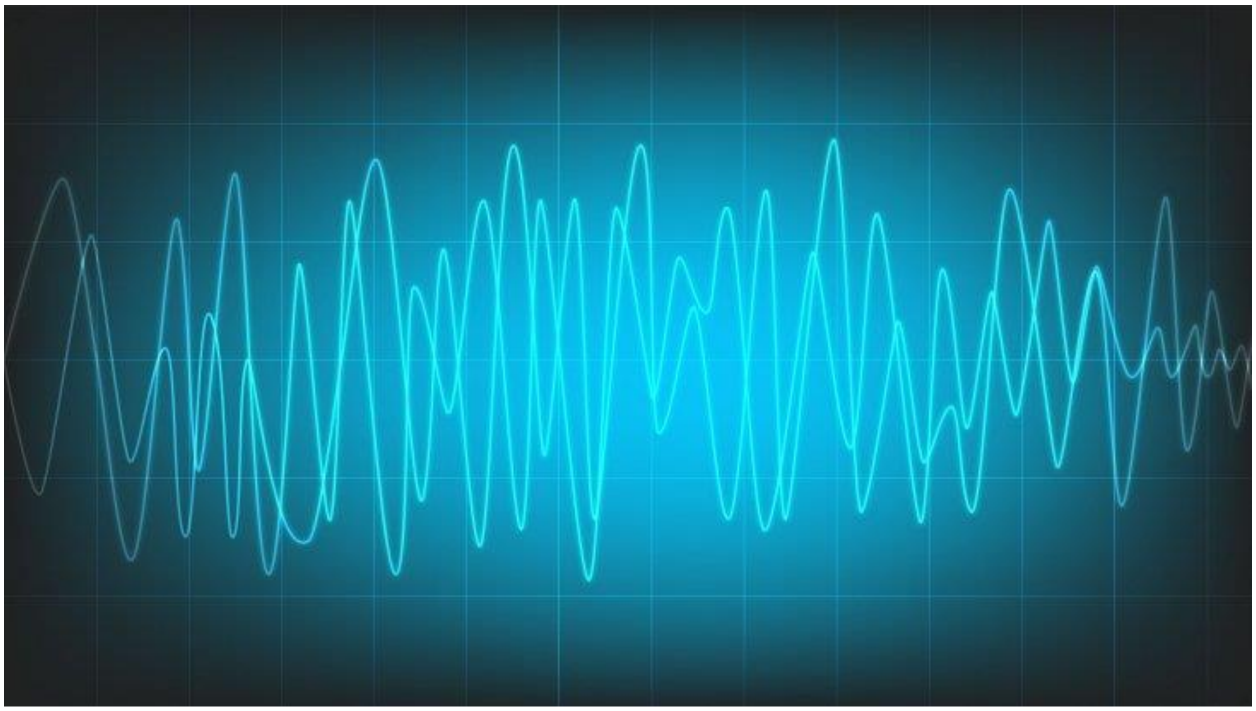
USRP Hardware Lab #2

EELE 445 Telecommunication Systems

Montana State University

Introduction to IQ DSP using SDRs



Author

Alex Brisson

Preface

At Montana State University, EELE 445 Telecommunication Systems is a course which aims to introduce undergraduate students to the theory behind many types of communications systems. Most notably, the course focuses on modulation techniques to optimize channel capacity, distortions introduced by channel effects, and the structures of the transmitter and receiver that enable such systems. An integral part in solidifying the understanding of these concepts is to simulate and analyze practical communication systems in the lab. By creating real-time simulations that incorporate telecommunications hardware, such as the Universal Software Radio Peripheral (USRP), students will be exposed to practical implementations and the engineering challenges involved in designing and building several types of communications systems.

Pre-Lab

Please read the following introduction before coming to lab.

Introduction

In the previous lab, you were introduced to IQ signals, the concept of up-conversion and down-conversion using SDRs, and implementations to optimize spectral efficiency by "shaping" the transmit pulses of a BPSK signal. In this lab, we will try and understand two Digital Signal Processing (DSP) algorithms that work to compensate some impairments in our received complex signal. These two algorithms are known as the Matched Filter Polyphase Filter Bank and the Costas Loop. We will first observe the effects we wish to compensate in our BPSK received signal, implement both algorithms to correct the effects, and finally observe how the algorithms converge to eventually make sense of our transmitted data.

The Matched Filter Polyphase Filter Bank

The Matched Filter Polyphase Filter Bank (MF-PFB) is a DSP algorithm that is a pre-coded block in the gnuradio software. The purpose of this algorithm is to take the received data and perform matched filtering in a unique way. As we have learned before, we want to use raised cosine pulses to take advantage of their spectral efficiency and the minimum ISI benefit. Therefore, in lab #1 you implemented two root raised cosine filters at the transmitter and receiver such that the raised cosine pulse could be "matched" to the transmit pulse through a convolution operation at the receiver. The MF-PFB does exactly this, however in doing so, is able to "pick out" the optimal sample in the RC pulse. You have seen in part 3 of lab #1 that this optimal sample is the sampling instant that occurs at the peak of the RC pulse, although we know that the transmitter and receiver clocks are not perfectly matched in frequency and phase, therefore this optimal sampling instant *may not always* occur at the peak of the pulse. To illustrate this point, consider the following figure. The RC pulses in the figure represent 4 consecutive symbols with value +1. Since we are pretending the signal is encoded using BPSK, never mind their amplitudes. Notice that the optimal sample occurs when all the other symbols' tails are zero.

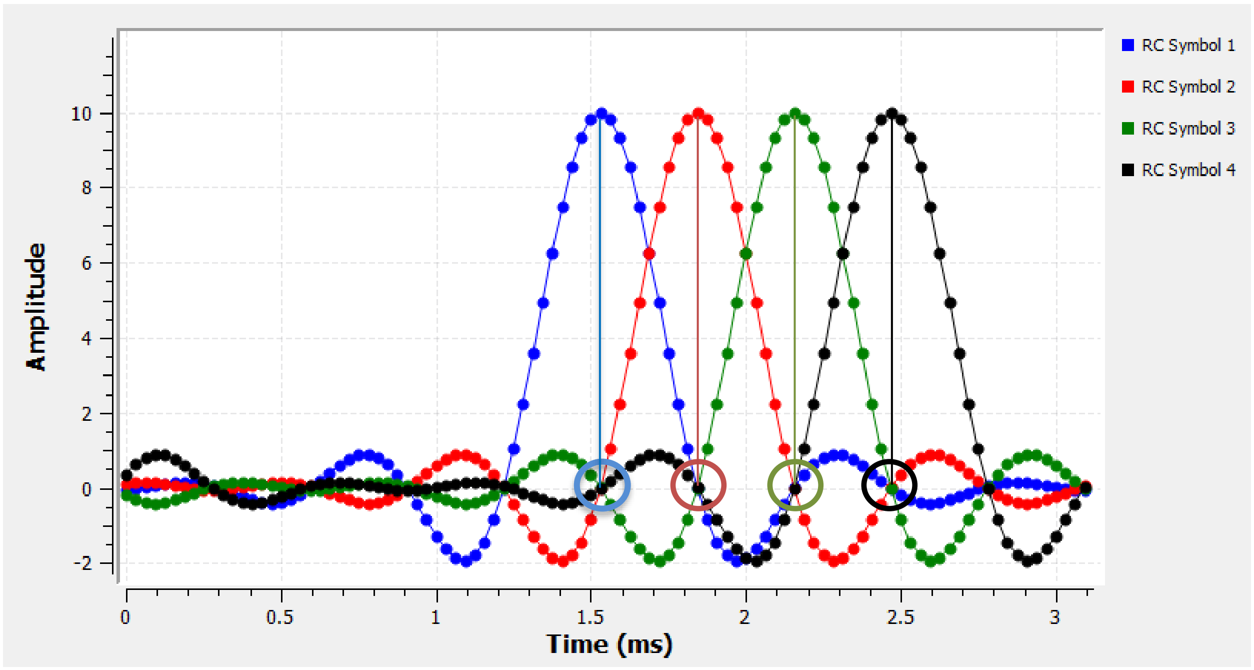


Figure 1: Ideal Sampling Instances RC Pulses

Due to the receiver clock never being perfectly in phase with the transmitter clock and exhibit slight changes in frequency, these "perfect" sampling instances rarely, if ever, occur and we begin to see contributions of ISI into our pulses, figure 2.

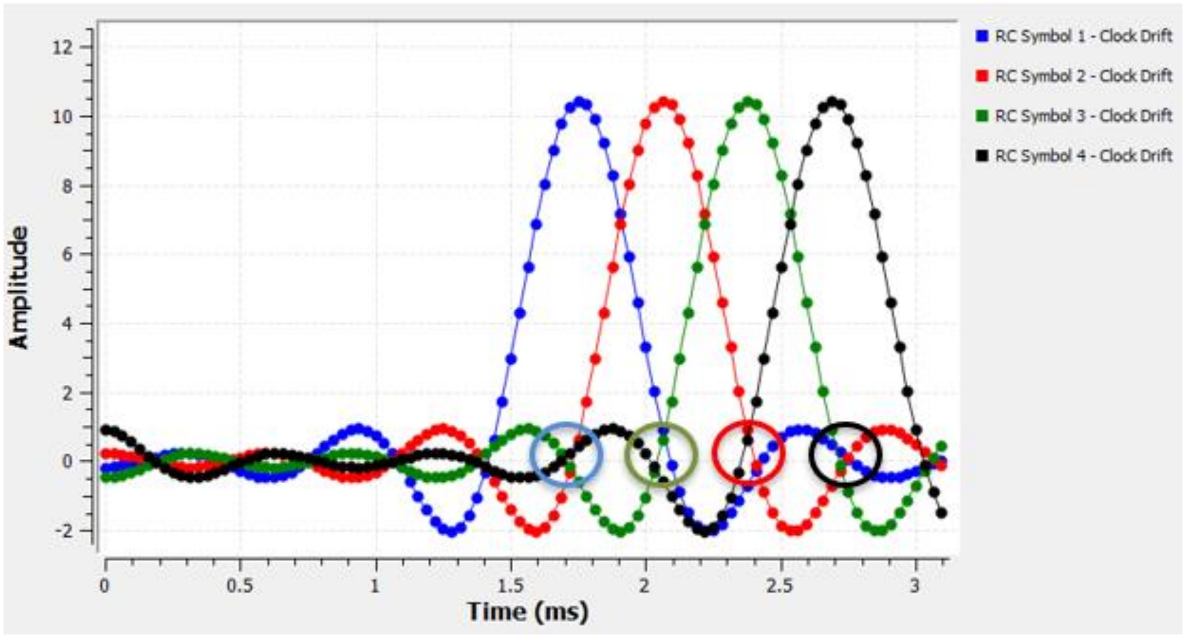


Figure 2: Sampling Instances due to Clock Drift and ISI Contributions

The MF-PFB's job is then to estimate how far the clock has drifted and attempt to compensate the drift such that the receiver always chooses the optimal sampling instant. Okay, all good, but how does the MF-PFB know which sample is the *optimal* one? Well, we have defined the optimal sample to be the one that occurs at the peak of the pulse where $\frac{dP}{dt} = 0$, so we can take a time derivative of our incoming pulses, P(t), and see which sample corresponds to the derivative being zero. Although one may notice again that there's a problem here. I told you that the clock can drift around so it may not sample the pulse at exactly its peak, and the corresponding derivative may never be zero for any of the samples taken by the ADC. This is especially true if we lower our sampling rate to the Nyquist limit. This problem can be solved by using a fractional interpolating scheme which is a process to manually insert samples in between the hardware samples obtained by the ADC to increase the time resolution sufficiently so that we can pick the optimal sample, however this technique can be very computationally expensive. Therefore, to avoid this, the MF-PFB *pre-computes* up-sampled *phase shifted versions* of the RRC matched filter. Each phase-shifted version fills up a bank of filters that the receiver can choose from to obtain and lock to the samples that correspond to $\frac{dP}{dt} = 0$. See figures 3-5 for illustrative descriptions.

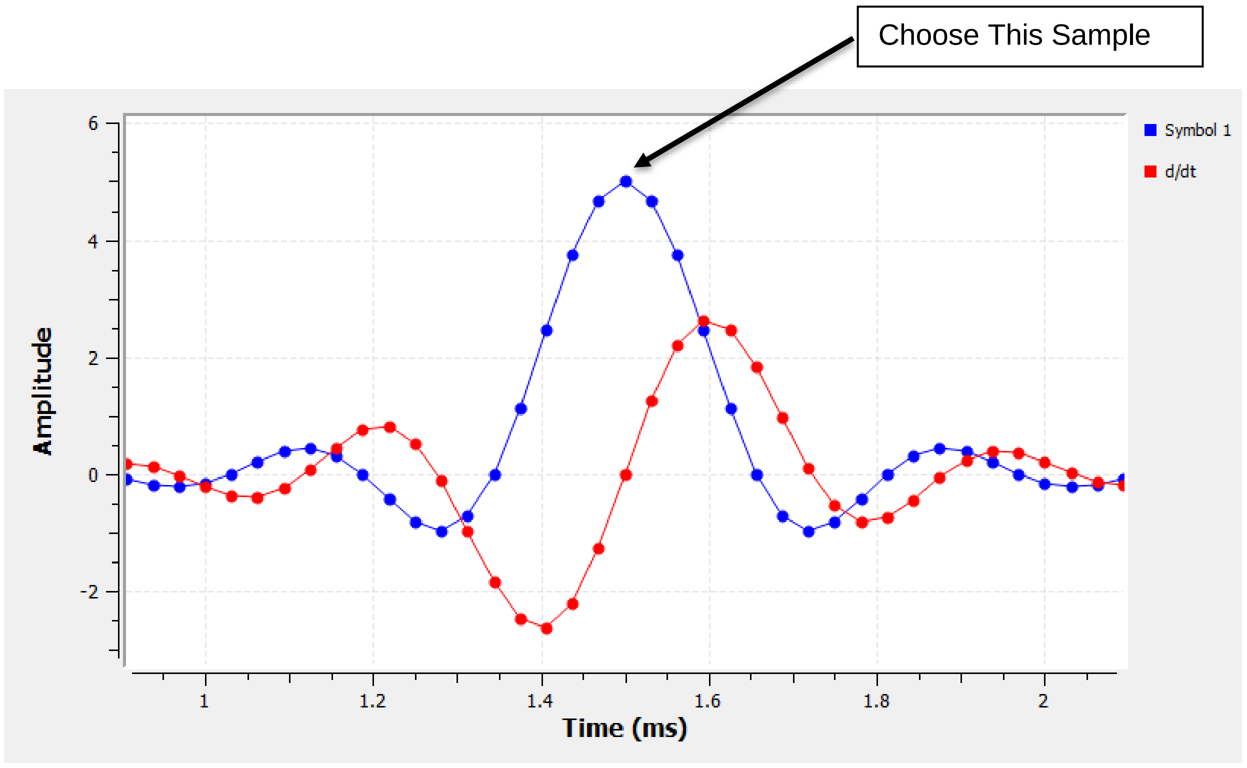


Figure 3: The ideal sampling instant occurs when $\frac{dP}{dt} = 0$.

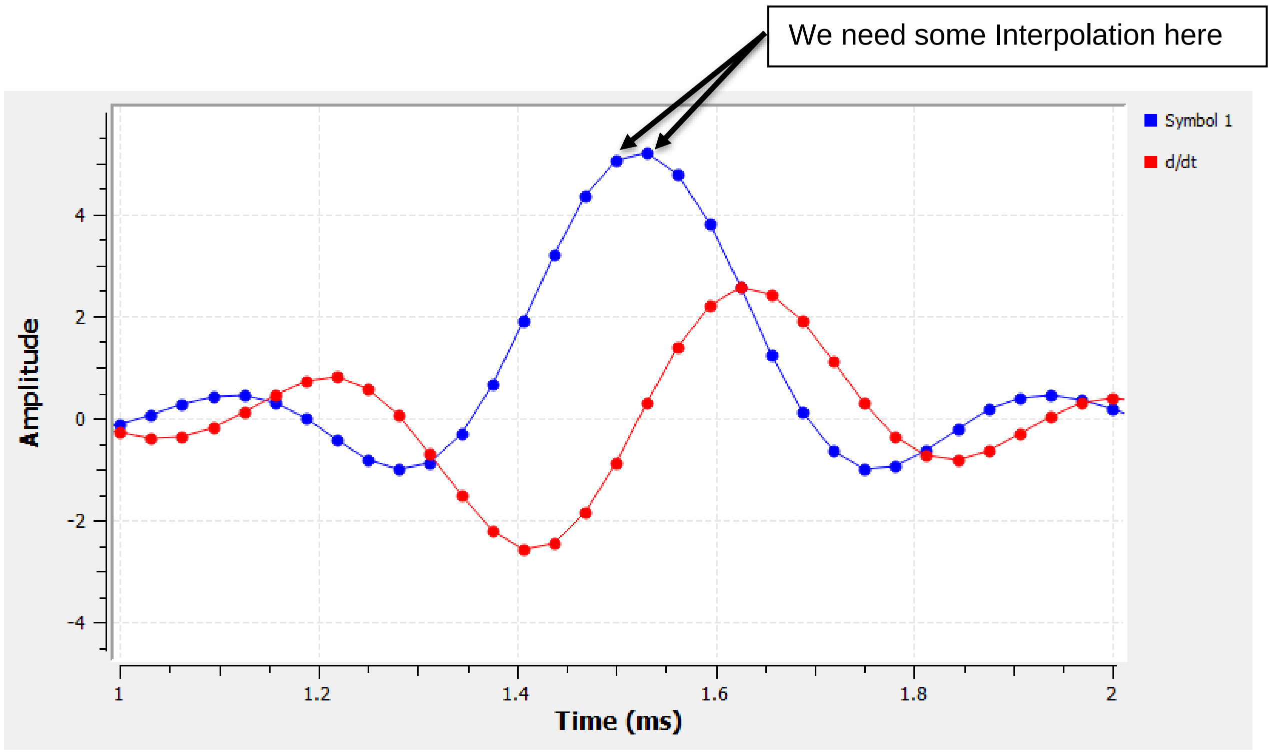


Figure 3: Demonstrate the need for interpolation

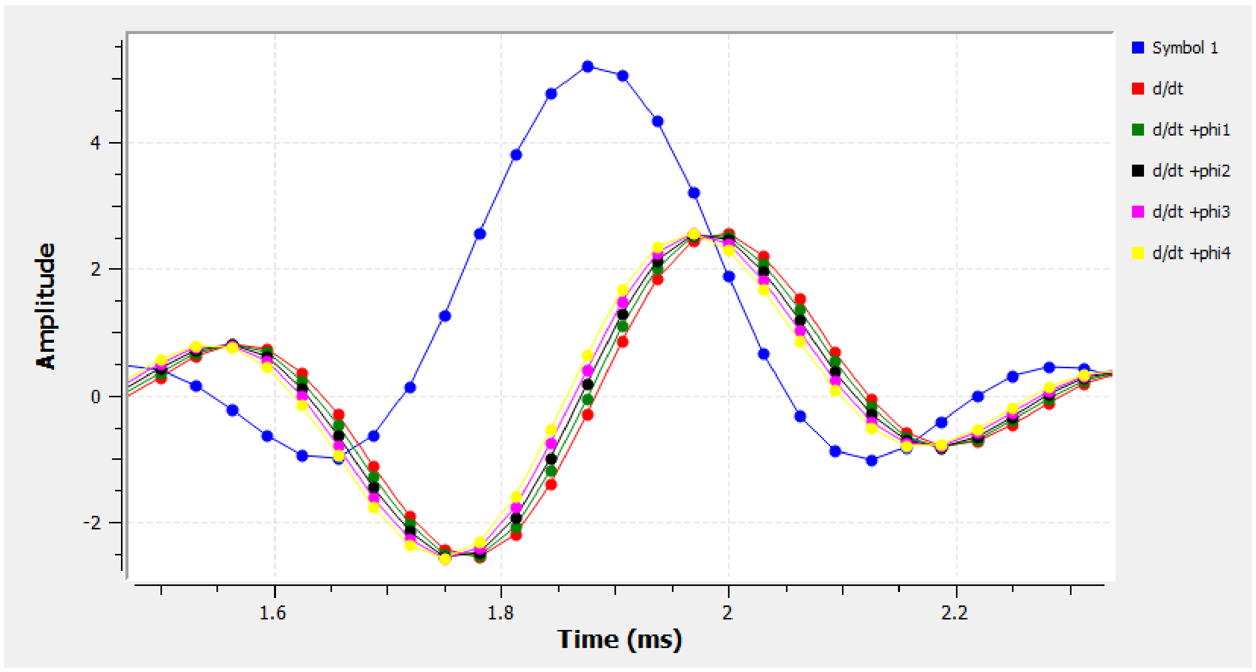


Figure 4: Bank of Phase-Shifted Filters- In this case, phi1 is the correct phase-shifted filter to use.

The Costas Loop

The next DSP algorithm we will use is known as the Costas Loop. This algorithm is also a pre-coded block in the gnuradio software, and it works to estimate and correct any phase and small frequency errors of the receive signal. It is important to note that this algorithm will be implemented after we correct the clock drift and have received the optimal sample that represents the amplitudes of our I and Q signals. We therefore assume, after clock synchronization, that our I and Q amplitudes are *correct* i.e., for PSK signals, their superposition is always going to be equal to 1. However, due to impairments caused by the channel and hardware, may not be superimposing in the correct phase which represents the ideal symbol locations in signal space. Basically, the "points" in the constellation will be shifted in phase away from their nominal positions in signal space. If there are any residual frequency offsets between clocks after the coarse frequency correction, these points will rotate slowly in a circle. Figure 5 illustrates this where the black points represent the "ideal" symbol locations in signal space for BPSK and the green points are smeared across the unit circle with phase and small frequency errors.

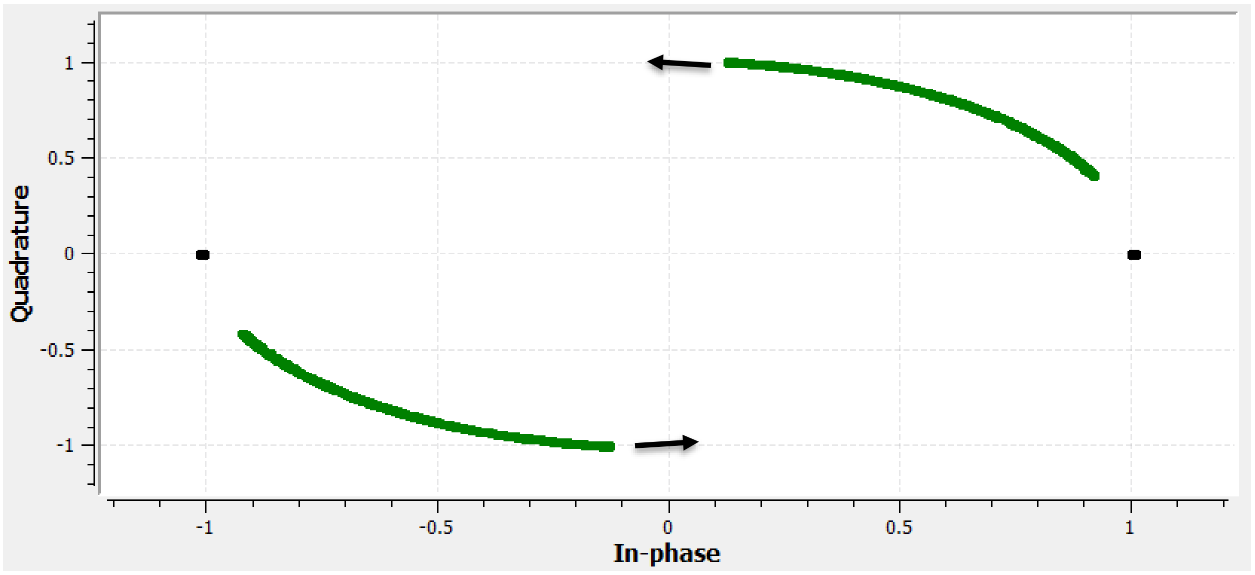
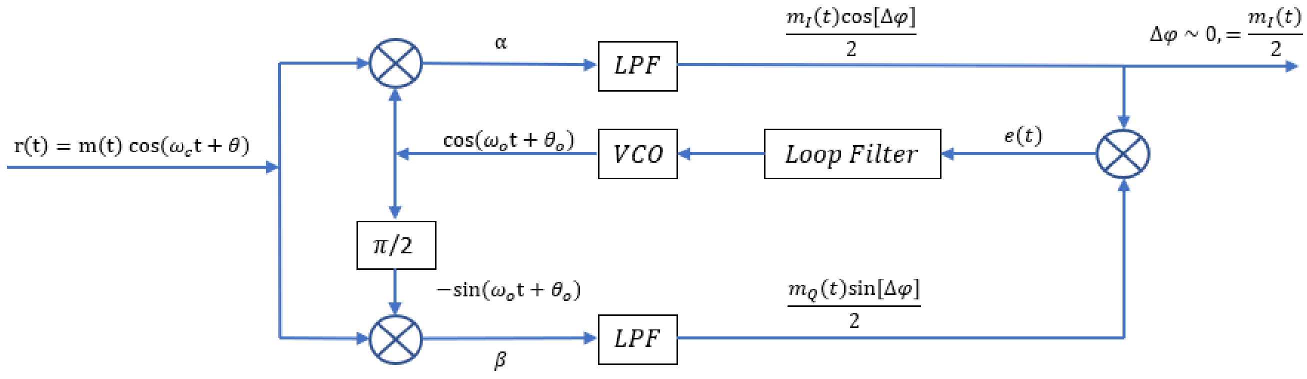


Figure 5: Phase and Frequency Error in a Received Signal

So, how can this effect be corrected? The answer comes through the implementation of the Costas Loop algorithm, and its BPSK structure is illustrated in figure 6.



$r(t) = m(t)\cos(\omega_c t + \theta)$
α
LPF
$\frac{m_I(t)\cos[\Delta\varphi]}{2}$
$\Delta\varphi \sim 0, = \frac{m_I(t)}{2}$
$\cos(\omega_o t + \theta_o)$
VCO
Loop Filter
$e(t)$
$\pi/2$
$-\sin(\omega_o t + \theta_o)$
$\frac{m_Q(t)\sin[\Delta\varphi]}{2}$
LPF
β

Figure 6: Costas Loop DSP Circuit for BPSK

First, the complex signal at the carrier frequency must be down converted to baseband by multiplying the received signal, r(t) by two quadrature tones at a frequency ω_o which we assume is close to the carrier frequency ω_c of our received waveform, r(t). This multiplication generates two copies of our I and Q spectrums at sum and difference frequencies.

$$\alpha = m_I(t)\left[\frac{\cos[(\omega_c - \omega_o)t + \theta - \theta_o] + \cos[(\omega_c + \omega_o)t + \theta + \theta_o]}{2}\right]$$

$$\beta = m_Q(t)\left[\frac{\sin[(\omega_c - \omega_o)t + \theta - \theta_o] - \sin[(\omega_c + \omega_o)t + \theta + \theta_o]}{2}\right]$$

The fast-oscillating sum terms that come out in the multiplications are of no interest to us, and we therefore low-pass filter them out of the down converted spectrum. The baseband I and Q waveforms can then be represented by the following equations, where $\Delta\varphi$ represents the argument of the sine and cosine functions after down-conversion.

$$\frac{m_I(t)\cos[\Delta\varphi]}{2}$$

$$\frac{m_Q(t)\sin[\Delta\varphi]}{2}$$

$$\Delta\varphi = (\omega_c - \omega_o)t + \theta - \theta_o$$

The Costas Loop then multiplies the baseband I and Q waveforms by each other and the error function, $e(t)$, becomes proportional to $\Delta\varphi$ through the following equation.

$$e(t) = \frac{m_Q(t)m_I(t)\sin[2\Delta\varphi]}{4}$$

This error function is then sent to a *loop filter* which is used to “smooth out” and attenuate any high frequencies that exist in the error function. The next block, and perhaps the most important one, is the *voltage-controlled oscillator* (VCO) which takes the error signal as an input voltage and uses the magnitude of this voltage to “nudge” its center frequency, ω_o, around until the error signal becomes minimized. So, the question now becomes in which direction and by how much the VCO should nudge its center frequency such that any residual phase and frequency errors are corrected. Consider the case in figure 5 where the constellation points are rotating counterclockwise around the unit circle. In this case, the VCO must correct the error by nudging its center frequency in the opposite direction (clockwise). If the frequency error is slow (small error function), the VCO will not need to nudge its center frequency by much, but if the frequency difference is relatively large (larger error function), the VCO must change its center frequency by a larger amount. Once the VCO has controlled its center frequency sufficiently close to ω_c, the error term becomes zero and in the case of BPSK, the quadrature component disappears. The result is we are left with just a scaled version of the in-phase component we can use to represent the BPSK signal.

Gnuradio Download and Install Instructions: Optional

Gnu Radio Companion is an open-source graphical user interface with python host language. The application allows the user to simulate communications systems and control many types of SDRs with the capability to display real-time performance visualizations.

1. Proceed to the following webpage: Releases · ryanvolz/radioconda (github.com)
2. Find the latest release (top of page) and under the "Assets" tab, download the executable file (.exe) "radioconda" for your computer's operating system.
3. Follow instructions to install. Once completed, you should have access to the application *gnu radio companion.*

Additional Note: It is recommended you come to lab with a printed copy of this manual since you will be using your lab computer to do the lab. You are also free to use your own computer, so long as you have access to an ethernet port for the USRP.

<u>Part 1: Clock Recovery and the MF-PFB Algorithm</u>

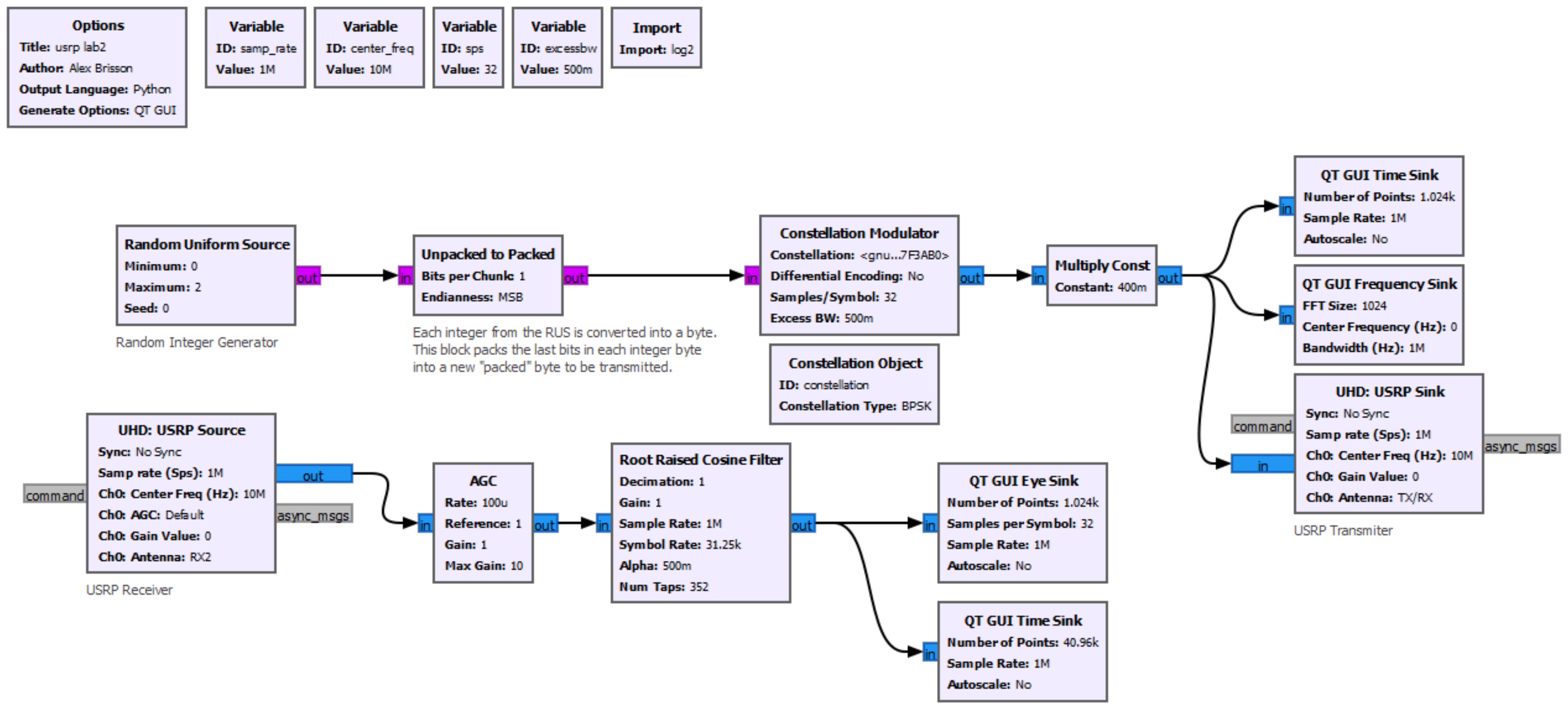
1. Open your revised BPSK file from lab #1 by selecting *file* -> *open* in gnuradio.

2. Save a copy of this file and name it something like "usrp_lab2_pskdsp".

3. Since the ADC/DAC clocks in the usrp are matched well enough, we will not need to do any *coarse* frequency corrections. Therefore, we can delete the commented blocks out of the script. These blocks are "QT GUI Range", "Signal Source", "Multiply", "Chunks to Symbols", and "Repeat". For now, we will not bother looking at the received spectrum, so let's delete that block too. To do this, click on each block and press "delete" on your keyboard.

4. We will now change the following flowgraph parameters.

   a. Set the 'excessbw' variable to a middle-of-the-road value of 0.5 for our RC pulses.
   b. As I have said, the usrp clocks are very close relative to most real systems, but to emphasize the effect, let's insert a small frequency offset between ADC/DAC clocks by adding a ½ Hz relative offset to the ADC clock in our receiver. Go to the RF Options tab in the USRP Source block and add 0.5 to the center_freq variable (center_freq +0.5). Note: this is a *very* small offset—on the order of one part in a million compared to the tuning LO frequency. As we'll see, these small offsets matter in real systems and can change all the time, therefore they must be corrected using an adaptive algorithm such as the Costas Loop.
   c. In the Time Sink GUI after the RRC Filter block, set the "Y min" and "Y max" to span the range [-2,2] and the GUI Hint to 1,0,1,2 to expand the plot out and cover more range in the simulation window. Then, click on the eye sink block and set its "Y min" and "Y max" to span the range [-2,2] and change the plot labels to "Real Part (I)" and "Imaginary Part (Q)" by tabbing over to "config" (Line 1 Label should be I and Line 2 Label should be Q). After you are finished making changes, your flowgraph should look like the following figure.



Options
Title: usrp lab2
Author: Alex Brisson
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: center_freq
Value: 10M

Variable
ID: sps
Value: 32

Variable
ID: excessbw
Value: 500m

Import
Import: log2

Random Uniform Source
Minimum: 0
Maximum: 2
Seed: 0

Random Integer Generator

Unpacked to Packed
Bits per Chunk: 1
Endianness: MSB

Each integer from the RUS is converted into a byte.
This block packs the last bits in each integer byte
into a new "packed" byte to be transmitted.

Constellation Modulator
Constellation: <gnu...7F3AB0>
Differential Encoding: No
Samples/Symbol: 32
Excess BW: 500m

Constellation Object
ID: constellation
Constellation Type: BPSK

Multiply Const
Constant: 400m

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 1M
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

UHD: USRP Sink
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: Gain Value: 0
Ch0: Antenna: TX/RX

USRP Transmiter

UHD: USRP Source
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: AGC: Default
Ch0: Gain Value: 0
Ch0: Antenna: RX2

USRP Receiver

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 10

Root Raised Cosine Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Symbol Rate: 31.25k
Alpha: 500m
Num Taps: 352

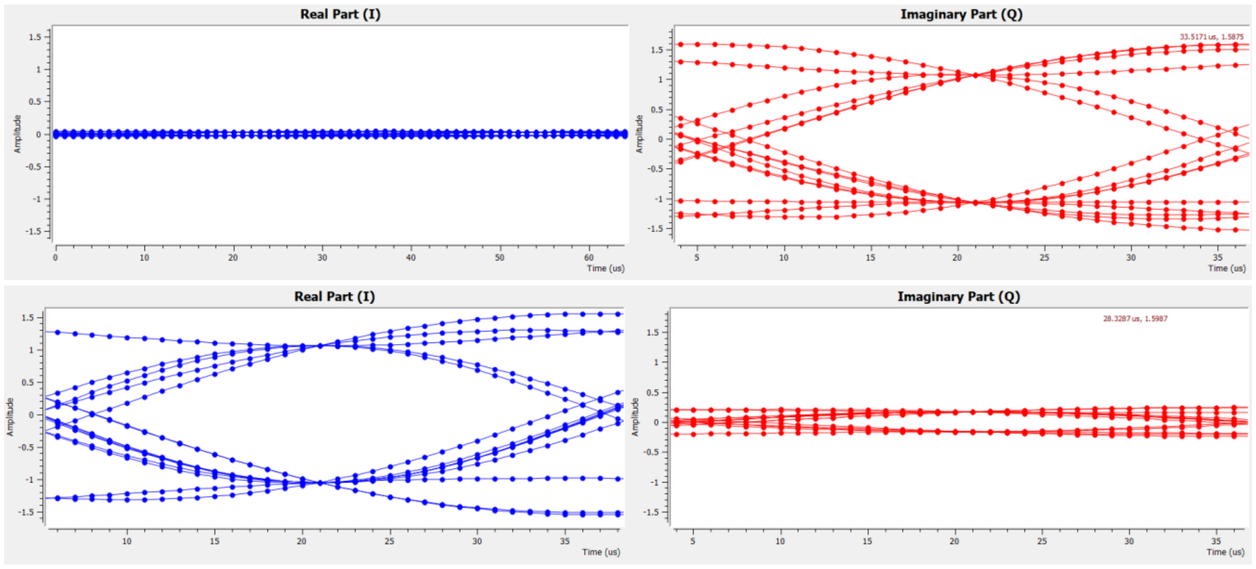
QT GUI Eye Sink
Number of Points: 1.024k
Samples per Symbol: 32
Sample Rate: 1M
Autoscale: No

QT GUI Time Sink
Number of Points: 40.96k
Sample Rate: 1M
Autoscale: No

5. Go ahead and run the file. You should see the in-phase and quadrature eye diagrams sloshing back and forth in amplitude as a result of this slow residual frequency offset between clocks. Before attempting to correct this frequency offset using the Costas Loop, let's first look to see how our receiver clock is doing in terms of sampling at the peak of our symbols.

6. Open the scope menu on the eye diagrams by clicking down on the mouse wheel. Plot the ADC samples for the in-phase part (I) by going to "Real Part (I)" -> "Line Marker" -> "Circle". Do this for the quadrature part "Imaginary Part (Q)" as well. Stop the acquisition when the sloshing reaches a maximum using the scope menu and zoom in on the eye of the eye diagram. <u>How accurate was the clock at sampling the peak of the symbols for the in-phase and quadrature signals? Could it be better?</u>

#6 Results



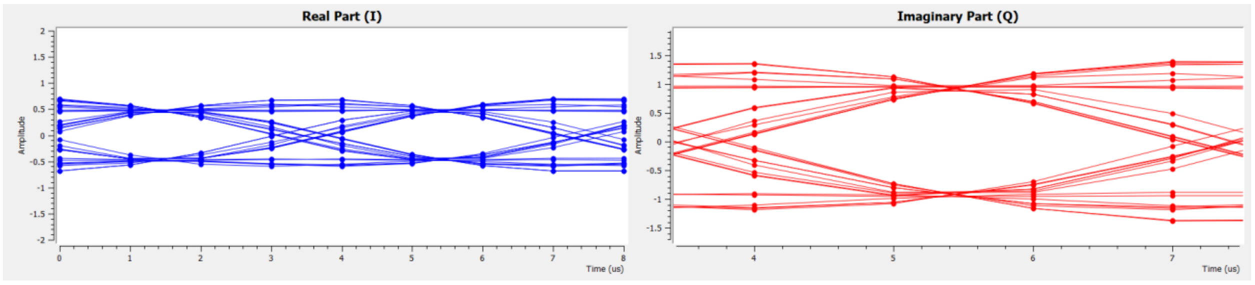
32 samples per symbol

The ADC seems to sample the pulses very close to their maximum, this is evident with 32 samples per symbol, however the sample closest to the peak doesn't seem to occur exactly at the peak of the transition in the real and imaginary parts.

7. Let's now try reducing the samples per symbol variable 'sps' to 4. Click on the variable 'sps' and set its value to 4 and reduce the number of points in the eye diagram to 256. Rerun the file and complete the steps in #6. <u>Compare the cases in #'s 6 and 7. What differences do you notice and was it expected?</u> Nyquist tells us that in order to reconstruct a signal in the time domain, or rather completely extract all the frequency content of a signal without aliasing, we must sample the signal at least twice in the duration of its period i.e., the sample rate must be at least twice the bandwidth of the signal. This means that we must take at least 2 samples per symbol duration. Try reducing the 'sps' variable to 2 and rerun the file. <u>Explain what happens to the sampling instances and</u>

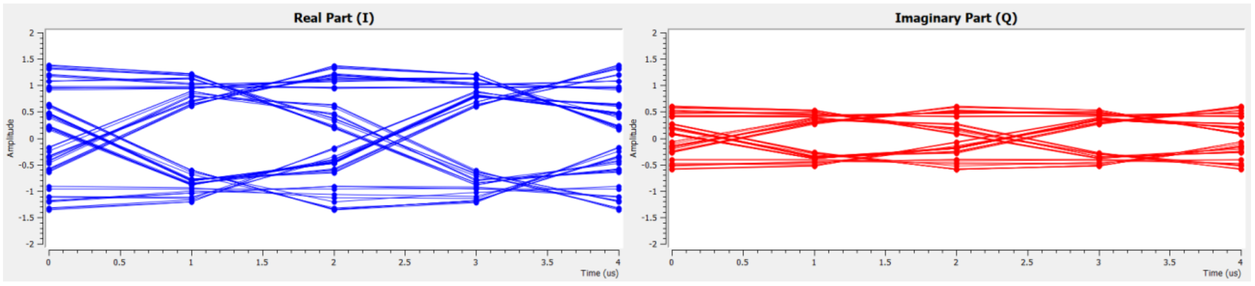
compare to the previous cases. What must we do in a “practical sense” in order to take advantage of lower bandwidth clocks but still gain sufficient time resolution to pick the *optimal* sample? Think back to the introduction.

#7 Results



4 samples per symbol

With 4 samples per symbol, we can clearly see the ADC is no longer sampling close to the peak of the symbol. This is expected, with fewer samples per symbol compared to #5, the peak of the pulse will likely not be sampled by the ADC and demonstrates the need for interpolation.

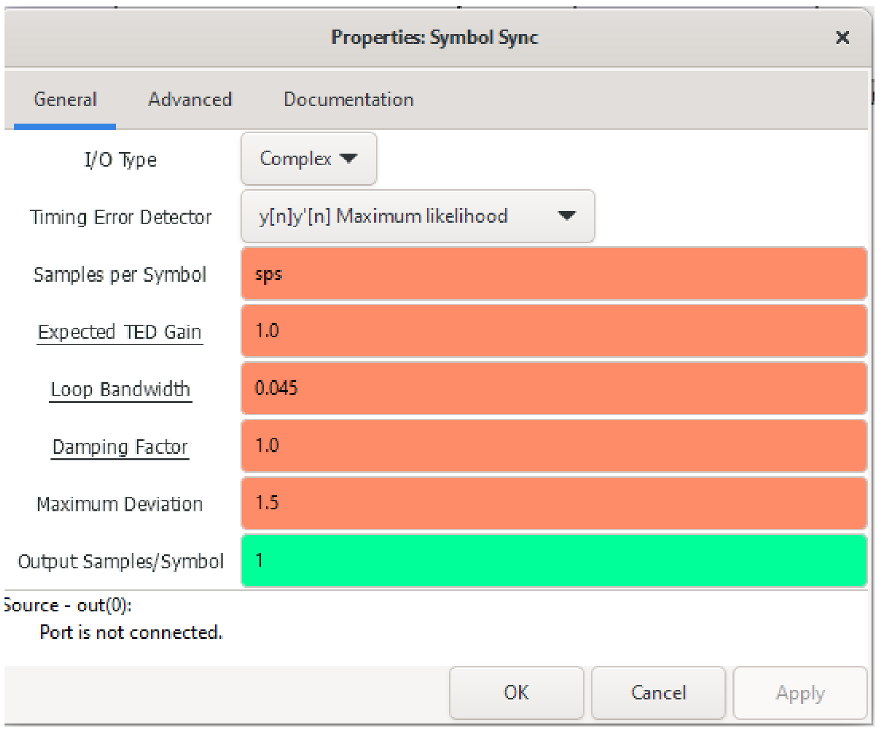


2 samples per symbol

With 2 samples per symbol, the need for interpolation becomes even more apparent, however Nyquist states this sampling is sufficient for extracting all the information from the signal and the peak can therefore be found by implementing more interpolation.

8. Let’s now try and use the Matched Filter Polyphase Filter Bank algorithm to pick out the optimal sample and observe the result in the constellation diagram. Search the block “Symbol Sync” and place it in the flowgraph. We will now need to create some taps and variable blocks for the filter bank. Search “variable” and place a new variable block and set its ID and value to “nfilts” and 32, respectively. This variable will be used to set the number of phase-shifted filters we will have in our “bank” of filters. Search “RRC Filter Taps” and place the block into the flowgraph. This block will create the taps or the weighted impulses that make up the RRC filter’s impulse response for the convolution operation that the MF-PFB will do at the receiver. You will need to set all the parameters correctly in both the Symbol Sync and RRC Filter Taps blocks, so I’ve included a few

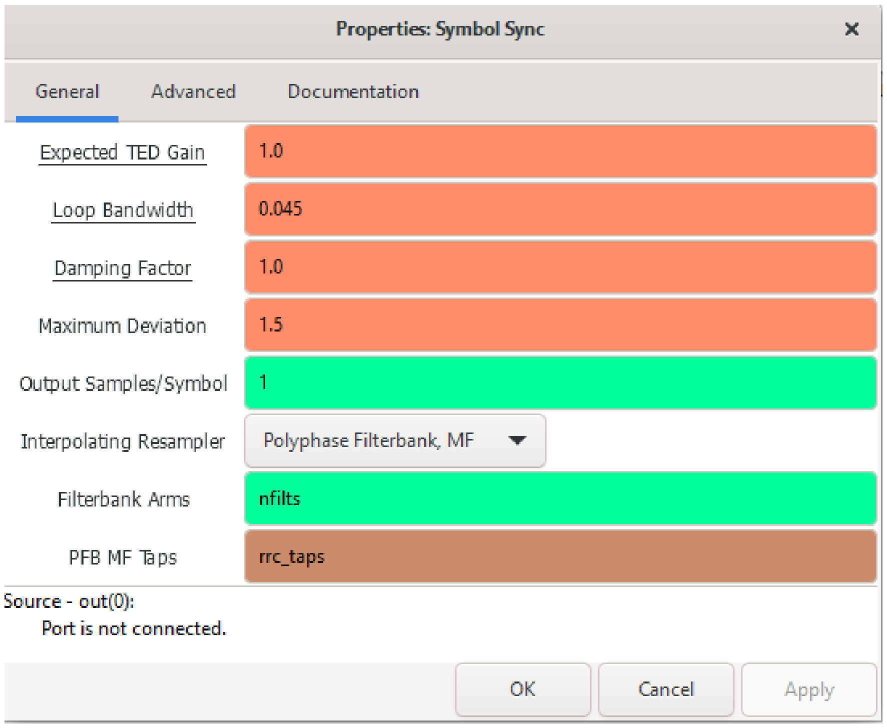
figures for you to use in order to set the values and IDs—see below. Many of these values are default, but some are experimentally derived. For our purposes, we will just use the default values.



I/O Type	Complex
Timing Error Detector	y[n]y'[n] Maximum likelihood
Samples per Symbol	sps
Expected TED Gain	1.0
Loop Bandwidth	0.045
Damping Factor	1.0
Maximum Deviation	1.5
Output Samples/Symbol	1

Source - out(0):
Port is not connected.

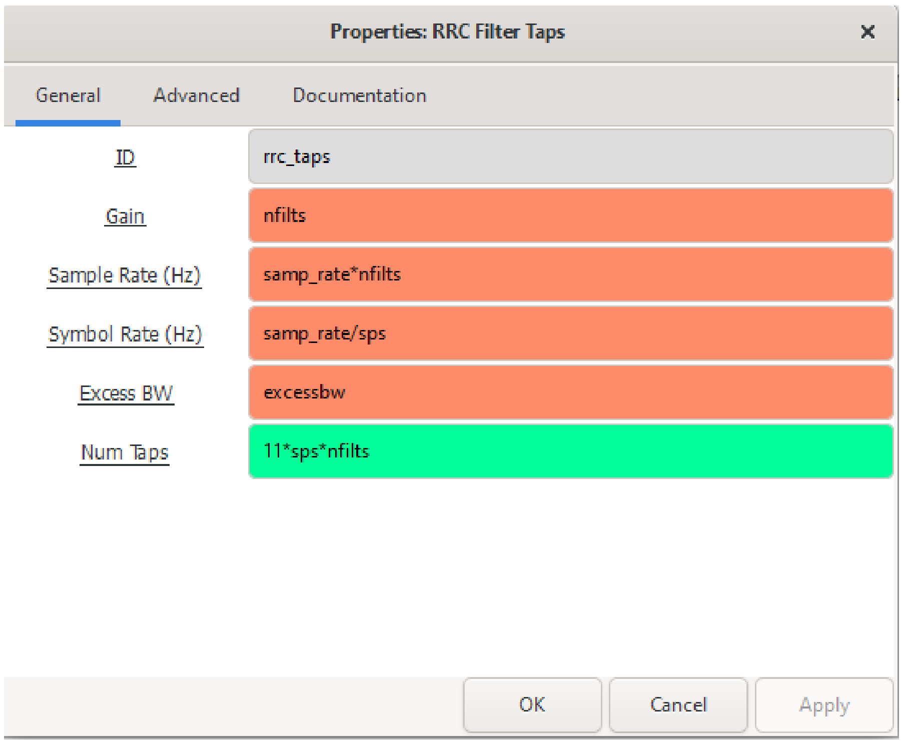
Figure 8: Symbol Sync Block Variables #1



Expected TED Gain	1.0
Loop Bandwidth	0.045
Damping Factor	1.0
Maximum Deviation	1.5
Output Samples/Symbol	1
Interpolating Resampler	Polyphase Filterbank, MF
Filterbank Arms	nfilts
PFB MF Taps	rrc_taps

Source - out(0):
Port is not connected.

Figure 9: Symbol Sync Block Variables #2

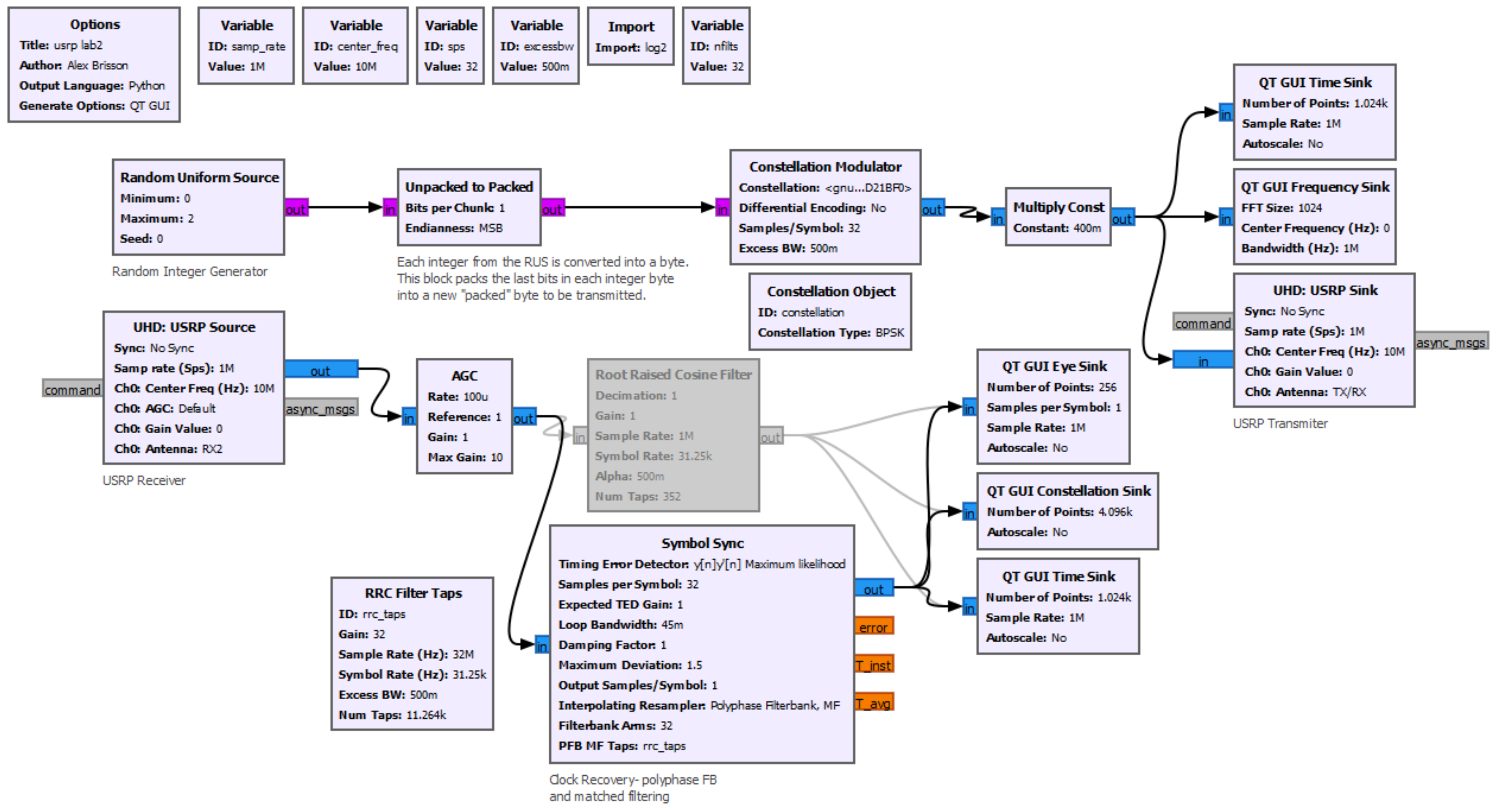


ID	rrc_taps
Gain	nfilts
Sample Rate (Hz)	samp_rate*nfilts
Symbol Rate (Hz)	samp_rate/sps
Excess BW	excessbw
Num Taps	11*sps*nfilts

Figure 10: RRC Filter Taps Variables and ID

9. If the MF-PFB block does its job as suggested in the introduction, we should expect the "Symbol Sync" block to output a single sample which occurred at the *optimal* sampling instant to represent the amplitudes of our I and Q pulses. Now, let's set up the visualization blocks to see this happening in real time. First, comment out the "Root Raised Cosine Filter" because we know that our MF-PFB will do the matched filtering for us, and we no longer need the block. Connect the AGC block to the Symbol Sync block's input. Then, make a connection from the Symbol Sync to the Eye and Time Sync blocks. Since the Symbol Sync block only outputs 1 sample per symbol at the optimal sampling instant, change the samples per symbol parameter to 1 in the eye sink block.

10. Search the block "QT GUI Constellation Sink" and place it in the flowgraph. Set the number of points to 1024*4 and connect the output of the Symbol Sync block to the input of this block. Reset the value of 'sps' to 32. Once you have competed all the changes, your flowgraph should look like the following figure.



Options
Title: usrp lab2
Author: Alex Brisson
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: center_freq
Value: 10M

Variable
ID: sps
Value: 32

Variable
ID: excessbw
Value: 500m

Import
Import: log2

Variable
ID: nfilts
Value: 32

Random Uniform Source
Minimum: 0
Maximum: 2
Seed: 0

Random Integer Generator

Unpacked to Packed
Bits per Chunk: 1
Endianness: MSB

Each integer from the RUS is converted into a byte.
This block packs the last bits in each integer byte
into a new "packed" byte to be transmitted.

Constellation Modulator
Constellation: <gnu...D21BF0>
Differential Encoding: No
Samples/Symbol: 32
Excess BW: 500m

Constellation Object
ID: constellation
Constellation Type: BPSK

Multiply Const
Constant: 400m

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 1M
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

UHD: USRP Sink
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: Gain Value: 0
Ch0: Antenna: TX/RX

USRP Transmiter

UHD: USRP Source
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: AGC: Default
Ch0: Gain Value: 0
Ch0: Antenna: RX2

USRP Receiver

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 10

Root Raised Cosine Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Symbol Rate: 31.25k
Alpha: 500m
Num Taps: 352

QT GUI Eye Sink
Number of Points: 256
Samples per Symbol: 1
Sample Rate: 1M
Autoscale: No

QT GUI Constellation Sink
Number of Points: 4.096k
Autoscale: No

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 1M
Autoscale: No

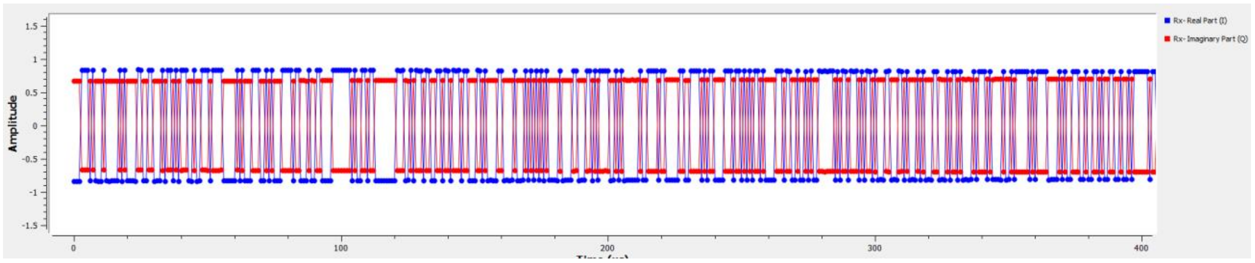
RRC Filter Taps
ID: rrc_taps
Gain: 32
Sample Rate (Hz): 32M
Symbol Rate (Hz): 31.25k
Excess BW: 500m
Num Taps: 11.264k

Symbol Sync
Timing Error Detector: y[n]y'[n] Maximum likelihood
Samples per Symbol: 32
Expected TED Gain: 1
Loop Bandwidth: 45m
Damping Factor: 1
Maximum Deviation: 1.5
Output Samples/Symbol: 1
Interpolating Resampler: Polyphase Filterbank, MF
Filterbank Arms: 32
PFB MF Taps: rrc_taps

Clock Recovery- polyphase FB
and matched filtering

11. Run the file and first observe the time domain IQ data after the symbol sync block. Plot the samples of this data using the scope menu and notice that we are no longer looking at RC pulses. Each impulse represents the *optimal* samples of the IQ waveforms.

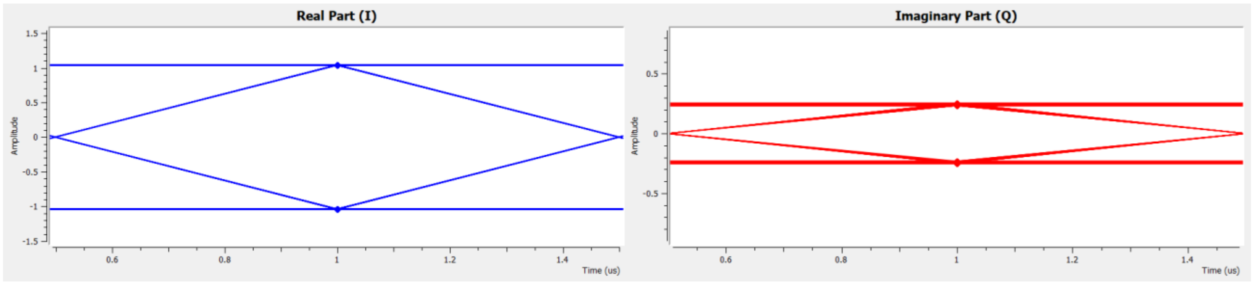
#11 Results



Time Domain View of 1sps I and Q after Clock Synchronization – still residual frequency/phase offsets.

12. Next, observe the eye diagrams for I and Q. <u>Does the Symbol Sync block really give you the sample right at the peak of the symbol transitions? How do you know?</u>

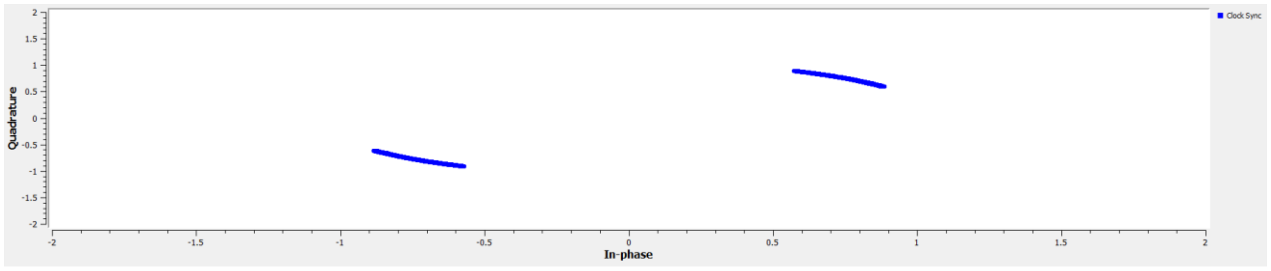
#12 Results



Single Sample occurs right at the peak of the transitions, where the time derivative of P(t) is zero.

13. To fully answer the question in 11, look at the constellation diagram. You should see the constellation "points" or the instantaneous "superpositions" of I and Q rotating at roughly 1/2 rotations per second with some phase error. <u>What is unique about this constellation, in other words, what is significant about the magnitudes of all these points and is it expected? Again, think back to the introduction.</u>

#13 Results



Constellation Diagram after Clock Synchronization- points are smeared with slow and positive frequency error

The points are orbiting the origin along the unit circle, which means we must be picking samples right at the peaks of I and Q, despite the clock frequency error and phase errors between them.

Part 2: Fine Frequency and Phase Error Correction using the Costas Loop Algorithm

1. Before we correct for the fine frequency and phase errors using the Costas Loop, let's first look more into this sloshing around of the I and Q signals and confirm the small frequency error between clocks. To start, comment out the time sink block after the symbol sync block. Search and place another time sink block just after the AGC block. We need to set the number of points in the time sink to be high enough so that we can see a full half period of the slow modulation happening as a result of the clock offset. The time sink outputs data at the sample rate, which is 1 MHz, so in order to see a full half period of the offset which is on the order of Hz we need at least ~1,000,000 points (i.e., with samples happening every micro-second). Therefore set the number of points value in the time sink to 1,200,000 by typing "int(1.2e6)" to see just beyond the half-period transitions. Set the Y limits to cover the range [-2,2] and turn the grid on. Run the file and stop the acquisition once a full half-period comes into view. You should see the following time plot.

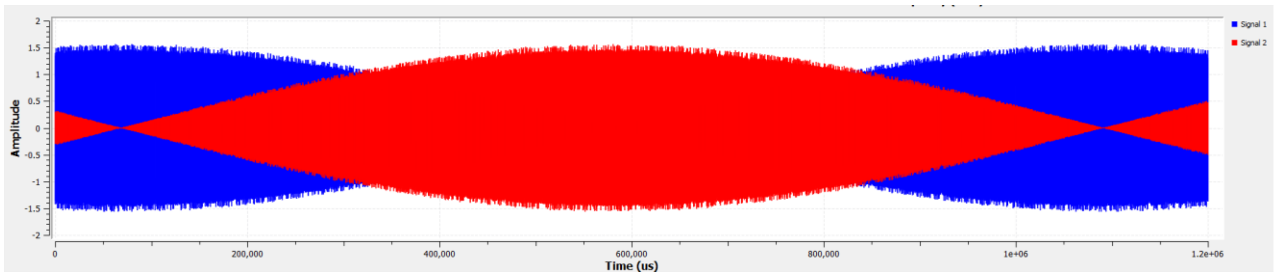


Figure 11: Fine Frequency Error Analysis

2. You now need to zoom into the beginning and end transitions and record the time stamps to confirm the frequency error using your cursor. Take two-time stamps at the beginning and end of the half period, subtract the two, and double the result. This is the slow periodic modulation on our signal in units of micro-seconds. Confirm its value in units of Hz. <u>Is the result expected and why?</u>

Record Time Stamp at Transitions

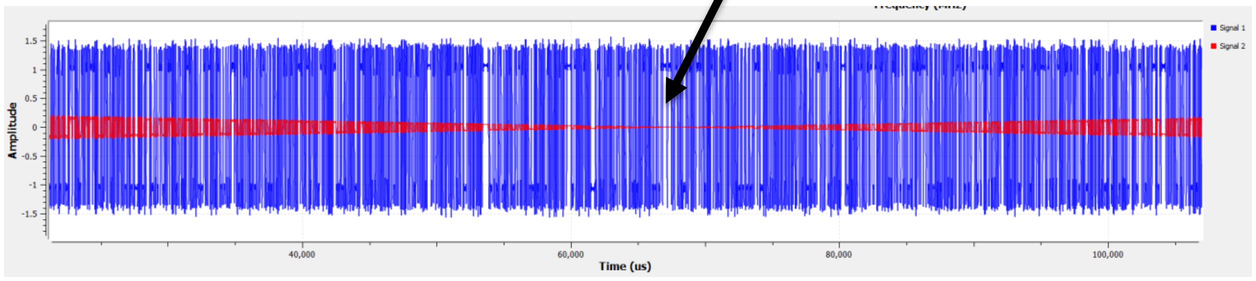
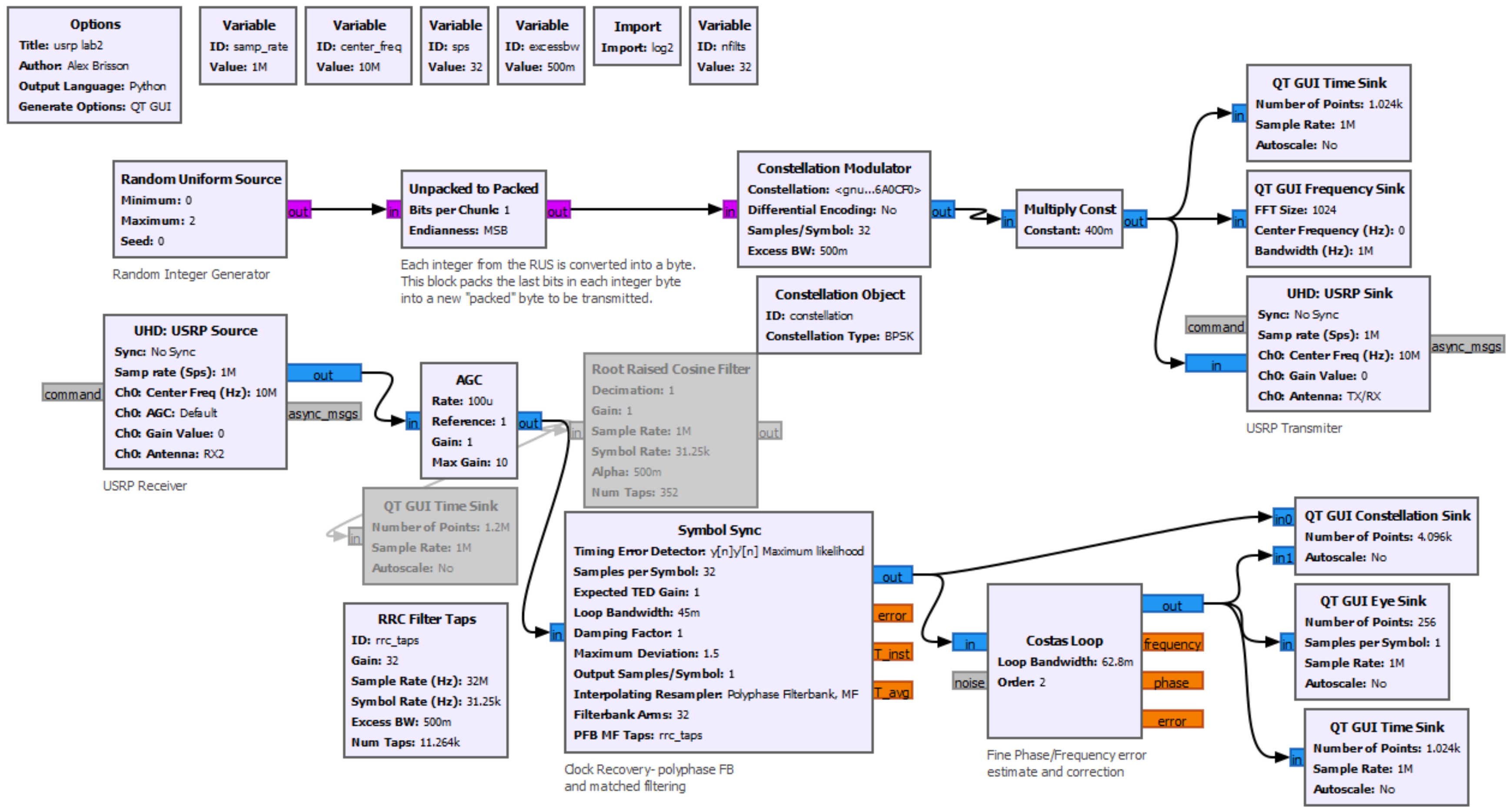


Figure 12: Recording Time Stamps

#2 Results

The half period spans ~88336μs - ~1110998μs. The difference is ~1.023s. This is a half-period duration, so double the value and divide by 1 to obtain the frequency error. $\frac{1}{2.046s} \sim 0.489\ Hz$. Neglecting error in recording the actual time stamps, we might expect the *actual* frequency error to be around ~0.01 Hz.

3. We will now use the Costas Loop to compensate the frequency and phase errors in the constellation. Comment out the time sink after the AGC and uncomment the time sink after the symbol sync. Search and add the "Costas Loop" block and connect it after the Symbol Sync block. Change the Costas Loop Bandwidth parameter to 62.8e-3 and type "constellation.arity()" into the Order field. Open the constellation sink block and set the number of inputs to 2, then, tab over to config and type "Clock Sync" in the Line 1 Label field and "Costas" in the Line 2 Label field. Connect the Costas Loop output to the second input port "in1" of the constellation sink. Disconnect the eye and time sinks from the symbol sync and reconnect them to the output of the Costas Loop. Once you are finished your flowgraph should look like the following figure.



Options
Title: usrp lab2
Author: Alex Brisson
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: center_freq
Value: 10M

Variable
ID: sps
Value: 32

Variable
ID: excessbw
Value: 500m

Import
Import: log2

Variable
ID: nfilts
Value: 32

Random Uniform Source
Minimum: 0
Maximum: 2
Seed: 0

Random Integer Generator

Unpacked to Packed
Bits per Chunk: 1
Endianness: MSB

Each integer from the RUS is converted into a byte. This block packs the last bits in each integer byte into a new "packed" byte to be transmitted.

Constellation Modulator
Constellation: <gnu...6A0CF0>
Differential Encoding: No
Samples/Symbol: 32
Excess BW: 500m

Constellation Object
ID: constellation
Constellation Type: BPSK

Multiply Const
Constant: 400m

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 1M
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

UHD: USRP Sink
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: Gain Value: 0
Ch0: Antenna: TX/RX

USRP Transmiter

UHD: USRP Source
Sync: No Sync
Samp rate (Sps): 1M
Ch0: Center Freq (Hz): 10M
Ch0: AGC: Default
Ch0: Gain Value: 0
Ch0: Antenna: RX2

USRP Receiver

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 10

Root Raised Cosine Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Symbol Rate: 31.25k
Alpha: 500m
Num Taps: 352

QT GUI Time Sink
Number of Points: 1.2M
Sample Rate: 1M
Autoscale: No

RRC Filter Taps
ID: rrc_taps
Gain: 32
Sample Rate (Hz): 32M
Symbol Rate (Hz): 31.25k
Excess BW: 500m
Num Taps: 11.264k

Symbol Sync
Timing Error Detector: y[n]y'[n] Maximum likelihood
Samples per Symbol: 32
Expected TED Gain: 1
Loop Bandwidth: 45m
Damping Factor: 1
Maximum Deviation: 1.5
Output Samples/Symbol: 1
Interpolating Resampler: Polyphase Filterbank, MF
Filterbank Arms: 32
PFB MF Taps: rrc_taps

Clock Recovery- polyphase FB and matched filtering

Costas Loop
Loop Bandwidth: 62.8m
Order: 2

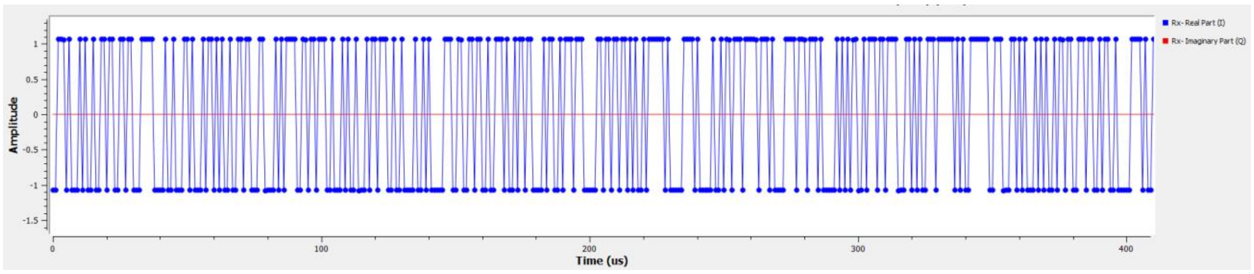
Fine Phase/Frequency error estimate and correction

QT GUI Constellation Sink
Number of Points: 4.096k
Autoscale: No

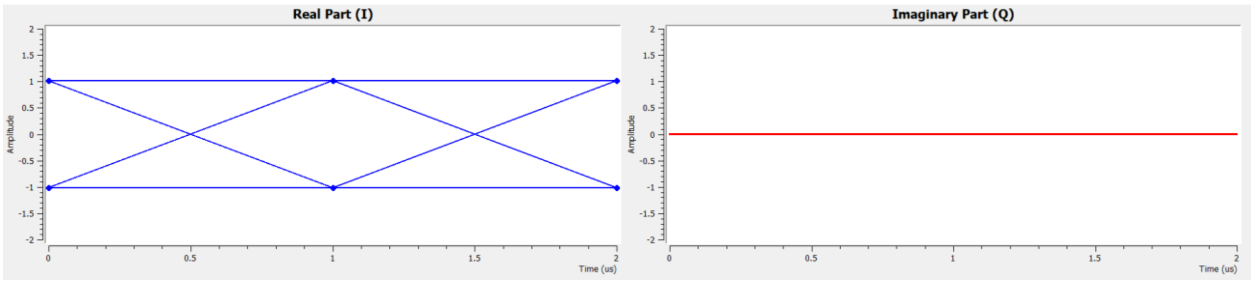
QT GUI Eye Sink
Number of Points: 256
Samples per Symbol: 1
Sample Rate: 1M
Autoscale: No

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 1M
Autoscale: No

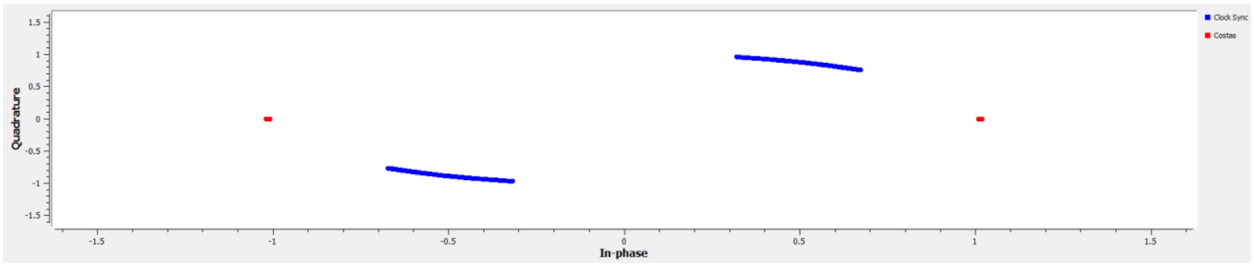
4. Run the flowgraph and observe the convergence of the Costas Loop. Describe what happened/changed to the time domain and eye diagrams. What about the constellation diagram? Is this constellation what you expected? Feel free to capture any screen shots you think would be helpful in justifying your results.



The time domain represents the decoded symbols in the real part (I-component) representing +1s and -1s without any amplitude fluctuations while the Q-component is zero as expected. This is due to the Costas Loop correcting the fine frequency and phase errors.



The eye diagrams are no longer sloshing around in amplitude, and the optimal sample of the I-component is taken.



The Costas Loop (red) points are converging to the ideal symbol locations for BPSK while the clock recovery (blue) points are still orbiting around with phase and frequency errors.

5. As a fun exercise, and for extra credit, try playing around with the small frequency offset in the USRP source block (go in small steps up to 1 Hz). What happens if you increase/decrease the offset? What about making the offset negative? How does making the offset negative relate back to lab #1, specifically, Euler's formula and the *necessity* of complex numbers?

The students should notice the change in direction and speed of the points after clock synchronization is changing as they increase or change the sign of the offset. This effect must be compensated by the VCO in the Costas loop in order to converge to the correct symbol locations.