EE 314 TERM PROJECT
FINAL REPORT

5G NR PDSCH and PUSCH PHY Waveform Design (LDPC Coding) and Tests

Erkin Koç / 26880

Ömer Said Öztürk / 26471

Utku Gürsoy / 26572

Suat Utku İnal / 27838

Faculty of Engineering and Natural Sciences

10.06.2022

TABLE OF CONTENTS

INTRODUCTION

Low-Density Parity-Check (LDPC) codes are linear error-correcting codes. According to the 3rd Generation Partnership Project (3GPP), two channel coding codes, i.e., Polar codes and LDPC codes, are recommended for the Fifth-generation (5G) New Radio (NR). Polar codes are applied to 5G NR control channels. LDPC codes are suitable for 5G NR shared channels due to its high throughput, low latency, low decoding complexity and rate compatibility. LDPC codes can be used to different block sizes with varying code rates because of the design of rate-compatible base graphs. Another advantage of 5G NR LDPC codes is that the performance of LDPC codes has an error floor around or below block error rate (BLER) 10^{-5} for all code sizes and code rates. So LDPC codes play an important role in channel coding for 5G communication.

PROJECT BACKGROUND

Project Description

5G NR PDSCH and PUSCH PHY waveform design (LDPC Coding) and tests

- Tx Chain: Data Buffering, Data Segmentation and CRC Encoding, Filler Bits Adding/Extraction, LDPC Encoding, Rate Matching, Interleaver, Modulation, pulse shaping filter. BPSK Modulation.
- Impairments: Channel (AWGN)
- RX Chain: Soft-Decision / Hard-Decision Demodulator, Deinterleaver, Rate Recovery, LDPC Decoder, Filler Bits Adding/Extraction, Data Desegmentation and CRC Control
- BER simulations for different frame lengths, coding rates and modulation types. Constellation Diagrams, Eye Diagrams.

Literature Review

Low-Density Parity-Check Codes (LDPC Codes)

LDPC code is a method of transmitting a message over a noisy transmission channel.

Data delivered from the MAC layer to the physical layer is termed as a transport block. For the downlink shared channel (DL-SCH), a transport block goes through the processing stages of:

- CRC attachment,
- Code block segmentation and code block CRC attachment,
- Channel coding using LDPC,
- Rate matching and code block concatenation

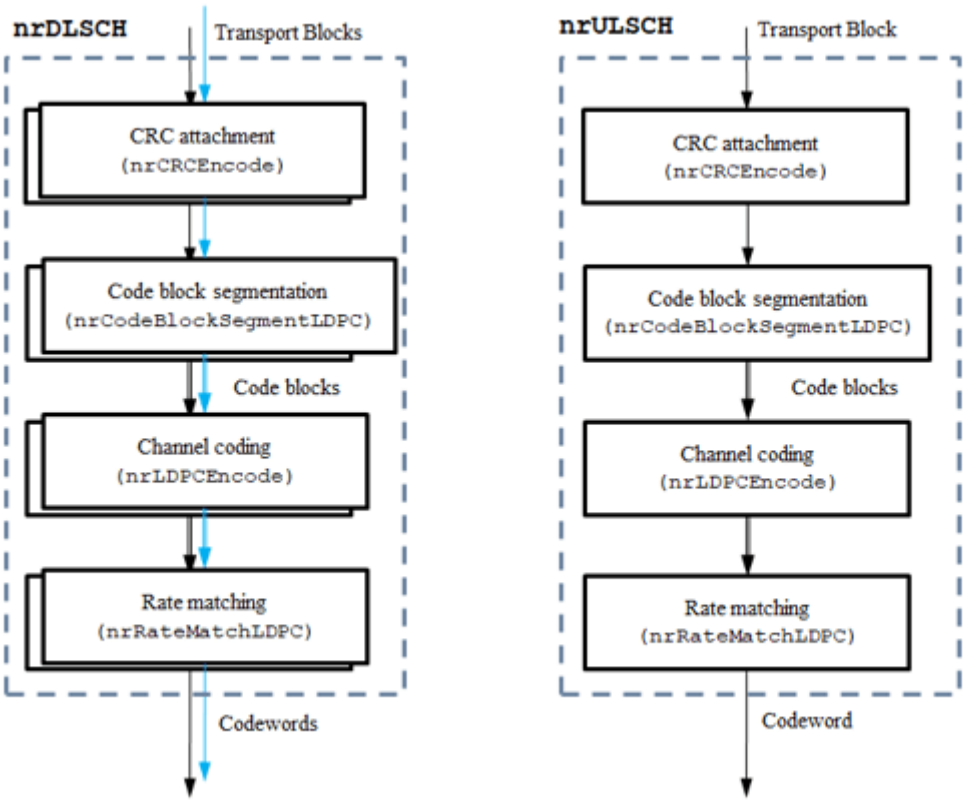
before being passed on to the physical downlink shared channel (PDSCH) for scrambling, modulation, layer mapping and resource/antenna mapping.

The output number of bits from the rate matching and code block concatenation process must match the bit capacity of the PDSCH, based on the available resources.

Similar processing applies for the UL-SCH, where the physical uplink shared channel (PUSCH) is the recipient of the UL-SCH codeword. The schematics in Figure 1 depict the processing for the two channels.

The receive end processing for the DL-SCH channel comprises of the corresponding dual operations to the transmit end that include:

- Rate recovery
- LDPC decoding
- Code block desegmentation and CRC decoding
- Transport block CRC decoding



(Figure 1: LDPC processing for the two channels)

The DL-SCH encoding process consists of cyclic redundancy check (CRC), code block segmentation and CRC, low-density parity-check (LDPC) encoding, rate matching, and code block concatenation.

The DL-SCH decoding process consists of rate recovery, low-density parity-check (LDPC) decoding, desegmentation, and cyclic redundancy check (CRC) decoding.

Data Segmentation and CRC Encoding

Data segmentation fragments a large transport block into smaller code blocks to reduce memory requirements of the interleaver. Cyclic-Redundancy-Check (CRC) is an error-detecting code commonly used in digital networks to detect accidental changes to digital data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. CRC bits expand the message without adding any information.

Rate Matching

The task of rate matching is to extract from the blocks of code bits, delivered by the LDPC encoder, to the exact set of bits to be transmitted within a given transmission time interval, depending on the existing channel conditions.

Interleaver

Purpose of the interleaver is to reduce spurious noise that can corrupt data. Interleaver disperses sequence of bits in a bit stream to minimize the effect of burst errors introduced in transmission. Error detecting codes cannot work successfully is there is an error burst in a transmission channel. Hence, the bits of the codeword is permuted in order to prevent the accumulation of the error bits is a specific portion of the data.

Modulation

Data to be transmitted has to be modulated before transmitting through a channel, and it has to be demodulated at the receiver end. Without modulation, information would be lost along the channel, since we use frequencies of the message carriers in to obtain and detect the desired message.

Pulse Shaping

Bits are send over the channel as electrical pulses. Designing the pulse waveform is important to inhibit ISI.

Methodology / Tools

Tools:

- ➔ Software: MATLAB version 2021b
- ➔ 5G Toolbox

Methodology:

- ➔ Generation of MATLAB scripts as simulation templates
- ➔ Changing frame length, coding rate and modulation parameters one at a time
- ➔ Obtaining corresponding eye diagrams, constellations and BER simulations for each simulation
- ➔ Interpreting the results

DL-SCH coding parameters

The DL-SCH coding parameter applies the downlink shared channel (DL-SCH) encoder processing chain to one or two transport blocks. The DL-SCH encoding process consists of cyclic redundancy check (CRC), code block segmentation and CRC, low-density parity-check (LDPC) encoding, rate matching, and code block concatenation.

% Random transport block data generation

The signal is created randomly with length A with a magnitude of at most 1.

```
% Random transport block data generation
in = randi([0 1],A,1,'int8');   % returns an Ax1 matrix whose elements are ranged in between [0,1] and data type 8 bit integer
```

% Transport block CRC attachment

Transport block CRC calculates the CRC defined by the polynomial for the input data. CRC encoded data is returned, which is a copy of the input data with the CRC parity bits appended.

```
% Transport block CRC attachment
tbIn = nrCRCEncode(in,cbsInfo.CRC); % returns the CRC encoded data, a copy of the input data with the CRC parity bits appended
```

% Code block segmentation and CRC attachment

LDPC code block segmentation and CRC attachment split the input data block into code block segments based on the base graph number. Cyclic redundancy check (CRC) and filler bits to each code block segment are appended (if applicable). LDPC code block segmentation provides input to low-density parity-check (LDPC) codes in transport channels, including downlink and uplink shared channels, and paging channels.

```
% Code block segmentation and CRC attachment
cbsIn = nrCodeBlockSegmentLDPC(tbIn,cbsInfo.BGN); % splits the input data block into code block segments based on the base graph number,
                                                  % appends cyclic redundancy check (CRC) and filler bits to each code block segment
```

% LDPC encoding

Low-density parity-check (LDPC) encoding is used for returning the LDPC-encoded output matrix. Filler bits that have been appended before are represented by -1 and are treated as 0 when performing encoding. The encoding includes the puncturing of some of the systematic information bits.

```
% LDPC encoding
enc = nrLDPCEncode(cbsIn,cbsInfo.BGN); % returns the LDPC-encoded output matrix
                                       % Filler bits are represented by -1 and are treated as 0 when performing encoding
```

% Rate matching and code block concatenation

Rate matching and code block concatenation include the stages of bit selection and interleaving defined for LDPC-encoded data and code block concatenation.

```
% Rate matching and code block concatenation
outlen = ceil(A/rate);
chIn = nrRateMatchLDPC(enc,outlen,rv,modulation,nlayers);   % includes the stages of bit selection and interleaving defined for LDPC-encoded data
                                                            % and code block concatenation
```

% Modulation

M-ary PSK modulator modulates a baseband representation of the modulated signal using the M-ary phase-shift keying method implemented on a graphics processing unit (GPU).

```
% Modulation
modData = pskModulator(chIn);
```

% Pulse Shaping Filter

Pulse Shaping filter returns a raised cosine transmit FIR filter System object, interpolating an input signal using a raised cosine FIR filter. The filter uses an efficient polyphase FIR interpolation structure and it has unit energy.

```
% Waveform Design - Tx
chIn = txfilter(modData);
```

% Channel

Additive White Gaussian Noise (AWGN) is added as the signal sent through the channel.

```
% AWGN Channel
channelOutput = channel(modData);
```

%% Receive Processing using LDPC Decoding

The receive end processing for the DL-SCH channel comprises of the corresponding dual operations to the transmit end that include Rate recovery, LDPC decoding, Code block desegmentation and CRC decoding, and Transport block CRC decoding.

```
% Waveform Design - Rx
demodData = rxfilter(channelOutput);

% Demodulation
demodOut = pskDemodulator(channelOutput);
```

% Rate recovery

Rate recovery is the inverse of rate match and performs the inverse of the code block concatenation, bit interleaving, and bit selection stages at the receiver end.

```
% Rate recovery
raterec = nrRateRecoverLDPC(demodOut,A,rate,rv,modulation,nlayers);
% the inverse of nrRateMatchLDPC and performs the inverse of the code block concatenation,
% bit interleaving, and bit selection stages at the receiver end.
```

% LDPC decoding

Low-density parity-check (LDPC) decoding returns the LDPC-decoded output matrix out for the input data matrix in, base graph number, and a maximum number of decoding iterations. This block also returns the actual number of iterations and the final parity checks per its codeword.

```
% LDPC decoding
decBits = nrLDPCDecode(raterec,cbsInfo.BGN,25);
```

% Code block desegmentation and CRC decoding

LDPC code block desegmentation and CRC decoding concatenate the input code block segments into a single output data block of its length. Data is validated by dimensions of the input based on the specified base graph number and output block length. Filler bits are removed and type-24B cyclic redundancy check (CRC) bits are present in the input code block segmentation. The output is the result of the type-24B CRC decoding (if applicable). This process is the inverse of the low-density parity-check (LDPC) code block segmentation

```
% Code block desegmentation and CRC decoding
[blk,blkErr] = nrCodeBlockDesegmentLDPC(decBits,cbsInfo.BGN,A+cbsInfo.L);
% removes any filler bits and type-24B cyclic redundancy check (CRC) bits present in the input
% inverse of the low-density parity-check (LDPC) code block segmentation
```

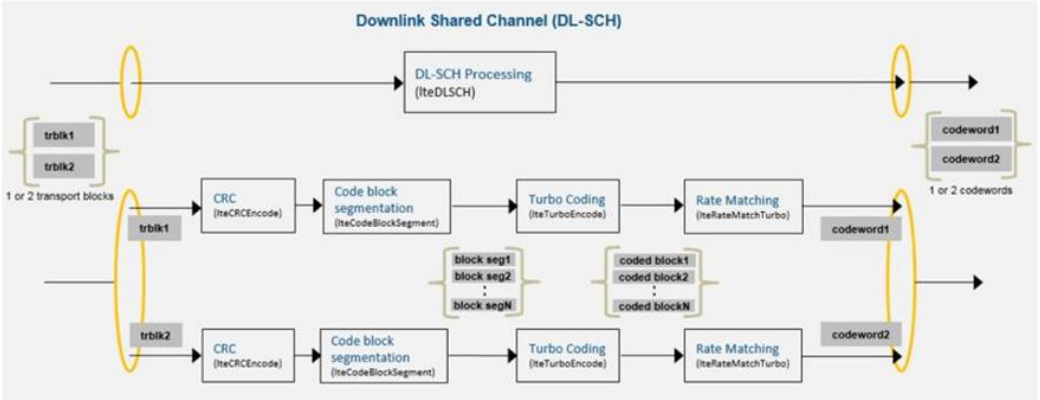
% Transport block CRC decoding

Decode and remove cyclic redundancy check checks the input data for a CRC error. We assume that the input data comprises the CRC parity bits associated with the polynomial. The function returns CRC decoded data which is the data part of the input CRC encoded data. We also observe logical difference (XOR) between the CRC comprised in the input and the CRC recalculated across the data part of the input. If the Logical CRC difference is not equal to 0, either an error has occurred or the input CRC has been masked.

```
% Transport block CRC decoding
[out,tbErr] = nrCRCDecode(blk,cbsInfo.CRC);
```

Notes:

- The output number of bits from the rate matching and code block concatenation process must match the bit capacity of the PDSCH, based on the available resources. In this example, as the PDSCH is not modeled, this is set to achieve the target code rate based on the transport block size previously selected.
- Similar processing applies for the UL-SCH, where the physical uplink shared channel (PUSCH) is the recipient of the UL-SCH codeword.



(Figure 2: DL-SCH Processing Chain)*

* In Figure 2, Turbo Coding is replaced with LDPC Coding in 5G channels.

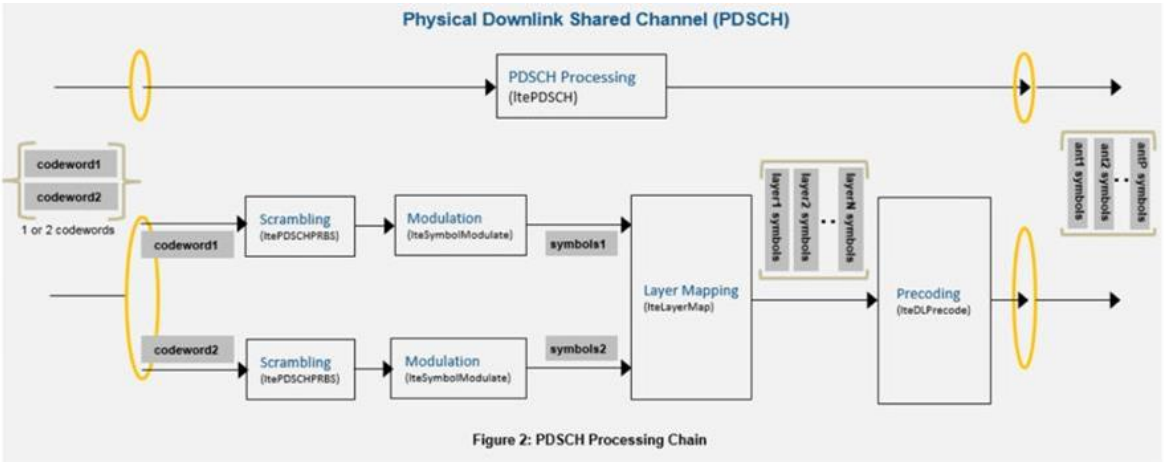
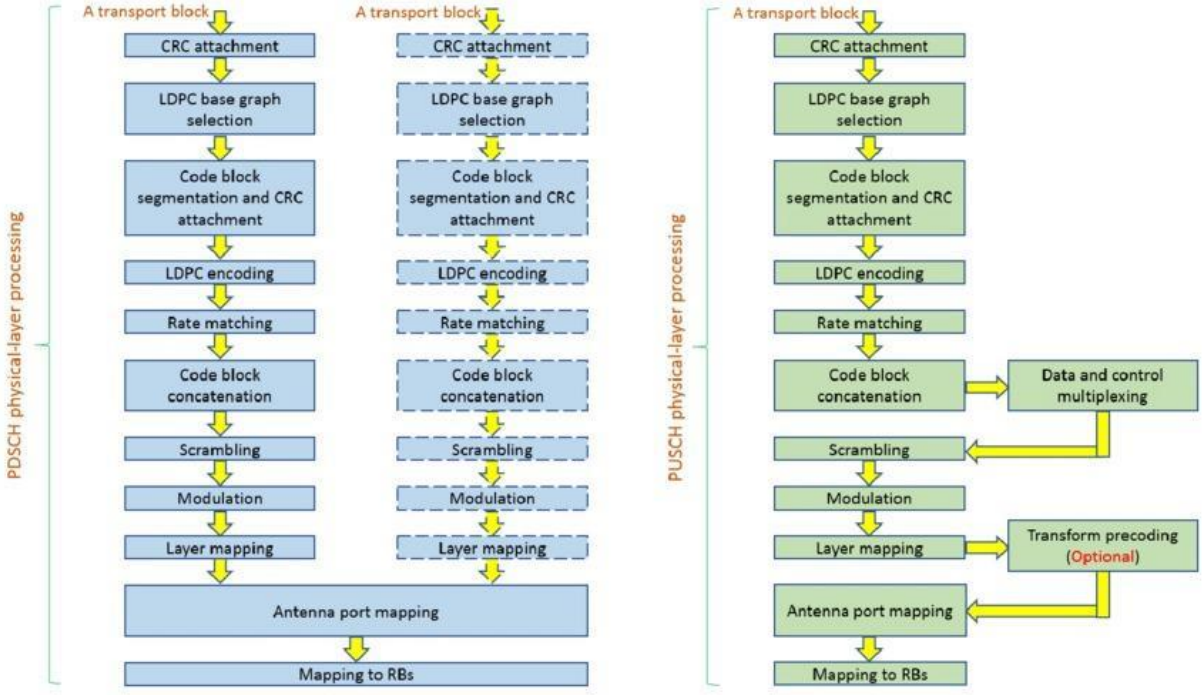


Figure 2: PDSCH Processing Chain

(Figure 3: PDSCH Processing Chain)



(Figure 4: PDSCH and PUSCH layer comparison)

Mathematics of LDPC Coding

Definition and historical background:

LDPC (low-density parity-check) codes are a class of linear block code. The term "low density" refers to the characteristic of the parity check matrix which contains only a few '1's in comparison to '0's. LDPC codes are arguably the best error correction codes in existence at present. LDPC codes were first introduced by R: Gallager in his Phd thesis in 1960 and soon forgotten due to introduction of Reed-Solomon codes and the implementation issues with limited technological knowhow at that time. The LDPC codes were rediscovered in mid 90s by R. Neal and D. Mackay at the Cambridge University.

Define N bit long LDPC Code in terms of M number of parity check equations and describing those parity check equations with a $M \times N$ parity check matrix $\boldsymbol{H}$, i.e.

- Let M be the number of parity check equations
- Let N be the number of bits in the codeword

Regular LDPC Codes: The number of '1's in any row of parity check matrix H will be equal and the same applies to column also. An example of regular H matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Irregular LDPC Codes: The number of '1's will be different in rows and columns of parity check matrix $\boldsymbol{H}$. An example of irregular parity check matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- Row weight (W_R) – number of '1's in a row
- Column wight (W_C) – number of '1's in a column

The parity check matrix defines a rate $R = \frac{K}{N}$, (N, K) code where $K = N - M,\ N > M$

Codeword is said to be vaild if it satisfies the **syndrome** calculation equation:

$$\boldsymbol{z} = \boldsymbol{c} \cdot \boldsymbol{H^T} = \mathbf{0}$$

where **c** is the codeword matrix defined as

$$\boldsymbol{c} = \boldsymbol{M} \cdot \boldsymbol{G}$$

where **M** is the message matrix and **G** is the generator matrix which is obtained by taking the transpose of the parity matrix **P** and **I** is the identity matrix:

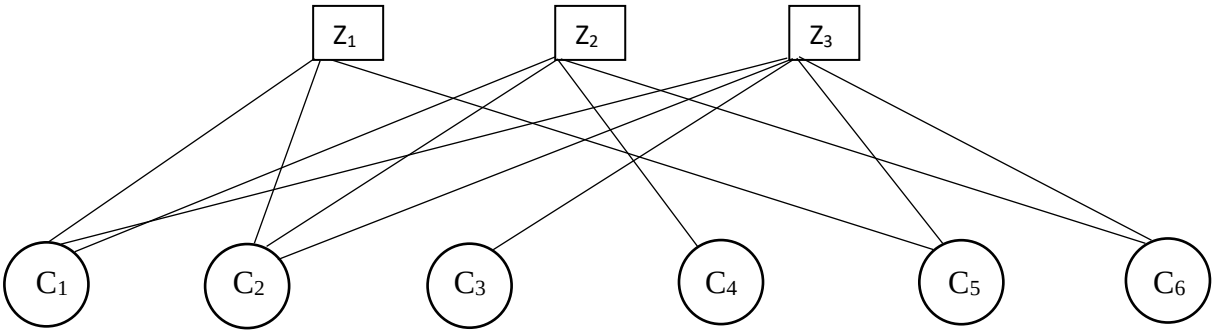
$$\boldsymbol{G} = [\boldsymbol{P}^{\boldsymbol{T}}_{K \times M} \quad | \boldsymbol{I}_K]$$

$$\boldsymbol{G} \cdot \boldsymbol{H^T} = \mathbf{0}$$

Tanner Graph Representation of the parity check matrix:

Parity check matrix specifies the parity check equations. Tanner Graph is a bipartite graph consists of N number of variable nodes and M number of check nodes. m^{th} check node is connected to n^{th} variable node if and only if n^{th} element in m^{th} row (h_{mn}) in parity check matrix $\boldsymbol{H}$ is a '1':

e.g. Tanner Graph of $\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$ can be constructed as:



Where Z_i are the variable nodes, and C_j are the check nodes.

- $C_1 \oplus C_2 \oplus C_5 = 0$
- $C_1 \oplus C_2 \oplus C_4 \oplus C_6 = 0$
- $C_1 \oplus C_2 \oplus C_3 \oplus C_5 \oplus C_6 = 0$

Mathematics of Cyclic Redundancy Check (CRC)

Append k redundant bits of '0's to the end of message frame M of length m. Transceiver and receiver has the same polynomial coefficient bits P of length p. We repeatedly divide M to P until a remainder K is obtained of length k. Then, we substitute the redundant '0' bits with this remainder K and send resultant bit stream. At the receiver side, we repeatedly divide the receiver bit stream to P. If the result is zero, it means that no error occurred during the transmission.

* It is still possible to have zero as result and still have bit errors, but this is very unlikely. There are other methods to figure these kind of error situations.

Tx side:

```
1101 ) 10110 000 ( 11001
       1101
       1100
       1101
         0010
         0000
           0100
           0000
             1000
             1101
              101 (CRC Bit)
```

Data plus extra zeroes

Rx side:

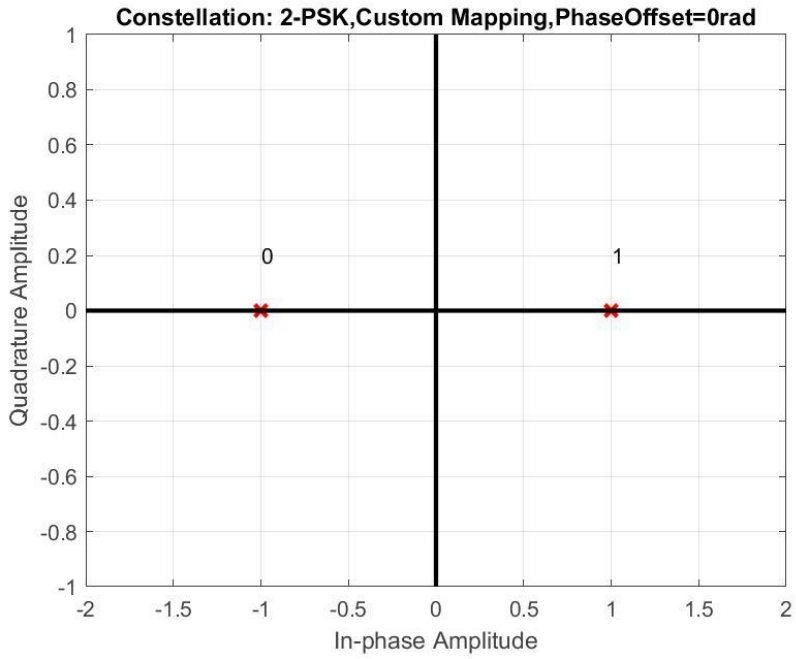
```
1101 ) 10110110  ( 11001
       1101
       1100
       1101
         0011
         0000
           0110
           0000
             1100
             1101
              001 (CRC Bit)
```

Since the remainder bits in Rx side is not zero, we conclude that error has occurred during the transmission.

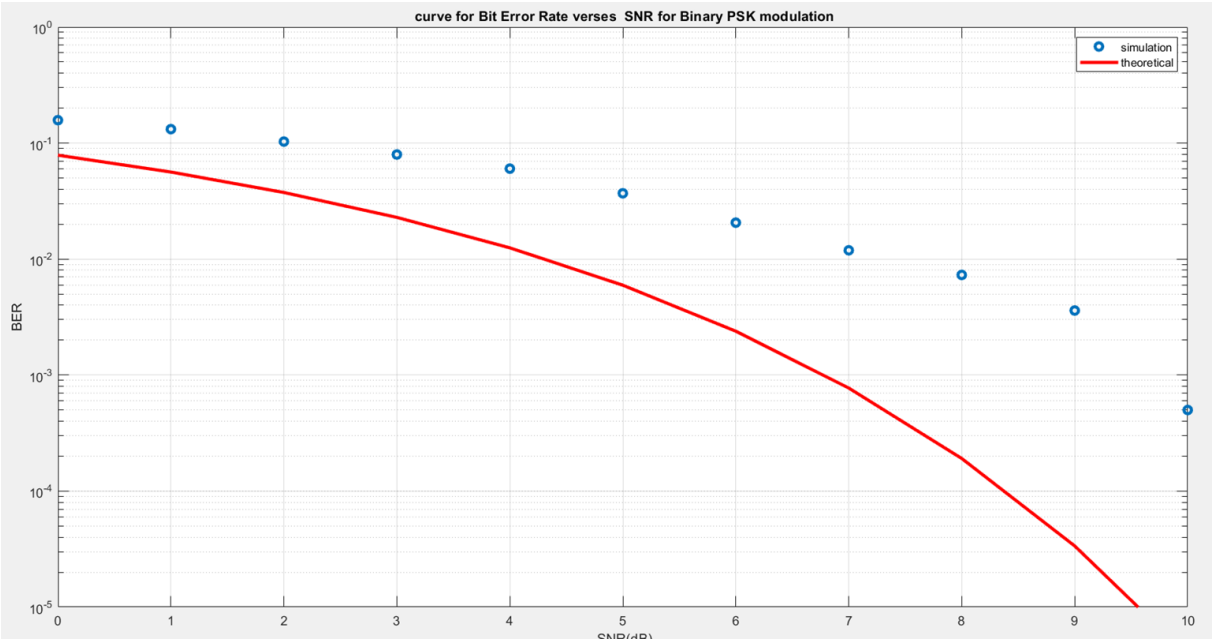
RESULTS

Waveform: Raised Cosine

BPSK Simulations for Waveform Desgin

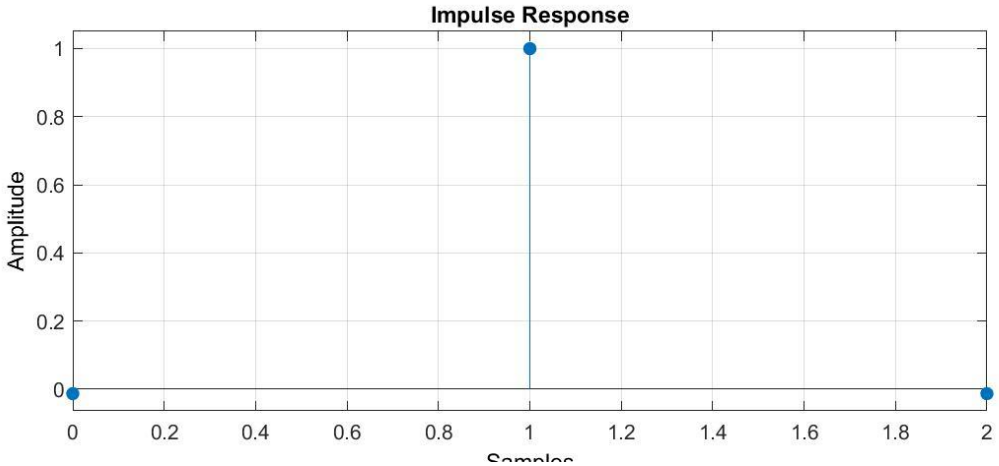


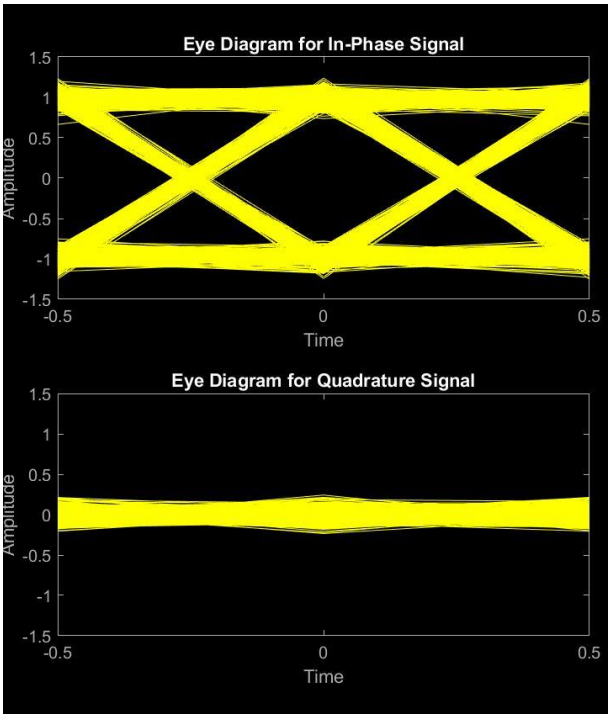
BER Simulation:

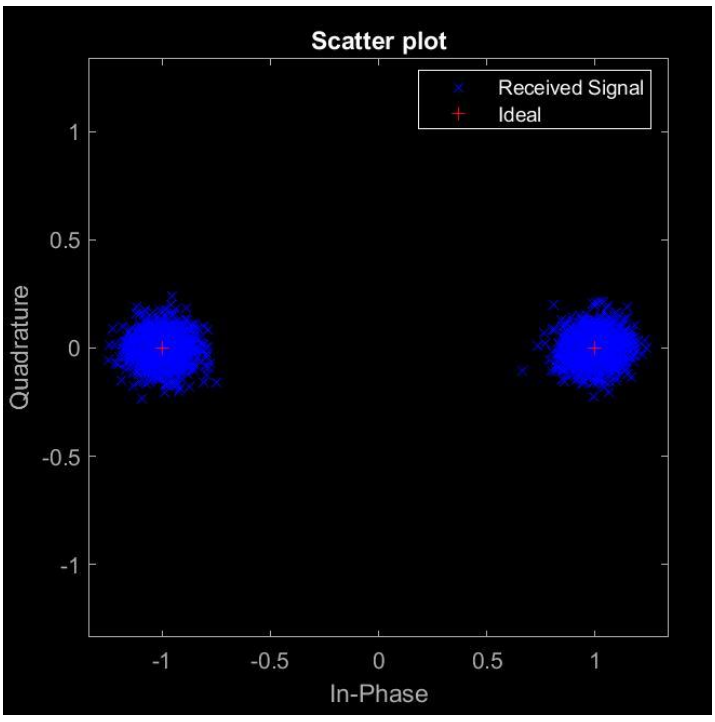


Frame length: 1000 Rate: 449/1024, Span: 2, Roll-off factor: 0.05

Waveform:

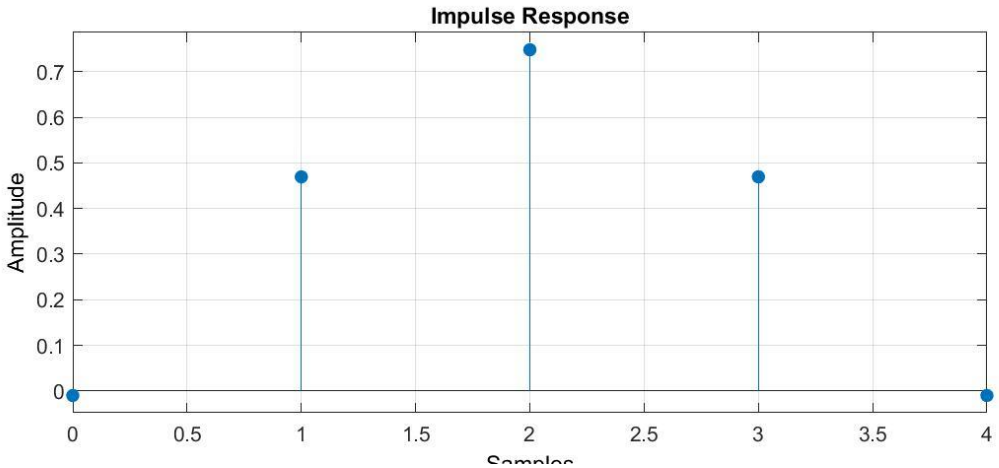


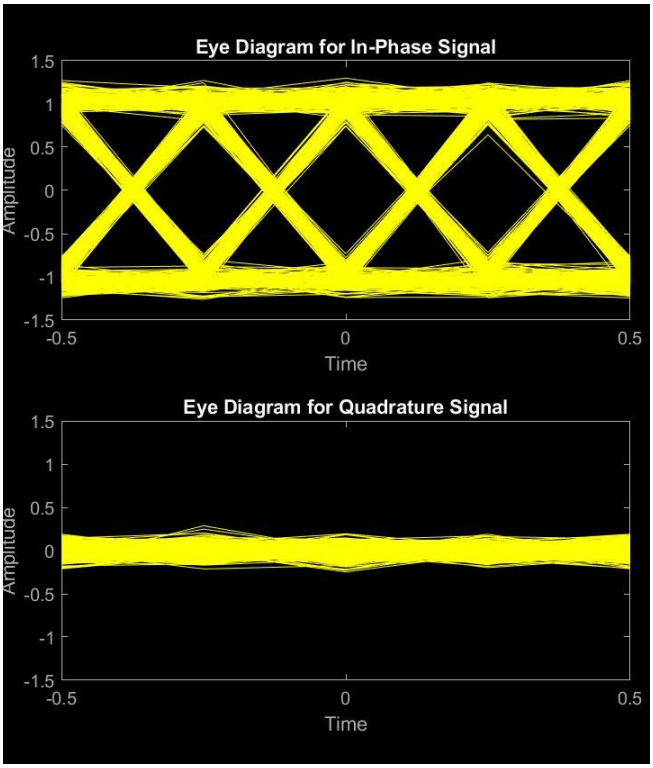


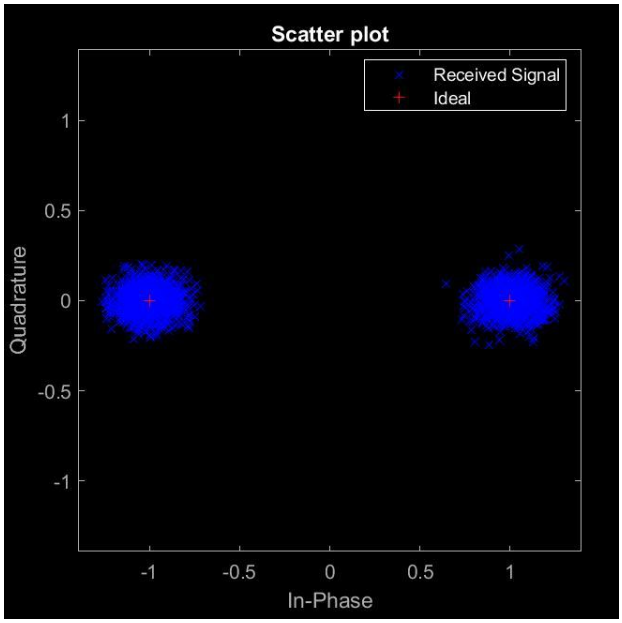


Frame length: 1000 Rate: 449/1024, Span: 2, Roll-off factor: 0.5

Waveform:

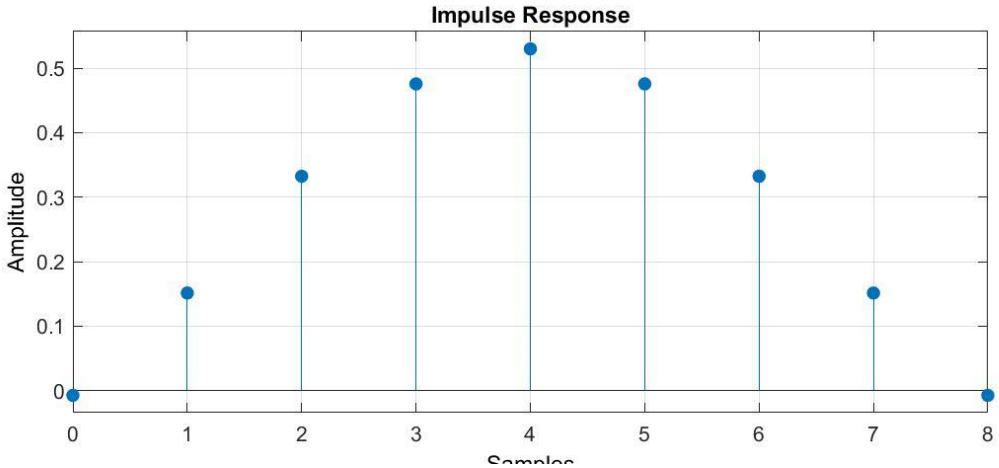


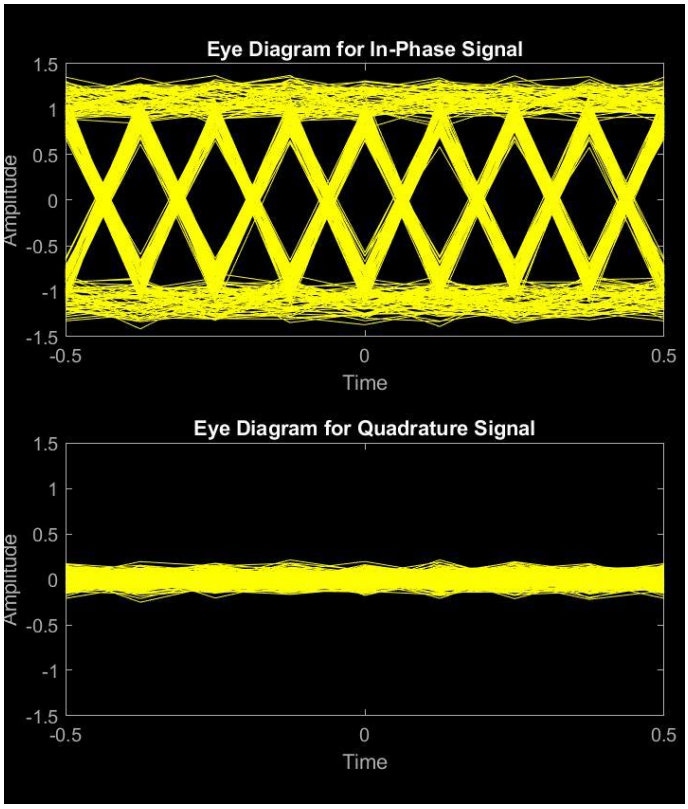


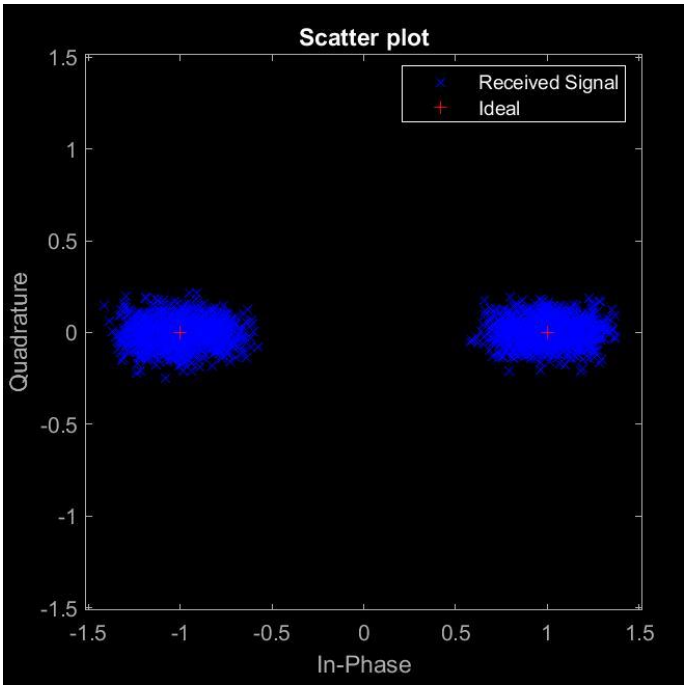


Frame length: 1000 Rate: 449/1024, Span: 4, Roll-off factor: 0.5

Waveform:

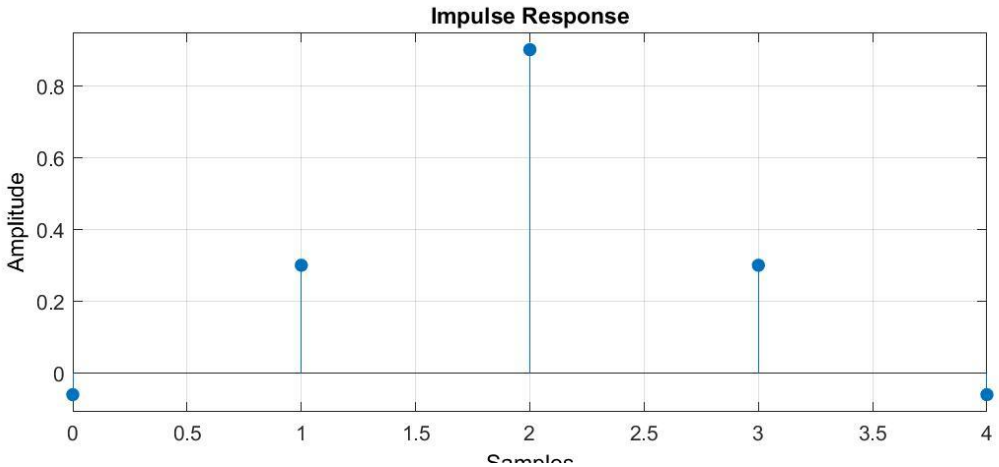


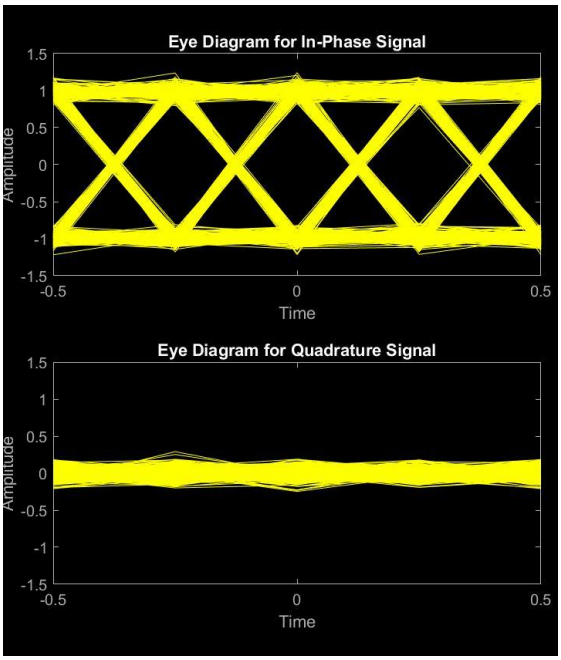


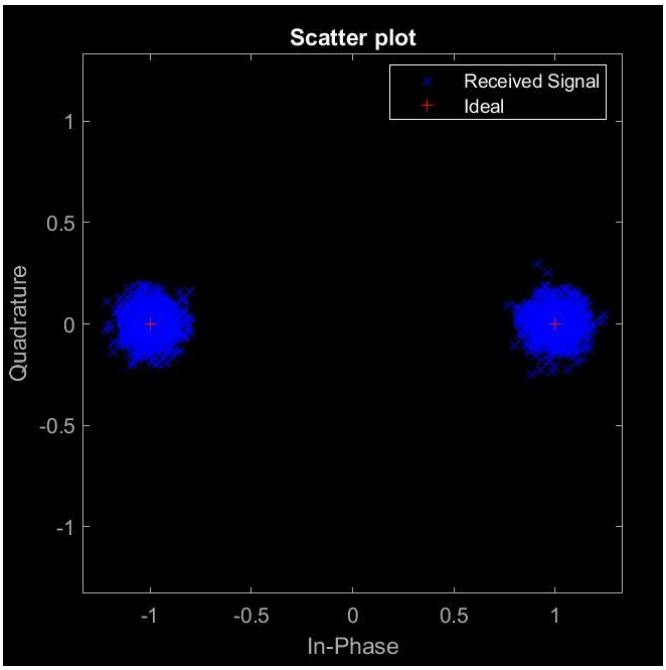


Frame length: 1000 Rate: 449/1024, Span: 2 Roll-off factor: 1

Waveform:

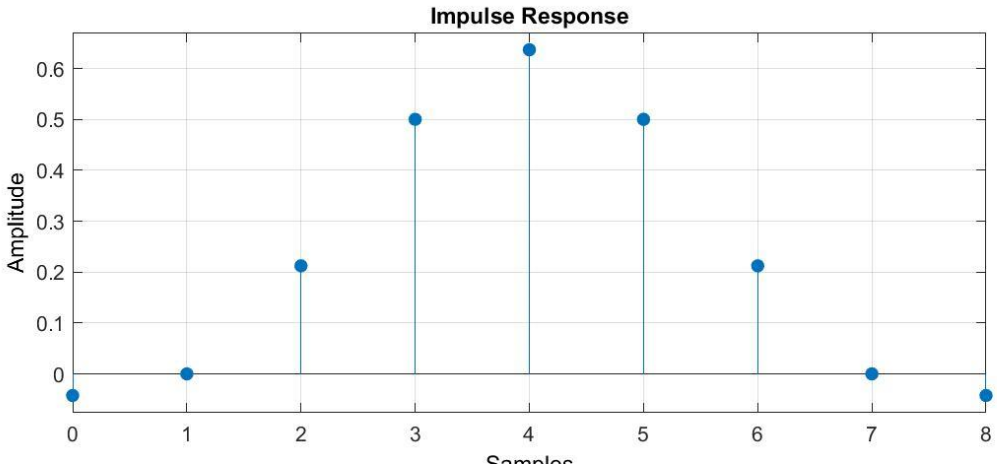


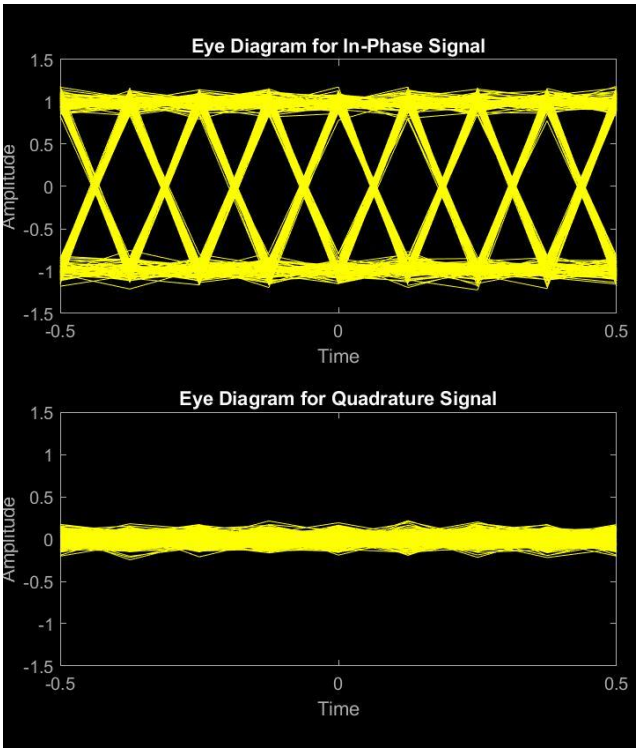


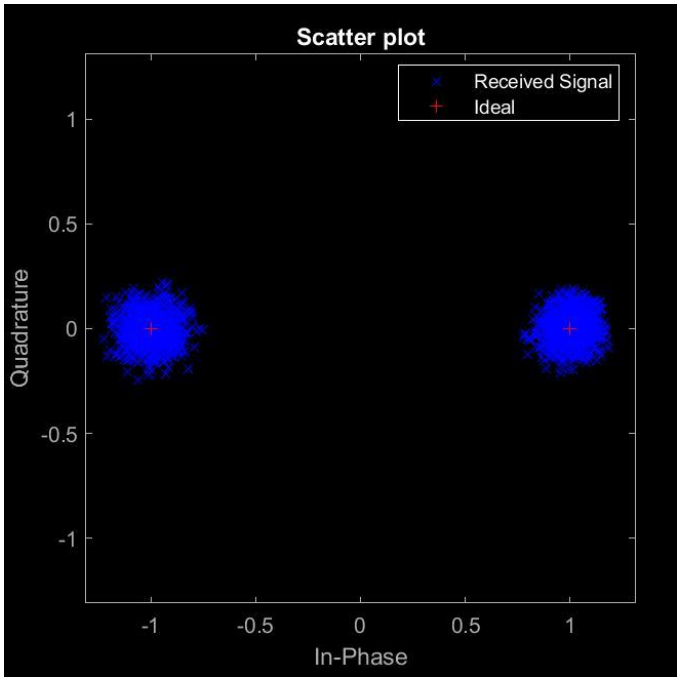


Frame length: 1000 Rate: 449/1024, Span: 6 Roll-off factor: 1

Waveform:

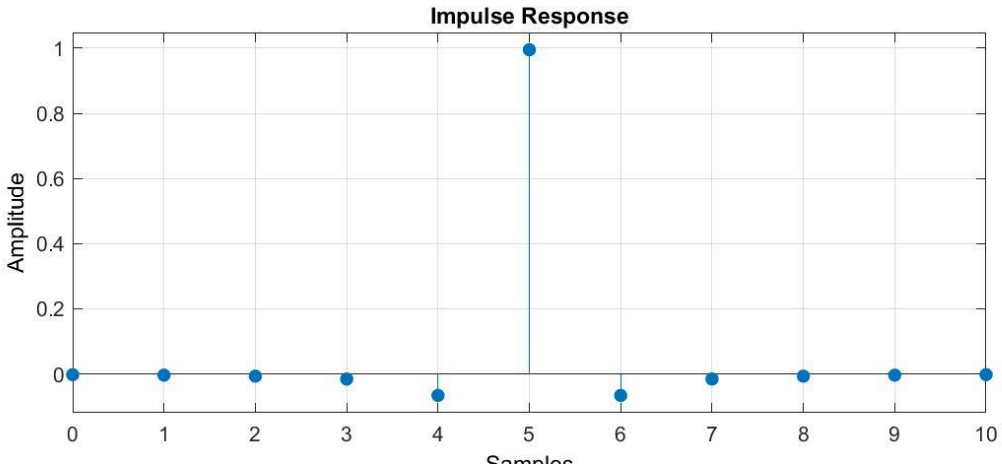


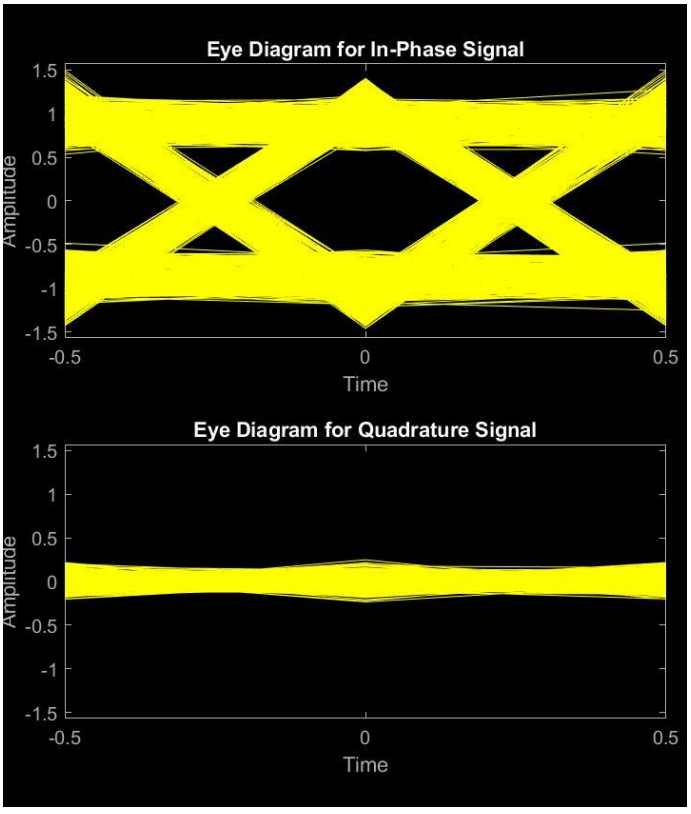


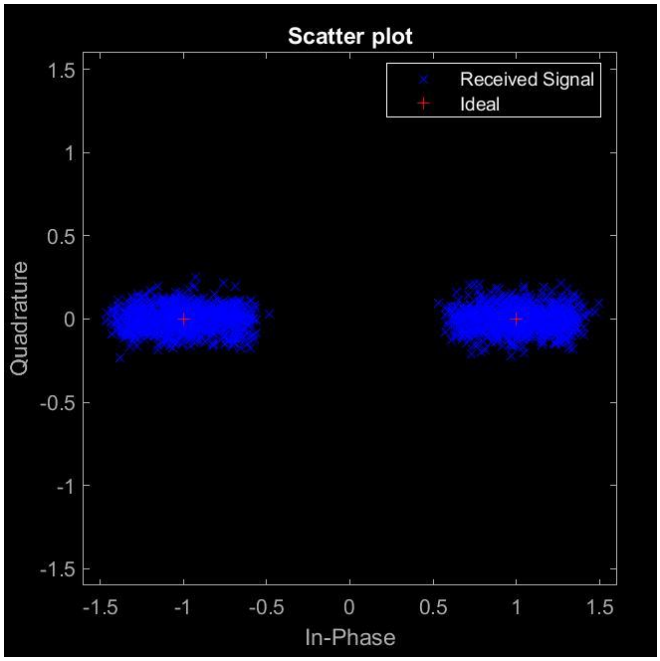


Frame length: 1000 Rate: 449/1024, Span: 2, Roll-off factor: 1

Waveform:

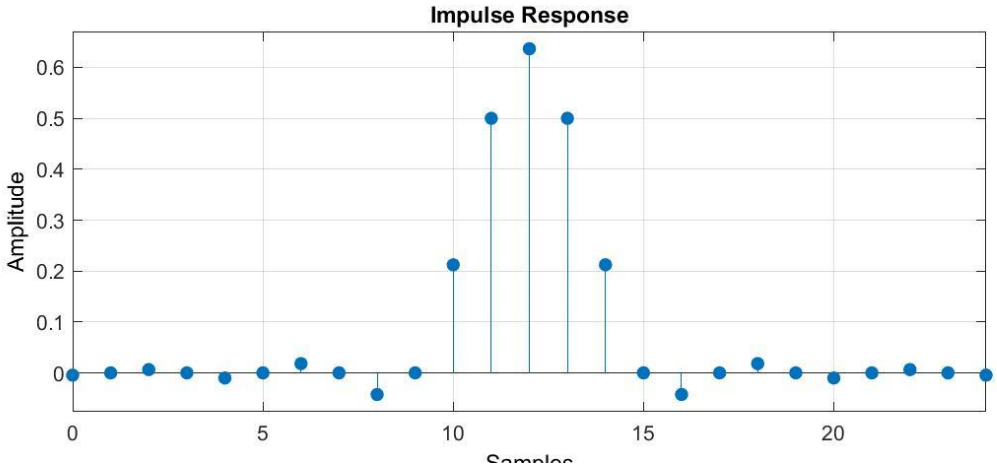


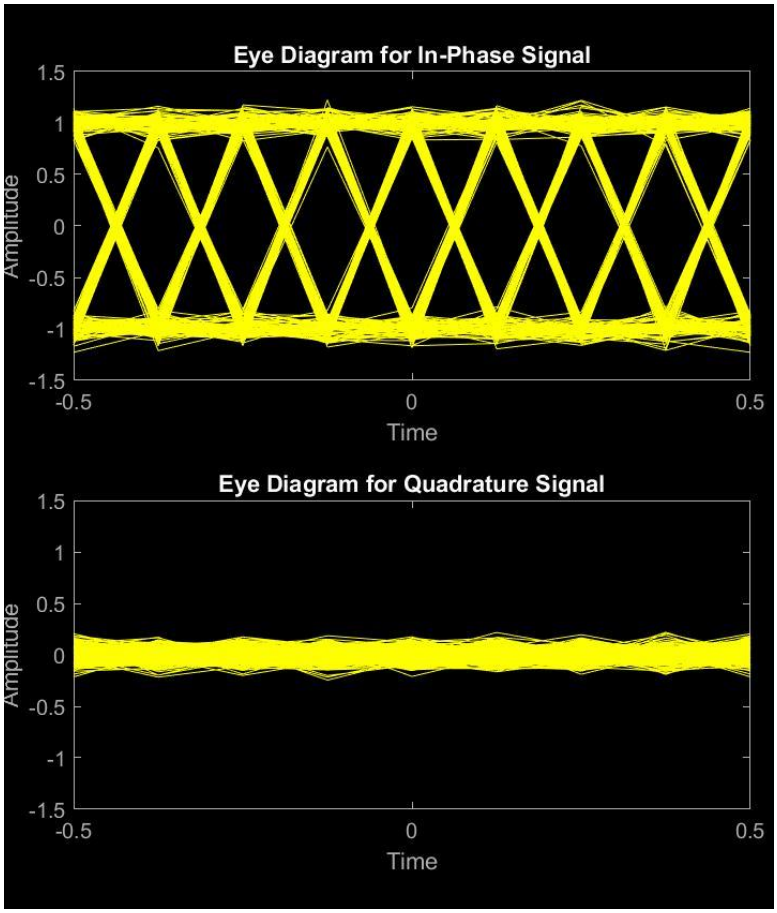


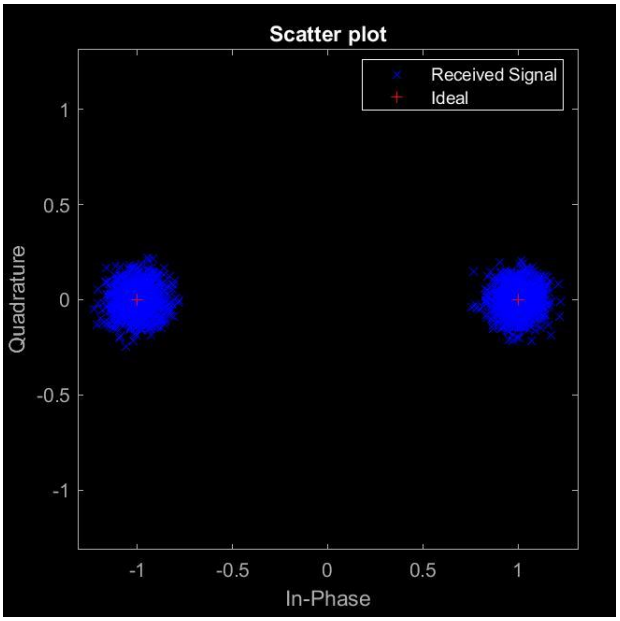


Frame length: 1000 Rate: 449/1024, Span: 6, Roll-off factor: 1

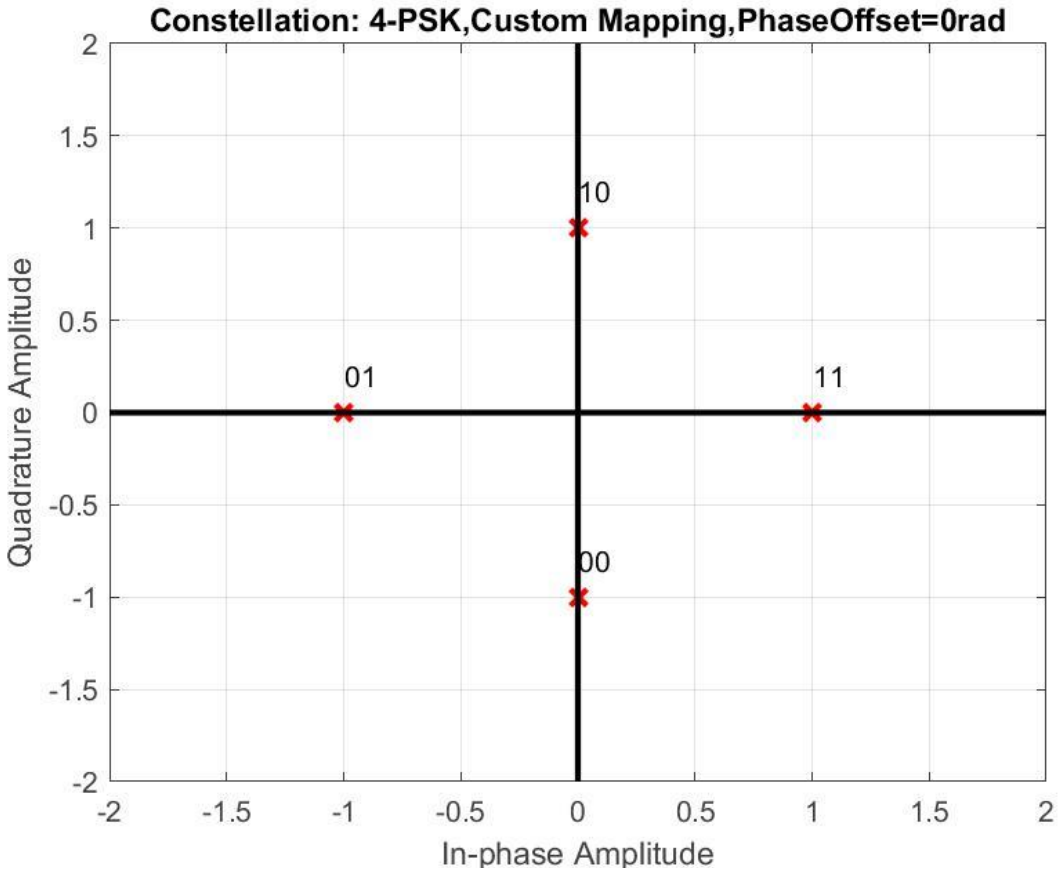
Waveform:

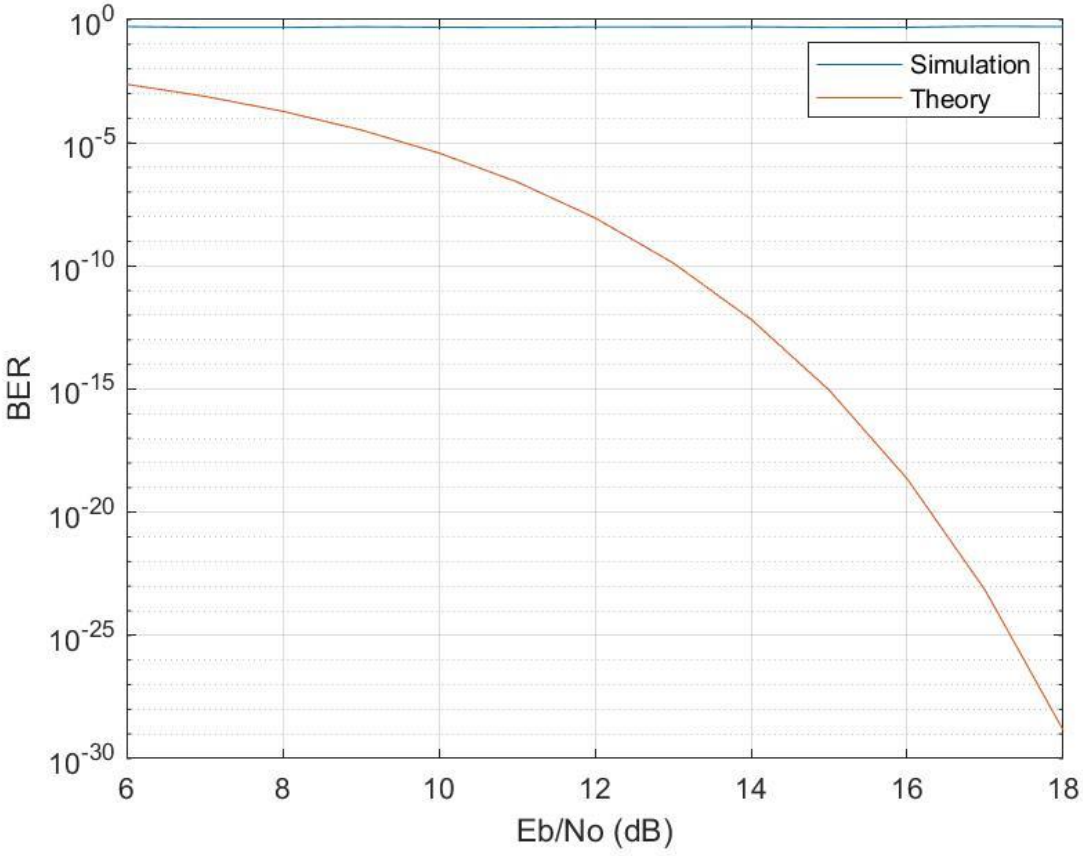






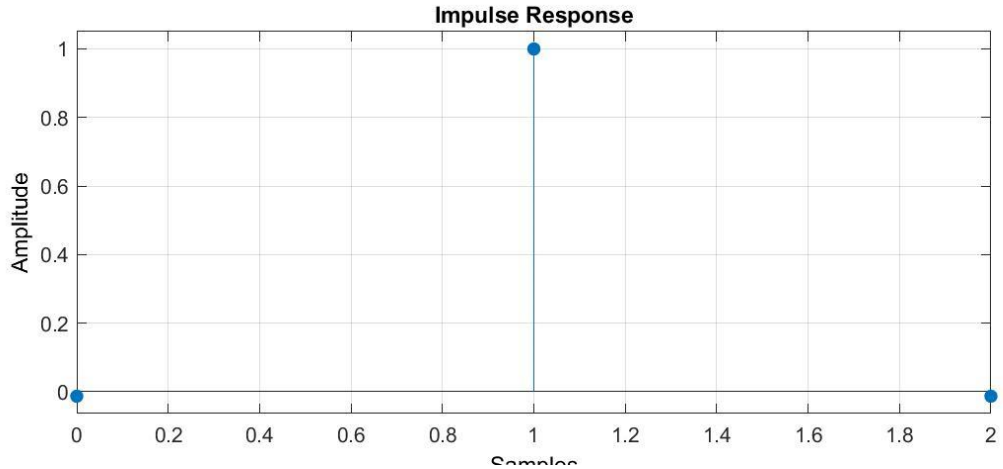
QPSK Simulations for Waveform Design

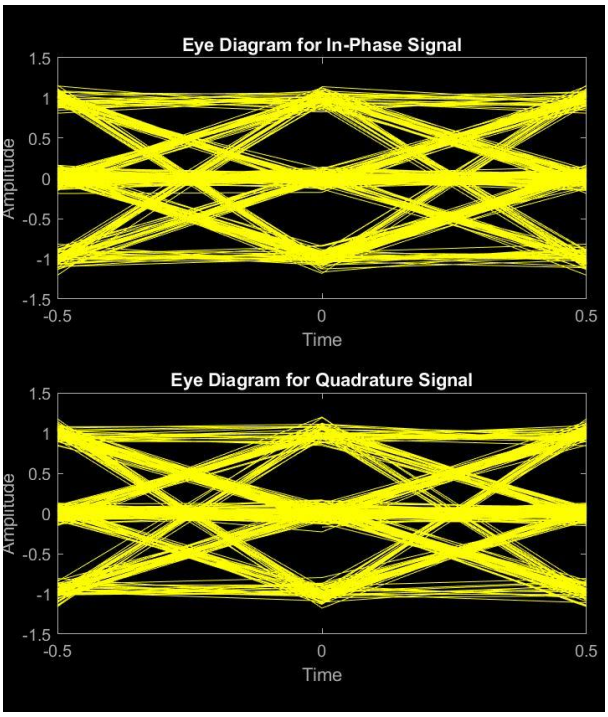


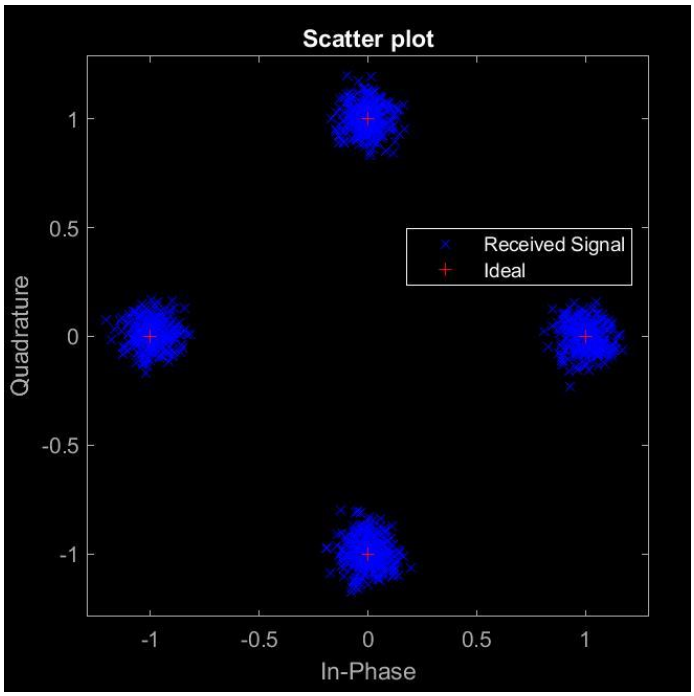


Frame length: 1000 Rate: 449/1024, Span: 2, Roll-off factor: 0.05

Waveform:

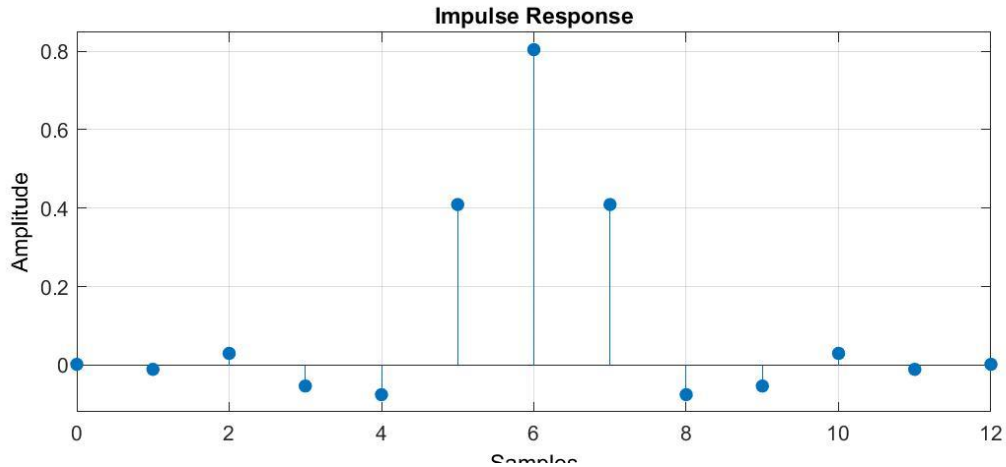


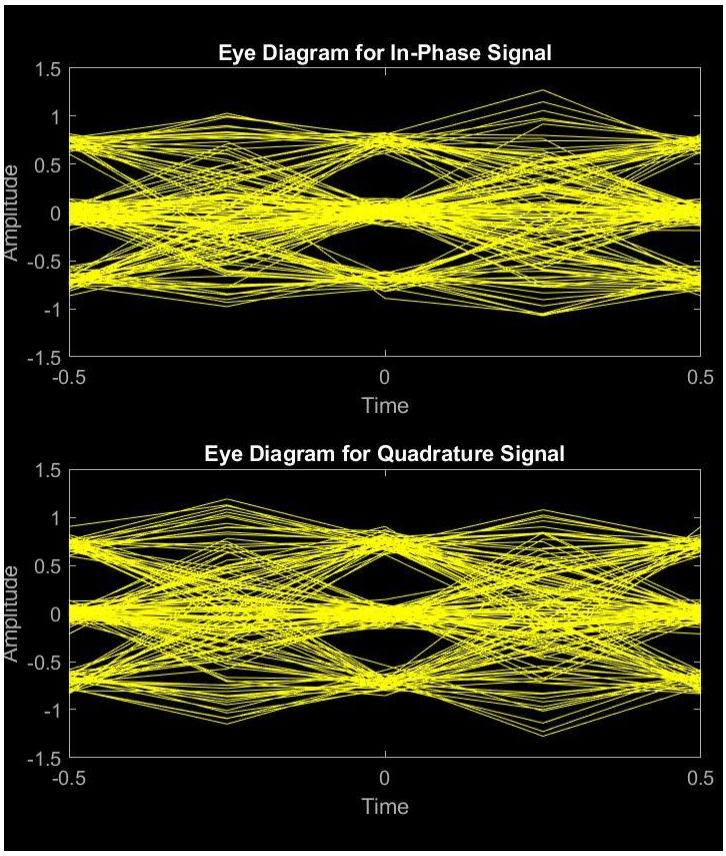


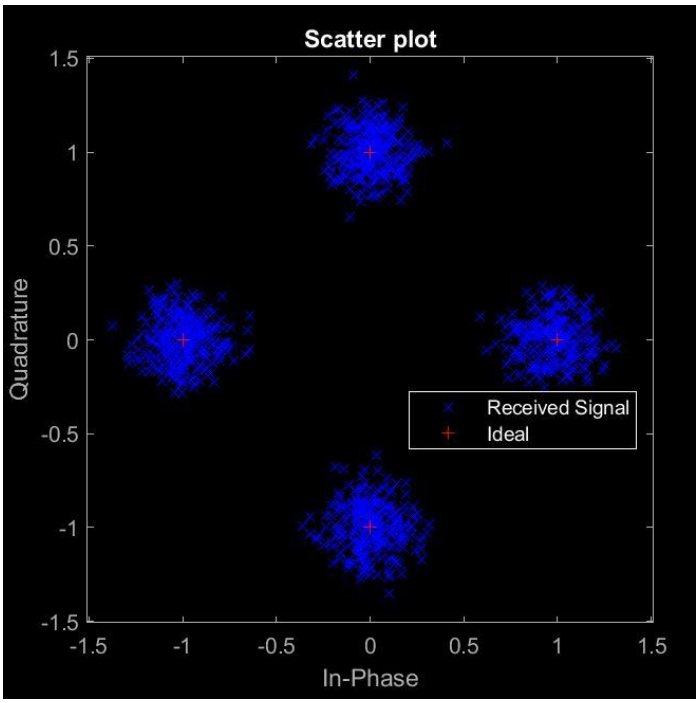


Frame length: 1000 Rate: 449/1024, Span: 6, Roll-off factor: 0.5

Waveform:

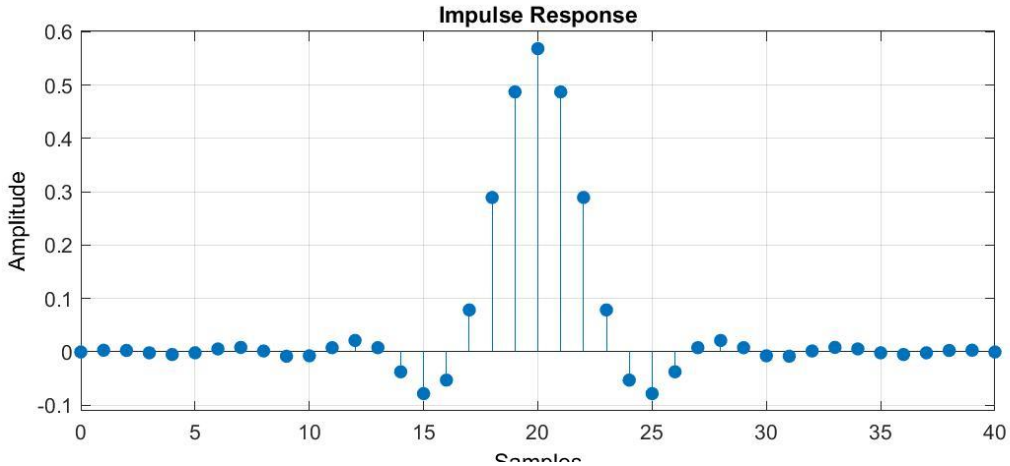


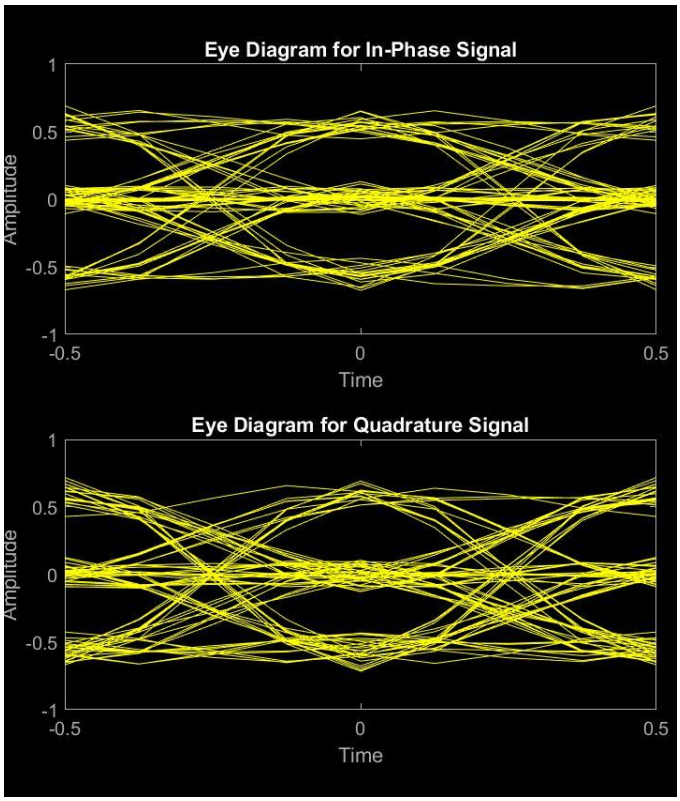


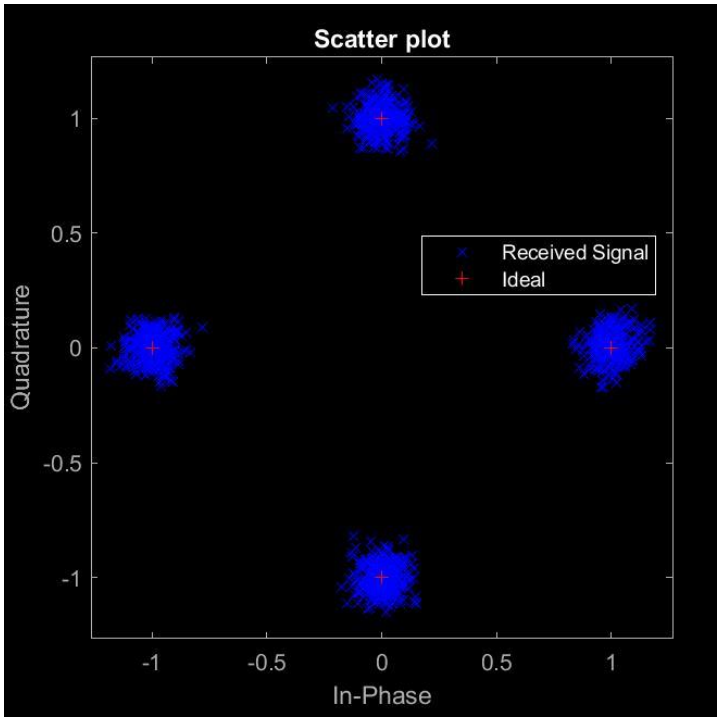


Frame length: 1000 Rate: 449/1024, Span: 10, Roll-off factor: 0.5

Waveform:

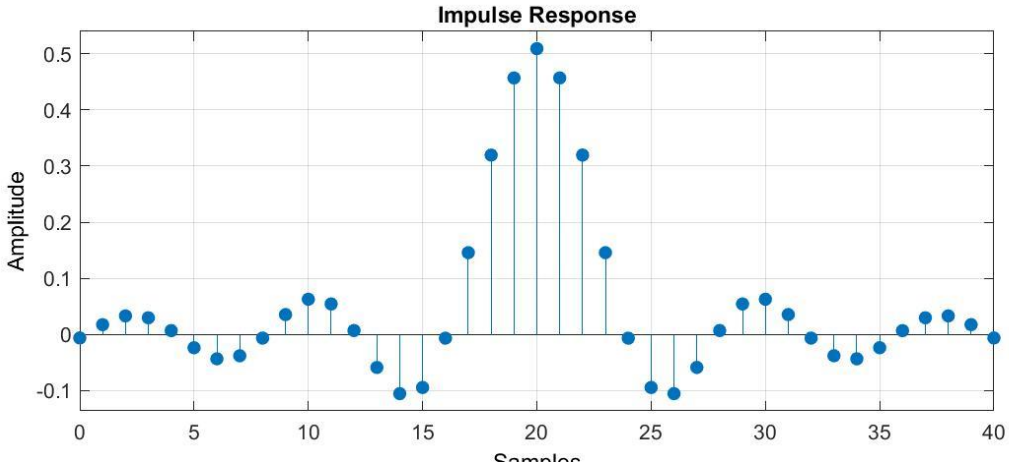


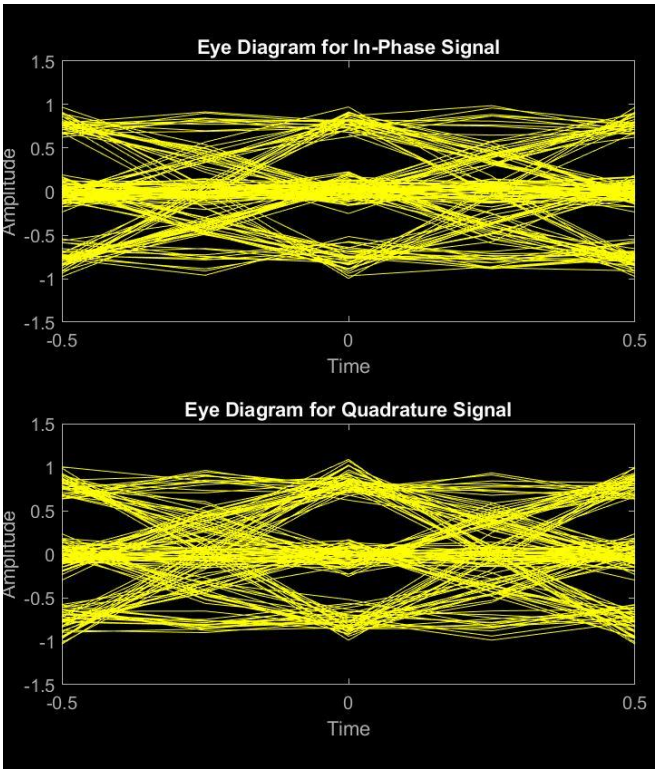


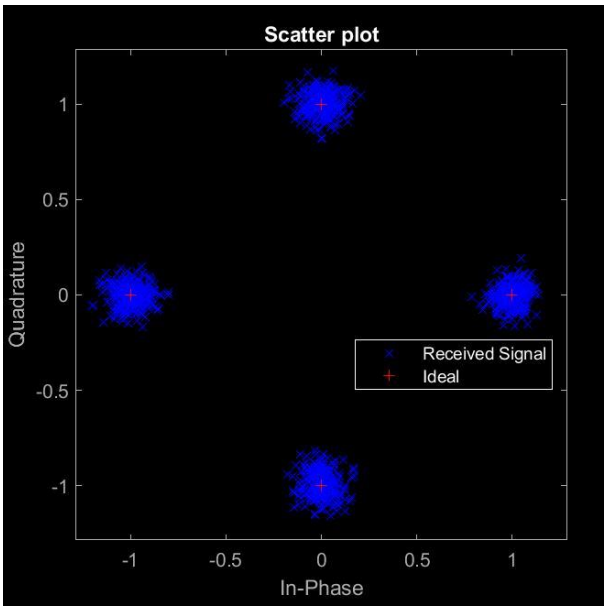


Frame length: 1000 Rate: 449/1024, Span: 10, Roll-off factor: 0.05

Waveform:

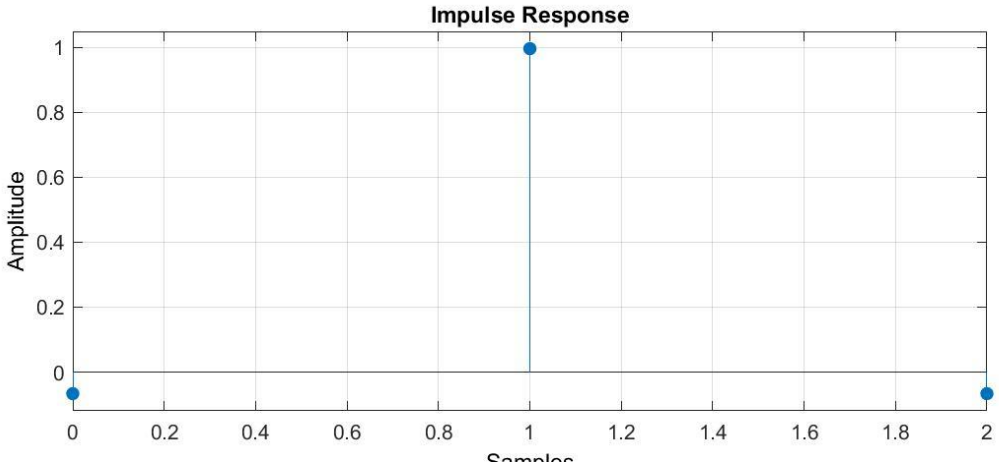


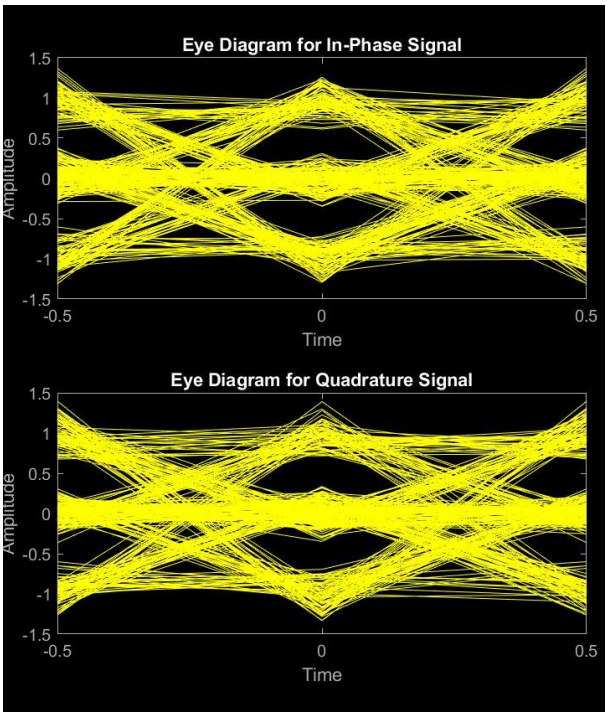


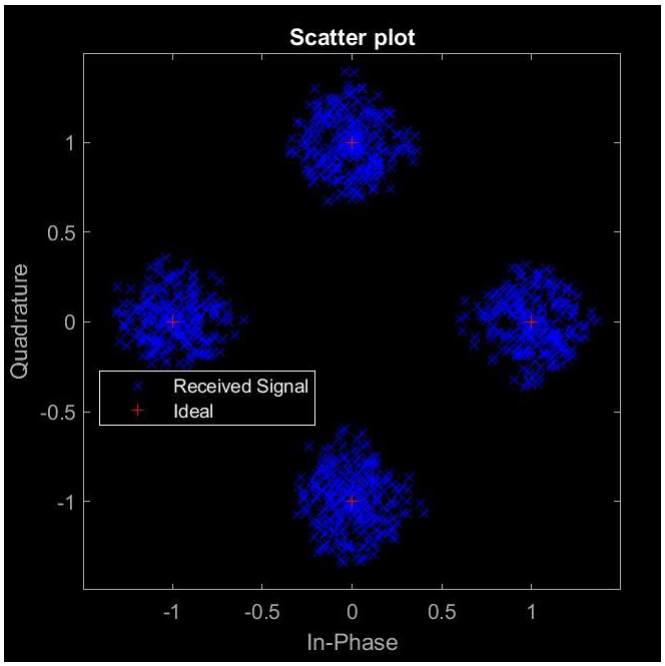


Frame length: 1000 Rate: 449/1024, Span: 2, Roll-off factor: 1

Waveform:



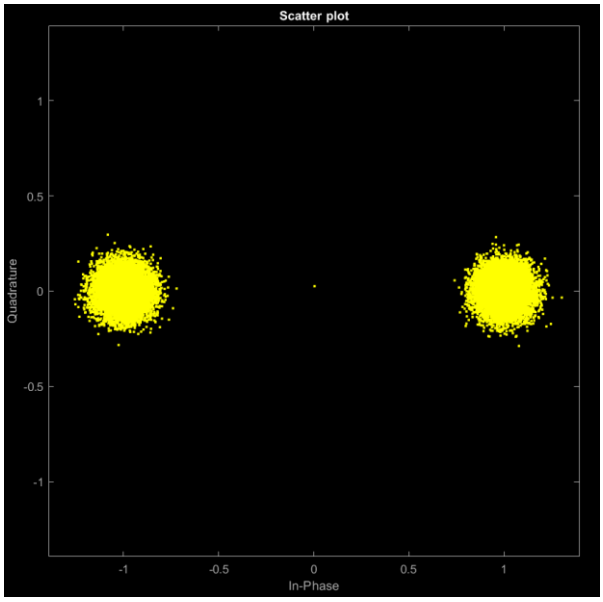


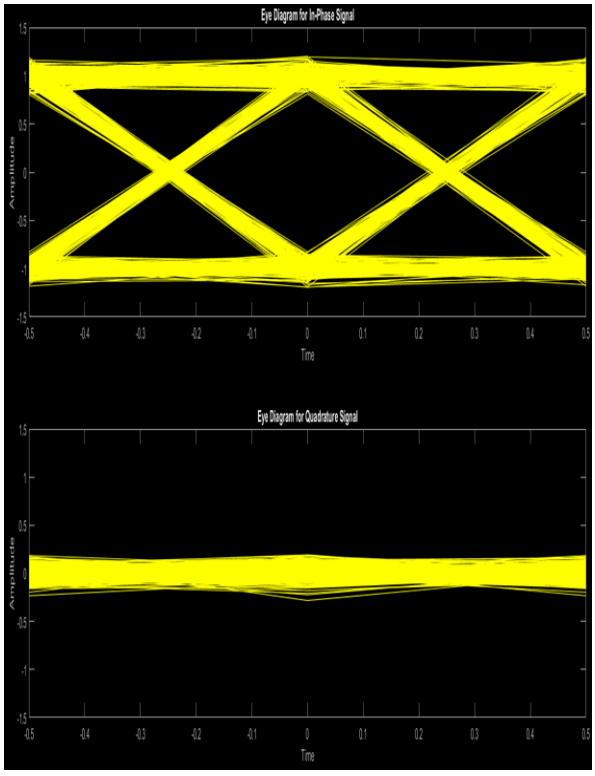


Parameter Comparisons

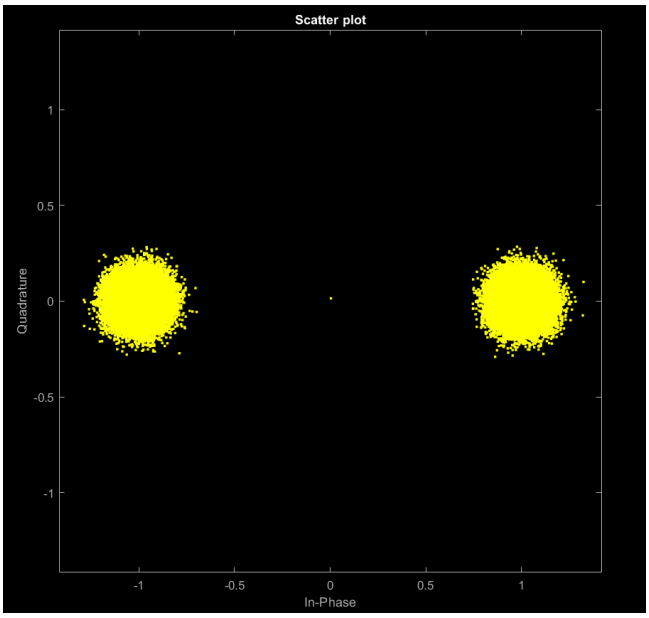
BPSK:

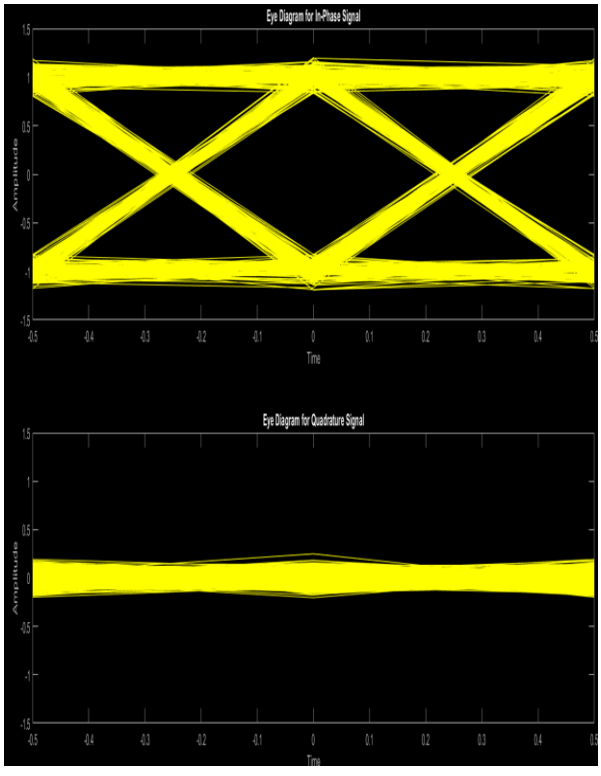
Coding Rate = 449/1024





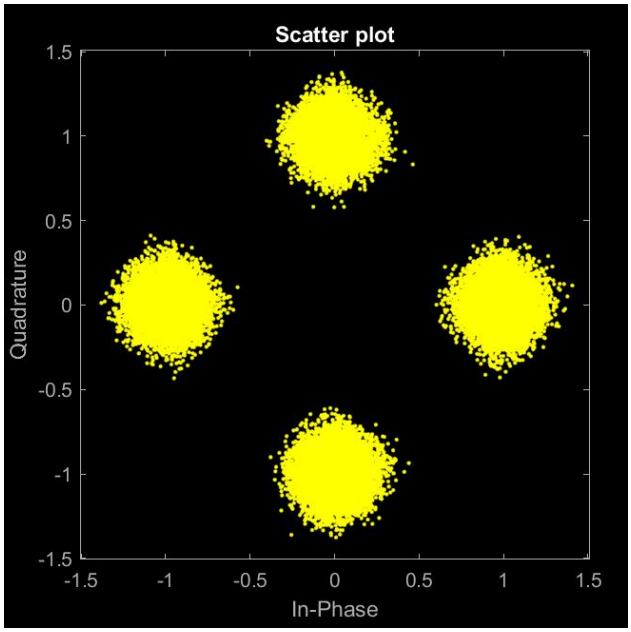
Coding Rate = 120/1024

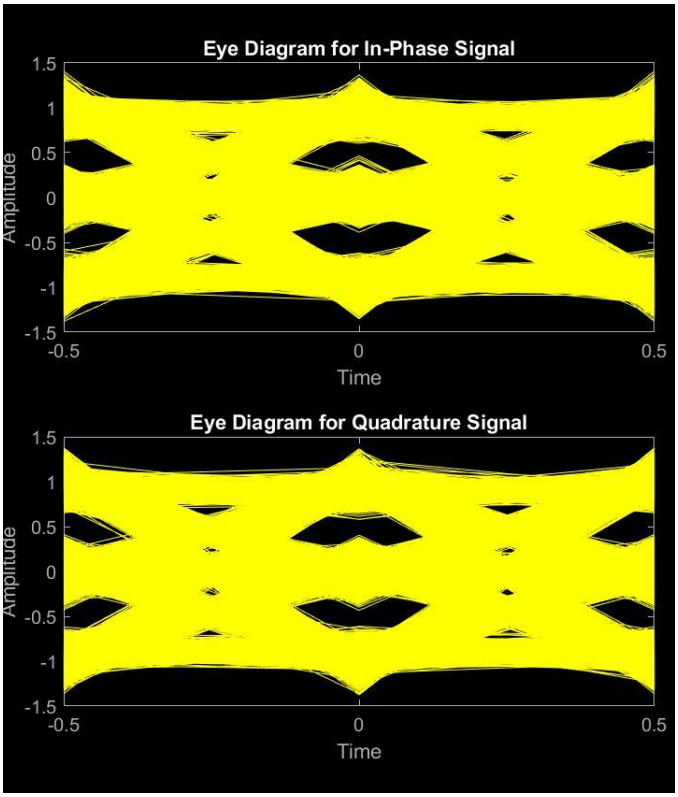




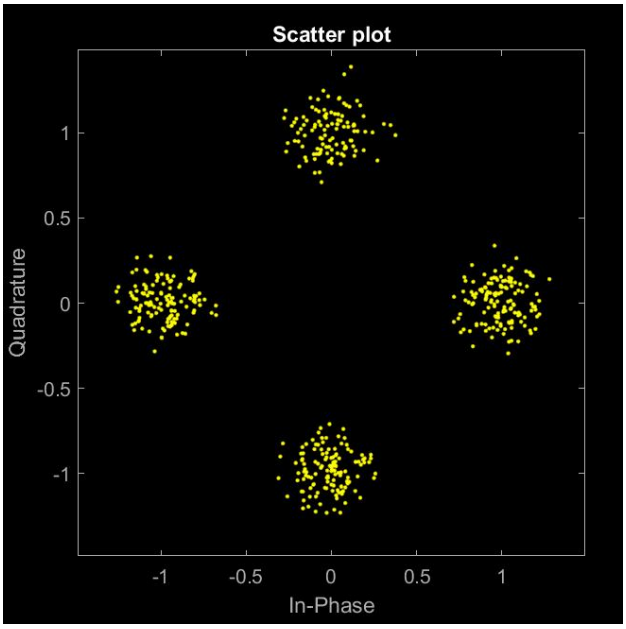
QPSK:

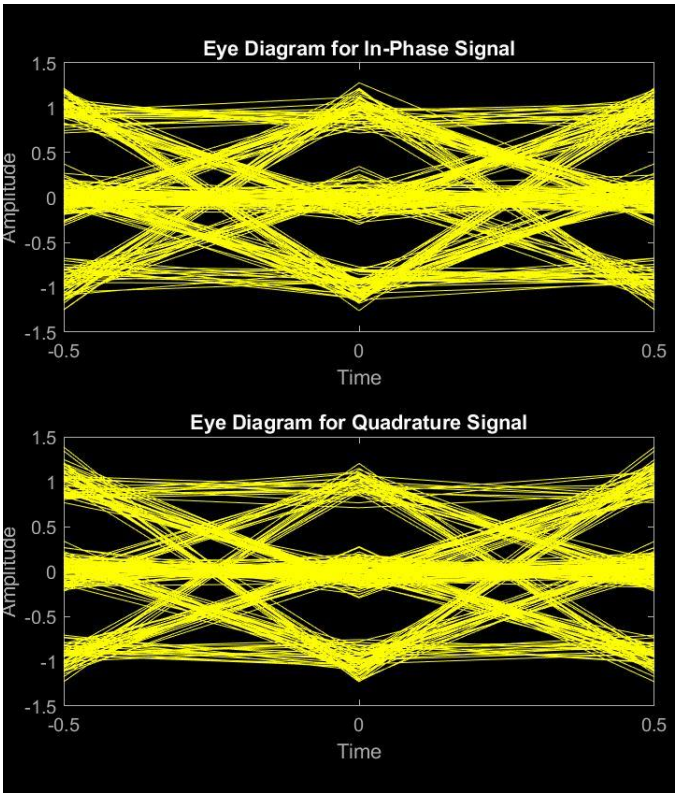
Coding Rate = 16/1024



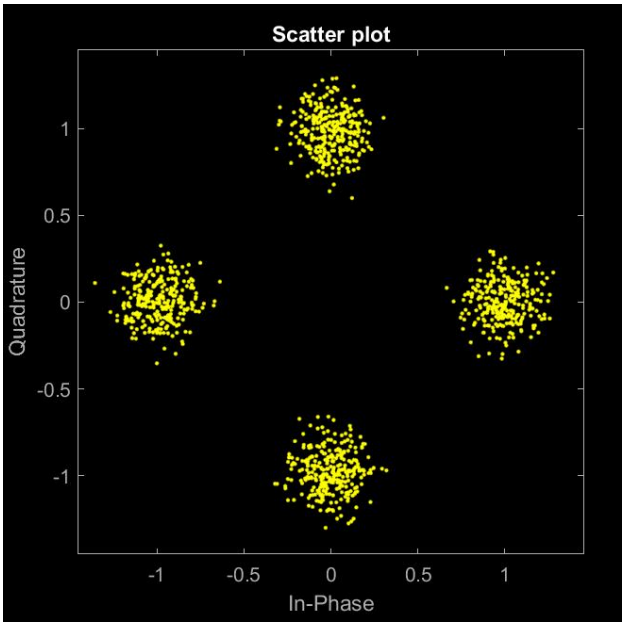


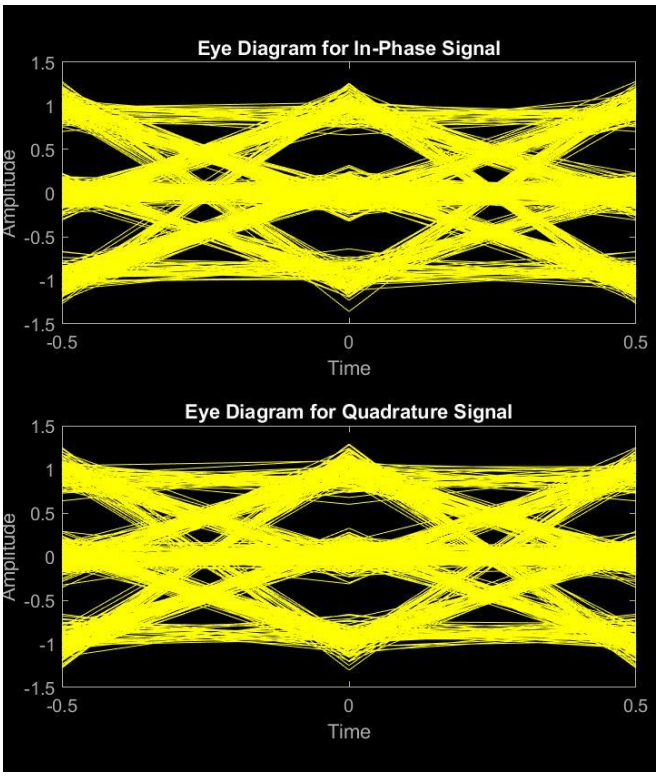
Coding Rate = 1008/1024



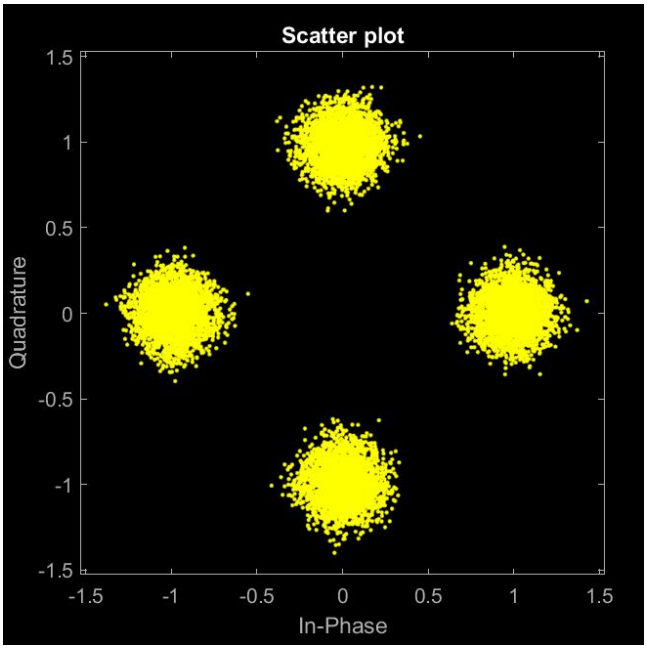


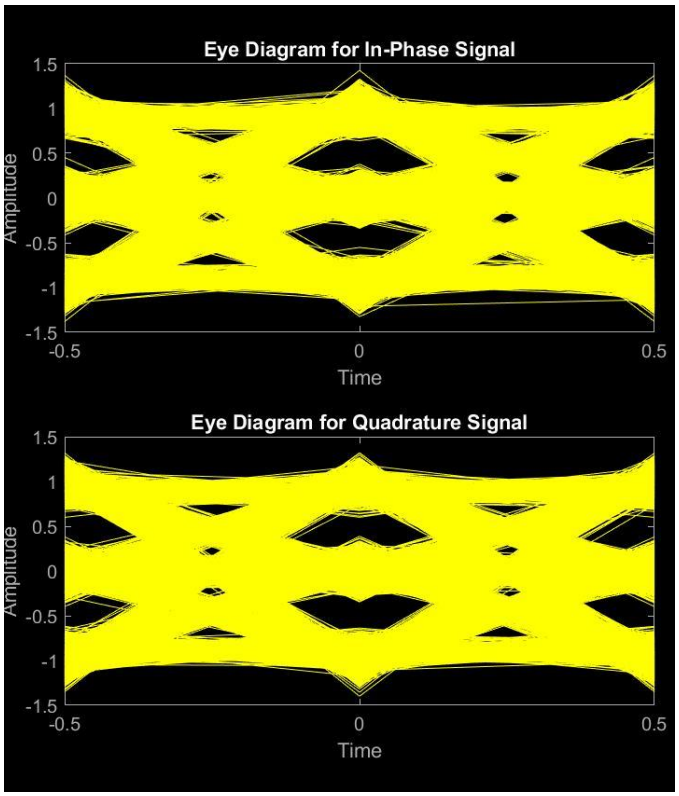
Frame Rate = 1000





Frame Rate = 10000





Discussion of Results

Raised Cosine filter is selected as signal transmission waveform and one transmitter one receiver filter is implemented to the both ends of the AWGN channel. For the parameter tuning, frame length and data rate is kept constant while span and roll-off factor parameters are adjusted according to the obtained simulation results. After the raised cosine specifications are finalized, new simulations are done by changing the frame length and data date parameters. Raised cosine with span 10 and roll-off factor 0.5 gave better results than the other simulations in terms of eye diagram opening and constellation diagram spread for QPSK modulation. For BPSK modulation, span of 2 and roll-off factor of 0.05 gave better results than the others.

Eye diagram is a data dependent electrical measurement used to evaluate high speed data quality. Increase in the SNR ratio leads more open eye shape in the eye diagram, where if the SNR ratio decreases, i.e. noise in the channel increases, eye opening shrinks. In the simulations, it is aimed to obtain wider eye openings.

A constellation diagram is a representation of a signal modulated by a digital modulation scheme such as quadrature amplitude modulation or phase-shift keying. It displays the signal as a two-dimensional plane scatter diagram in the complex plane at symbol sampling instants. It is possible to observe from the simulations that BPSK has two constellation points on the x axis and QPSK has four constellation points two on the x axis and two on the y axis. There is a phase shift of $\pi/4$ in the QPSK modulation, hence constellations do not constitute a perfect squared shape, but a 45 degrees rotated square at the counterclockwise direction.

Raised-cosine filter is an implementation of a low-pass Nyquist filter. A typical use of raised cosine filtering is to split the filtering between transmitter and receiver. Both transmitter and receiver employ square-root raised cosine filters. The combination of transmitter and receiver filters is a raised cosine filter, which results in minimum ISI. We specify a square-root raised cosine filter by setting the shape as 'Square root'. Changing the span of the raised cosine filter lead the impulse response to change. For example, when the span is small raised cosine turned out to be single impulse but when the span is increased, a sinc function is better observed. Roll-off factor is a measure of the excess bandwidth of the filter and it takes values between 0 and 1. Roll-off factor determine the spread of the cosine along the x axis. Smaller roll-off factor leads wider raised cosine, bigger roll-off factor leads shrink raised cosine. An example of raised cosine image can be found in the appendix.

QPSK provide very good noise immunity. It provides low error probability Bandwidth is twice efficient is compared to BPSK modulation. For the same BER, the bandwidth required by QPSK is reduced to half as compared to BPSK. It is more efficient utilization of the available bandwidth of the transmission channel. However, QPSK is not more power efficient modulation technique compare to other modulation types as more power is required to transmit two bits. QPSK is more complex compared to BPSK receiver due to four states needed to recover binary data information.

CONCLUSION

LDPC Coding is the encoding/decoding technique of digital data used for 5G communications. Modulation type effects the quality of signal transmission together with the data rate. Raised cosine can be used as the waveform for signal transmission, however incorrect tuning of its parameters may degrade the quality of the signal transmission and can increase the bit error rate at the receiver. Constellation diagrams visually show the likelihood of received data points. Wide eye diagram is a desired property for communication system, wider the eye, better the communication quality.

Future Work

In order to better understand the modulation effects and differences between different digital encoding techniques, other modulation types such as QAM or FSK simulations can be conducted with other codings such as Turbo Encoding or Polar Encoding.

APPENDIX

a) For BPSK Simulations

```
clear all; close all;clc;

rng(210);              % Set RNG state for repeatability
A = 10000;             % Transport block length, positive integer
rate = 449/1024;       % Target code rate, 0<R<1
rv = 0;                % Redundancy version, 0-3
modulation = 'pi/2-BPSK';   % Modulation scheme, QPSK, 16QAM, 64QAM, 256QAM
nlayers = 1;           % Number of layers, 1-4 for a transport block

M = 2;                  % Modulation alphabet size
bps = log2(M);            % Bits/symbol
custMap = [1 0];
phase_offset = 0;       % Phase offset (radians)
EbNo = 20; % (dB)

span = 2; % Filter span in symbols
rolloff = 0; % Rolloff factor
sps = 1; % Samples per symbol

pskModulator = comm.PSKModulator(M,'BitInput',true,
'SymbolMapping','Custom','CustomSymbolMapping',custMap, 'PhaseOffset',
phase_offset);
pskDemodulator = comm.PSKDemodulator(M,'BitOutput',true, 'DecisionMethod', 'Log-
likelihood ratio','SymbolMapping','Custom','CustomSymbolMapping',custMap);
channel = comm.AWGNChannel('EbNo',EbNo,'BitsPerSymbol',bps);
txfilter = comm.RaisedCosineTransmitFilter('RolloffFactor',rolloff,
'FilterSpanInSymbols',span,'OutputSamplesPerSymbol',sps);
rxfilter = comm.RaisedCosineReceiveFilter('RolloffFactor',rolloff,
'FilterSpanInSymbols',span,'InputSamplesPerSymbol',sps, 'DecimationFactor',sps);


% DL-SCH coding parameters
cbsInfo = nrULSCHInfo(A,rate);
disp('UL-SCH coding parameters')
disp(cbsInfo)

% Random transport block data generation
in = randi([0 1],A,1,'int8');

% Transport block CRC attachment
tbIn = nrCRCEncode(in,cbsInfo.CRC);

% Code block segmentation and CRC attachment
cbsIn = nrCodeBlockSegmentLDPC(tbIn,cbsInfo.BGN);

% LDPC encoding
enc = nrLDPCEncode(cbsIn,cbsInfo.BGN);

% Rate matching and code block concatenation
outlen = ceil(A/rate);
modIn = nrRateMatchLDPC(enc,outlen,rv,modulation,nlayers);

% Modulation
modData = pskModulator(modIn);
```

```
% Waveform Design - Tx
%chIn = txfilter(modData);

% AWGN Channel
channelOutput = channel(modData);


% Waveform Design - Rx
demodData = rxfilter(channelOutput);

% Demodulation
demodOut = pskDemodulator(channelOutput);


% Rate recovery
raterec = nrRateRecoverLDPC(demodOut,A,rate,rv,modulation,nlayers);

% LDPC decoding
decBits = nrLDPCDecode(raterec,cbsInfo.BGN,25);

% Code block desegmentation and CRC decoding
[blk,blkErr] = nrCodeBlockDesegmentLDPC(decBits,cbsInfo.BGN,A+cbsInfo.L);

disp(['CRC error per code-block: [' num2str(blkErr) ']'])

% Transport block CRC decoding
[out,tbErr] = nrCRCDecode(blk,cbsInfo.CRC);
out = [out(3:end); 0; 1];
disp(['Transport block CRC error: ' num2str(tbErr)])
disp(['Recovered transport block with no error: ' num2str(isequal(out,in))])


% PLOTS
n = 1;                  % Plot every nth value of the signal
offset = 0;             % Plot every nth value of the signal, starting from
offset+1

h = scatterplot(demodData(span+1:end-span),n,offset,'bx'); hold on
scatterplot(modData,n,offset,'r+',h)
legend('Received Signal','Ideal','location','best')

constellation(pskModulator)

eyediagram(chIn(sps*span+1:sps*span+1000),2*sps)
eyediagram(demodData(sps*span+1:sps*span+1000),2*sps)
scatterplot(modData)
title('Modulated Data');
scatterplot(channelOutput)
fvtool(txfilter,'impulse')
% channel.EbNo = 10;
% channelOutput = channel(modData);
% scatterplot(channelOutput)
```

b) For QPSK Simulations and saving figures more efficiently

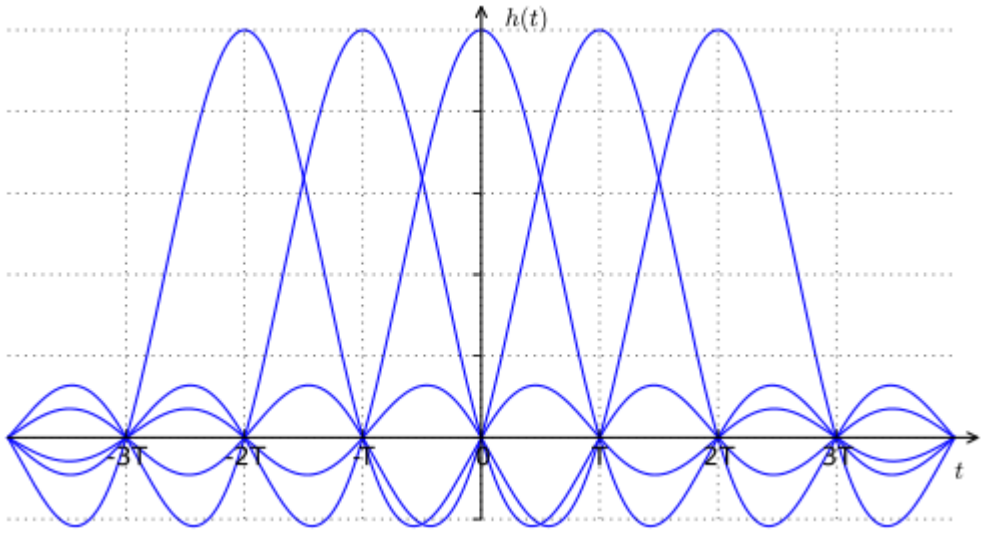
```
for A = [1000] % [1000 10000]
    for rate = [16/1024 1008/1024] % [449/1024 800/1024]
        for span = [10] % [2 6 10]
            for rolloff = [0.5] %[1 0.5 0.05]
                for sps = [1] % [1 2 4]
                    disp('Loop Parameters:')
                    disp(sps)
                    disp(rolloff)
                    disp(span)
                    disp(rate)
                    disp(A)


                    close all

                    rng(210);               % Set RNG state for repeatability
                    % A = 10000;              % Transport block length, positive
integer
                    % rate = 449/1024;        % Target code rate, 0<R<1 449
                    rv = 0;                 % Redundancy version, 0-3
                    modulation = 'QPSK';    % Modulation scheme, QPSK, 16QAM,
64QAM, 256QAM
                    nlayers = 1;            % Number of layers, 1-4 for a transport
block

                    M = 4;                   % Modulation alphabet size
                    bps = log2(M);             % Bits/symbol
                    custMap = [3 2 1 0];
                    phase_offset = 0;        % Phase offset (radians)
                    EbNo = 20; % (dB)

                    %span = 10; % Filter span in symbols
                    %rolloff = 0.8; % Rolloff factor
                    %sps = 4; % Samples per symbol

                    fname =
strcat('frame/A',num2str(A,'%d'),'Rate',num2str(rate,'%f'),'Span',num2str(span,'%d
'),'Mod',modulation,'Rff',num2str(rolloff,'%f'),'SPS',num2str(sps,'%d'));


                    pskModulator = comm.PSKModulator(M,'BitInput',true,
'SymbolMapping','Custom','CustomSymbolMapping',custMap, 'PhaseOffset',
phase_offset);
                    pskDemodulator = comm.PSKDemodulator(M,'BitOutput',true,
'DecisionMethod', 'Log-likelihood
ratio','SymbolMapping','Custom','CustomSymbolMapping',custMap);
                    channel = comm.AWGNChannel('EbNo',EbNo,'BitsPerSymbol',bps);
                    txfilter =
comm.RaisedCosineTransmitFilter('RolloffFactor',rolloff,
'FilterSpanInSymbols',span,'OutputSamplesPerSymbol',sps);
                    rxfilter =
comm.RaisedCosineReceiveFilter('RolloffFactor',rolloff,
'FilterSpanInSymbols',span,'InputSamplesPerSymbol',sps, 'DecimationFactor',sps);

                    errorRate = comm.ErrorRate;
                    ebnoVec = 6:18;
                    ber = zeros(size(ebnoVec));
```

```
                    % DL-SCH coding parameters
                    cbsInfo = nrULSCHInfo(A,rate);
                    disp('UL-SCH coding parameters')
                    disp(cbsInfo)

                    for n = 1:length(ebnoVec)
                        reset(errorRate)
                        errVec = [0 0 0];
                        channel.EbNo = ebnoVec(n);

                        while errVec(2) < 200 && errVec(3) < 1e7
                            % Random transport block data generation
                            in = randi([0 1],A,1,'int8');

                            % Transport block CRC attachment
                            tbIn = nrCRCEncode(in,cbsInfo.CRC);

                            % Code block segmentation and CRC attachment
                            cbsIn = nrCodeBlockSegmentLDPC(tbIn,cbsInfo.BGN);

                            % LDPC encoding
                            enc = nrLDPCEncode(cbsIn,cbsInfo.BGN);

                            % Rate matching and code block concatenation
                            outlen = ceil(A/rate);
                            modIn =
nrRateMatchLDPC(enc,outlen,rv,modulation,nlayers);

                            % Modulation
                            modData = pskModulator(modIn);

                            % Waveform Design - Tx
                            chIn = txfilter(modData);

                            % AWGN Channel
                            channelOutput = channel(chIn);


                            % Waveform Design - Rx
                            demodData = rxfilter(channelOutput);

                            % Demodulation
                            demodOut = pskDemodulator(demodData);


                            % Rate recovery
                            raterec =
nrRateRecoverLDPC(demodOut,A,rate,rv,modulation,nlayers);

                            % LDPC decoding
                            decBits = nrLDPCDecode(raterec,cbsInfo.BGN,25);

                            % Code block desegmentation and CRC decoding
                            [blk,blkErr] =
nrCodeBlockDesegmentLDPC(decBits,cbsInfo.BGN,A+cbsInfo.L);
```

```
                            disp(['CRC error per code-block: [' num2str(blkErr)
']'])

                            % Transport block CRC decoding
                            [out,tbErr] = nrCRCDecode(blk,cbsInfo.CRC);
                            %out = [out(3:end); 0; 1];
                            disp(['Transport block CRC error: ' num2str(tbErr)])
                            disp(['Recovered transport block with no error: '
num2str(isequal(out,in))])

                            % Collect the error statistics
                            errVec = errorRate(in,out);
                        end

                        % Save the BER data
                        ber(n) = errVec(1);
                    end

                    berTheory = berawgn(ebnoVec,'psk',M,'nondiff');
                    figure
                    semilogy(ebnoVec,[ber; berTheory])
                    xlabel('Eb/No (dB)')
                    ylabel('BER')
                    grid
                    legend('Simulation','Theory','location','ne')


                    % PLOTS
                    n = 1;                  % Plot every nth value of the signal
                    offset = 0;             % Plot every nth value of the signal,
starting from offset+1

                    h = scatterplot(demodData(span+1:end-span),n,offset,'bx');
hold on
                    scatterplot(modData,n,offset,'r+',h)
                    legend('Received Signal','Ideal','location','best')

                    constellation(pskModulator)
                    finito = (sps*span+1) + 1000;

                    eyediagram(chIn(sps*span+1:length(chIn)),2*sps)

                    eyediagram(demodData(sps*span+1:length(demodData)),2*sps)


eyediagram(channelOutput(sps*span+1:length(channelOutput)),2*sps)

                    scatterplot(modData)
                    title('Modulated Data');
                    scatterplot(channelOutput)
                    fvtool(txfilter,'impulse');
                    % channel.EbNo = 10;
                    % channelOutput = channel(modData);
                    % scatterplot(channelOutput)

                    mkdir("./" + fname);
                    saveas(figure(1),[pwd strcat('/',fname,'/1.jpg')]);  % here
you save the figure
```

```
                    saveas(figure(2),[pwd strcat('/',fname,'/2.jpg')]);  % here
you save the figure
                    saveas(figure(3),[pwd strcat('/',fname,'/3.jpg')]);  % here
you save the figure
                    saveas(figure(4),[pwd strcat('/',fname,'/4.jpg')]);  % here
you save the figure
                    saveas(figure(5),[pwd strcat('/',fname,'/5.jpg')]);  % here
you save the figure
                    saveas(figure(6),[pwd strcat('/',fname,'/6.jpg')]);  % here
you save the figure
                    saveas(figure(7),[pwd strcat('/',fname,'/7.jpg')]);  % here
you save the figure
                    saveas(figure(8),[pwd strcat('/',fname,'/8.jpg')]);  % here
you save the figure
                    saveas(figure(9),[pwd strcat('/',fname,'/9.jpg')]);  % here
you save the figure
                    saveas(figure(10),[pwd strcat('/',fname,'/10.jpg')]);  % here
you save the figure

                end
            end
        end
    end
end
```

c) Raised Cosine Waveform:



REFERENCES

"NR; Multiplexing and channel coding." 3rd Generation Partnership Project; Technical Specification Group Radio Access Network.

Bae, Jung & Abotabl, Ahmed & Lin, Hsien-Ping & Song, Kee-Bong & Lee, Jungwon. (2019). An overview of channel coding for 5G NR cellular communications. APSIPA Transactions on Signal and Information Processing. 8. 10.1017/ATSIP.2019.10.

Introducing Low-Density Parity-Check Codes Sarah J. Johnson School of Electrical Engineering and Computer Science The University of Newcastle Australia

Kenneth Andrews, Chris Heegard, A Theory of Interleavers, Cornell, 1996

Tilo Strutz, Low-Density Parity-Check codes An Intruduction, 2016, James Cook University

LIFANG WANG, Implementation of Low-Density Parity-Check codes for 5G NR shared channels, KTH ROYAL INSTITUTE OF TECHNOLOGY, 2021