****

ENG4052: Digital Communication 4 (2022-23)

Digital Modulation and Demodulation

**Ran Shuai (2633609R)**

Content Tables

[BPSK 3](#_Toc125580506)

[The modulation and demodulation 3](#_Toc125580507)

[Basic Parameters 3](#_Toc125580508)

[Modulation 3](#_Toc125580509)

[What I have Done 3](#_Toc125580510)

[The results 4](#_Toc125580511)

[Demodulation 6](#_Toc125580512)

[What I have Done 6](#_Toc125580513)

[The results 8](#_Toc125580514)

[QPSK 9](#_Toc125580515)

[Modulation 9](#_Toc125580516)

[What I have Done 9](#_Toc125580517)

[The results 9](#_Toc125580518)

[Demodulation 11](#_Toc125580519)

[What I have Done 11](#_Toc125580520)

[The results 12](#_Toc125580521)

[Conclusion 13](#_Toc125580522)

[The problem I met 13](#_Toc125580523)

[Which one is more efficient? 13](#_Toc125580524)

[Advantages and disadvantages of QPSK and BPSK 13](#_Toc125580525)

[Appendix 14](#_Toc125580526)

[BPSK 14](#_Toc125580527)

[QPSK 17](#_Toc125580528)

# BPSK

## The modulation and demodulation

图示

描述已自动生成

Figure 1.1(b): The modulation and demodulation Process

## Basic Parameters

The parameter bit\_len means that we take 16 sample points for one bit.

The parameter fc is the normalized carrier frequency.

When fc equals to 0.125 and bit\_len = 8, it means two whole carrier wave periods only represent one bit.

## Modulation

BPSK (Binary Phase Shift Keying) is a method of digital signal encoding that uses two different phases to represent two different information symbols. Typically, one phase is used to represent "0" and the other phase is used to represent "1".

### What I have Done

To modulate the signal, the signal needs to multiply by a carrier wave. This is because the information we want to transfer is always low frequency but the low frequency is not easy to be transferred. The carrier frequency is used here to transfer the signal. One of the main reasons is that by carrying the signal on a carrier wave, the signal is more resistant to interference and noise.

By this formula in figure 1.1, we get the modulated signal.

图片包含 徽标

描述已自动生成

Figure 1.1(b): The modulation Formula

The Operation in the code:

(2\*s[i]-1)\*np.cos(2\*np.pi\*fc\*(i\*bit\_len+j))

### The results

图表, 直方图

描述已自动生成

Figure 1.2: The Original Signal

图表, 条形图, 直方图

描述已自动生成

Figure 1.2: The modulated Signal

图表, 直方图

描述已自动生成

Figure 1.3: The modulated signal which is zoomed in

The first two elements of the 24-bit binary array of my student number are false. From Figure 1.3, it is implied that A cosine function with a phase of 180 represents 0, while a cosine function with a phase of 0 represents 1.

图表, 直方图

描述已自动生成

Figure 1.4: The spectrum of Modulated Signal

From figure 1.4, it is implied that carrier frequency has higher power, but sidebands have lower power. It is not effective, because the carrier only carries small information. To be more efficient, the sidebands should have more power.

## Demodulation

This process is to convert the modulated signal to the original signal.

### What I have Done

The achieve the original signal, the modulated signal should multiply the carrier wave again.

The formula to translate this operation

图片包含 文本

描述已自动生成

Figure 1.5: The demodulation Formula

This formula shows that after multiplying the modulated signal by the carrier signal again. We get half of the original signal and a signal which has twice the frequency of the carrier signal.

图表, 直方图

描述已自动生成

Figure 1.6: Spectrum of Modulated signal Multiplied by Carrier signal again

From the figure, we can see a signal with a frequency twice that of the carrier frequency. To get the original signal, we should filter it out. So a lowpass filter is used here to achieve this aim.

图表, 直方图

描述已自动生成

Figure 1.7: Signal which passed LPF

This is the signal which has already passed the LPF. But it looks like an analogue signal, and not like the original signal. We can use a threshold function to convert it back to the original signal.

### The results

图表, 直方图

描述已自动生成

Figure 1.8: The demodulated Signal

This is the demodulated signal I finally get. There is hardly any delay in the demodulated signal compared to the original signal because I add numtaps//2 dummy samples to the end of my bit array.

# QPSK

## Modulation

QPSK encoding method uses changes in phase to transmit information, with each code symbol consisting of two binary information bits, resulting in four possible states.

### What I have Done

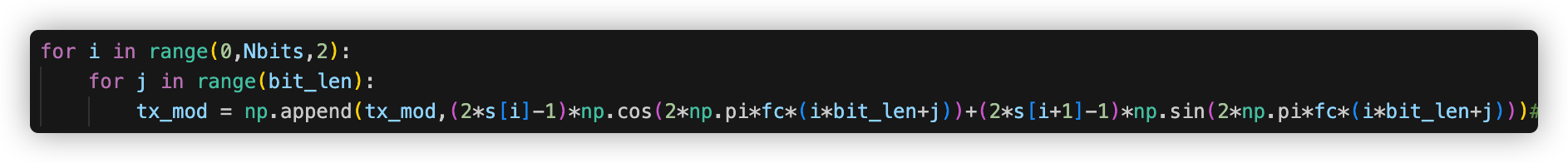


Figure 2.1: The modulation process

The cosine and sine functions are used here to implement phase shifting, where fc is the carrier frequency and i\*bit\_len+j is the time point. Through this method, the phase of the signal changes between 0 degrees, 90 degrees, 180 degrees and 270 degrees according to the values of s[i] and s[i+1], achieving QPSK encoding.

### The results

图表, 直方图

描述已自动生成

Figure 2.2: The original signal

图表, 条形图, 直方图

描述已自动生成

Figure 2.3: The modulated signal

图表, 直方图

描述已自动生成

Figure 2.4: The spectrum of modulated Signal

## Demodulation

Just like BPSK, but in QPSK we need to demodulate the cos carrier wave and sin carrier wave separately.

### What I have Done

文本

描述已自动生成

Figure 2.5: The QPSK Demodulation Processing

In this piece of code, rx\_demod\_cos and rx\_demod\_sin are variables used to store the demodulated signal. In the loop, demodulation is achieved by multiplying the modulated signal with the cosine and sine functions.

And then the two signals pass the LPF, so we get the original signal. But we still need to use a threshold function to deal with the original signal.

### The results

图表, 直方图

描述已自动生成

Figure 2.6: Modulated signal \* coswct passed LPF

The threshold function will be applied here to make this signal more like a digital signal.

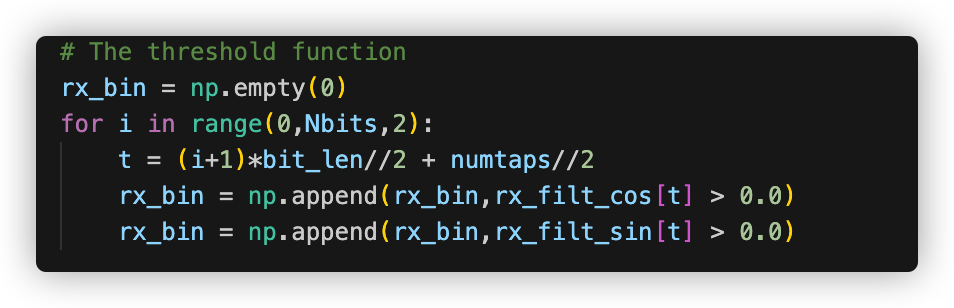


Figure 2.7: The threshold function

图表, 直方图

描述已自动生成

Figure 2.8: The demodulated signal

# Conclusion

## The problem I met

When my binary sequence is 24-bit binary, there is always a problem when demodulating the last bit. So I added some zeros to the original sequence at the end.

图表

中度可信度描述已自动生成

Figure 3.1: The problem

## Which one is more efficient?

From figures 1.4 and 2.4, it is implied The sideband power of the signal after BPSK encoding is lower than the QPSK encoding. QPSK is more efficient because the sidebands carry more information than the carrier frequency.

## Advantages and disadvantages of QPSK and BPSK

BPSK (Binary Phase Shift Keying):

Advantages: Simple, easy to implement, high bandwidth efficiency, only two possible symbol states, low error rate

Disadvantages: Low symbol rate, high signal-to-noise ratio requirements

QPSK (Quadrature Phase Shift Keying):

Advantages: High symbol rate, low signal-to-noise ratio requirements, more tolerant to multipath fading and interference

Disadvantages: Complex, difficult to implement, lower bandwidth efficiency

# Appendix

## BPSK

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Mon Jan 23 18:32:03 2023

@author: ranshuai

"""

# BPSK

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)

# import 24 bit digital data

id\_num = 2633609

Nbits = 24

tx\_bin = bin\_array(id\_num, Nbits) # the 24 bit binarry array

# When transmitting the last signal, errors always occur,

# so I add several zero values to the original signal, making the last signal the third last signal.

# tx\_bin = np.append(tx\_bin,0)

# tx\_bin = np.append(tx\_bin,0)

# Nbits = Nbits + 2

##############################################

bit\_len = 16 #

fc = 0.125 # normalized frequency

s = np.copy(tx\_bin) # s = original signal

print(tx\_bin)

tx\_mod = np.empty(0) # The modulated signal

plt.figure()

plt.plot(tx\_bin)

plt.title("Original signal")

plt.xlim(0,24)

plt.show()

############### modulation###############

for i in range(0,Nbits):

for j in range(bit\_len):

tx\_mod = np.append(tx\_mod,(2\*s[i]-1)\*np.cos(2\*np.pi\*fc\*(i\*bit\_len+j)))# signal \* coswct carrier frequency

plt.figure()

plt.plot(tx\_mod)

plt.title("The modulated Signal")

plt.show()

plt.figure()

plt.plot(np.abs(fft.fft(tx\_mod))) # converting a complex number to its magnitude.

plt.title("The Spectrum of modulated Signal")

plt.show()

##########demodulation###############

numtaps = 64

delays = np.arange(numtaps) #

b1 = signal.firwin(numtaps, 0.1) # 0.1 means the cut off frequency which is normalised

rx\_demod = np.empty(0)

for i in range(Nbits):

for j in range(bit\_len):

rx\_demod = np.append(rx\_demod,tx\_mod[i\*bit\_len+j]\*np.cos(2\*np.pi\*fc\*(i\*bit\_len+j)))#

# the process of modulating process :s(t)\*cos"\*cos

plt.figure()

plt.plot(np.abs(fft.fft(rx\_demod))) #因为是傅立叶变换是 complex quantity 复数 方便在spectrum analyser上查看

plt.title("Spectrum of Modulated Signal \* cos(wct)")

plt.show()

rx\_filt = signal.lfilter(b1,1,rx\_demod)

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

plt.figure()

plt.plot(rx\_filt)

plt.title("Signal which passed Lpf")

plt.show()

demodulated\_signal = np.empty(0) # modulated signal

for i in range(Nbits):

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

demodulated\_signal = np.append(demodulated\_signal,rx\_filt[t] > 0.0) # threshold function

plt.figure()

plt.plot(demodulated\_signal)

plt.title("The demodulated Signal")

plt.xlim(0,24)

plt.show()

## QPSK

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Mon Jan 23 18:32:03 2023

@author: ranshuai

"""

# QPSK

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)

# import 24 bit digital data

id\_num = 2633609

Nbits = 24

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

bit\_len = 16 #

fc = 0.125 # normalized frequency

# When transmitting the last signal, errors always occur,

# so I add several zero values to the original signal, making the last signal the fifth last signal.

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

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

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

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

Nbits = Nbits + 4

##############################################

s = np.copy(tx\_bin) # s = original signal

tx\_mod = np.empty(0) # the modulated signal

plt.figure()

plt.plot(tx\_bin)

plt.title("Original signal")

plt.xlim(0,24)

plt.show()

############### modulation###############

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

for j in range(bit\_len):

tx\_mod = np.append(tx\_mod,(2\*s[i]-1)\*np.cos(2\*np.pi\*fc\*(i\*bit\_len+j))+(2\*s[i+1]-1)\*np.sin(2\*np.pi\*fc\*(i\*bit\_len+j)))# 把信号追加到tx\_mod中

plt.figure()

plt.plot(tx\_mod)

plt.title("The modulated Signal")

plt.show()

plt.figure()

plt.plot(np.abs(fft.fft(tx\_mod))) #converting a complex number to its magnitude.

plt.title("The Spectrum of modulated Signal")

plt.show()

##########Low Pass Filter###############

numtaps = 64

delays = np.arange(numtaps) # taos

b1 = signal.firwin(numtaps, 0.1) # coefficients of the LPF

#########demodulation#################

rx\_demod\_cos = np.empty(0)

rx\_demod\_sin = np.empty(0)

t = 0

## The demodulated processing

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

for j in range(bit\_len):

rx\_demod\_cos = np.append(rx\_demod\_cos,tx\_mod[t]\*np.cos(2\*np.pi\*fc\*t))

rx\_demod\_sin = np.append(rx\_demod\_sin,tx\_mod[t]\*np.sin(2\*np.pi\*fc\*t))

t += 1

# Low pass filter

rx\_filt\_cos = signal.lfilter(b1,1,rx\_demod\_cos)

rx\_filt\_cos = np.append(rx\_filt\_cos,-np.ones(numtaps//2)/2)

rx\_filt\_sin = signal.lfilter(b1,1,rx\_demod\_sin)

rx\_filt\_sin = np.append(rx\_filt\_sin,-np.ones(numtaps//2)/2)

plt.figure()

plt.plot(rx\_filt\_cos,color = "b")

plt.plot(rx\_filt\_sin,color = "r")

plt.show()

# The threshold function

rx\_bin = np.empty(0)

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

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

rx\_bin = np.append(rx\_bin,rx\_filt\_cos[t] > 0.0)

rx\_bin = np.append(rx\_bin,rx\_filt\_sin[t] > 0.0)

rx\_bin = rx\_bin[:-1]

plt.figure()

plt.plot(rx\_bin)

plt.xlim(0,24)

plt.show()