Project 2: SDR Text Transmission

Author:	Name：高荣、曹正阳　　Student ID: 12110301, 12110623

Introduction:

1. Basic principles of Digital Carrier Wave Transmission

Basic principles:

- **I/Q signal representation:** In digital communication, complex baseband signal representation is a method of representing a modulated signal in the complex plane. This involves using two channels to represent the signal, with one channel representing the real part and the other channel representing the imaginary part. This is often used in digital communication systems such as wireless communication and digital television broadcasting.

  By using complex baseband signal representation, it is possible to simplify the signal processing and to take advantage of the mathematical properties of complex numbers. In particular, complex baseband signal representation enables the use of linear and circular modulation techniques, which can be used to transmit digital information over a communication channel.

  The complex baseband signal representation can be expressed as follows:

  $s(t) = I(t) + jQ(t)$

  where s(t) is the complex baseband signal, I(t) is the in-phase component of the signal and Q(t) is the quadrature component of the signal. These components are combined to produce a single signal that represents the modulated signal.

  The I and Q channels are typically generated using a phase shift keying (PSK) or quadrature amplitude modulation (QAM) technique, and then combined to produce the complex baseband signal. The complex baseband signal can then be upconverted to a higher frequency using a mixer, and then transmitted over a communication channel.

- **Data Encapsulation:** Data encapsulation is a term used in digital communication to describe the process of including data in a specific format that allows it to be transmitted over a network. The data is wrapped in layers of protocol headers, which provide information about the type of data, the origin, and the destination of the data. Encapsulation helps to ensure that data is transmitted correctly over a network.

  In a network, data is sent in the form of packets. These packets are composed of various elements, including a header and a payload. The header provides information about the packet, such as the source and destination addresses and the protocol being used. The payload contains the actual data being transmitted.

  When data is encapsulated, the payload is wrapped in a header that contains additional information, such as the type of data being sent. The header and payload together make up the packet, which is then transmitted over the network. At the receiving end, the packet is de-encapsulated, and the data is extracted from the payload.

- **PSK Modulation and Demodulation**: Phase-shift keying (PSK) is a digital modulation technique where the phase of a carrier wave is changed between two or more values to represent information. In PSK modulation, a digital signal (which is a sequence of 1s and 0s) is modulated onto a carrier wave of a fixed frequency by periodically changing the phase of the carrier. This creates a phase-shifted waveform where the carrier is shifted by a certain angle based on the binary value of the digital signal.

  For example, with binary PSK (BPSK), a binary '1' would shift the phase of the carrier by 180 degrees, while a binary '0' would leave the carrier phase unchanged. With quadrature PSK (QPSK), there are four different possible phase shift angles: 0, 90, 180, and 270 degrees, which allows for two bits to be transmitted with each shift.

  At the receiver's end, the PSK signal is demodulated by comparing the phase of the received signal to a locally generated carrier wave of the same frequency and phase, and determining the binary value depending on whether the phase shift is detected for each symbol. This is achieved through a demodulator circuit which compares the phase of the received signal with the phase of a locally generated carrier wave.
- **Frame Synchronization Principle:** Frame synchronization is an essential process in communication systems that helps us to accurately extract information from a received signal. In communication systems, information is transmitted in frames or packets, with each frame typically consisting of several information-bearing symbols. A frame synchronization module identifies the beginning of each frame in a received signal, enabling us to extract the information from it.

  In LabVIEW, we can implement frame synchronization by using correlation techniques in combination with thresholding. We correlate the incoming signal with the expected preamble sequence to detect the presence of the preamble. Once the trailing edge of the preamble is detected, we know the frame's beginning and can start extracting the information bits.
- **Queue:** In LabVIEW, a queue is a simple way to transfer data between multiple parallel threads or loops in a LabVIEW program. A queue is a First-In-First-Out (FIFO) buffer that can store a variable number of elements. A producer loop adds elements to the queue, while a consumer loop removes elements from the queue.

  Queues in LabVIEW allow for asynchronous communication between parallel loops, meaning that both the producer and consumer loops can operate independently of each other and at their own rate. This can be useful in applications where timing is not critical or where different loops need to operate at different rates or on different data.
- **Producer-Consumer model:** Producer-Consumer model is a common design pattern in LabVIEW communication programs, and it refers to a communication mechanism between two or more parallel loops that allows one loop (producer) to generate data and send it to another loop (consumer) for processing.

  In the Producer-Consumer model, the producer loop generates data and places it in a queue, while the consumer loop dequeues the data and processes it. The queue acts as a buffer between the producer and consumer loops, ensuring that data is transferred efficiently and without errors.

2.LabVIEW Express Module:

- **MT Modulate PSK Module:**

  The MT Modulate PSK (Phase Shift Keying) module in LabVIEW is used to modulate a digital signal by altering the phase of a sine wave carrier. PSK can be used to modulate binary data (2-PSK) or quadrature data (QPSK) with a constellation diagram that has four points.

  The function of MT Modulate PSK is to modulate a binary digital signal into a phase-shift keyed (PSK) signal. This modulation scheme is commonly used in digital communication systems to transmit data over radio, satellite, or fiber-optic communication channels. The function operates by mapping each binary symbol onto a phase shift of a sinusoidal carrier wave, such that the phase of the wave corresponds to the value of the binary symbol. The MT Modulate PSK function supports various PSK modulation types, such as binary PSK, quadrature PSK (QPSK), and higher-order PSK schemes, which can improve the data rate and error performance of the system.

- **MT DeModulate PSK Module:**

  In digital communication systems, PSK (Phase-Shift Keying) modulation is a common way to transmit digital data over a carrier signal. And, in order to receive the data on the receiver side, the received signal needs to be demodulated. MT DeModulate PSK module in LabVIEW is a module that is used to demodulate a Phase Shift Keying (PSK) modulated signal. It uses the principle of phase detection to demodulate the signal in real-time, and it is particularly suitable for demodulating binary PSK signals (BPSK), quadrature phase shift keying signals (QPSK), or M-ary PSK signals (MPSK). To use the MT DeModulate PSK module, the input signal is first passed through a matched filter to remove noise and then the module performs demodulation using a reference wave generated by a local oscillator. The module generates a binary signal output, which represents the demodulated bits of the original signal. The module is quite useful since it reduces the complexity of the demodulation process, which traditionally required complex mathematical equations to be performed manually. The MT Demodulate PSK module in LabVIEW is a convenient way to demodulate PSK signals for further processing or display purposes.

- **MT Generate System Parameters Module:**

  The "MT Generate System Parameters" module in LabVIEW is used to generate system parameters for frame synchronization in communication systems. Frame synchronization is crucial in communication systems to ensure that transmitted data is correctly aligned and received by a receiver. This module generates system parameters for frame synchronization based on the "Maximum-Likelihood Timing Recovery" method. The module computes key system parameters such as timing recovery filter coefficients, timing error detector coefficients, interpolation factor, and decimation factor. These parameters are used to synchronize signals and make sure that the transmitted data is correctly aligned with the receiving system. To use the "MT Generate System Parameters" module in LabVIEW, we would typically input the sampling frequency, desired interpolation and decimation factors, and the length of the timing filter. we would also need to specify the timing error detector algorithm and the type of signal we are processing (real or complex). Once the module has generated the system parameters, we can use them to synchronize signals and ensure that the transmitted data is correctly received.

- **MT Generate Filter Coefficients Module:**
  The MT Generate Filter Coefficients module in LabVIEW is a tool used to generate coefficients for Finite Impulse Response (FIR) filters, which are used in digital signal processing applications. FIR filters can be used to selectively filter out unwanted parts of a digital signal, such as noise or other disturbances. The MT Generate Filter Coefficients module takes in user-defined parameters such as filter type, passband frequency, stopband frequency, and filter order, and then uses these parameters to generate the corresponding filter coefficients. These coefficients can then be used in LabVIEW or other software to implement the FIR filter and perform signal processing. The module offers several options for filter type, including low-pass, high-pass, band-pass, and band-stop, allowing for a range of filter designs to be created. The MT Generate Filter Coefficients module is useful for engineers and researchers who work with digital signals and need an efficient way to design and implement FIR filters.
- **MT Generate Synchronization Parameters Module:**
  The MT Generate Synchronization Parameters module in LabVIEW is a tool for generating synchronization parameters for frame synchronization operations in communication systems.
  When transmitting data over a communication channel, it is important to ensure that the sender and receiver are synchronized with each other to avoid errors in data transmission. The MT Generate Synchronization Parameters module helps achieve this synchronization by generating a set of specific synchronization parameters that can be used by both the sender and receiver systems to synchronize their communications.
  The synchronization parameters generated by this module include the Frame Start, Word Start, and Bit Start Values. These values are used to determine the exact starting point of each frame being transmitted and received. The module also provides features for setting parameters such as the data rate, polarity, and edge to use for synchronization.
  Overall, the MT Generate Synchronization Parameters module is a powerful tool for synchronization in communication systems built using LabVIEW, allowing for more efficient and reliable communication.
- **MT Format Eye Diagram Module:**
  The MT Format Eye Diagram module in LabVIEW is a tool used for signal analysis in digital communication systems. The eye diagram is a common technique to analyze the quality of a digital communication signal. The module provides a graphical representation of the signal waveform, which is useful for determining the signal's quality and identifying potential issues.
  The eye diagram module in LabVIEW generates an eye diagram by overlaying multiple copies of the signal waveform on top of each other, allowing observation of how the signal changes over time.

The eye diagram displays a 2D representation of the signal that resembles an eye, hence its name. The opening and closing of the eye represent the signal's transition between different states, and its shape provides insight into jitter, noise, and distortion.

The MT Format Eye Diagram module in LabVIEW allows users to adjust parameters such as the signal's amplitude, frequency, and phase to visualize the signal's eye diagram. It also offers features that allow the user to view and analyze the signal's statistical properties, such as the signal's rise time, fall time, amplitude, and noise.

Overall, the MT Format Eye Diagram module in LabVIEW provides a powerful tool for analyzing digital communication signals and helps users identify potential signal issues that may affect data transmission and reception quality.

- **MT Format Constellation Module:**

The MT Format Constellation module in LabVIEW is used to show the constellation of a digital signal. A constellation diagram represents the relationship between the bits in a digital signal and the analog signal that is transmitted over a communication channel. The MT Format Constellation module in LabVIEW is a graphical tool that displays the position of each transmitted symbol in a signal against a reference constellation. The constellation diagram is used to visually inspect the digital signal to ensure correct transmission, reception, and decoding of the message.

MT Format Constellation module can be used with a variety of digital modulation techniques, such as phase shift keying (PSK), amplitude shift keying (ASK), and quadrature amplitude modulation (QAM). The module allows we to configure the reference constellation that we want to use, such as a square box, circular, or any custom shape that we need.

To use the MT Format Constellation module in LabVIEW, we will need to provide the digital signal that we want to visualize as an input. The module then maps that signal to the appropriate symbols in the reference constellation and displays the result graphically. This tool is particularly useful in communication systems for troubleshooting and optimizing symbol timing, phase detection, and other system parameters.

- **sub_PSKMod_Temp Module:**

The "sub_PSKMod_Temp" module in LabVIEW is a sub-VI (sub-Virtual Instrument) that is used to perform phase-shift keying (PSK) modulation. In digital communication, PSK is a modulation scheme used to transmit digital data over a communication channel by varying the phase of a carrier wave. The sub_PSKMod_Temp module is designed to take binary data and modulate it onto a carrier wave using PSK modulation.

This module has a number of input and output terminals, including a binary input signal terminal, an output carrier signal terminal, and a phase offset terminal. The binary input signal terminal is used to input the digital data that needs to be modulated. The carrier signal output terminal generates the modulated signal. The phase offset terminal allows the user to specify the phase shift applied to each symbol which is essential for differentiating between two adjacent symbols.

Overall, the sub_PSKMod_Temp module is an essential component of any digital communication system that utilizes PSK modulation, allowing the user to efficiently modulate digital data onto a carrier wave in LabVIEW.

- **sub_resample_and_demodulate_Temp Module:**

  The "sub_resample_and_demodulate_Temp" module in LabVIEW is a subVI (Sub-Virtual Instrument) that can be used to perform sub-sampling, demodulation, and filtering of a complex baseband signal.

  Sub-sampling is the process of reducing the sampling rate of a signal, which can be useful when processing large sets of data or conserving computational resources. Demodulation is the process of extracting the modulating information, such as the amplitude or frequency variations, from a modulated carrier signal. Filtering is the process of removing unwanted frequencies or noise from a signal. The "sub_resample_and_demodulate_Temp" module takes as input a complex baseband signal, which is represented as a series of in-phase and quadrature (I/Q) samples. It then performs sub-sampling to reduce the number of samples in the signal, demodulation to extract the underlying information from the signal, and filtering to remove any noise or unwanted frequencies.

- **subCallConfigTx Module:**

  The SubCallConfigTx module in LabVIEW is a sub-VI (subroutine) used to configure the transmission of data through a specific wireless communication protocol known as the WCDMA (Wideband Code Division Multiple Access) protocol. This module is typically used in communication systems that rely on WCDMA for data transmission, such as cellular networks.

  This sub-VI allows the user to configure various transmission parameters such as the modulation technique, data rate, transmit power, and frame synchronization for the WCDMA system. These settings can be adjusted to optimize the transmission of data based on factors such as channel conditions, available bandwidth, and signal interference.

- **subCallConfigRx Module:**

  The SubCallConfigRx module in LabVIEW is a sub-VI that is used to configure the reception of data through a specific wireless communication protocol known as WCDMA (Wideband Code Division Multiple Access). This module is typically used in communication systems that rely on WCDMA for data reception, such as cellular networks.

  This sub-VI allows the user to configure various reception parameters such as the modulation technique, data rate, receive power, and carrier frequency for the WCDMA system. These settings can be adjusted to optimize the reception of data based on factors such as channel conditions, available bandwidth, and signal interference.

- **Obtain Queue Module:**

  The Obtain Queue function in LabVIEW is a programming element that allows we to read data from a queue in a producer-consumer architecture. The queue is a data structure that can hold and transfer data between different parts of a LabVIEW application. When a producer (such as a sensor or other data acquisition device) acquires data, it can send it to a consumer (such as a processing or analysis module) for further use. The Obtain Queue function allows the consumer to access data from the queue without interfering with the producer's operation. When the queue contains data, the Obtain Queue function reads the oldest element from the queue and removes it. If the queue is empty, the function waits until new data arrives.

- **Enqueue Element and Dequeue Element Module:**

  Enqueue Element and Dequeue Element are two essential modules in LabVIEW that are used for implementing FIFO (First-In-First-Out) data structures.

  Enqueue Element adds an element to the end of the queue. It takes the queue reference as an input and appends the element to the end of the queue.

  Dequeue Element, on the other hand, removes and returns the first element from the queue. It also takes the queue reference as an input and returns the element to the output terminal.

  Together, these two modules can be used to implement a queue data structure in LabVIEW, enabling the processing of data in the order it was received. This is particularly useful when dealing with real-world systems, where data needs to be processed in the order it was generated.

  It's worth noting that LabVIEW also provides other array manipulation tools that can be used for various data manipulation processes.

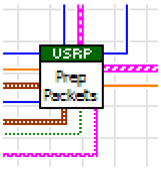
Lab results & Analysis:

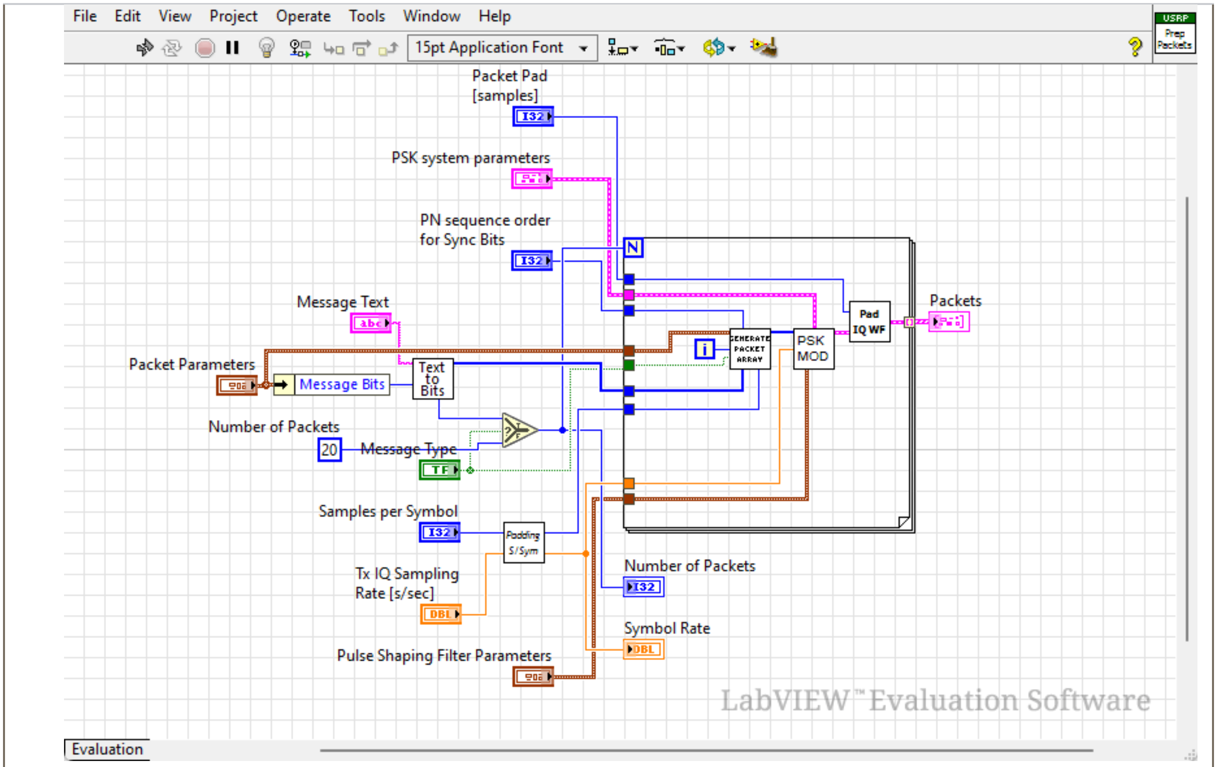
1. **Complex baseband waveform of the packet**

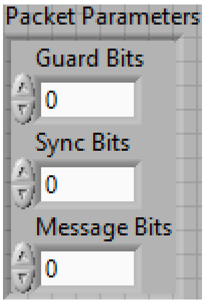
   The complex baseband waveform of a packet is the modulated signal that is transmitted over a wireless communication channel. In general, a packet consists of a header and payload. The header carries information about the packet, such as the sender's and receiver's addresses, the length of the packet, and error detection and correction bits. The payload contains the actual data being transmitted.

   The modulation process converts the digital packet into an analog signal, which can then be transmitted over the wireless communication channel. At the receiver, the complex baseband waveform is demodulated to recover the original packet. The header is used to synchronize the receiver's demodulator with the carrier wave and properly decode the modulated signal. By decoding the header, the receiver can extract the address information and payload length, which allows it to properly process and use the information contained in the payload.

   In the program, the job of packing is done bt the subVI called subGenertaePackert.vi in the subVI library supplied by Dr. Wu.







Packet Parameters

Guard Bits
0

Sync Bits
0

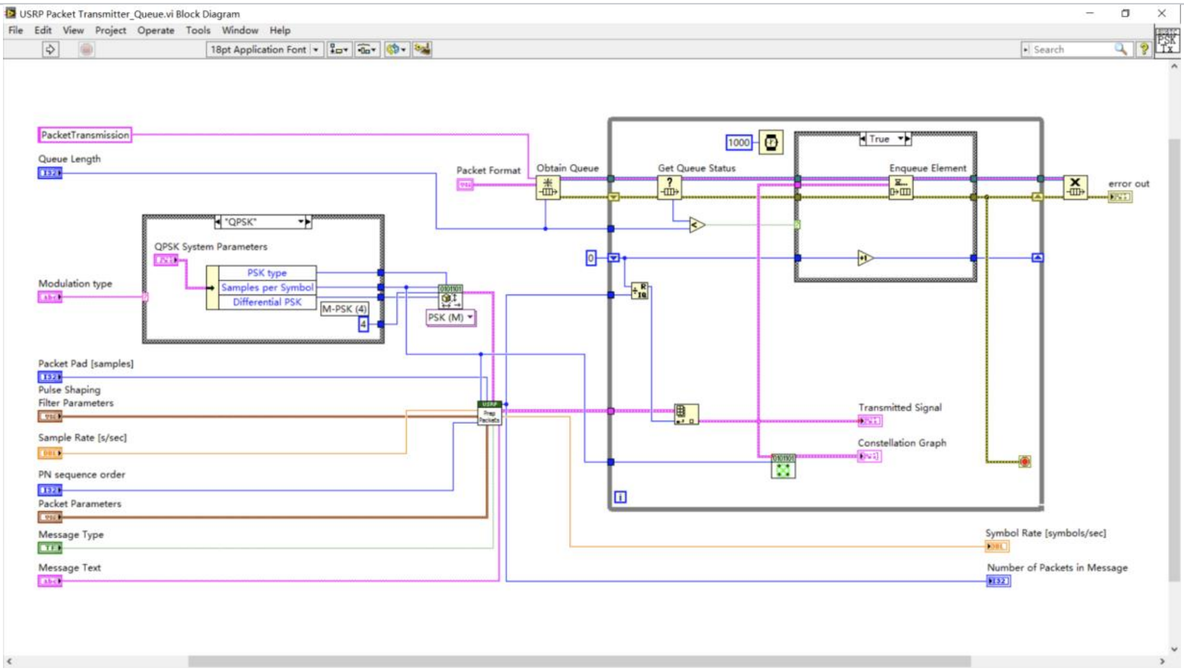
Message Bits
0

The difination of the packet contains three parts, guard bits, sync bits and message bits. The sync bits and message bits is too causal to talk about. The guard bits are bits added to a message to detect errors that may occur during transmission, which might contain some cryptography and some correction scheme like hamming code. I hope to learn more about these in the coming course.

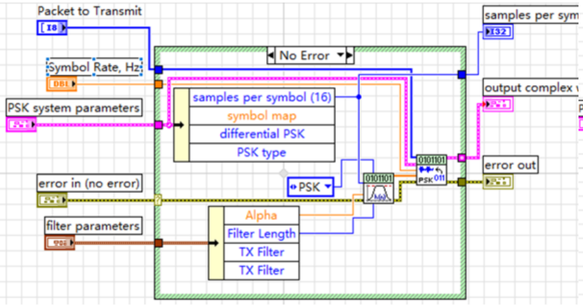
2. Queue-based text transfer simulation

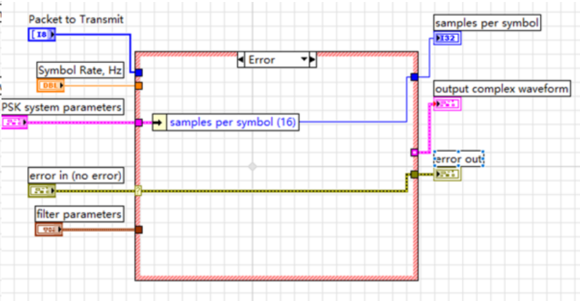
In LabVIEW, queue-based text transfer simulation is a way of simulating a digital communication system where data is transferred between two or more devices using queues.

In a queue-based text transfer simulation, a producer loop generates text data and sends it to a queue, while a consumer loop retrieves the text data from the queue and processes it. The producer and consumer loops are synchronized using the queue, so that data is only transferred when there is data available to transfer. This helps avoid synchronization issues that can arise when sharing data between parallel loops.

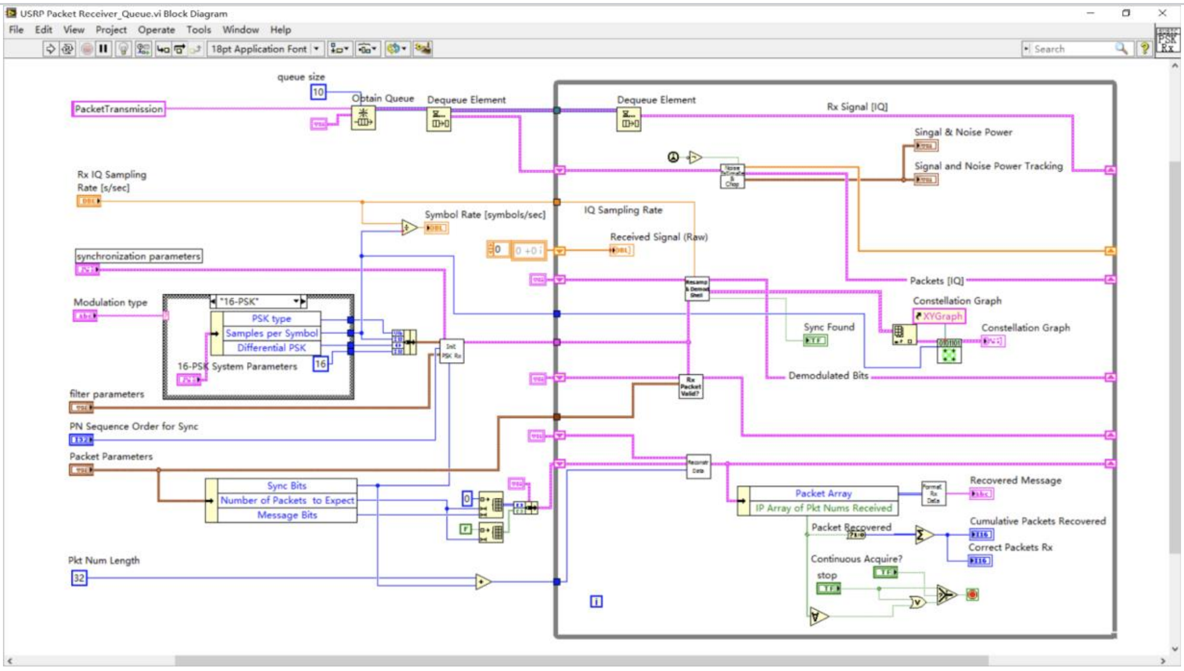


The core of the queue simulation is the programing of two file, "sub_resample_and_demodulate" and "sub_PSKMod" which decide the psk type. Thanks to the nice naming of the input and output parameters, we reverse enginering the file with the help of lab7 experience, which is also psk modulation.

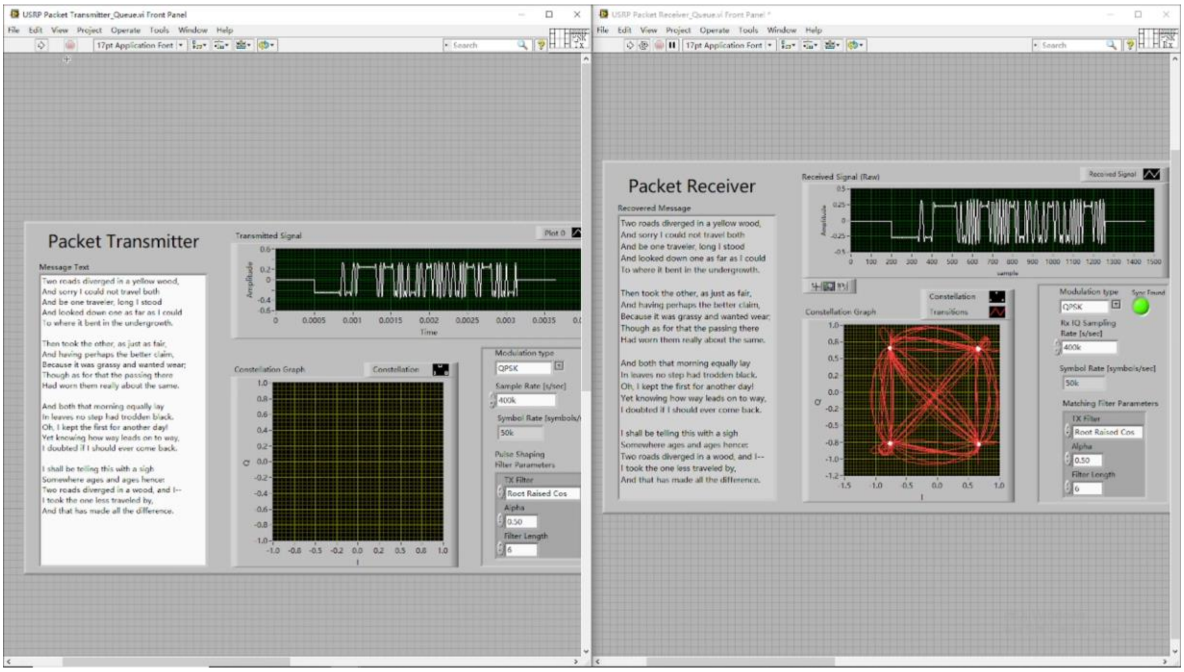




However, we spend lots of time on demodulation. The data can be access from queue but we can't revover it into the readable data we want, i think something about packeting was misunderstood. So we did not manage to finish the demodulation part, that is "sub_resample_and_demodulate". To complete the rest of project, we use the encrypted version to repalce the subVI.



The text could be normaly transfered through queue and recoverde from modulation.



Packet Transmitter

Message Text

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth.

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same.

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I--
I took the one less traveled by,
And that has made all the difference.

Modulation type: QPSK
Sample Rate [s/sec]: 400k
Symbol Rate [symbols/sec]: 50k
Pulse Shaping Filter Parameters
TX Filter: Root Raised Cos
Alpha: 0.50
Filter Length: 6

Packet Receiver

Recovered Message

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth.

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same.

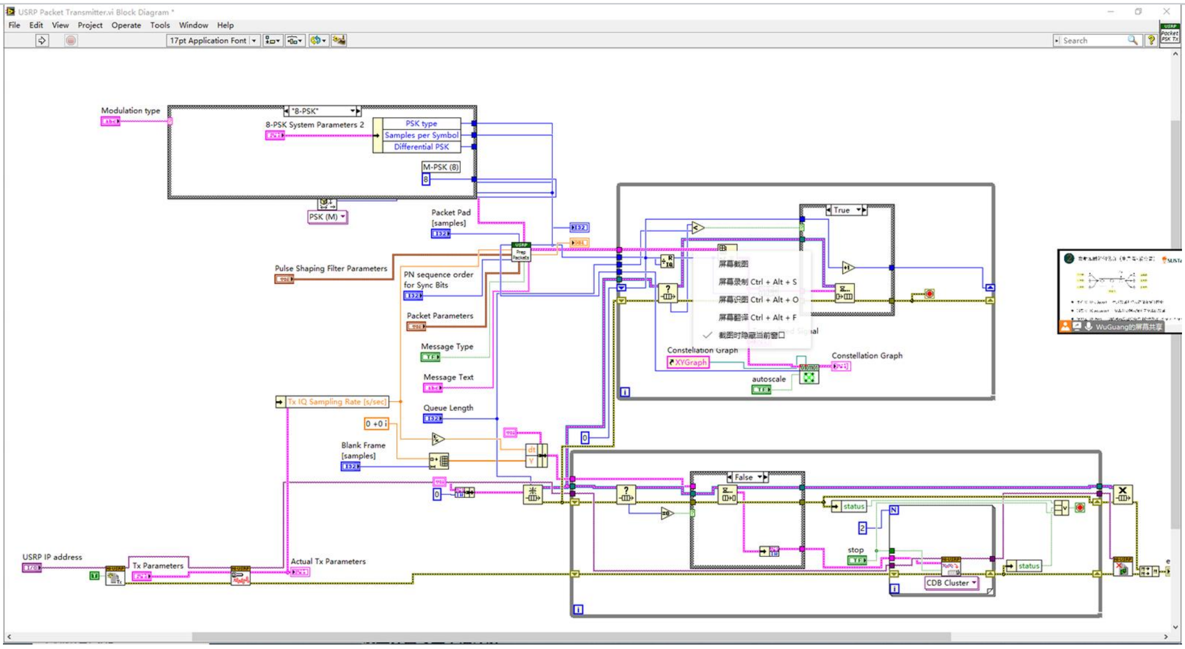
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

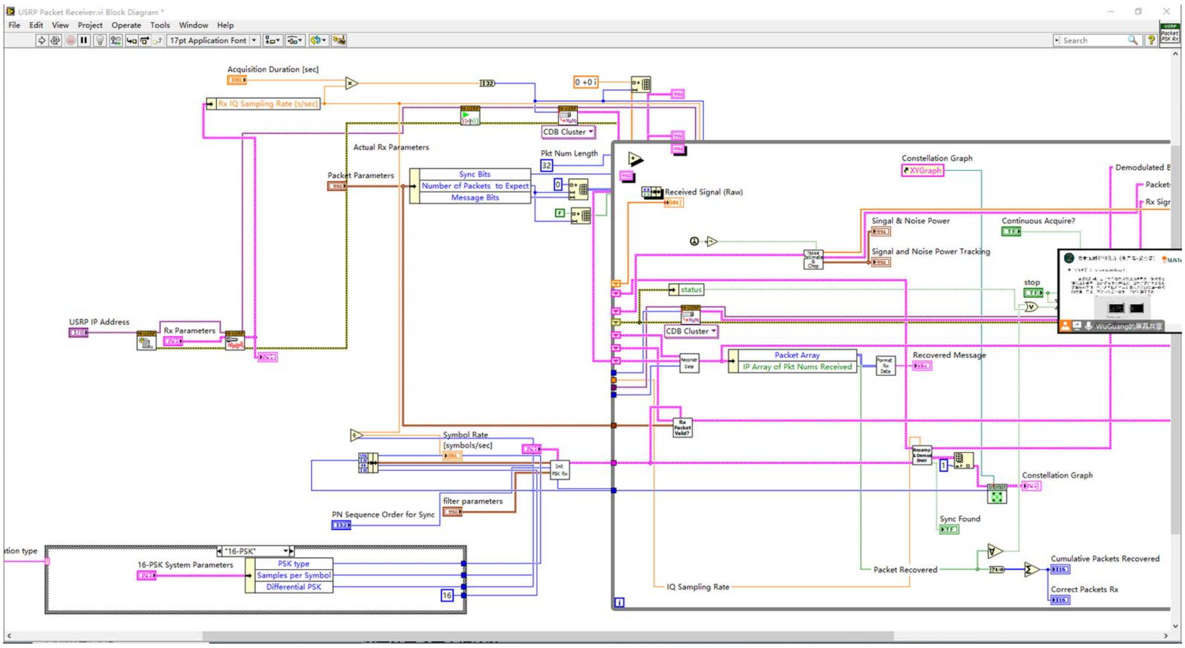
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I--
I took the one less traveled by,
And that has made all the difference.

Modulation type: QPSK
Rx IQ Sampling Rate [s/sec]: 400k
Symbol Rate [symbols/sec]: 50k
Matching Filter Parameters
TX Filter: Root Raised Cos
Alpha: 0.50
Filter Length: 6

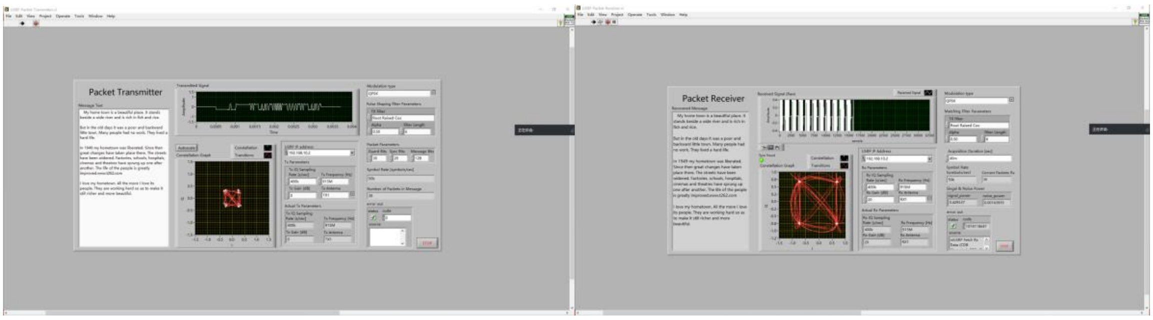
3. USRP-based packet transmission

Following the program posted on blackboard, we finish the driver program of usrp in both of the transceiver.





The result is as below.



4. Advantages and Disadvantages of Digital carrier transmission system and Its Value

Advantages:

- **Robustness:** PSK modulation is more immune to noise and distortion than other types of digital modulation techniques, making it a reliable option in noisy environments.
- **Efficient use of bandwidth:** PSK modulation requires less bandwidth than analog modulation techniques since it can transmit multiple bits per symbol. This makes it more efficient in terms of spectrum utilization.
- **Flexibility:** PSK modulation can be used with various communication systems, making it versatile for different applications.

Disadvantages:

- **Increased complexity:** PSK modulation can be more complex than other modulation techniques, especially as the number of bits per symbol increases.
- **Need for synchronization:** PSK modulation requires precise synchronization between the transmitter and receiver, which can be challenging in some applications.
- **Limited distance:** Since PSK modulation is sensitive to phase errors, it has a limited range of transmission, making it less suitable for long-distance communication.

Value:

Digital carrier wave systems hold significant social value due to their impact on communication, connectivity, and information exchange. For example:

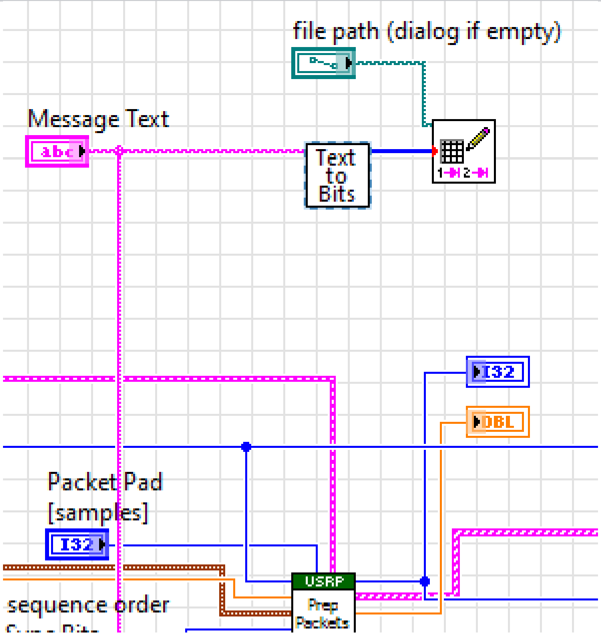
- **Military Applications:** In military communication, PSK modulation system plays a vital role in transmitting information securely and efficiently. It is also capable of supporting the transmission of encrypted data, preventing enemies from intercepting data transfers.
- **Medical Applications:** PSK modulation is used in the design of medical telemetry systems, primarily in implantable medical devices. It is used to transmit a large number of data concurrently with high precision and minimum errors.

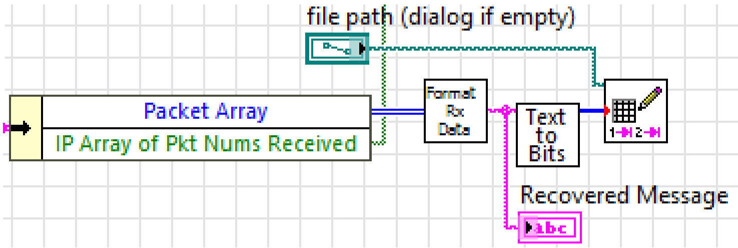
5. The realationship between alpha, bandwidth with BER

Alpha is a parameter in Pulse Shaping Filter Parameters, witch indicat the roll-off factor of raised-cos function.

In packet communication, the bandwidth is a function of the data rate and the packet duration. We can calculate packet duration by the number of bits in the message divided by the bit rate, plus the duration of other packet overheads such as preamble, sync bits, guard bits, and header. The bandwidth can then be calculated as the inverse of the packet duration.

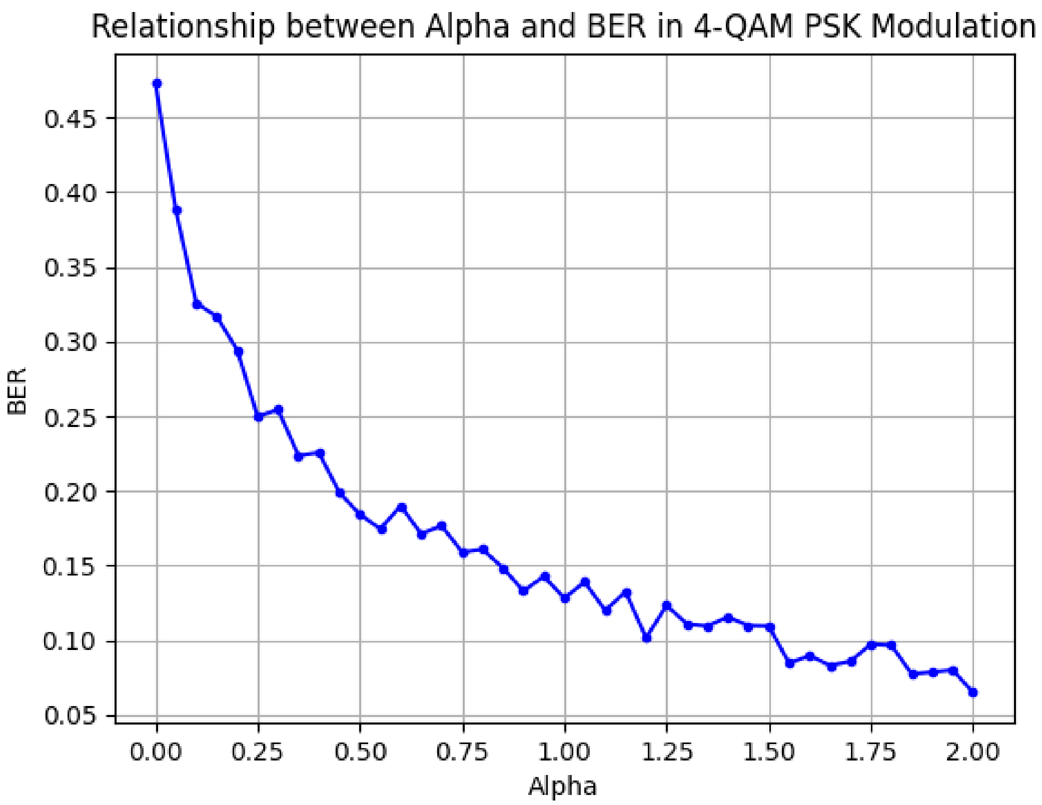
As for BER, we can do the same in lab7, collecting the transfered data and origin data, calculate BER with a python script. Below is how we do this:



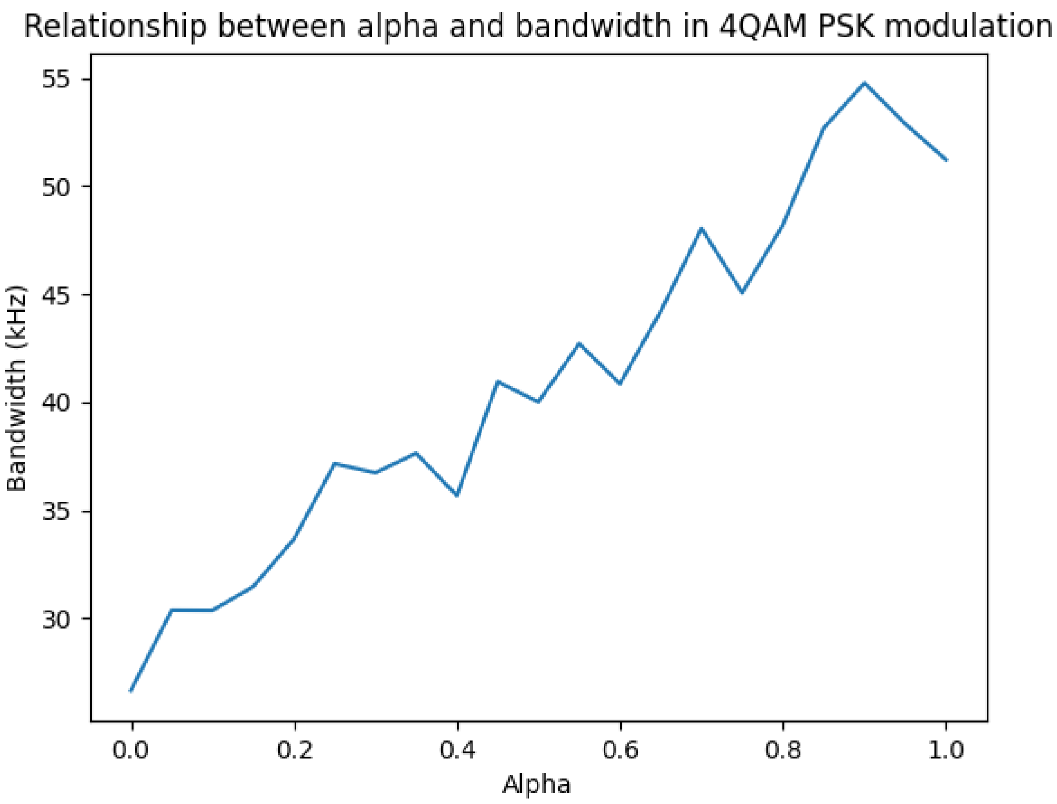


We reuse the "sub_Text_To_Bitstream" in subVI library, then save it into a csv sheet.

Using the python script below calculate the ber, then plot it out.

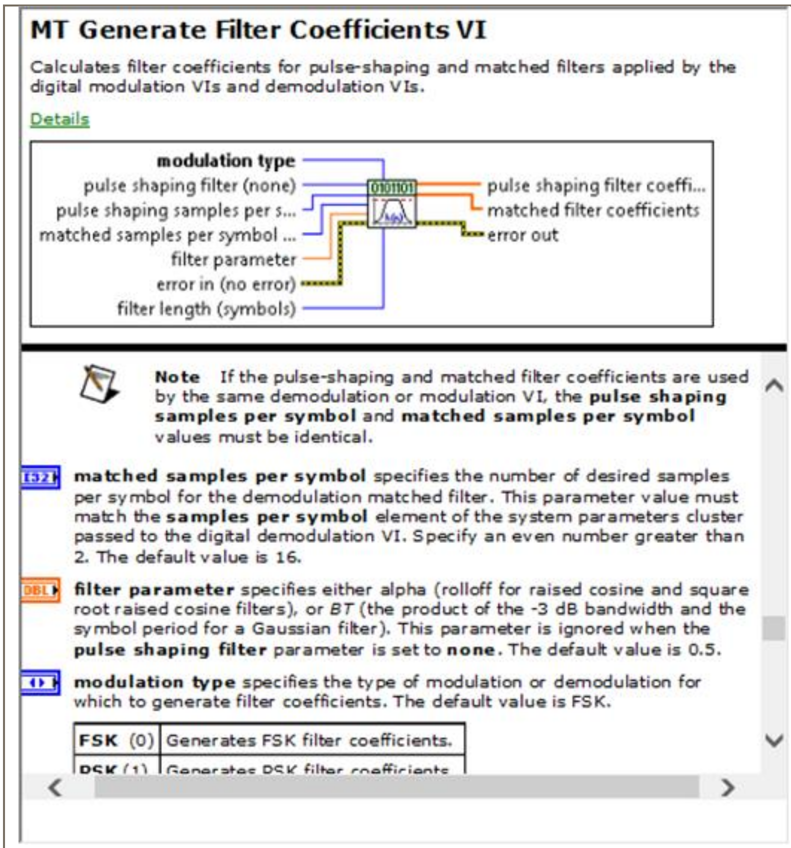


As for bandwith, it is basic linear to alpha, so the relationship between bandwidth with BER is similar to alpha.



Experience:

In the programming process, docmentation helps a lot. It provides an in-depth understanding of how each function works and what its inputs and outputs mean. And it assists in troubleshooting and debugging errors in the program. For example, we don't know what does "alpha" mean in the lowest pskMOD.vi, but the help doc tells that this is a filter parameter, witch help us think of the alpha we learned in class.



MT Generate Filter Coefficients VI

Calculates filter coefficients for pulse-shaping and matched filters applied by the digital modulation VIs and demodulation VIs.

Details

modulation type
pulse shaping filter (none)
pulse shaping samples per s...
matched samples per symbol ...
filter parameter
error in (no error)
filter length (symbols)
0101101
pulse shaping filter coeffi...
matched filter coefficients
error out

Note If the pulse-shaping and matched filter coefficients are used by the same demodulation or modulation VI, the **pulse shaping samples per symbol** and **matched samples per symbol** values must be identical.

I32 **matched samples per symbol** specifies the number of desired samples per symbol for the demodulation matched filter. This parameter value must match the **samples per symbol** element of the system parameters cluster passed to the digital demodulation VI. Specify an even number greater than 2. The default value is 16.

DBL **filter parameter** specifies either alpha (rolloff for raised cosine and square root raised cosine filters), or *BT* (the product of the -3 dB bandwidth and the symbol period for a Gaussian filter). This parameter is ignored when the **pulse shaping filter** parameter is set to **none**. The default value is 0.5.

modulation type specifies the type of modulation or demodulation for which to generate filter coefficients. The default value is FSK.

FSK (0)	Generates FSK filter coefficients.
PSK (1)	Generates PSK filter coefficients

Score:	高荣：96　　曹正阳：95

```
# BER calculate script
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import make_interp_spline


def str_to_array(string):
    array = []
    # removing the first and last empty strings from the list
    string = string.split('\t')[1:-1]
    array = np.array([int(bit) for bit in string])
    return array


def calBER(a, b):
    if len(a) != len(b):
        raise ValueError("Arrays of unequal length")
    errorbits = sum(bit1 != bit2 for bit1, bit2 in zip(a, b))
    ber = errorbits / len(a)
    return ber


# list of file names
files = []

for i in range(24):
    file_name = str(i) + '.csv'
    files.append(file_name)


# initial ber
BERS = []
```

```
# loop over files
for i, file in enumerate(files):

    # read csv file into dataframe
    df = pd.read_csv(file)

    # initialize variables
    errorbits = 0
    size = len(df.columns)

    # turn the bits string into bits array
    # stupic asshole labview forget about the head row, i gotta
manually add a head for every single file, fxxk!
    origin = str_to_array(df.iloc[0, 0])
    recover = str_to_array(df.iloc[1, 0])

    # compare origin and recover columns to calculate bit errors
    BER = calBER(origin, recover)
    BERS.append(BER)
    print(BERS)

# plot BER vs file index
# Create an x-axis array to match the length of BERS
x = np.arange(24)

# Smooth the curve
x_new = np.linspace(x.min(), x.max(), 300)
spline = make_interp_spline(x, BERS, k=3)
BERS_smooth = spline(x_new)

# Plot the graph
plt.plot(x_new, BERS_smooth)
# set plot parameters
plt.xlabel('E_b/N_0(dB)')
plt.ylabel('Bit Error Rate')
plt.title('BER vs E_b/N_0')
plt.plot(x_new, BERS_smooth, label="4-PSK")
plt.legend()
plt.show()
```