

Санкт-Петербургский государственный политехнический  
университет Петра Великого

**Высшая школа интеллектуальных систем и  
суперкомпьютерных технологий**

Лабораторная работа №4

Шум

Выполнил студент 3-го курса  
группа 3530901/80201  
Матвеец Андрей Вадимович

Преподаватель:  
Богач Наталья Владимировна

Санкт-Петербург

2021

# Содержание

1	Часть №1: A Soft Murmur	6
2	Часть №2: Метод Барлетта	16
3	Часть №3: BitCoin	18
4	Часть №4: Счетчик гейгера	21
5	Часть №5: Алгоритм Восса-МакКартни	24
6	Выводы	27

## Список иллюстраций

1	Звуки дождя . . . . .	6
2	Сегмент звуков дождя . . . . .	6
3	Полученный спектр сегмента . . . . .	7
4	Полученный логарифмический спектр сегмента . . . . .	8
5	Другой сегмент звуков дождя . . . . .	8
6	Полученный спектр нового сегмента . . . . .	9
7	Полученный логарифмический спектр нового сегмента . . . . .	10
8	Полученная спектограмма сегмента со звуками дождя . . . . .	10
9	Звуки волны . . . . .	11
10	Сегмент звуков волны . . . . .	11
11	Полученный спектр сегмента . . . . .	11
12	Полученный логарифмический спектр сегмента . . . . .	12
13	Другой сегмент звуков дождя . . . . .	12
14	Полученный спектр нового сегмента . . . . .	13
15	Полученный логарифмический спектр нового сегмента . . . . .	14
16	Полученная спектограмма сегмента со звуками волны . . . . .	15
17	Полученные спектры метода Бартлетта . . . . .	16
18	Таблица с данными цен за <b>BitCoin</b> . . . . .	18
19	<b>Wave</b> на основе данных таблицы . . . . .	19
20	Построенный логарифмический спектр . . . . .	19
21	Наклон спектра . . . . .	20
22	Созданный сигнал . . . . .	21

23	Количество частиц . . . . .	21
24	Визуализация полученных значений . . . . .	22
25	Полученная мощность спектра . . . . .	22
26	Наклон спектра . . . . .	23
27	Текст класса и перевод в аудио . . . . .	24
28	Просмотр полученной аудиодорожки . . . . .	25
29	Полученный спектр мощности . . . . .	25
30	Наклон спектра . . . . .	26

## Листинги

1	Получение спектра сегмента . . . . .	6
2	Получение логарифмического спектра сегмента . . . . .	7
3	Получение спектра нового сегмента . . . . .	8
4	Получение логарифмического спектра нового сегмента . . . . .	9
5	Получение спектограммы для сегмента . . . . .	10
6	Получение спектра сегмента . . . . .	11
7	Получение логарифмического спектра сегмента . . . . .	12
8	Получение спектра нового сегмента . . . . .	12
9	Получение логарифмического спектра нового сегмента . . . . .	13
10	Получение спектограммы для сегмента . . . . .	14
11	Реализованный метод Бартлетта . . . . .	16
12	Получение спектограммы для сегмента . . . . .	16
13	Получение таблицы с данными цен за <b>BitCoin</b> . . . . .	18
14	Получение данных из таблицы . . . . .	18
15	Построение <b>wave</b> . . . . .	18
16	Построение логарифмического спектра . . . . .	19
17	Построение логарифмического спектра . . . . .	21
18	Визуализация полученных значений . . . . .	21
19	Мощность спектра . . . . .	22
20	Класс <b>voss</b> . . . . .	24
21	Просмотр полученной аудиодорожки . . . . .	24
22	Получения спектра мощности . . . . .	25

# 1 Часть №1: A Soft Murmur

В первой части лабораторной работы нам необходимо скачать файлы с шумом природы, например дождь или морские волны. Выделить из этих сигналов спектры и установить, на какой шум похож каждый сигнал.

В качестве аудиофайлов были выбраны звуки дождя и звуки прибрежных волн. Начнем со звуков дождя.

Прочитаем этот файл и сохраним в переменную `wave`, после чего выведем полученную аудиодорожку:

```
In [3]: from thinkdsp import read_wave

        wave = read_wave('243627__lebaston100__heavy-rain.wav')
        wave.make_audio()
```

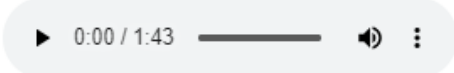
Out[3]: 

Рис. 1: Звуки дождя

После этого выделим сегмент длиной 1с :

```
In [4]: segment = wave.segment(start=2, duration=1.0)
        segment.make_audio()
```


Out[4]: 

Рис. 2: Сегмент звуков дождя

Теперь посмотрим на спектр данного сегмента:

```
1 spectrum = segment.make_spectrum()
2 spectrum.plot_power()
3 decorate(xlabel='Frequency (Hz)')
```

Листинг 1: Получение спектра сегмента

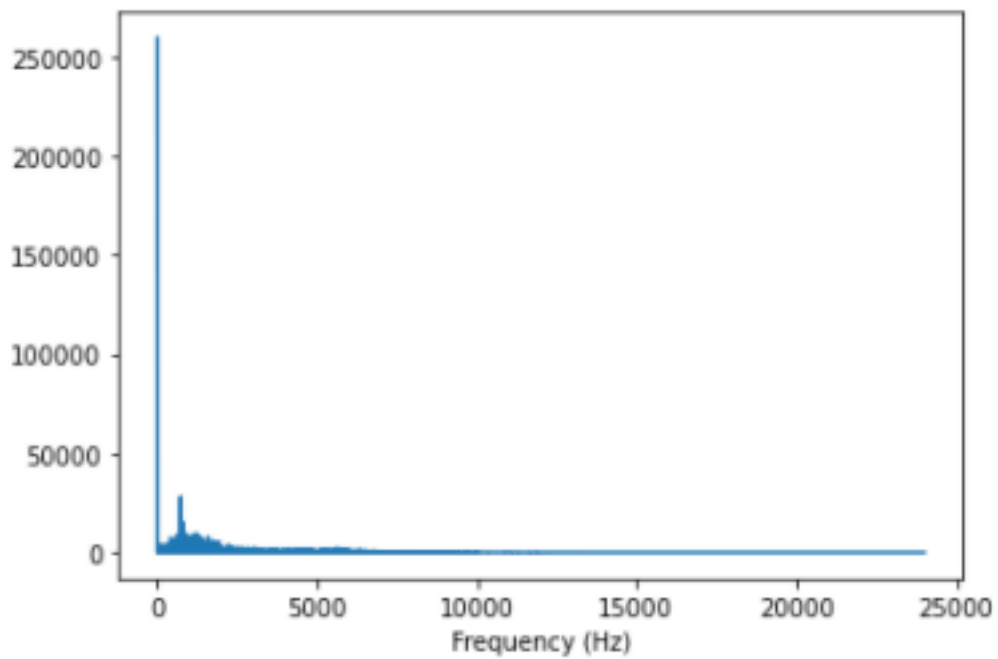


Рис. 3: Полученный спектр сегмента

Судя по тому, что большей амплитуде соответствует меньшее значение частоты, то можно сделать вывод о том, что это "красный" или "розовый" шум.

Теперь снова построим спектр сегмента, только в этот раз в логарифмическом масштабе:

```

1 spectrum.plot_power()
2 loglog = dict(xscale='log', yscale='log')
3 decorate(xlabel='Frequency (Hz)', **loglog)

```

Листинг 2: Получение логарифмического спектра сегмента

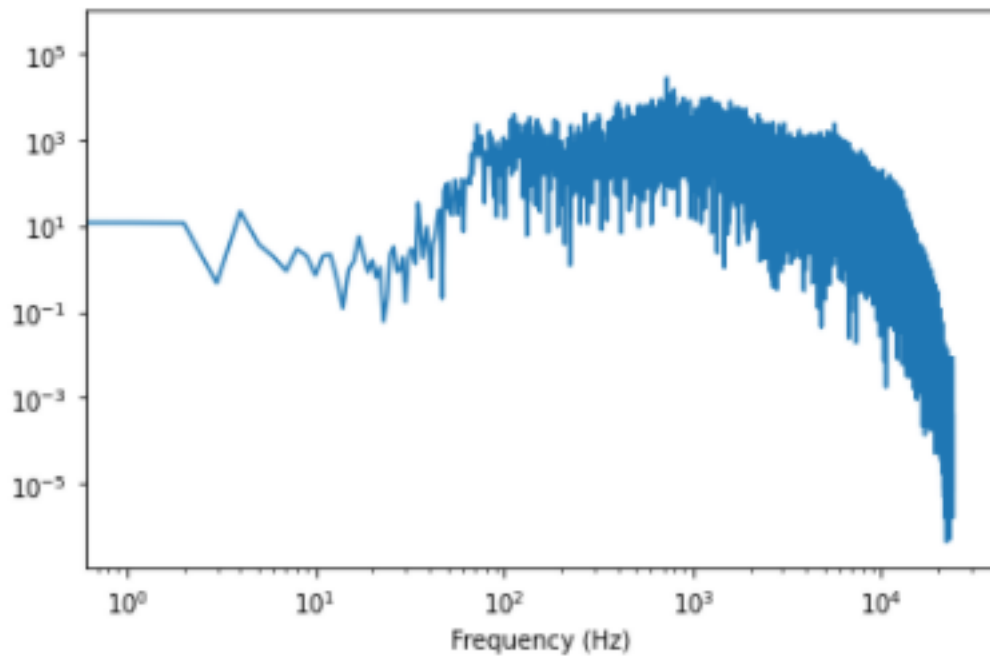


Рис. 4: Полученный логарифмический спектр сегмента

Далее возьмем какой-нибудь другой сегмент нашего сигнала и произведем все те же самые манипуляции:

```
In [7]: segment2 = wave.segment(start=5, duration=1.0)
        segment2.make_audio()
```

Out[7]:

▶ 0:00 / 0:01 🔊 ⋮

Рис. 5: Другой сегмент звуков дождя

Также сначала получим спектр для нового сегмента:

```
1 spectrum2 = segment2.make_spectrum()
2 spectrum.plot_power(alpha=0.5)
3 spectrum2.plot_power(alpha=0.5)
4 decorate(xlabel='Frequency (Hz)',
5          ylabel='Amplitude')
```

Листинг 3: Получение спектра нового сегмента



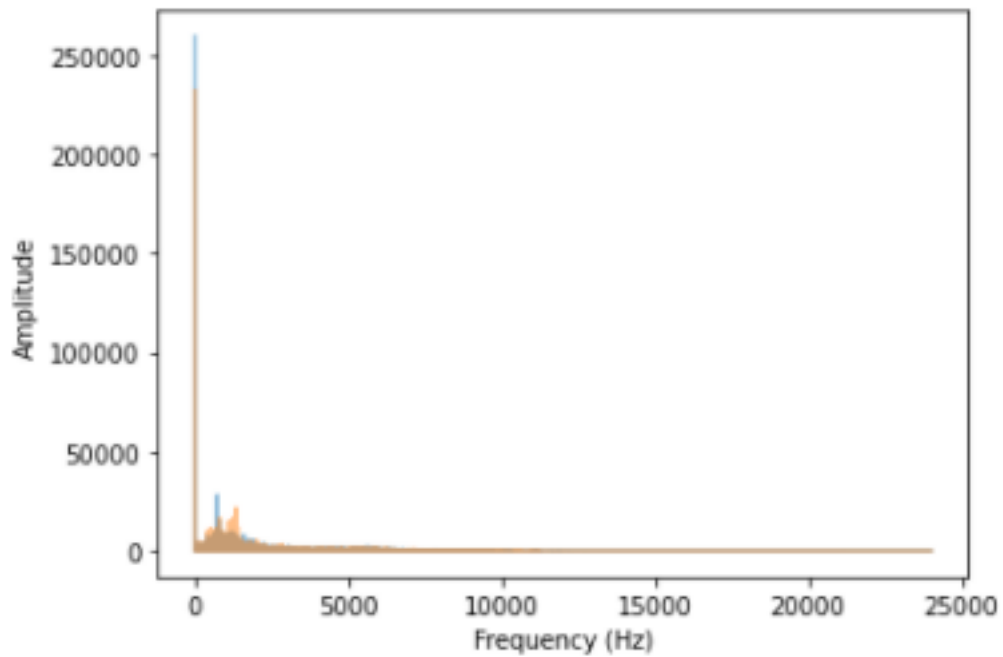


Рис. 6: Полученный спектр нового сегмента

Здесь, на основании полученного спектра можно так же предположить, что это "красный" или "розовый" шум.

Далее так же построим для нового сегмента спектр в логарифмическом масштабе:

```

1 spectrum.plot_power(alpha=0.5)
2 spectrum2.plot_power(alpha=0.5)
3 decorate(xlabel='Frequency (Hz)',
4          ylabel='Amplitude',
5          **loglog)
```

Листинг 4: Получение логарифмического спектра нового сегмента

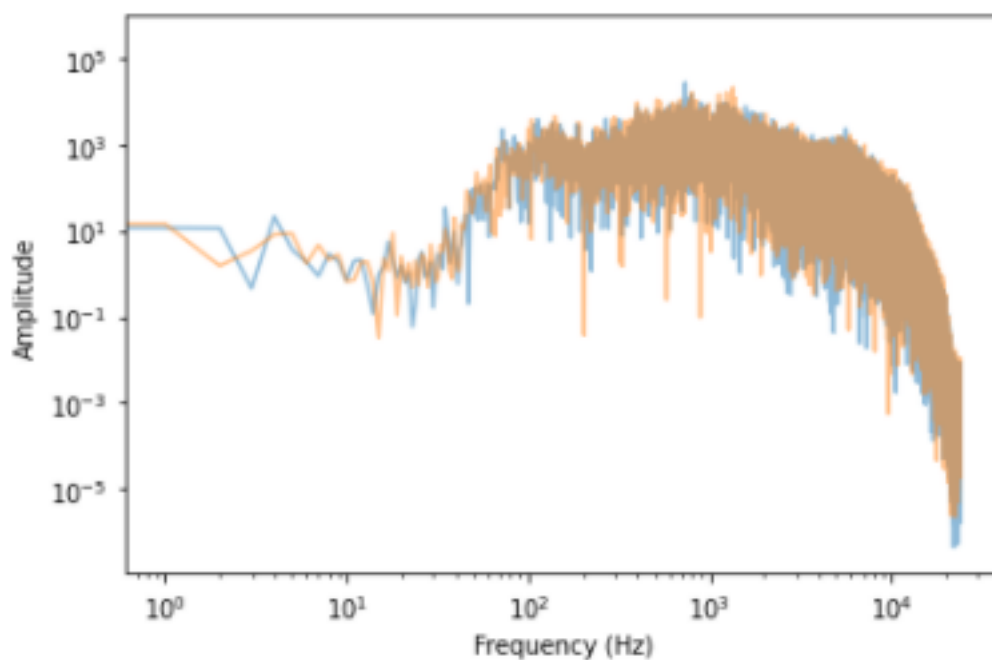


Рис. 7: Полученный логарифмический спектр нового сегмента

По полученным спектрам можно понять, что сигнал сильно не меняется на своем протяжении.

Теперь получим спектограмму для нашего сегмента:

```
1 segment.make_spectrogram(512).plot(high=5000)
2 decorate(xlabel='Time(s)', ylabel='Frequency (Hz)')
```

Листинг 5: Получение спектограммы для сегмента

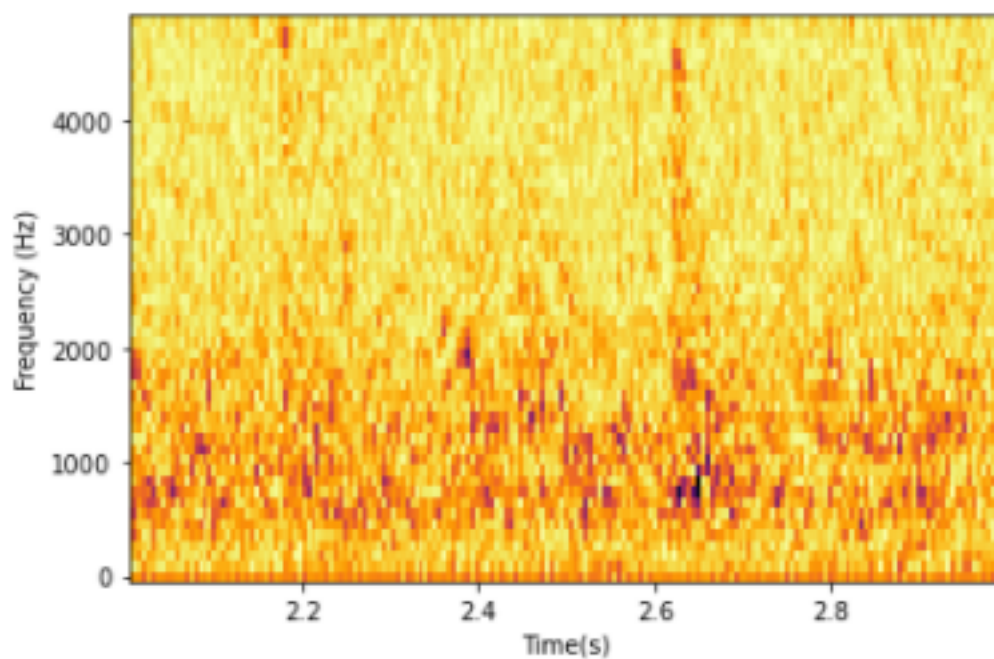


Рис. 8: Полученная спектограмма сегмента со звуками дождя

Теперь возьмем второй аудиофайл со звуками волны. Для начала так же сохраним его в переменную `wave`, после чего выведем полученную аудиодорожку:

```
In [11]: wave = read_wave('412308_straget_big-waves-hit-land.wav')
         wave.make_audio()
```

Out[11]:

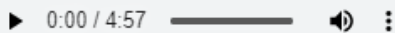


Рис. 9: Звуки волны

После этого выделим сегмент длиной 1с :

```
In [12]: segment = wave.segment(start=3, duration=1.0)
         segment.make_audio()
```

Out[12]:

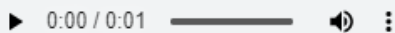


Рис. 10: Сегмент звуков волны

Теперь посмотрим на спектр данного сегмента:

```
1 spectrum = segment.make_spectrum()
2 spectrum.plot_power()
3 decorate(xlabel='Frequency (Hz)')
```

Листинг 6: Получение спектра сегмента

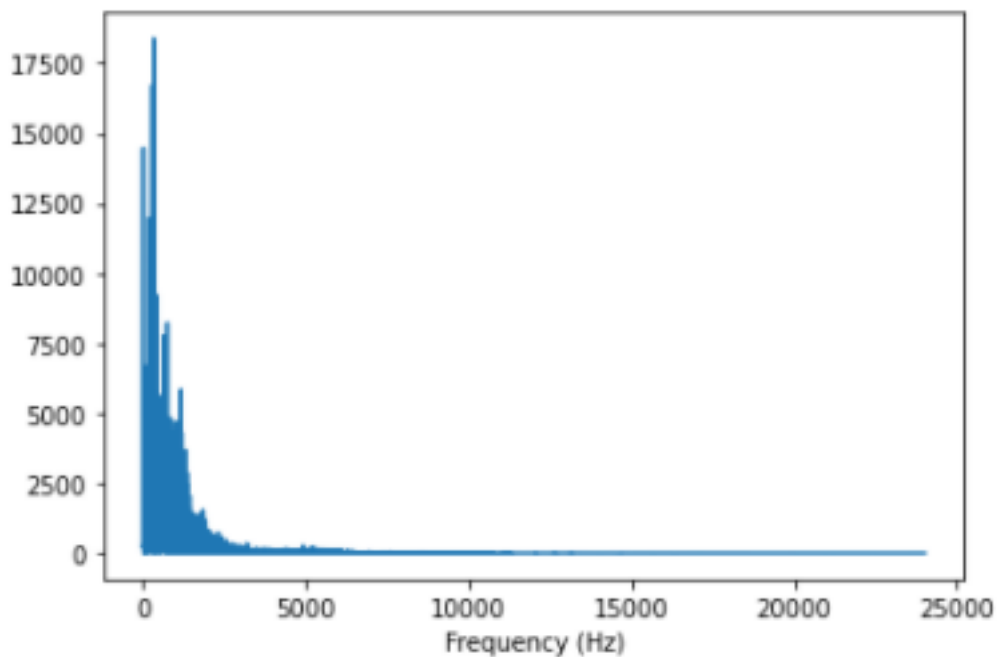


Рис. 11: Полученный спектр сегмента

Для данного сегмента со звуками волн, судя по тому, что большей амплитуде соответствует меньшее значение частоты, также можно сделать вывод о том, что это "красный" или "розовый" шум.

Теперь снова построим спектр сегмента, только в этот раз в логарифмическом масштабе:

```
1 spectrum.plot_power()
2 loglog = dict(xscale='log', yscale='log')
3 decorate(xlabel='Frequency (Hz)', **loglog)
```

Листинг 7: Получение логарифмического спектра сегмента

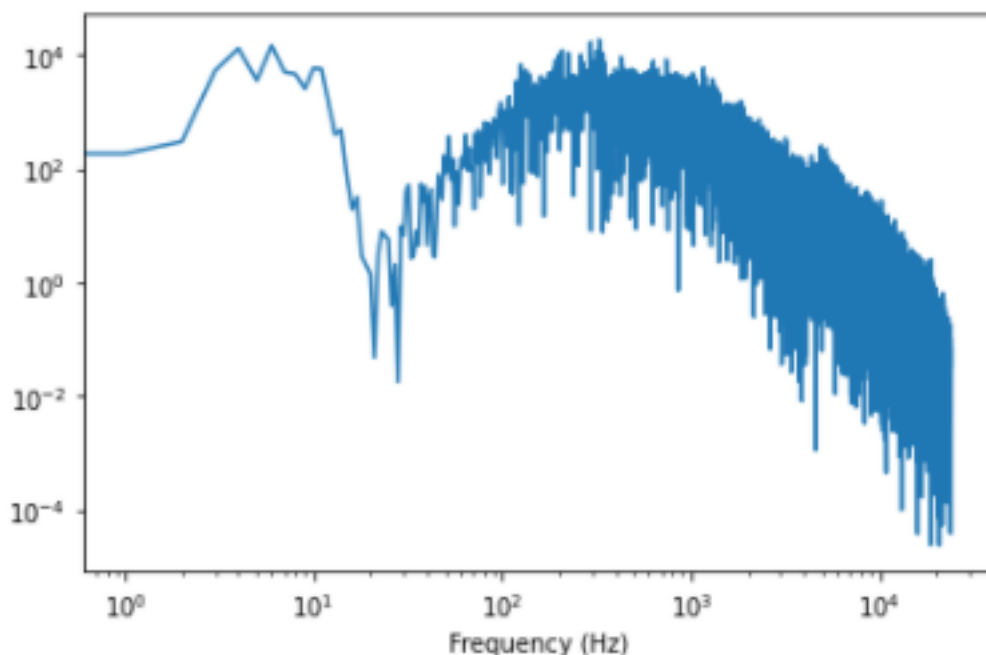


Рис. 12: Полученный логарифмический спектр сегмента

Теперь также как и до этого возьмем какой-нибудь другой сегмент нашего сигнала и произведем все те же самые манипуляции:

```
B [15]: segment2 = wave.segment(start=7, duration=1.0)
        segment2.make_audio()
```

Out[15]:

▶ 0:00 / 0:01 🔊 ⋮

Рис. 13: Другой сегмент звуков дождя

Также сначала получим спектр для нового сегмента:

```
1 spectrum2 = segment2.make_spectrum()
2 spectrum.plot_power(alpha=0.5)
3 spectrum2.plot_power(alpha=0.5)
```

```

4   decorate(xlabel='Frequency (Hz)',
5           ylabel='Amplitude')

```

Листинг 8: Получение спектра нового сегмента

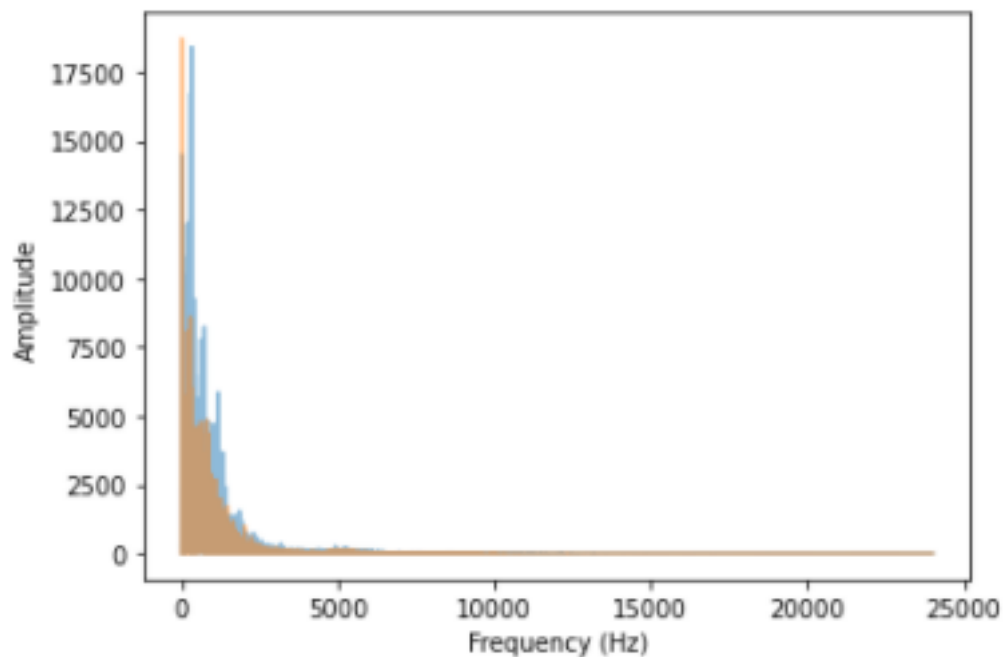


Рис. 14: Полученный спектр нового сегмента

В данном случае как и до этого, на основании полученного спектра можно так же предположить, что это "красный" или "розовый" шум.

Далее так же построим для нового сегмента спектр в логарифмическом масштабе:

```

1   spectrum.plot_power(alpha=0.5)
2   spectrum2.plot_power(alpha=0.5)
3   decorate(xlabel='Frequency (Hz)',
4           ylabel='Amplitude',
5           **loglog)

```

Листинг 9: Получение логарифмического спектра нового сегмента

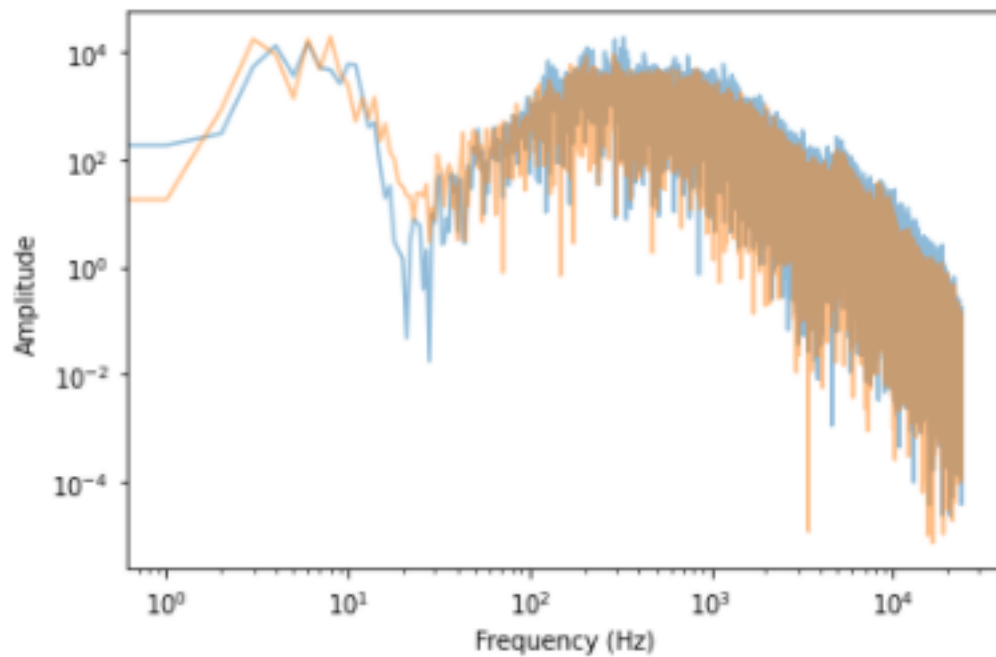


Рис. 15: Полученный логарифмический спектр нового сегмента

По полученным спектрам можно понять, что сигнал со звуками сильно не меняется на своем протяжении, только в первой части есть небольшая разница в амплитудах сигналов.

Теперь получим спектограмму для нашего сегмента:

```

1 segment.make_spectrogram(512).plot(high=5000)
2 decorate(xlabel='Time(s)', ylabel='Frequency (Hz)')
```

Листинг 10: Получение спектограммы для сегмента

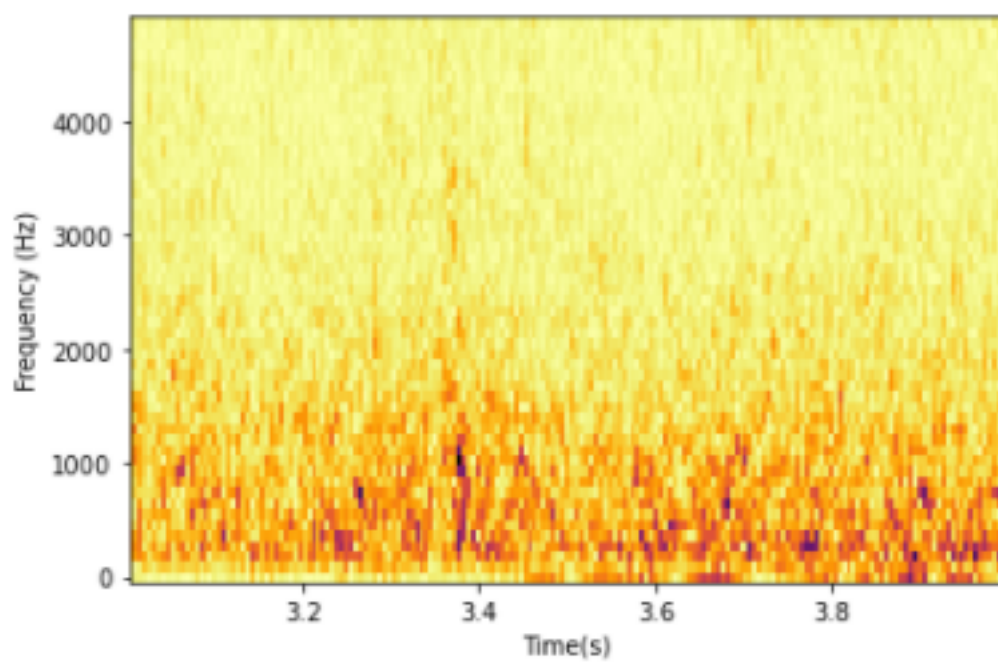


Рис. 16: Полученная спектограмма сегмента со звуками волны

## 2 Часть №2: Метод Барлетта

Во втором пункте лабораторной работы нам необходимо реализовать метод Барлетта и использовать его для оценки спектра мощности шумового сигнала.

Реализуем метод Барлетта:

```
1 def bartlett_method(wave, seg_length=512, win_flag=True):
2     spectro = wave.make_spectrogram(seg_length, win_flag)
3     spectrums = spectro.spec_map.values()
4     psds = [spectrum.power for spectrum in spectrums]
5     hs = np.sqrt(sum(psds) / len(psds))
6     fs = next(iter(spectrums)).fs
7     spectrum = Spectrum(hs, fs, wave.framerate)
8     return spectrum
```

Листинг 11: Реализованный метод Барлетта

Проверим данный метод, вызвав его два раза и выведем полученные спектры на одном графике:

```
1 psd = bartlett_method(segment)
2 psd2 = bartlett_method(segment2)
3 psd.plot_power()
4 psd2.plot_power()
5 decorate(xlabel='Frequency (Hz)',
6          ylabel='Power',
7          **loglog)
```

Листинг 12: Получение спектограммы для сегмента

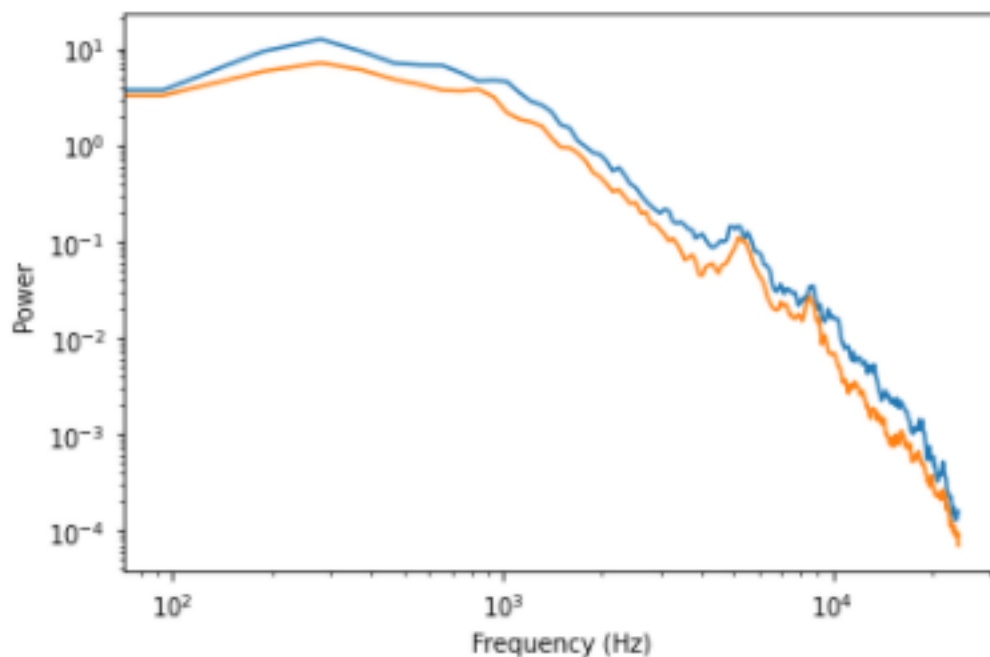


Рис. 17: Полученные спектры метода Барлетта



Проанализировав полученные спектры можно сделать вывод, что в них есть связь между частотой и амплитудой, а сама зависимость нелинейная.

### 3 Часть №3: BitCoin

В третьей части лабораторной работы нам необходимо скачать CSV-файл с историческими данными цен на BitCoin. Необходимо вычислить спектр цен BitCoin как функцию времени и установить, на какой шум он больше похож.

Для начала скачаем CSV-файл с данными цен за BitCoin:

```
1 import pandas as pd
2 data = pd.read_csv('BTC_USD_2013-10-01_2020-03-26-CoinDesk.csv')
3 data
```

Листинг 13: Получение таблицы с данными цен за BitCoin

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2013-10-01	123.654990	124.304660	124.751660	122.563490
1	BTC	2013-10-02	125.455000	123.654990	125.758500	123.633830
2	BTC	2013-10-03	108.584830	125.455000	125.665660	83.328330
3	BTC	2013-10-04	118.674660	108.584830	118.675000	107.058160
4	BTC	2013-10-05	121.338660	118.674660	121.936330	118.005660
...	...	...	...	...	...	...
2354	BTC	2020-03-22	5884.340133	6187.042146	6431.873162	5802.553402
2355	BTC	2020-03-23	6455.454688	5829.352511	6620.858253	5694.198299
2356	BTC	2020-03-24	6784.318011	6455.450650	6863.602196	6406.037439
2357	BTC	2020-03-25	6706.985089	6784.325204	6981.720386	6488.111885
2358	BTC	2020-03-26	6721.495392	6697.948320	6796.053701	6537.856462

2359 rows × 6 columns

Рис. 18: Таблица с данными цен за BitCoin

Теперь считаем все данные из этой таблицы:

```
1 datac = data['Closing Price (USD)']
2 datai = data.index
```

Листинг 14: Получение данных из таблицы

После чего построим wave из полученных данных:

```
1 from thinkdsp import Wave
2 wave = Wave(datac, datai, framerate=1)
3 wave.plot()
4 decorate(xlabel='Time (days)')
```

Листинг 15: Построение wave

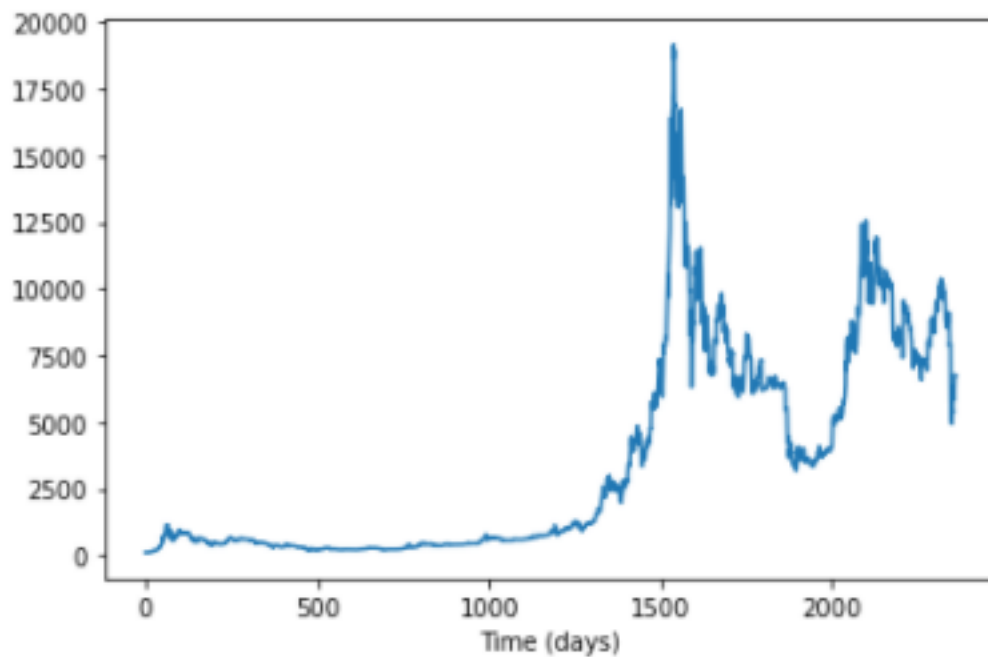


Рис. 19: Wave на основе данных таблицы

Наконец, построим логарифмический спектр для наших данных:

```

1 spectrum = wave.make_spectrum()
2 spectrum.plot_power()
3 decorate(xlabel='Frequency (1/days)', **loglog)

```

Листинг 16: Построение логарифмического спектра

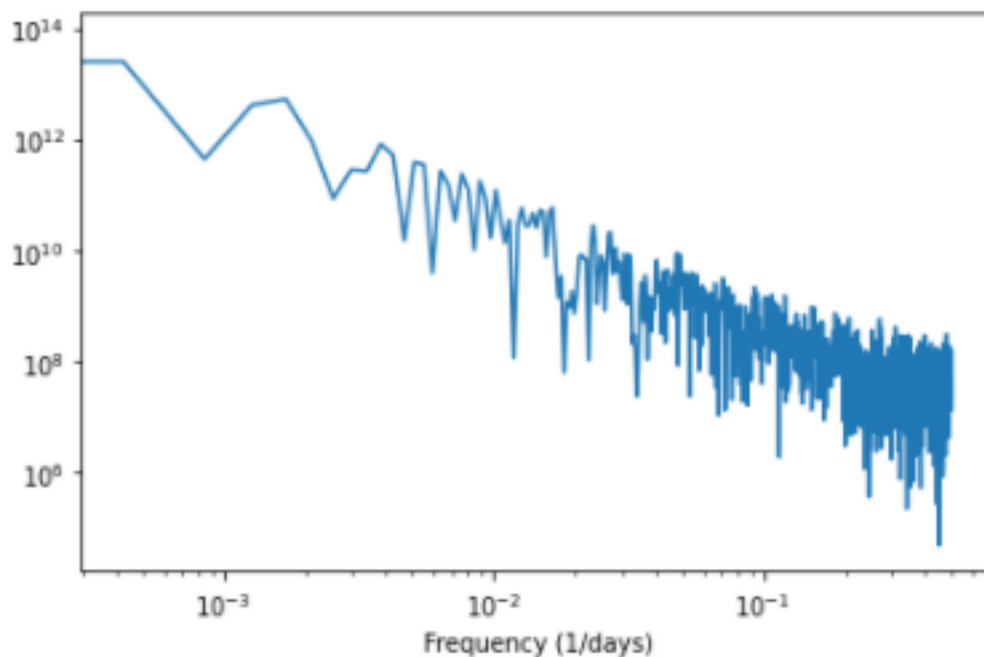


Рис. 20: Построенный логарифмический спектр

На основании полученного спектра можно осделать вывод, что это либо

«красный», либо «розовый» шум

Посмотрим на наклон полученного спектра:

```
In [25]: spectrum.estimate_slope()[0]  
Out[25]: -1.7332540936758956
```

Рис. 21: Наклон спектра

Наклон получился равным -1,73, в то время как для «красного» шума характерен наклон -2, поэтому можно сделать вывод, что перед нами «розовый» шум.

## 4 Часть №4: Счетчик гейгера

В четвертом пункте лабораторной работы нам необходимо реализовать класс `UncorrelatedPoissonNoise`, наследующий `Noise` и предоставляющий `evaluate`. Необходимо сгенерировать случайные величины из распределения Пуассона, а так же пару секунд UP и прослушать. При малых значениях `amp` звук будет похож на счетчик Гейгера, а при больших на белый шум. Вычислить и напечатать спектр для этих сигналов.

Начнем с написания класса `UncorrelatedPoissonNoise`:

```
1 from thinkdsp import Noise
2 class UncorrelatedPoissonNoise(Noise):
3     def evaluate(self, ts):
4         ys = np.random.poisson(self.amp, len(ts))
5         return ys
```

Листинг 17: Построение логарифмического спектра

После чего сразу создадим сигнал с `amp = 0,001` и представим его в виде аудио:

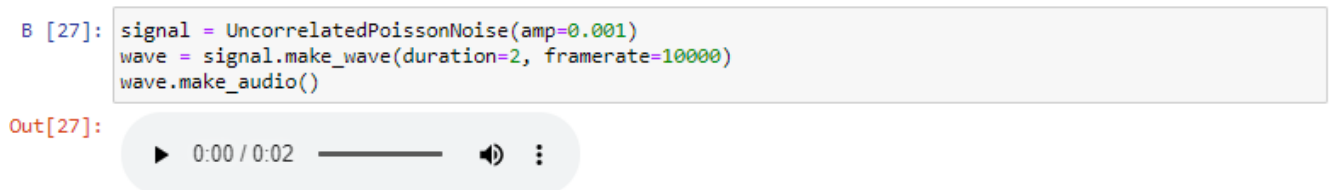


Рис. 22: Созданный сигнал

Прослушав полученную аудиодорожку можно сделать вывод, что она очень сильно похожа на звук, издаваемый счетчиком Гейгера.

Сверим ожидаемое количество частиц с полученным:

```
B [28]: expected = 0.001 * 10000 * 2
        actual = sum(wave.ys)
        print(expected, actual)

20.0 17
```

Рис. 23: Количество частиц

После этого построим полученные значения:

```
1 wave.plot()
```

Листинг 18: Визуализация полученных значений

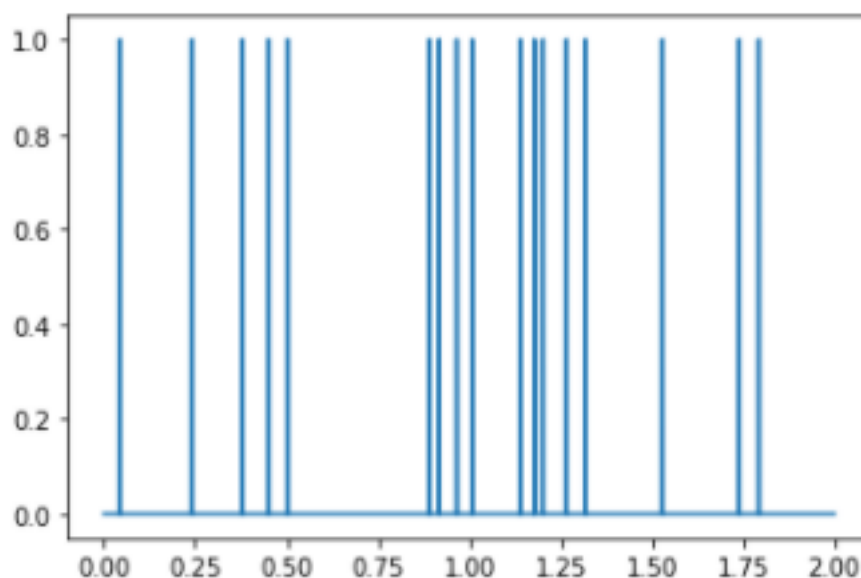


Рис. 24: Визуализация полученных значений

Теперь нам необходимо узнать мощность спектра:

```

1 spectrum = wave.make_spectrum()
2 spectrum.plot_power()
3 decorate(xlabel='Frequency (Hz)',
4          ylabel='Power',
5          **loglog)

```

Листинг 19: Мощность спектра

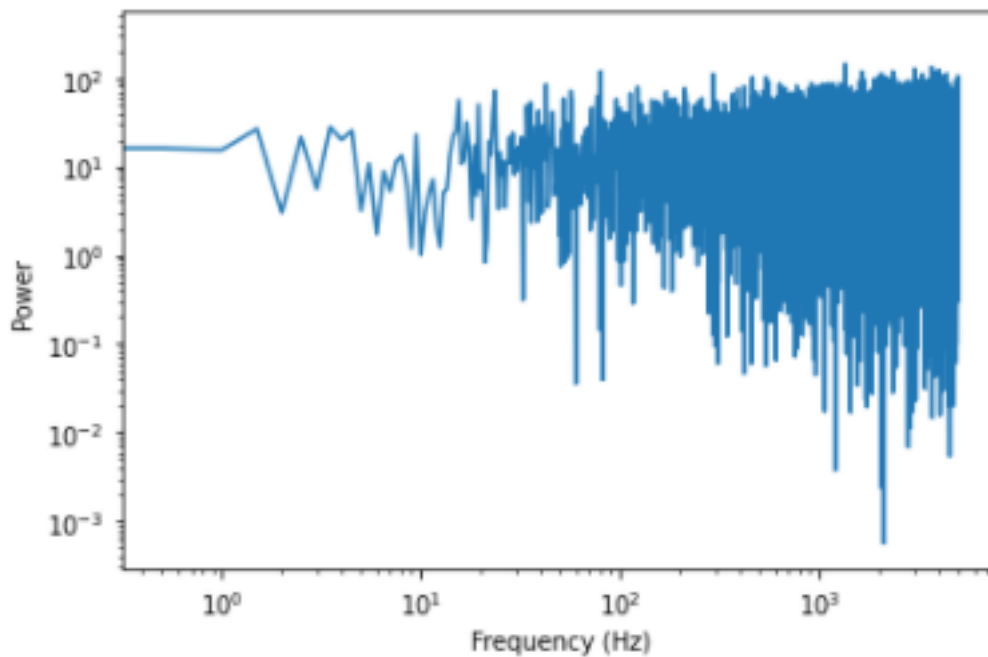


Рис. 25: Полученная мощность спектра

Также посмотрим на наклон полученного спектра:

```
In [31]: spectrum.estimate_slope().slope
Out[31]: -0.00775411243898777
```

Рис. 26: Наклон спектра

На основе полученной мощности спектра и его наклона можно сделать вывод, что наш сигнал представляет собой белый шум

## 5 Часть №5: Алгоритм Восс-МакКартни

В пятом пункте четвертой лабораторной работы нам необходимо реализовать алгоритм Восс-МакКартни, вычислить спектр и убедиться, что соотношение между мощностью и частотой соответствующие.

Для начала реализуем класс `voss`:

```
1 def voss(nrows, ncols=16):
2     array = np.empty((nrows, ncols))
3     array.fill(np.nan)
4     array[0, :] = np.random.random(ncols)
5     array[:, 0] = np.random.random(nrows)
6     n = nrows
7     cols = np.random.geometric(0.5, n)
8     cols[cols >= ncols] = 0
9     rows = np.random.randint(nrows, size=n)
10    array[rows, cols] = np.random.random(n)
11    df = pd.DataFrame(array)
12    df.fillna(method='ffill', axis=0, inplace=True)
13    total = df.sum(axis=1)
14    return total.values
```

Листинг 20: Класс `voss`

После чего сразу протестируем его, переведя полученный сигнал в аудио:

```
B [33]: wave = Wave(voss(11025))
        wave.unbias()
        wave.normalize()
        wave.make_audio()
```

Out[33]:

▶ 0:00 / 0:01 🔊 ⋮

Рис. 27: Текст класса и перевод в аудио

Полученная аудиодорожка представляет собой простой шум

Посмотрим как он выглядит:

```
1 wave.plot()
```

Листинг 21: Просмотр полученной аудиодорожки



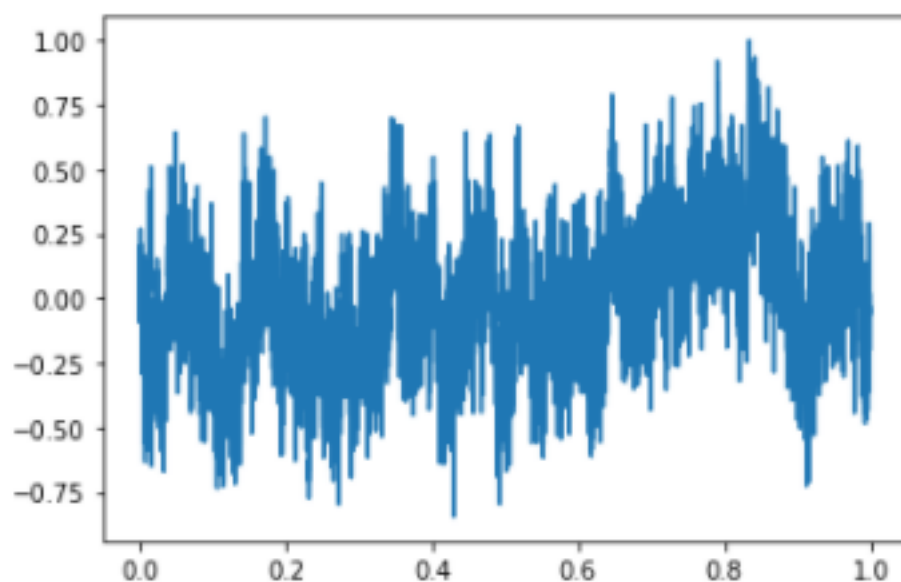


Рис. 28: Просмотр полученной аудиодорожки

После этого посмотрим на спектр мощности, чтобы убедиться в правильности зависимости частоты от амплитуды:

```

1 spectrum = wave.make_spectrum()
2 spectrum.hs[0] = 0
3 spectrum.plot_power()
4 decorate(xlabel='Frequency (Hz)',
5          **loglog)

```

Листинг 22: Получения спектра мощности

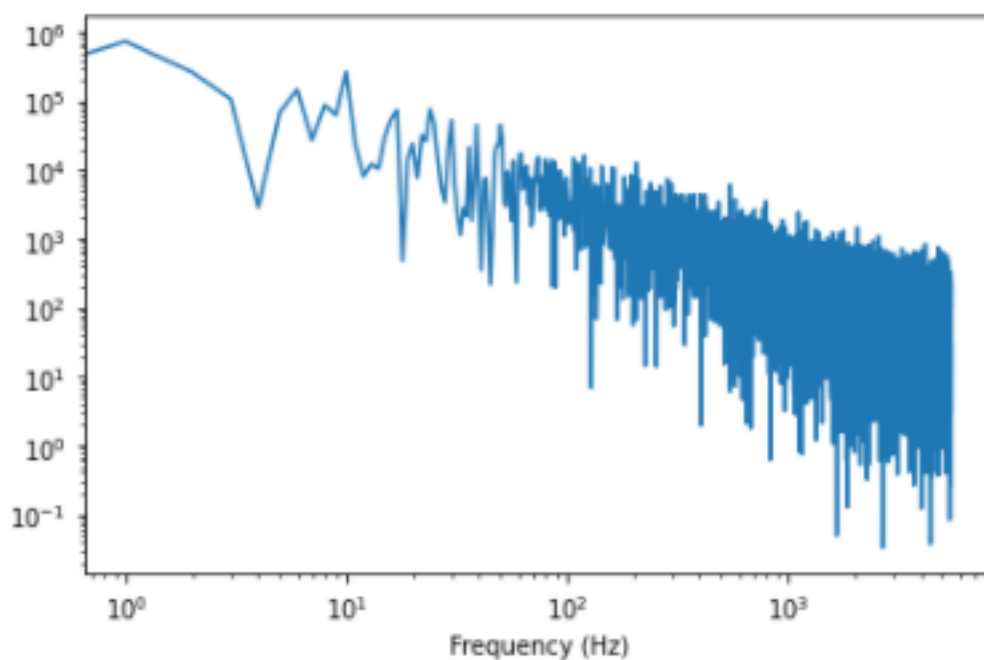


Рис. 29: Полученный спектр мощности

Также посмотрим на наклон полученного спектра:

```
In [36]: spectrum.estimate_slope().slope  
Out[36]: -0.9869492986183374
```

Рис. 30: Наклон спектра

В результате можно сделать вывод, что полученный нами сигнал является «розовым» шумом, т.к. оценка наклона дает величину близкую к -1.

## 6 Выводы

В результате выполнения лабораторной работы мы изучили, что такое шум и какие бывают виды шума, а также научились строить спектры мощности шумов. Нами были преобразованы данные по ценам за **BitCoin** и представлены в виде шума. Кроме того были созданы классы для генерации случайных величин из распределения Пуассона и для реализации алгоритма Восса-МакКартни.