Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

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

Выполнил студент гр. 3530901/80201 В.А. Пучкина

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение 4.1

Это упражнение направлено на знакомство с шумом. А потому необходимо скачать записи некоторых природных источников шума и вычислить спектры каждого из них.

1.1 Шум моря

Для начала была скачена эта запись шума морских волн. Теперь необходимо считать скаченный файл и получить `wave`. Тут же прослушаем запись.

```
sea_wave = read_wave('resources/Sounds/task1_north_sea.wav')
sea_wave.make_audio()
```

Листинг 1: Чтение скаченного файла.

Теперь выберем походящий сегмент и получим его линейный спектр.

```
sea_segment = sea_wave.segment(start=3, duration=1)
sea_spectrum = sea_segment.make_spectrum()
sea_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)')
sea_segment.make_audio()
```

Листинг 2: Выбор сегмента и получение его линейного спектра.

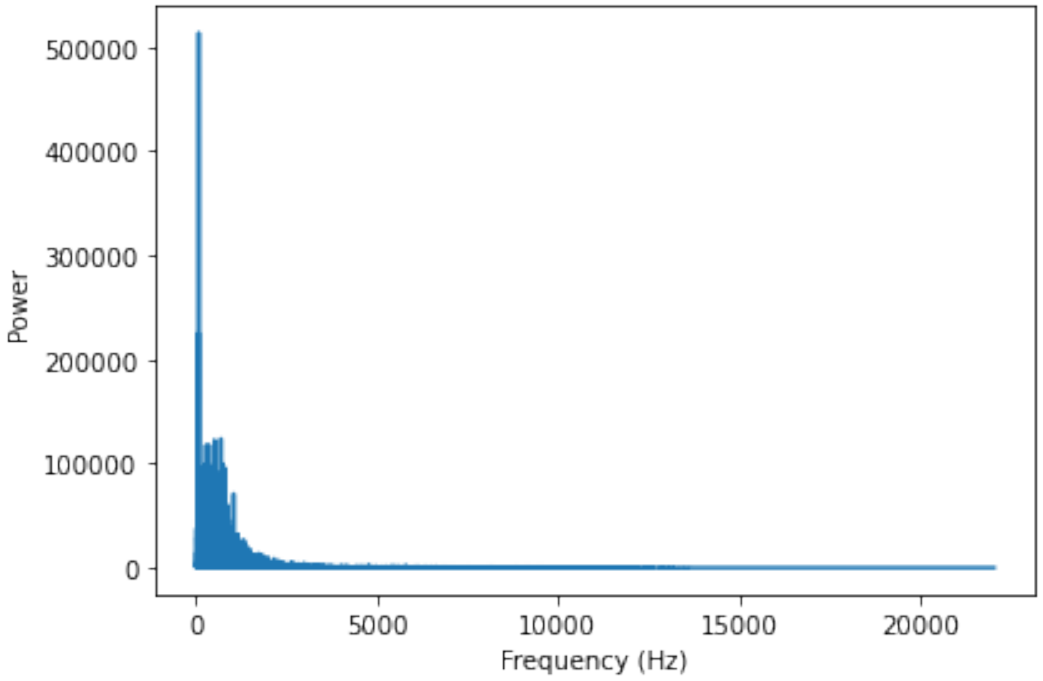

Рис. 1: Линейный спектр шума моря.

Получим логарифмический спектр этого сегмента.

```
sea_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 3: Получение логарифмического спектра.

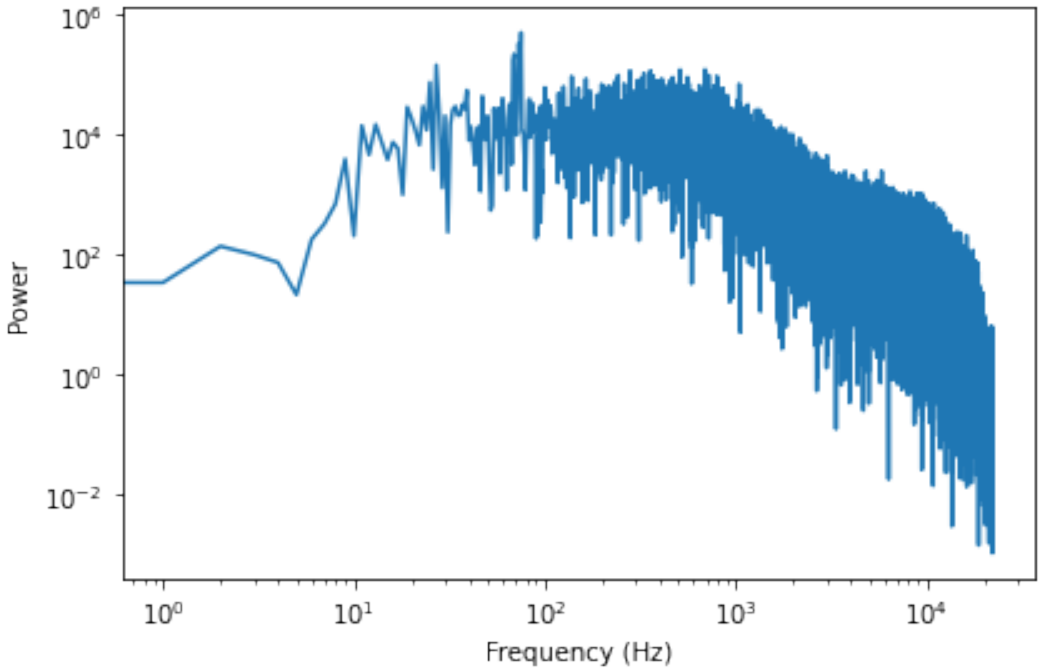

Рис. 2: Логарифмический спектр шума моря.

По полученным спектрам можно отнести этот шум к розовому или красному шуму (не к белому, потому что у нашего сигнала с увеличением частоты падает мощность). Однако отнести этот шум к какому-либо конкретному типу сложно. Также по логарифмическому спектру (Рис.2) видно, что мощность сначала увеличивается, а потом уменьшается. Это обычное поведение для естественных источников шума.

Также получим спектрограмму этого сегмента.

```
sea_segment.make_spectrogram(256).plot(5000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 4: Получение спектрограммы сегмента.

Теперь давайте посмотрим, как спектр изменяется во времени. Для этого выберем другой сегмент и сравним спектры второго сегмента с первым.

```
sea_segment2 = sea_wave.segment(start=5, duration=1)
sea_spectrum2 = sea_segment2.make_spectrum()
sea_spectrum2.plot_power(color='red', alpha=0.5)
sea_spectrum.plot_power(color='blue', alpha=0.5)
decorate(ylabel='Power', xlabel='Frequency (Hz)')
sea_segment2.make_audio()
```

Листинг 5: Сравнение линейных спектров двух сегментов.

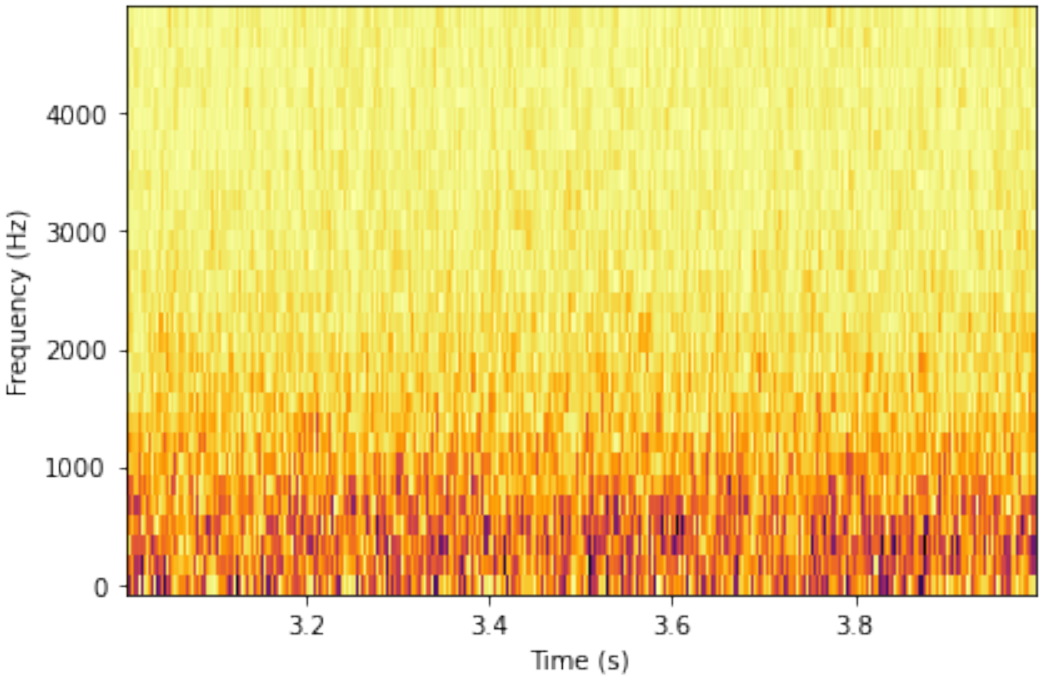

Рис. 3: Спектрограмма шума моря.

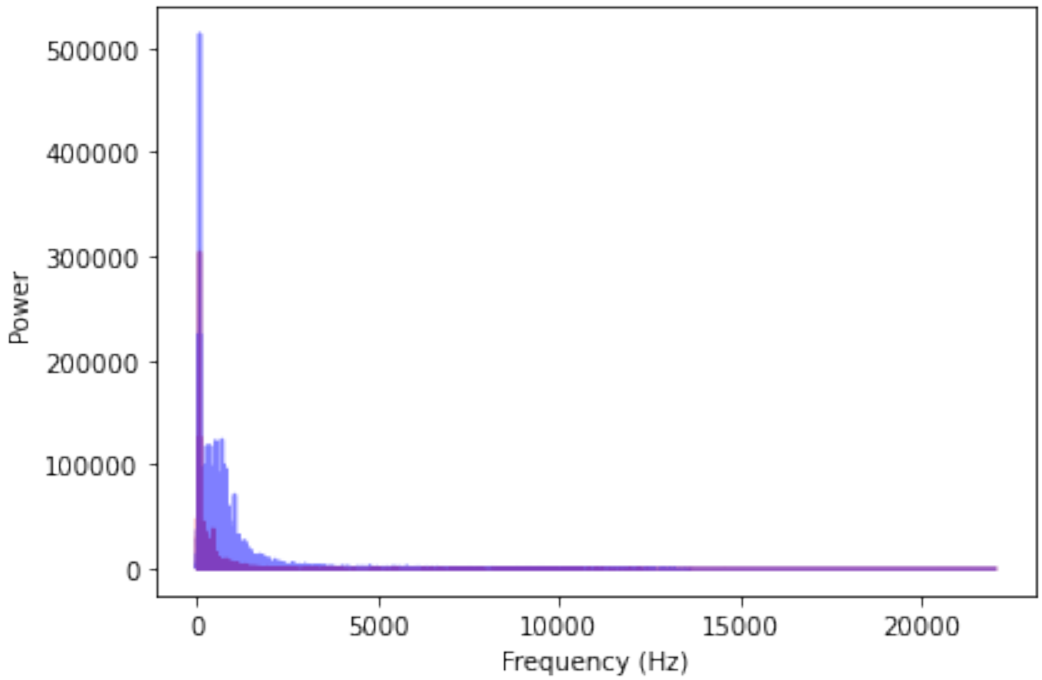

Рис. 4: Сравнение линейных спектров первого (синий) и второго (красный) сегментов.

И сравним логарифмические спектры.

```
sea_spectrum.plot_power(color='blue', alpha=0.5)
sea_spectrum2.plot_power(color='red', alpha=0.5)
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 6: Сравнение логарифмических спектров двух сегментов.

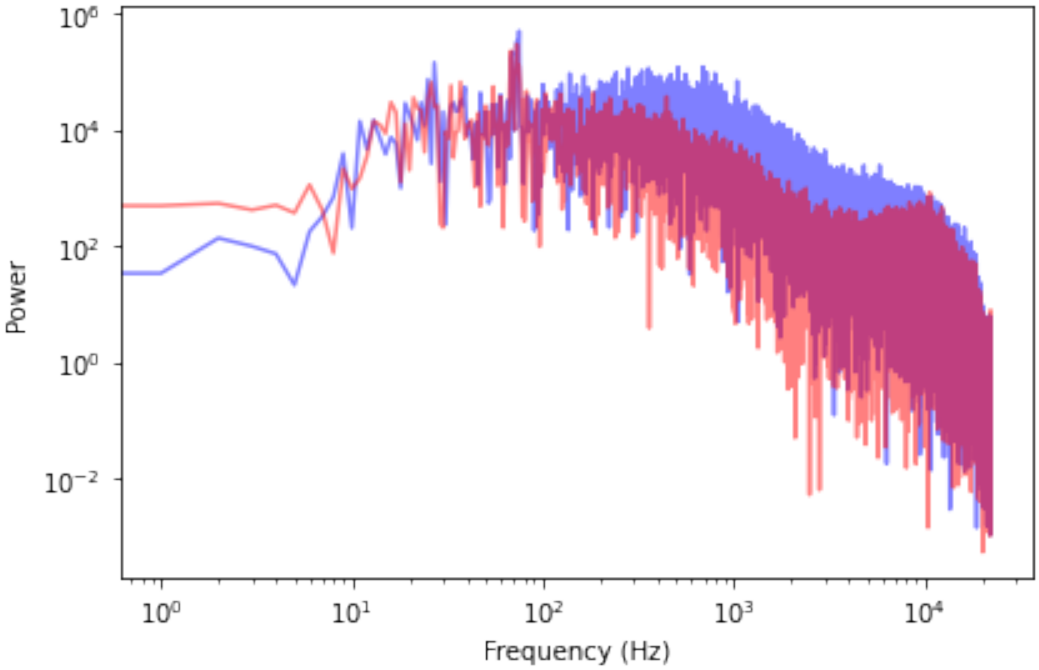

Рис. 5: Сравнение логарифмических спектров первого (синий) и второго (красный) сегментов.

Из полученных спектров (Рис.4 и Рис.5) можно сделать вывод, что спектр шума моря с течением времени не изменяется.

1.2 Шум костра

Теперь изучим запись шума костра, скаченную отсюда. Считаем файл, выберем походящий сегмент и получим его линейный спектр.

```
fire_wave = read_wave('resources/Sounds/task1_fireplace.wav')
fire_segment = fire_wave.segment(start=12, duration=1)
fire_spectrum = fire_segment.make_spectrum()
fire_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)')
fire_segment.make_audio()
```

Листинг 7: Чтение файла, выбор сегмента и получение линейного спектра.

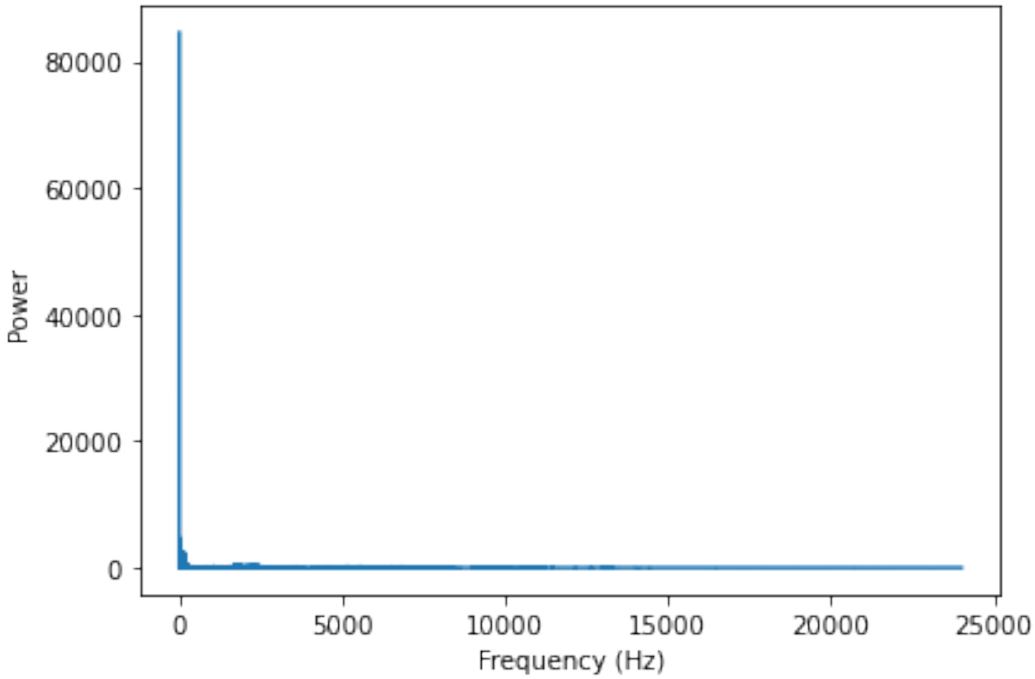

Рис. 6: Линейный спектр шума костра.

Получим логарифмический спектр выбранного сегмента.

```
fire_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 8: Получение логарифмического спектра.

По полученным спектрам шум костра так же, как и шум моря, можно отнести к розовому или красному шуму. Получим спектрограмму нашего сегмента.

```
fire_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 9: Получение спектрограммы сегмента.

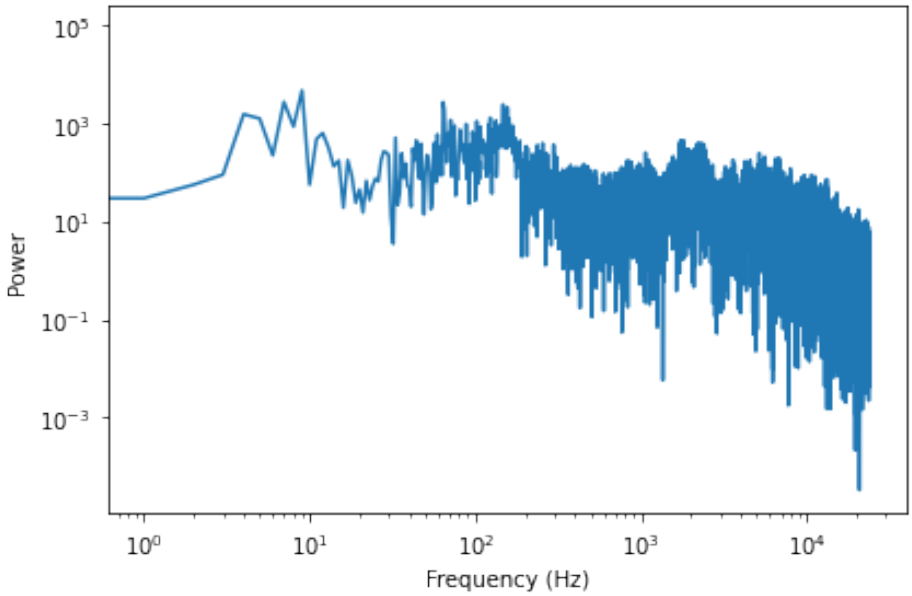

Рис. 7: Логарифмический спектр шума костра.

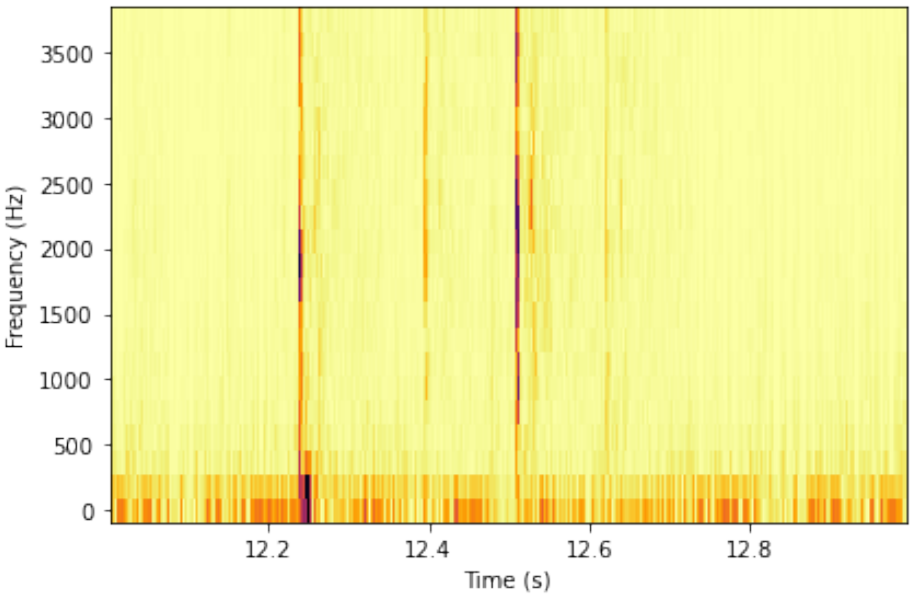

Рис. 8: Спектрограмма шума костра.

Теперь посмотрим, как спектр изменяется во времени. Для этого выберем другой сегмент и сравним линейные спектры второго сегмента с первым.

```
fire_segment2 = fire_wave.segment(start=27, duration=1)
fire_spectrum2 = fire_segment2.make_spectrum()
fire_spectrum2.plot_power(color='red', alpha=0.5)
fire_spectrum.plot_power(color='blue', alpha=0.5)
decorate(ylabel='Power', xlabel='Frequency (Hz)')
fire_segment2.make_audio()

```

Листинг 10: Сравнение линейных спектров двух сегментов.

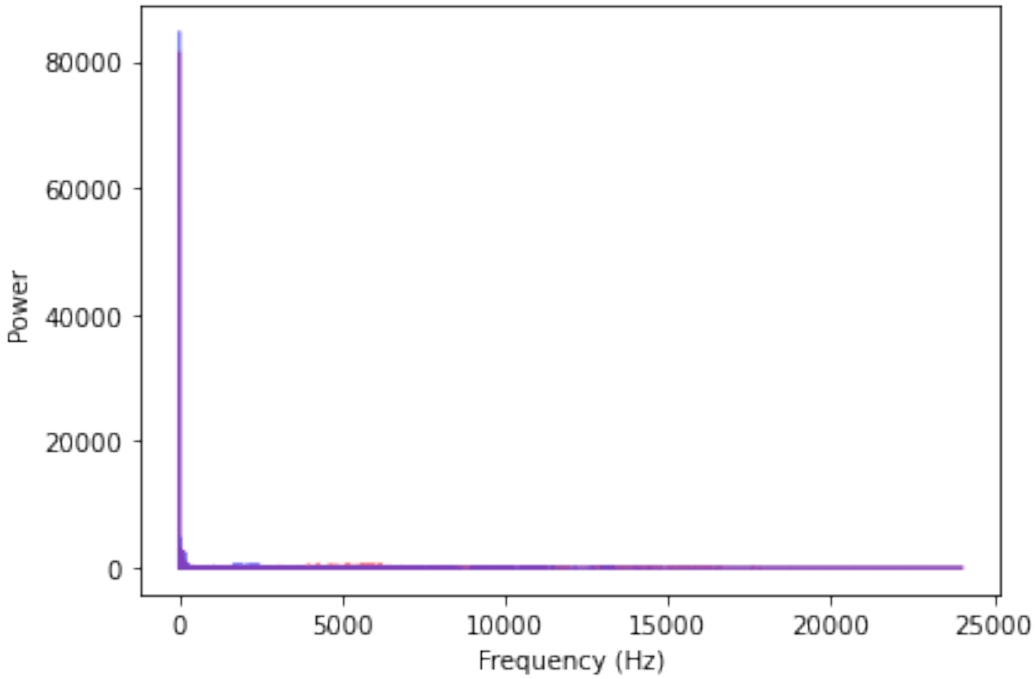

Рис. 9: Сравнение линейных спектров первого (синий) и второго (красный) сегментов.

Сравним также логарифмические спектры.

```
fire_spectrum.plot_power(color='blue', alpha=0.5)
fire_spectrum2.plot_power(color='red', alpha=0.5)
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 11: Сравнение логарифмических спектров двух сегментов.

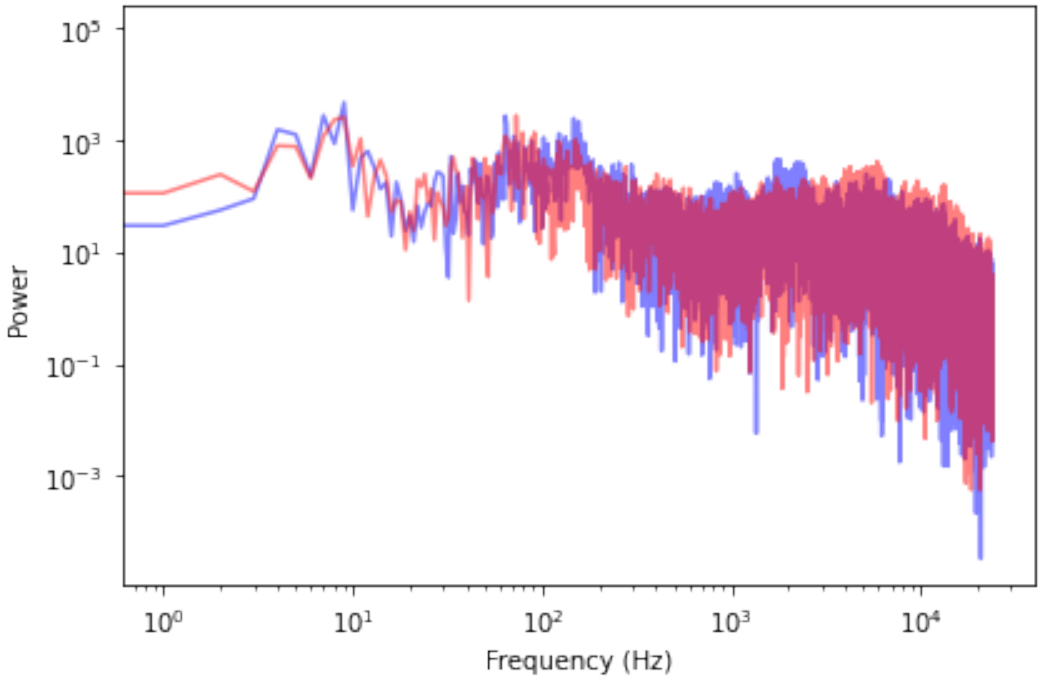

Рис. 10: Сравнение логарифмических спектров первого (синий) и второго (красный) сегментов.

Из полученных спектров (Рис.9 и Рис.10) можно сделать вывод, что спектр шума костра с течением времени не изменяется.

В ходе выполнения данного упражнения были проанализированы линейные и логарифмические спектры шума моря и костра. Также были рассмотрены их спектрограммы и проанализировано изменение их спектров в течение времени. Был сделан вывод, что спектры этих шумов с течением времени не изменяются.

2 Упражнение 4.2

В данном упражнении необходимо реализовать метод Бартлетта и использовать его для оценки спектра мощности шумового сигнала.

Итак, начнём с реализации метода Бартлетта.

```
def bartlett (wave, seg_length=512, win_flag=True):
    spectrogram = wave.make_spectrogram(seg_length, win_flag)
    spectrums = spectrogram.spec_map.values()
    psds = [spectrum.power for spectrum in spectrums]
    hs = numpy.sqrt(sum(psds) / len(psds))
    fs = next(iter(spectrums)).fs
    spectrum = Spectrum(hs, fs, wave.framerate)
    return spectrum
```

Листинг 12: Метод Бартлетта.

Теперь используем наш метод для оценки спектра мощности шумового сигнала. В качестве шумового сигнала возьмём шум моря из § 1.1.

```
bartlett(sea_segment).plot_power(color='blue')
bartlett(sea_segment2).plot_power(color='red')
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 13: Оценка спектра мощности двух сегментов.

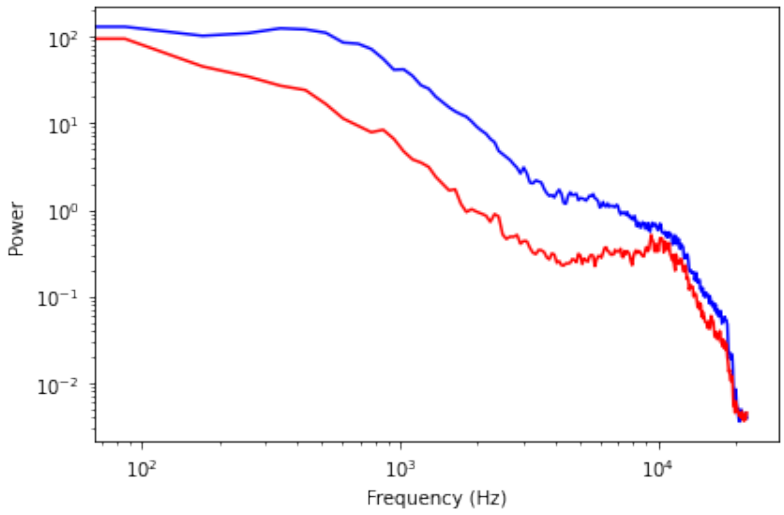

Рис. 11: Сравнение спектров мощности двух сегментов.

Из Рис.11 видно, что мощность и частота связаны. Конечно, они имеют не линейную, а более сложную связь, однако для разных сегментов эта связь одинакова.

В ходе данного упражнения был реализован метод Бартлетта, после чего с его помощью была проведена оценка спектра мощности шума моря. Был сделан вывод, что мощность и частота имеют сложную связь, однако для разных сегментов эта связь одинакова.

3 Упражнение 4.3

В этом упражнении необходимо провести следующий эксперимент: скачать с этого сайта исторические данные о ежедневной цене BitCoin в виде CSV-файла, затем вычислить спектр цен BitCoin и оценить, на какой вид шума похож полученный спектр.

Итак, скачаем курс BTC/USD за всё время с указанного выше сайта, считаем и преобразуем его в `wave`.

```
import pandas

df = pandas.read_csv('resources/task3_BTC_USD_2013-10-01_2021-05-07.csv',
    parse_dates=[0])
ys = df['Closing Price (USD)']
ts = df.index

wave_btc = Wave(ys, ts, framerate=1)
wave_btc.plot()
decorate(xlabel='Time (days)')
```

Листинг 14: Перобразование файла с курсом BitCoin в `wave`.

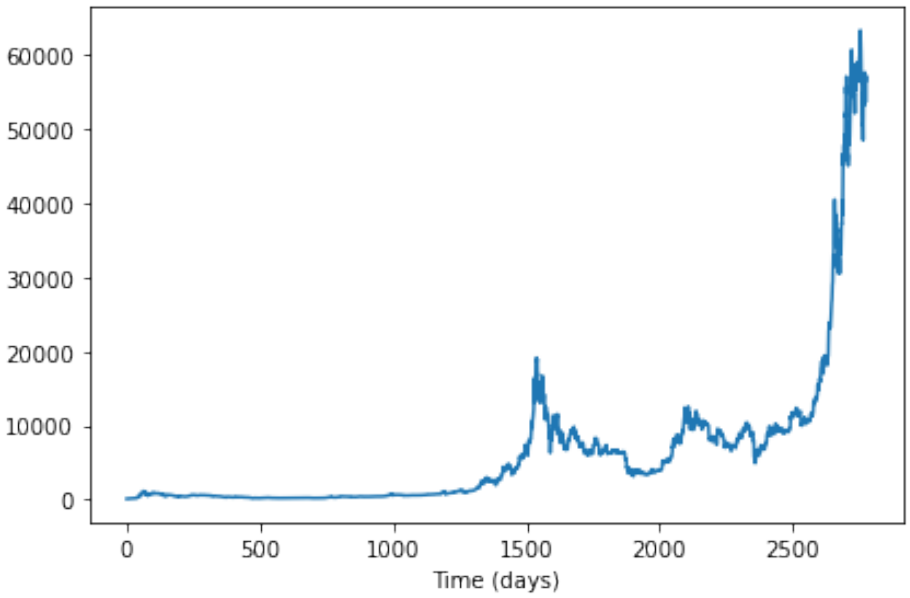

Рис. 12: Курс BitCoin.

Теперь получим спектр курса BitCoin и его значение его наклона.

```
spectrum_btc = wave_btc.make_spectrum()
spectrum_btc.plot_power()
decorate(xlabel='Frequency (1/days)', xscale='log', yscale='log')
spectrum_btc.estimate_slope()[0]
```

Листинг 15: Получение спектра и наклона курса BtCoin.

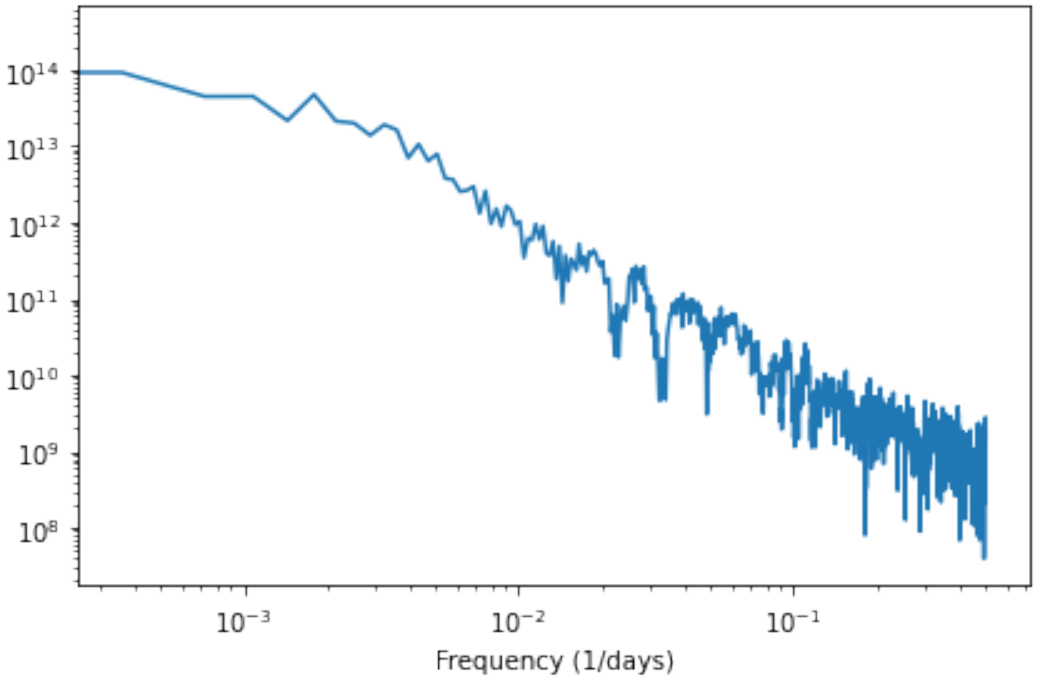

Рис. 13: Спектр курса BitCoin.

Полученный спектр (Рис.13) напоминает спектры розового или красного шумов. Кроме того, наклон получился равным -1.803467606343994, что можно округлить до -1,8. Красный шум имеет наклон -2, а потому курс BitCoin это либо красный, либо розовый шум.

В ходе данного упражнения был проведён интересный эксперимент с BitCoin. Был скачан курс BTC/USD за всё время, после чего получен его `wave` и спектр. По полученным результатам был сделан вывод, что курс BTC/USD сложно отнести к конкретному виду шума, так как он похож и на красный, и на розовый шумы.

4 Упражнение 4.4

В этом упражнении необходимо написать класс `UncorrelatedPoissonNoise`, наследующий `thinkdsp._Noise` и имеющий `evaluate`. Для генерации случайных величин из распределения Пуассона следует использовать `Np.random.poisson`. Затем с помощью этого класса необходимо сгенерировать и прослушать сигнал в несколько секунд, после чего вычислить его спектр мощности.

Итак, начнём с написания класса `UncorrelatedPoissonNoise`.

```
class UncorrelatedPoissonNoise(Noise):
    def evaluate(self, ts):
         ys = numpy.random.poisson(self.amp, len(ts))
         return ys
```

Листинг 16: Класс `UncorrelatedPoissonNoise`.

4.1 Сигнал с amp = 0.001

Теперь с помощью этого класса сгенерируем сигнал (`amp = 0.001`), получим его `wave` и прослушаем его.

```
poisson_signal = UncorrelatedPoissonNoise(amp=0.001)
poisson_wave = poisson_signal.make_wave(duration=1, framerate=10000)
poisson_wave.plot()
poisson_wave.make_audio()
```

Листинг 17: Генерация сигнала с `amp = 0.001`.

Полученный звук действительно напоминает «щелчки» счётчика Гейгера. Вычислим ожидаемое и реальное количество «щелчков».

```
expected = 0.001 * 10000 * 1
actual = sum(poisson_wave.ys)
print('expected = ', expected, '\nactual = ', actual)
```

Листинг 18: Проверка количества частиц (`amp = 0.001`).

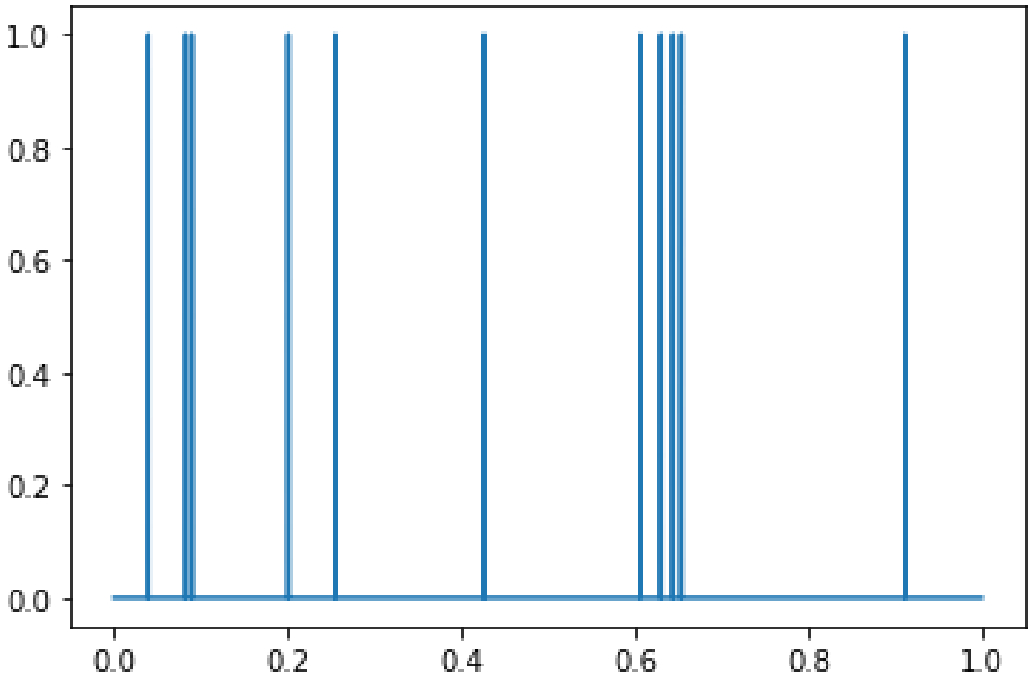

Рис. 14: `Wave` сгенерированного сигнала (`amp = 0.001`).

Был получен следующий результат: expected = 10.0, actual = 11. Эти значения близки, а потому всё верно. Также можно заметить, что количество фактических «щелчков» совпадает с количеством пиков на Рис.14.

Теперь получим логарифмический спектр сигнала.

```
poisson_spectrum = poisson_wave.make_spectrum()
poisson_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 19: Вычисление логарифмического спектра сигнала (`amp = 0.001`).

Как можно видеть из Рис.15, спектр сгенерированного сигнала напоминает спектр белого шума. Чтобы в этом убедиться, оценим его наклон.

```
poisson_spectrum.estimate_slope().slope
```

Листинг 20: Вычисление наклона (`amp = 0.001`).

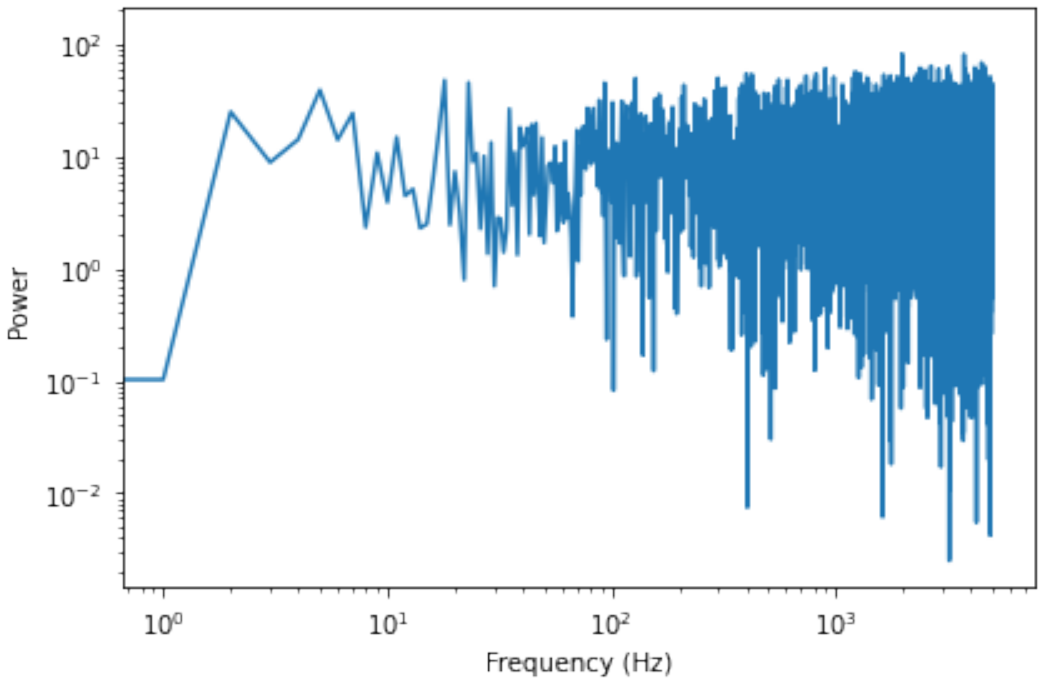

Рис. 15: Логарифмический спектр сгенерированного сигнала (`amp = 0.001`).

Было получено значение 0.0064347261378367 79, что близко к 0. А это как раз характерно для белого шума.

4.2 Сигнал с amp = 1

Теперь сгенерируем сигнал с большим значением amp. И проделаем для него то же самое.

```
poisson_signal = UncorrelatedPoissonNoise(amp=1)
poisson_wave = poisson_signal.make_wave(duration=1, framerate=10000)
poisson_wave.plot()
poisson_wave.make_audio()
```

Листинг 21: Генерация сигнала с amp = 1.

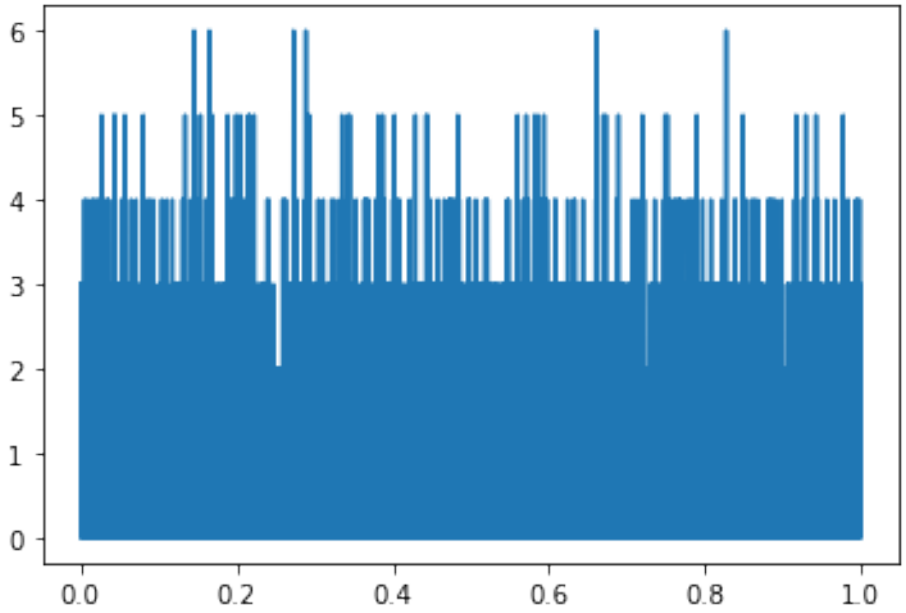

Рис. 16: Wave сгенерированного сигнала (amp = 1).

Этот звук уже напоминает белый шум. Вычислим ожидаемое и реальное количество «щелчков».

```
expected = 1 * 10000 * 1
actual = sum(poisson_wave.ys)
print('expected = ', expected, '\nactual = ', actual)
```

Листинг 22: Проверка количества частиц (amp = 1).

Был получен следующий результат: expected = 10000, actual = 10069. Эти значения близки, а потому всё верно.

Получим логарифмический спектр сигнала.

```
poisson_spectrum = poisson_wave.make_spectrum()
poisson_spectrum.plot_power()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
```

Листинг 23: Вычисление логарифмического спектра сигнала (amp = 1).

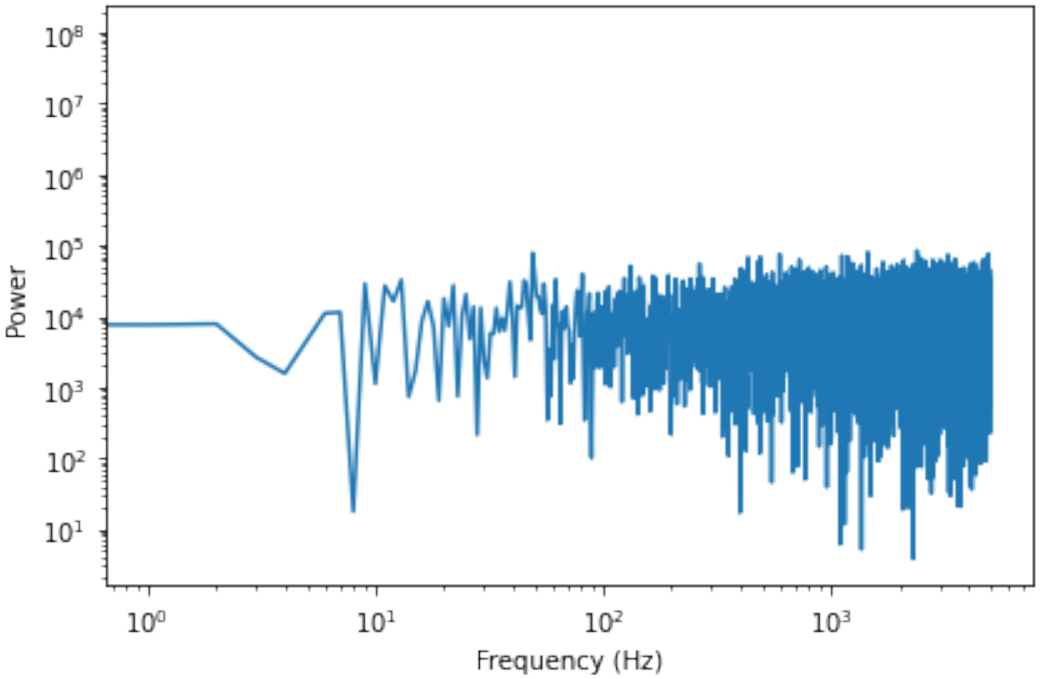

Рис. 17: Логарифмический спектр сигнала (`amp = 1`).

Как можно видеть из Рис.17, спектр этого сигнала тоже напоминает спектр белого шума. Оценим его наклон.

```
poisson_spectrum.estimate_slope().slope
```

Листинг 24: Вычисление наклона (`amp = 1`).

Было получено значение 0.019870408290482002, что близко 0. А это, как уже говорилось ранее, характерно для белого шума.

В ходе выполнения данного упражнения сначала был написан требуемый класс `UncorrelatedPoissonNoise`, генерирующий некоррелированный шум Пуассона. С помощью этого класса были созданы два сигнала с разной амплитудой, были получены их `wave` и спектры, а также оценён наклон. После этих действий был сделан вывод, что полученные сигналы можно отнести к белому шуму.

5 Упражнение 4.5

В этом упражнении необходимо реализовать алгоритм Voss-McCartney для генерации розового шума. Затем с помощью этого алгоритма нужно сгенерировать сигнал и убедиться, что соотношение между его мощностью и частотой соответствуют розовому шуму.

Основная идея алгоритма Voss-McCartney заключается в сложении нескольких последовательностей случайных чисел, которые обновляются с разной частотой. Первый источник обновляется на каждом шаге, второй - каждый второй шаг, третий - каждый третий и так далее.

Напишем метод voss_mccartney, реализующий алгоритм Voss-McCartney.

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

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

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

    return total.values
```

Листинг 25: Метод voss_mccartney.

Теперь проверим корректную работу метода. Для этого сгенерируем 15000 значений с помощью этого метода и, используя их в качестве значений сигнала, преобразуем их в wave.

```
ys = voss_mccartney(15000)
voss_wave = Wave(ys)
voss_wave.unbias()
voss_wave.normalize()
voss_wave.plot()
voss_wave.make_audio()
```

Листинг 26: Генерация значений и их преобразование в wave.

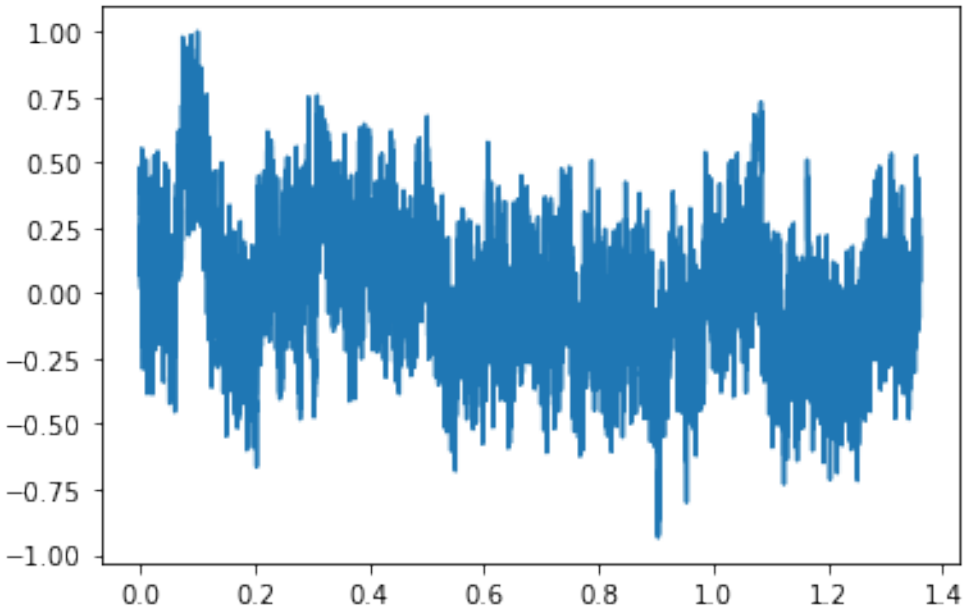

Рис. 18: `Wave`, полученный из сгенерированных значений.

Звук же сгенерированного сигнала действительно похож на шум. Теперь получим спектр мощности и заодно вычислим наклон.

```
voss_spectrum = voss_wave.make_spectrum()
voss_spectrum.hs[0] = 0
voss_spectrum.plot()
decorate(ylabel='Power', xlabel='Frequency (Hz)', xscale='log', yscale='log')
voss_spectrum.estimate_slope().slope
```

Листинг 27: Вычисление логарифмического спектра.

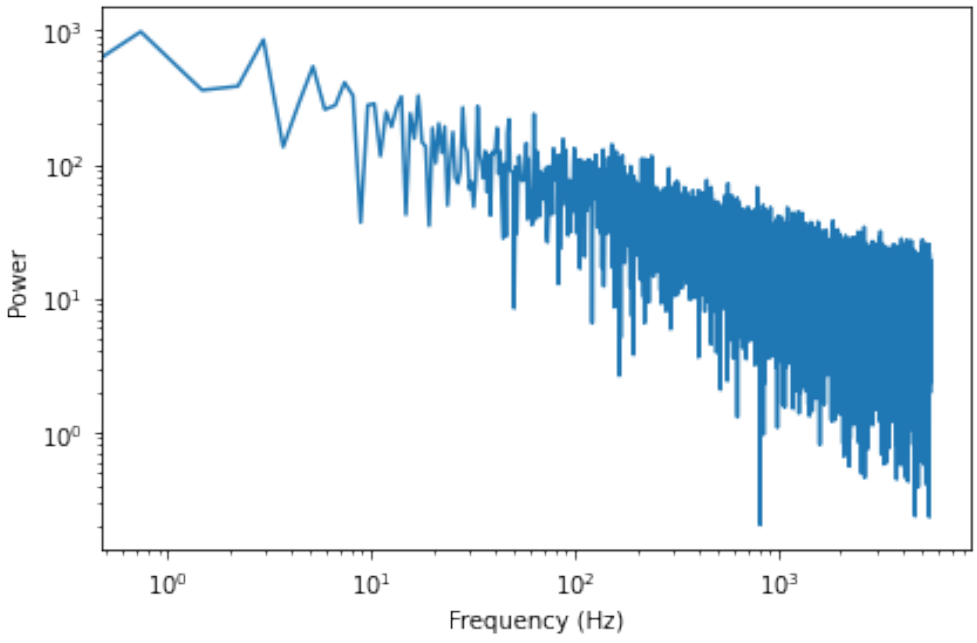

Рис. 19: `Wave`, полученный из сгенерированных значений.

Полученный спектр (Рис.19) действительно похож на спектр розового шума. Значение наклона же получилось -1.0064492830615028, что можно округлить до -1. А это как раз соответствует розовому шуму.

В ходе данного упражнения был написан метод `voss_mccartney`, генерирующий заданное количество значений по алгоритму `Voss-McCartney`. Затем эти значения были преобразованы в `wave`, у которого был получен спектр мощности. Также был рассчитан наклон. В соответствии с полученными результатами был сделан вывод, что написанный метод работает корректно и подходит для генерации розового шума.

6 Выводы

В ходе выполнения данного лабораторной работы были изучены различные виды шумов: белый, красный и розовый. Кроме того, были изучены линейные и логарифмические спектры мощности. Также была проведена работа с шумами, имеющими природный источник. Был написан метод Бартлетта для оценки спектра мощности шумового сигнала, класс `UncorrelatedPoissonNoise` для генерации некоррелированного шума Пуассона и реализован алгоритм `Voss-McCartney` для генерации розового шума.