Санкт-Петербургский Государственный Политехнический Университет
Институт Компьютерных Наук и Технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №4
на тему
Шум

Работу выполнил
Студент группы 3530901/80203
Курняков П.М.
Преподаватель
Богач Н.В.

Санкт-Петербург, 2021 год

1 Теоритическая часть о свойствах преобразования фурье

Преобразование Фурье функции f вещественной переменной является интегральным и задаётся следующими формулами:

$$\text{Прямое:} \quad F(\nu) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i \nu t} dt$$
$$\text{Обратное:} \quad f(t) = \int_{-\infty}^{\infty} F(\nu) e^{2\pi i \nu t} d\nu$$

2 Свойства

2.1 Линейность

По определению, для некоторого векторного пространства $(V, K, +, \cdot)$, $a, b \in V$, $\gamma \in K$:

$$f : V \to V \text{ - линейна} \iff \begin{cases} \gamma \cdot f(a) = f(\gamma \cdot a) \\ f(a) + f(b) = f(a+b) \end{cases}$$

Очевидно, что преобразование Фурье (ПФ) удовлетворяет этому условию (как функция на $(\mathbb{R} \to \mathbb{R}, \mathbb{C}, +, \cdot)$), а следовательно:

$$\begin{aligned} Fourier\left(\sum_i \alpha_i \phi_i(t)\right) &= \sum_i \alpha_i \cdot Fourier(\phi_i(t)) \\ &= \sum_i \alpha_i \Phi_i(\nu) \end{aligned}$$

2.2 Смещение функции

При смещении функции $\phi(t)$ на Δt результат ПФ умножается на $e^{2\pi i \nu \Delta t}$. Пусть $t' = t + \Delta t$, тогда:

$$\begin{aligned} Fourier\left(\phi(t + \Delta t)\right) &= \int_{-\infty}^{\infty} \phi(t + \Delta t) e^{-2\pi i \nu t} dt \\ &= \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \nu (t' - \Delta t)} dt \end{aligned}$$

Так как $dt' = d(t + \Delta t) = dt$, то:

$$\begin{aligned} \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \nu (t' - \Delta t)} dt' &= e^{2\pi i \nu \Delta t} \cdot \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \nu t'} dt' \\ &= e^{2\pi i \nu \Delta t} \cdot F(\nu) \end{aligned}$$

2.3 Масштабирование функции

Пусть $t' = \alpha t$, тогда:

$$\begin{aligned} Fourier\left(\phi(\alpha t)\right) &= \int_{-\infty}^{\infty} \phi(\alpha t) e^{-2\pi i \nu t} dt \\ &= \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \nu \frac{t'}{\alpha}} dt \end{aligned}$$

Так как $dt' = \alpha dt$, то для $a > 0$:

$$\begin{aligned} \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \nu \frac{t'}{\alpha}} dt &= \frac{1}{\alpha} \int_{-\infty}^{\infty} \phi(t') e^{-2\pi i \frac{\nu}{\alpha} t'} dt \\ &= \frac{1}{\alpha} \Phi\left(\frac{\nu}{\alpha}\right) \end{aligned}$$

Для $a < 0$ получится $dt' < 0$ при $dt > 0$. При этом нужно поменять пределы интегрирования местами, тогда получим результат с отрицательным знаком:

$$-\frac{1}{\alpha}\Phi\left(\frac{\nu}{\alpha}\right)$$

Таким образом, в одной форме это:

$$\frac{1}{|\alpha|}\Phi\left(\frac{\nu}{\alpha}\right)$$

Вывод: при сжатии функции по времени в α раз, её ПФ расширяется по частоте в α раз.

2.4 Перемножение функции

ПФ произведения двух функций - это свёртка их ПФ.

$$\begin{aligned}
Fourier\left(\phi(t)\xi(t)\right) &= \int_{-\infty}^{\infty} \phi(t)\xi(t)e^{-2\pi i\nu t}dt \\
&= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} \Phi(k)e^{2\pi ikt}dk\right)\xi(t)e^{-2\pi i\nu t}dt \\
&= \int_{-\infty}^{\infty} \Phi(k)\left(\int_{-\infty}^{\infty} \xi(t)e^{2\pi i(k-\nu)t}dt\right)dk \\
&= \int_{-\infty}^{\infty} \Phi(k)\left(\int_{-\infty}^{\infty} \xi(t)e^{-2\pi i(\nu-k)t}dt\right)dk \\
&= \int_{-\infty}^{\infty} \Phi(k)\Xi(\nu-k)dk \\
&= \left(\Phi * \Xi\right)(\nu)
\end{aligned}$$

2.5 Свёртывание функции

ПФ свёртки двух функций есть произведение ПФ этих функций. Доказывается аналогично в силу «симметрии» прямого и обратного преобразований Фурье.

2.6 Дифференцирование функции

При дифференцировании $\phi(t)$ по t её ПФ умножается на $2\pi i\nu$.

$$\begin{aligned}
Fourier\left(\frac{d\phi(t)}{dt}\right) &= \int_{-\infty}^{\infty} \frac{d\phi(t)}{dt}e^{-2\pi i\nu t}dt \\
&= \int_{-\infty}^{\infty} e^{-2\pi i\nu t}d\phi(t) \\
&= \phi(t)e^{-2\pi\nu t}\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \phi(t)d\left(e^{-2\pi i\nu t}\right) \\
&= \phi(t)e^{-2\pi\nu t}\Big|_{-\infty}^{\infty} + 2\pi i\nu\int_{-\infty}^{\infty} \phi(t)e^{-2\pi i\nu t}dt \\
&= \phi(t)e^{-2\pi\nu t}\Big|_{-\infty}^{\infty} + 2\pi i\nu\cdot\Phi(\nu)
\end{aligned}$$

Прямое и обратное преобразование Фурье существует для функций с ограниченной энергией, то есть:

$$\int_{-\infty}^{\infty} |\phi(t)|^2dt \neq \infty$$

И из этого следует, что первое слагаемое равно 0.

2.7 Интегрирование функции

При интегрировании ПФ делится на $2\pi i\nu$.

$$
\begin{aligned}
Fourier\left(\int_{-\infty}^{t}\phi(t')dt'\right) &= \int_{-\infty}^{\infty}\left(\int_{-\infty}^{t}\phi(t')dt'\right)e^{-2\pi i\nu t}dt \\
&= -\frac{1}{2\pi i\nu}\cdot\int_{-\infty}^{\infty}\left(\int_{-\infty}^{t}\phi(t')dt'\right)d\left(e^{-2\pi i\nu t}\right) \\
&= -\frac{1}{2\pi i\nu}\cdot\left[e^{-2\pi i\nu t}\int_{-\infty}^{t}\phi(t')dt'\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty}e^{-2\pi i\nu t}d\left(\int_{-\infty}^{t}\phi(t')dt'\right)\right] \\
&= -\frac{1}{2\pi i\nu}\cdot\left[e^{-2\pi i\nu t}\int_{-\infty}^{t}\phi(t)dt\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty}e^{-2\pi i\nu t}\phi(t)dt\right] \\
&= -\frac{1}{2\pi i\nu}\cdot\left[0 - \int_{-\infty}^{\infty}e^{-2\pi i\nu t}\phi(t)dt\right] \\
&= \frac{1}{2\pi i\nu}\cdot\int_{-\infty}^{\infty}e^{-2\pi i\nu t}\phi(t)dt \\
&= \frac{1}{2\pi i\nu}\cdot\Phi(\nu)
\end{aligned}
$$

0 возникает потому, что $\int_{-\infty}^{\infty}\phi(t')dt' = 0$.

2.8 Обратимость

Преобразования обратимы, причём обратное преобразование имеет практически такую же форму, как и прямое преобразование.

3 Настройка проекта

Перед тем как выполнять задания необходимо настроить проект и сделать все необходимые импорты:

```
from __future__ import print_function, division

import thinkdsp
import thinkplot
import thinkstats2

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

%matplotlib inline
```

Рис. 1: 2

4 Упражнение номер №1

Необходимо скачать звук природных источников шума. Определить похож ли спектр мощности скаченного звука на белый розовый или броуновский шумы.

Для данного задания я скачал звук волн черного моря:

```python
wave = thinkdsp.read_wave('task1.wav')
wave.make_audio()
```

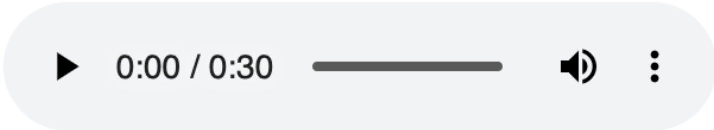

Рис. 2: 2

Выберем небольшой сегмент:

```python
segment = wave.segment(start=1.5, duration=1.0)
segment.make_audio()
```

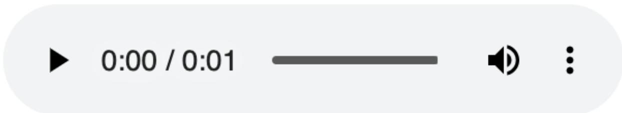

Рис. 3: 2

Распечатаем спектр:

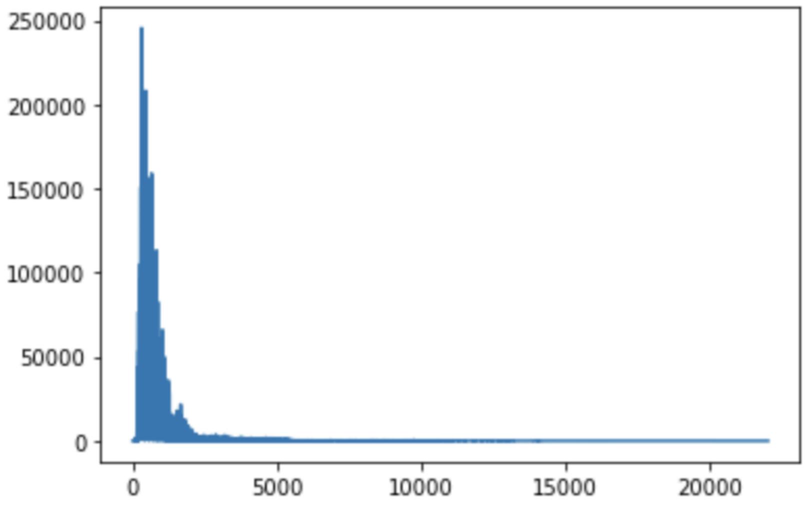

Рис. 4: 2

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

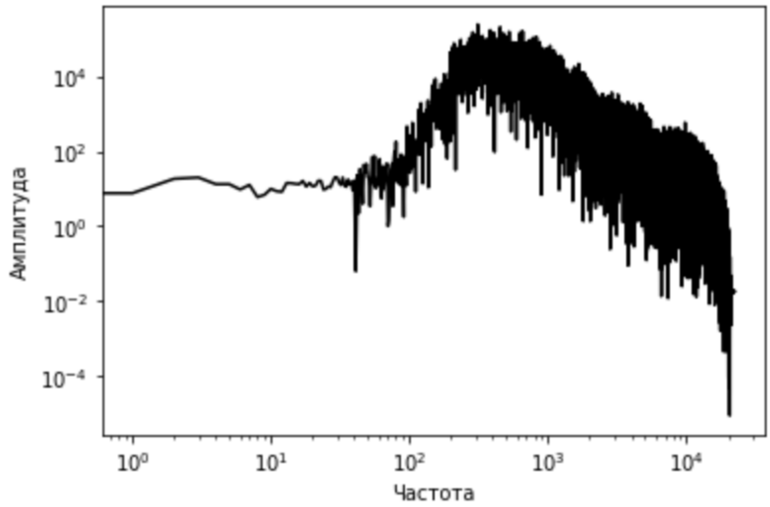

Рис. 5: 2

Амплитуда этой структуры сначала увеличивается, а затем уменьшается. Это обычное явлением для естественных источников шума. Выберем другой отрезок:

```
segment2 = wave.segment(start=2.5, duration=1.0)
segment2.make_audio()
```

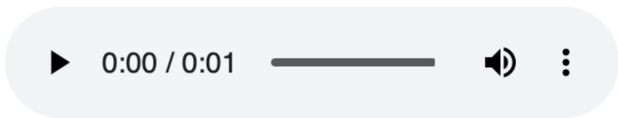

Рис. 6: 2

Отобразим спектр этих 2 сегментов:

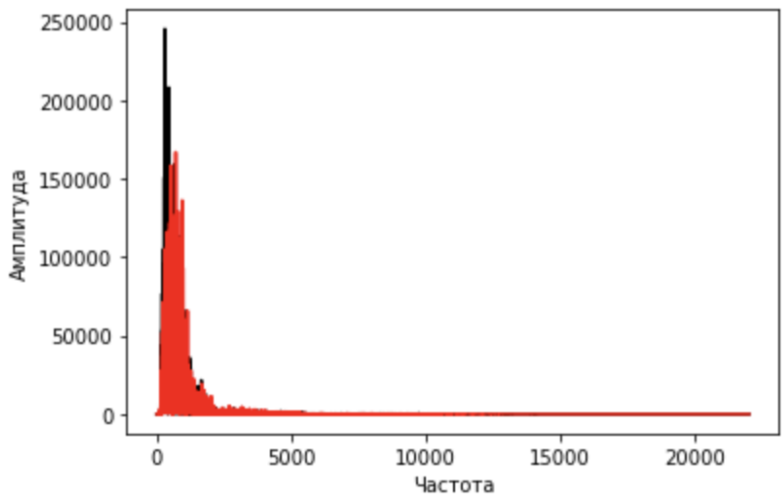

Рис. 7: 2

Рассмотрим их в логарифмической метрики:

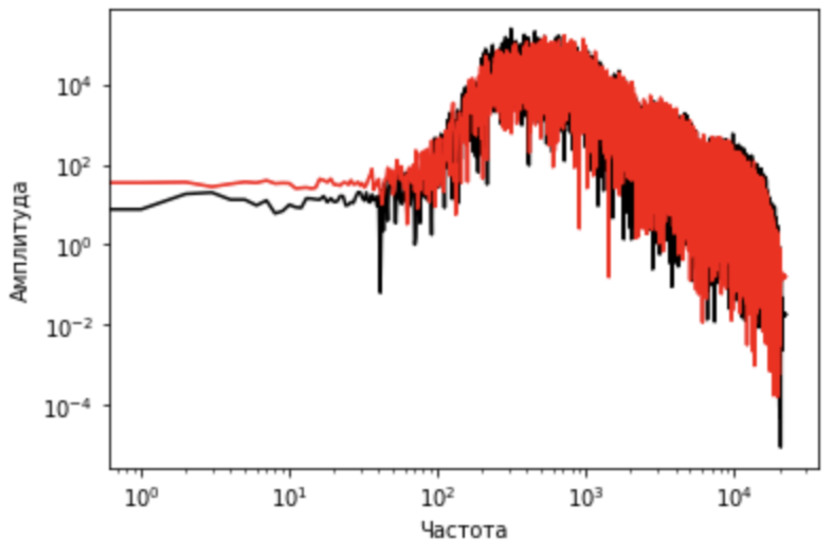

Рис. 8: 2

Из данного сравнения мы можем заметить что поведение данной структуры остается практически неизменным с течением времени.

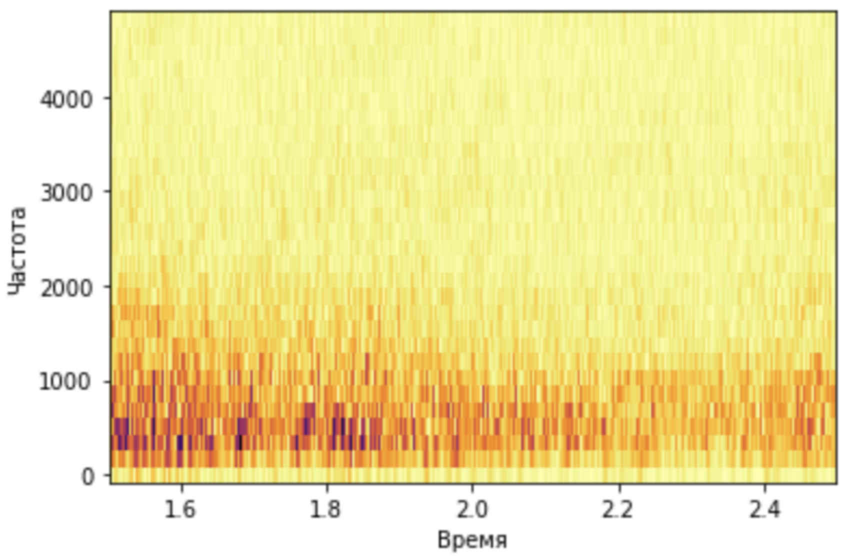

Рис. 9: 2

В этом сегменте общая амплитуда падает, но смесь частот кажется стабильной.

5 Упражнение номер №2

Реализовать метод Бартлетта и с помощью него провести оценку спектра мощности шумового сигнала.

bartlett_method создает спектрограмму и извлекает spec_map, который отображает время на объекты Spectrum. Он вычисляет PSD для каждого спектра, складывает их и помещает результаты в объект Spectrum.

Реализуем метод:

```
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
```

Рис. 10: 2

Построим график:

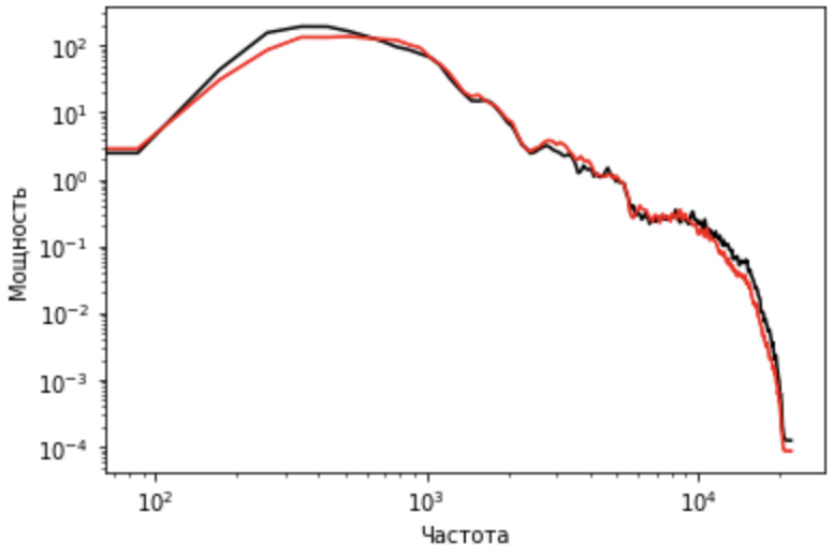

Рис. 11: 2

Теперь мы можем более четко увидеть взаимосвязь между мощностью и частотой. Это не простая линейная зависимость, но она одинакова для разных сегментов, даже в деталях, таких как выемки около 1000 Гц, 4000 Гц и тд..

6 Упражнение номер №3

Открыть файл об исторических данных о ежедневной цене биткоина. Вычислить спектр цен на биткоин как функцию времени. Определить схожесть с белым розовым или броуновским шумами.

Отобразим результаты из файла:

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2020-01-02	7174.744012	7179.957689	7237.014866	7152.992402
1	BTC	2020-01-03	6955.487580	7174.712357	7190.188749	6914.857474
2	BTC	2020-01-04	7291.219505	6955.487580	7390.041835	6852.093401
3	BTC	2020-01-05	7337.636670	7291.217504	7390.762935	7263.178696
4	BTC	2020-01-06	7347.433264	7337.421391	7487.333871	7316.763370
...	...	...	...	...	...	...
362	BTC	2020-12-29	26718.029463	26226.066130	27447.551384	26046.625578
363	BTC	2020-12-30	26975.729565	27038.735676	27169.225558	25875.049786
364	BTC	2020-12-31	28768.836208	27349.327233	28928.214391	27349.283204
365	BTC	2021-01-01	29111.521567	28872.829775	29280.045328	27916.625059
366	BTC	2021-01-02	29333.605121	28935.810981	29601.594898	28753.412314

367 rows × 6 columns

Рис. 12: 2

Выгрузка даты об изменении цены на биткоин в долларх в период с 1 января 2020 по 1 января 2021

Построим график:

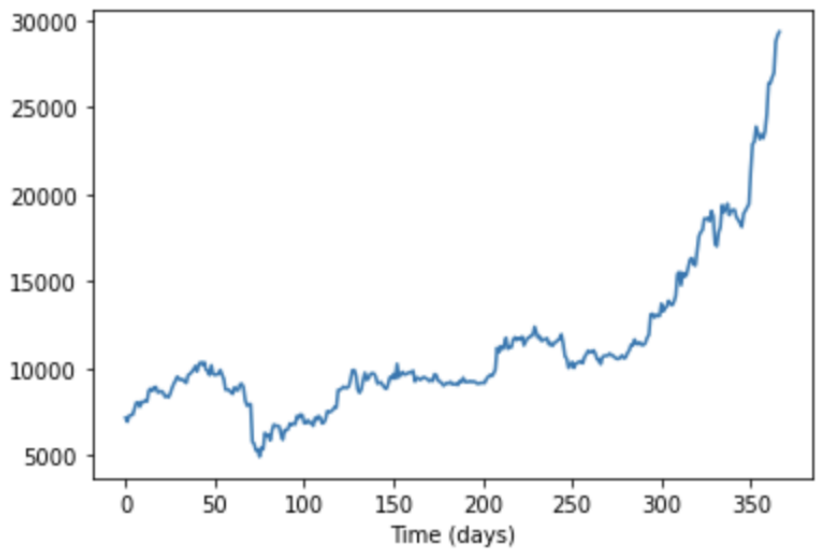

Рис. 13: 2

Построим спектр, где частота 1/дни:

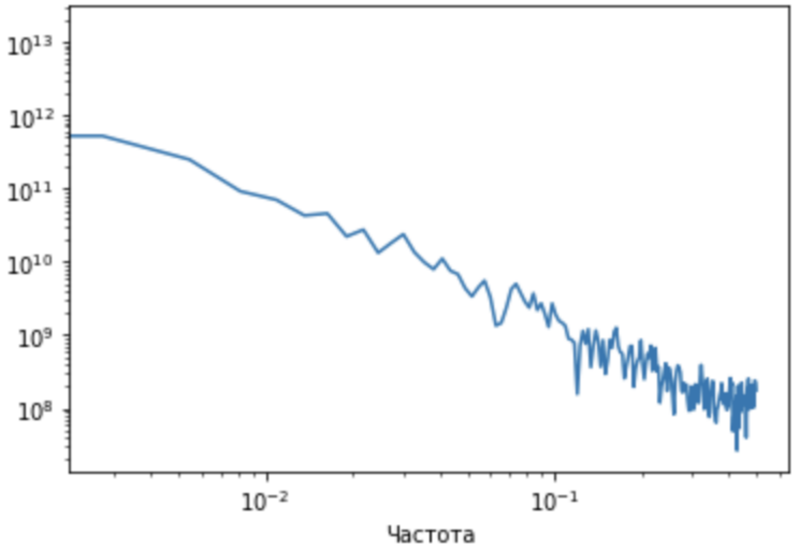

Рис. 14: 2

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

```
-1.740740204158508
```

Рис. 15: 2

Красный шум должен иметь наклон -2. Наклон этого спектра близок к -1,7, сложно сделать вывод о принадлежности его к розовому или красному шумам.

7 Упражнение номер №4

Необходимо реалищовать класс, UncorrelatedPoissonNoise, который наследуется от _Noise и обеспечивает оценку. Он должен использовать np.random.poisson для генерации случайных значений из распределения Пуассона. Параметр этой функции lam - среднее количество частиц за каждый интервал. Для этого мы можем использовать атрибут amp, чтобы указать lam. Например, если частота кадров составляет 10 кГц, а ампер - 0,001, мы ожидаем около 10 «щелчков» в секунду.

Реализуем класс:

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

Рис. 16: 2

Выведем звук при низком уровне радиации:

```
amp = 0.001
framerate = 10000
duration = 1

signal = UncorrelatedPoissonNoise(amp=amp)
wave = signal.make_wave(duration=duration, framerate=framerate)
wave.make_audio()
```

Рис. 17: 2

Чтобы убедиться, что все работает, мы сравниваем ожидаемое количество частиц и фактическое количество:

```
expected = amp * framerate * duration
actual = sum(wave.ys)
print(expected, actual)
```

```
10.0 10
```

Рис. 18: 2

Рассмотрим волну:

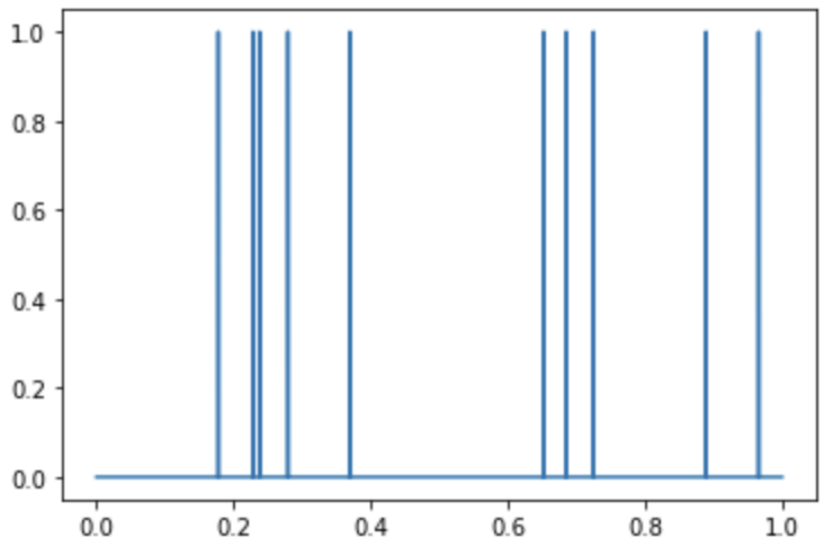

Рис. 19: 2

Также рассмотрим спектр мощности в логарифмической метрики:

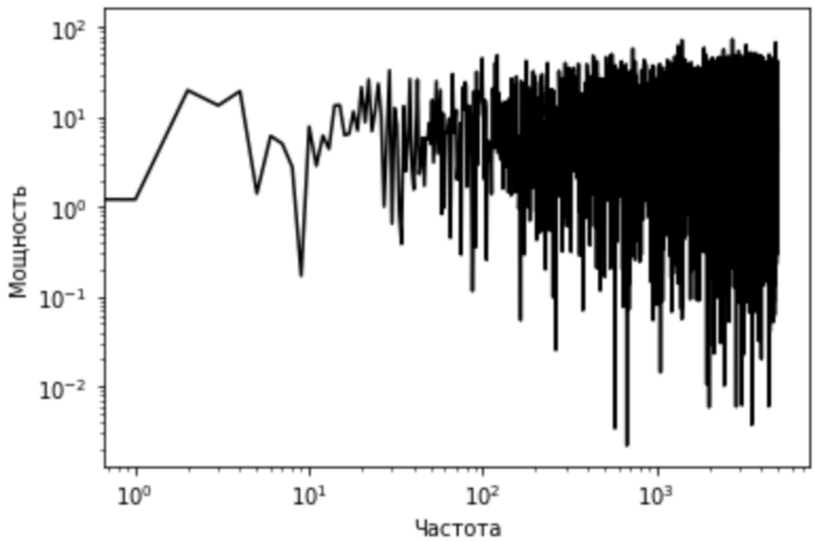

Рис. 20: 2

```
: spectrum.estimate_slope().slope
: 0.0018917076197293935
```

Рис. 21: 2

Наклон составляет 0.0019, что практически соотвестувует белому шуму.
Выведем звук при высоком уровне радиации:

```
amp = 1
framerate = 10000
duration = 1

signal = UncorrelatedPoissonNoise(amp=amp)
wave = signal.make_wave(duration=duration, framerate=framerate)
wave.make_audio()
```

Рис. 22: 2

Отобразим волну на графике:

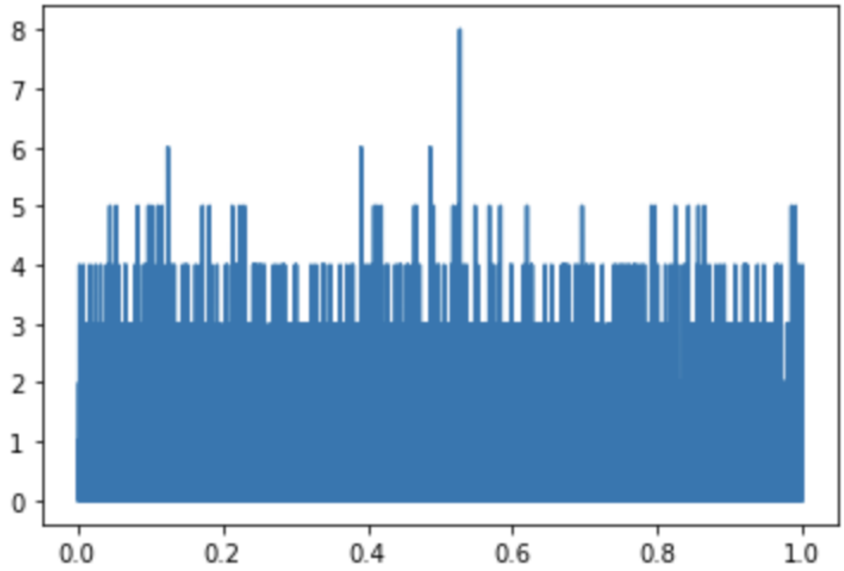

Рис. 23: 2

И спектр сходится на гауссовском шуме.

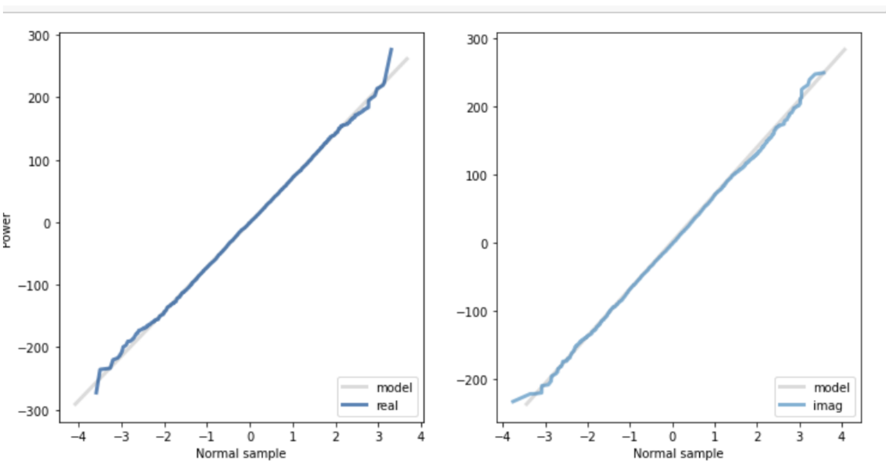

Рис. 24: 2

8 Упражнение номер №5

Изучить способ более эффективного варианта реализации алгоритма для генерации розового шума. Вычислить спектр результата и убедиться, что соотношение между мощностью и частотой соответствующее.

Используем алгоритм Восса-Маккартни.

```python
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
```

Рис. 25: 2

Для проверки сгенерируем 10000 значений:

```
array([7.43336077, 8.31916604, 8.25946949, ..., 9.26148084, 8.9171
6615,
       8.4883063 ])
```

Рис. 26: 2

Построим волну:

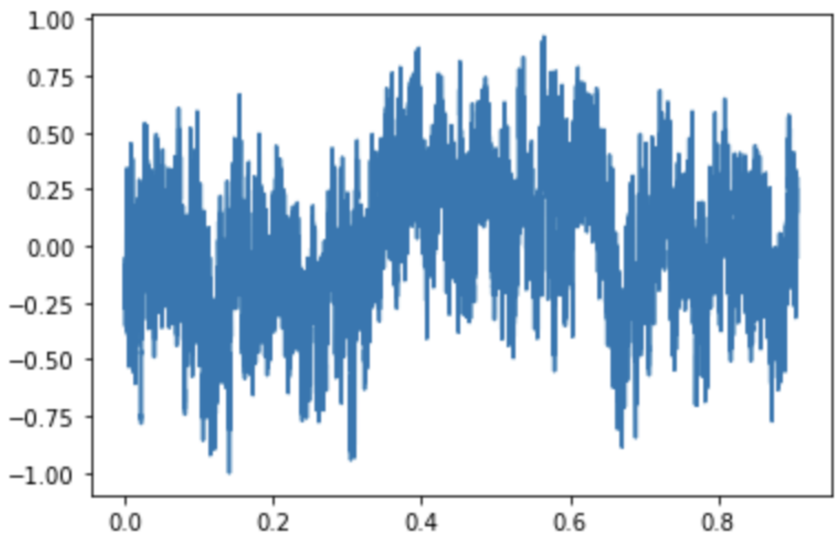

Рис. 27: 2

Можем прослушать данный сигнал:

```
)]: wave.make_audio()
```

)]:

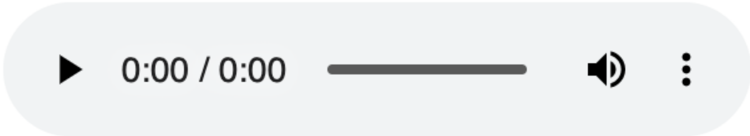

Рис. 28: 2

Вычислим спектр мощности:

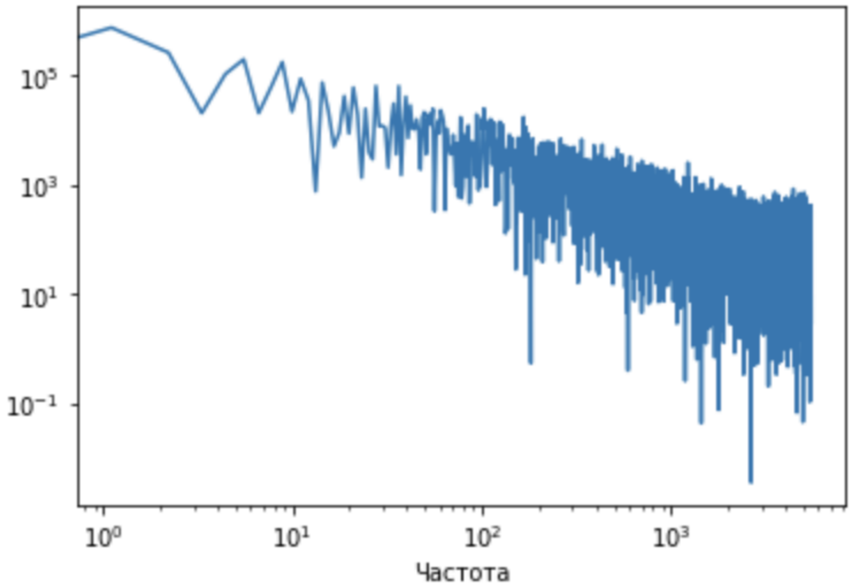

Рис. 29: 2

```
: spectrum.estimate_slope().slope
: -1.0197331142428645
```

Рис. 30: 2

Вычислив наклон мы видим что он близок к -1. Для более лучшего понимания среднего спектра мощности мы должны сгенерировать более длинную выборку

```
seg_length = 64 * 1024
iters = 100
wave = Wave(voss(seg_length * iters))
len(wave)
```

```
6553600
```

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

```
32769
```

```
spectrum.plot_power()
thinkplot.config(xlabel='Частота', xscale='log', yscale='log')
```

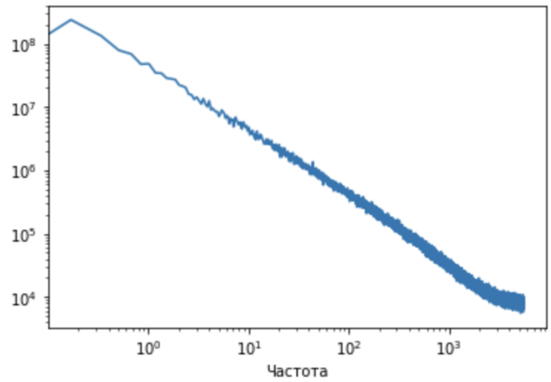

Рис. 31: 2

```
spectrum.estimate_slope().slope
```

```
-1.0017682125992708
```

Рис. 32: 2

Это довольно близко к прямой линии с некоторой кривизной на самых высоких частотах. Наклон близок к -1