

Федеральное агентство связи

СибГУТИ

**Кафедра телекоммуникационных сетей и
вычислительных средств (ТС и ВС)**

Дисциплина

Сети ЭВМ и телекоммуникации 2.0

Лабораторная работа №4

Модуляция. Передача данных. Контроль ошибок.

Выполнил: студент группы ИА-832

Тиванов.Д.Е

Проверил: преподаватель

Ахпашев Р.В

Новосибирск 2021

Задание

- Взять готовый файл (аудио, изображение, видео и т.д.), в программе получить из него битовую последовательность (на языке Python 3.x перевод числа в биты производится с помощью функции `bin(x)`).
- Реализовать функцию модуляции (по варианту) на основе формул в спецификации 3GPP TS38.901 (пункт 5.2.1.1). Соответственно кодер и декодер.
- Разработать клиент-серверное приложение. Сервером будет являться базовая станция (BS - Base Station), клиентом будет являться абонентское устройство (UE - User Equipment). Пример простейшего клиент-серверного приложения на языке Python 3.x [<https://github.com/fzybot/simpleClientServer>]
- Реализовать передачу данных от базовой станции к абонентскому устройству модуляции для нескольких поднесущих, используя при этом преобразование Фурье, оценить скорость передачи данных по количеству переданных символов. Размер пакета (количество байт, которое будет отправлено за одну операцию) определяете самостоятельно (далее будет варьировать в зависимости от условий).

Ход работы:

реализация QAM64 модуляции с добавлением шума

```
for i in range(0, len(binary_sequence) - 1, 6):
    dx.append(1 / np.sqrt(42) * ((1 - 2 * int(binary_sequence[i])) * (
        4 - (1 - 2 * int(binary_sequence[i + 2])) * (2 - (1 - 2 * int(binary_sequence[i + 4]))))))
    dy.append(1 / np.sqrt(42) * ((1 - 2 * int(binary_sequence[i + 1])) * (4 - (1 - 2 * int(binary_sequence[i + 3])))) * (
        2 - (1 - 2 * int(binary_sequence[i + 5])))))
    if dx[j] < 0:
        if dy[j] < 0:
            phase.append(5 * np.pi / 4)
        else:
            phase.append(3 * np.pi / 4)
    else:
        if dy[j] < 0:
            phase.append(7 * np.pi / 4)
        else:
            phase.append(np.pi / 4)
```

```
j += 1

for i in range(0, len(phase)):
    signals.append(np.sin(2 * np.pi * time + phase[i]))
```

Вывод

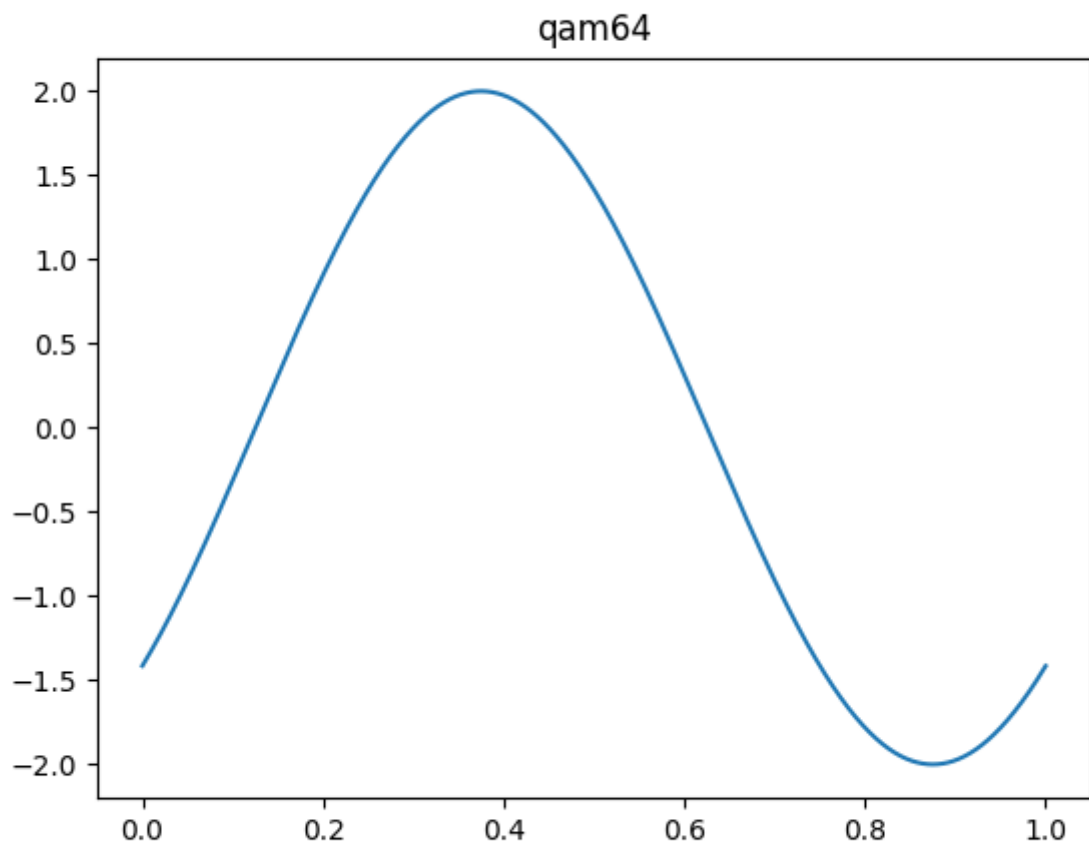
```
for i in range(len(signals)):
    signalSum = signalSum + signals[i]

signalFFT = np.fft.fft(signalSum)
signalFFTabs = 2 * np.abs(signalFFT) / fs

# График: сигнал во времени
plt.subplot(1, 1, 1)
plt.plot(time, signalSum)
plt.show()
```

график зависимости после подавления шума

Figure 1



x=0.617 y=-1.21

Преобразования Фурье для передачи данных

```
signalFFT = np.fft.fft(signalSum)
signalFFTabs = 2 * np.abs(signalFFT) / fs
```

Результат программы:

server.py

```
C:\Windows\system32\cmd.exe - python server.py
Microsoft Windows [Version 10.0.19041.985]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\danil>cd Desktop
C:\Users\danil\Desktop>python server.py
server is running, please, press ctrl+c to stop
Traceback (most recent call last):
  File "server.py", line 53, in <module>
    plt.hist(sig, bins=list(range(-s*3, s*3)), color='C1', label=r'$\mu = %d, \sigma$ = %d' % (m, s))
NameError: name 'sig' is not defined

C:\Users\danil\Desktop>python server.py
server is running, please, press ctrl+c to stop
new connection from ('127.0.0.1', 57214)
b'hello from client number 0'
Traceback (most recent call last):
  File "server.py", line 64, in <module>
    connection.send(float('signal= ' + signalFFTabs, encoding='UTF-8'))
numpy.core._exceptions.UFuncTypeError: ufunc 'add' did not contain a loop with signature matching types (dtype('<U32'), dtype('<U32')) -> dtype('<U32')

C:\Users\danil\Desktop>python server.py
server is running, please, press ctrl+c to stop
new connection from ('127.0.0.1', 57270)
b'hello from client number 0'
new connection from ('127.0.0.1', 57271)
b'hello from client number 0'
```

client.py

```
C:\Windows\system32\cmd.exe
_pickle.UnpicklingError: invalid load key, '\x19'.

C:\Users\danil\Desktop>python client.py
b'\x19\xe2\x17\xb7\x99\x9f<\x00\x00\x00\x00\x00\x00\x00\x00.1 \xda\xef{\x98<7\x02v\xbd\x9b_ \x8c&\x86\xbb%\x8a\xc6\x86<\xd
6MCHt<\x97MH\x0e0\x12v<\xc1~ \xf5K\xcc|<\x81\xd6_ \x8b\xdf\xebT<\xd0\xfa\x8sNWg<V\x07\x94\x17x\xa9r<\xf5\x11\x91\xa1,K
t<fD\xcd{\xfdf f<0?\x81o\x9b\x13Y<\x8e\xd1\xd8jN\xdcY<<H\xaa\x85b\x07g<\x0c&Y]\x93)X<\xb1\xddAsB\xdf{<\xb9\xb7\xdcI&\x06T
<\xef\x1c3u8\xcf\x0e<H\r\x12\xa6L\x0eq<\xfb\x98\x85-\x88\xad]<\xbcb\x0e\x13\x12jx]<\x84#)\x1e\nTa<v\x14\x08\xcax\x8dm<e*\
xab\xbd7C<\xaeL@<\xd7\x8e\x906<]I\t\xfb5\xfc"t<Q\xbb7\xad\xa6\x02\xbb8_<o1\xdd\x04/\xbbe<\xa8C\xde4\x09\x97G<\x86\x96\xfb
c\xcd-R\x7f<\xd8\x89z_ \x1c\xbb9:<\x0c\x99\x05\xcc-nw<\xc7!!\xbbb^PT<\xa8E(9(\x02\<\x0c;8\x00zYn< \x966\xfa\x00\x0fr<`Ae\x
1e\x978B<\xe3\x85\x1a\x90\x92\x14d</\xc0\xfa]\x93\x8fe<B\x02;\x7f\x02;p<[-\xf5\x1a\xa53g<V\xfb\x0b}\x97\xcb^<0\n\xfb3\xfe
:\x89h<d\xef\xefa\x09:\x08<\x9a\x0ef\x9a\x98\xbaD<9a\x03\x1e\x8dT\x83<\xd6\x06\xab\xdba\x03g<M\x0aey%\xec\xa5u<\xeb\x03\x
883k'\x0c-\x18,\xc9C<\xa1(a\x05\x1a\x00[<\xf0X,j\x07[<d8bm<\xddf<1ru\x08\x9anW<\xc3\x07\x85C\x8c\x9cN<\x0e\xde\xfb6\x18
\x07\xa0x<0]"<\xe4\x058d<\xe8\x92\x07#\xae\x9e5<\xaa\xa216\x05\xfb6Q<\xf0(\xeb\xdd\x81\x03<3<j<\xd3\x07\xfa\x06d<k\x86^Z<[\x
fdY<8\xeb\xfb3\x07\xfb8Z<?v\x01\x9a Cb<\xad\x09C\xde,\x13]<\xf2.\x97\x94\x1f\x13_<I\xa9\x95\x9d\x0aW_<B8\x90\xfb6X"\x83<Z\
n\x04\x1c\xa8\xa2R<\xc0\x09\xec\x9d\x12\x84h<\xaa\x9dI\x86z.m<\n\xa5V\x03T\x05\x80<\x02\x06\xcc\x12\xfb5%I<\x835\xbd\x09\
x91Ke<\xfdf\xaf\x124\xcb\x1dJ<\xf4g\xfa\xfbj\x94g<\xee\x07Q7\x05\x0d1k<\x16\nZ\x04\xfbfNM<\xb9Wb\x9a\xab3t<\xb3\x16\x04\x0e1
\x83%\x81<\xdb^\x00\x9a\xa3ja<ZE\xfe\x0ff\x15g<4\x80\x0b0\x13\xfb3.y<\xac\xdd\xec\x81=~~\xb3\xcat\x03\x0d1\x15Q<\xa7-\xf5\
x0e\x1d\x12\x85<\x0b` \x87\xab*\xb9y<\x14\xbe%\xba\x82\x0c1<\x94\x1a\x8cj8!e<\x14\x18"\xb0E\xdag<0\x09\xfb6\x07\x19\xdaC<m
<\xc15=Ms<\x1bT[\xfdf\xfb9b<\x9a\x0b8.t\x00g<\xb2\xcc\x0b9\x0d\x1a\x055<\x05\x0b7\x13\x9d~<\xc78\x04o\x94\x18r<X^7{
\x09Jb<\x1f5\xca\xcaq\x9es<\xd447\x05\x0Pw<T4\x12\x08Waf<\xa72\xfb4\xee\xcf\x07a<_ \x8b5\x08\x0b7\xfb5I<\x01y5\x05F\x0acv<\x
f30\xdc\x873|J<= \x0b\x07\xbd]-{<\xa2\x88\xfb9\x07p\x16j<\xf6\x00\x09\x0d\x1f0m<\r\xdc\xce\xfb7\n\x0a0Q<\xa3\x01\xfb1f\x03\x1
0s<\x1cF\x16E\xbaH8<4\x81\xba\x8f\x82\x07d<\x93\x06NA+\x98G<\xebB\xa4\x02\x0b\x0bdA<&\x08T\x90\x85..<v\x0f\x86\xa7Nht<\xd
9\xfb0\x1b8\x058P<z\xa04)\xdd)t<\x8f\x14\x05U\x8d\xaf\<\<\x8d\x8dH\x0f\xfb\x1fs<4\xa3\x01\x0b\x9a\xcdG<\x8a\x08\x0b6\xfb2F\x
b8Z<\x12E\x0b0\x06P\x93X<\xf0\xfb5\tz$\xabp<\xe0eI\x1c\x09bP<\xa4\x04\x04:\xf4?]<'
Traceback (most recent call last):
  File "client.py", line 19, in <module>
    data_arr = pickle.loads(data)
_pickle.UnpicklingError: invalid load key, '\x19'.

C:\Users\danil\Desktop>
```

Код программы:

server.py :

```
import socket
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, rfft

SERVER_ADDRESS = ('localhost', 8686)

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(SERVER_ADDRESS)
server_socket.listen(10)
print('server is running, please, press ctrl+c to stop')

binary_sequence = "000011110110011101111110010100000100100110010111"
fs = 10000
time = np.arange(np.ceil(fs)) / fs
dx = []
dy = []
signals = []
phase = []
j = 0
x = []
for i in range(0, fs * len(binary_sequence)):
    x.append(i * 0.0001)
for i in range(0, len(binary_sequence) - 1, 6):
    dx.append(1 / np.sqrt(42) * ((1 - 2 * int(binary_sequence[i])) * (
        4 - (1 - 2 * int(binary_sequence[i + 2])) * (2 - (1 - 2 *
int(binary_sequence[i + 4]))))))
    dy.append(1 / np.sqrt(42) * ((1 - 2 * int(binary_sequence[i + 1])) * (4 -
(1 - 2 * int(binary_sequence[i + 3])) * (
        2 - (1 - 2 * int(binary_sequence[i + 5]))))))
    if dx[j] < 0:
        if dy[j] < 0:
            phase.append(5 * np.pi / 4)
        else:
            phase.append(3 * np.pi / 4)
    else:
        if dy[j] < 0:
            phase.append(7 * np.pi / 4)
        else:
            phase.append(np.pi / 4)
    j += 1

for i in range(0, len(phase)):
    signals.append(np.sin(2 * np.pi * time + phase[i]))

signalSum = 0
for i in range(len(signals)):
    signalSum = signalSum + signals[i]

signalFFT = np.fft.fft(signalSum)
signalFFTabs = 2 * np.abs(signalFFT) / fs

# График: сигнал во времени
plt.subplot(1, 1, 1)
plt.plot(time, signalSum)
plt.show()

while True:
```

```
connection, address = server_socket.accept()
print("new connection from {address}".format(address=address))

data = connection.recv(1024)
print(data)

connection.send(bytes(signalFFTabs))

connection.close()
```

client.py

```
import pickle
import socket
import time

MAX_CONNECTIONS = 1
address_to_server = ('localhost', 8686)

clients = [socket.socket(socket.AF_INET, socket.SOCK_STREAM) for i in
range(MAX_CONNECTIONS)]
for client in clients:
    client.connect(address_to_server)

for i in range(MAX_CONNECTIONS):
    clients[i].send(bytes("hello from client number " + str(i),
encoding='UTF-8'))

for client in clients:
    data = client.recv(1024)
    print(str(data))
    data_arr = pickle.loads(data)
    sock.close()
    print(repr(data_arr))
```