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 последовательностью (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. 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 поднескущих, использую при этом преобразование Фурье, оценить скорость передачи данных по количеству переданных символов. Размер пакета (количество байт, которое будет отправлено за одну операцию) определяете самостоятельно (дальше 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]))
```

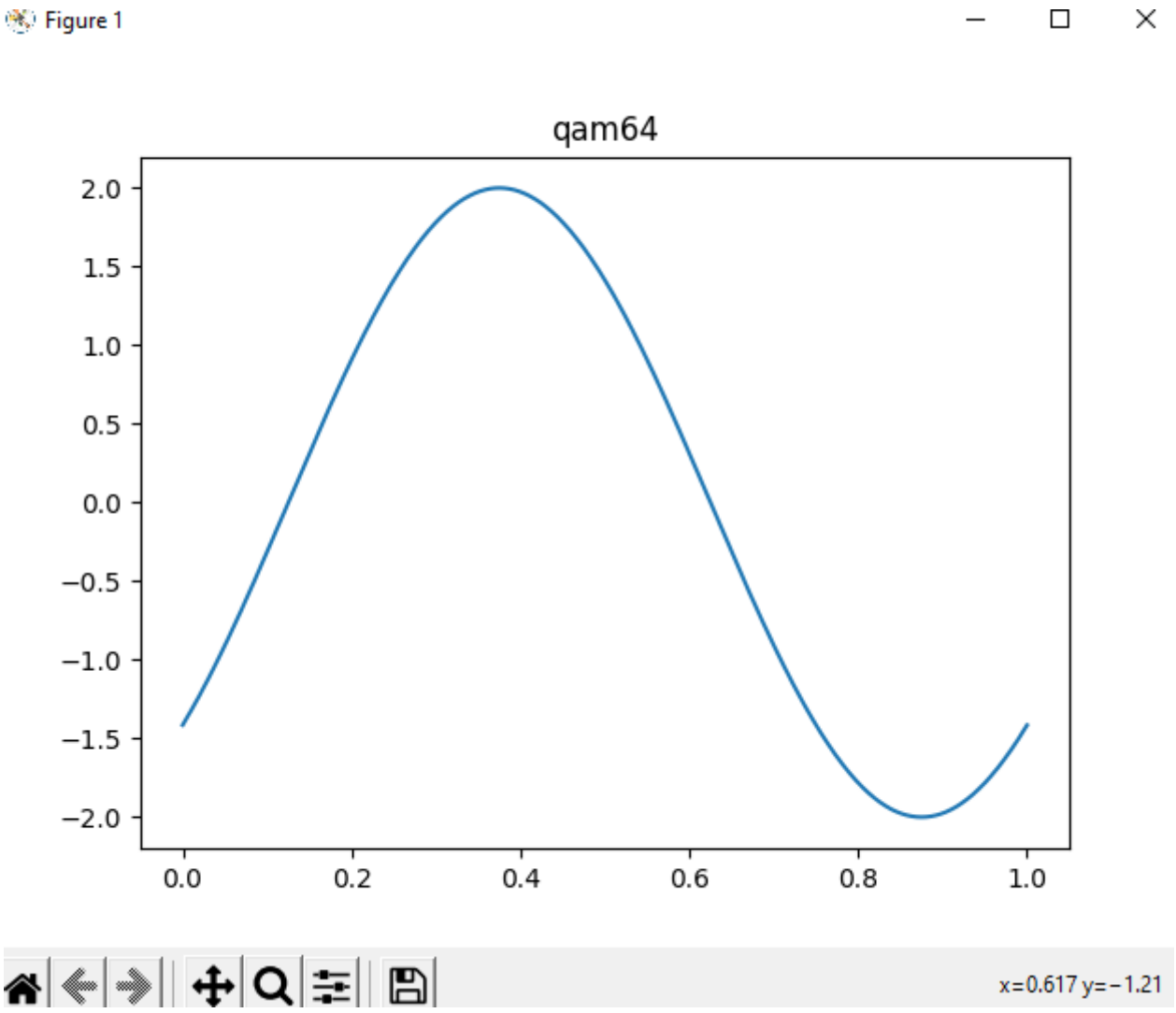
Вывод

```
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

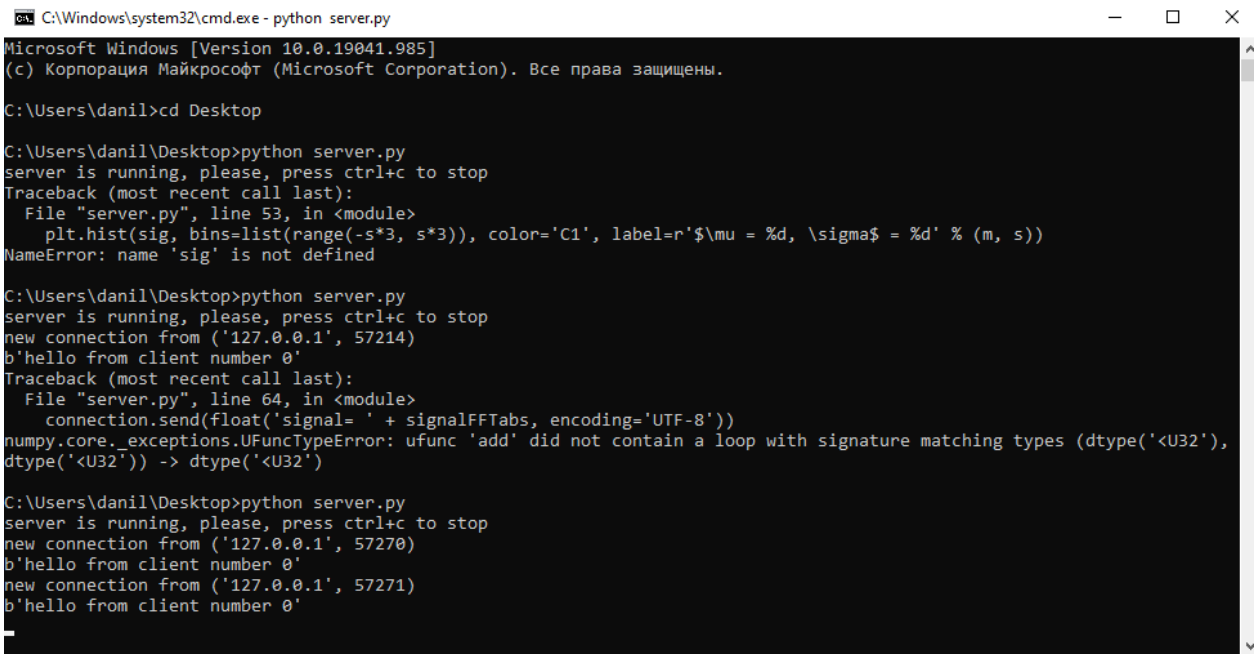


Преобразования Фурье для передачи данных

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

Результат программы:

server.py



client.py

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))
```