Федеральное агентство связи

СибГУТИ

Кафедра телекоммуникационных сетей и

вычислительных средств (ТС и ВС)

Дисциплина

Сети ЭВМ и телекоммуникации 2.0

Лабораторная работа №3

“Преобразование Фурье”

Выполнил: студент группы ИА-832

Тиванов.Д.Е

Проверил: преподаватель

Ахпашев Р.В

Новосибирск 2021

Задание на лабораторную работу №3

Необходимо:

1) Из лабораторной №1 взять за основу сигнал, сгенерированный вами.

2) С помощью преобразования Фурье получить спектр сигнала из пункта (1). Построить график спектральной составляющей.

3) Произвести обратное преобразование фурье для, получившейся спектральной составляющей. Сравнить с оригиналом в **пункте 1**.

4) Записать с помощью микрофона свой голос (ниже показан пример).

5) Проанализировать влияние частоты **семплирования** на качество воспроизводимого звука. Сделать выводы.

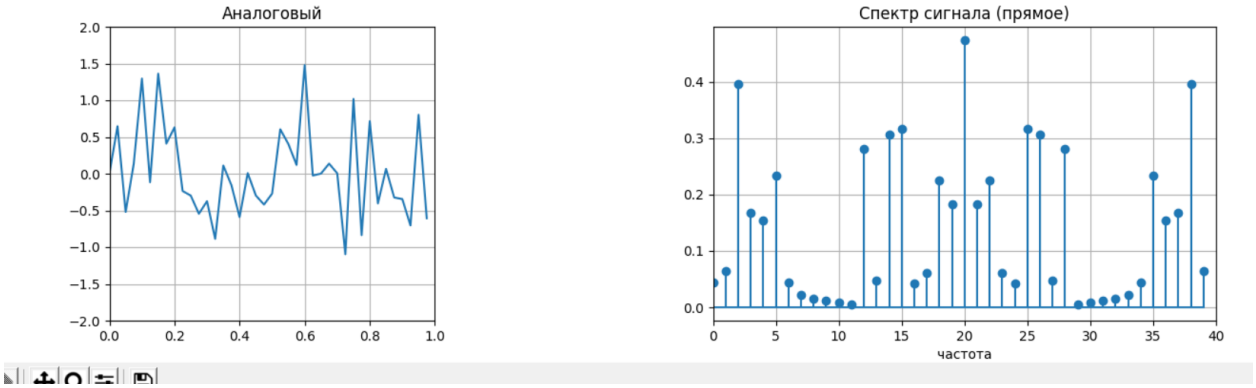
6) Получить частотный спектр, записанного звука с помощью преобразования Фурье.

7) Составить отчет.

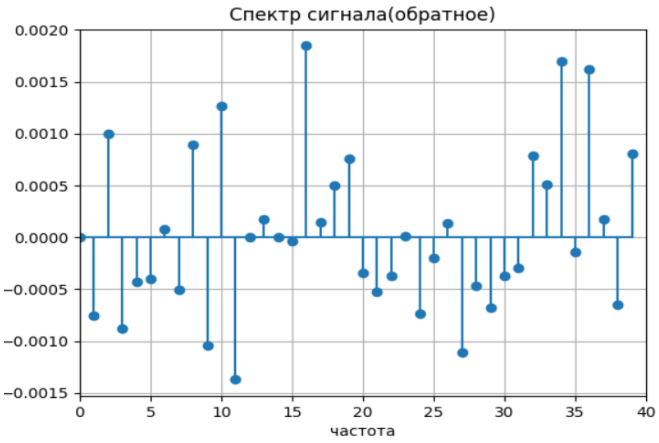
8) Для доп. баллов реализовать функцию преобразования Фурье (Дискретное преобразование Фурье сложностью самостоятельно.

```
1) import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd

duration = 1  # длительность сигнала в секундах
amplitude = 0.3  # амплитуда  (в пределах: +-1.0)
frequency = 100  # частота сигнала в [Гц]
# т.к. невозможно программно организовать аналоговый сигнал, необходимо
обозначить
# количество временных отчетов, т.е. частоту дискретизации
fs = 40
n = 128
# Расчет преобразования Фурье
timeSamples = np.arange(np.ceil(duration * fs)) / fs
SIGNALS = np.array(
    [
        amplitude * np.sin(2 * np.pi * 361.87 * timeSamples),
        amplitude * np.sin(2 * np.pi * 35.30 * timeSamples),
        amplitude * np.sin(2 * np.pi * 261.63 * timeSamples),
        amplitude * np.sin(2 * np.pi * 180.26 * timeSamples),
        amplitude * np.sin(2 * np.pi * 93.88 * timeSamples),
        amplitude * np.sin(2 * np.pi * 162.35 * timeSamples),
        amplitude * np.sin(2 * np.pi * 280.00 * timeSamples),
        amplitude * np.sin(2 * np.pi * 55.06 * timeSamples),
        amplitude * np.sin(2 * np.pi * 12.05 * timeSamples)
    ], float
                )


signalSum = 0
for i in range(len(SIGNALS)):
    signalSum = signalSum + SIGNALS[i]
#sd.play(SIGNALS, fs)
```

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

# Построение графиков
fig = plt.figure(figsize=(15, 4), dpi=100)
plt.subplot(1, 3, 1)
plt.title("Аналоговый")
plt.plot(timeSamples, signalSum)
plt.xlim([0, 1])
plt.yticks(np.linspace(np.floor(np.min(signalSum)),
np.ceil(np.max(signalSum)), 9))
plt.grid(True)

# График: спектр сигнала (прямое преобразование)
plt.subplot(1, 2, 2 )
plt.title('Спектр сигнала (прямое)')
plt.stem(signalFFTabs, basefmt='C0')
plt.xlim([0, fs])
plt.xlabel('частота')
plt.grid()
plt.show()
```



3)
```
signalIFFT = np.fft.ifft(signalSum)
signalIFFTabs = 2 * np.fft.ifft(signalIFFT) / fs
# График: спектр сигнала (обратное преобразование)
plt.subplot(1, 1, 1 )
plt.title('Спектр сигнала(обратное)')
plt.stem(signalIFFTabs, basefmt='C0')
plt.xlim([0, fs])
plt.xlabel('частота')
plt.grid()
plt.show()
```



4)
```
fs = 40000
duration = 3  # seconds
myrecording = sd.rec(duration * fs, samplerate=fs, channels=1,
dtype='float64')
myrecording = myrecording.reshape(myrecording.size)
print ("Recording Audio")
sd.wait()
print ("Audio recording complete , Play Audio")
sd.play(myrecording)
sd.wait()
```

5)
```
Чем ниже частота, тем скорость воспроизведения будет больше, при высокой всё противоположно
```

6)
```
mysignalFFT = np.fft.fft(myrecording)
mysignalFFTabs = 2 * np.abs(mysignalFFT) / fs

plt.subplot(1, 1, 1)
plt.title('Спектр сигнала записи')
plt.stem(mysignalFFTabs, basefmt='C0')
plt.xlim([0, fs])
plt.xlabel('частота')
plt.grid()
plt.show()
```