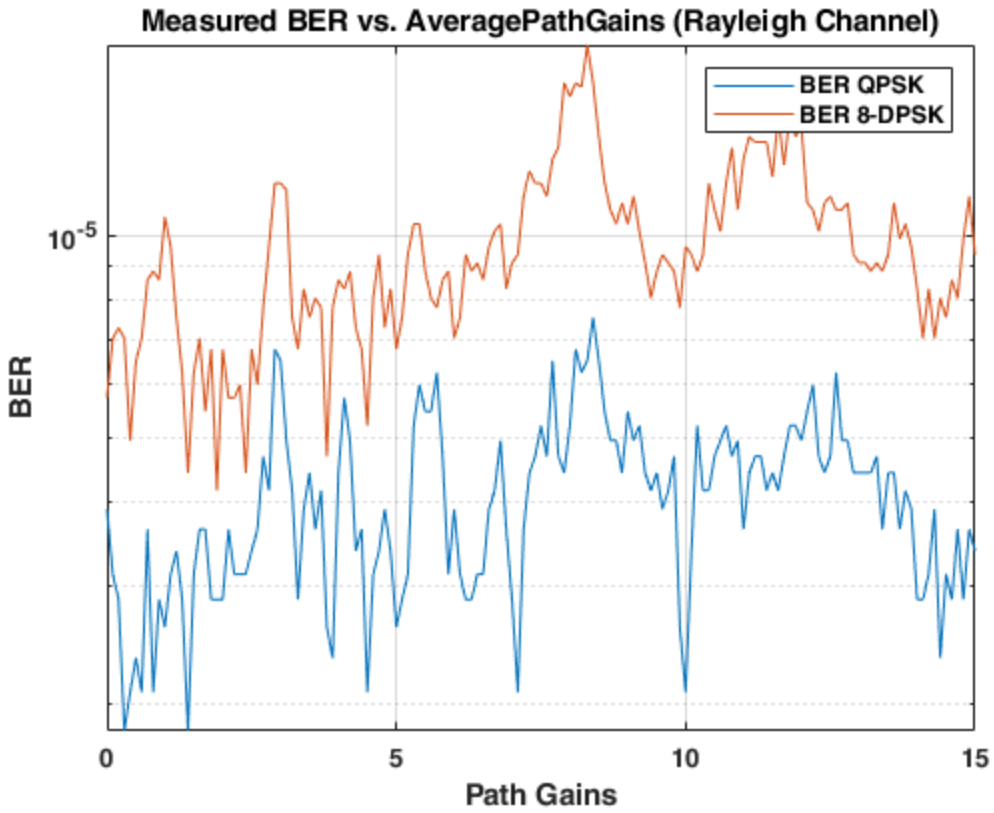
```
%AWGN
```

```
BER =[];
SNR = 0:0.1:15;
for i =1: length(SNR)
    awgnchannel = comm.AWGNChannel('EbNo',SNR(i),'BitsPerSymbol',2);
    audio_mod_awgn= awgnchannel(mod_bits);
    out_bits=(demodulator(audio_mod_awgn)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
figure();
semilogy(SNR, BER); title("Measured BER (AWGN Channel)");
 xlabel("SNR(E_0/N_0)"); ylabel("BER"); grid on; hold on;

BER =[];
SNR = 0:0.1:15;
for i =1: length(SNR)
    awgnchannel = comm.AWGNChannel('EbNo',SNR(i),'BitsPerSymbol',3);
    audio_mod_awgn= awgnchannel(mod_bits2);
    out_bits=(demodulator2(audio_mod_awgn)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
semilogy(SNR, BER); legend('BER QPSK', 'BER 8-DPSK');
```



BER Rayleigh Channel

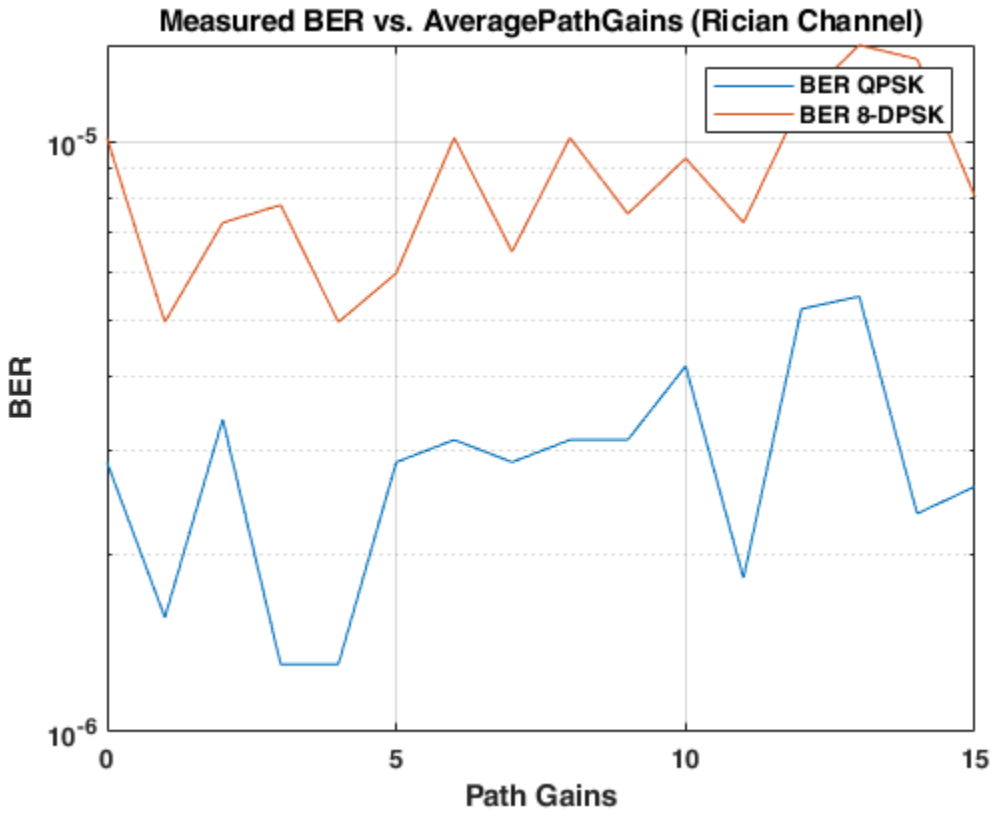
```
BER =[];
PathGains = 0:0.1:15;
for i =1: length(PathGains)
    rayleighchan = comm.RayleighChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'PathDelays',[0.0 200]*1e-9, ...
    'AveragePathGains',[1 PathGains(i)], ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                 22, ...
    'FadingTechnique','Filtered Gaussian noise');

    audio_mod_re= rayleighchan(mod_bits);
    out_bits=(demodulator(audio_mod_re)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
figure();
semilogy(PathGains, BER); title("Measured BER vs. AveragePathGains
 (Rayleigh Channel)"); xlabel("Path Gains"); ylabel("BER"); grid on;
 hold on;

BER =[];
PathGains = 0:0.1:15;
for i =1: length(PathGains)
    rayleighchan = comm.RayleighChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'PathDelays',[0.0 200]*1e-9, ...
    'AveragePathGains',[1 PathGains(i)], ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                 22, ...
    'FadingTechnique','Filtered Gaussian noise');

    audio_mod_re= rayleighchan(mod_bits2);
    out_bits=(demodulator2(audio_mod_re)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
semilogy(PathGains, BER); legend('BER QPSK', 'BER 8-DPSK');
```



BER Rician Channel

```
BER =[];
PathGains = 0:1:15;
for i =1: length(PathGains)
    rayleighchan = comm.RicianChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                22,...
    'PathDelays',[0.0 100 200]*1e-9, ...
    'AveragePathGains',[1 0.5 PathGains(i)], ...
    'KFactor',5, ...
    'FadingTechnique','Filtered Gaussian noise');

    audio_mod_re= rayleighchan(mod_bits);
    out_bits=(demodulator(audio_mod_re)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
figure();
semilogy(PathGains, BER); title("Measured BER vs. AveragePathGains
 (Rician Channel)"); xlabel("Path Gains"); ylabel("BER"); grid on;
 hold on;

BER =[];
```

```
PathGains = 0:1:15;
for i =1: length(PathGains)
    rayleighchan = comm.RicianChannel('SampleRate', fs,...
    'MaximumDopplerShift',30, ...
    'RandomStream','mt19937ar with seed', ...
    'Seed',                22,...
    'PathDelays',[0.0 100 200]*1e-9, ...
    'AverageGains',[1 0.5 PathGains(i)], ...
    'KFactor',5, ...
    'FadingTechnique','Filtered Gaussian noise');

    audio_mod_re= rayleighchan(mod_bits2);
    out_bits=(demodulator2(audio_mod_re)');
    [nu,ratio]=biterr(out_bits,Bit_Stream);
    BER=[BER,ratio];
end
semilogy(PathGains, BER); legend('BER QPSK', 'BER 8-DPSK');
```



Published with MATLAB® R2021a