Lab 2 - Sampling, Filtering, and Data/Noise Generation

General Instructions

1. Course Notes (Papen/Blahut) are uploaded in Piazza: Papen_Blahut_comms_notes.pdf. Suggested reading has been highlighted with some questions.
2. Download and read Sections 1 and 2 of the paper on pseudo-random binary sequences (PRBS.pdf) which is under the “Resources" tab on Piazza. We will be using these sequences throughout the class and understanding the correlation properties and power spectrum of these sequences is essential. These properties are addressed in a Post Lab simulation.
3. Some pre-written matlab functions have been provided in Piazza (pngen.m, c_corr.m)

Goal

Quantifying the effect of sampling, pulse shaping, and filtering of random signals such as noise and binary random signals. Study pseudo-random signals, through simulation.

1 Generation of Noise Waveforms

For this part of the labs we will generate white Gaussian noise and examine the temporal, spectral and statistical characteristics of noise-like signals. A general goal of modern communications is to make the transmitted waveforms as “noise-like” as possible as it can be shown that these types of waveforms can convey the maximum amount of information in finite bandwidth.

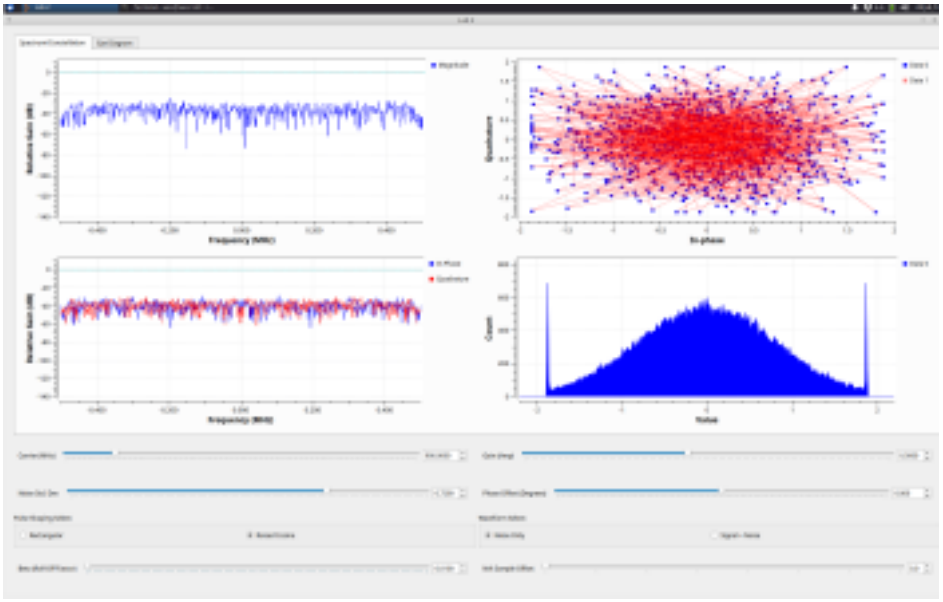
Synthetic noise waveforms are generated by first generating an array of normally distributed random numbers with an rms voltage value σ. Ideally, the (baseband) noise waveform generated from this array has a power density spectrum of $\sigma^2/(f_s/2)$ (W/Hz) where f_s is the sampling rate in Hz. The noise power over a (baseband) noise bandwidth B_N is then

$$P = \frac{2B_N}{f_s}\sigma^2$$

where the baseband noise bandwidth is defined in Eq. (2.3.14) in Papen/Blahut. For this class, the baseband noise bandwidth is set by the internal filtering in the Pluto and the sampling rate so $B_N = f_s/2$ and $N_0 = \sigma^2$ as one might expect. The complex-baseband noise process is twice this so that $N_0 = 2\sigma^2$.

1. Clone the latest version of the labs on your system. (See Lab 1 for details.) Don't forget to to set up the code required to run the labs using ./setup.sh in the course directory.

2. Open course/grc/lab2/lab2.py . You should see some similar to the following panel

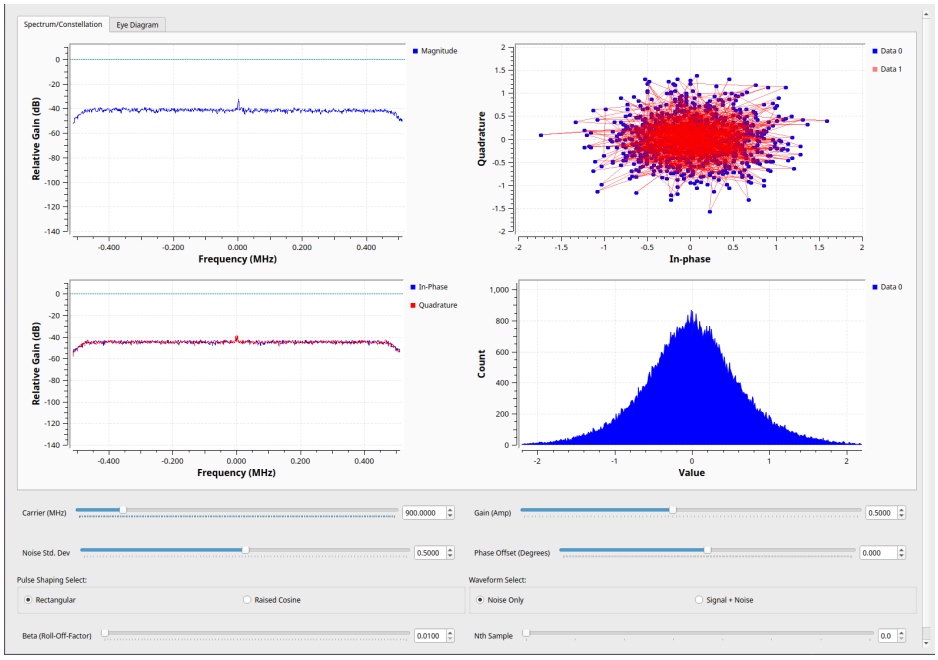


3. Select "Noise only" for the waveform and a carrier frequency of 900 MHz, which should be the default value.

4. Set the noise standard deviation σ = 0.5. (Note that is for both the *I* and *Q* components.)

   On the front panel, the left side of the panel shows the power density spectra in the same format as the spectrum analyzer used in Lab 1 where the upper trace is the magnitude of the complex-baseband noise process $n(t)$ and the lower two traces are the power spectra for $n_I(t)$ and $n_Q(t)$. (See Section 2.3.3 in Papen). The upper right trace is a live display of the *I*/*Q* samples of the noise and should resemble a circle if the window is properly scaled. The lower right trace is the histogram of the in-phase component of the noise and should resemble a gaussian. The peaks on either side are an artifact of the histogram binning when the gain is too large. If you see them, turn the "Gain (Amp)" setting down until you don't see them. Also, if you see a spike at a value of zero in the histogram, it means that there are "dropouts" in the data samples. Try and adjust the VM and python settings to prevent these from occurring as it will affect the results from Part 3.

5. Save the screen trace. Is the center value of the histogram *I*, which is the sample mean *E*(*X*), consistent with the true mean?



The true mean of white Gaussian noise is 0, as you can see on our histogram, the sample mean is also about 0 so this is consistent with our expectations.

6. Estimate the standard deviation σ from the histogram for the **in-phase** component. Is it consistent with the value of σ set on the front panel?

The standard deviation is about 0.5~0.6 compared to the noise standard deviation that we set to 0.5.

Write-Up for Part 1

The write-up for this part consists of the screen trace for the noise only along with the answers to the questions posed during the steps. Additional questions for this part of lab:

1. Using the measured value for the PSD noise power and **assuming** that the noise power $N_0 = \sigma^2$ as discussed above, determine a factor K so that

$$N_0 = KN_{meas}$$

where N_{meas} is the measured power in dB relative to full scale shown on the front panel, K is the scaling factor, and $N_0 = \sigma^2$. (Note that the scaling factor K is **not** a constant as it depends on the automatic gain control (AGC) setting and will vary from Pluto to Pluto. Therefore it can vary depending on the incident signal and the front panel settings.

$N_0 = KN_{meas}$

$N_0 = \sigma^2$

$\sigma^2 = KN_{linear}$

$N_{meas = 40\ DB} \Rightarrow N_{linear} = 10^{\frac{40}{10}} = 10{,}000$

$\sigma^2 = (0.5)^2 = 0.25$

$K = \frac{0.25}{10{,}000} = 2.5 \cdot 10^{-5}$

2. Compare the total noise power for the complex-baseband noise process $n(t) = n_I(t) + jn_Q(t)$ with the noise power in each component (either $n_I(t)$ or $n_Q(t)$. Specifically

   a) What is the relationship between σ^2 and N_0 for a baseband noise signal (either $n_I(t)$ or $n_Q(t)$)?

   For baseband signals σ^2 is the variance of the noise signal and N_0 is the total noise power then, the total noise power is equal to the variance of the noise so for baseband signals: $N_0 = \sigma^2$

   b) What is the relationship between σ^2 and N_0 for a complex baseband noise signal $n(t)$?

   N_0 is just the sum of the powers of the in phase and quadrature components represented by this equation: $N_0 = \sigma_I^2 + \sigma_Q^2$. If both have the same variance then we can rewrite the equation to be $N_0 = 2\sigma^2$.

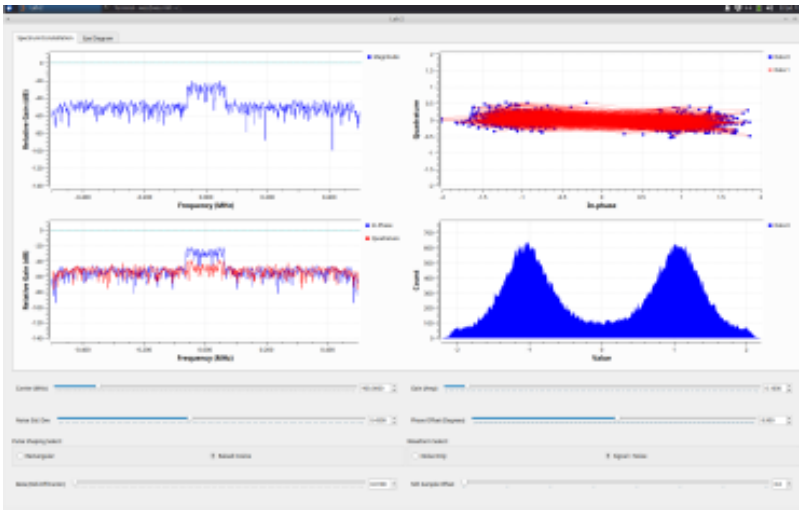
   c) What is the relationship between σ^2 and N_0 for a real-passband noise signal $n(t)$?

   For a real-passband signal N_0 will depend on the Bandwidth,B, so we can write an equation to be $N_0 = B \times \sigma^2$

2 Tx Pulse Shaping

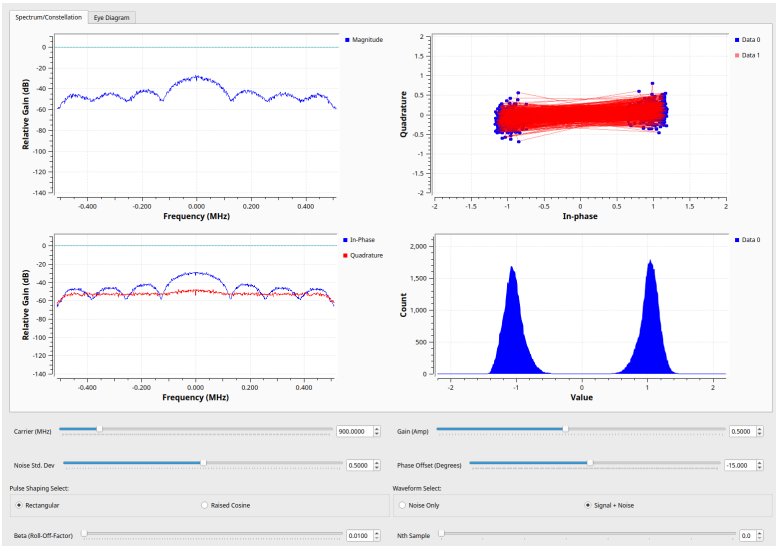
In this part of the lab we will examine a random data waveform in gaussian noise. 1. Select

“Signal + Noise ” for the waveform. The front panel should resemble the figure below



2. For this part of the lab, we will capture four screenshots for four different Tx pulse shapes and compare the power density spectra and the live *I*/*Q* samples of the signal constellation.

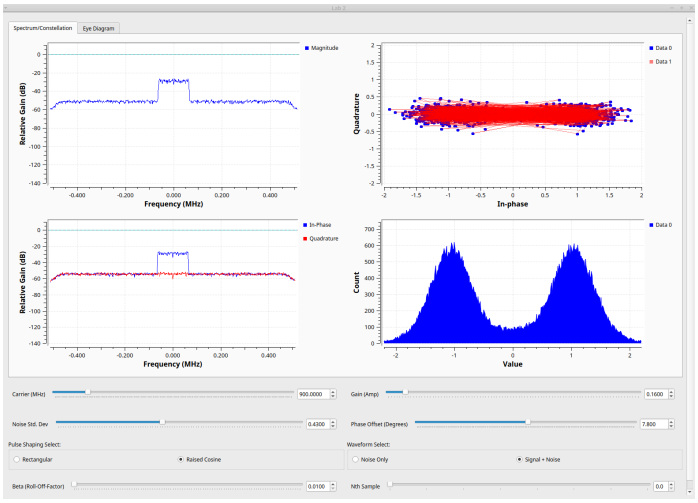
a) First, we will use a simple rectangular pulse by selecting rect under “Pulse Shaping Select”. Adjust the phase offset so that the sampled signal constellation lies along the *I* axis as is shown in the above figure.



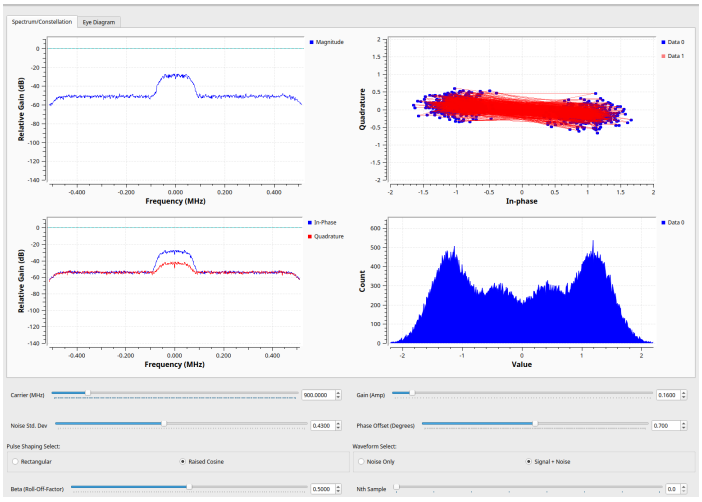
b) Now select raised cosine under “Pulse Shaping Select”. Adjust the phase offset so that the sampled signal constellation lies along the *I* axis as in part (a). Capture screen traces for the following values of β being sure to adjust the phase so that the

constellation is oriented along the *I* axis.

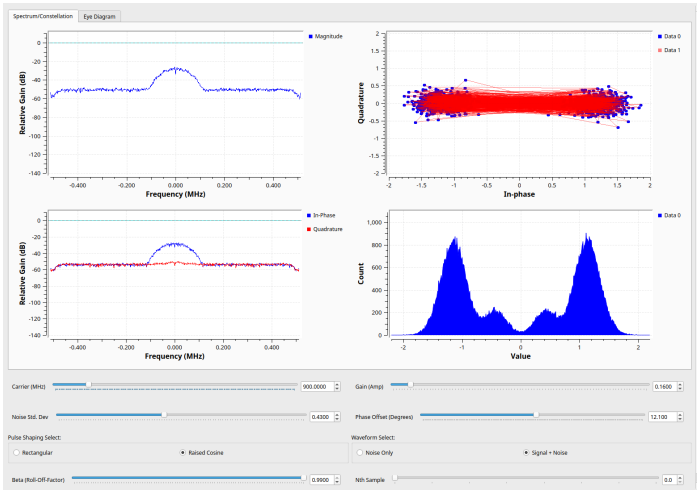
i. β = 0.01



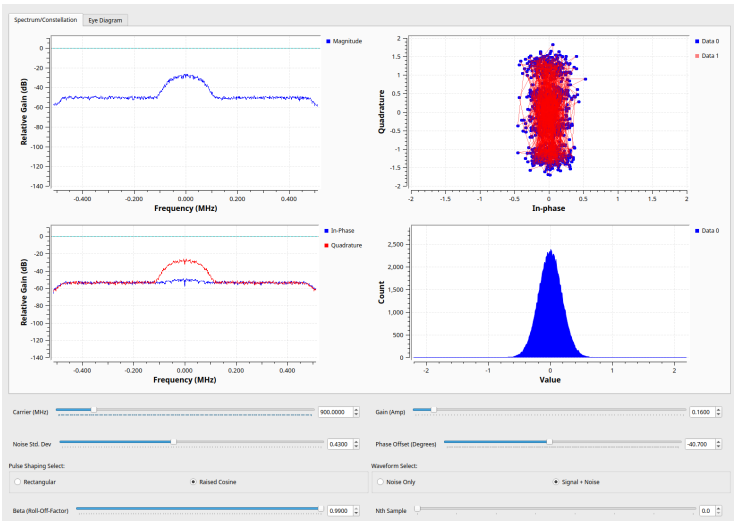
ii. β = 0.5



iii. β = 0.99



c) For the last trace, adjust the phase offset so that the constellation is completely along the *Q* axis. Explain what you see for the histogram. Does this make sense?



Adjusting the phase offset so that the constellations are completely along the Q axis causes the histogram to look tall/skinny centered around 0, this makes sense because the real part is effectively 0, resulting in a histogram that is concentrated around 0 on the Q axis.

Write-Up for Part 2

The write-up for this part of the lab consists of the four screen shots of the transmitted waveforms. For each of the four measured waveforms, compare the measured power density spectrum to the expected power density spectrum from theory. (Note that the frequency of a raised-cosine pulse is given by Eq. (3.4.11) in Papen.) Also compare the bandwidth for the four waveforms at the same symbol rate. What conclusions can be drawn from this comparison?

- A lower roll-off factor (β) results in a wider main lobe in the frequency domain, which leads to a broader bandwidth.
- A higher roll-off factor (β) results in a narrower main lobe and lower sidelobes in the frequency domain, which leads to a narrower bandwidth.

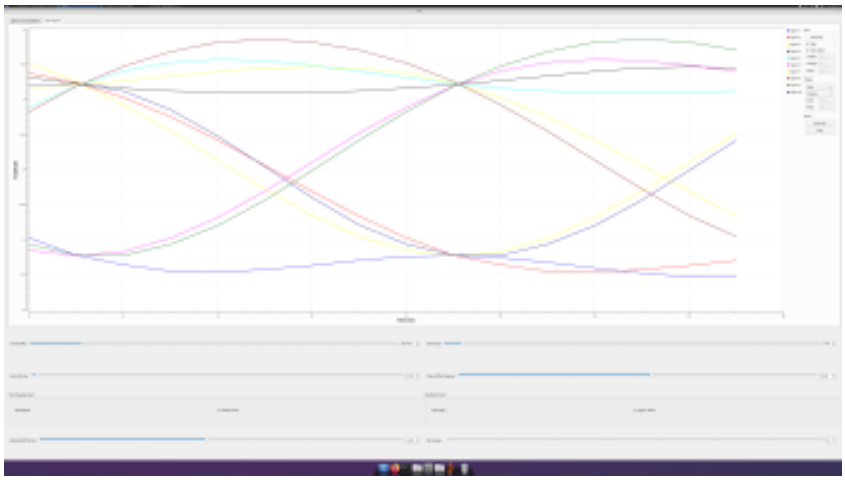
For each of the measured waveforms, their measured power density spectrum was as expected compared to the power density spectrum from theory.

3 Estimating the BER from an Eye Diagram

The superimposed trajectories of multiple waveforms over multiple modulation intervals forms the eye diagram. This eye diagram or pattern can be used to assess the system performance. For a binary-modulated waveform, for each symbol interval, the waveform either does not change from the previous value, or changes from a high to a low, or changes from a low to a high. The superimposed transitions form the eye pattern. The shape of the eye pattern is given by the pulse shape $q(t)$ **after** the detection filter. Referring to Chapter 3 of Papen/Blahut (Figure 3.12 and Eq. (3.4.7)), the *target pulse* $q(t)$ is the composite of the transmitted pulse $x(t)$, the natural dispersion $h(t)$ in the physical channel, and the intentional filtering $y(t)$ in the receiver shown in Figure 3.12a so that

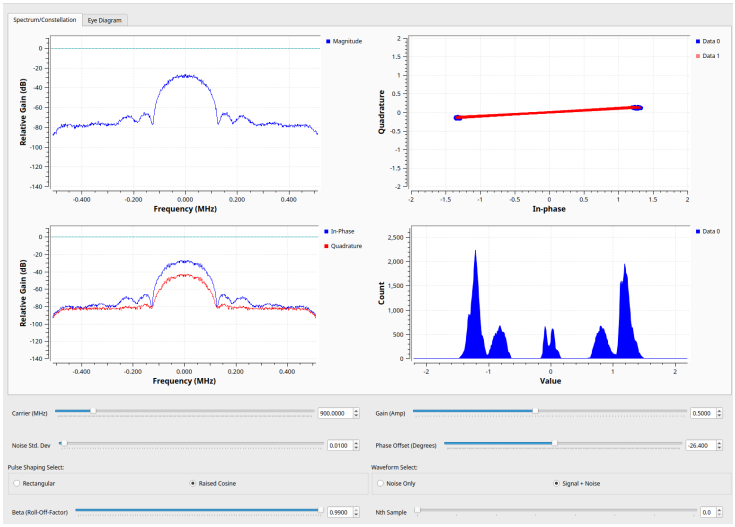
$$q(t) = x(t) \circledast h(t) \circledast y(t). \quad (2)$$

When the sequence of target pulses after the filter has a large eye opening in amplitude, the system is tolerant of noise or errors in the threshold used for hard-decision detection. When the waveform has a wide eye opening in time, the system is tolerant of errors in the sampling time. In this section, we generate an eye-pattern for a sequence of raised-cosine pulses and examine the effect of β on the shape of the eye and on the optimal sampling time. The eye-pattern for a sequence of raised cosine pulses with $\beta = 0.5$ is shown below.



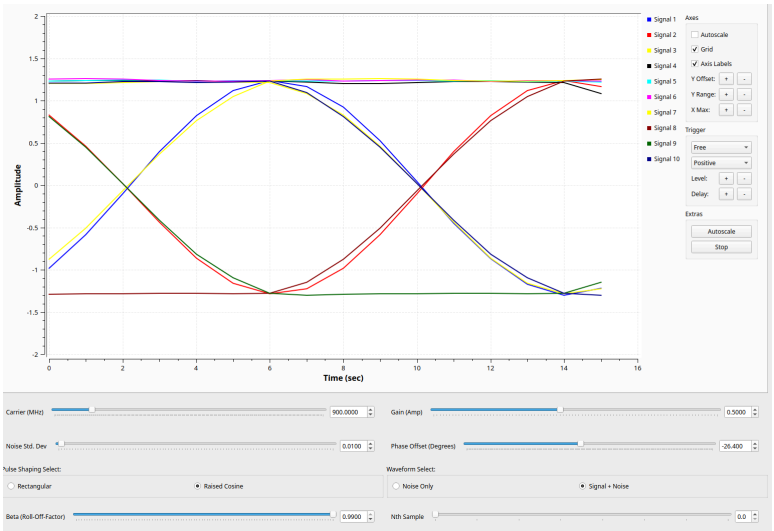
Important: Note that there can be no “dropouts” for this part of the lab to work as the “dropout” indicates that the eye-pattern is no longer synchronized. When this occurs, the eye-pattern is not stable, the histogram will have many samples at zero, and this part of the lab will not work. Please be aware of this as the processing of the eye-pattern is data intensive. You may have to boot directly into Ubuntu for this part to work.) If it does not work, you may have to simulate the data instead of collecting real data.

1. Set β = 0.99. What kind of pulse is this?

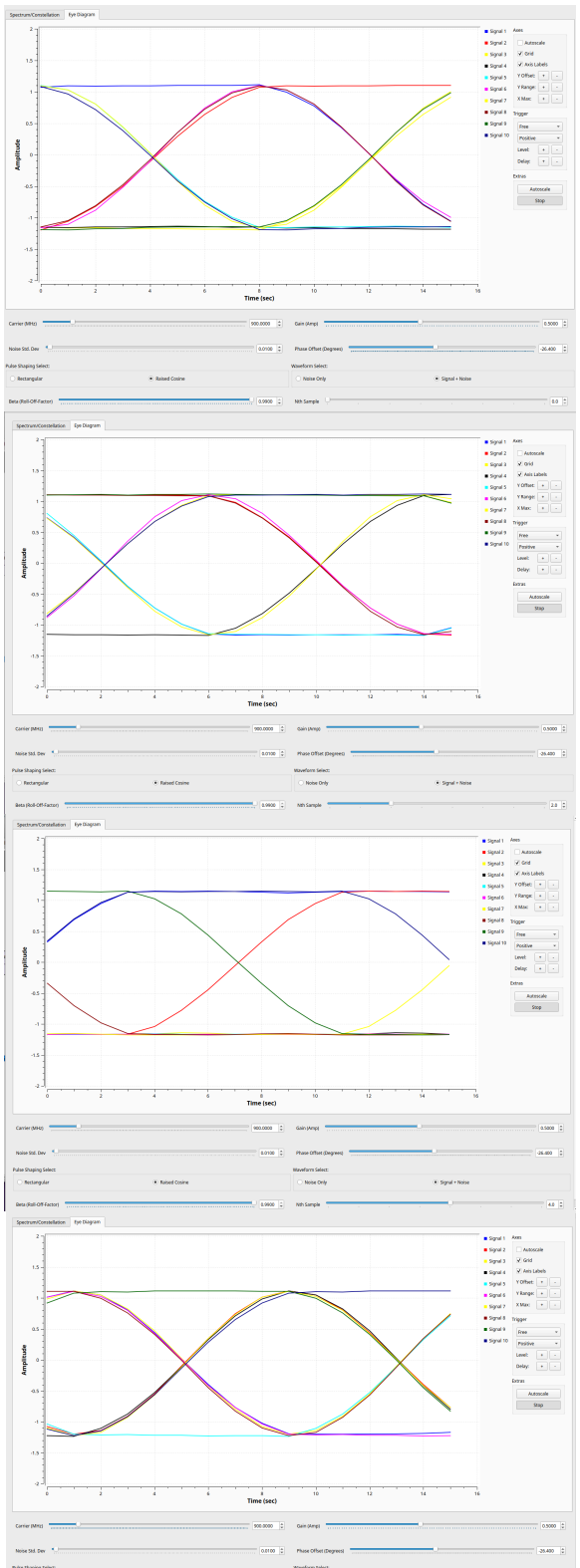


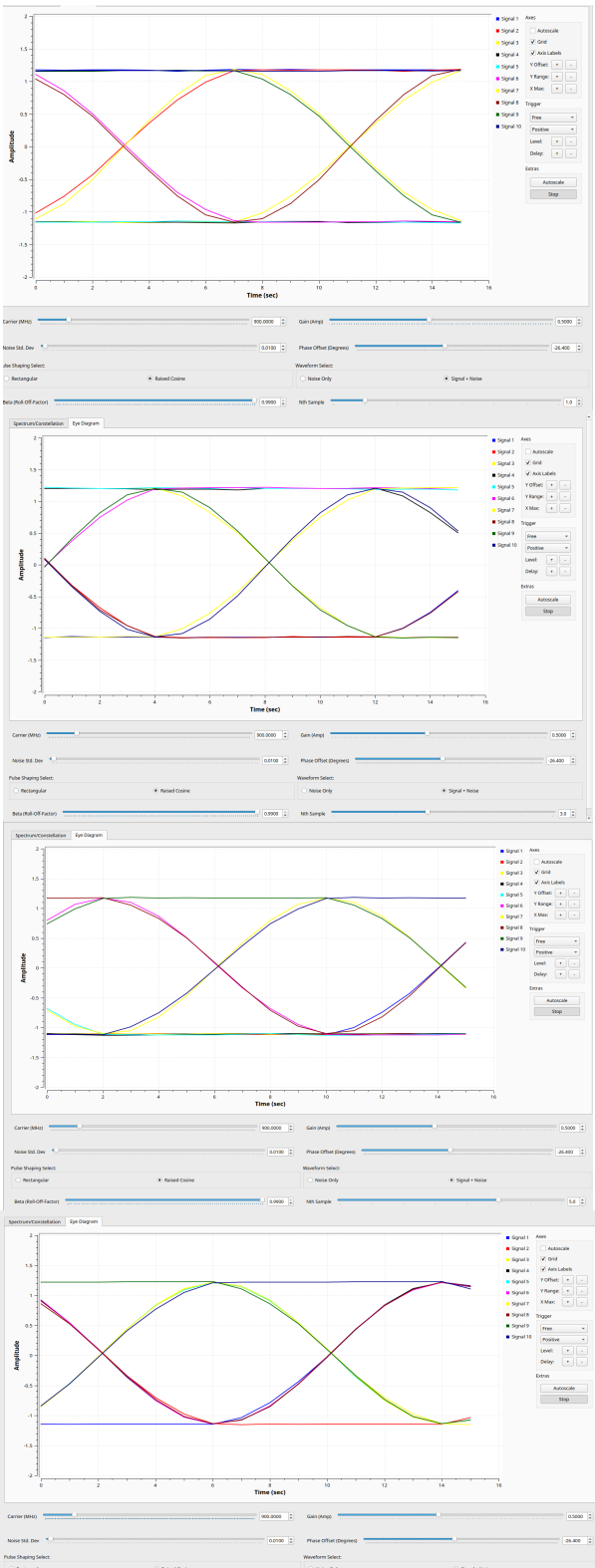
This is a raised cosine pulse shape but the 0.99 means that the pulse shape has a very smooth transition between symbols, almost resembling a square pulse.

2. In the upper-left part of the front panel, click the “Eye Diagram” tab. Adjust the phase offset slider such that you maximize the eye opening (i.e. all the energy in the BPSK signal lies on the in-phase axis of the constellation plot).



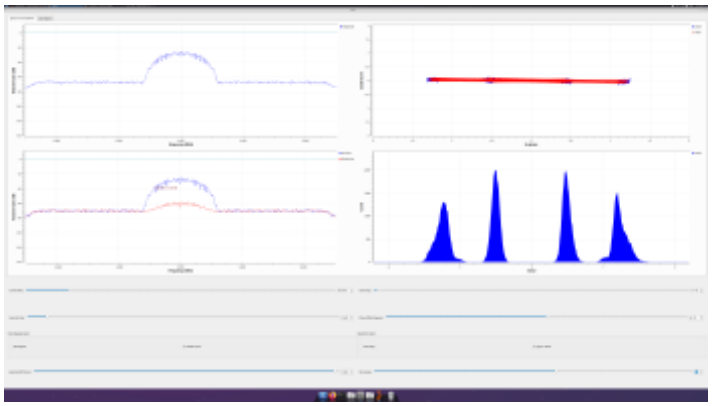
3. We will now “scan” the eye to find the sampling time that maximizes the eye opening. To do this, use the "nth sample(0 − samplespersymbol)" control and step through the values from zero to the value you specified under "samples per symbol".





With every sample, the eye diagram shifts.

4. When the eye-pattern is stable, you should see a gradual change in the location of the "crossing" on the eye-pattern. Returning to the panel that shows the histograms, as you change the value *n* of the sample time, the time "slice" histogram may show either two, three, or four distributions. An example of a "slice showing four parts of the distribution is shown below.



5. Take a screenshot for each of the sampling times *n*, where $n \in$ [0, Samples Per Symbol]. For each value of *n*, estimate the mean value s_1 for the part of the "mark" or logical "one" histogram that is closest to zero. Repeat for s_0, which is the part of the "space" or logical "zero" histogram that is closest to zero. The magnitude (not the sign) of these values should be close. Why? Also estimate σ.

   Under the condition that the noise is gaussian and zero mean, using

   $$A = s_1 - s_0$$

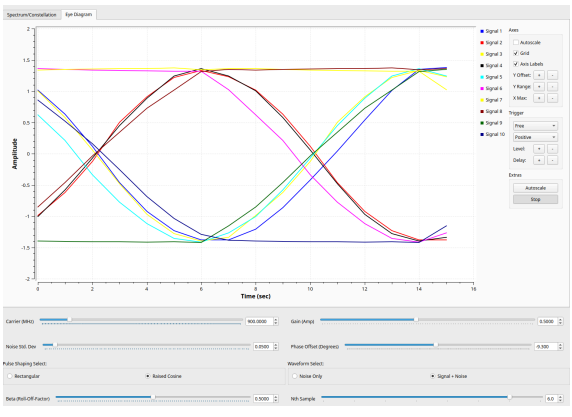
   and the rms noise value σ, estimate the probability p_e of a bit error (BER) using (cf. Slides 19-20 from Lecture 1B) where

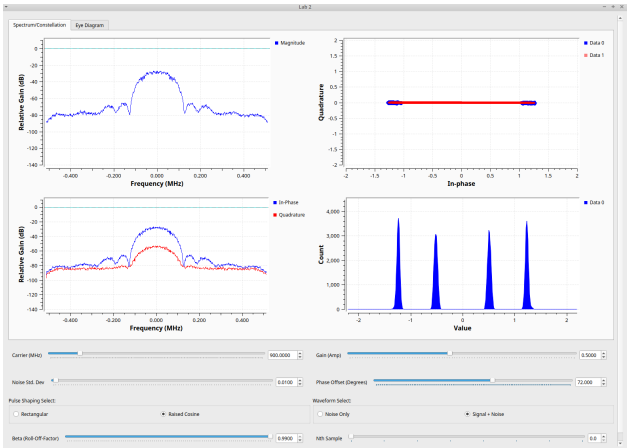
$$\mathrm{BER} = \tfrac{1}{2}\mathrm{erfc}\left(A/\sqrt{2}\sigma\right) = Q(A/\sigma)$$

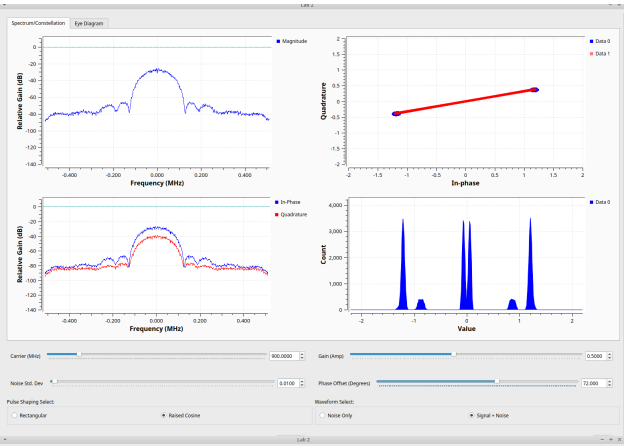
$$Q(x) \doteq \tfrac{1}{2}\mathrm{erfc}(x/\sqrt{2}).$$

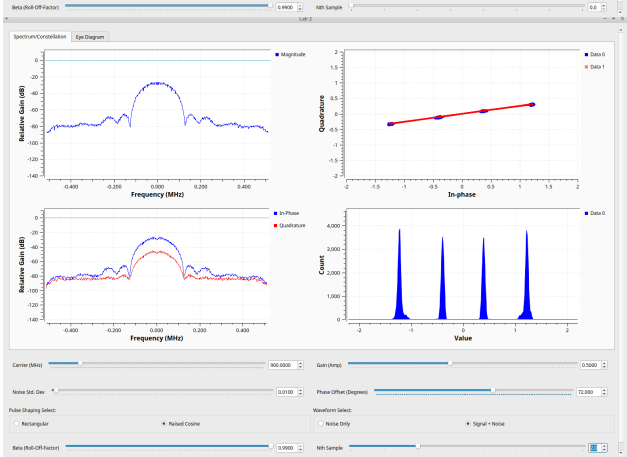
Record the screenshot for the sampling time n_{max} that produces the largest eye opening and the values $n_{max} \pm 1$ on either side. This will give three screenshots. Determine the BER for each value of *n*.

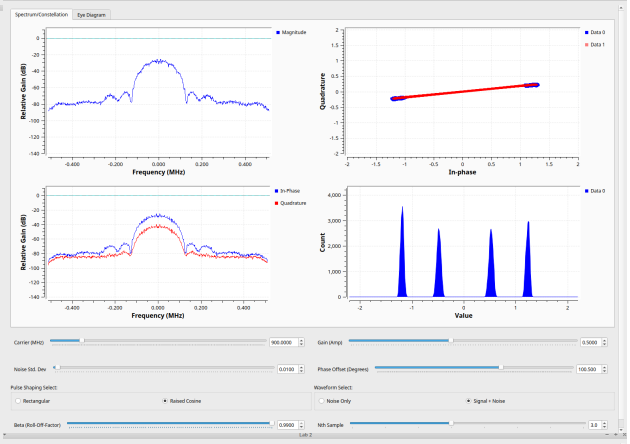
N_max (0.99): n = 6

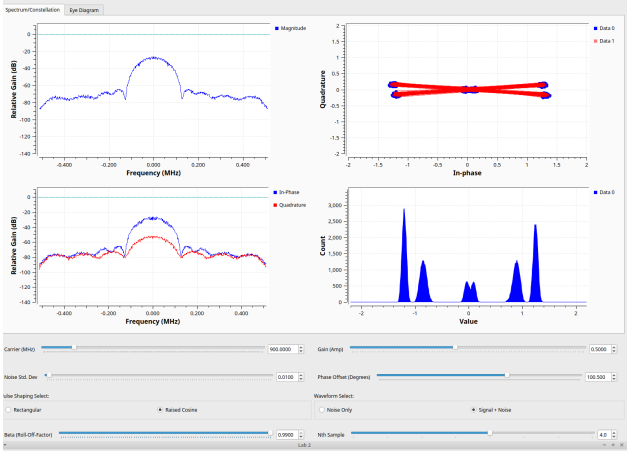


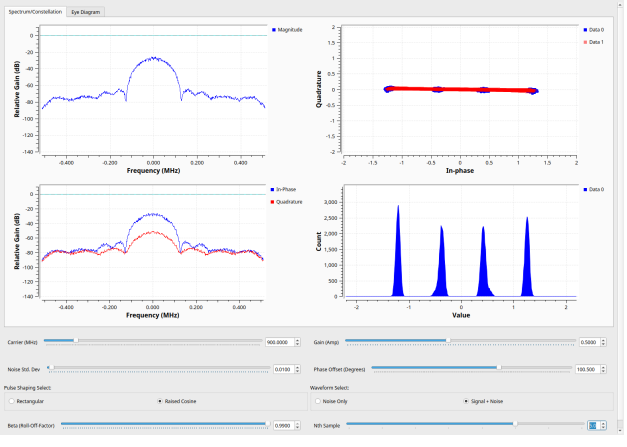

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
72.000
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample
8.0

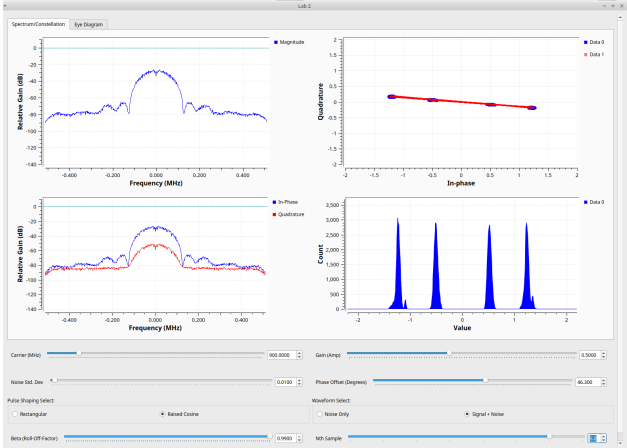

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
72.000
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise

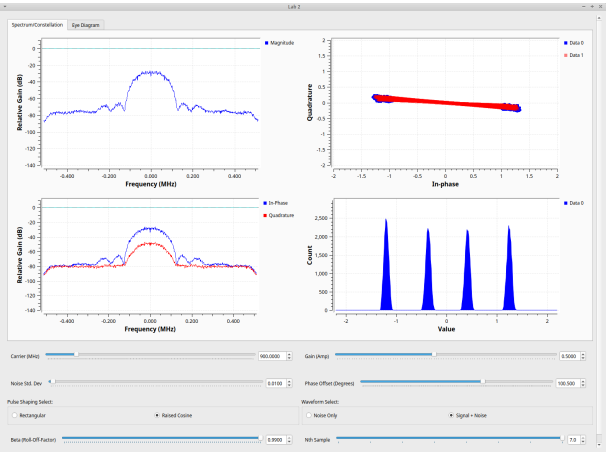

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
72.000
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample


Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
100.500
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample
3.0


Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
100.500
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample
4.0


Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
100.500
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample

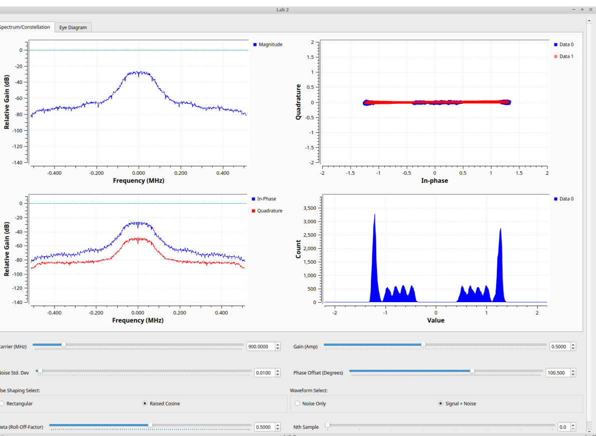

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
46.000
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample

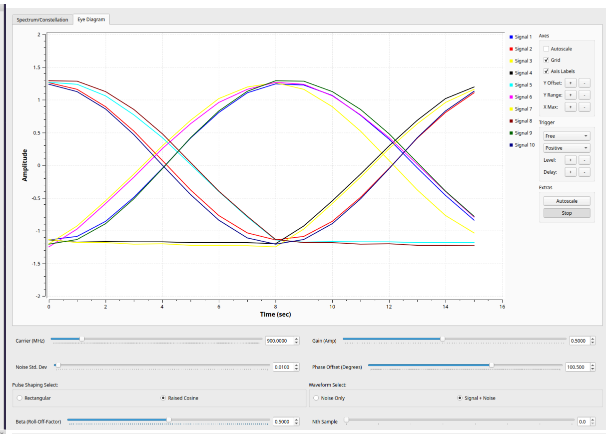

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0100
Phase Offset (Degrees)
100.500
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.9900
Nth Sample
7.0

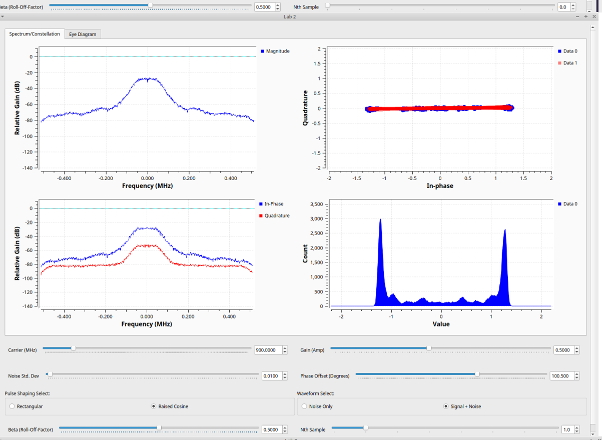
N:	0	1	2	3	4	5	6	7
A:	1	0.1	0.78	1	1.75	.8	1	0.8
BER:	$7.6 \cdot 10^{-24}$	0.25	$4.8 \cdot 10^{-5}$	$2.9 \cdot 10^{-7}$	$2.33 \cdot 10^{-4}$	$3.17 \cdot 10^{-5}$	$6.2 \cdot 10^{-3}$	0.0227

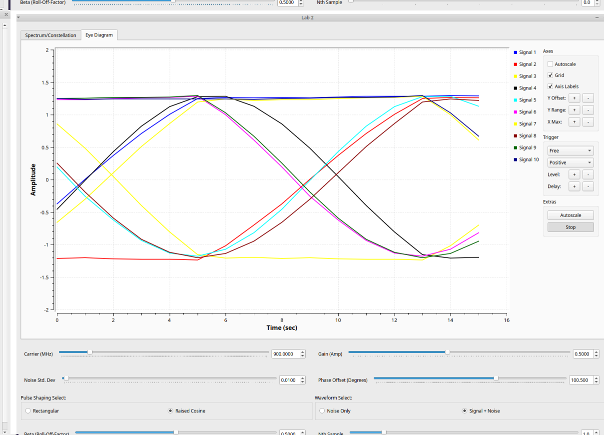
6. Repeat step (5) for $\beta = 0.5$ and $\beta = 0$ (sinc pulses).

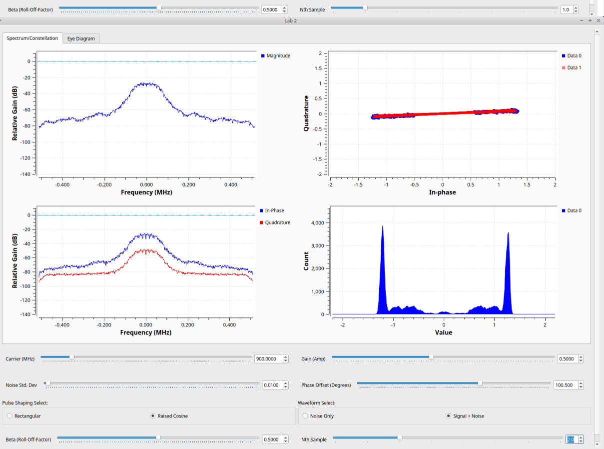
$\beta = 0.5$

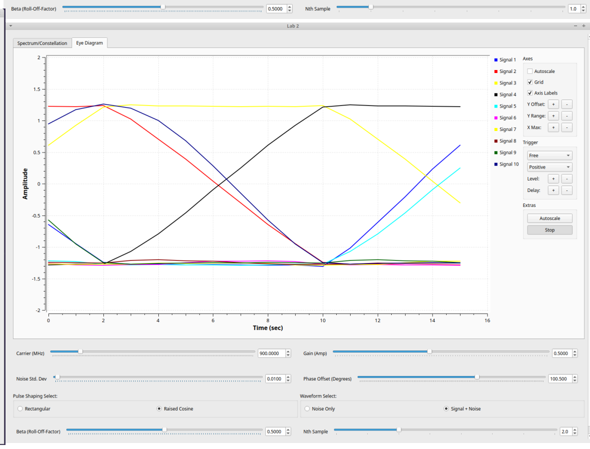


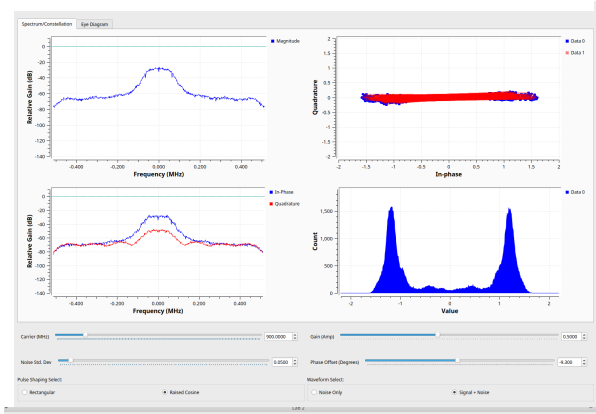


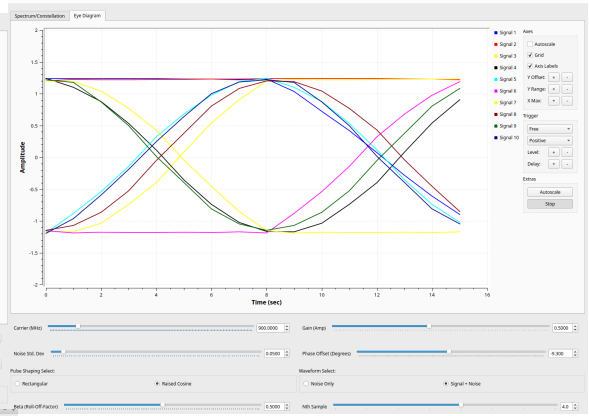


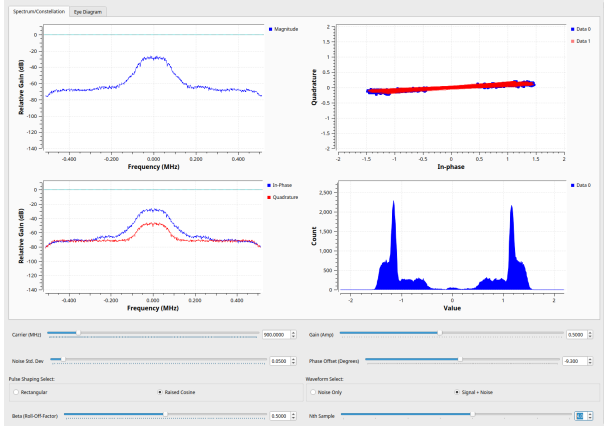


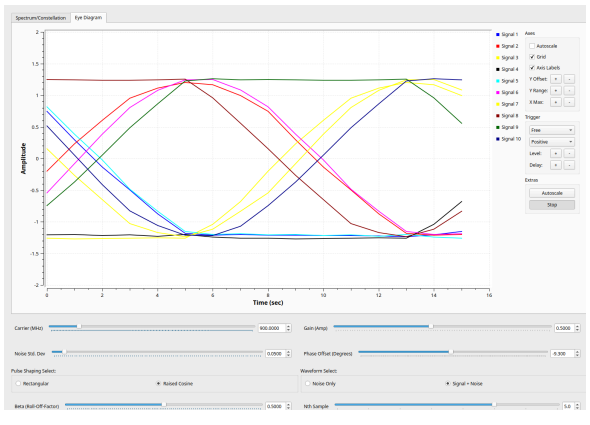


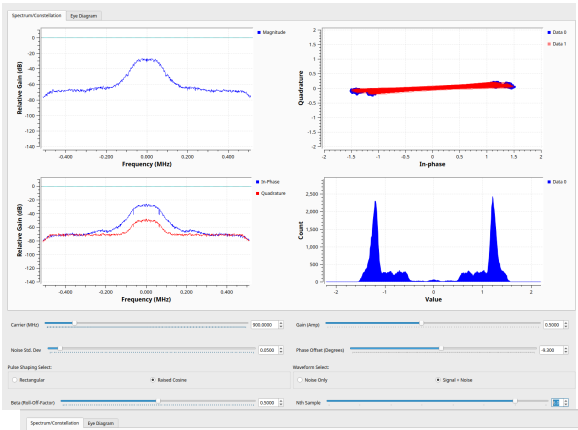


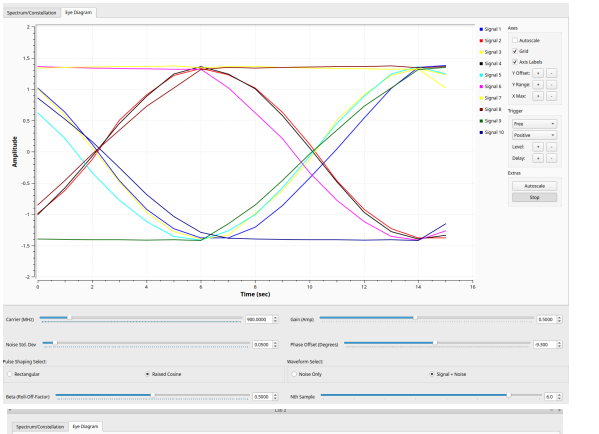

Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
4.0

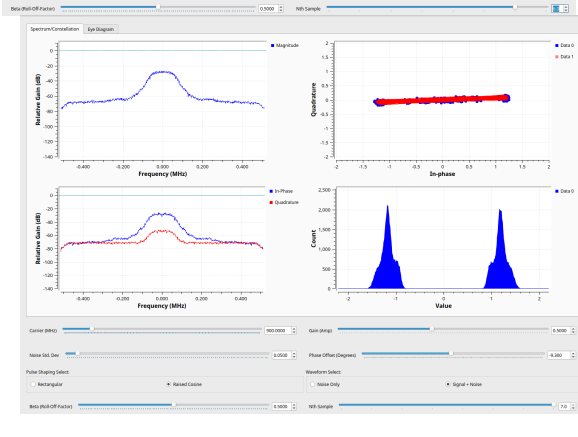

Spectrum/Constellation
Eye Diagram
Signal 1
Signal 2
Signal 3
Signal 4
Signal 5
Signal 6
Signal 7
Signal 8
Signal 9
Signal 10
Axes
Autoscale
Grid
Axis Labels
Y Offset:
Y Range:
X Max:
Trigger
Free
Positive
Level:
Delay:
Extras
Autoscale
Stop
Amplitude
Time (sec)
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
4.0

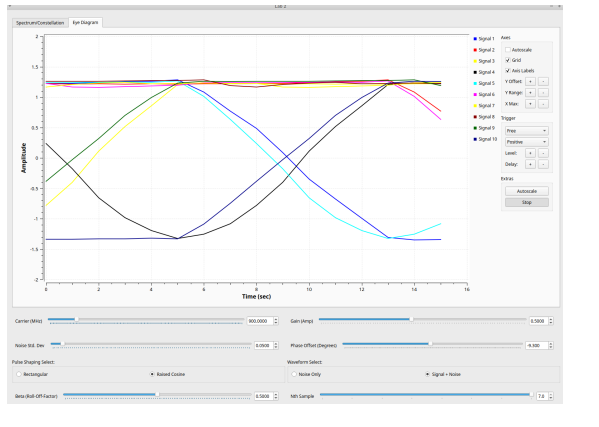

Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample


Spectrum/Constellation
Eye Diagram
Signal 1
Signal 2
Signal 3
Signal 4
Signal 5
Signal 6
Signal 7
Signal 8
Signal 9
Signal 10
Axes
Autoscale
Grid
Axis Labels
Y Offset:
Y Range:
X Max:
Trigger
Free
Positive
Level:
Delay:
Extras
Autoscale
Stop
Amplitude
Time (sec)
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
5.0


Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample

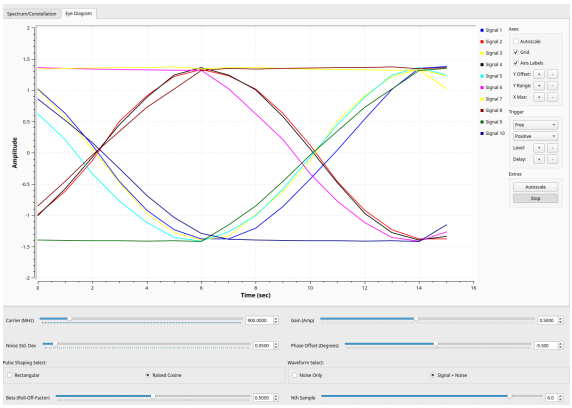

Spectrum/Constellation
Eye Diagram
Signal 1
Signal 2
Signal 3
Signal 4
Signal 5
Signal 6
Signal 7
Signal 8
Signal 9
Signal 10
Axes
Autoscale
Grid
Axis Labels
Y Offset:
Y Range:
X Max:
Trigger
Free
Positive
Level:
Delay:
Extras
Autoscale
Stop
Amplitude
Time (sec)
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
6.0


Spectrum/Constellation
Eye Diagram
Magnitude
Data 0
Data 1
Relative Gain (dB)
Frequency (MHz)
Quadrature
In-phase
In Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample

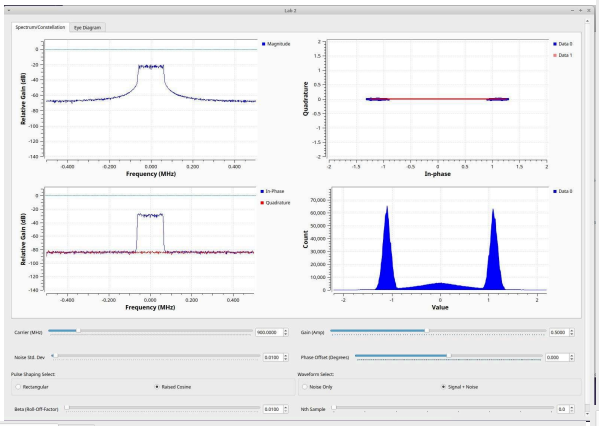

Spectrum/Constellation
Eye Diagram
Signal 1
Signal 2
Signal 3
Signal 4
Signal 5
Signal 6
Signal 7
Signal 8
Signal 9
Signal 10
Axes
Autoscale
Grid
Axis Labels
Y Offset:
Y Range:
X Max:
Trigger
Free
Positive
Level:
Delay:
Extras
Autoscale
Stop
Amplitude
Time (sec)
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
-9.300
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
7.0

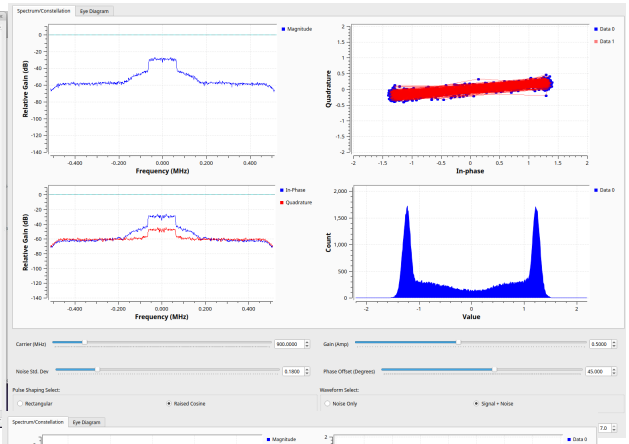
N:	0	1	2	3	4	5	6	7
A:	2.5	2.4	2.45	2.5	2.4	2.4	2.6	2.4
BER:	$1.1 \cdot 10^{-214}$	$4.9 \cdot 10^{-198}$	$2.8 \cdot 10^{-206}$	$1.1 \cdot 10^{-214}$	$4.9 \cdot 10^{-214}$	$9.9 \cdot 10^{-10}$	$6.1 \cdot 10^{-39}$	$9.9 \cdot 10^{-10}$

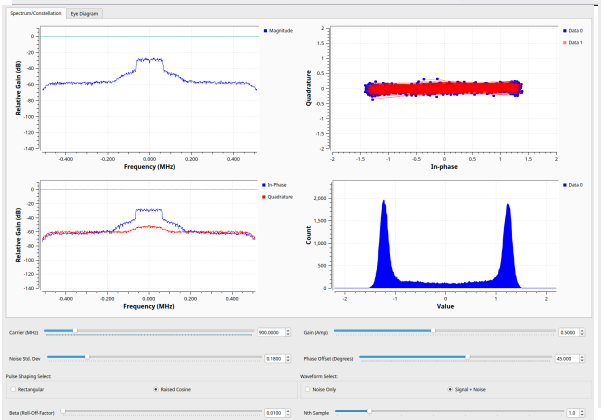
N_max for 0.5: n=6

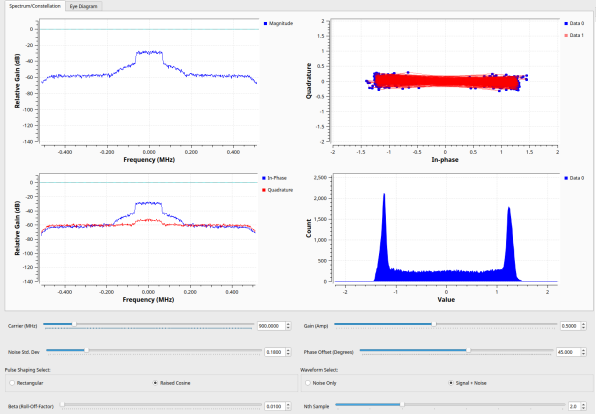


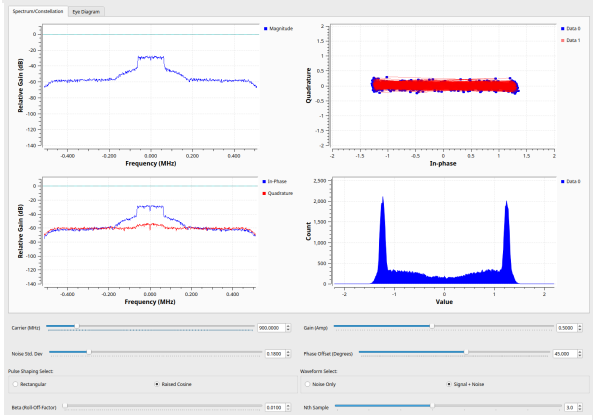
β = 0:

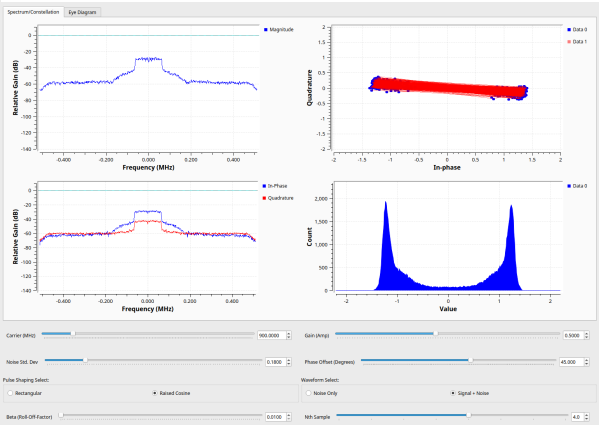


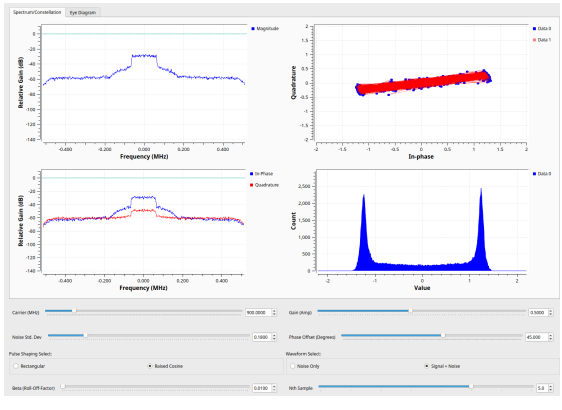


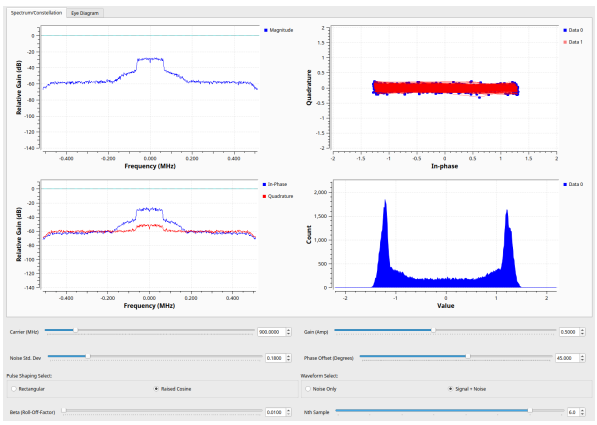


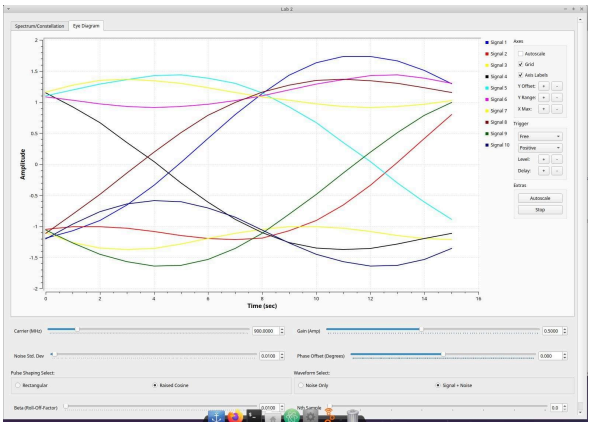








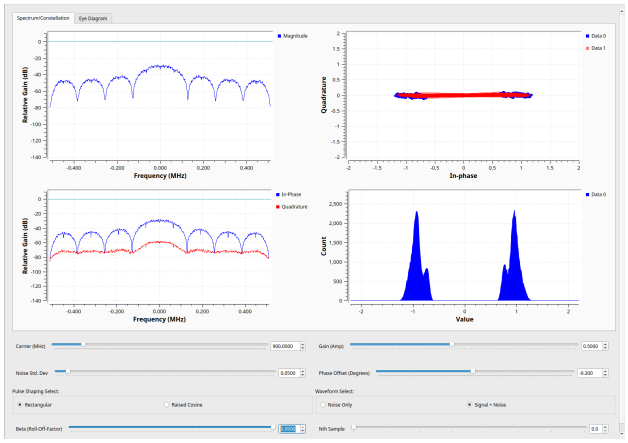


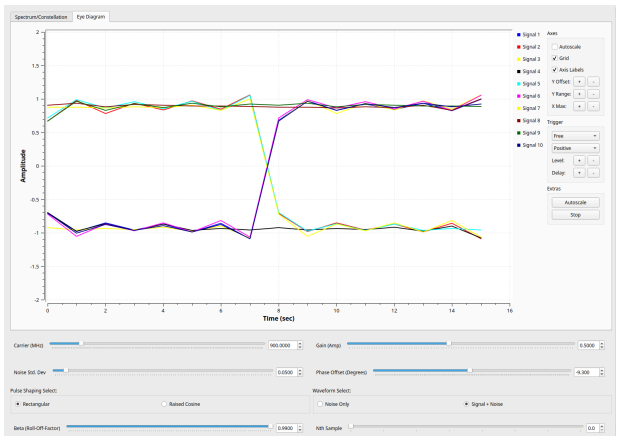


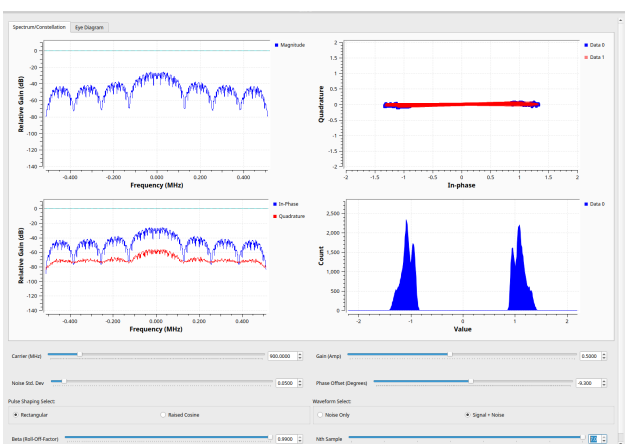
N_max for 0.01: n=6

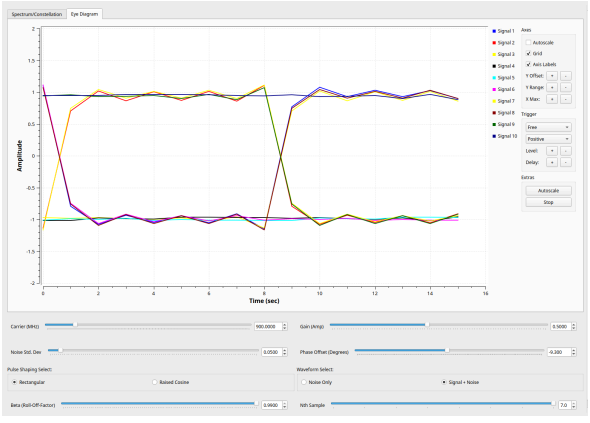
N:	0	1	2	3	4	5	6	7
A:	2.1	2.4	2.5	2.4	2.4	2.45	2.4	2.45
BER:	$4.2 \cdot 10^{-26}$	$1.8 \cdot 10^{-33}$	$3.7 \cdot 10^{-36}$	$1.8 \cdot 10^{-33}$	$1.8 \cdot 10^{-33}$	$8.4 \cdot 10^{-35}$	$1.8 \cdot 10^{-33}$	$8.4 \cdot 10^{-35}$

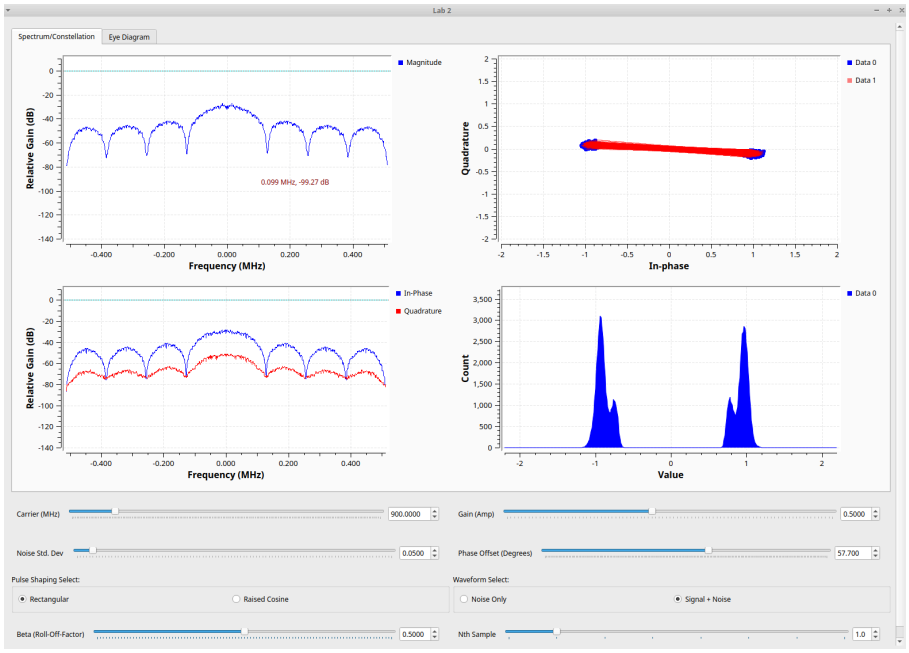
7. Set the transmit pulse to rect and repeat step (5). Note that this part will produce Samples Per Symbol+1 screenshots with several of the screenshots being similar.

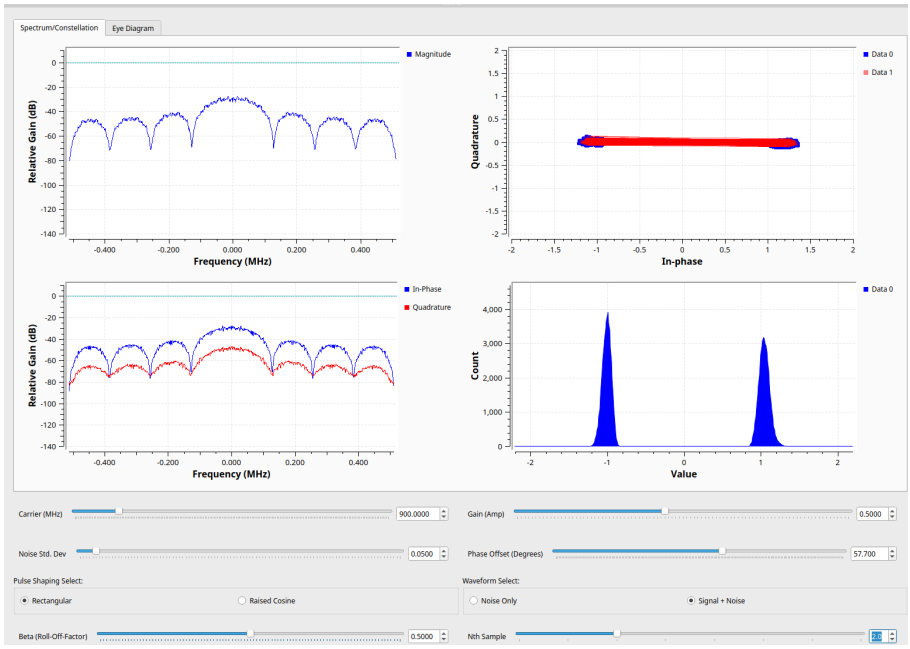


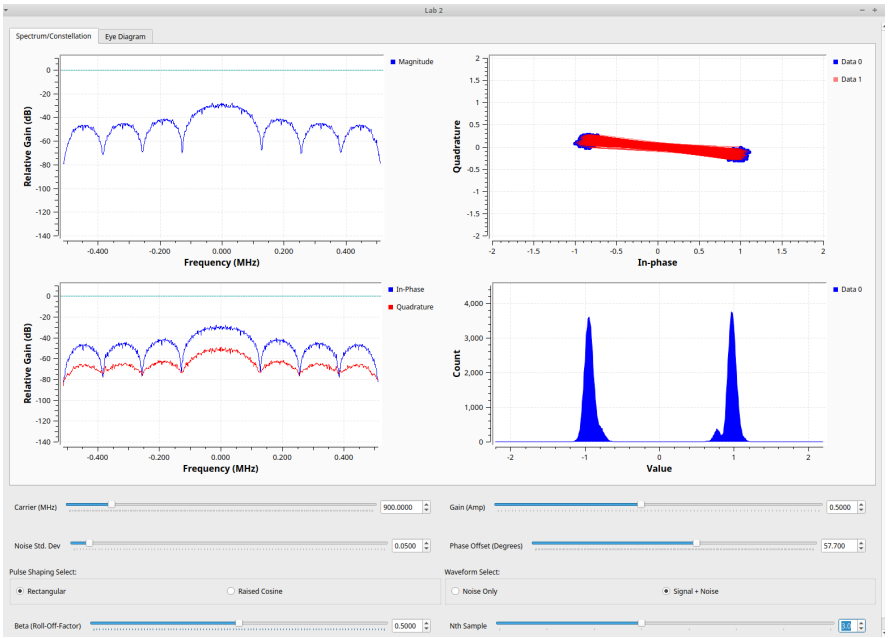


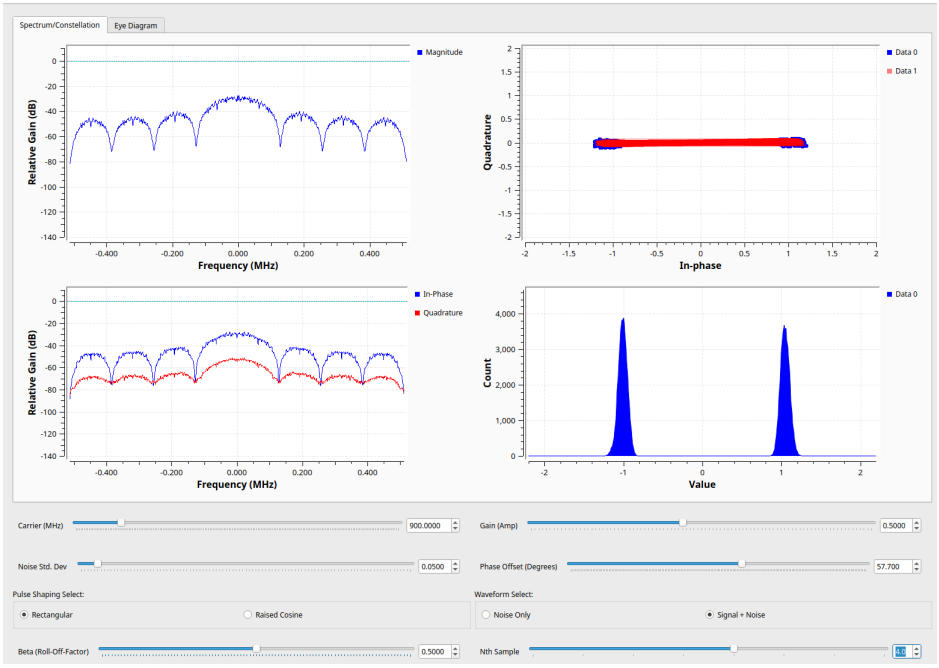


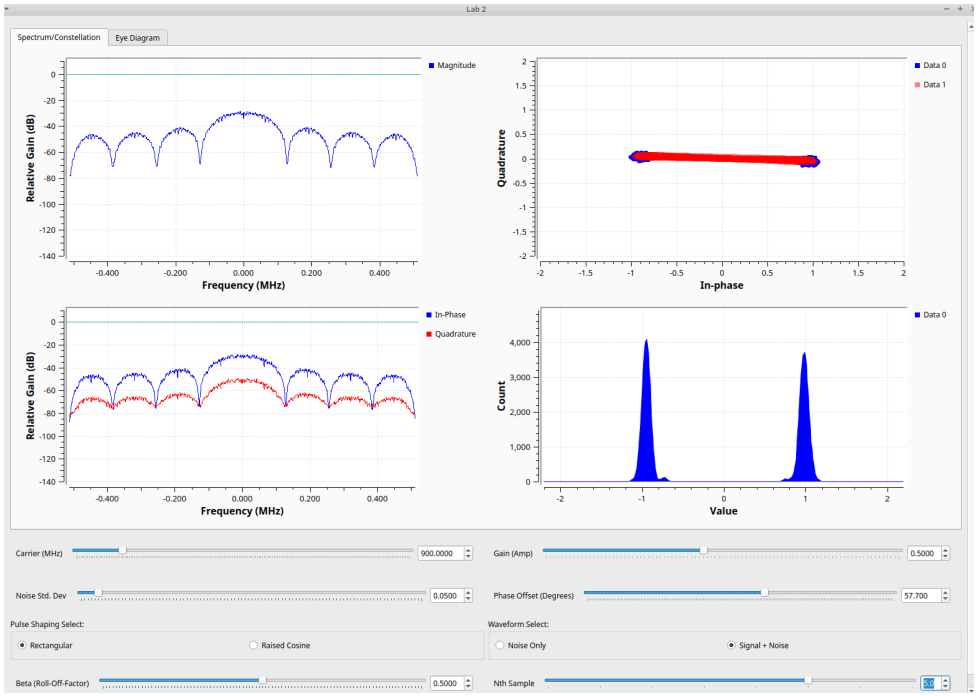


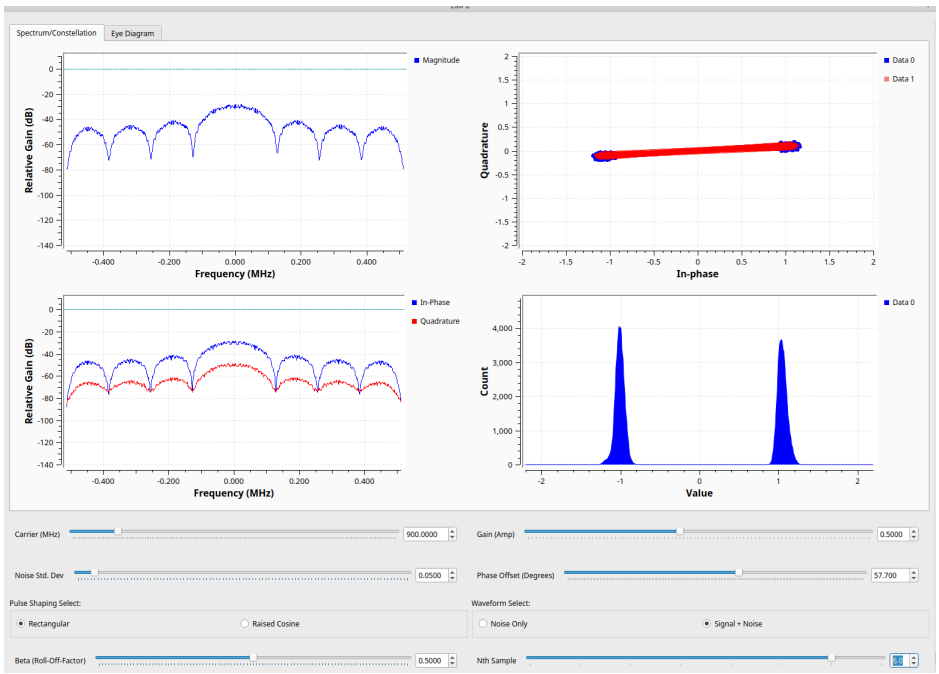

Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
0.099 MHz, -99.27 dB
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Relative Gain (dB)
Frequency (MHz)
Data 0
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample
1.0

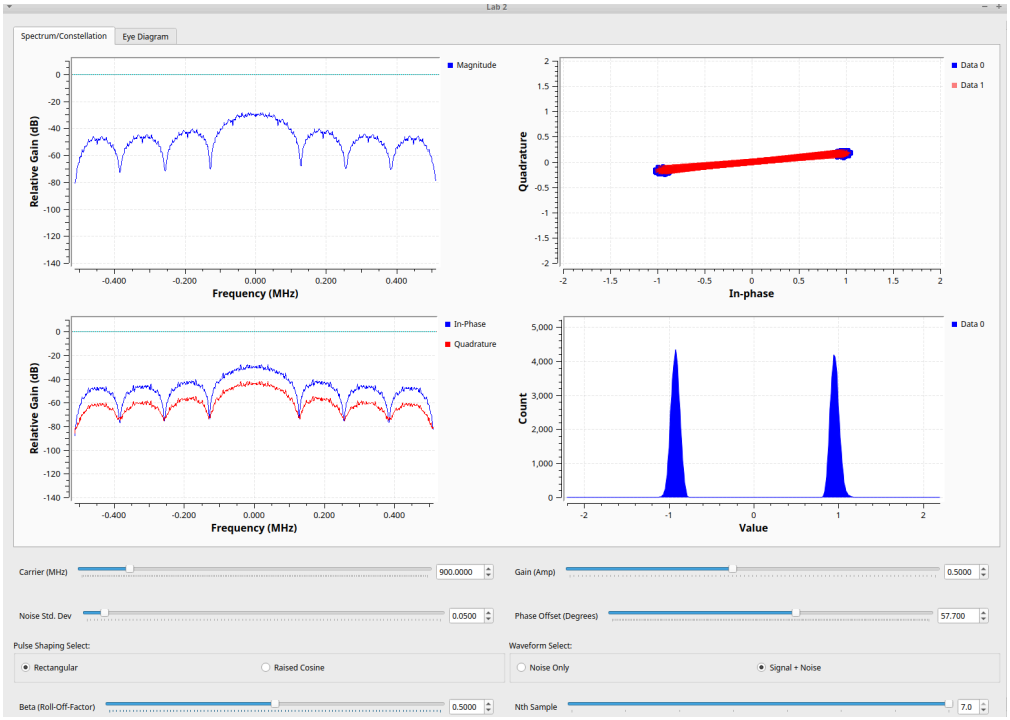

Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Relative Gain (dB)
Frequency (MHz)
Data 0
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample


Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample


Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample


Lab 2
Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample


Spectrum/Constellation
Eye Diagram
Magnitude
Relative Gain (dB)
Frequency (MHz)
Data 0
Data 1
Quadrature
In-phase
In-Phase
Quadrature
Count
Value
Carrier (MHz)
900.0000
Gain (Amp)
0.5000
Noise Std. Dev
0.0500
Phase Offset (Degrees)
57.700
Pulse Shaping Select:
Rectangular
Raised Cosine
Waveform Select:
Noise Only
Signal + Noise
Beta (Roll-Off-Factor)
0.5000
Nth Sample



N_max = 6

N:	0	1	2	3	4	5	6	7
A:	2	2.2	2.05	1.9	2.05	2	2.1	1.7
BER:	$7.6 \cdot 10^{-24}$	$1.3 \cdot 10^{-38}$	$8.7 \cdot 10^{-34}$	$2.7 \cdot 10^{-29}$	$8.7 \cdot 10^{-34}$	$3 \cdot 10^{-32}$	$2.34 \cdot 10^{-35}$	$7.6 \cdot 10^{-22}$

Write-Up for Part 3

The write-up for this part of the lab consists of the following:

1. Screenshots of the eye-pattern for the three raised-cosine pulse waveforms (β = 0, β = 0.5 β = 0.99) and the rectangular pulse waveform. This is a total of four screenshots.

2. Screenshots of the histogram for the optimal sampling time n_{max} for each of the three raised cosine spectrum waveforms and the corresponding calculation of the BER using estimates of A and σ derived from the screen shots.

3. BER calculations for $n_{max} \pm 1$ for the three raised-cosine spectrum waveforms. You do not need to turn in the screenshots, but you will need them to estimate A (and σ) for each sample time. Note that for sampling times that produce histograms with multiple part of the histogram as shown in the figure, use the part of the histogram that is separated by the minimum distance.

4. Quantitatively discuss why the approximation of using just of the parts of the histogram that are separated by the minimum distance is reasonable.

5. Answers to the following questions.

   a) How sensitive is the BER to the sampling time for each value of β?

   Lower values of β: Higher BER

   Higher values of β: Lower BER

   b) Which value of β causes the BER to be most sensitive to the sampling time?

   $\beta = 0.01$ is the most sensitive to sampling time because it results in a wider main lobe in the frequency domain, which leads to more significant overlap between adjacent symbols when the sampling time is off. This increased overlap causes more ISI and a higher BER for small variations in the sampling time.

   c) Which value of β is the least sensitive? Why?

   $\beta = 0.99$ is the least sensitive to sampling time because it has a narrower main lobe and lower sidelobes, which means that the pulse shape is more concentrated in time. Even if the sampling time is slightly off, the impact on adjacent symbols is small, leading to less ISI and a lower BER.

6. Compared to the raised cosine pulses, how sensitive is the estimated BER to the sampling time *n* for rectangular pulses? How is this sensitivity related to the bandwidth of the power density spectrum of a sequence of rectangular pulses compared to a sequence of raised cosine pulses? For these observations, what conclusions can be made about the accuracy of the sampling and the bandwidth of the pulse?

   For rectangular pulses, the sensitivity of the BER to the sampling time is higher than the raised cosine pulses because it has wide bandwidth and exhibits ISI even for small variations in sampling time. The sensitivity of the BER is related to the bandwidth because the wider the bandwidth, the more sensitive the system is to variation in the sampling time, this sensitivity is due to the fact that wider bandwidth means that the pulse is spread out more over time, leading to more overlap between symbols and increased ISI. While the raised cosine pulses have a narrower bandwidth which means less ISI and the BER is less sensitive to variations in the sampling time. Overall, the accuracy of the sampling and the bandwidth are closely related to the sensitivity of the BER to the sampling time.

4 Post Lab Simulation - Generation of Pseudo Random Binary Sequence (PRBS)

In this post lab, you will be observing the difference between the power spectrum of a PRBS (pseudo random binary sequence) and a random binary sequence using square pulses. This simulation will upsample a stream of pseudo-random and random binary symbols by a factor of L and then filter the resulting sequence of pulses using a rectangular interpolation filter (i.e. $h(n) = 1,\ 0 \leq n < L$)

4.1 Spectra of PRBS Sequences

1. Using the matlab script *pngen.m* which can be found on the website under the “labs” section, create a 1024 bit pseudo-random sequence.

$$x = \text{pngen}(7, 1024)$$

This creates a periodic pseudo-random binary sequence with a period of $2^7 - 1$ bits. The value of M is called the **polynomial order** of the sequence.

a) Convert the 1024-sample sequence into an interpolated 8192 sample sequence by using the *upsample* and *filter* functions in Matlab. (Note: the upsampling factor for this simulation is 8).

i. Convert this stream of logical symbols into BPSK symbols. This is called **symbol mapping**.

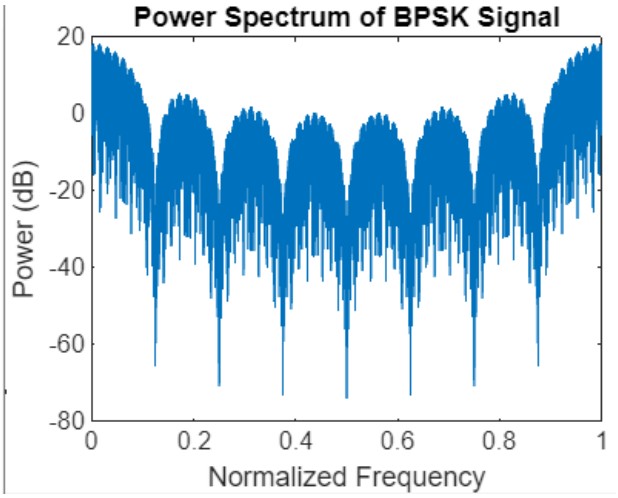
```
x = pngen(7, 1024);
up_x = upsample(x, 8);
h1 = ones(1, 8);
filtered_x = filter(h1, 1, up_x);
bpsk_x = 2 * filtered_x - 1;

fs = 8192;
N = length(bpsk_x);
frequencies = (0:N-1) / N;

fft_x = fft(bpsk_x);
power_x = abs(fft_x).^2 / N;
power_dB = 10 * log10(power_x);

figure(1);
plot(frequencies, power_dB);
title('Power Spectrum of BPSK Signal');
xlabel('Normalized Frequency');
ylabel('Power (dB)');
```

ii. Plot the power spectrum of this signal. Use a dB-scale and normalized frequency units.



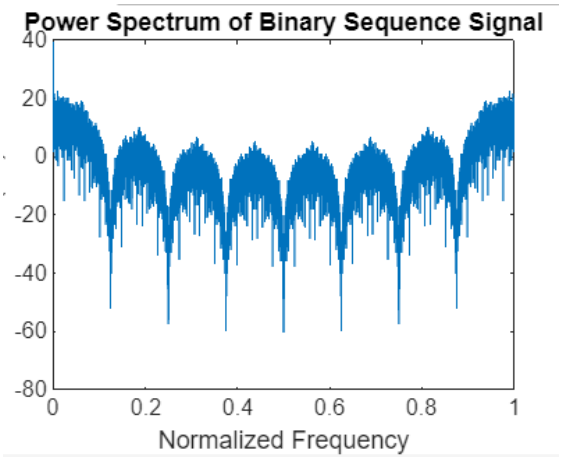
b) Repeat part (a) for a random binary sequence of ±1's. You can use the *randsrc(.)* matlab function.

```
x = randsrc(1, 1024, [-1,1]);
up_x = upsample(x, 8);
h1 = ones(1, 8);
filtered_x = filter(h1, 1, up_x);
bpsk_x = 2 * filtered_x - 1;

fs = 8192;
N = length(bpsk_x);
frequencies = (0:N-1) / N;

fft_x = fft(bpsk_x);
power_x = abs(fft_x).^2/N;
power_dB = 10 * log10(power_x);

figure(2);
plot(frequencies, power_dB);
title('Power Spectrum of Binary Sequence Signal');
xlabel('Normalized Frequency');
ylabel('Power (dB)');
```



c) The power spectrum for the PRBS consists of a sequence of lines that have a $\text{sinc}^2(x)$ envelope. Given that the signal $s(t)$ has a Fourier series representation given by

$$s(t) \sim \sum_{n=-\infty}^{\infty} A_n e^{j\omega_c n t}$$

determine the power in the "DC" component (i.e. corresponding to Fourier coefficient A_0) for both the PRBS and the random binary sequence. Discuss your results.

```
% PRBS sequence
N = 1024; % Length of the PRBS sequence
prbs_sequence = pngen(7, N); % Generate the PRBS sequence
prbs_fft = fft(prbs_sequence) / N; % Compute the Fourier transform
A0_prbs = prbs_fft(1); % DC component (A0)
power_dc_prbs = abs(A0_prbs)^2; % Power in the DC component

% Random binary sequence
rand_sequence = randsrc(1, N, [-1, 1]); % Generate the random binary sequence
rand_fft = fft(rand_sequence) / N; % Compute the Fourier transform
A0_rand = rand_fft(1); % DC component (A0)
power_dc_rand = abs(A0_rand)^2; % Power in the DC component

% Display the results
disp(['Power in the "DC" component for PRBS: ', num2str(power_dc_prbs)]);
disp(['Power in the "DC" component for random binary sequence: ', num2str(power_dc_rand)]);
```

```
Power in the "DC" component for PRBS: 0.25688
Power in the "DC" component for random binary sequence: 0.0015259
```

For the PRBS, the power in the "DC" component is non-zero due to the sinc² envelope of the power spectrum. But for the random binary sequence, the power in the "DC" component is pretty much zero due to the flat power spectrum.

d) Determine the frequency spacing between the spectral lines in the power density spectrum as well as the spacing between the nulls for the PRBS.

```
% Given parameters
N = 1024; % Length of the PRBS sequence
M = 7; % Polynomial order of the PRBS

% Calculate frequency spacing between spectral lines
delta_f = 1 / N;

% Calculate null-to-null spacing
null_to_null_spacing = 1 / N * (2^M - 1);

% Display the results
disp(['Frequency spacing between spectral lines: ', num2str(delta_f), ' Hz']);
disp(['Null-to-null spacing: ', num2str(null_to_null_spacing), ' Hz']);
```

```
Frequency spacing between spectral lines: 0.00097656 Hz
Null-to-null spacing: 0.12402 Hz
```

e) Determine the frequency spacing between the nulls and the spacings between the spectral lines as the number of samples per symbol varies from N = 4, 8, 16*?*

```
% Given parameters
M = 7; % Polynomial order of the PRBS
Ns = [4, 8, 16]; % Number of samples per symbol

for i = 1:length(Ns)
    N = Ns(i);
    % Calculate frequency spacing between spectral lines
    delta_f = 1 / N;

    % Calculate null-to-null spacing
    null_to_null_spacing = 1 / N * (2^M - 1);

    % Display the results
    disp(['For N = ', num2str(N)]);
    disp(['Frequency spacing between spectral lines: ', num2str(delta_f), ' Hz']);
    disp(['Null-to-null spacing: ', num2str(null_to_null_spacing), ' Hz']);
end
```

```
For N = 4
Frequency spacing between spectral lines: 0.25 Hz
Null-to-null spacing: 31.75 Hz
For N = 8
Frequency spacing between spectral lines: 0.125 Hz
Null-to-null spacing: 15.875 Hz
For N = 16
Frequency spacing between spectral lines: 0.0625 Hz
Null-to-null spacing: 7.9375 Hz
```

Since both the spectral line spacing and the null to null spacing are reliant on N, as N increases, the spacing decreases for both.

f) Now, set *N* = 8. For this value of *N*, how does the spectral line spacing change when you vary the polynomial order *M* of the PRBS from *M* = 5, 6, 7?

```
N = 8; % Number of samples per symbol
Ms = [5, 6, 7]; % Polynomial orders

for i = 1:length(Ms)
    M = Ms(i);
    % Calculate frequency spacing between spectral lines
    delta_f = 1 / N;

    % Display the result
    disp(['For M = ', num2str(M)]);
    disp(['Frequency spacing between spectral lines: ', num2str(delta_f), ' Hz']);
end
```

```
For M = 5
Frequency spacing between spectral lines: 0.125 Hz
For M = 6
Frequency spacing between spectral lines: 0.125 Hz
For M = 7
Frequency spacing between spectral lines: 0.125 Hz
```

Since spectral lines aren't dependent on M, the spacing stays the same.

4.2 Autocorrelation Properties of PRBS Sequences and Preamble Design

In this part, you will compare the performance of three different sequences that can be used as preambles to packetized data in the presence of additive white gaussian noise (AWGN).

1. The cross-correlation of a known sequence $x(n)$ with a noisy copy of the same sequence $y(n) = x(n) + w(n)$ is given by

$$R_{xy}(m) = \sum_{n=0}^{L-1} x(n)y(m+n)$$

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

where $w(n)$ is a sequence of zero-mean independent gaussian random variables with variance σ^2 that simulates AWGN. When the known sequence $x(n)$ is present in the noisy sequence of sampled data $y(n)$ at index m of the array, the cross-correlation $R_{xy}(m)$ should have a maximum at index m_{max}.

2. To start, generate three different types of binary sequences $x[n]$.

   a) "All Ones": a sequence of all ones given by $x[n]$

$$x[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

   b) "Pseudo-Random": a pseudo-random sequence of $+1's$ and $-1's$ using provided function *pngen.m*

$$x = pngen\ (log_2 L, L)$$

   c) "Random": a random sequence of $+1's$ and $-1's$ using any function of choice except *pngen.m*. Make sure to generate a sequence of length L.

3. To observe the auto-correlation (or the cross-correlation of a sequence $x[n]$ with itself, since in this case, $x[n] = y[n]$) of each sequence $x[n]$, we use the built-in MATLAB function *xcorr(.)* For example the autocorrelation function R_{xx} can be written as

$$R_{xx} = \text{xcorr}(x, x)$$

   The file "c_corr.m" provided on the shared drive implements this function. We'll denote the index that maximizes the correlation R_{xx} (or equivalently defines the peak) as $m_{\max} = L$.

   Note: Ideally, for "good" sequences, this "peak" should occur at index $m = L$. However, in the presence of a significant amount of noise, this may not be the case.

   To compare the performance between the three sequences generated in part 2, we will use the **detection error** defined as as

$$\text{Detection Error Rate} = \frac{\#\text{ of times the peak occurs at } m \neq L}{\#\text{ of trials or realizations}}$$

4. For each of the three signal sequences $x[n]$ of length L = 128, form the cross-correlation $R_{xy}(m)$ for 1000 realizations of each noise sequence $w(n)$ using the same value of σ = 0.5.

$$y[n] = x[n] + w(n)$$

$$w = \sigma * \text{randn}(1,L)$$

$$R_{xy} = \text{xcorr}(x, y)$$

For each noise realization, determine the value m for which $R_{xy}(m)$ is maximum. Some realizations may not produce a maximum at the anticipated location. This produces a sequence detection error. The rate of detection errors is the total number of errors divided by the total number of realizations.

For each signal sequence, calculate the sequence detection error rate of each of the three values of the noise parameter σ = 0.5, 1.0, 2.0 . This gives a total of nine values (i.e. three sequences for three values of σ).

Which sequence(s) performs the best under additive white gaussian noise (AWGN). Why?

```
realization = 1000;
L_s = [128; 256; 512];
sigmas = [0.5, 1.0, 2.0];

error_rates = zeros(3, 3);

x_ones = ones(1,L_s(1));
x_psr = pngen(log2(L_s(1)),L_s(1));
x_randm = sign(randn(1, L_s(1)));


% functions
for i=1:length(sigmas)

    sigma = sigmas(i);

   for seq = 1:3
    signal = '';
    switch seq
        case 1
            signal = x_ones;
        case 2
            signal = x_psr;
        case 3
            signal = x_randm;
    end


    error_count = 0;

    for j=1:realization

        w_n = sigmas(i) * rand(1,L_s(1));

        y_n = signal + w_n;
        R_xy = c_corr(signal,y_n);
        [~,max_Rxy] = max(R_xy);

```

```
38
39
40              if max_Rxy ~= L_s(1)
41                  error_count = error_count + 1;
42              end
43          end
44
45
46          det_err_rate = error_count / realization;
47          error_rates(seq, i) = det_err_rate;
48        end
49
50    end
51
52    % Display results
53    disp('Sequence Detection Error Rates:');
54    disp(error_rates);
55
```

```
>> Lab_2
Sequence Detection Error Rates:
    {'Sigma     '}    {'All ones'}    {'Pseudo-Random'}    {'Random'}

    0.5000         0         0         0
    1.0000         0         0         0
    2.0000         0         0         0
```

After running this a couple of times, I still get 0s for detection error rates.

5. Now, vary the length L = 256, 512 of the sequence and determine how the length of the sequence affects the sequence detection error rate. Assuming that you construct a packet using a preamble x[n] of length L followed by a payload p[n] of length L_p, what does this tell you about the trade-off between detection error rate of preamble sequences x[n] and Overhead?

$$\text{Overhead} \doteq \frac{L}{L_p + L}$$

```
realization = 1000;
L_s = [128; 256; 512];
sigmas = [0.5, 1.0, 2.0];

error_rates = zeros(length(L_s), length(sigmas), 3); % Matrix to store error rates


signal_name = {'All ones', 'Pseudo-Random', 'Random'};


for i=1:length(sigmas)

    sigma = sigmas(i);

   for seq_lgnth = 1:length(L_s)

       L = L_s(seq_lgnth);


       for seq = 1:3
            signal = '';
            switch seq
                case 1
                    signal = ones(1,L);
                case 2
                    signal = pngen(log2(L),L);
                case 3
                    signal = sign(randn(1, L));
            end


            error_count = 0;

            for j=1:realization

                w_n = sigma * randn(1,L);

               y_n = signal + w_n;
               R_xy = c_corr(signal,y_n);
               [~,max_Rxy] = max(R_xy);


               if max_Rxy ~= L
                   error_count = error_count + 1;
               end
           end


           det_err_rate = error_count / realization;
           error_rates(seq_lgnth, i, seq) = det_err_rate;
       end
   end
end

disp('Sequence Detection Error Rates for Different Sequence Lengths and Sigma Values:');
disp('Dimensions: [Sequence Lengths, Sigma Values, Signal Sequences]');
disp(error_rates);
```

```
>> part4
Sequence Detection Error Rates for Different Sequence Lengths and Sigma Values:
Dimensions: [Sequence Lengths, Sigma Values, Signal Sequences]

(:,:,1) =

    0.0210    0.1940    0.4580
    0.0200    0.1840    0.4720
    0.0260    0.1760    0.4880


(:,:,2) =

         0         0    0.2120
         0         0    0.0340
         0         0         0


(:,:,3) =

   1.0e-03 *

         0         0    1.0000
         0         0         0
         0         0         0
```

Each matrix represents the signals: ones, pseudorandom and random of my choice

The length of the preamble, L, and payload length, Lp,can improve robustness in synchronization and reduce detection error rate, but the cost of overhead is increased as more resources are given to transmit the preamble. The trade-off is between better synchronization or lower detection error rates with longer preamble and reducing overhead to maximize efficiency of data transmission.

Additional questions for post lab:

1. Derive a general formula for the spectral line spacings and null spacings as a function of the f_s, *N*, and *M* where:

a) Sampling Rate f_s

b) Samples Per Symbol *N*

c) Polynomial Order *M*

Line spacing = N/fs * M

Null to null spacing = fs/N * 1/M

OR

Spectral line spacing = 1/N

Null to null spacing = 1/N * (2^M - 1)

To derive a general formula for this, the spectral line is related to the sampling rate: $f_{line} = \frac{f_s}{N}$.

As for the null spacing, the nulls happen at intervals of $\frac{f_s}{2^M}$.

$f_{null} = \frac{f_s}{2^M}$.

2. PRBS sequences are often used in packet headers to establish frame synchronization. Why do you think this is the case?

Although PRBS, as implied by its name, has pseudo random behavior or appears random, its structure is known and is generated deterministically. Because its behavior and structure is known it is reliable for detection of synchronization.