Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №4
Дисциплина: Телекоммуникационные технологии
Тема: Шум

Работу выполнил:
Ляшенко В.В.
Группа: 3530901/80201
Преподаватель:
Богач Н.В.

Санкт-Петербург
2021

Оглавление

Список иллюстраций

Листинги

Глава 1

Упражнение 4.1

Скачаем с сайта `https://freesound.org` несколько примеров шума: шум огня в камине и стрекот сверчков. Затем вычислим их спектры.

1.1 Камин

Начнем с шума огня.

```
from thinkdsp import read_wave

wave = read_wave('sounds/132534__inchadney__fireplace.wav')
segment = wave.segment(start=0, duration=1.0)
segment.make_audio()
```

Листинг 1.1: Выделение сегмента

Получим спектр этого сегмента (Рис.1.1).

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

Листинг 1.2: Получение спектра

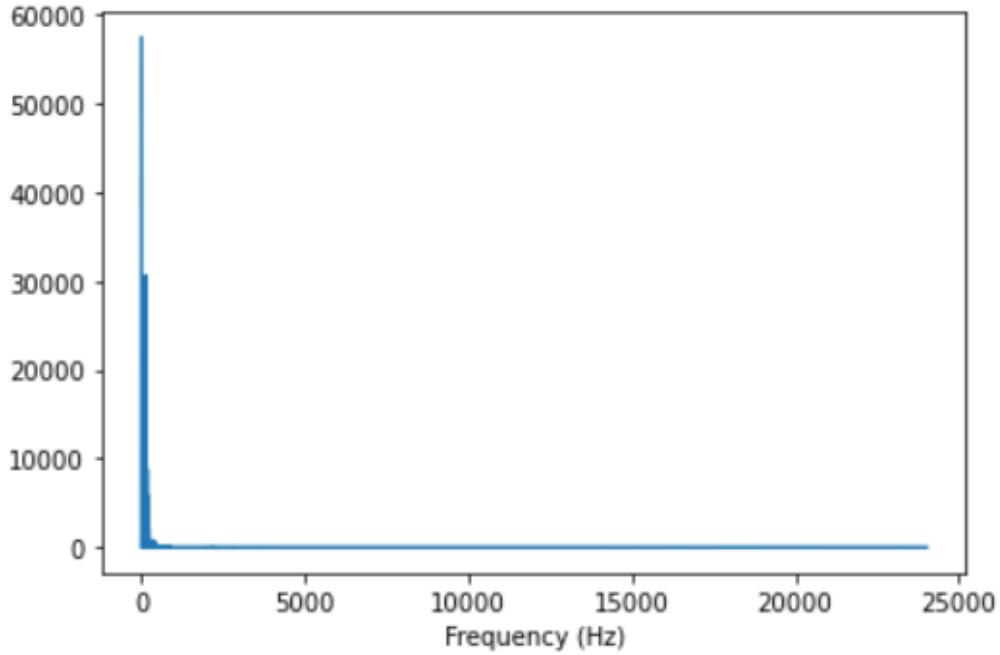

Рис. 1.1: Спектр звука

Амплитуда падает с частотой, поэтому это либо красный шум, либо розовый. Мы можем проверить это, посмотрев на спектр мощности в логарифмической шкале.

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

Листинг 1.3: Получение спектра в логарифмической шкале

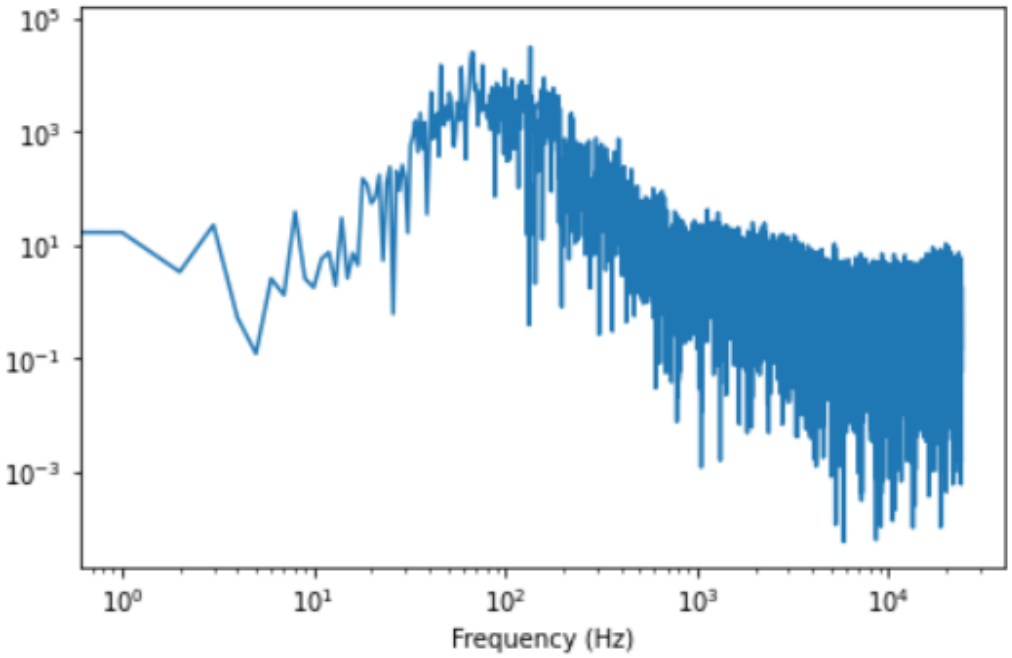

Рис. 1.2: Спектр мощности

На рис.1.2 мы можем видеть, что амплитуда сначала увеличивается, а потом умень-

шается, что является обычным для естественных источников шума.

Теперь построим спектрограмму.

```
segment.make_spectrogram(256).plot(high=2000)
decorate(xlabel='Time(s)', ylabel='Frequency (Hz)')
```

Листинг 1.4: Получение спектрограммы

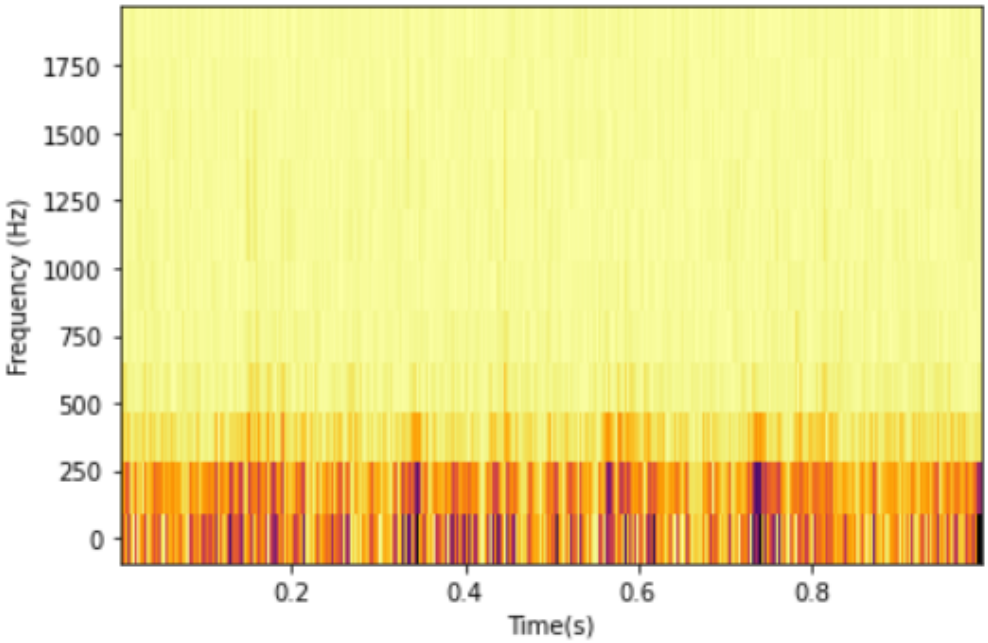

Рис. 1.3: Спектрограмма звука

Из рис.1.3 можно сделать вывод, что это белый шум.

1.2 Сверчки

Теперь проанализируем стрекотание сверчков.

```
from thinkdsp import read_wave

wave = read_wave('sounds/22604__martypinso__dmp010037-crickets-texas.wav')
segment = wave.segment(start=0, duration=1.0)
segment.make_audio()
```

Листинг 1.5: Выделение сегмента

Получим спектр этого сегмента (Рис.1.4).

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

Листинг 1.6: Получение спектра

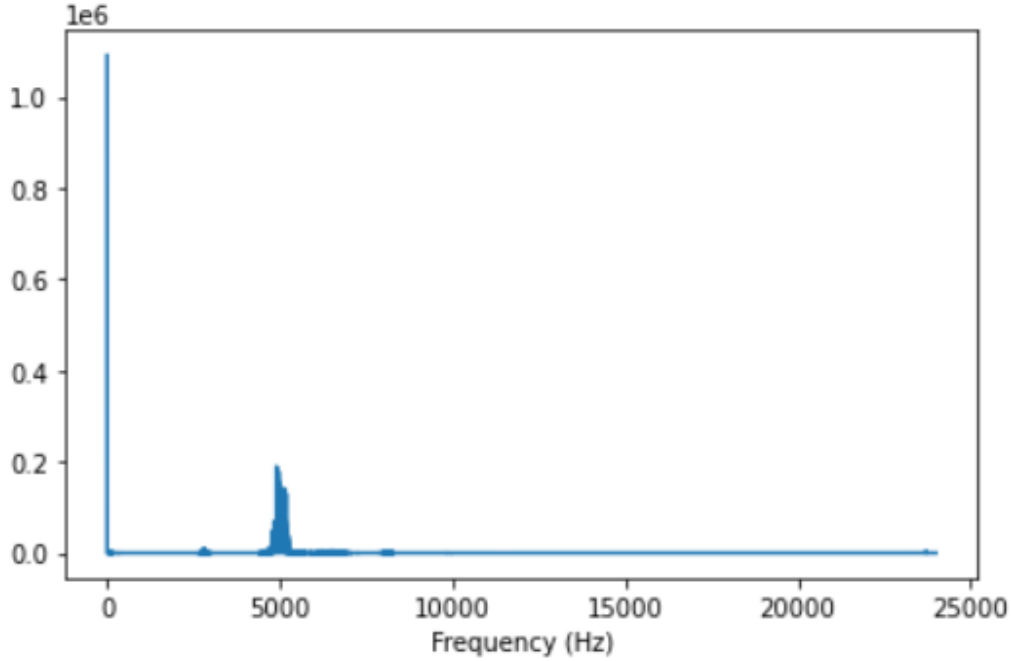

Рис. 1.4: Спектр звука

Посмотрим на спектр мощности в логарифмической шкале.

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

Листинг 1.7: Получение спектра в логарифмической шкале

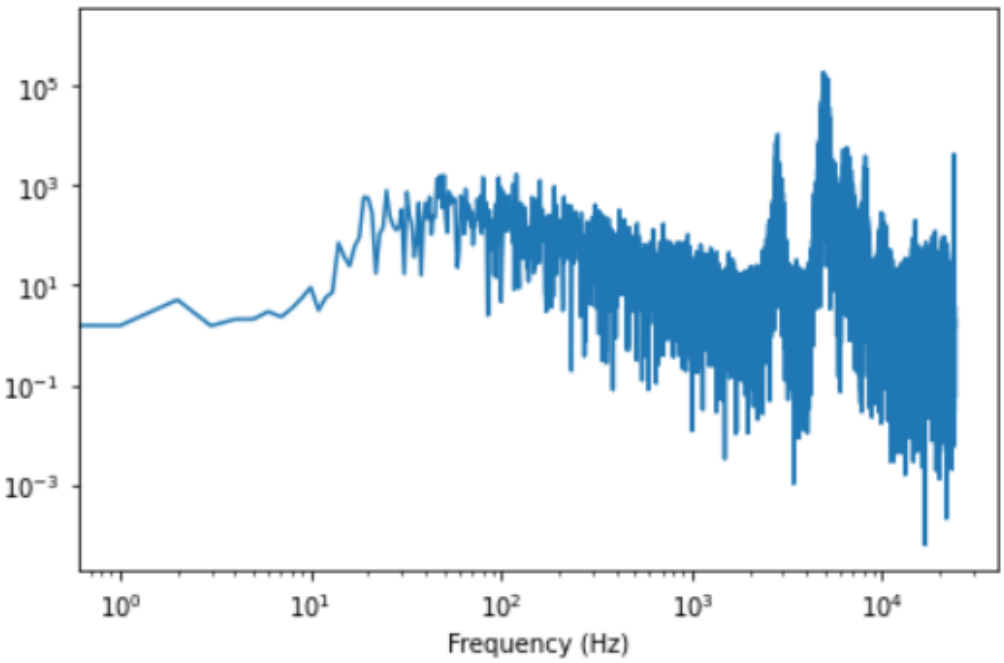

Рис. 1.5: Спектр мощности

Из рис.1.5 мы можем предположить, что это либо розовый шум, либо белый. Теперь построим спектрограмму.

```
segment.make_spectrogram(256).plot(high=2000)
decorate(xlabel='Time(s)', ylabel='Frequency (Hz)')
```

Листинг 1.8: Получение спектрограммы

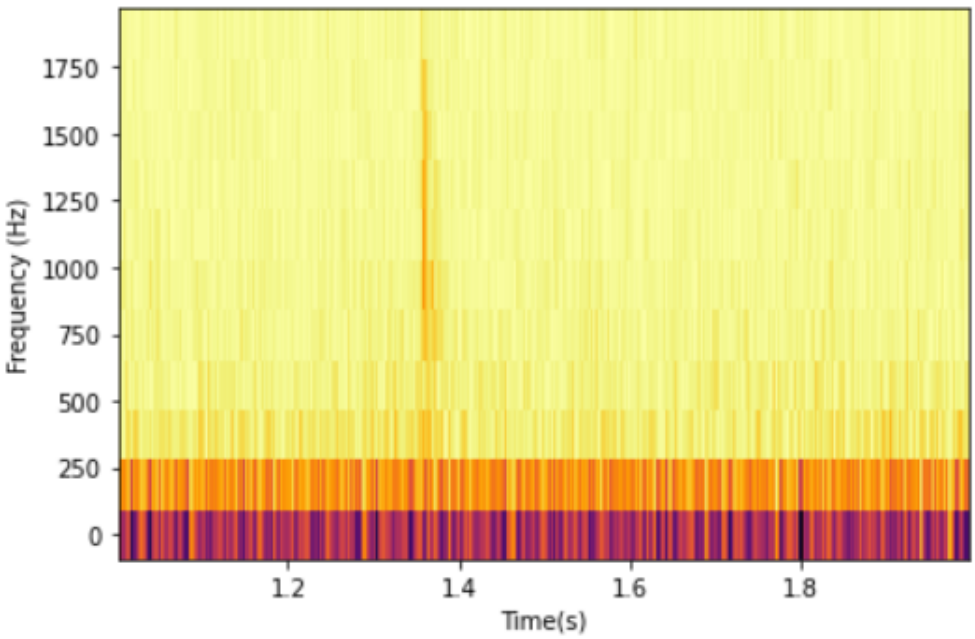

Рис. 1.6: Спектрограмма звука

Из рис.1.6 можно сделать вывод, что это белый шум.

Глава 2

Упражнение 4.2

2.1 Метод Бартлетта

Реализуем метод Бартлетта.

```
from thinkdsp import Spectrum

def bartlett_method(wave, seg_length=512, win_flag=True):
    spectro = wave.make_spectrogram(seg_length, win_flag)
    spectrums = spectro.spec_map.values()

    psds = [spectrum.power for spectrum in spectrums]

    hs = np.sqrt(sum(psds) / len(psds))
    fs = next(iter(spectrums)).fs

    spectrum = Spectrum(hs, fs, wave.framerate)
    return spectrum
```

Листинг 2.1: Метод Бартлетта

2.2 Использование метода

Затем используем его для оценки спектра мощности шумового сигнала. Для этого создадим два сегмента одного сигнала. Пусть это будет шум огня в камине.

```
wave = read_wave('sounds/22604__martypinso__dmp010037-crickets-texas.wav')
segment1 = wave.segment(start=1.0, duration=1.0)
segment2 = wave.segment(start=2.5, duration=1.0)

bart1 = bartlett_method(segment1)
bart2 = bartlett_method(segment2)

bart1.plot_power()
bart2.plot_power()

decorate(xlabel='Frequency (Hz)', ylabel='Power', **loglog)
```

Листинг 2.2: Использование метода

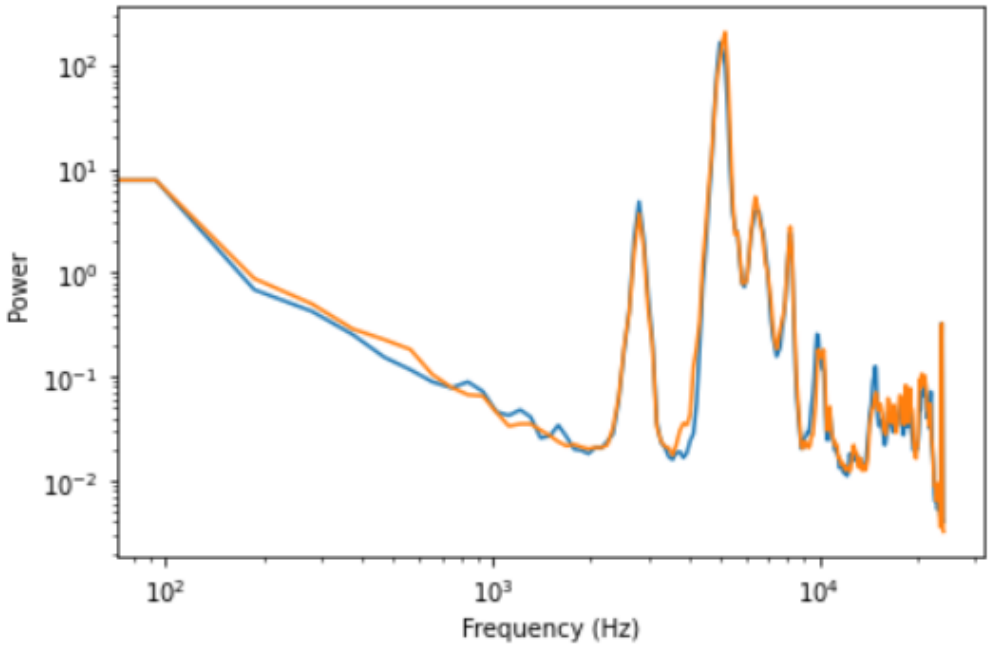

Рис. 2.1: Соотношение мощности и частоты

На рис.2.1 мы можем видеть, что мощность и частота взаимосвязаны. Их взаимосвязь не линейна, а имеет необычную форму. Но для разных сегментов она одинакова.

Глава 3

Упражнение 4.3

Скачаем с сайта `http://www.coindesk.com` исторические данные о ежедневной цене BitCoin. Откроем этот файл и вычислим спектр цен BitCoin как функцию времени. Вначале нам необходимо загрузить данные (Рис.3.1).

```
import pandas as pd

df = pd.read_csv('files/BTC_USD_2020-04-12_2021-04-11-CoinDesk.csv')
df
```

Листинг 3.1: Получение данных о BitCoin

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2020-04-12	6873.848495	6872.137266	6949.788875	6777.889694
1	BTC	2020-04-13	7043.438864	6873.848555	7201.990355	6797.779063
2	BTC	2020-04-14	6889.863772	7043.438893	7050.771025	6600.771494
3	BTC	2020-04-15	6887.554908	6889.863978	6991.326397	6772.322132
4	BTC	2020-04-16	6718.799950	6887.548898	6942.907080	6701.021116
...	...	...	...	...	...	...
360	BTC	2021-04-07	58040.187602	59133.655740	59484.199475	57421.853085
361	BTC	2021-04-08	56508.942864	58030.621849	58645.772971	55541.906134
362	BTC	2021-04-09	57880.905684	55996.080360	58179.656864	55758.491178
363	BTC	2021-04-10	58171.909019	58094.744128	58880.821608	57717.859778
364	BTC	2021-04-11	59295.950044	58149.650591	61065.222625	57924.075264

Рис. 3.1: Данные о BitCoin

Представим данные в виде графика (Рис.3.2).

```
from thinkdsp import Wave

ys = df['Closing Price (USD)']
ts = df.index
wave = Wave(ys, ts, framerate=1)
wave.plot()
decorate(xlabel='Time (days)')
```

Листинг 3.2: Представление данных в виде графика

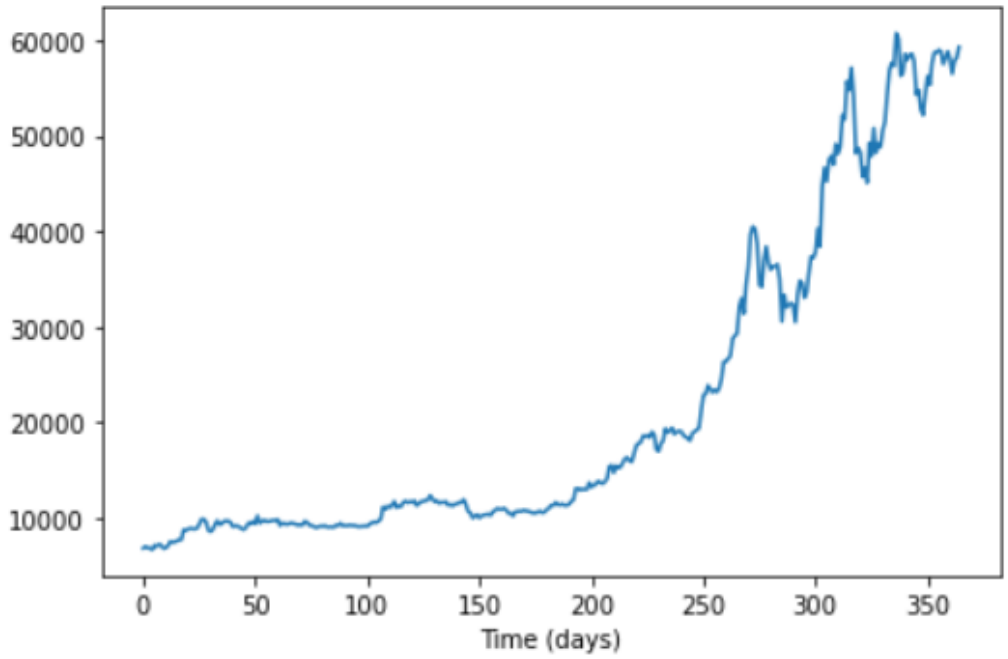

Рис. 3.2: График данных о BitCoin

Получим спектр BitCoin.

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

Листинг 3.3: Вычисление спектра BitCoin

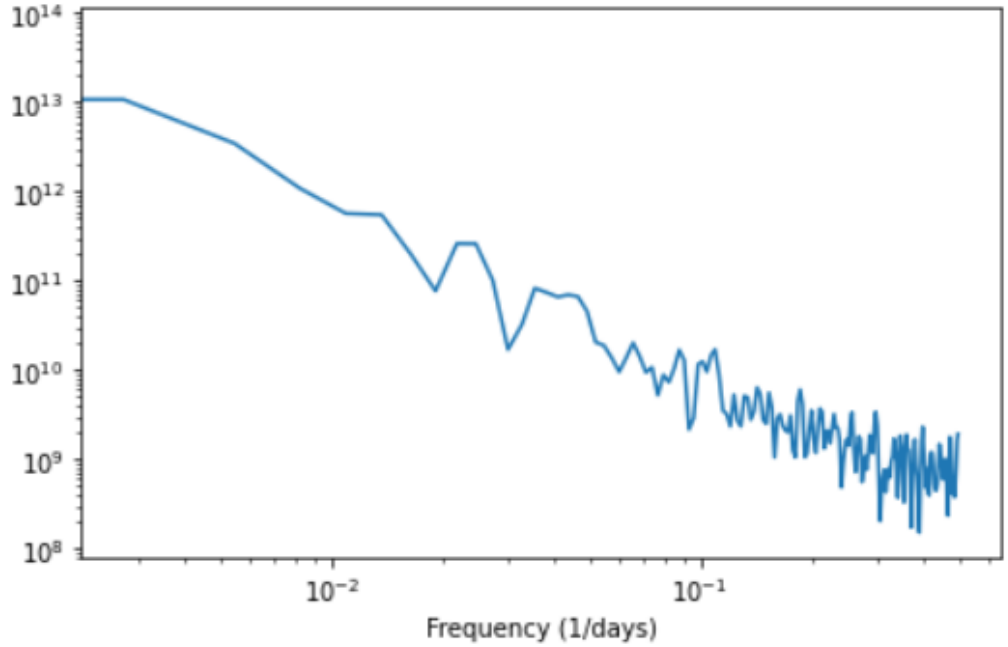

Рис. 3.3: Спектр BitCoin

Из рис.3.3 мы можем предположить, что это либо красный, либо розовый шум. Чтобы определить более точно, вычислим угол наклона графика.

```
spectrum.estimate_slope()[0]
```

Листинг 3.4: Вычисление угла наклона

Красный шум должен иметь наклон -2. Наклон этого графика имеет значение примерно -1,7, поэтому трудно сказать, следует ли считать этот красным шумом или розовым.

Глава 4

Упражнение 4.4

4.1 Класс UncorrelatedPoissonNoise

Напишем класс `UncorrelatedPoissonNoise`, наследующий `thinkdsp._Noise` и предоставляющий `evaluate`.

```
from thinkdsp import Noise

class UncorrelatedPoissonNoise(Noise):

    def evaluate(self, ts):
        ys = np.random.poisson(self.amp, len(ts))
        return ys
```

Листинг 4.1: Класс UncorrelatedPoissonNoise

4.2 Получение звука

Сгенерируем пару секунд UP и послушаем. Сначала зададим маленькое значение `amp`.

```
signal1 = UncorrelatedPoissonNoise(amp=0.0005)
wave1 = signal1.make_wave(duration=1, framerate=10000)
wave1.plot()
wave1.make_audio()
```

Листинг 4.2: Получение звука при малых значениях amp

При малых значениях `amp` звук как у счётчика Гейгера. График сигнала представлен на рис.4.1.

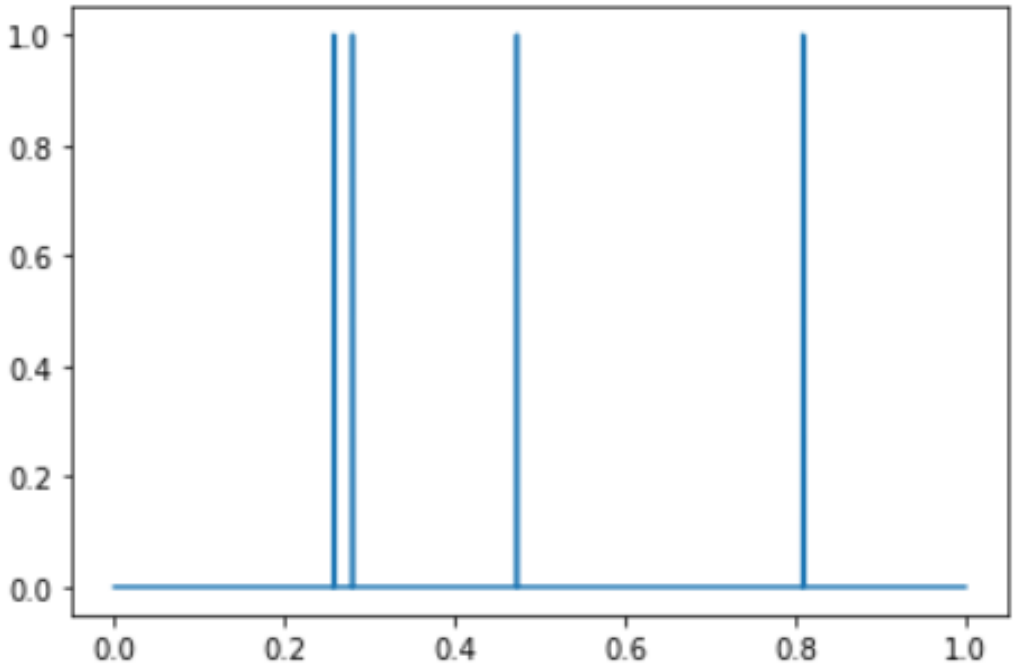

Рис. 4.1: График при малом amp

Вычислим ожидаемое и полученное количество частиц, чтобы убедиться в корректности работы.

```
expected = 0.0005 * 10000 * 1
actual = sum(wave1.ys)
print(expected, actual)
```

Листинг 4.3: Число частиц при малых значениях amp

Ожидаемое число = 5, полученное = 4. Значения близки к друг другу, значит, всё верно.

Теперь зададим большое значение `amp`.

```
signal2 = UncorrelatedPoissonNoise(amp=1)
wave2 = signal2.make_wave(duration=1, framerate=10000)
wave2.plot()
wave2.make_audio()
```

Листинг 4.4: Получение звука при больших значениях amp

При больших значениях звук похож на белый шум. График сигнала представлен на рис.4.2.

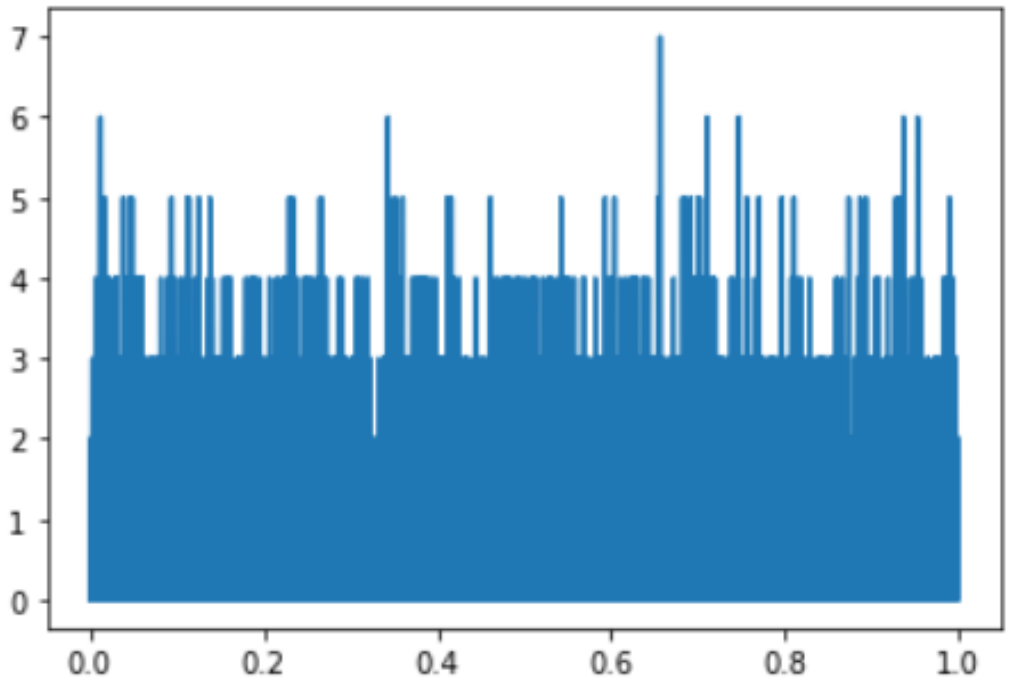

Рис. 4.2: График при большом amp

Так же вычислим ожидаемое и полученное количество частиц, чтобы убедиться в корректности работы.

```
expected = 1 * 10000 * 1
actual = sum(wave1.ys)
print(expected, actual)
```

Листинг 4.5: Число частиц при малых значениях amp

Ожидаемое число = 10000, полученное = 10046. При больших значениях всё тоже в порядке.

4.3 Спектр мощности

Вычислим и напечатаем спектры мощности обоих сигналов.

```
spectrum1 = wave1.make_spectrum()
spectrum2 = wave2.make_spectrum()

spectrum1.plot_power(alpha=0.5)
spectrum2.plot_power(alpha=0.5)
decorate(xlabel='Frequency (Hz)', ylabel='Power', **loglog)
```

Листинг 4.6: Вычисление спектров мощности

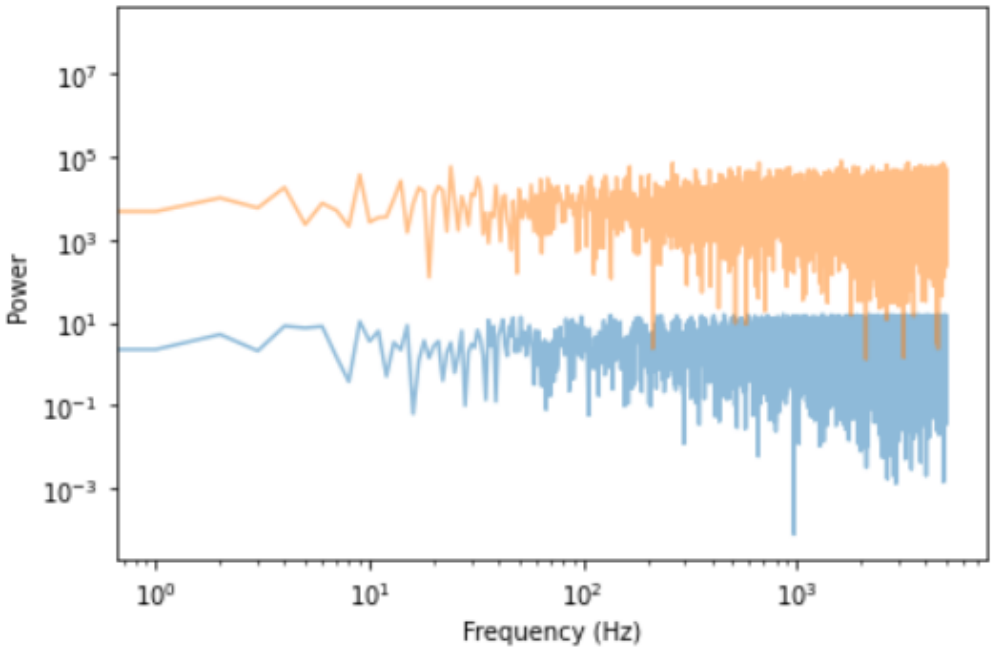

Рис. 4.3: Спектры мощности

Как мы можем видеть на рис.4.3, графики почти идентичны. К тому же, они не имеют наклона, так что их можно отнести к белому шуму.

Глава 5

Упражнение 4.5

5.1 Алгоритм Voss-McCartney

Реализуем алгоритм Voss-McCartney для генерации розового шума и вычислим спектр результата.

```
def voss(nrows, ncols=16):
array = np.empty((nrows, ncols))
array.fill(np.nan)
array[0, :] = np.random.random(ncols)
array[:, 0] = np.random.random(nrows)

n = nrows
cols = np.random.geometric(0.5, n)
cols[cols >= ncols] = 0
rows = np.random.randint(nrows, size=n)
array[rows, cols] = np.random.random(n)

data = pd.DataFrame(array)
data.fillna(method='ffill', axis=0, inplace=True)
total = data.sum(axis=1)

return total.values
```

Листинг 5.1: Алгоритм Voss-McCartney

Чтобы проверить работу метода сгененрируем 20000 значений и построим график (Рис.5.1).

```
from thinkdsp import Wave
wave = Wave(voss(20000))
wave.unbias()
wave.normalize()
wave.plot()
wave.make_audio()
```

Листинг 5.2: Генерация розового шума

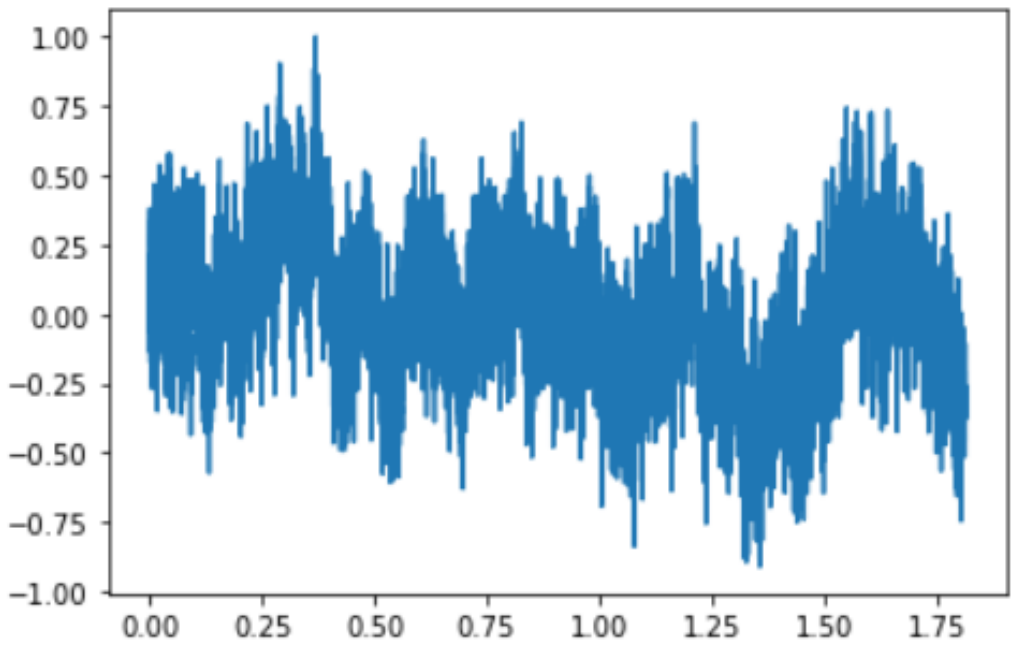

Рис. 5.1: Шум

Послушаем данный звук и убедимся, что он действительно похож на шум. Теперь построим спектр мощности.

```
spectrum = wave.make_spectrum()
spectrum.hs[0] = 0
spectrum.plot_power()
decorate(xlabel='Frequency (Hz)', **loglog)
```

Листинг 5.3: Получение спектра мощности

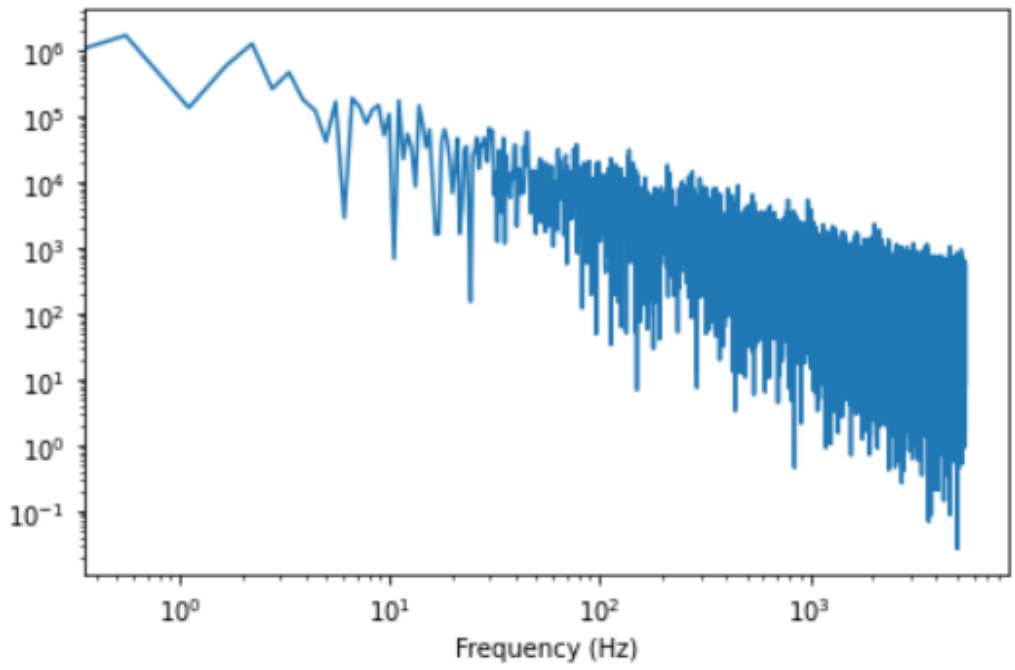

Рис. 5.2: Спектр мощности

Как мы можем видеть на рис.5.2, это действительно похоже на розовый шум. Чтобы в этом точно убедиться, расчитаем наклон с помощью `spectrum.estimate_slope().slope`. Полученное значение можно округлить до -1. Значит, точно получен розовый шум.

5.2 Соотношение между мощностью и частотой

Теперь необходимо убедиться, что соотношение между мощностью и частотой соответствующее. Для этого мы сгенерируем более длинную выборку и используем метод Бартлетта.

```
seg_length = 64 * 2048
wave = Wave(voss(seg_length * 100))
len(wave)
```

Листинг 5.4: Создание выборки

Полученное значение: 13107200.
Применим метод Бартлетта.

```
spectrum = bartlett_method(wave, seg_length=seg_length, win_flag=False)
spectrum.hs[0] = 0
len(spectrum)
```

Листинг 5.5: Использование метода Бартлетта

Полученное значение: 65537.
Построим график соотношения мощности и частоты.

```
spectrum.plot_power()
decorate(xlabel='Frequency (Hz)', ylabel='Power', **loglog)
```

Листинг 5.6: Построение соотношения мощности и частоты

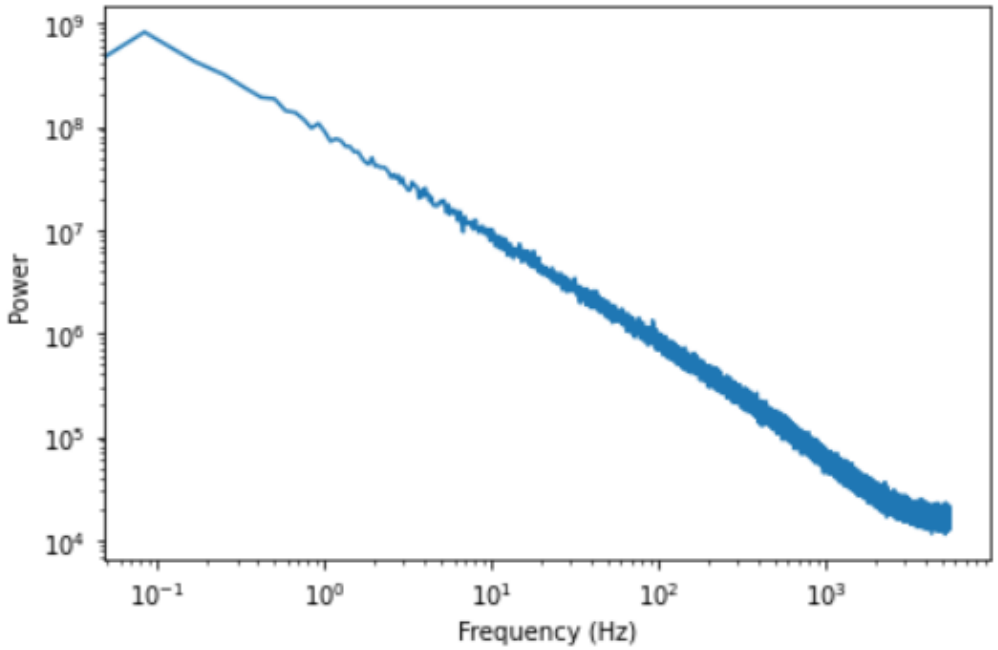

Рис. 5.3: Соотношение мощности и частоты

На рис.5.3 мы можем видеть, что взаимосвязь мощности и частоты похожа на прямую.

Вычислим её наклон с помощью `spectrum.estimate_slope().slope`. При округлении получается -1.

Глава 6

Выводы

В результате выполнения данной работы мы познакомились с различными видами шумов и для работы с ними изучили метод Бартлетта и алгоритм Voss-McCartney.

Метод Бартлетта используется для оценки спектров мощности, а алгоритм Voss-McCartney позволяет генерировать розовый шум.