Lab 3 - Bit Error Rate and Parameter Estimation for a Basic Modem

Goal

Operation of a matched filter with clock synchronization. Carrier phase estimation of passband signals using feedback demodulation based on phase-locked loops.

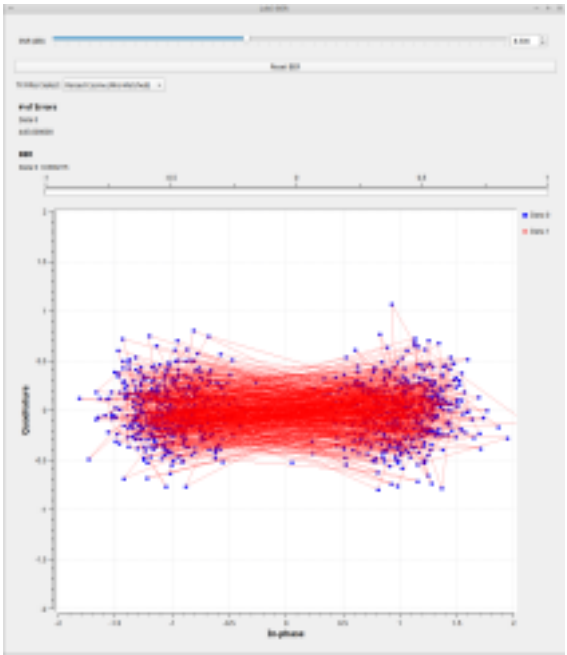
1 Part 1 - BER vs. E_b/N_0 for BPSK

Generation

We will measure the BER of BPSK generating a data stream on one Pluto and demodulating the received signal on another Pluto. Both the clock phase and carrier phase will be estimated and used to measure the effect of several system parameters on the BER.

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. Connect **both** Plutos and be sure they are mounted on the system. (You should not have to set the addresses.) Open the front panel of course*/*grc*/*lab3*/*lab3_BER.py. The front panel should look like this



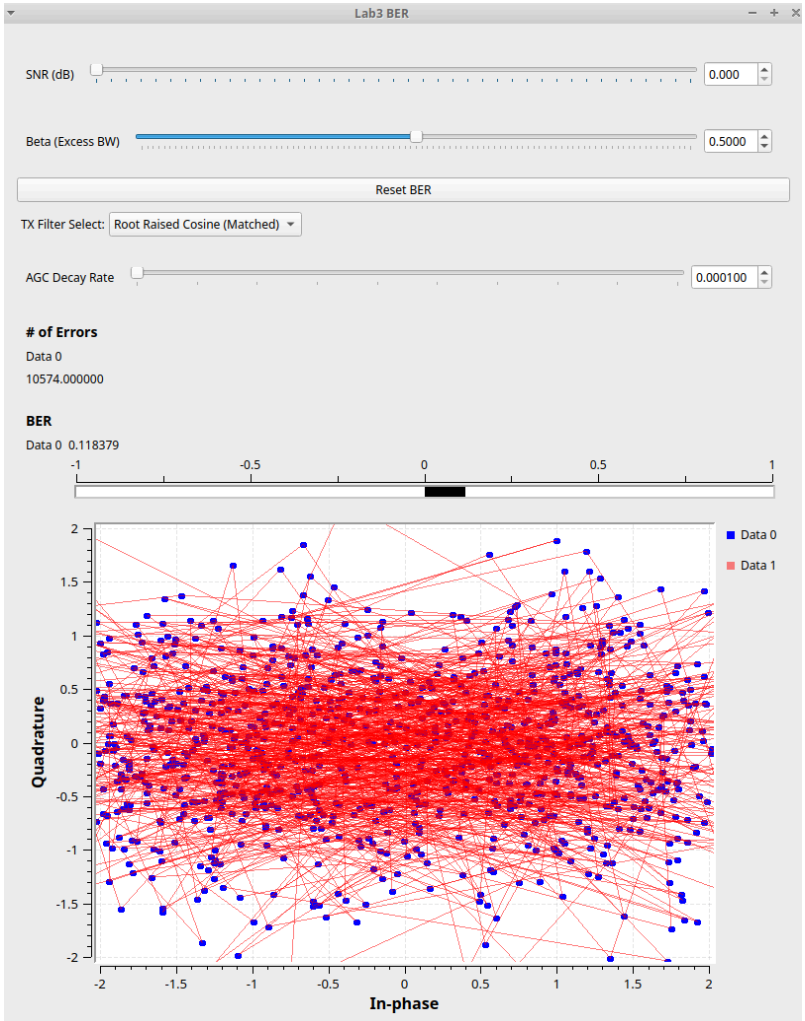
3. For this lab, the data is generated on one Pluto with pulses set by the Tx filter select. For this part of the lab, the Tx pulse shape is always set to a root-raised cosine pulse. This Tx filter uses the square root of a raised cosine pulse in the frequency domain with a filter parameter β = 0.5. The second Pluto receives the waveform and the detection filter is controlled through the front panel. To start, choose "Root Raised Cosine (Matched)", which is the same pulse shape at the Tx pulse. Referring to Chapter 3 of Papen/Blahut (Figure 3.12 and Eq. (3.4.7), let $q(t)$ represent 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

$$q(t) = x(t) \circledast h(t) \circledast y(t). \quad (1.1)$$

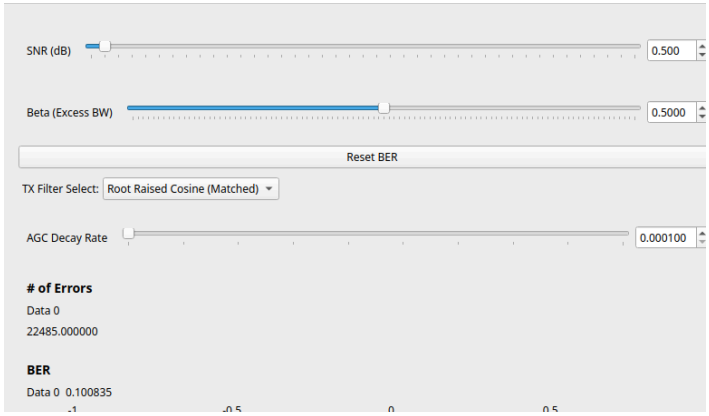
The desired impulse response $q(t)$ at the location where the waveform is sampled is called the *target pulse*. The goal is to make this pulse a Nyquist pulse using a detection filter that is a matched filter. For this lab $h(t)$ is an impulse and for this first part, **both** the Tx filter $x(t)$ and Rx filter $y(t)$ are root-raised cosines so that $q(t)$ is a raised cosine pulse.

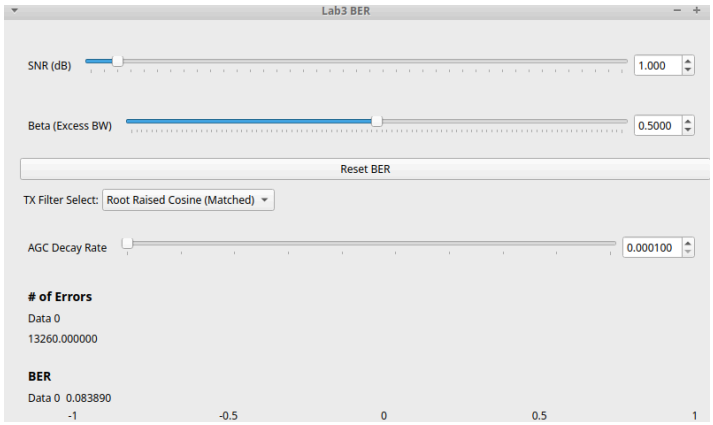
4. On the top of the front panel set the SNR, which is equal to E_b/N_0 to 0 dB.

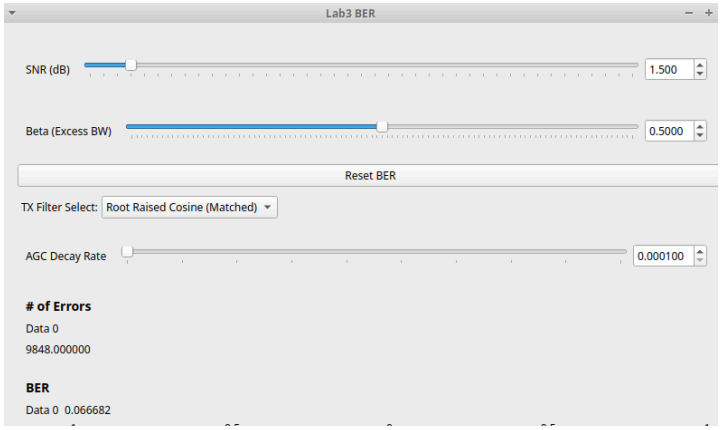
a) Reset the BER and collect <u>at least</u> 1000 errors. Record the SNR and the BER.

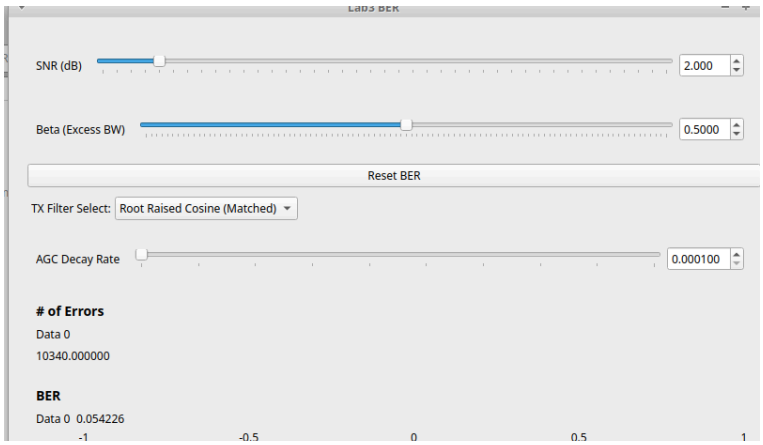


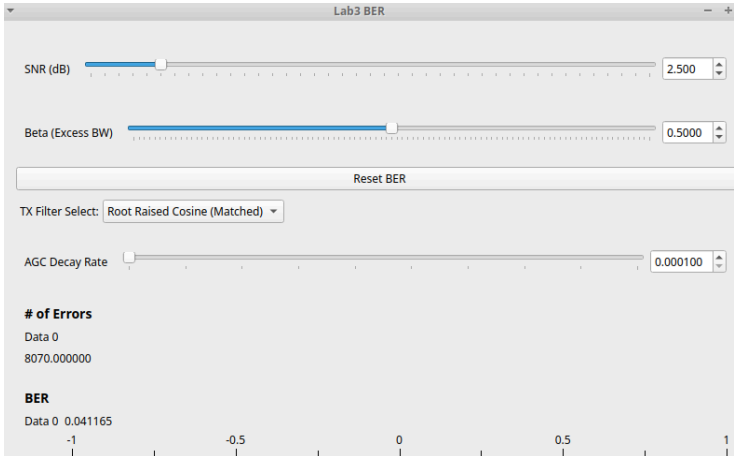
b) Increase the SNR in 0.5 dB steps and repeat up to an SNR of 8 dB still collecting at least 1000 errors. Remember to reset the BER for each measurement. (The last few measurements will take several minutes to generate 1000 errors.)

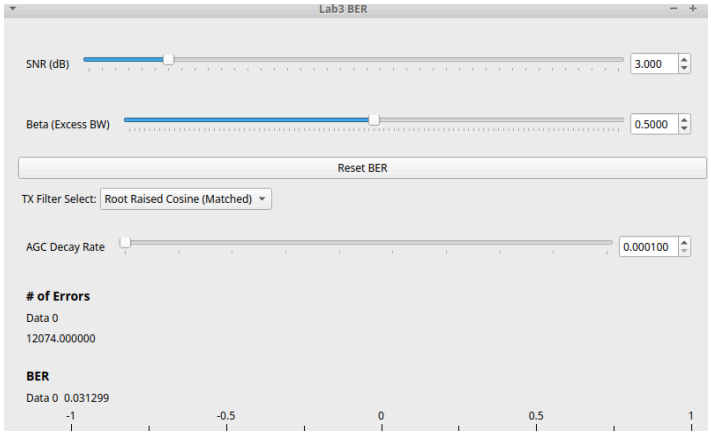


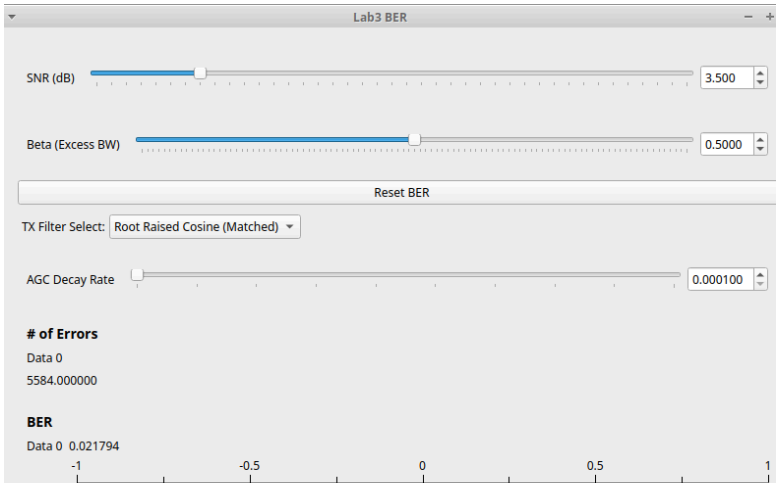


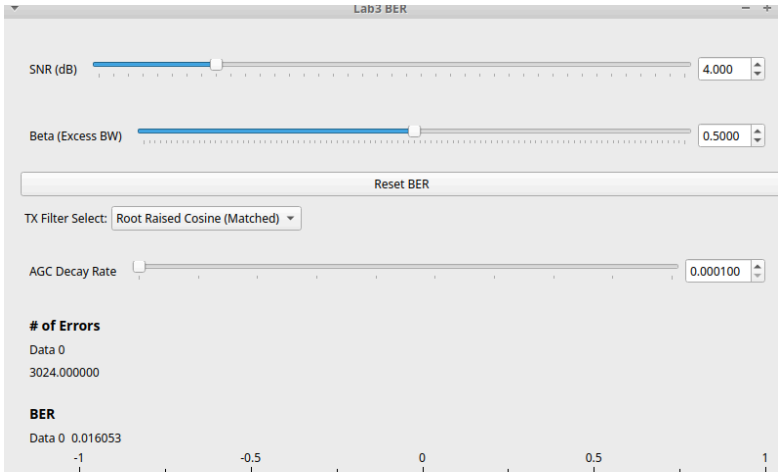


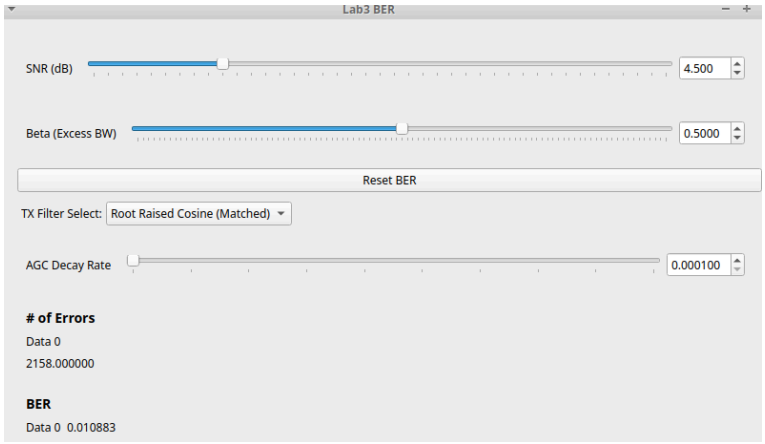

SNR (dB)
2.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
10340.000000
BER
Data 0 0.054226
-1
-0.5
0
0.5
1

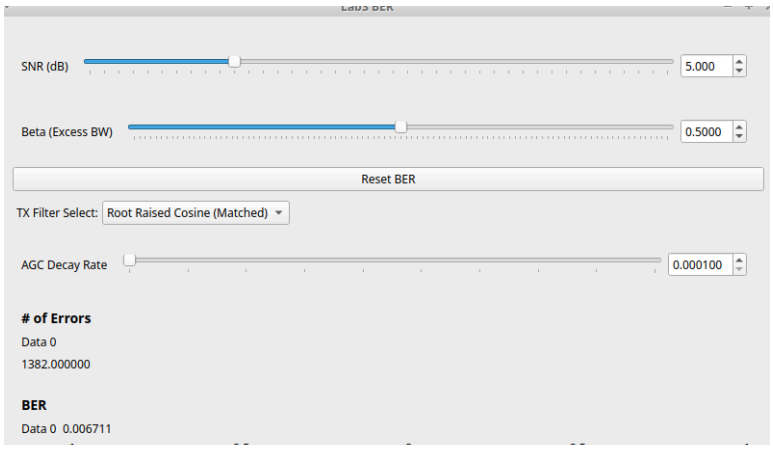

Lab3 BER
SNR (dB)
2.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
8070.000000
BER
Data 0 0.041165
-1
-0.5
0
0.5
1

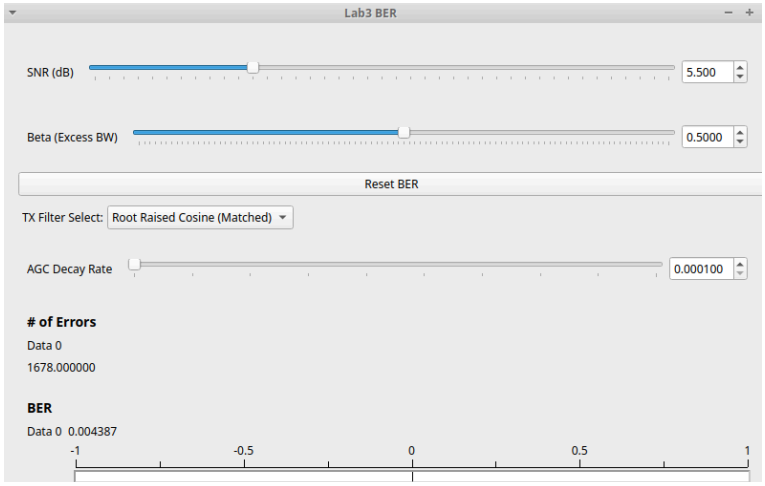

Lab3 BER
SNR (dB)
3.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
12074.000000
BER
Data 0 0.031299
-1
-0.5
0
0.5
1

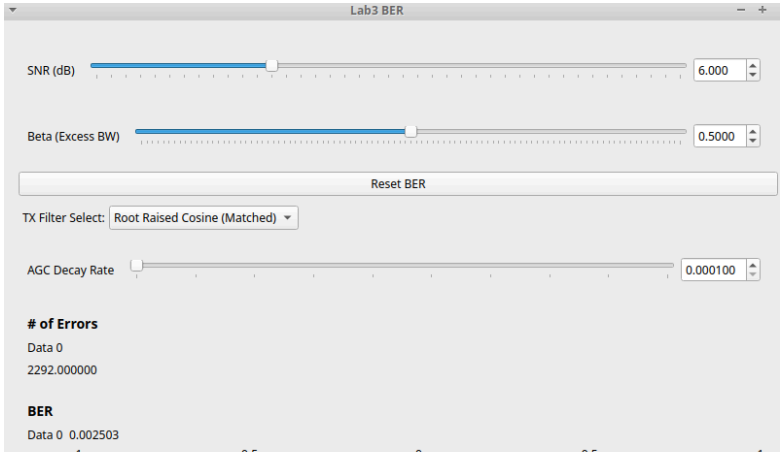

Lab3 BER
SNR (dB)
3.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
5584.000000
BER
Data 0 0.021794
-1
-0.5
0
0.5
1

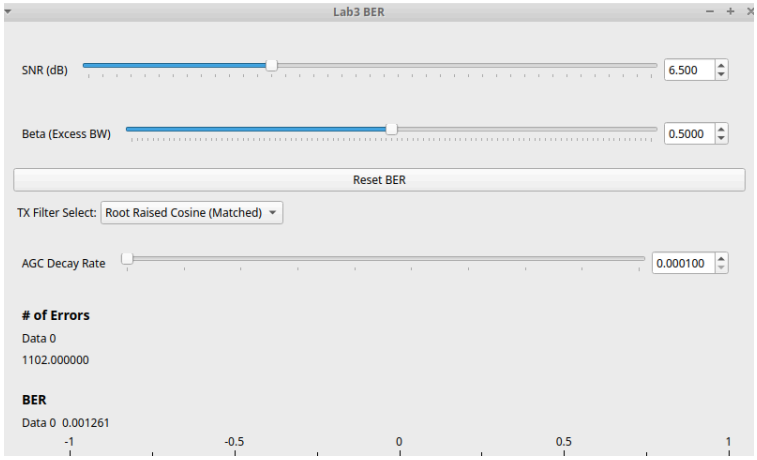

Lab3 BER
SNR (dB)
4.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
3024.000000
BER
Data 0 0.016053
-1
-0.5
0
0.5
1

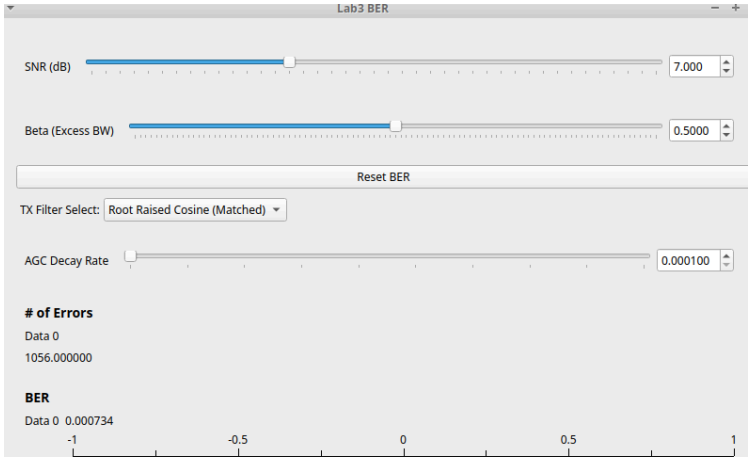

Lab3 BER
SNR (dB)
4.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
2158.000000
BER
Data 0 0.010883

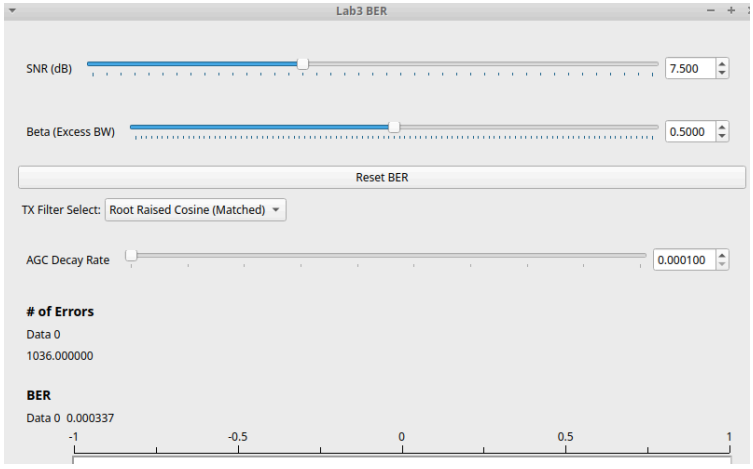

SNR (dB)
5.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1382.000000
BER
Data 0 0.006711

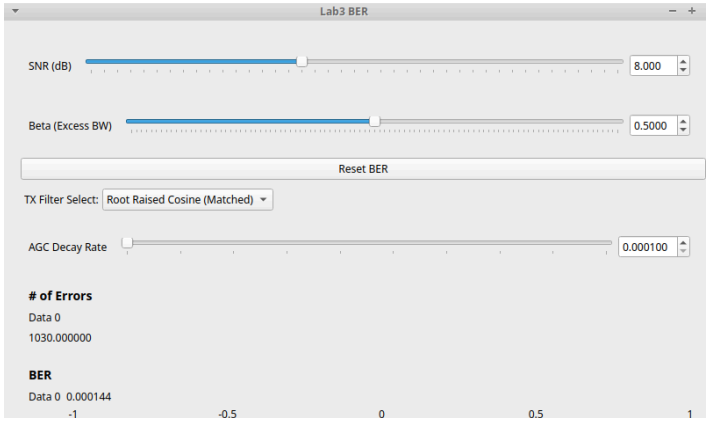

Lab3 BER
SNR (dB)
5.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1678.000000
BER
Data 0 0.004387
-1
-0.5
0
0.5
1


Lab3 BER
SNR (dB)
6.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
2292.000000
BER
Data 0 0.002503

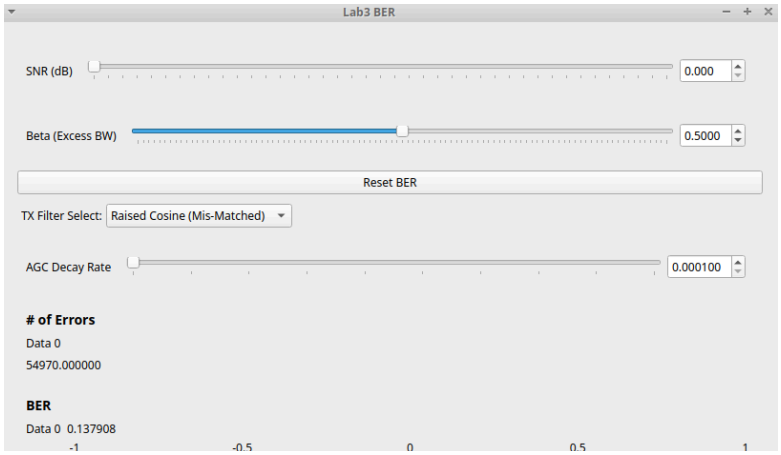

Lab3 BER
SNR (dB)
6.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1102.000000
BER
Data 0 0.001261
-1
-0.5
0
0.5
1

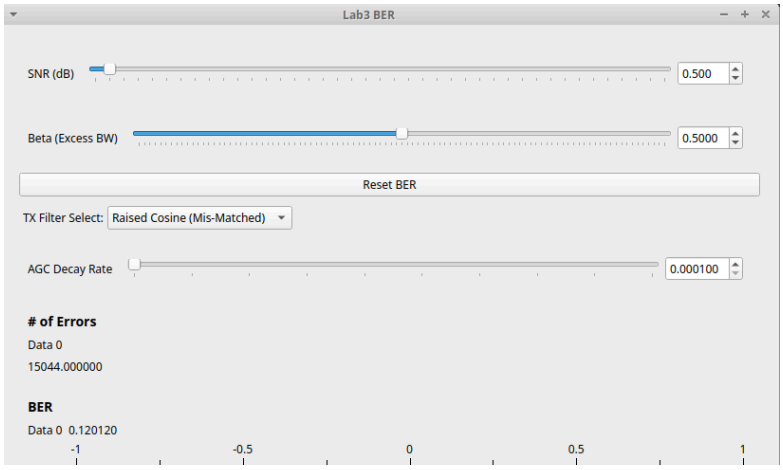

Lab3 BER
SNR (dB)
7.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1056.000000
BER
Data 0 0.000734
-1
-0.5
0
0.5
1

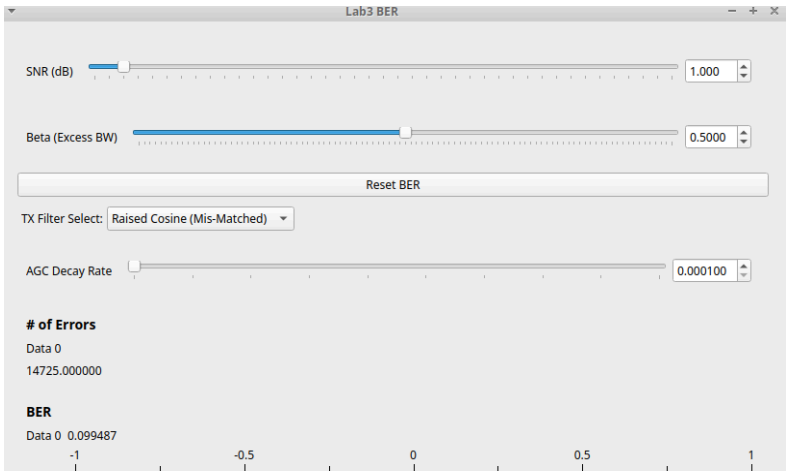

Lab3 BER
SNR (dB)
7.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Root Raised Cosine (Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1036.000000
BER
Data 0 0.000337
-1
-0.5
0
0.5
1

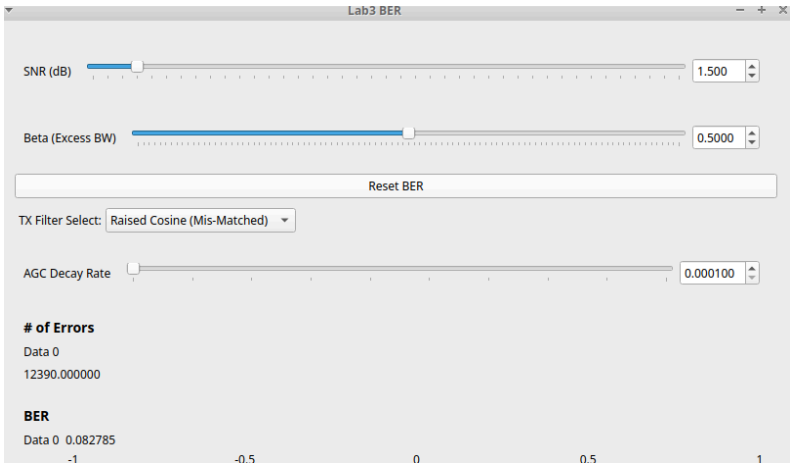


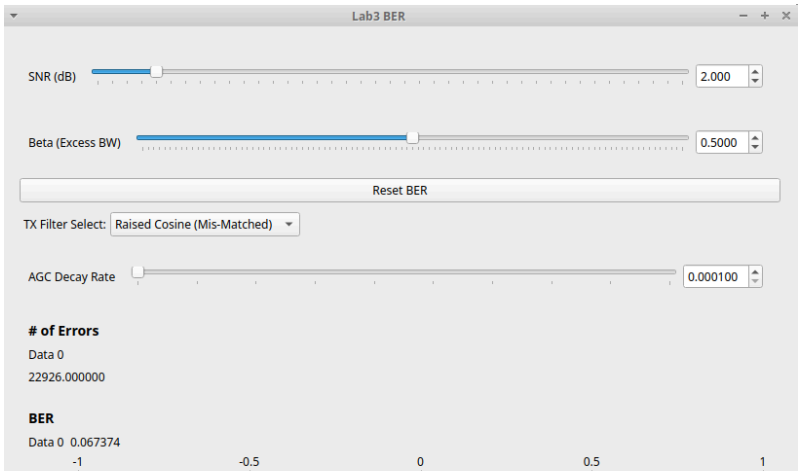
c) Repeat part (b) for the mismatched filter for SNR values up to 9 dB. This filter is a rectangular detection filter.

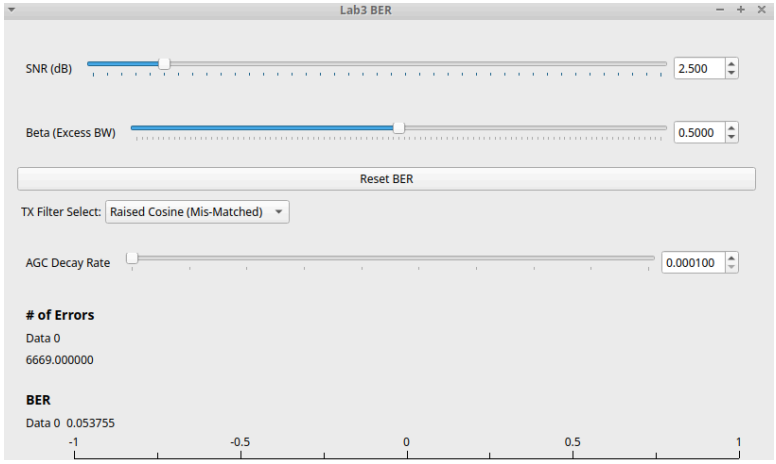


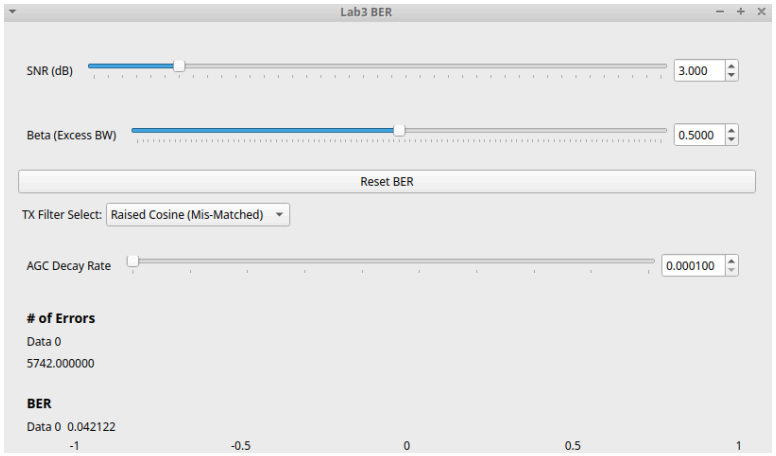


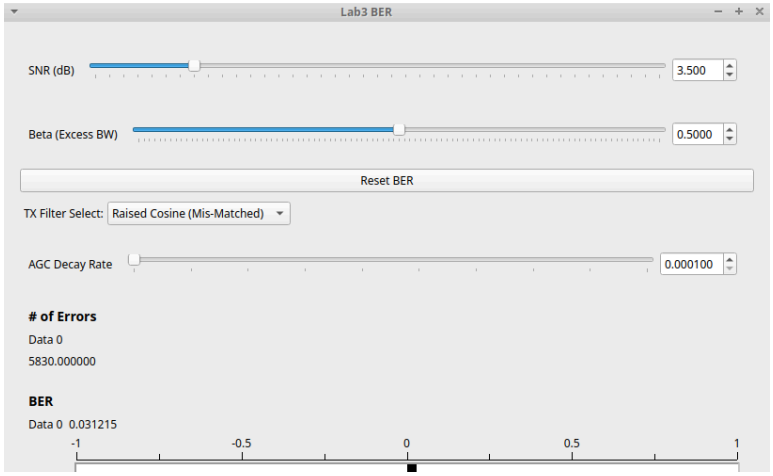

Lab3 BER
SNR (dB)
1.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
14725.000000
BER
Data 0 0.099487
-1
-0.5
0
0.5
1

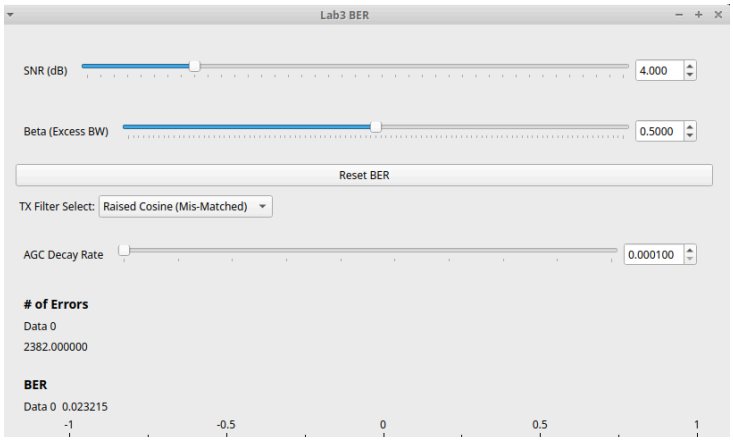

Lab3 BER
SNR (dB)
1.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
12390.000000
BER
Data 0 0.082785
-1
-0.5
0
0.5
1

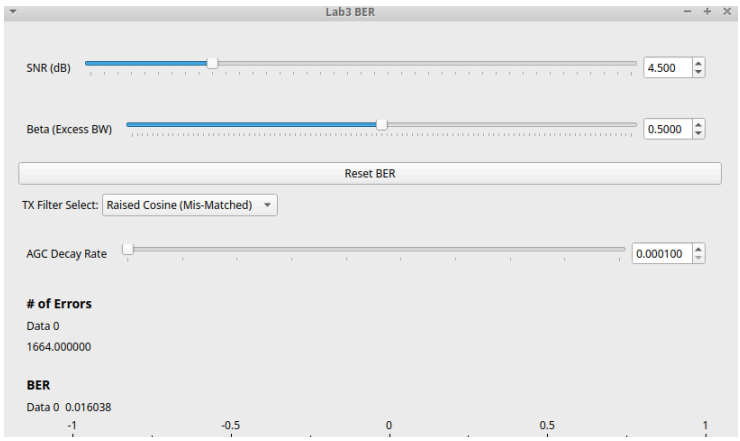

Lab3 BER
SNR (dB)
2.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
22926.000000
BER
Data 0 0.067374
-1
-0.5
0
0.5
1

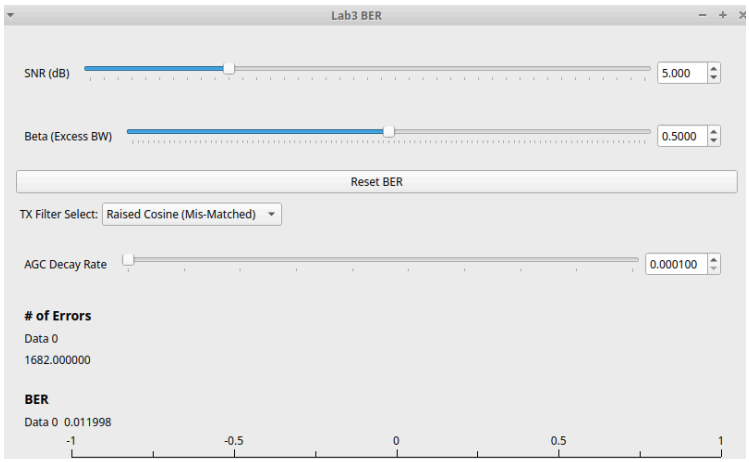

Lab3 BER
SNR (dB)
2.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
6669.000000
BER
Data 0 0.053755
-1
-0.5
0
0.5
1

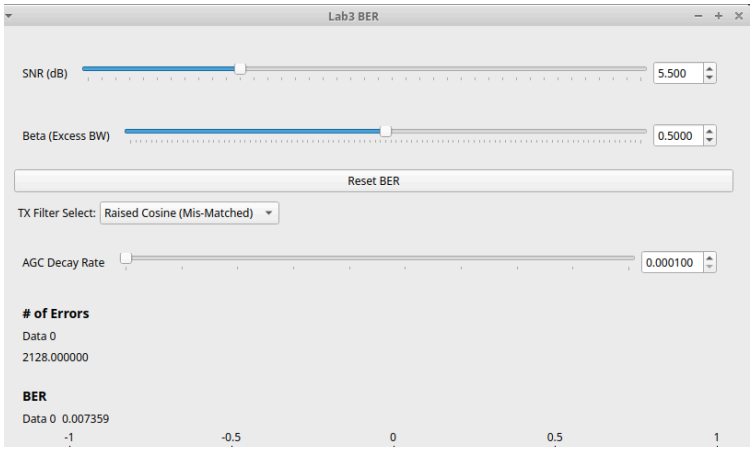

Lab3 BER
SNR (dB)
3.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
5742.000000
BER
Data 0 0.042122
-1
-0.5
0
0.5
1

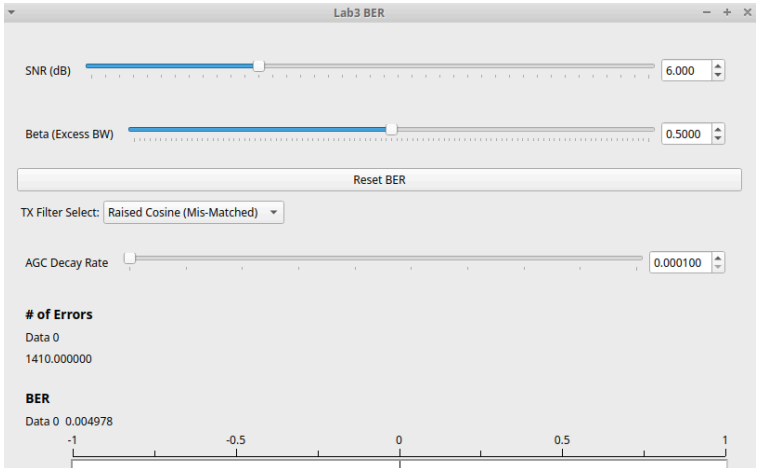

Lab3 BER
SNR (dB)
3.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
5830.000000
BER
Data 0 0.031215
-1
-0.5
0
0.5
1

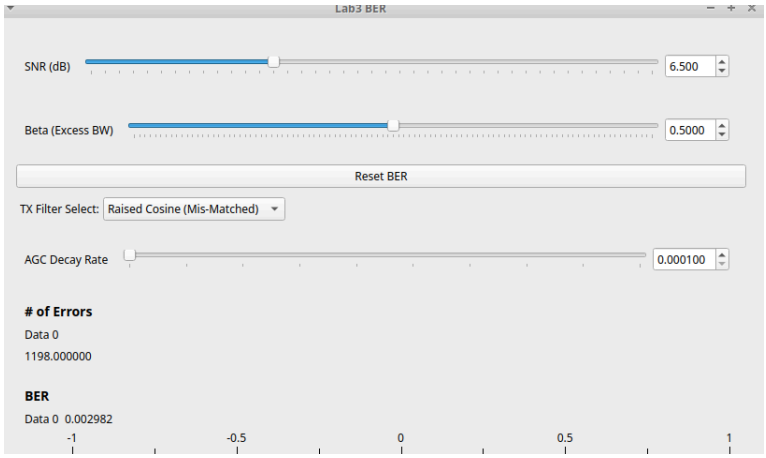

Lab3 BER
SNR (dB)
4.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
2382.000000
BER
Data 0 0.023215
-1
-0.5
0
0.5
1

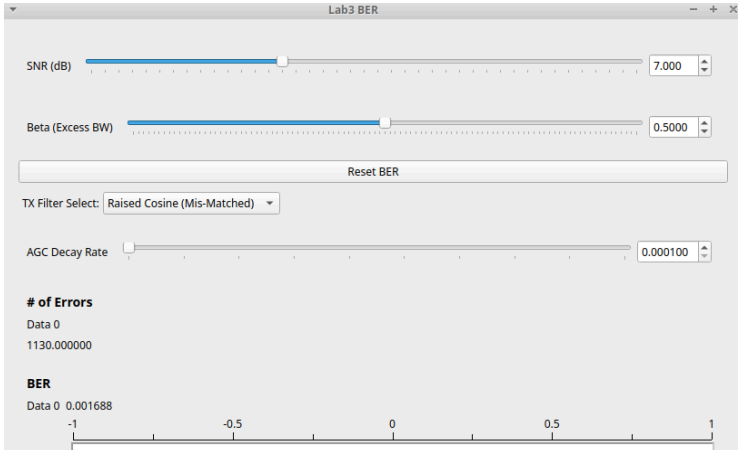

Lab3 BER
SNR (dB)
4.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1664.000000
BER
Data 0 0.016038
-1
-0.5
0
0.5
1

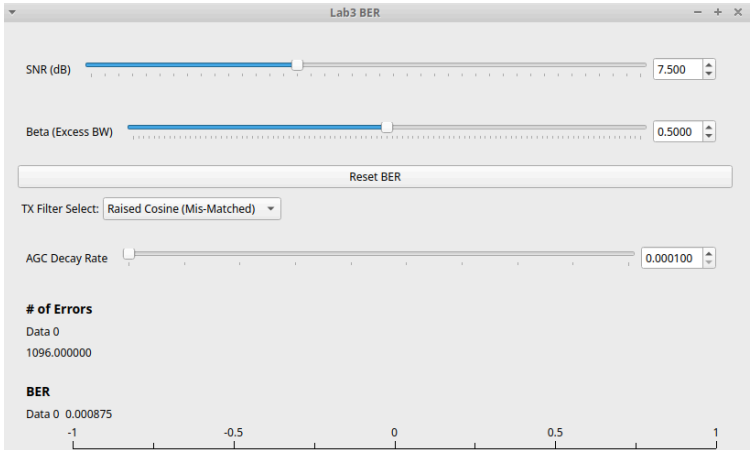

Lab3 BER
SNR (dB)
5.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1682.000000
BER
Data 0 0.011998
-1
-0.5
0
0.5
1

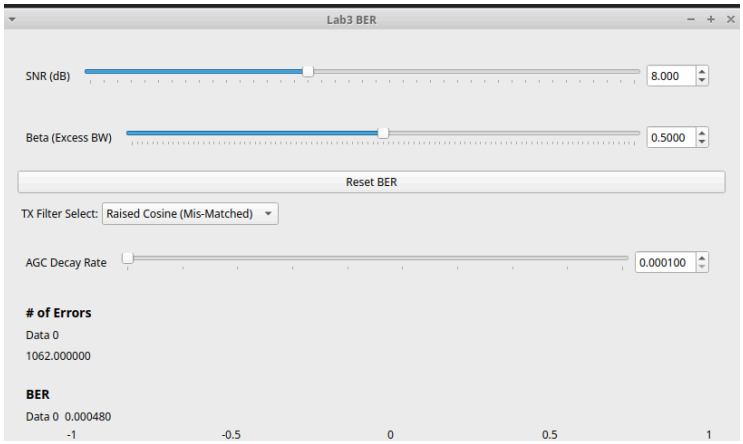

Lab3 BER
SNR (dB)
5.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
2128.000000
BER
Data 0 0.007359
-1
-0.5
0
0.5
1

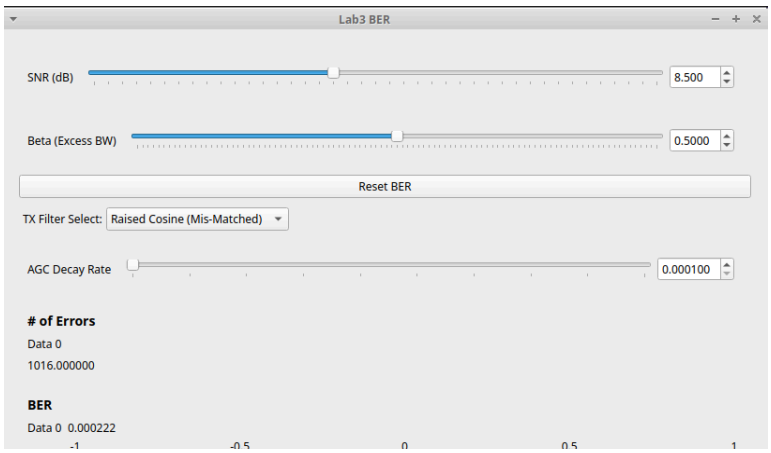

Lab3 BER
SNR (dB)
6.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1410.000000
BER
Data 0 0.004978
-1
-0.5
0
0.5
1

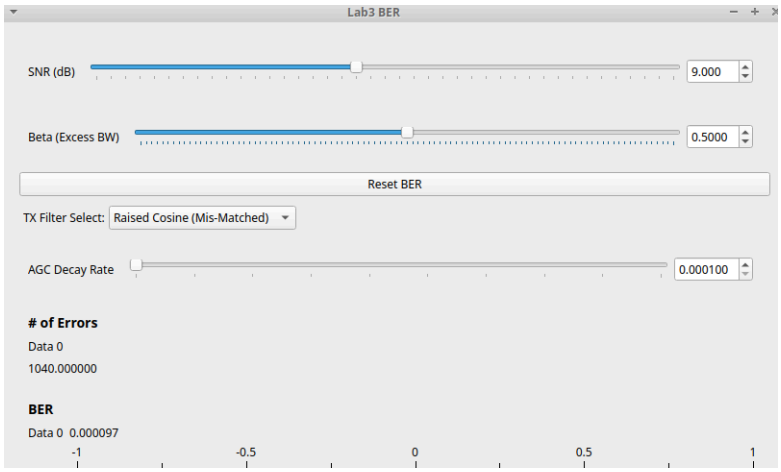

Lab3 BER
SNR (dB)
6.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1198.000000
BER
Data 0 0.002982
-1
-0.5
0
0.5
1


Lab3 BER
SNR (dB)
7.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1130.000000
BER
Data 0 0.001688
-1
-0.5
0
0.5
1


Lab3 BER
SNR (dB)
7.500
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1096.000000
BER
Data 0 0.000875
-1
-0.5
0
0.5
1

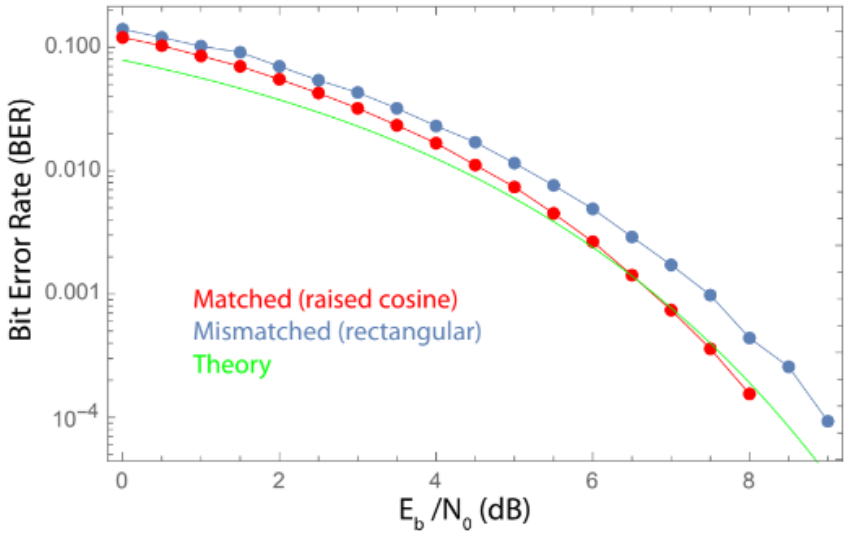

Lab3 BER
SNR (dB)
8.000
Beta (Excess BW)
0.5000
Reset BER
TX Filter Select:
Raised Cosine (Mis-Matched)
AGC Decay Rate
0.000100
# of Errors
Data 0
1062.000000
BER
Data 0 0.000480
-1
-0.5
0
0.5
1

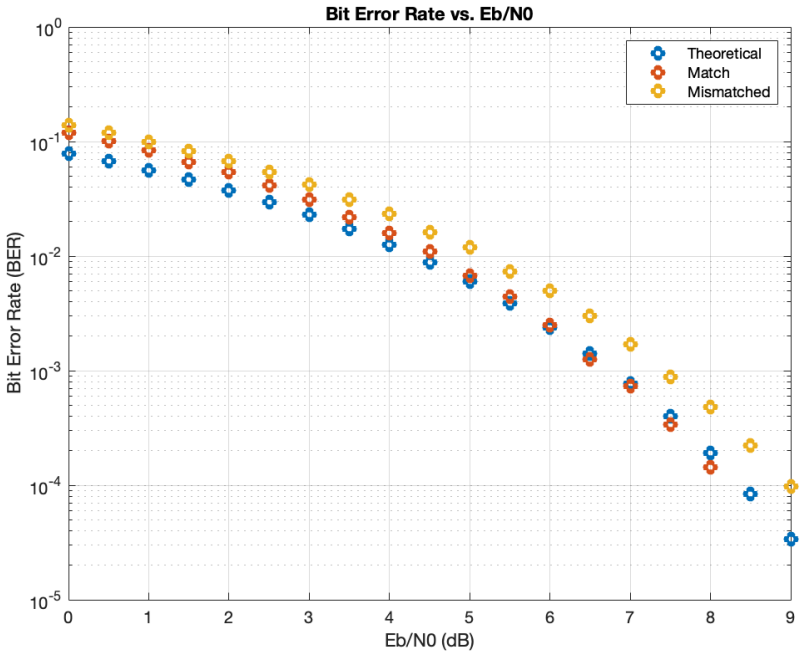




Write-Up for Part 1

1. On the same graph, plot the log of the BER vs the SNR in dB for both the matched filter and the mismatched filter. Also plot the theoretical curve. The final plot should look similar to the following figure, which was generated with real data collected from a set of Plutos.





2. Using the results of Theory Question 6 from Prelab 2 and assuming that the underlying distribution that is sampled to determine the BER is gaussian, determine the statistical confidence of each of the measurements. This is expressed as the root-mean-squared value σ of the estimate of the mean error rate.

Using the expression given in Prelab 2, with Accuracy = $100(1 - \frac{1}{\sqrt{N}})$, collecting a minimum of 1000 errors will give a statistical confidence of 96.84%. This means that each one of our measurements has at least a 96.84% chance of being a value in the expected range. This accuracy can be seen from our generated plots, as both the matched and mismatched BER curves closely follow the shape of the theoretical curve.

3. Quantitatively (i.e. provide a formula) that explains the shift of the curves with respect to one another. What is the approximate additional E_b/N_0 (dB) required for the mis-matched filter to achieve the same performance as matched filter? This quantity is called the *power penalty*.

The Power Penalty is approximately 1 dB (1.25 in linear scale), as the mismatched filter requires about an extra dB throughout all SNRs to achieve the same BER as the matched filter. By approximating the matched BER to the theoretical BER, the relation between the two curves can be expressed with the expression below.

$$BER_{Mismatched} = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0} - 1.25}\right) = Q\left(\sqrt{2\left(\frac{E_b}{N_0} - 1.25\right)}\right)$$

4. Compare the direct BER measurements for β = 0.5 and estimated BER for β = 0.5 from the histogram method used in Lab 2 where the sample time for the Lab 2 data is chosen to produce the minimum BER.

   a) Are the results in this section consistent with the estimate of the BER derived from the eye-pattern histograms measured in Lab 2?

   The results in this section aren't consistent with the estimated BER from the histograms in lab 2.

   b) Discuss the accuracy of estimating the BER using the histograms vs. the direct measurement of the BER. When does the histogram method produce a reasonable estimate.? When does it not?

   In practice, histogram-based methods can be a useful tool for quick and preliminary BER estimation, but for precise and reliable results, especially in critical applications, direct BER measurement remains the preferred approach. As we can see in our results, the histogram method got more accurate as the SNR increased, which indicates that this is preferable for large SNR values. However, the histogram method does not produce reliable estimates when we are at low SNR. I would say that the histogram estimates are close to the theoretical values from lab 2 which indicates that it may be "good enough" for simple modulated schemes like BSPKs and QPSKs.

2 Channel Estimation using a PN Sequence

For this part of the lab, we will simulate a “two-path” or “single echo” multipath channel by summing a modulated signal with a delayed copy of itself. The received signal $x(t)$ is then given as

$$x(t) = s(t) + \alpha_k s(t - T)$$

$$|\alpha_k| = A < 1$$

where T is the delay and A is the magnitude strength of the delay component relative to the “line-of-site” component. This received signal is equivalent to generating a signal $s(t)$ and passing it through a “channel” or filter $h_c(t)$ with impulse response given below:

$$h_c(t) = \delta(t) + A\delta(t - T)$$

$$x(t) = \int_\tau h_c(\tau) s(t - \tau) d\tau$$

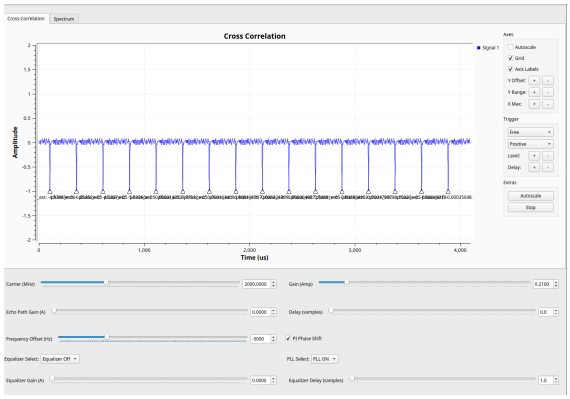
In lab, we will estimate the impulse response of the channel $h_c(t)$ and attempt to undo the distortion caused by fading. One method of doing this is by creating the channel estimate $\hat{h}_c(t)$ and convolving the signal $x(t)$ by the inverse of the channel impulse response $g(t) = \hat{h}_c^{-1}(t)$. Under the condition that the channel response is estimated perfectly (i.e. $\hat{h}_c(t) = h_c(t)$), we have $\int h_c(\tau) g(t - \tau) = \delta(t)$. The process of undoing the effects of the channel is called “equalization”.

1. Clone the latest version of the labs on your system. (See Lab 1 for details.)
2. Open the front panel of course/grc/lab4/lab4.py and locate the front panel controls labeled “Echo Path Gain " and “Delays”. The Echo Path Gain controls the value of A for the delayed signal The delay controls the amount of value of T. Because we are using a sampling rate of 1 MSample/sec, each sample corresponds to 1 μs of delay. Using these two controls specifies the fading profile of your transmitted signal.

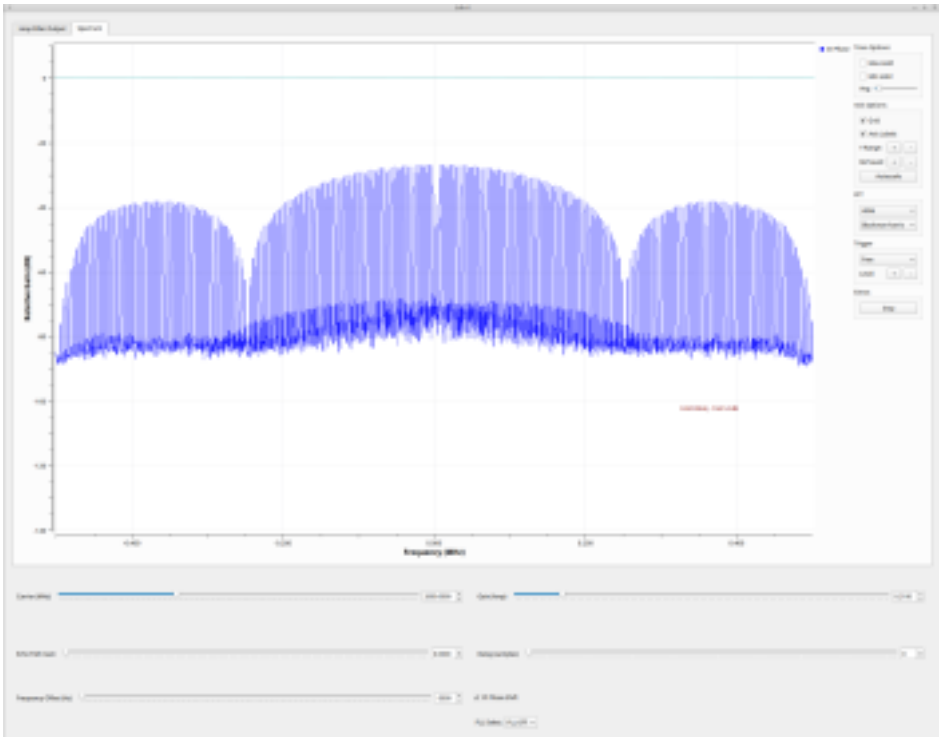
3. Set your fading profile to the following:

   a) Echo Path Gain = [0]

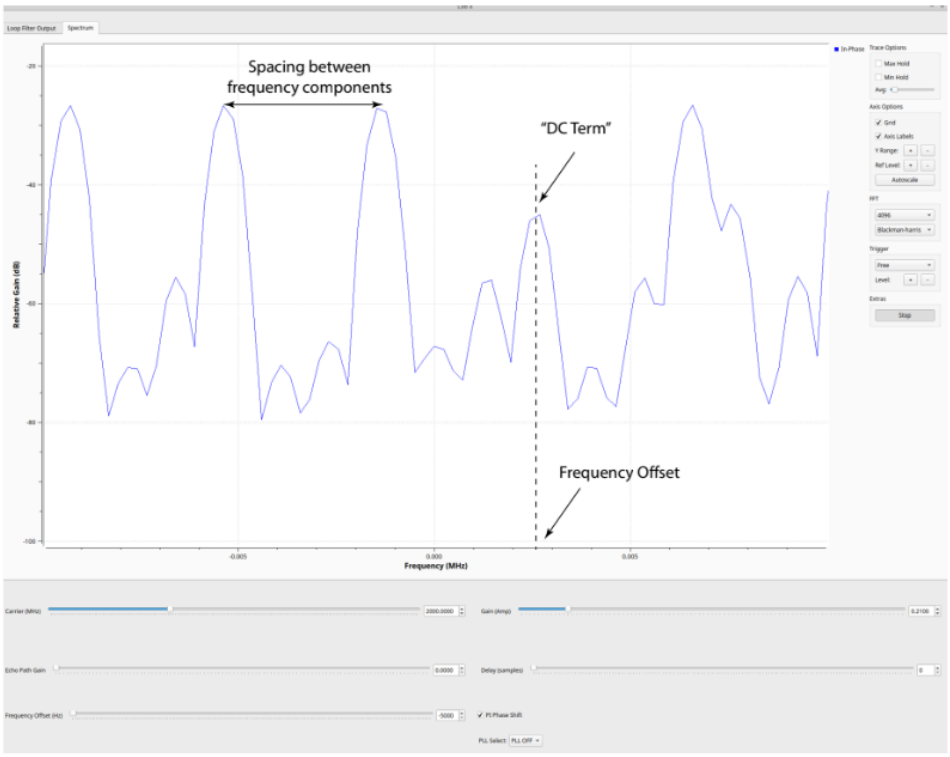
   b) Delays = [0]



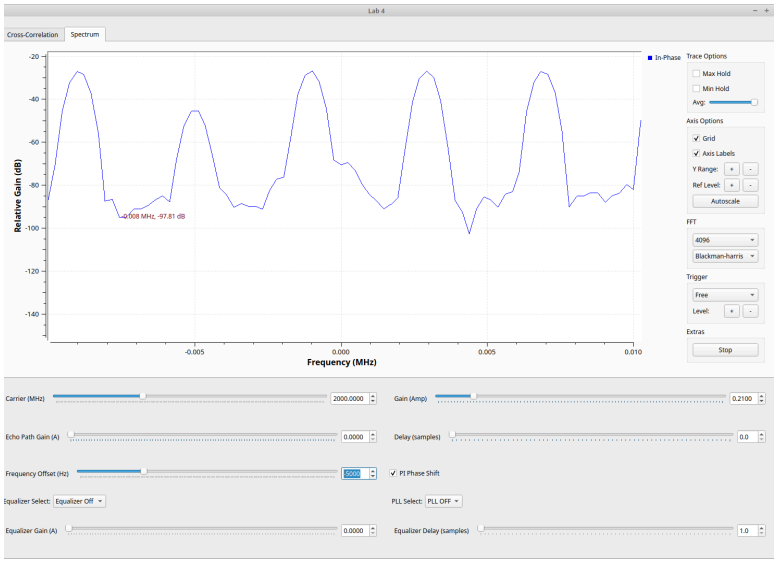
4. Set the tab at the top of the control panel to “Spectrum”. You should see something like the following figure.



5. Disable the PLL control at the bottom of the front panel and then using the cursor, zoom the frequency range of the power spectrum graph to the approximate interval $F \in [-10kHz, 10kHz]$ so that you can see the DC component. Refer to the screenshot below.



6. Measure the spectra-line spacing of the sine-spectrum corresponding to your transmitted periodic training sequence (PN sequence). Refer to the screenshot above.



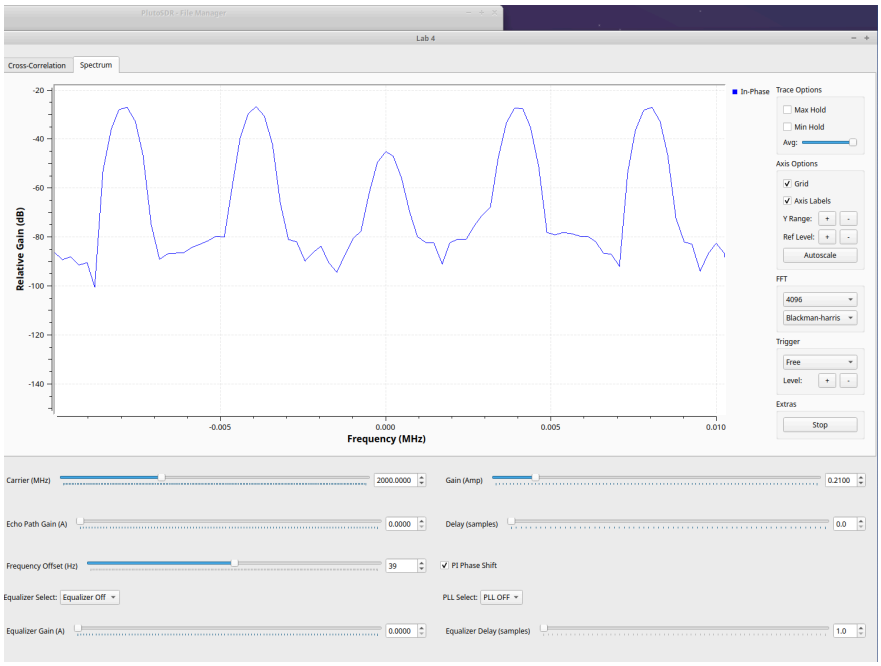
The Spectral line spacing is about 0.004MHz, or 4kHz.

7. Using mouse-over function, measure the frequency at which the DC component (i.e. the component corresponding to the A_0 term of the fourier series) of the sinc spectrum appears. Note, this value will not occur at $f = 0Hz$. Explain why.

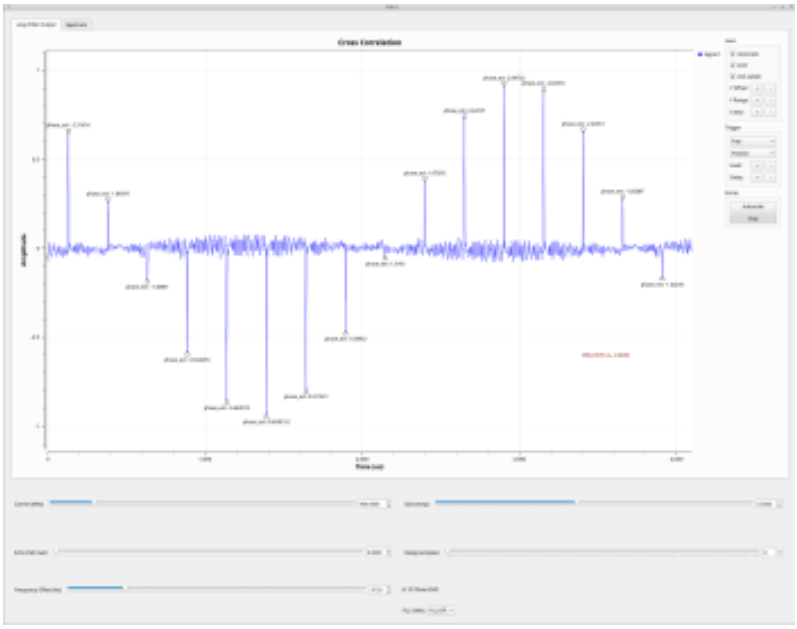
DC @ frequency offset: -45.93dB

Based on the generated sinc spectrum, we detected the DC component to be located at a frequency offset of -0.005MHz. The DC component did not occur exactly at f = 0Hz due to effects caused by the randomness of the PN sequence. These can include slight offset caused during the process of signal transmission, the signal being passed through the channel, or when the transmitted signal is received, which can lead to a slight offset like the one we observed in the sinc spectrum above.

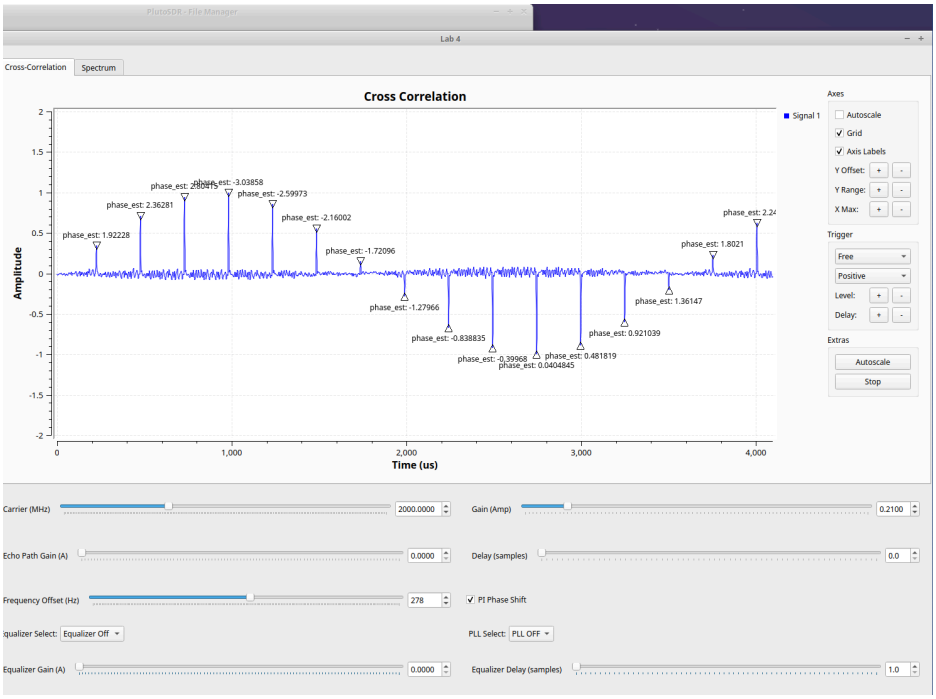
8. Now using the “Frequency Offset” slider, adjust the DC component so that is close to zero.

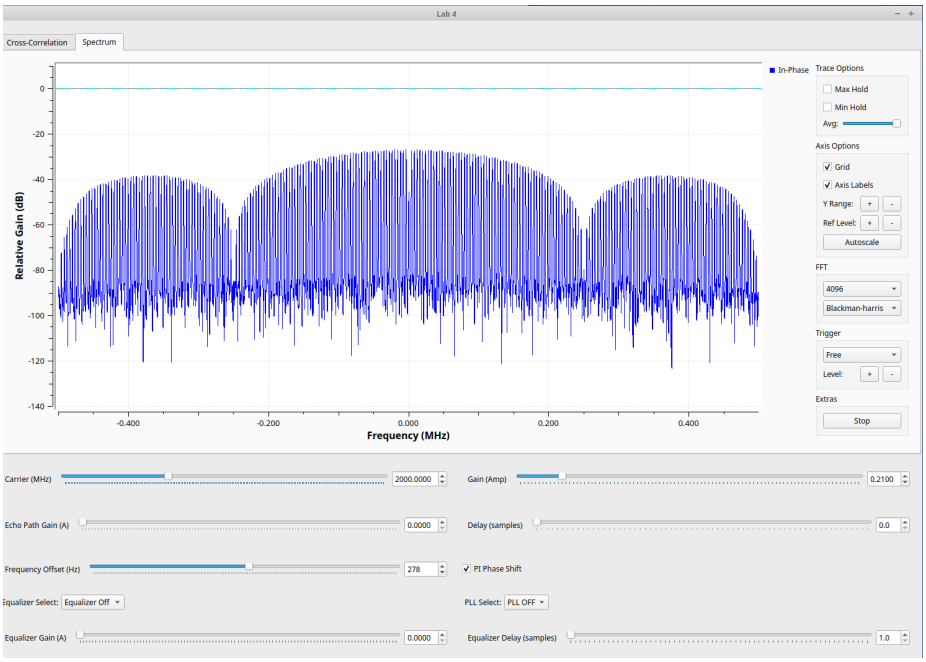


9. Switch to the “Cross-Correlation” tab and adjust the “Frequency Offset” slider to produce one period of the “envelope” of the AC. It should look like the following screenshot for which each spike is a single cross-correlation of the reference PN sequence with the incoming sequence.



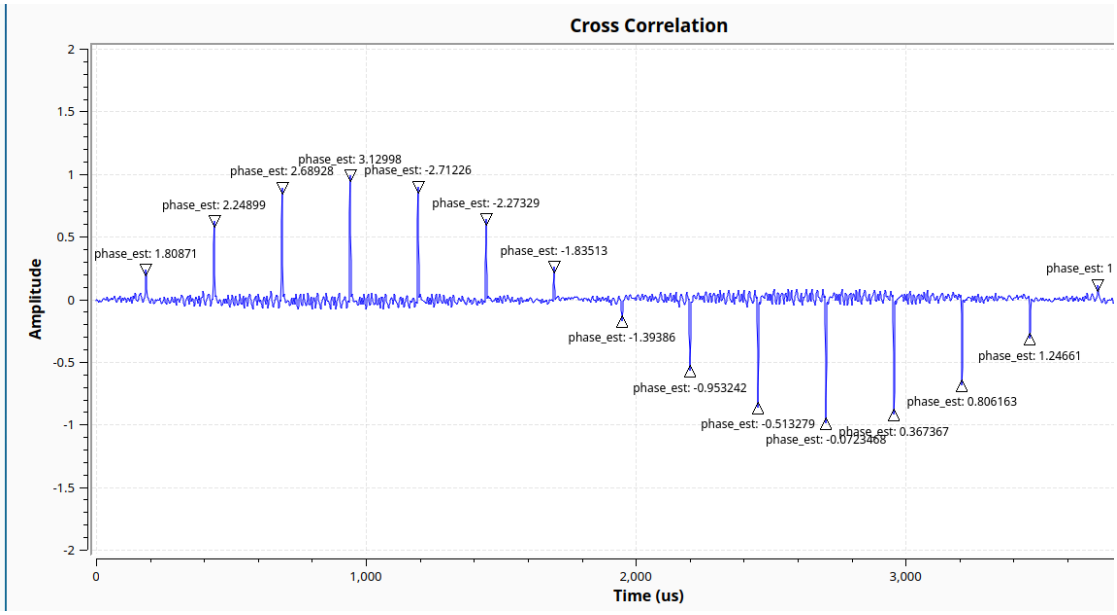
10. Measure the spacing in time between the “delta-like spikes” in the autocorrelation graph.





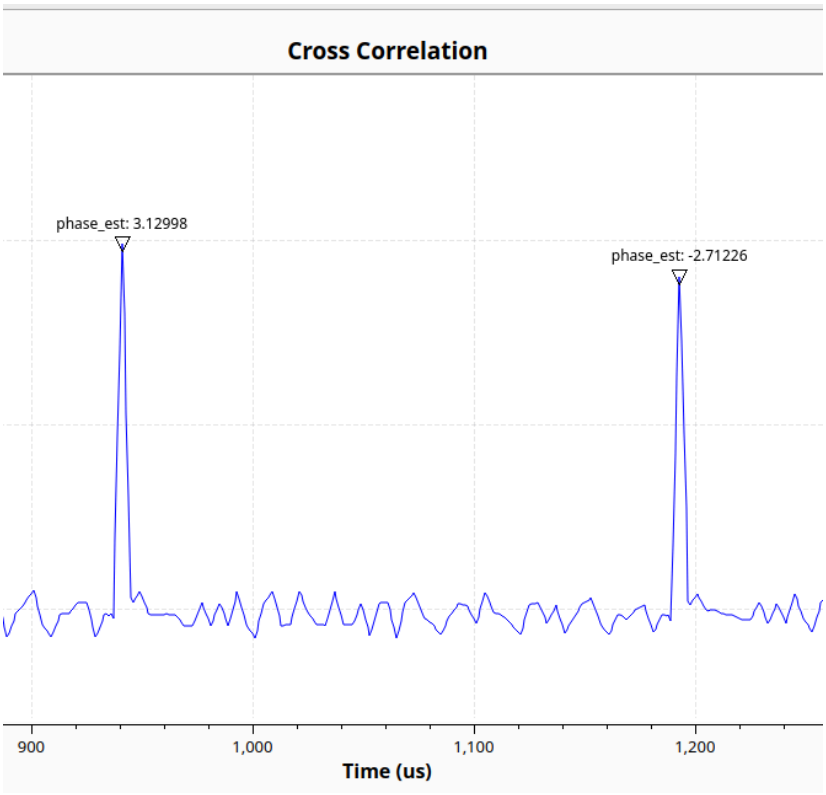
185us to 437us (252us spacing)

11. Measure the period of the envelope of the autocorrelation graph.



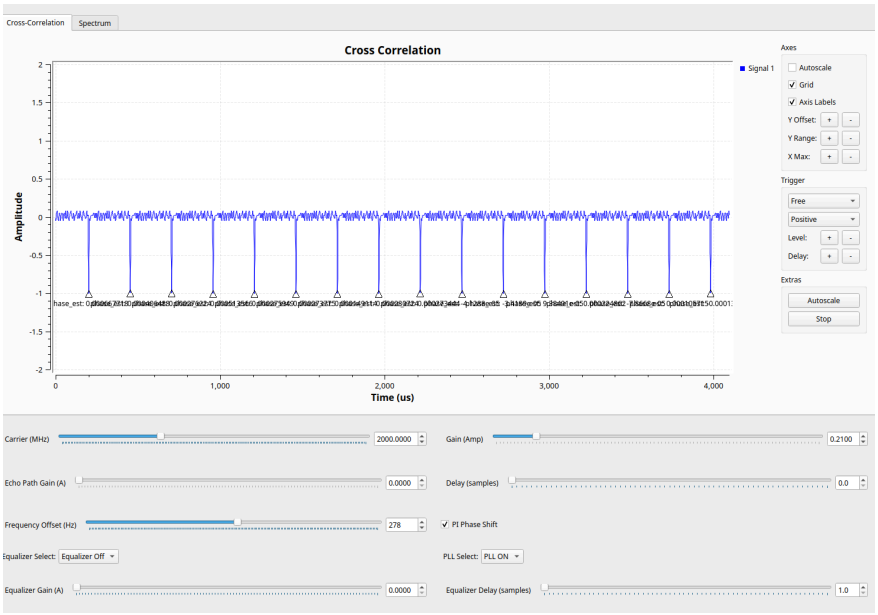
187us to 3463us (3276us spacing)

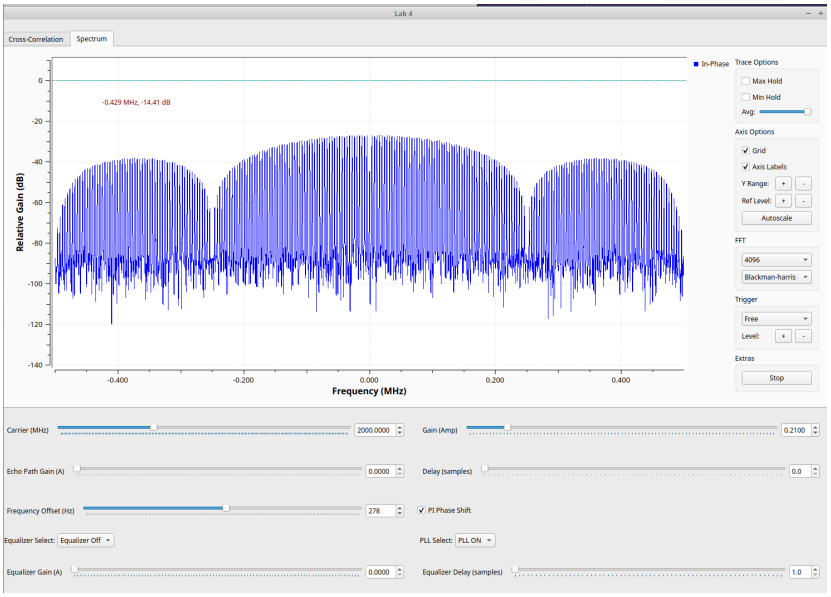
12. Write down the measured phase values of two consecutive spikes corresponding to the "*phase_est*" label. (Note: these phase estimate values are given in radians). Denote these two phase measurements $\phi(t_0)$ and $\phi(t_1)$ where $t_0 < t_1$.



$\phi(t_0)$ = 3.12998 and $\phi(t_1)$ = -2.71226

13. Now, enable the PLL control. What do you observe in your frequency spectrum and autocorrelation graphs? Why?





When we turned on the Phase-Locked Loop (PLL) control for the PN (pseudo-random noise) sequence, the graph is now a bunch of uniform delta spikes. This means that the PLL has locked onto the sequence and is effectively synchronizing with the incoming signal.

Write Up and Scoring for This Part of Lab (40 points)

1. (20 points) Screenshots and answers to all questions posed in the Section.

2. (10 points) Based on your measured phase values $\phi(t_0)$ and $\phi(t_1)$ from part (12). calculate the estimated frequency offset between the TX and RX chain of the SDR's. Recall, for a sinusoid $x(t)$ where

$$x(t) = cos\ (\phi(t))$$

$$\phi(t) = 2\pi f_o t + \phi_0$$

the instantaneous phase is given by $\phi(t)$ and f_o is the frequency offset. Do this for at least 2 different f_o values to confirm if this technique to estimate f_o works.

$\phi(t_0)$ = 3.12998, t0 ~ 940us and $\phi(t_1)$ = -2.71226 rad = 3.58 rad t1 ~ 1190 us

$\phi(t_1) - \phi(t_0) = 2\pi f_o(1190-940)*10\text{^-}6$

$3.58 - 3.13 = 2\pi f_o(1190-940)*10\text{^-}6$

$f_0 = 0.45/2\pi(250*10\text{^-}6)$

$f_0 = 286.5Hz$

The frequency offset on the slider was 278Hz, so this technique to estimate f_0 is quite accurate.

3. (10 points) Recalling that the autocorrelation function is the Fourier transform of the power density spectrum, compare your frequency measurements from parts (6)-(7) to the time measurements from parts (10)-(11). Discuss how the key parameters for one function can be inferred from the other function.

We know that the sinc spectrum lies in the frequency domain and the autocorrelation spectrum lies in the time domain. Given that the sinc spectrum in the frequency domain is represented by a rectangle pulse in the time domain, we can use this fact to infer key parameters of the autocorrelation function based on the parameters of the sinc function. Based on the characteristics of the sinc function, if the first lobe of the sinc function goes from 1/T to -1/T, then the corresponding rectangular pulse lasts for time T. We also know that the width of the first lobe on the autocorrelation function can be denoted as 2T. Based on the above characteristics of the sinc function and autocorrelation function, the width of the main lobe on the autocorrelation spectrum can be inferred from the width of the main lobe of the sinc spectrum.

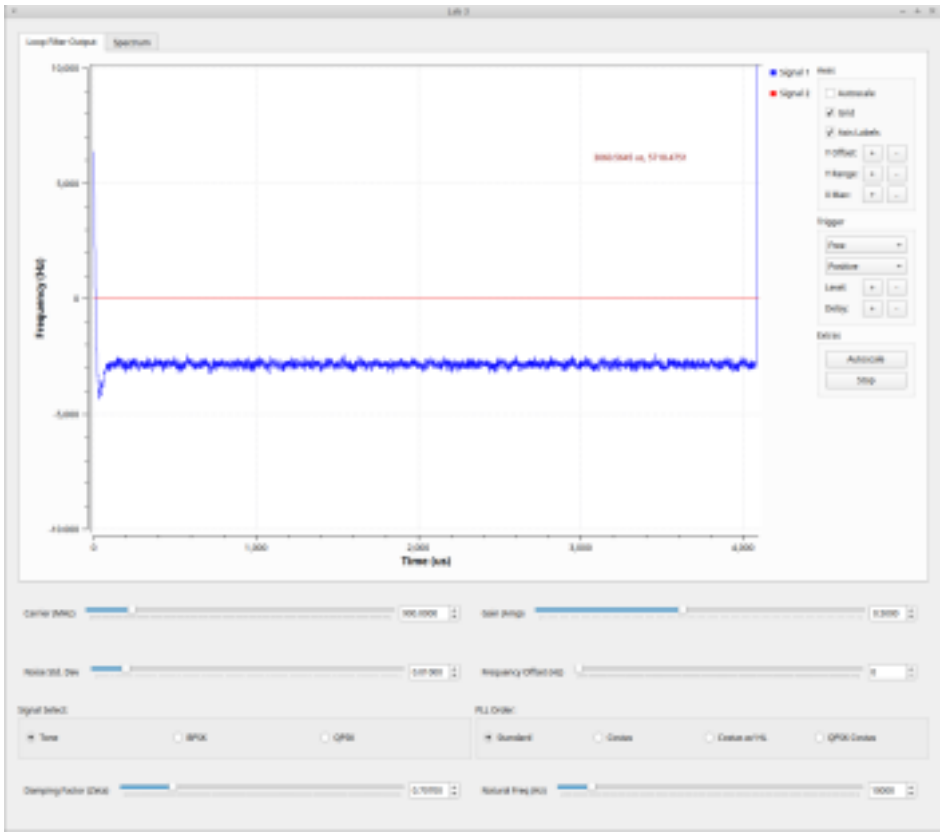
3 Phase Estimation Using PLL

This part of the lab measures the performance of the two PLLs discussed in lecture:

1. Standard PLL with a proportional-integral (PI) filter.
2. Costas Loop with a proportional-integral (PI) filter

3.1 Tracking the Frequency of a Sinusoidal Signal

Open the front panel of course/grc/lab3/lab3.py. The front panel should look like this

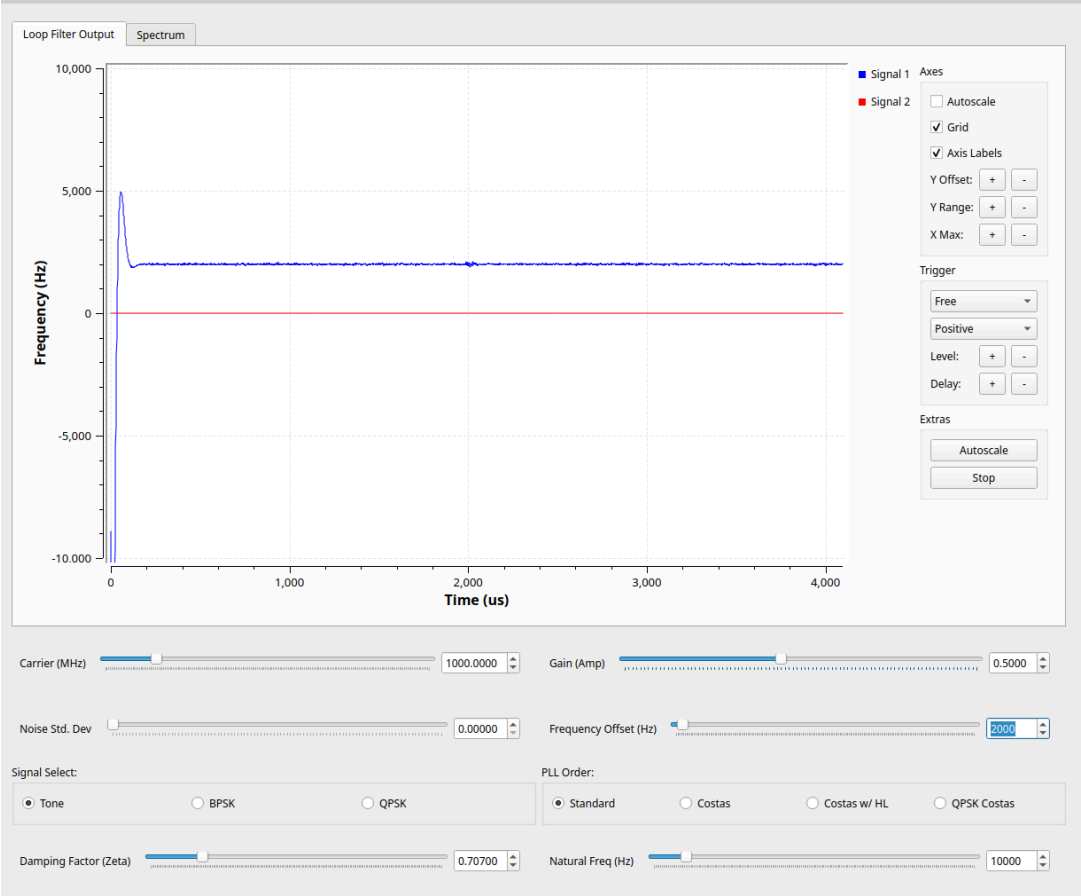


This front panel displays both the magnitude spectrum of the received complex baseband spectrum and loop filter output on separate tabs.

3.1.1 Locking to a Single Tone

Start with the PLL front panel configured for PLL Order: “Standard” and the signal to “Tone”. Display the “Loop filter out” tab. The parameters of the front panel are set according to the parameters listed below. Use a carrier frequency of 1 GHz for all the sets. Take a screenshot of each of the parameters settings shown below for both the loop filter output and the spectrum. Switch the PLL mode to a Costas loop and determine if there is any difference in the response. Use the autoscale button and the stop button to freeze the trace before the screenshot. You can also “drag” a window to zoom the time axis. (Right click to zoom out.) The parameter sets are:

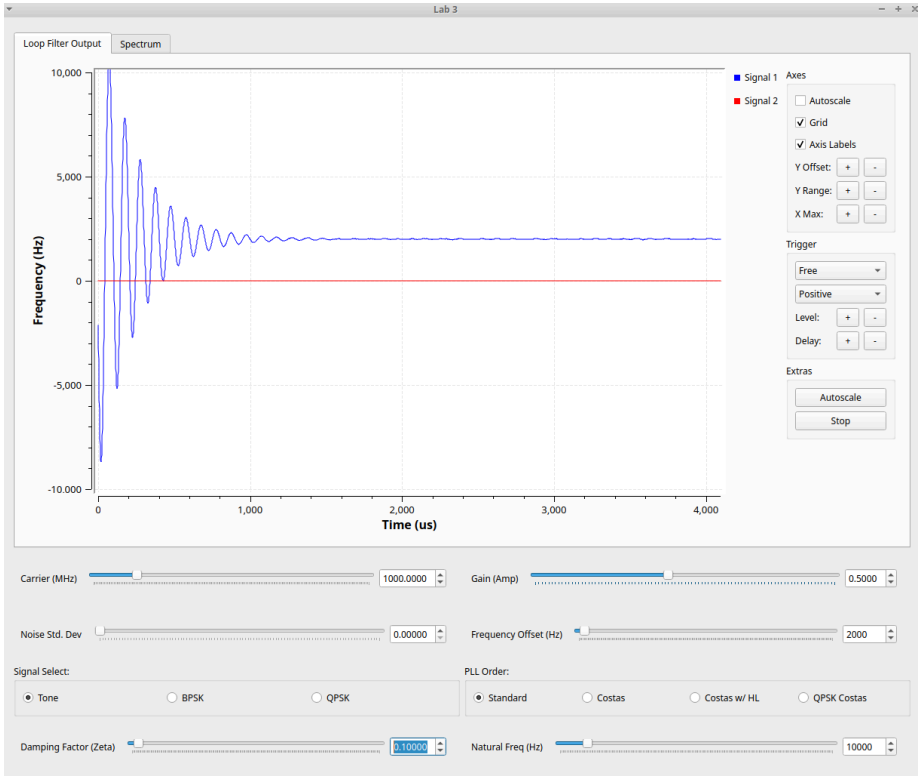
1. fn = 10 kHz, Δf = 2 kHz, $\sigma = 0$, $\zeta = 1/\sqrt{2}$. (linear, no noise, optimal damping).



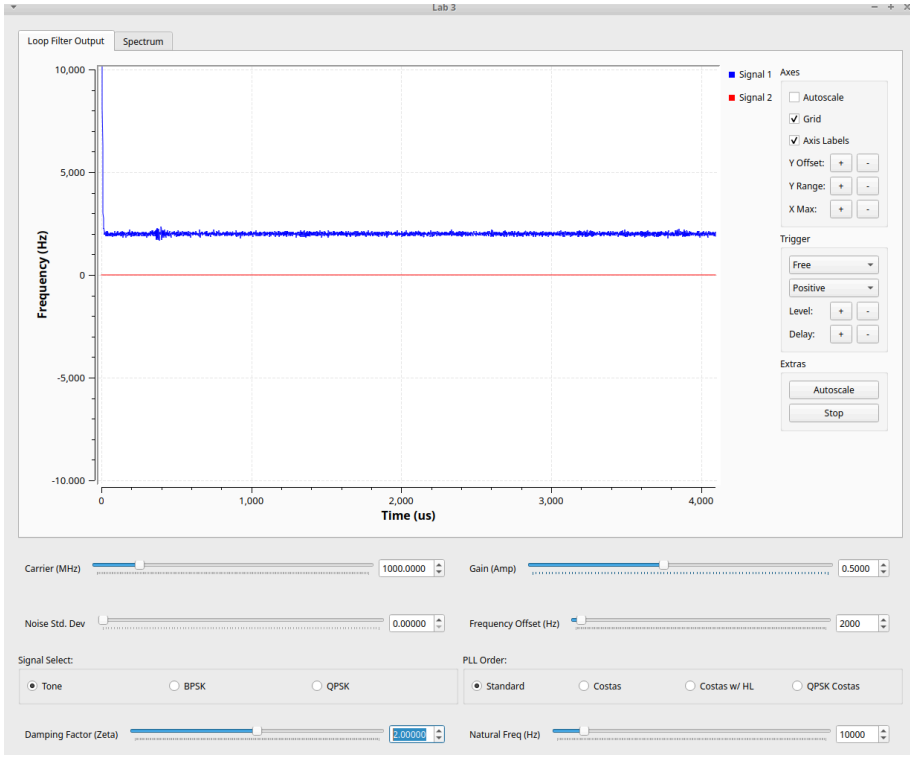
For this set, move one of the Plutos by hand rapidly away from the other Pluto and write down what you observe:

The signal starts to oscillate faster, somewhat at the speed that we are moving the Plutos with respect to each other.

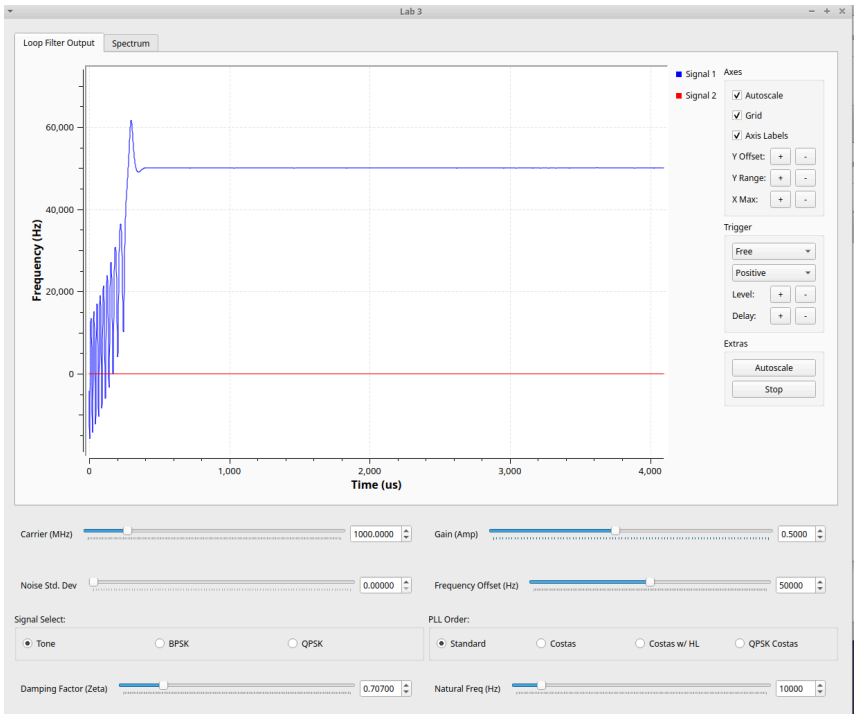
2. fn = 10 kHz, Δf = 2 kHz, σ = 0 , ζ = 0.1 . (linear, no noise, under damped).



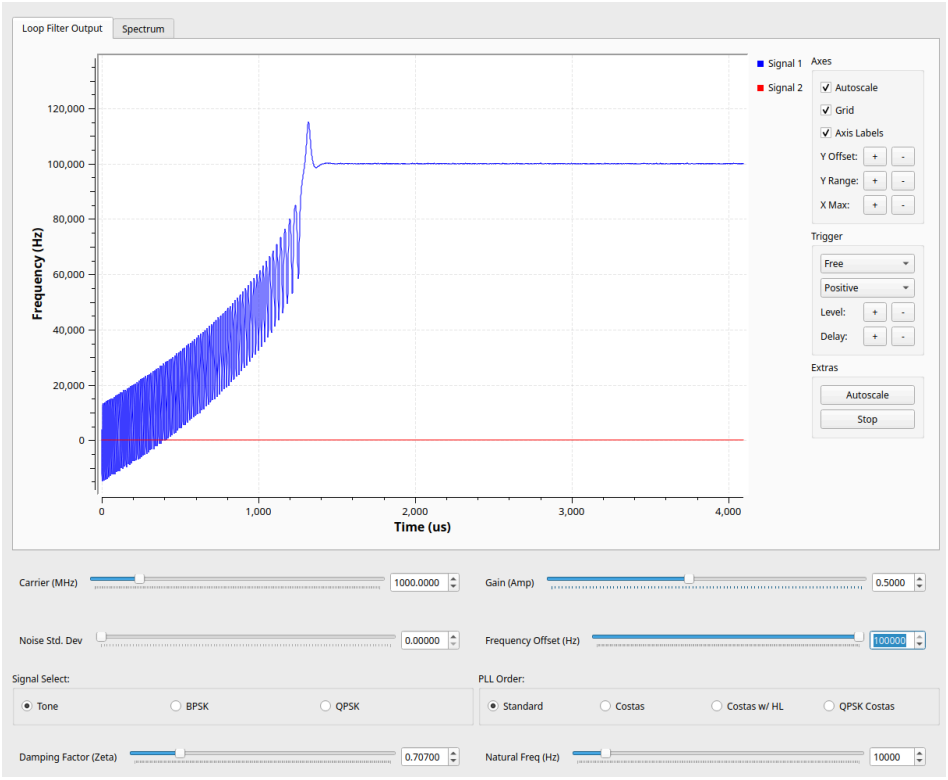
3. fn = 10 kHz, Δf = 2 kHz, σ = 0 , ζ = 2 . (linear, no noise, over damped).



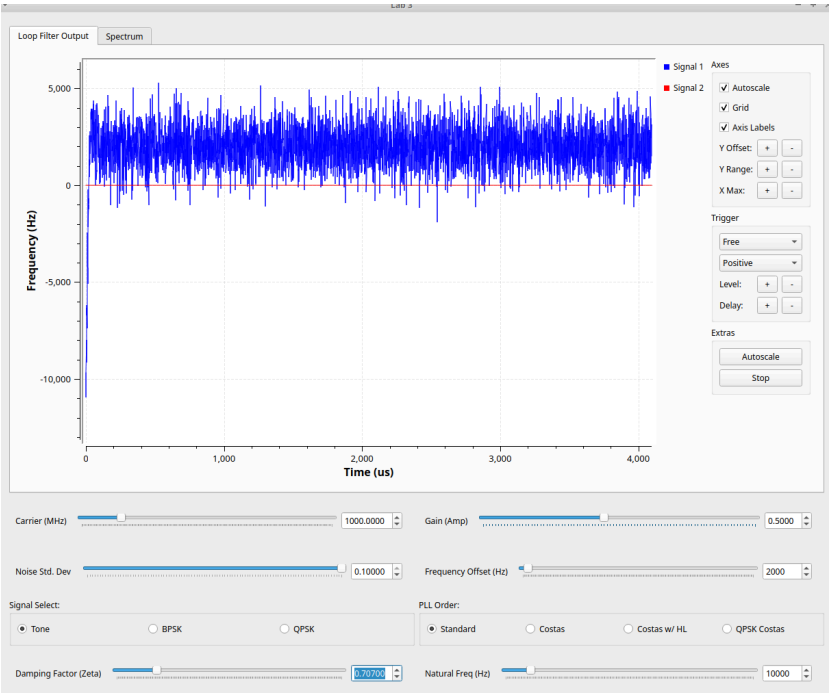
4. fn = 10 kHz, Δf = 50 kHz, σ = 0, $\zeta = 1/\sqrt{2}$. (nonlinear, no noise, optimal damping).



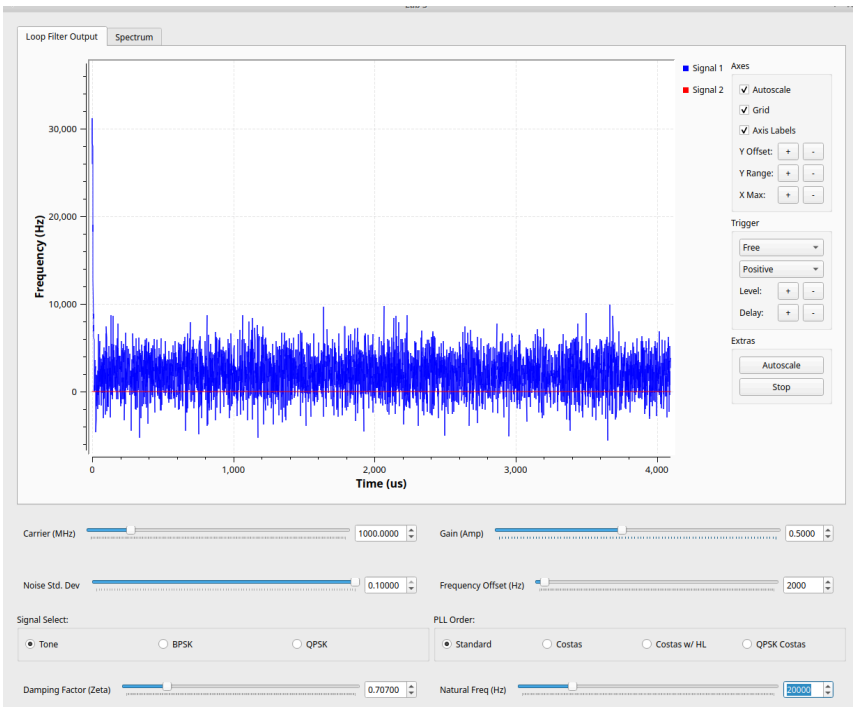
5. fn = 10 kHz, Δf = 100 kHz, σ = 0, $\zeta = 1/\sqrt{2}$. (very nonlinear, no noise, optimal damping)—shown below.



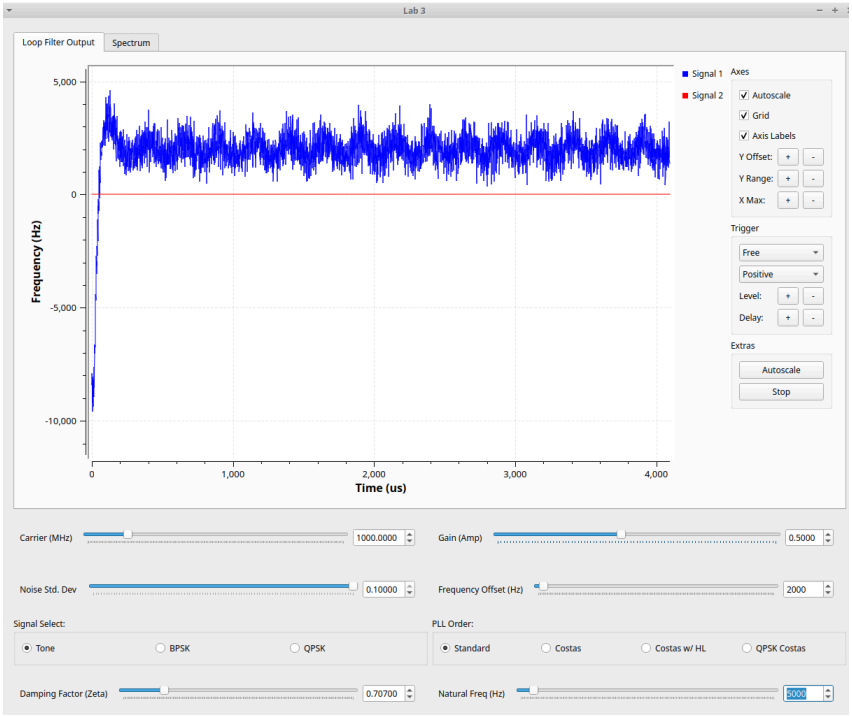
6. f_n = 10 kHz, Δf = 2 kHz, σ = 0.1 . (linear, noise, optimal damping).



7. f_n = 20 kHz, Δf = 2 kHz, σ = 0.1 . (linear, more noise, optimal damping).



8. f_n = 5 kHz, Δf = 2 kHz, σ = 0.1 . (linear, less noise, optimal damping).



All together you should have eight screenshots for the loop filter output for this section with the two sets of three measured over the same time interval to facilitate comparison.

Write-up for Part 3.1

You need to turn in the 8 screenshots for this section and answer the following questions.

1. Using these screenshots, estimate the frequency pull-in time and phase pull-in time for each set of parameters.
   Note: Frequency pull-in time is the time at which the PLL first reaches the desired frequency. 8

   Phase pull-in time is the time at which the PLL settles within 0.5% of the desired frequency after the oscillations.

	Frequency pull-in time	Phase pull-in time
1. fn = 10 kHz, Δf = 2 kHz, σ = 0 , ζ = 1/√2	50μs	100μs
2. fn = 10 kHz, Δf = 2 kHz, σ = 0 , ζ = 0.1	100μs	1400μs
3. fn = 10 kHz, Δf = 2 kHz, σ = 0 , ζ = 2	30μs	30μs
4. fn = 10 kHz, Δf = 50 kHz, σ = 0, ζ = 1/√2	370μs	430μs
5. fn = 10 kHz, Δf = 100 kHz, σ = 0, ζ = 1/√2	1300μs	1450μs

6. f_n = 10 kHz, Δf = 2 kHz, σ = 0.1	100µs	180µs
7. f_n = 20 kHz, Δf = 2 kHz, σ = 0.1	10µs	10µs
8. f_n = 5 kHz, Δf = 2 kHz, σ = 0.1	150µs	230µs

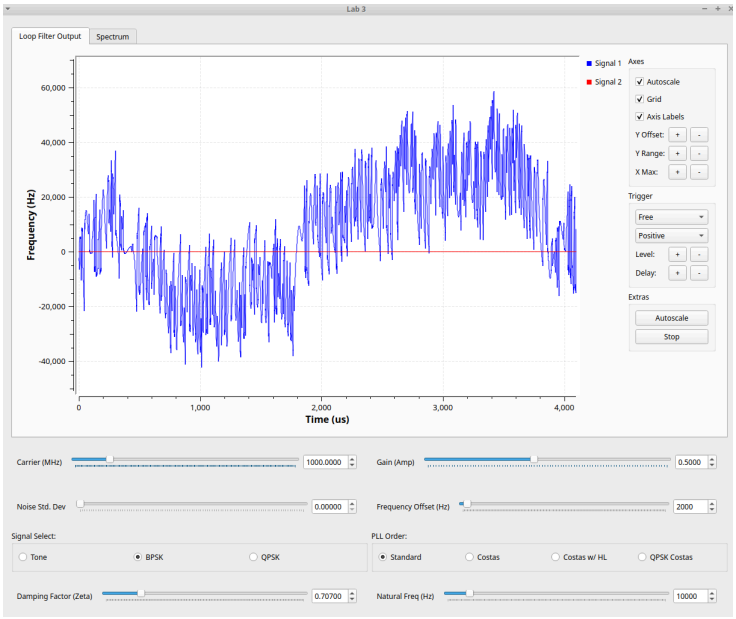
2. Using Data Sets 1-3, explain how changing the damping parameter ζ affects the frequency pull-in time and phase pull-in time.

For when ζ is less than optimal or the system is underdamped, the frequency pull-in time and phase pull-in time takes longer to reach a stable frequency and it produces a lot of ringing. The Optimal damping factor has very little overshoot and reaches the desired frequency pretty quickly from its frequency pull-in time to its phase pull-in time there is very little time. For the overdamping or when $\zeta > 1/\sqrt{2}$, there is a big overshoot but then settles very quickly to the desired frequency as seen in the graph. Its phase-pull in time takes a short time to reach the desired frequency.

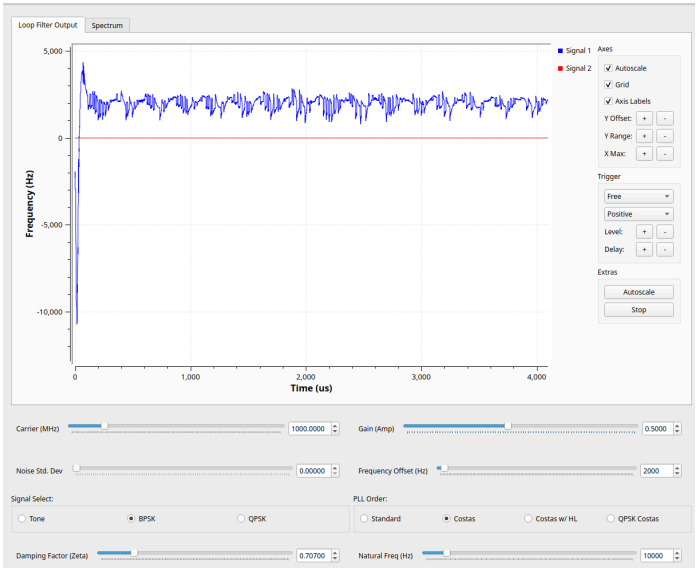
3.2 Locking to a Data Modulated Signal (BONUS section)

This section measures the performance of the PLL for data-modulated *BPSK* and *QPSK* waveforms. On the front panel, change the "Signal Select" and "PLL order" to match the following parameters:

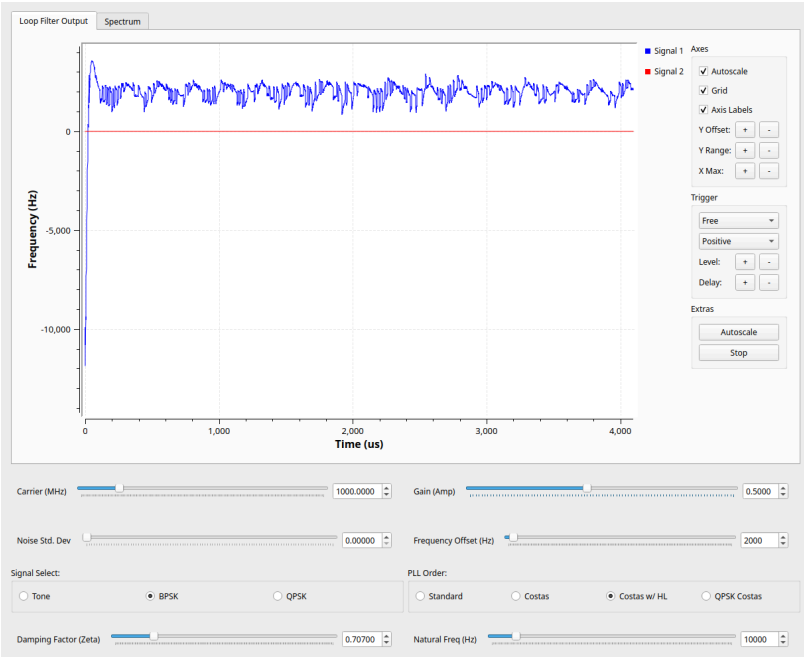
1. f_n = 10 kHz, Δf = 2 kHz, σ = 0, BPSK, Loop = "Standard". (nonlinear, no noise, BPSK).



2. f_n = 10 kHz, Δf = 2 kHz, σ = 0, BPSK, Loop = "Costas". (nonlinear, no noise, BPSK).

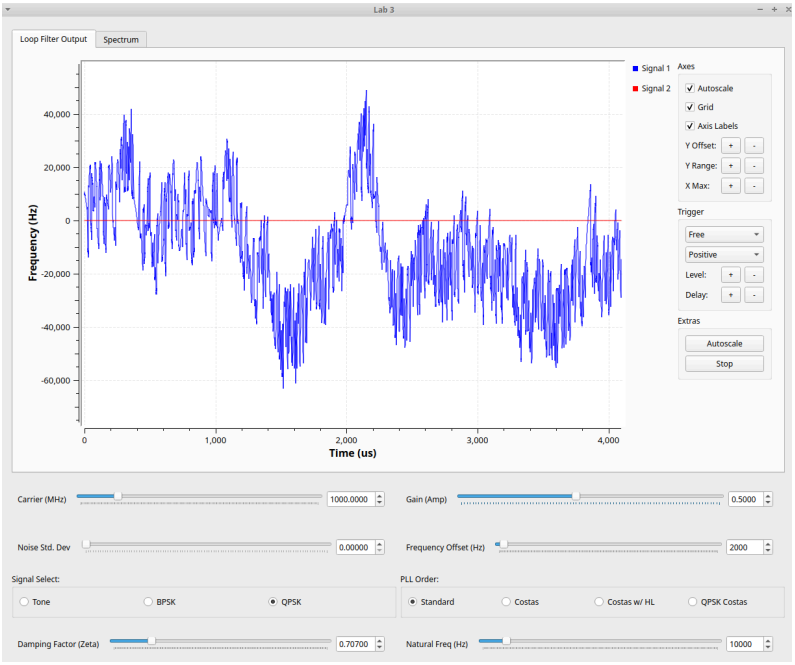


3. f_n = 10 kHz, Δf = 2 kHz, σ = 0, BPSK, Loop = “Costas w/HL”. (nonlinear, no noise, BPSK with hard listed output).

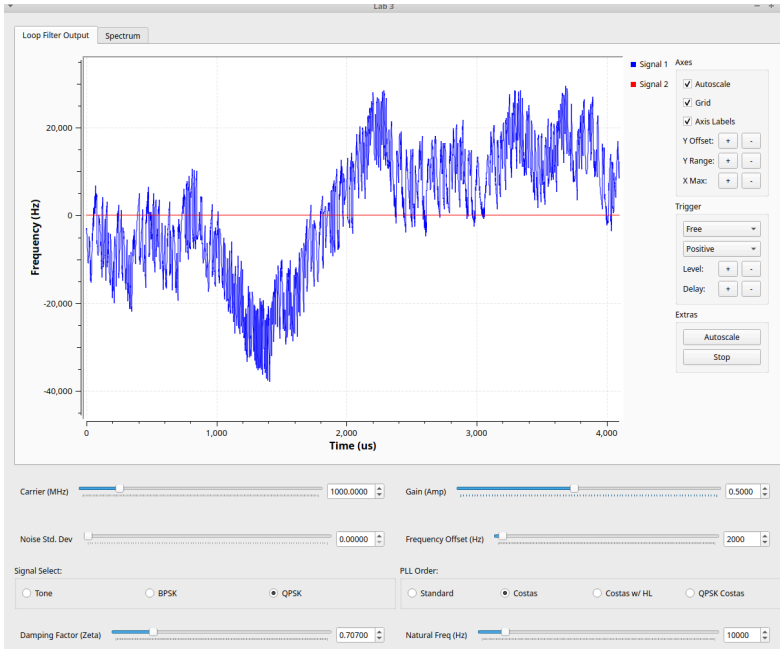


For this parameter set only, and based on the theory presented in lecture, try and deter mine different sets of parameters for which the hard-limited version of the Costas loop gives a significantly different response than the standard Costas loop. You are free to change any parameter to demonstrate this difference.

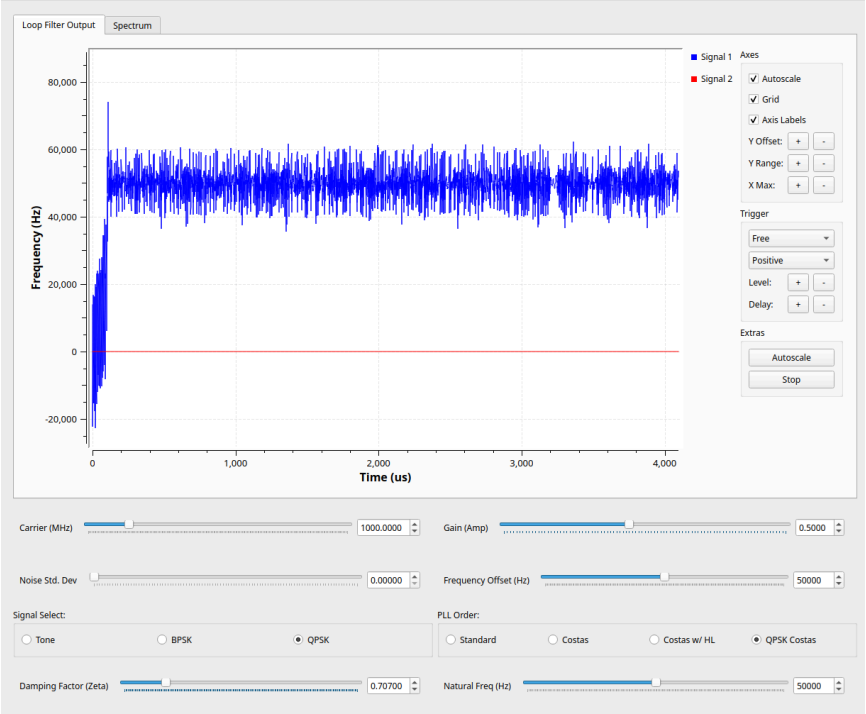
4. f_n = 10 kHz, Δf = 2 kHz, σ = 0, QPSK, *Loop* = “Standard”. (nonlinear, no noise, QPSK).



5. f_n = 10 kHz, Δf = 2 kHz, σ = 0, *QPSK*, *Loop* = “Costas”. (nonlinear, no noise, QPSK).



6. f_n = 50 kHz, Δf = 50 kHz, σ = 0, *QPSK*, *Loop* = “QPSK Costas”. (nonlinear, no noise, QPSK).



Write-up for Part 3.2

You will need to turn in your set of six screenshots and answer the following questions:

1. What happens when you try to track a data modulated signal with the *Standard PLL*? Why?

When we try to track a data modulated signal with the Standard PLL it may have difficulties to lock and maintain a stable lock and can cause high phase error. This is because the PLL are meant to track continuous and unmodulated carrier signals where phase and frequency are constant, but data modulated signals have phase.

2. Why does the QPSK Costas loop still work for BPSK?

The QPSK Costas loop can handle phase changes in both I and Q components and will still lock onto the phase info in the I component. Where the BPSK only phase component is modulated while Q components remain zero, the Costas loop won't sense Q path energy and the loop will be adjusted to nullify Q components.

3. For the parameter space tested, is there a significant difference in performance between the Costas loop and the Costas loop with a hard limiter? Why?

The frequency of the Costas loop with the hard limiter is slightly less than the frequency of just the Costas loop which shows signs of performance degradation. The standard Costas loop is preferable for achieving optimal performance in terms of frequency and phase tracking while the hard limiter is beneficial in noisy conditions, since there was no noise present, it introduced unnecessary distortions and information loss.

Additional Questions for Final Write-up

1. Using the data for both a single tone and a data-modulated waveform, discuss the relative performance of the PLLs with respect to the following parameters:

   a) Frequency pull-in time vs. phase pull-in with respect to damping parameter ζ

   With a single tone, the PPL can more easily achieve frequency and phase lock due to the simplicity and predictability of the signal, the effects of the damping factor is simple, high damping provides a smoother but slower pull-in and low damping offers faster but oscillatory pull-in.

   The BPSK and QPSK require the PLL to respond dynamically because of the complex phase transitions, however there are similar trade-offs, high damping provides stability but increases both frequency and phase pull-in times (slower but smoother response) and low damping (as we see with 0 damping) improves responsiveness (reduced pull-in times) but is unstable with a lot of oscillation.

   b) Phase pull-in time vs. total amount of phase noise with respect to the natural frequency f_n

   For single tone, a higher fn reduces phase pull-in time and lock quickly, but there's an increase in phase noise. Lower fn helped maintain a clean phase lock but was slower.

   For the QPSK and the BPSK, high fn improved the responsiveness to phase changes in the data but low fn reduced the impact of phase noise.

   c) Ability to lock in the presence of data (i.e. *BPSK* , *QPSK*)

   For BPSK, the PLL must handle 180-degree phase shifts. A well-tuned damping factor and fn are critical to balance quick response and stability. High (ensures stability but slower response; low improves responsiveness but risks overshooting.

   QPSK requires handling 90-degree phase shifts, making the lock process more complex than BPSK. The PLL must quickly adapt to phase changes without being destabilized by rapid transitions. Proper tuning of ζ and fn is essential for achieving a good lock and minimizing bit errors.

2. In order to track a BPSK signal, it is necessary to "square" the signal to remove the data (Figure 5.6 of Papen/Blahut). This process can be generalized to higher order $2^M - PSK$ signals. Explain how you would create a PLL to track a $2^M - PSK$ signal where the possible phase values ϕ_k are:

$$\phi_k = \frac{2\pi}{2^M} k$$

$$k = 0, 1, ..., 2^M - 1$$

1. Signal Demodulation: Demodulate the received signal to baseband using a mixer and a local oscillator at the carrier frequency.

2. Phase Detector: Use a phase detector that can handle the discrete phase values ϕk. It will compare the phase of the demodulated signal with the expected phase values ϕk.

3. Loop Filter: Process the output of the phase detector and generate a control voltage for the VCO.

4. Voltage-Controlled Oscillator (VCO): The VCO should generate a local oscillator signal at the carrier frequency with the phase adjusted based on the phase error detected by the phase detector.

5. Feedback Loop: Close the feedback loop by feeding the output of the VCO back to the phase detector. The phase detector compares this feedback signal with the demodulated signal to continuously adjust the VCO's phase.

6. Lock Detection: Implement a lock detection mechanism to indicate when the PLL has successfully locked onto the incoming signal.

7. Performance Optimization: Adjust the loop parameters, such as loop bandwidth and damping factor, to optimize the PLL's performance for the specific 2M−P SK signal and noise conditions.