Лабораторная работа № 4. Шум.

3530901/80201, Шелаев Н. Р.

30 мая 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Некоррелированный шум

Генерируем некоррелированнный равномерный шум.

```
        from thinkdsp import UncorrelatedUniformNoise

        np.random.seed(17)
        signal = UncorrelatedUniformNoise()
        wave = signal.make_wave(duration = 1, framerate =
    11025)
        segment = wave.segment(duration = 0.1)
        segment.plot()
        spectrum = wave.make_spectrum()
        spectrum.plot_power(linewidth = 0.5)

```

Листинг 1.1: Строим и исследуем некоррелированный шум

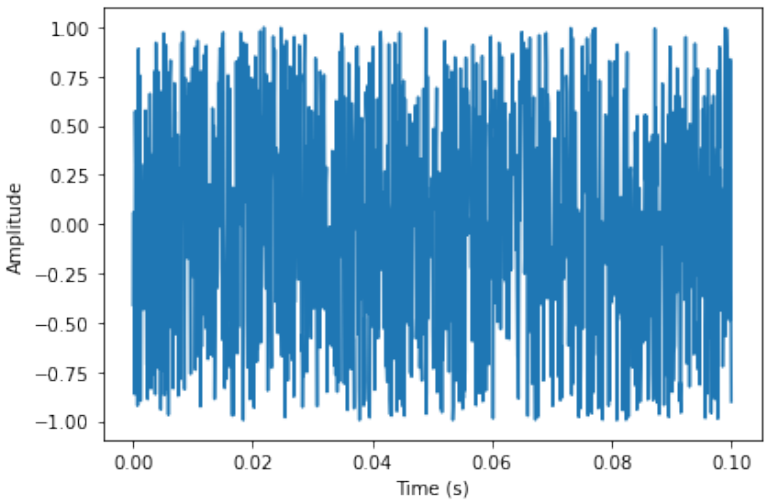

Рис. 1.1: Сегмент полученного сигнала

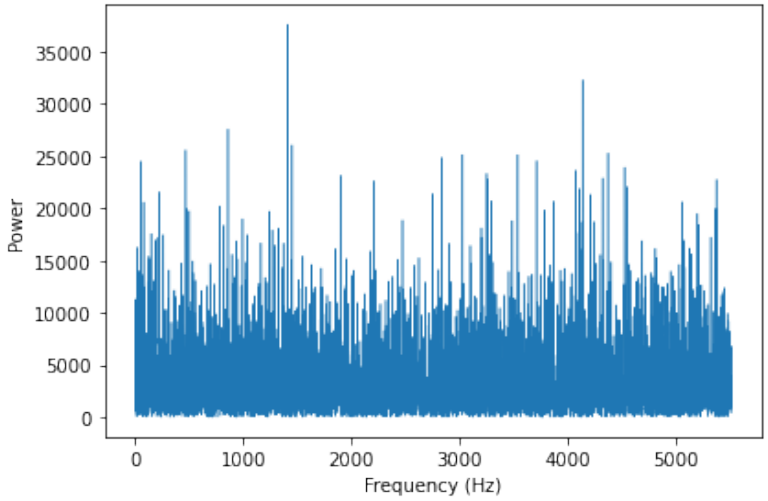

Рис. 1.2: Спектр энергии сигнала (квадрат амплитуды)

Некоррелированный шум в среднем имеет одинаковую мощность на всех частотах, что мы можем подтвердить, построив интегральный спектр (нормализованная сумма мощности).

```
integ = spectrum.make_integrated_spectrum()
integ.plot_power()

```

Листинг 1.2: Строим интегральный спектр шума

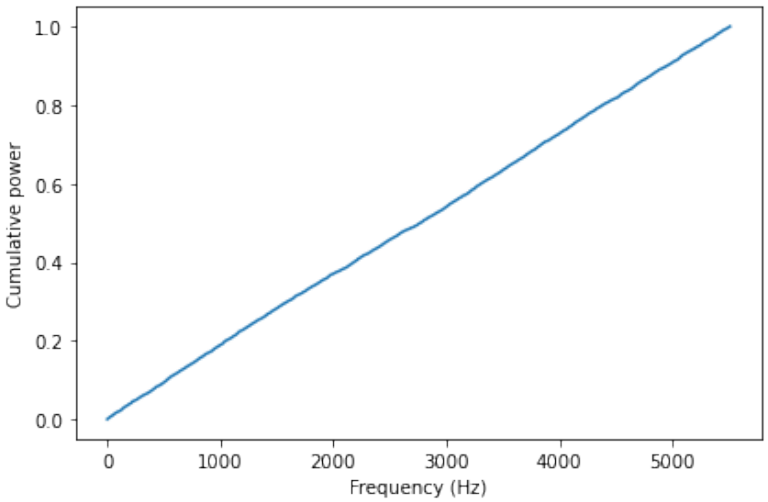

Рис. 1.3: Результат

Шум с одинаковой мощностью на всех частотах называется «Белым» шумом.

Глава 2

Броуновский шум

Изучим Броуновский шум.

```
1         from thinkdsp import BrownianNoise
2
3         signal = BrownianNoise()
4         wave = signal.make_wave(duration = 1, framerate =
    11025)
5         wave.plot(linewidth = 1)
6         spectrum = wave.make_spectrum()
7         spectrum.plot_power(linewidth = 0.5)
8
```

Листинг 2.1: Построение сигнала с Броуновским шумом

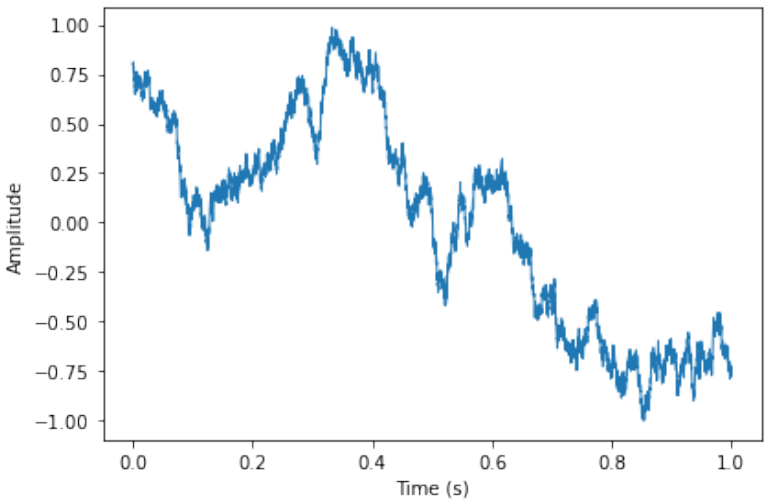

Рис. 2.1: Сигнал с Броуновским шумом

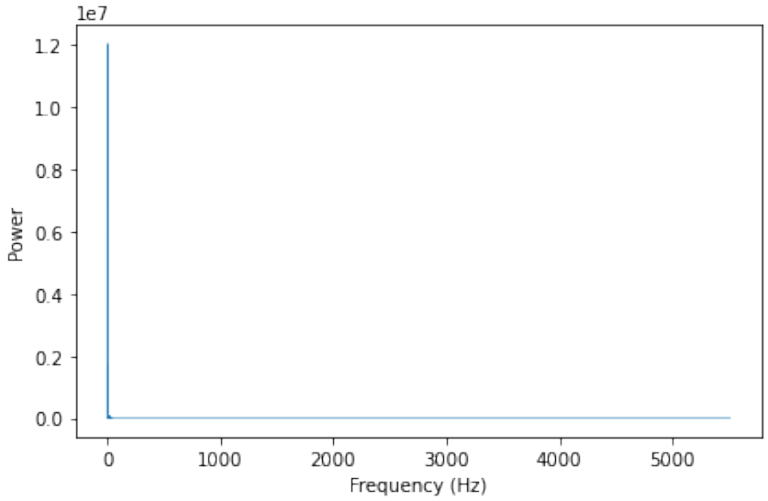

Рис. 2.2: Его зависимость энергии от частоты

Вся энергия сигнала находится на низких частотах. Поэтому, применим логарифмический масштаб.

```
        spectrum.hs[0] = 0
        spectrum.plot_power(linewidth = 0.5)
        loglog = dict(xscale = 'log', yscale = 'log')
        result = spectrum.estimate_slope().slope

```

Листинг 2.2: Сигнал с Броуновским шумом в логарифмическом масштабе

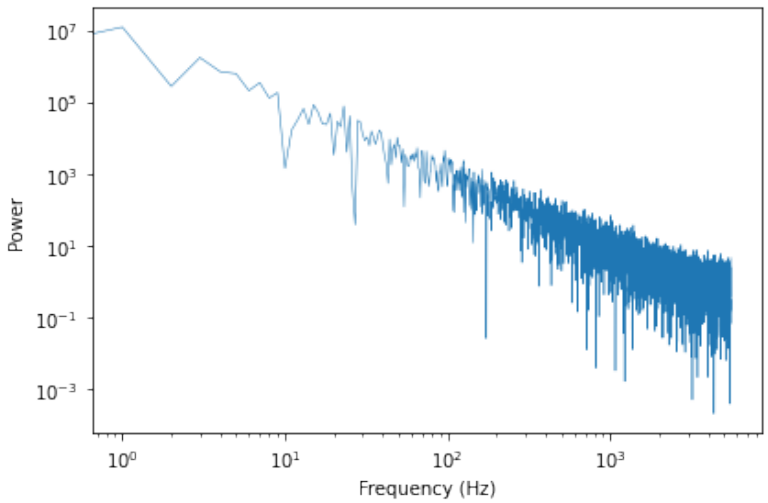

Рис. 2.3: Так лучше видна зависимость энергии сигнала от его частоты

Наклон этой линии приблизительно равен -2, что указывает на зависимость $\frac{1}{f^2}$.

Глава 3

Розовый шум

Изучаем `Розовый шум`. Розовый шум характеризуется параметром β, обычно между 0 и 2.

```
        for beta in [0, 1, 2]:
            signal = PinkNoise(beta = beta)
            wave = signal.make_wave(duration=1, framerate
    =1024)
            spectrum = wave.make_spectrum()
            spectrum.hs[0] = 0
            label = f'beta={beta}'
            spectrum.plot_power(linewidth = 1, alpha = 0.7,
    label = label)

```

Листинг 3.1: Построение Розового шума

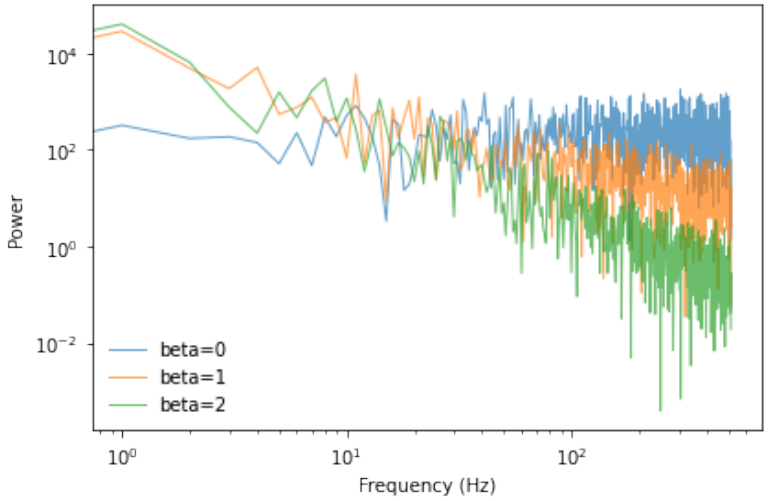

Рис. 3.1: Результаты построения Розового шума с различным значением аргумента

При $\beta = 0$ Розовый шум - «Белый» шум, а при $\beta = 2$ - «Красный» шум (Броуновский).

Глава 4

Гауссов шум

А теперь посмотрим на некоррелированный гауссов шум.

```
from thinkdsp import UncorrelatedGaussianNoise

signal = UncorrelatedGaussianNoise()
wave = signal.make_wave(duration=1, framerate=11025)
wave.plot(linewidth = 0.5)
spectrum = wave.make_spectrum()
spectrum.plot_power(linewidth = 1)

```

Листинг 4.1: Гауссов шум

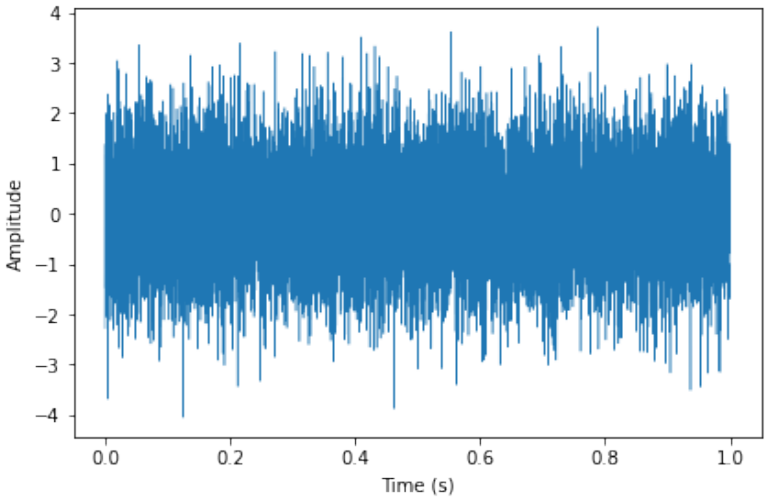

Рис. 4.1: Сигнал с гауссовым шумом

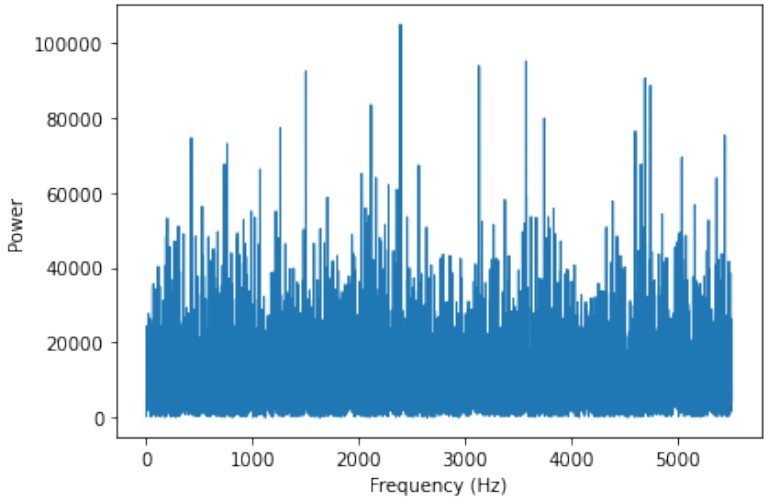

Рис. 4.2: Зависимость энергии от частоты

```
1         def normal_prob_plot(sample, fit_color = '0.8', **
  options):
2             n = len(sample)
3             xs = np.random.normal(0, 1, n)
4             xs.sort()
5             ys = np.sort(sample)
6             mean, std = np.mean(sample), np.std(sample)
7             fit_ys = mean + std * xs
8             plt.plot(xs, fit_ys, color = 'gray', alpha = 0.5,
   label = 'model')
9             plt.plot(xs, ys, **options)
10
11        normal_prob_plot(spectrum.real, color = 'C0', label =
   'real part')
12        normal_prob_plot(spectrum.imag, color = 'C1', label =
   'imag part')
13
```

Листинг 4.2: График нормальной вероятности

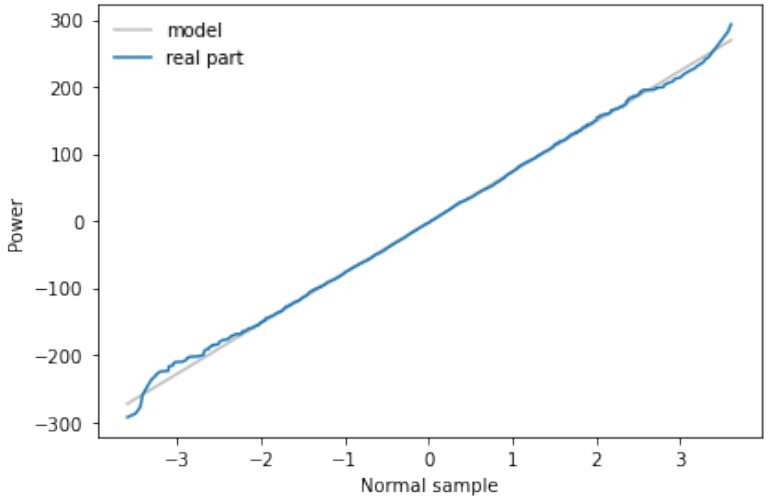

Рис. 4.3: Распределение действительной части спектра является гауссовым

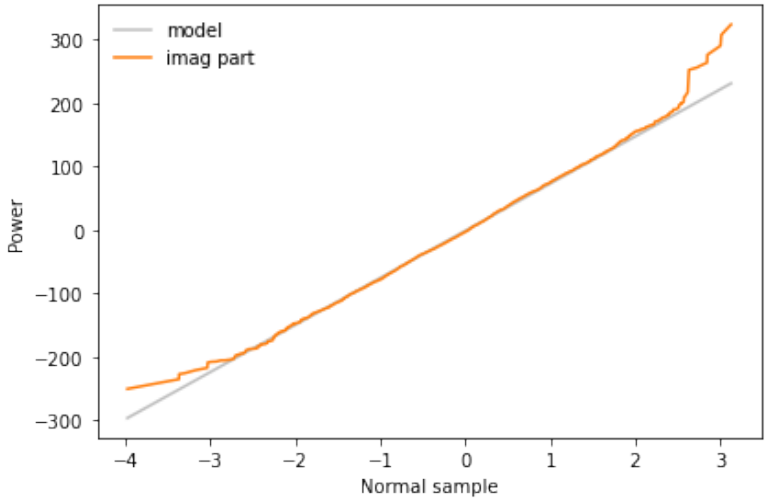

Рис. 4.4: Также как и его мнимая часть

Глава 5

Упражнения

5.1 Задание 1

Исследование природных источнников шума.

```
        from thinkdsp import read_wave

        wave=read_wave('132736__ciccarelli__ocean-waves.wav')
        segment = wave.segment(start = 5.0, duration = 1.0)
        segment2 = wave.segment(start = 10.0, duration = 1.0)
        spectrum = segment.make_spectrum()
        spectrum2 = segment2.make_spectrum()

        spectrum.plot_power(alpha = 0.5)
        spectrum2.plot_power(alpha = 0.5)
        decorate(xlabel = 'Frequency (Hz)', ylabel = '
    Amplitude')

        spectrum.plot_power(alpha = 0.5)
        spectrum2.plot_power(alpha = 0.5)
        decorate(xlabel = 'Frequency (Hz)', ylabel = '
    Amplitude', **loglog)

        segment.make_spectrogram(512).plot(high = 5000)

```

Листинг 5.1: Два сегмента сигнала с шумом моря

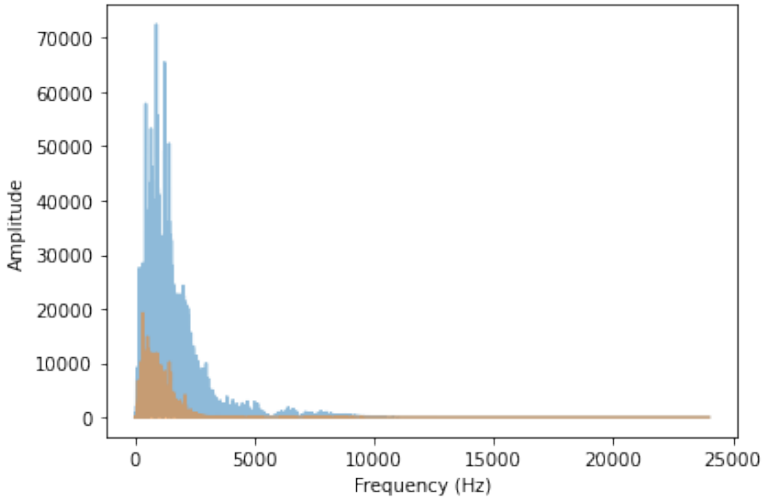

Рис. 5.1: Спектр сигнала

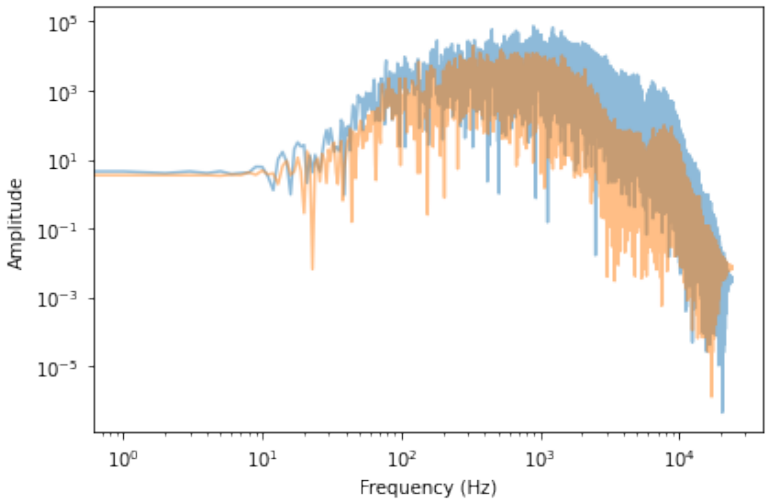

Рис. 5.2: Логарифмический масштаб

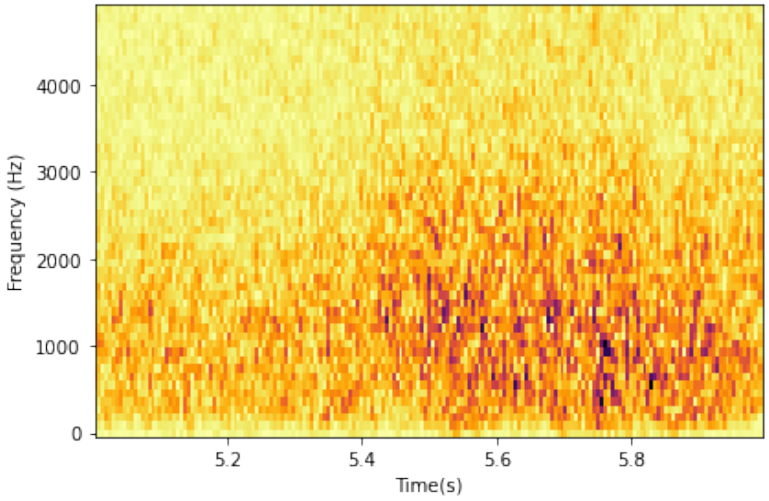

Рис. 5.3: Спектрограмма шума для первого сегмента

Общая амплитуда падает, но смесь частот кажется последовательной.

```
wave = read_wave('fire.wav')
segment = wave.segment(start = 1.0, duration = 1.0)
segment2 = wave.segment(start = 2.5, duration = 1.0)
...

```

Листинг 5.2: Взяли другой сигнал

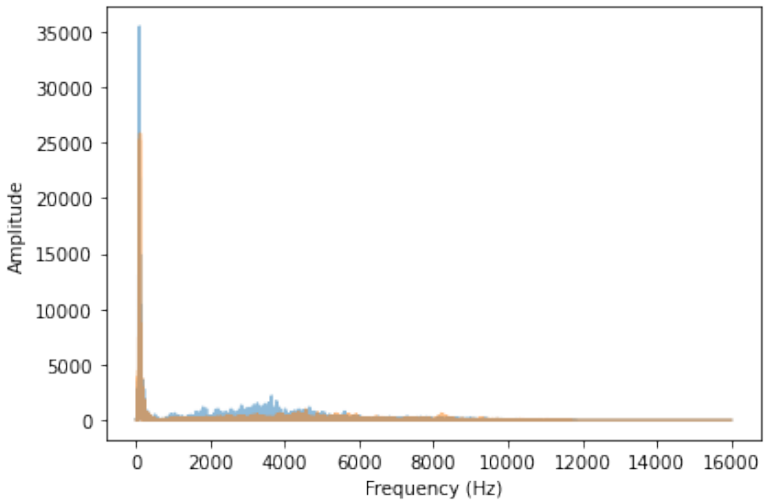

Рис. 5.4: Нашли его спектр

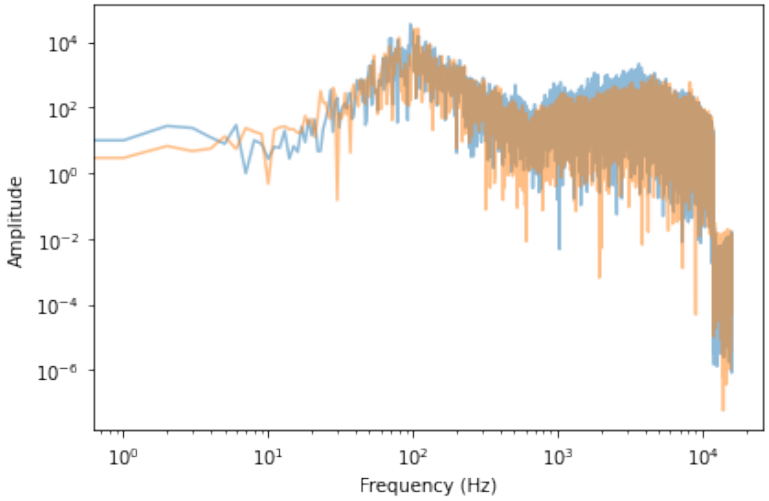

Рис. 5.5: Очень интересная форма

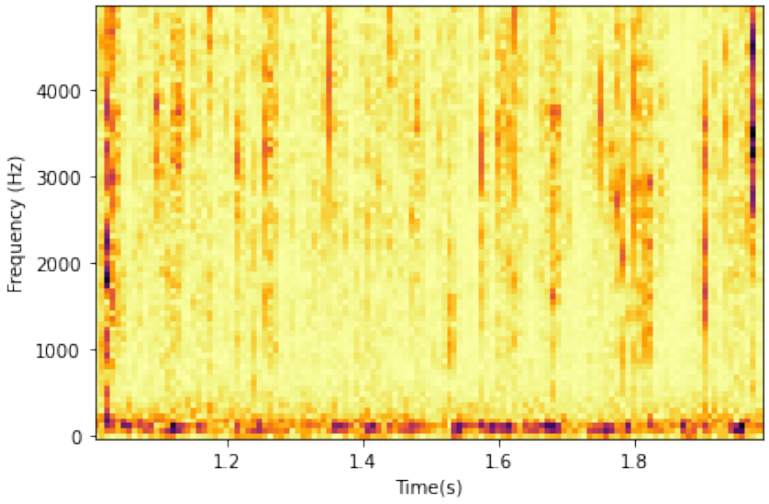

Рис. 5.6: Спектрограмма первого сегмента сигнала

Сигнал чем-то напоминает «Розовый» шум, но очень отдаленно.

5.2 Задание 2

Метод Бартлетта и оценка спектра мощности шумового сигнала.

```
from thinkdsp import Spectrum

```

```
3         def bartlett_method(wave, seg_length = 512, win_flag
    = True):
4             spectro = wave.make_spectrogram(seg_length,
    win_flag)
5             spectrums = spectro.spec_map.values()
6             psds = [spectrum.power for spectrum in spectrums]
7             hs = np.sqrt(sum(psds) / len(psds))
8             fs = next(iter(spectrums)).fs
9             spectrum = Spectrum(hs, fs, wave.framerate)
10            return spectrum
11
12        psd = bartlett_method(segment)
13        psd2 = bartlett_method(segment2)
14        psd.plot_power()
15        psd2.plot_power()
16
```

Листинг 5.3: Метод Бартлетта

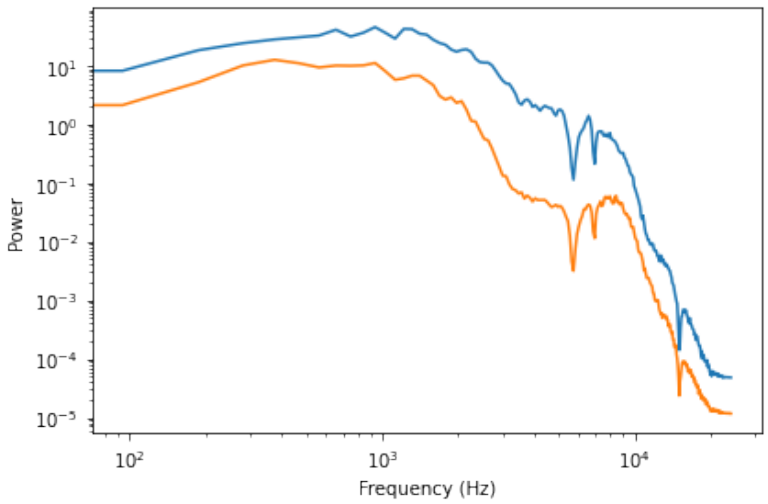

Рис. 5.7: Результат применения метода Бартлетта к сигналу шума моря

Мы можем более лучше видеть взаимосвязь между мощностью и частотой. Это не простая линейная зависимость, но она последовательна в разных сегментах.

5.3 Задание 3

Спектр цен BitCoin.

```
1         import pandas as pd
2         from thinkdsp import Wave
3
4         df = pd.read_csv('BTC_USD_2013-10-01_2020-03-26-
    CoinDesk.csv', parse_dates=[0])
5         ys = df['Closing Price (USD)']
6         ts = df.index
7         wave = Wave(ys, ts, framerate = 1)
8         wave.plot()
9         spectrum = wave.make_spectrum()
10        spectrum.plot_power()
11        spectrum.estimate_slope()[0]
12
```

Листинг 5.4: BitCoin

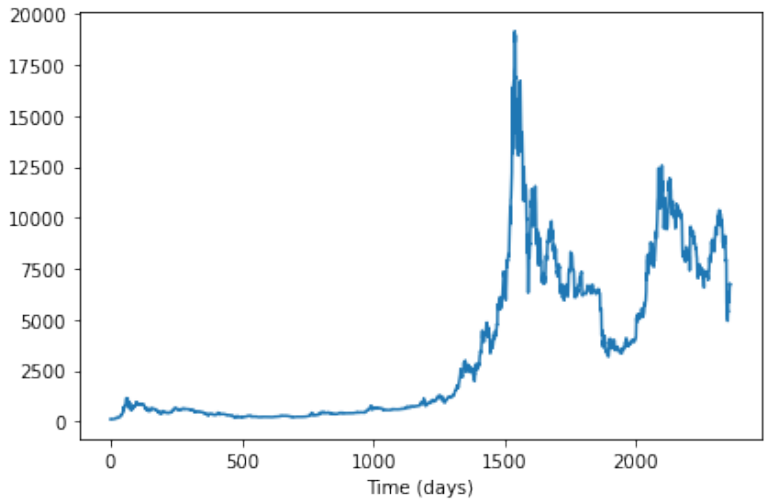

Рис. 5.8: Визуализация изменений цен BitCoin

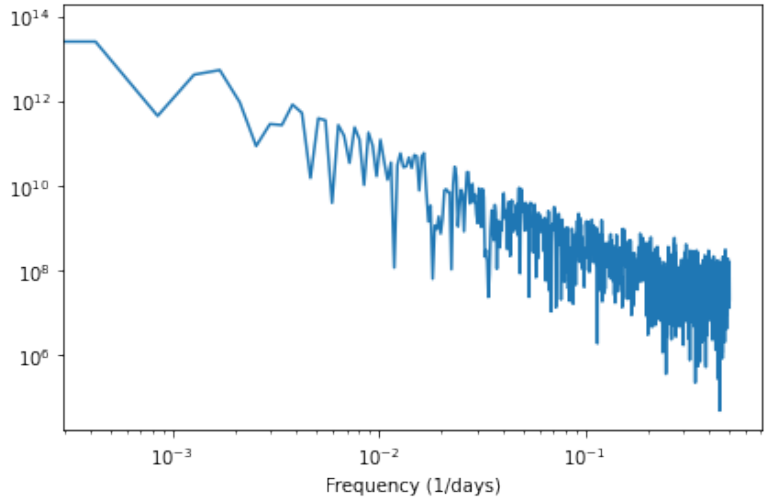

Рис. 5.9: Спектр полученного сигнала в логарифмическом масштабе

Наклон этой кривой близок к 1.7, поэтому трудно сказать, это «Красный» шум или своего рода «Розовый» шум.

5.4 Задание 4

Счётчик Гейгера

```
        from thinkdsp import Noise

        class UncorrelatedPoissonNoise(Noise):

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

        amp = 0.001
        framerate = 10000
        duration = 2
        signal = UncorrelatedPoissonNoise(amp = amp)
        wave = signal.make_wave(duration = duration,
  framerate = framerate)
        expected = amp * framerate * duration
        actual = sum(wave.ys)
        wave.plot()
        spectrum = wave.make_spectrum()
        spectrum.plot_power()

```

Листинг 5.5: Реализация счётчика Гейгера

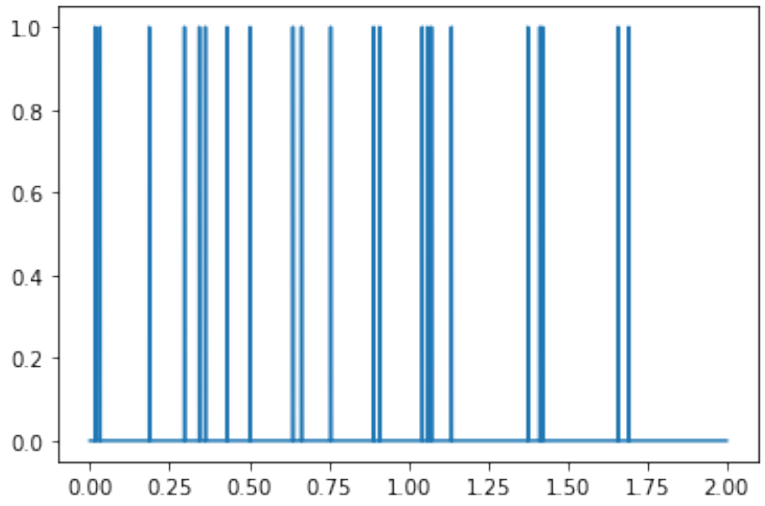

Рис. 5.10: Полученный сигнал

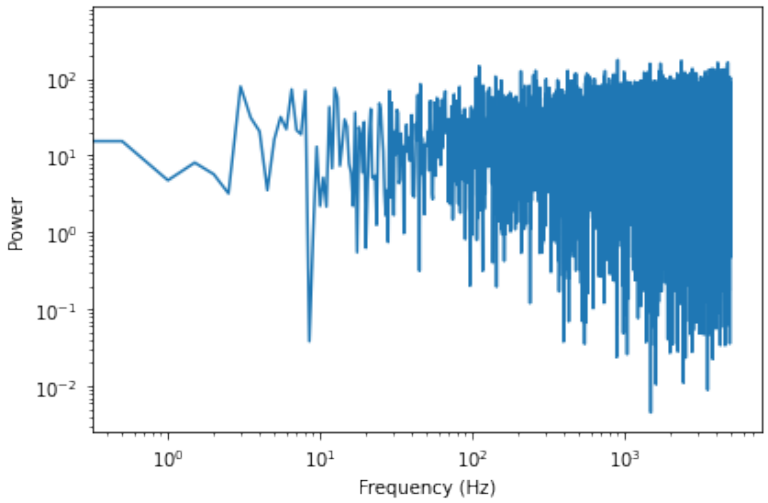

Рис. 5.11: Спектр счетчика Гейгера

Наклон прямой близок к 0, поэтому можно сказать, что это «Белый» шум.

```
        amp = 1
        framerate = 10000
        duration = 2
        signal = UncorrelatedPoissonNoise(amp = amp)
        wave = signal.make_wave(duration = duration,
framerate = framerate)
```

```
6           spectrum = wave.make_spectrum()
7           spectrum.hs[0] = 0
8           normal_prob_plot(spectrum.real, label = 'real')
9           normal_prob_plot(spectrum.imag, label = 'imag', color
        = 'C1')
10
```

Листинг 5.6: Сравнение спектра с гауссовым шумом

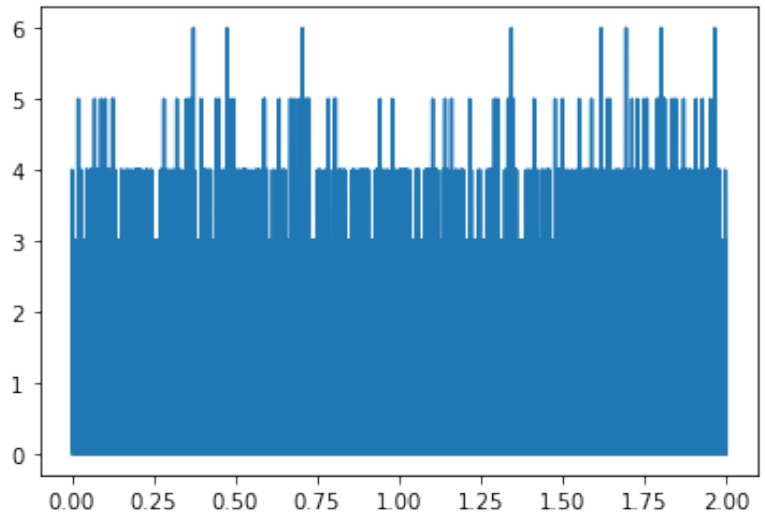

Рис. 5.12: Увеличили амплитуду сигнала

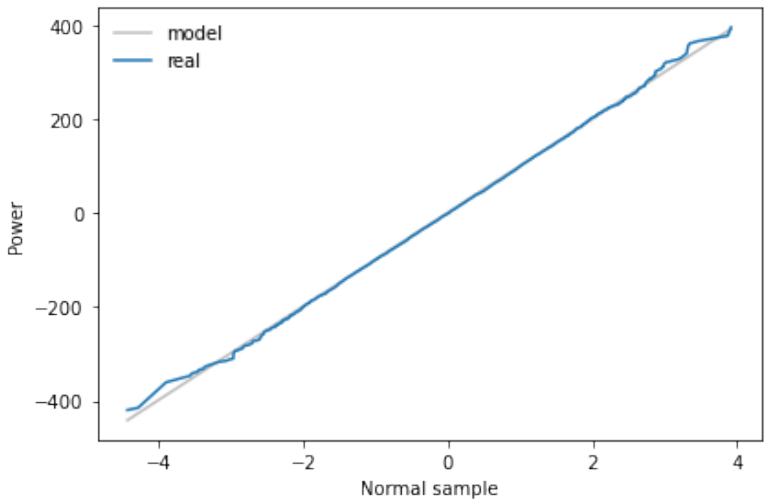

Рис. 5.13: Вещественная часть сигнала

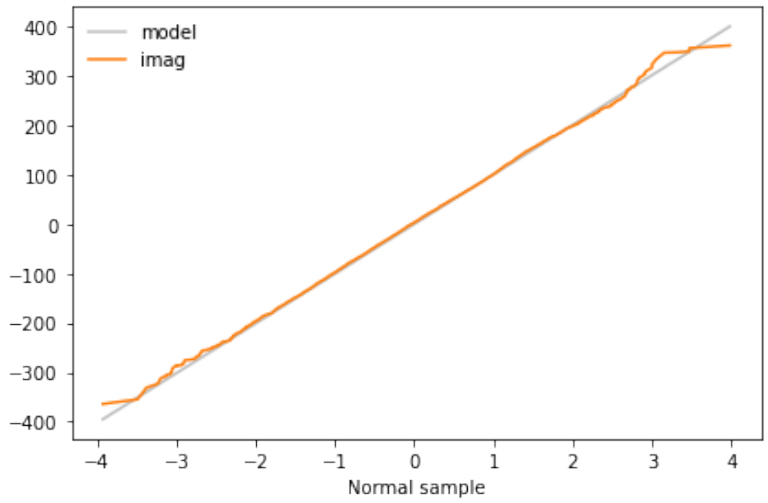

Рис. 5.14: Мнимая часть сигнала

5.5 Задание 5

Алгоритм Vocc-McCartney для генерации «Розового» шума.

```
        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)
            df = pd.DataFrame(array)
            df.fillna(method = 'ffill', axis = 0, inplace =
    True)
            total = df.sum(axis = 1)
            return total.values

        ys = voss(12500)
        wave = Wave(ys)
        wave.plot()
        spectrum = wave.make_spectrum()
        spectrum.hs[0] = 0
        spectrum.plot_power()

```

Листинг 5.7: Итоговая функция

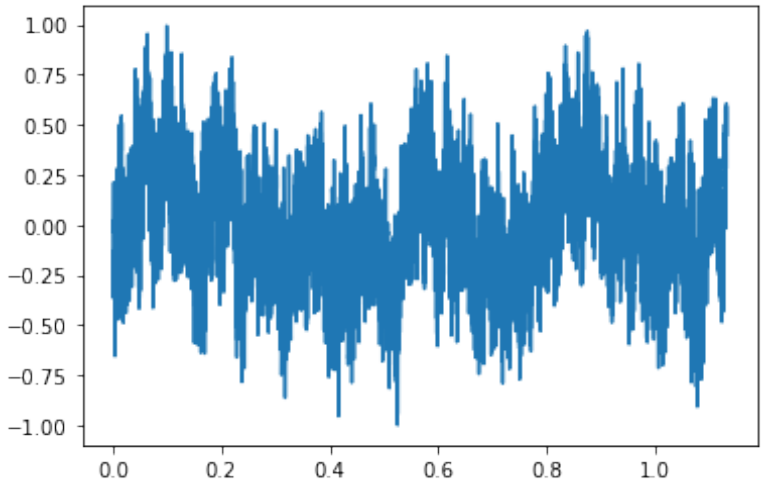

Рис. 5.15: Сигнал, полученный с помощью этой функции

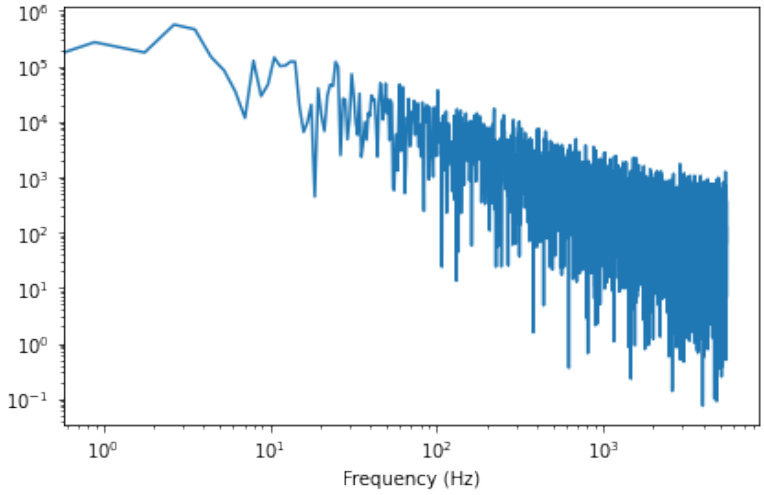

Рис. 5.16: Спектр сигнала в логарифмическом масштабе

Наклон прямой близок к -1.

```
        seg_length = 64 * 1024
        iters = 100
        wave = Wave(voss(seg_length * iters))
        spectrum = bartlett_method(wave, seg_length =
    seg_length, win_flag = False)
        spectrum.hs[0] = 0
        spectrum.plot_power()
```

Листинг 5.8: Более длинная выборка

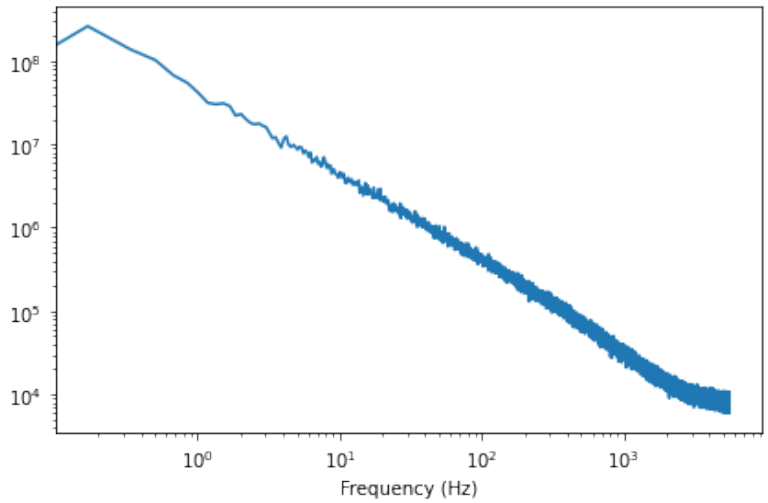

Рис. 5.17: Спектр сигнала с более длинной выборкой в логарифмическом масштабе

Наклон прямой очень близок к -1. Теперь точно можно сказать, что это «Розовый» шум.

Глава 6

Вывод

В данной работе мы познакомились с различными видами шума: «Белым», «Розовым» и «Красным». Определили, что их отличает друг от друга. Также рассмотрели спектры некоторых сигналов в логарифмическом масштабе.