****

ENG4052: Digital Communication 4 (2022-23)

Lab1: Digital Modulation and Demodulation

**Haoshi Huang (2635088H)**

**Submission Date**

**1/25/2023**

**1 Introduction**

Modulation and demodulation are keys in digital communication field. Digital modulation is the process of processing the coded information of signal source to make it suitable for transmission. Generally, a very high frequency relative to the baseband frequency sinusoidal signal, called carrier wave, is used to carry baseband signal. Then the modulated signal will be transmitted during communication. Modulation can change in amplitude, phase or frequency with the high frequency carrier according to the source signal changes. So, the three most basic digital modulation methods are ASK, FSK and PSK.

In this coding project, I will use Python3.10 in Visual Studio to implement and analyse two modulation and demodulation techniques, which are **Binary Phase Shift Keying** (BPSK or 2PSK) and **Quadrature Phase Shift Keying** (QPSK or 4PSK). Libraries NumPy1.23, SciPy1.9 are imported to implement advanced math operation. Library matplotlib3.6 is used to display graphics, which helps us to know about how signal changes after each processing.

**2 BPSK modulation**

**2.1 Create original signal**

In this lab, the transmitted information is student ID, a 7-digit decimal number. During transmission, digital signals need to be represented in binary, so the first thing is to convert the 7-digit decimal number to 24-digit binary number. My student ID is 2635088, and its binary number is 0010 1000 0011 0101 0101 0000. Method np.zfill(m) can fill zeros to high digits in a fixed m-digit binary number. Obviously, the first two zeros in this 24-digit binary number are filled by this way.

**2.2 Modulation**

Before modulation, the sample rate of signal source is initialised to 16, which means 1-bit digital signal has 16 sampling points. So, there will be 16\*24 = 384 sample points for the original signal. Because of Nyquist limit, the normalized frequency of carrier wave is at least initialized to double sample rate, 1/8. There are two periods of carrier wave for one bit information during transmission. The coded information is shown in Fig. 2.1.

图表, 直方图

描述已自动生成

*Figure 2.1 Original signal*

A BPSK modulated signal can be expressed by following formular:



In this expression, when original signal is 1, s(t) refers to 1; when signal is 0, s(t) refers to -1. In code, the expression 2\*s[i]–1 can build the map of values from {0, 1} to {-1, 1}, corresponding to 180 degrees phase difference in the constellation diagram of binary code elements. And then a modulated signal of length 384 can be calculated with the 24 size of outer loop and the 16 size of inner loop. The modulated signal shows in the Fig. 2.2. The Fig. 2.3 is the frequency domain signal, where there is fundamental wave around 56 Hz.

图表, 条形图, 直方图

描述已自动生成

*Figure 2.2 Modulated signal*

图表, 直方图

描述已自动生成

*Figure 2.3 Modulated signal in frequency domain*

We choose the first 6 periods of modulated signal to verify, corresponding to 001, as the Fig. 2.4 shows.

图表, 直方图

描述已自动生成

*Figure 2.4 Verify modulated signal*

**2.3 Demodulation**

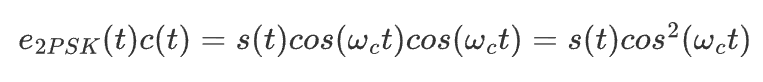
Demodulation is the inverse process of modulation. After demodulation, the original signal will be recovery from demodulated signal. The code implements coherent demodulation method, which means a reference signal, the same frequency and phase as the carrier wave, is multiplied with the modulated signal. The Fig. 2.5 shows the processing of coherent demodulation.

图示

描述已自动生成

*Figure 2.5 Processing of coherent demodulation*

The first thing is that 384 modulated signals need to do the multiplication with reference sinusoidal wave in time order as the following expression shows.



The Fig. 2.6 shows the signal after multiplication. The Fig. 2.7 shows the signal in frequency domain with FFT, where there is secondary harmonic around 112 Hz.

图表, 条形图

描述已自动生成

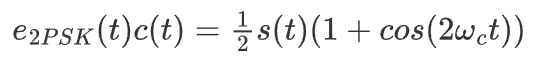
*Figure 2.6 Signal after multiplicator*

图表, 直方图

描述已自动生成

*Figure 2.7 Signal after multiplicator in frequency domain*

Mathematically, according to trigonometric identity transformation, the multiplier result can be expressed as following.



The Fig. 2.8 shows the signal after low pass filter, and the Fig. 2.9 shows the filtered signal in frequency domain, where there is only low frequency information left. The secondary harmonics has been filtered by LPF compared with Fig. 2.7.

图表, 直方图

描述已自动生成

*Figure 2.8 Filtered signal*

图表

描述已自动生成

*Figure 2.9 Filtered signal in frequency domain*

Finally, after sample judgement of filtered signal, we can get demodulated signal, which should be the same as original signal. We can also verify the demodulated signal by comparing the Fig. 2.10 with the Fig. 2.1.

图表, 直方图

描述已自动生成

*Figure 2.10 Demodulated signal*

**3 QPSK modulation**

QPSK can be shown as two phases dependent BPSK where there is 90 degrees each other.

**3.1 Modulation**

Using the same original signal in BPSK, like the processing of BPSK modulation, QPSK modulation involves in-phase (I component) and quadrature (Q component). With the same formular of BPSK, I component need to multiply with cos wave, and Q component need to multiply by sin wave. The two wave have the same frequency and phase. The modulated signal is shown as following Fig. 3.1. And the Fig. 3.2 shows the modulated signal in frequency domain.

图表, 直方图

描述已自动生成

*Figure 3.1 Modulated signal*

图表, 直方图

描述已自动生成

*Figure 3.2 Modulated signal in frequency domain*

**3.2 Demodulation**

When QPSK demodulation, IQ components also pass through separately in time order like processing of BPSK’s demodulation. The two filtered signal are shown in the Fig. 3.3.

图表, 折线图, 直方图

描述已自动生成

*Figure 3.3 Filtered signal*

Finally, we use sampling judgement methods to get the original signal as following Fig. 3.4, which is the same as Fig. 2.1.

图表, 直方图

描述已自动生成

*Figure 3.4 Filtered signal*

**4 Conclusion**

**4.1 Filter delay**

Because data stream passes through LPS filter, the result of LPS filter will generate the half of the number of taps. We need to append the same number of zeros after filtered data stream at the end separately.

**4.2 Zeros fill after valid original signal**

Although the transmitted signal has 24 bits, the edge effect will be generated when the low-pass filter does convolution during demodulation. Therefore, zero filling operation should be carried out at the end of the transmitted signal stream of BPSK and QPSK before modulation. Its noted that QPSK is a two-bit as one coded element, so the number of zero filling should be even. In the code, BPSK are filled with two zeros, and QPSK are filled with four zeros. The code are shown as following.

图片包含 图示

描述已自动生成

**Appendix:**

**BPSK.py**

import numpy as np

from matplotlib import pyplot as plt

from scipy import fft

from scipy import signal

def bin\_array(num, m):

# Convert a positive integer num into an m-bit bit vector

return np.array(list(np.binary\_repr(num).zfill(m))).astype(np.bool)

# Decimal to binary

id\_num = 2635088

Nbits = 24

tx\_bin = bin\_array(id\_num, Nbits)

print(tx\_bin)

plt.figure()

plt.title('Original signal')

plt.plot(tx\_bin)

plt.show()

tx\_bin = np.append(tx\_bin, [0, 0])

Nbits += 2

# BPSK modulation

# initialisation

bit\_len = 16 # 16 samples per bit, so sample rate = 1/16

fc = 0.125 # normalised carrier frequency because of Nyquist limit fc >= 2fs, so there will be 2 periods of carrier wave per bit

s = np.copy(tx\_bin)

s\_mod = np.empty(0)

t = 0

# based on IQ modulation

for i in range(Nbits):

for j in range(bit\_len):

# BPSK s(t):{0, 1} => {-1, 1}

s\_mod = np.append(s\_mod, (2\*s[i] - 1) \* np.cos(2\*np.pi\*fc\*t))

t += 1

plt.figure()

plt.title('Modulated signal')

plt.plot(s\_mod)

plt.show()

plt.figure()

plt.title('Modulated signal in frequency domain')

plt.plot(np.abs(fft.fft(s\_mod)))

plt.show()

# Demodulation(coherent detection method)

s\_demod\_multi = np.empty(0)

t = 0

for i in range(Nbits):

for j in range(bit\_len):

s\_demod\_multi = np.append(s\_demod\_multi, s\_mod[t] \* np.cos(2\*np.pi\*fc\*t))

t += 1

plt.figure()

plt.title('Signal after multiplicator')

plt.plot(s\_demod\_multi)

plt.show()

plt.figure()

plt.title('Signal after multiplicator in frequency domain')

plt.plot(np.abs(fft.fft(s\_demod\_multi)))

plt.show()

# Do filter, try to change coeffients of fir filter

numtaps = 64 # 32

cutoff = 0.1

fir = signal.firwin(numtaps, cutoff)

s\_demod\_lpf = signal.lfilter(fir, 1, s\_demod\_multi)

s\_demod\_lpf = np.append(s\_demod\_lpf, -np.ones(numtaps//2))

plt.figure()

plt.title('Filtered signal')

plt.plot(s\_demod\_lpf)

plt.show()

plt.figure()

plt.title('Filtered signal in frequency domain')

plt.plot(np.abs(fft.fft(s\_demod\_lpf)))

plt.show()

# Sample judgement

s\_demod\_bin = np.empty(0)

for i in range(Nbits):

t = (2\*i+1)\*bit\_len//2 + numtaps // 2 # use median sample to judge original signal

s\_demod\_bin = np.append(s\_demod\_bin, s\_demod\_lpf[t] > 0.0)

print(s\_demod\_bin[:24])

plt.figure()

plt.title('Demodulated signal')

plt.plot(s\_demod\_bin[:24])

plt.show()

**QPSK.py**

import numpy as np

from matplotlib import pyplot as plt

from scipy import fft

from scipy import signal

def bin\_array(num, m):

# Convert a positive integer num into an m-bit bit vector

return np.array(list(np.binary\_repr(num).zfill(m))).astype(np.bool)

id\_num = 2635088

Nbits = 24

tx\_bin = bin\_array(id\_num, Nbits)

print(tx\_bin)

tx\_bin = np.append(tx\_bin, [0, 0, 0, 0])

Nbits += 4

plt.figure()

plt.title('Original signal')

plt.plot(tx\_bin[:24])

plt.show()

# QPSK modulation

# initialise constants and variables

fc = 0.125

bit\_len = 16

s = np.copy(tx\_bin)

s\_mod = np.empty(0)

t = 0

# based on IQ component, constellation diagram {45, 135, 225, 315}

# 4PSK, 1 Baud presents 2 bits, two bits into one group

for i in range(0, Nbits, 2):

for j in range(bit\_len):

s\_mod = np.append(s\_mod, (2\*s[i] - 1) \* np.cos(2\*np.pi\*fc\*t) + (2\*s[i+1] - 1) \* np.sin(2\*np.pi\*fc\*t))

t += 1

# Show modulated signal

plt.figure()

plt.title('Modulated signal')

plt.plot(s\_mod)

plt.show()

# Use fft to frequency analyse

plt.figure()

plt.title('Modulated signal in frequency domain')

plt.plot(np.abs(fft.fft(s\_mod)))

plt.show()

# Demodulation, using coherent detection,

# First step: IQ components respectively multiply the carrier wave and qudrature carrier wave

s\_demod\_i = np.empty(0)

s\_demod\_q = np.empty(0)

t = 0

for i in range(0, Nbits, 2):

for j in range(bit\_len):

s\_demod\_i = np.append(s\_demod\_i, s\_mod[t]\*np.cos(2\*np.pi\*fc\*t))

s\_demod\_q = np.append(s\_demod\_q, s\_mod[t]\*np.sin(2\*np.pi\*fc\*t))

t += 1

# Second step: use low-pass filter to filter harmonic wave

# initilise filter coefficients

numtaps = 64

fir = signal.firwin(numtaps, 0.1)

# IQ component do filter

s\_filt\_i = signal.lfilter(fir, 1, s\_demod\_i)

s\_filt\_i = np.append(s\_filt\_i, -np.ones(numtaps//2)/2)

s\_filt\_q = signal.lfilter(fir, 1, s\_demod\_q)

s\_filt\_q = np.append(s\_filt\_q, -np.ones(numtaps//2)/2)

plt.figure()

plt.title('Filtered signal')

plt.plot(s\_filt\_i, color = 'b')

plt.plot(s\_filt\_q, color = 'r')

plt.show()

# Sample judgement

# in fact, QPSK can be considered as two qudrature BPSK

s\_demod\_bin = np.empty(0)

for i in range(0, Nbits, 2):

t = (i+1)\*bit\_len//2 + numtaps//2

s\_demod\_bin = np.append(s\_demod\_bin, s\_filt\_i[t] > 0.0)

s\_demod\_bin = np.append(s\_demod\_bin, s\_filt\_q[t] > 0.0)

s\_demod\_bin = s\_demod\_bin[:-1]

print(s\_demod\_bin)

plt.figure()

plt.title('Demodulated signal')

plt.plot(s\_demod\_bin[:24])

plt.show()