МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«Национальный исследовательский ядерный университет «МИФИ»**
ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ
КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (№12)

# Отчет по ПЗ №4

## по дисциплине «Основы теории и применения цифровой обработки данных»

**Тема:** Получение смеси сигнал + шум

Вариант: 2

Студент: Башев Григорий Алексеевич Группа С19-501
ФИО

Руководитель: Заева Маргарита Анатольевна
ФИО

Москва, 2023

## ХАРАКТЕРИСТИКА ВЫБРАННЫХ СРЕДСТВ РЕАЛИЗАЦИИ

Для выполнения данной лабораторной работы был выбран язык Python, и библиотеки Matplotlib, numpy, csv.

## ЦЕЛЬ

Цель. В выбранной среде программирования (моделирования) реализовать преобразование методом быстрого преобразования Фурье (БПФ) массивов (полученных в результате выполнения ПЗ №1-2-3).

## РАСЧЕТНАЯ ЧАСТЬ

Рассчитать длину БПФ, для сигнала – вычислить номер отсчета, соответствующего частоте заданного сигнала

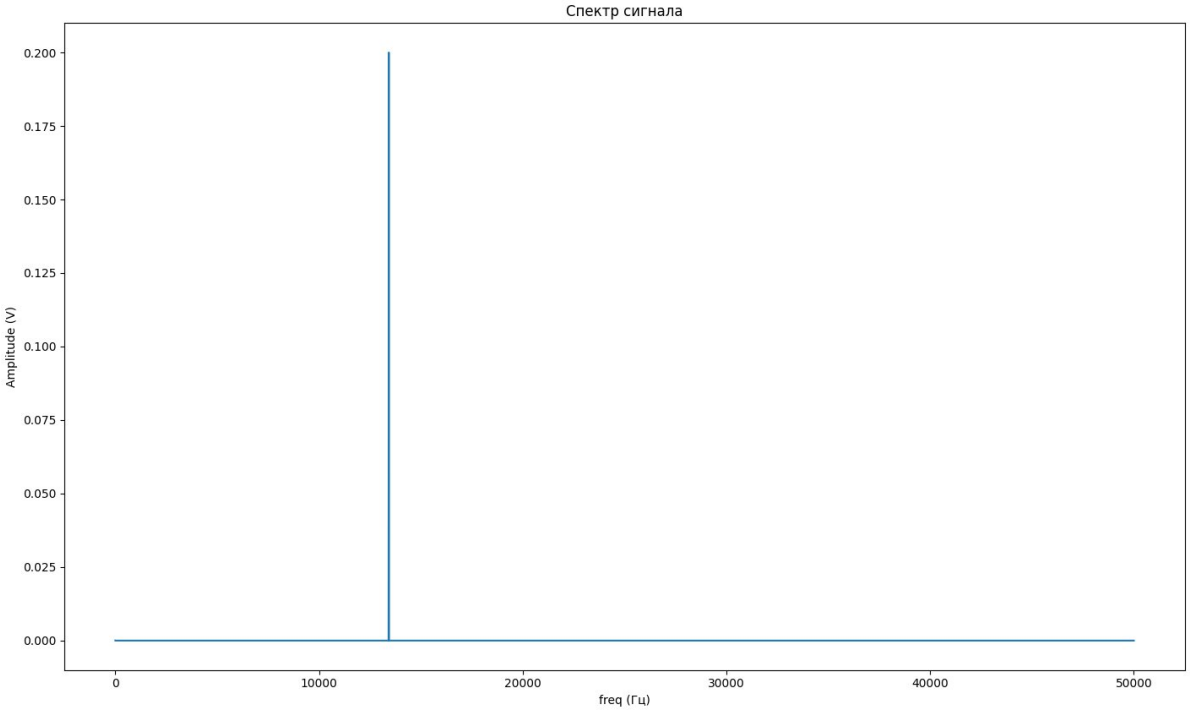
- $SNR = 1$ дБ
- N = 10 000
- $\nu = 13427.73438$ Гц
- $A_{sign} = 0{,}2$
- $\phi = 10$ град
- $\nu_{\text{дискр}} = 100$ кГц
- t = 0.1 с
- $\Delta = 0.5$ В

$$N_s = Max\left(2^k\right) < N;\ n \in \overline{0, +\infty}$$

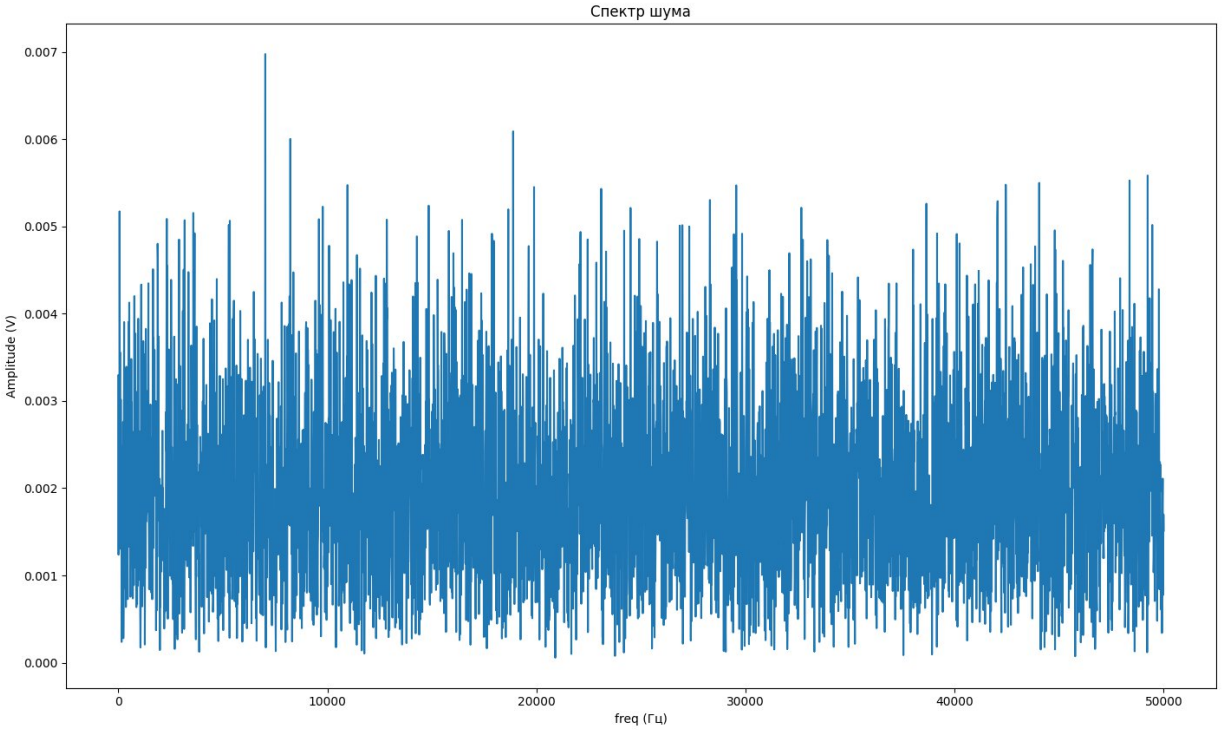
$$N_s = 8192$$

$$n = \nu \cdot \frac{N_s}{\nu_{\text{дискр}}} = 1100$$

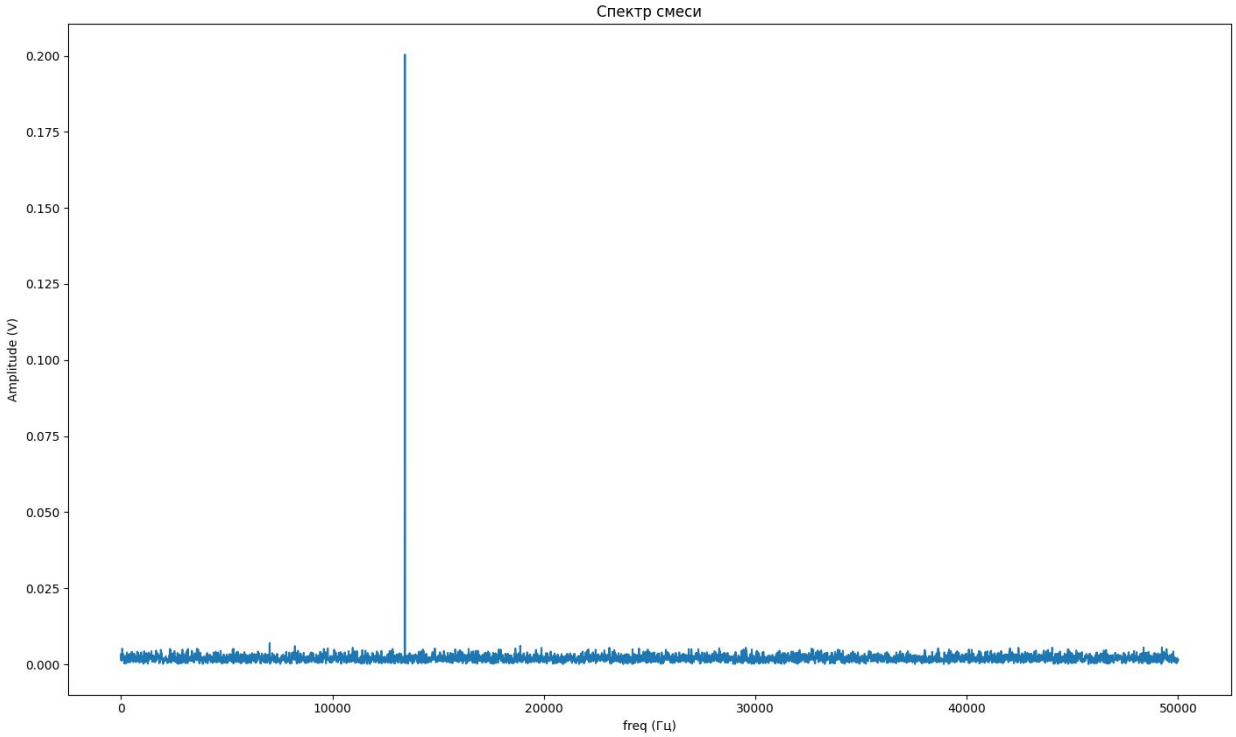
# ПРАКТИЧЕСКАЯ ЧАСТЬ



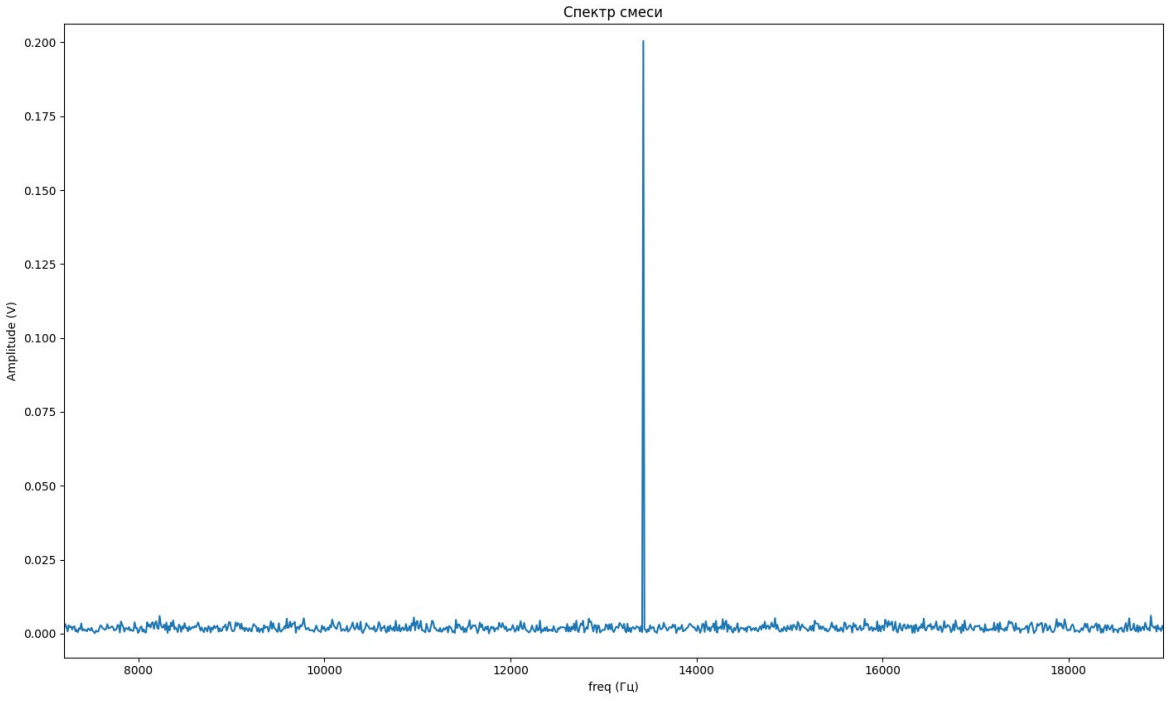

Спектр сигнала



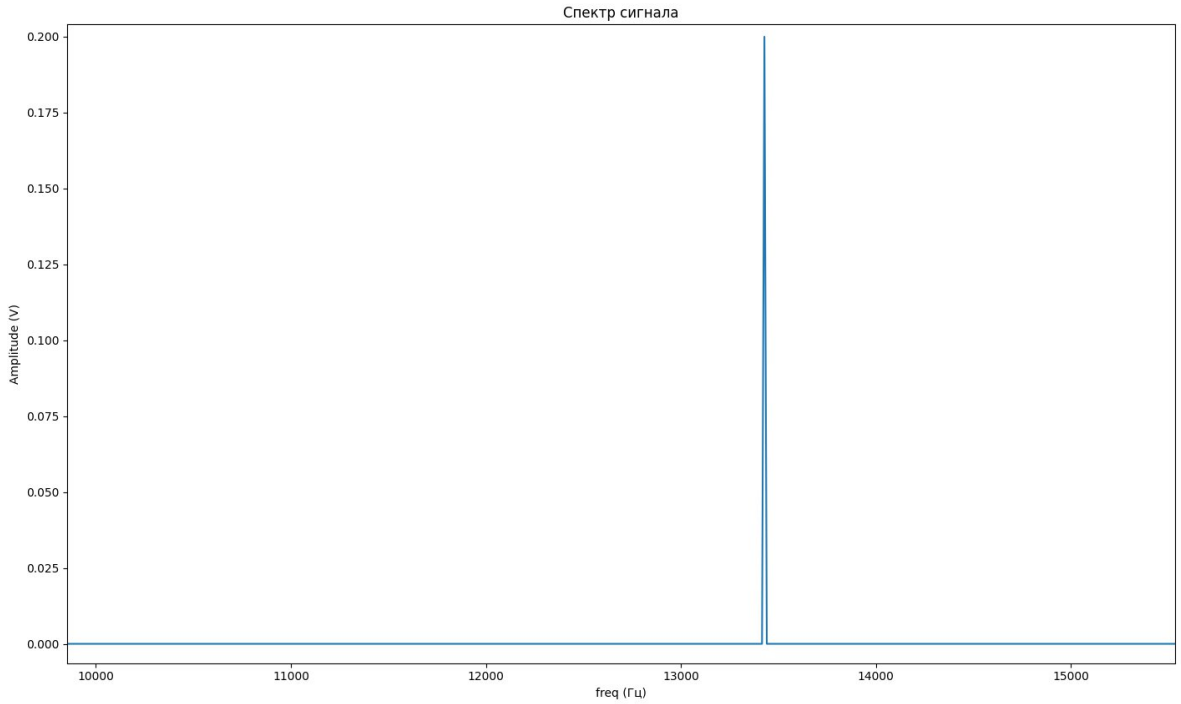

Спектр шума




Спектр смеси




Спектр смеси (5%)




Спектр сигнала (5%)

## КОД

```
import csv
import numpy as np
import matplotlib.pyplot as plt

aNoise = 0.17825018762674913
fftSamples = 8192

# чтение из файлов
with open('02_noise.csv', 'r') as f:
    reader = csv.reader(f)
    row1 = next(reader)
    noise = np.genfromtxt(f)

with open('02_signal.csv', 'r') as f:
    reader = csv.reader(f)
    row1 = next(reader)
    signal = np.genfromtxt(f)


with open('02_sn.csv', 'r') as f:
    reader = csv.reader(f)
    row1 = next(reader)
    sampling_freq = float(row1[1])
    mixture = np.genfromtxt(f)

# центрирование сигналов
signal = signal - np.mean(signal)
```

```
noise = noise * aNoise

# Расчет спектров с помощью БПФ
signal_fft = np.abs(np.fft.rfft(signal, fftSamples)) / (fftSamples / 2)

noise_fft = np.abs(np.fft.rfft(noise, fftSamples)) / (fftSamples / 2)

mixture_fft = np.abs(np.fft.rfft(mixture, fftSamples)) / (fftSamples / 2)

freqs = sampling_freq * np.arange(0, 4097) / fftSamples
# Отрисовка графиков
plt.plot(freqs, signal_fft)
plt.xlabel("freq (Гц)")
plt.ylabel("Amplitude (V)")
plt.title("Спектр сигнала")
plt.show()

plt.plot(freqs, noise_fft)
plt.xlabel("freq (Гц)")
plt.ylabel("Amplitude (V)")
plt.title("Спектр шума")
plt.show()

plt.plot(freqs, mixture_fft)
plt.xlabel("freq (Гц)")
plt.ylabel("Amplitude (V)")
plt.title("Спектр смеси")
plt.show()

# Расчет SNR
idx_signal = np.logical_and(freqs >= 13400, freqs <= 13450)
e_of_signal = np.trapz(mixture_fft[idx_signal]**2, x=freqs[idx_signal])

idx_noise = np.logical_or(freqs < 13400, freqs > 13500)
e_of_noise = np.trapz(mixture_fft[idx_noise]**2, x=freqs[idx_noise])

snr_db = 10 * np.log10(e_of_signal / e_of_noise)
print("SNR = ", snr_db)
print("Max index = ", mixture_fft.argmax())
```

SNR = 2.83237395662353
Max index = 1100

SNR не совпал с ожидаемым значением, это связано с погрешностью, возникающей при использовании БПФ.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был построены спектры шума, сигнала и смеси. Так с использованием спектра был расчитан SNR, который не совпал с используемым для изначальной генерации шума.

**Дополнительное задание:**

Рассчитаем SNR для 10 000 отсчетов, и проведем исследование зависимости SNR от количества включенных в энергию сигнала отсчетов:

```
import csv
import numpy as np

# чтение из файлов
with open('02_sn.csv', 'r') as f:
    reader = csv.reader(f)
    row1 = next(reader)
    sampling_freq = float(row1[1])
    mixture = np.genfromtxt(f)

fftSamples = 10000

# Расчет спектров с помощью БПФ
mixture_fft = np.abs(np.fft.rfft(mixture, fftSamples)) / (fftSamples / 2)
signalArg = np.argmax(mixture_fft)

# Рассмотрим разное колличество осчетов, которое может прибавляться к
энергии сигнала
print("index of signal =")
for radius in range(10):
    power_of_signal = 0
    for i in range (signalArg - radius, signalArg + radius + 1):
        power_of_signal += mixture_fft[i] ** 2
    power_of_noise = np.trapz(mixture_fft**2) - power_of_signal
    snr_db = 10 * np.log10(power_of_signal / power_of_noise)
    print("SNR для отсчетов [%d-%d] = %f" %(signalArg - radius, signalArg
+ radius, snr_db))
```

**Результат:**

index of signal = 1343
SNR для отсчетов [1343-1343] = 0.906332
SNR для отсчетов [1342-1344] = 2.098299
SNR для отсчетов [1341-1345] = 2.388053
SNR для отсчетов [1340-1346] = 2.483670
SNR для отсчетов [1339-1347] = 2.541564
SNR для отсчетов [1338-1348] = 2.583214
SNR для отсчетов [1337-1349] = 2.614013
SNR для отсчетов [1336-1350] = 2.642729
SNR для отсчетов [1335-1351] = 2.660754
SNR для отсчетов [1334-1352] = 2.683518

Из полученных результатов видно, что наиболее близко к целевому значению(SNR = 1) находится результат полученный при учете только отсчета самого сигнала, без учета соседних отсчетов.