Лабораторная работа 4

Луняк Николай

30 марта 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Свойства преобразования Фурье

1.1 Общее

На последней лекции Наталья Владимировна попросила добавить в ближайший отчет описание свойств преобразования Фурье, что и будет сделано в данном разделе.

О том, что такое само по себе преобразование Фурье, мы говорили на лекции, однако в качестве еще одного источника возможной интуиции мне показался примечательным ролик `3Blue1Brown` [3Bl].

Пусть формулы тоже будут под рукой. Для сигнала $f(t)$:

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

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

1.2 Свойства

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

$$Fourier\left(\sum_i \alpha_i \phi_i(t)\right) = \sum_i \alpha_i \cdot Fourier(\phi_i(t)) = \sum_i \alpha_i \Phi_i(\nu)$$

1.2.2 Смещение функции

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

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

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

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

1.2.3 Масштабирование функции

Аналогично, $t' = \alpha t$:

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

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

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

А вот для $a < 0$ мы получим $dt' < 0$ при $dt > 0$. Тогда надо поменять местами пределы интегрирования и выскочит минус в результате:

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

Тогда в одной форме это:

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

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

1.2.4 Перемножение функции

ПФ произведения двух функций - это свертка их ПФ.

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

1.2.5 Свертывание функции

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

1.2.6 Дифференциирование функции

При дифференциировании $\phi(t)$ по t ее 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 органиченной энергией, то есть:

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

И из этого следует, что первое слагаемое должно быть 0.

1.2.7 Интегрирование функции

Как можно предположить, тут произойдет обратное - деление на эту величину. Правда, теперь мы еще требуем, чтобы в функции не было константной составляющей (при дифференциировании она бы пропала, а вот при интегрировании - нет).

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

А 0 там появился из-за того, что $\int_{-\infty}^{\infty}\phi(t')dt' = 0$.

1.2.8 Обратимость

Ну и само собой то, что прямой и обратное преобразование Фурье - два брата-близнеца с точностью до знака.

Глава 2

Пробуем примеры из «A Soft Murmur»

Скачиваем с сайта три файла и смотрим на их спектры.

```
from thinkdsp import Signal, Sinusoid, SquareSignal,
    TriangleSignal, SawtoothSignal, ParabolicSignal
from thinkdsp import normalize, unbias, PI2, decorate
from thinkdsp import Chirp
from thinkdsp import read_wave

import numpy as np

from matplotlib import pyplot

files = [
    'Sounds/169289__qubodup__gong-bell-monkay-s-singing-bowl-
    modified.wav',
    'Sounds/22604__martypinso__dmp010037-crickets-texas.wav',
    'Sounds/136977__audionautics__crowd-long.wav',
]

def analyze(file):
    print(f'Analyzing "{file}"')
    wave = read_wave(file)
    segment = wave.segment(start=0, duration=5)
    pyplot.title('Waveform')
    segment.plot()
    pyplot.show()

    spectrum = segment.make_spectrum()

    pyplot.title('Spectrum (Linear Scale)')
    spectrum.plot_power()
    decorate(
```

```
29          xlabel='Frequency (Hz)',
30          ylabel='Power',
31      )
32      pyplot.show()
33
34      pyplot.title('Spectrum (Log-Log Scale)')
35      spectrum.plot_power(linewidth=0.5)
36      loglog = dict(xscale='log', yscale='log')
37      decorate(
38          xlabel='Frequency (Hz)',
39          ylabel='Power',
40          **loglog
41      )
42      pyplot.show()
43
44      pyplot.title('Spectrogram')
45      segment.make_spectrogram(512).plot(high=5000)
46      pyplot.show()
47
48      return segment.make_audio()
49
50  [analyze(it) for it in files]
```

Листинг 2.1: Анализ файлов

В изображениях ниже присутствует опечатка: те, что подписаны как «Amplitude», являются так же квадратом амплитуды, просто они отражены не в «log-log scale», а те, что подписаны как «Power», являются квадратом амплитуды, отраженном как раз в логарифмическом масштабе. Ввиду некомфортности работы с изображниями в LaTeX, я решил их не исправлять.

Начнем с гонга.

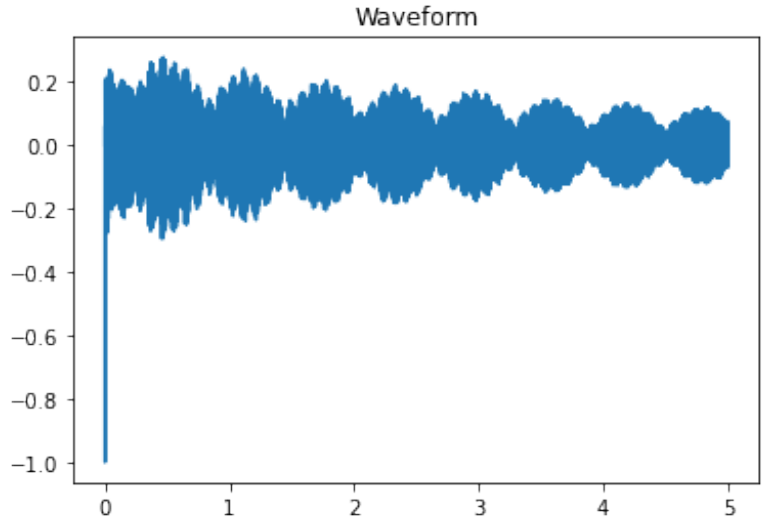

Рис. 2.1: Гонг

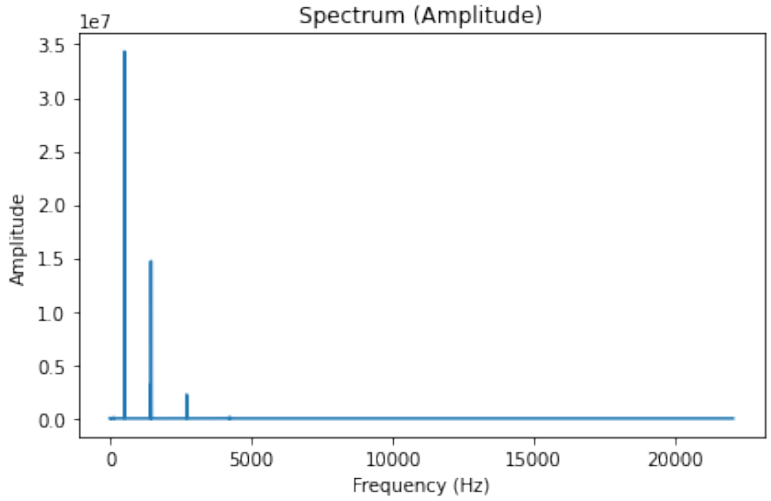

Рис. 2.2: Спектр гонга (линейный масштаб)

Большие значения аплитуды соответствуют малым значениям частоты, что дает основания полагать, что это «красный» bkb «розовый» шум, хотя наличие всего лишь трех «пиков» можно с натяжкой считать настоящим шумом, но такой уж файл скачался...

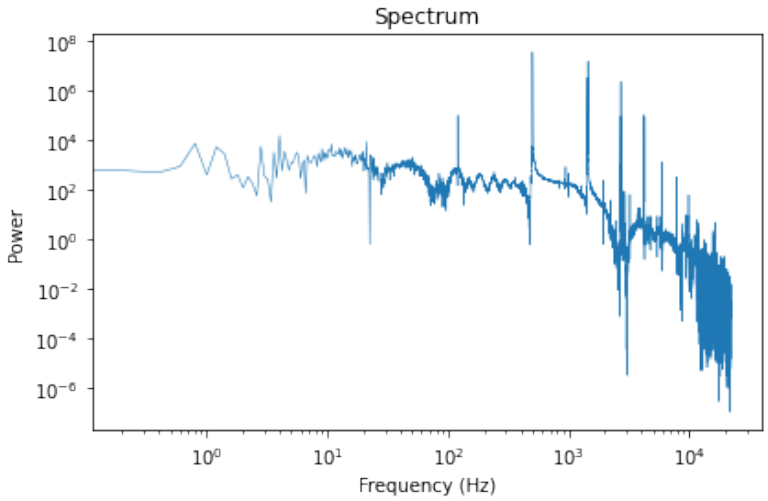

Рис. 2.3: Спектр гонга (логарифмический масштаб)

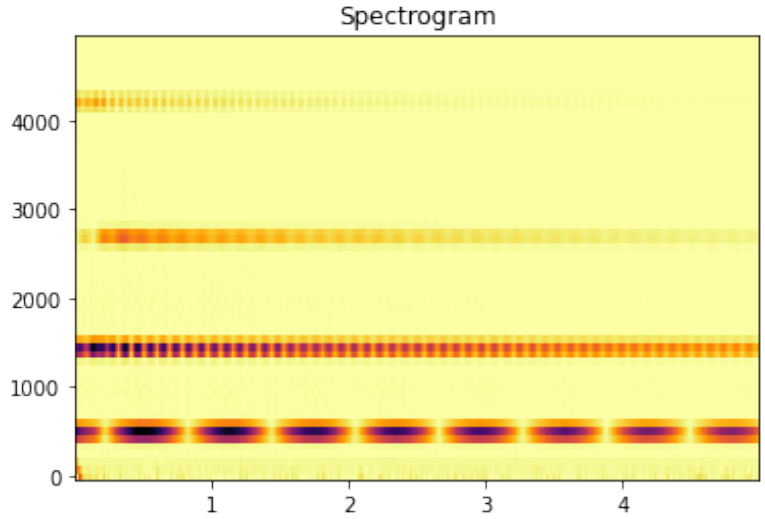

Рис. 2.4: Спектрограмма гонга

В спектрограмме виднеется определенный периодический «шаблон», а также присутствуют лишь определенные частоты, которые мы видели ранее, что четко дает понять, что мы работаем не с шумом, а со сигналом, обладающим стабильным четким «временным профилем».

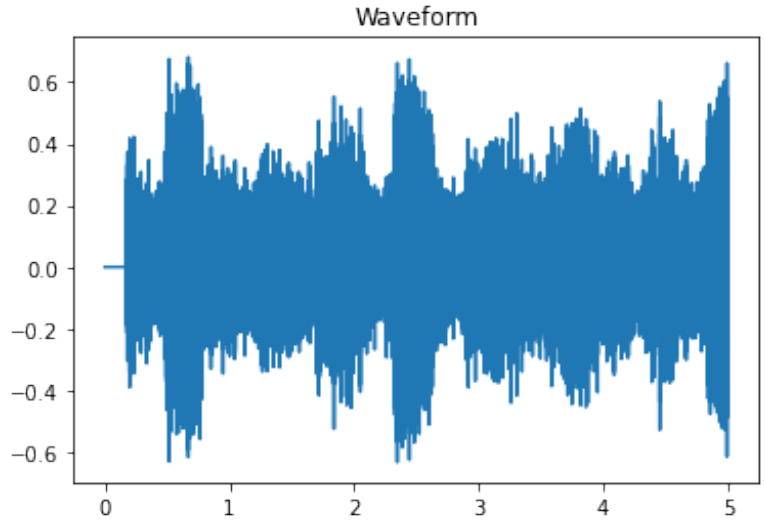

Рис. 2.5: Сверчки

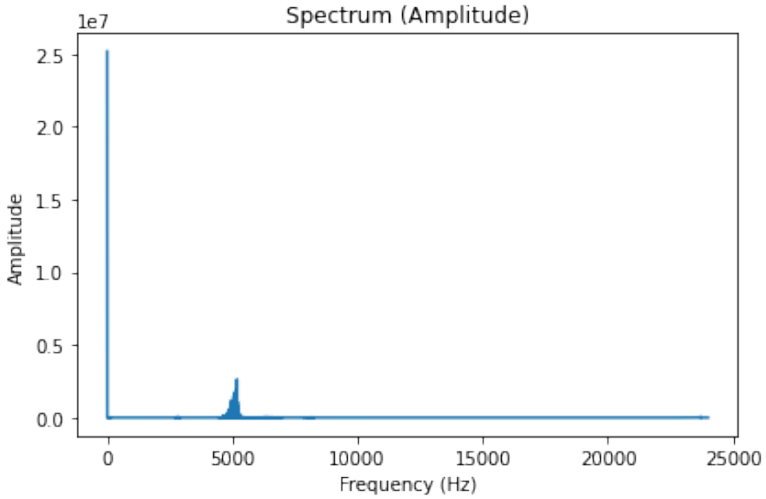

Рис. 2.6: Спектр сверчков (линейный масштаб)

Приблизим этот фрагмент, отрезав начало:

```
wave = read_wave('Sounds/22604__martypinso__dmp010037-
    crickets-texas.wav')
spectrum = wave.segment(start=0, duration=5).make_spectrum()
spectrum.high_pass(100)
spectrum.plot()
```

Листинг 2.2: Смотрим ближе

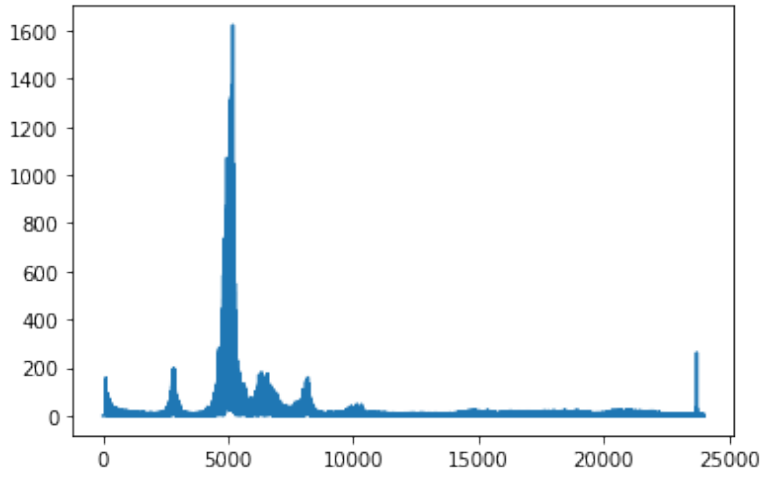

Рис. 2.7: Смотрим ближе

Тут действительно присутствуют много разных частот, однако это не похоже ни на «розовый» шум, ни на «белый». Это так же не походит и на Броуновский шум. Однако, этот «колокол» кое-на-что похож. Проведем эксперимент.

```
import thinkstats2
thinkstats2.NormalProbabilityPlot(spectrum.real)
pyplot.figure()
thinkstats2.NormalProbabilityPlot(spectrum.imag)
```

Листинг 2.3: Проверяем предположение

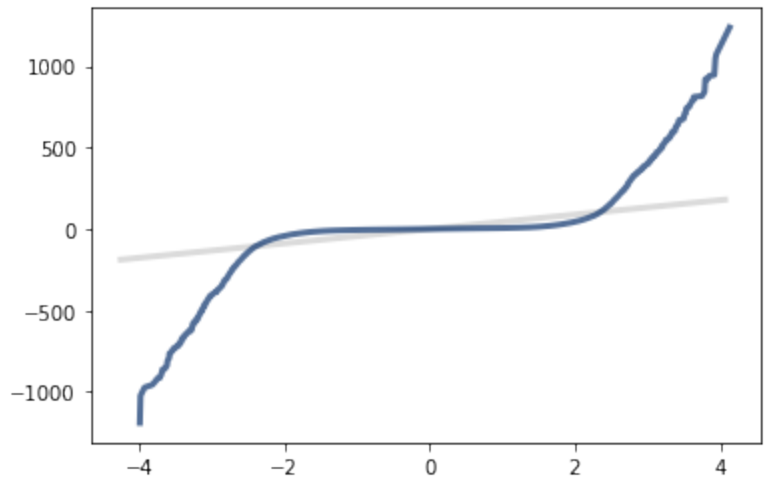

Рис. 2.8: Вещественная часть

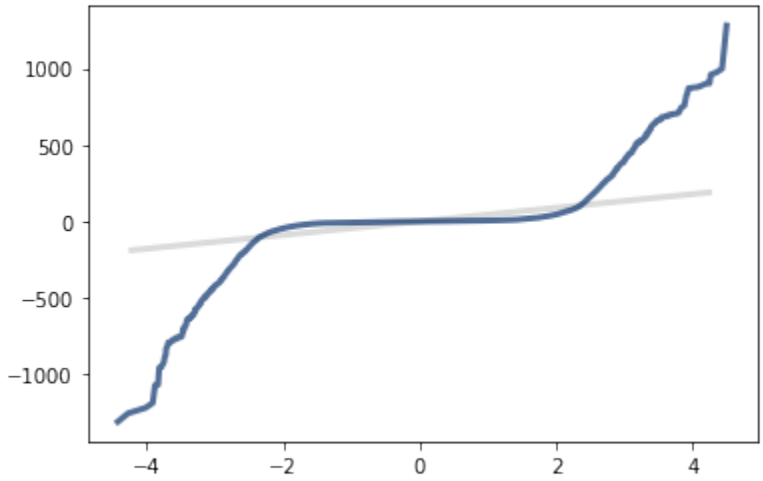

Рис. 2.9: Мнимая часть

Тут как будто бы есть 3 линейных сегмента. Возможно, тут мы имеем дело не с одним нормальным распределением, а несколькими, наложенными друг на дружку.

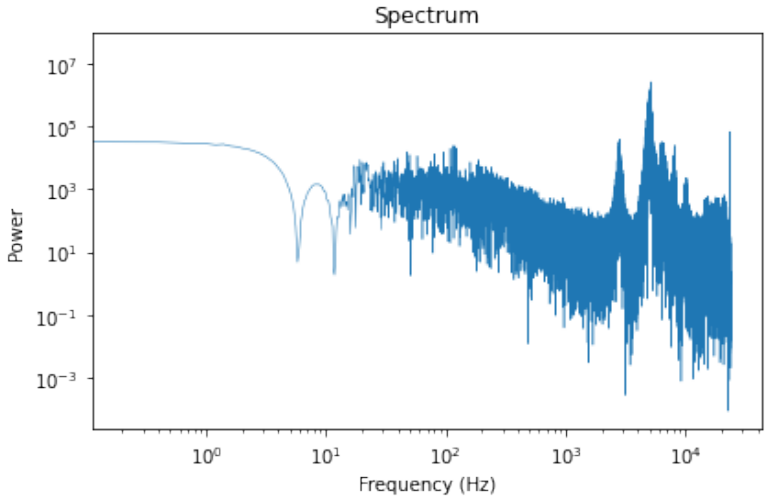

Рис. 2.10: Спектр сверчков (логарифмический масштаб)

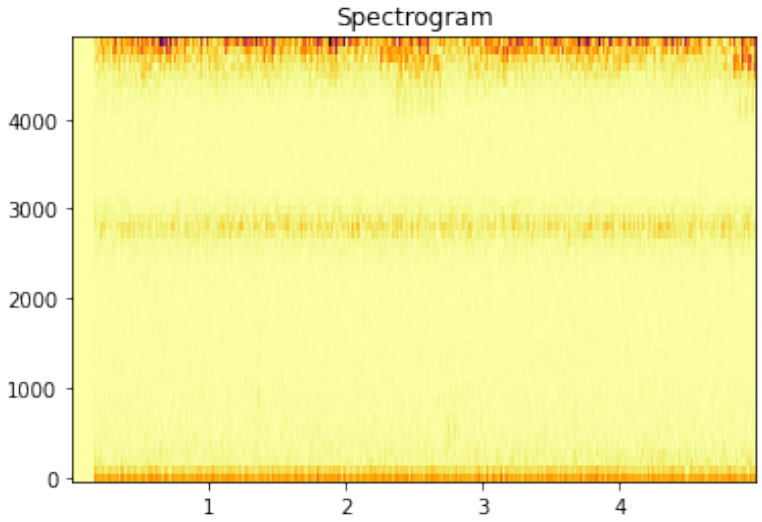

Рис. 2.11: Спектрограмма сверчков

Динамика этого сигнала уже позволяет его назвать шумом, просто он имеет «предпочтительную» частоты.

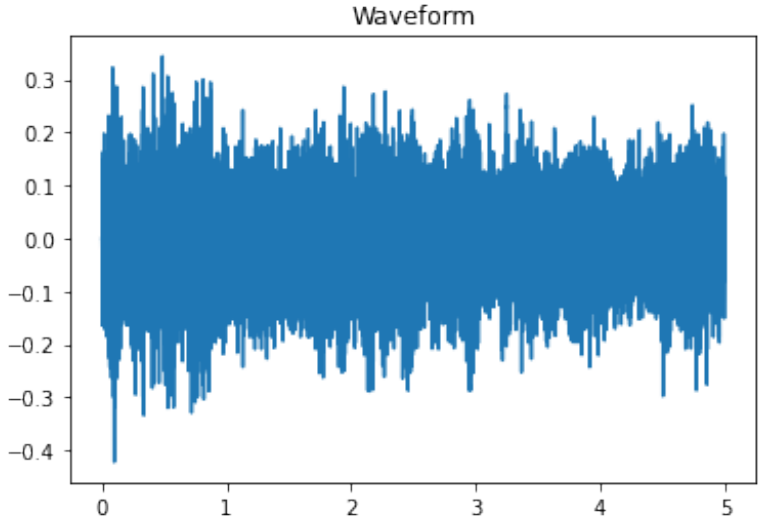

Рис. 2.12: Толпа

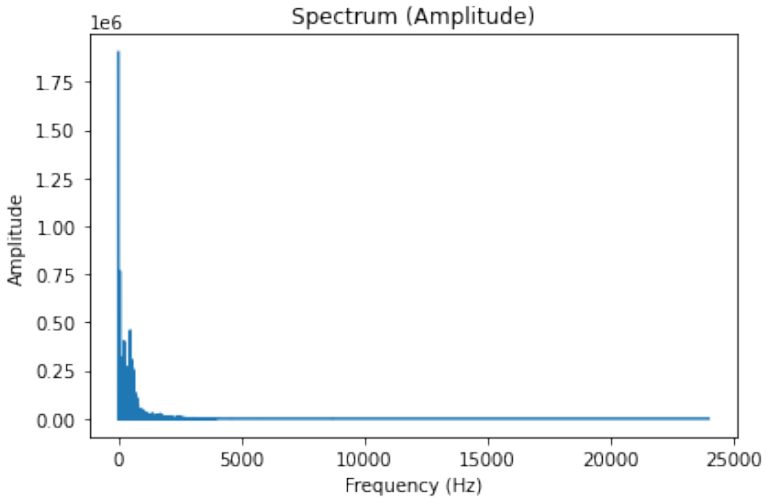

Рис. 2.13: Спектр толпы (линейный масштаб)

Вот это уже походит на «розовый» шум!

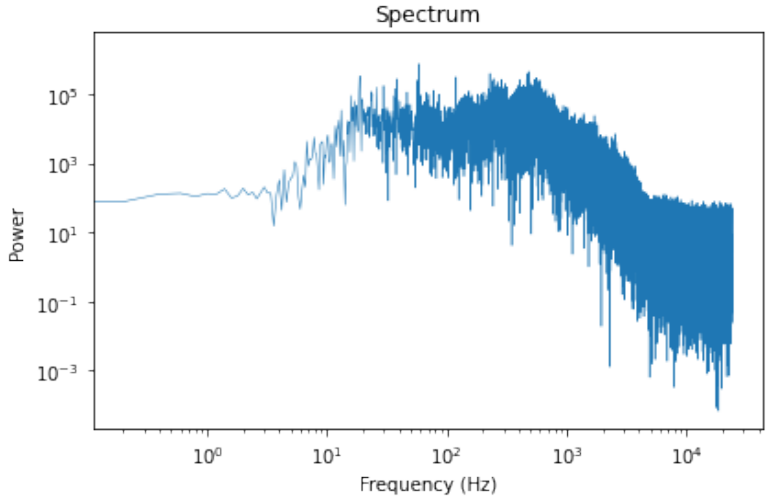

Рис. 2.14: Спектр толпы (логарифмический масштаб)

Однако данная спектрограмма в логарифмическом масштабе все же дает понять, что перед нами не он.

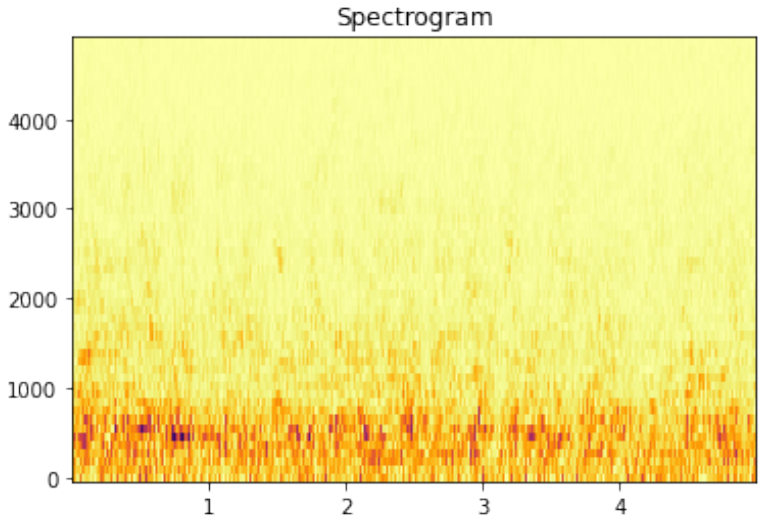

Рис. 2.15: Спектрограмма толпы

Поведение сигнала во времени не оставляет сомнений в том, что

Глава 3

Метод Барлетта

Разобъем наш сигнал на отдельные участки, в пределах которых будем брать спектры.

```
import numpy

wave = read_wave('Sounds/136977__audionautics__crowd-long.wav')

TIME_INTERVAL = 6 * 60
DURATION = 0.01
SAMPLES_COUNT = int(TIME_INTERVAL / DURATION)

summed = numpy.zeros((int(24001 * DURATION + 1)))

for it in range(SAMPLES_COUNT):
    spectrum = wave.segment(start=it * DURATION, duration=DURATION).make_spectrum()
    summed += spectrum.power

result = numpy.sqrt(summed / SAMPLES_COUNT)

pyplot.figure()
pyplot.plot(spectrum.fs, result)
pyplot.show()

average_spectrum = Spectrum(result, spectrum.fs, wave.framerate)

loglog = dict(xscale='log', yscale='log')

pyplot.figure()
average_spectrum.plot_power()
decorate(xlabel='Frequency (Hz)',
         ylabel='Power',
```

```
29              **loglog)
```

Листинг 3.1: Барлетт

В коде выше можно видеть, как длина временного участка накладывает ограничение на диапазон частот в соответствии с тем, как мы это обсуждали в предыдущей лекции.

Код выше почти эквивалентен коду, предложенному в `chap04soln.ipynb` с `seg_length` 512, но работающий чуть побыстрее.

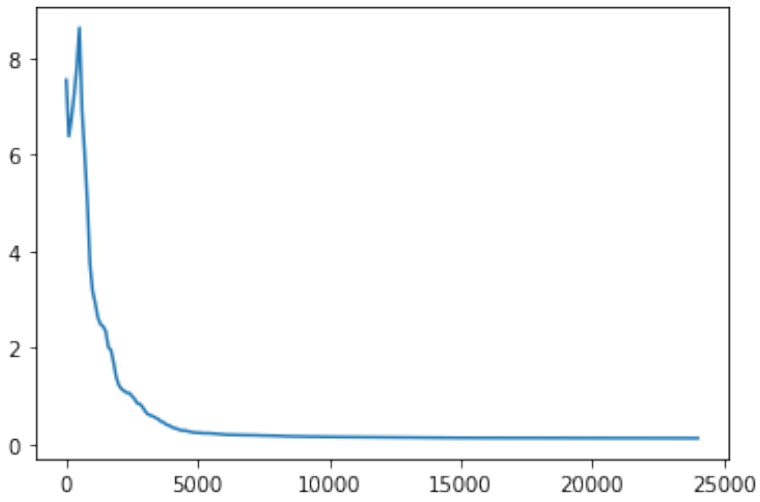

Рис. 3.1: Линейный масштаб

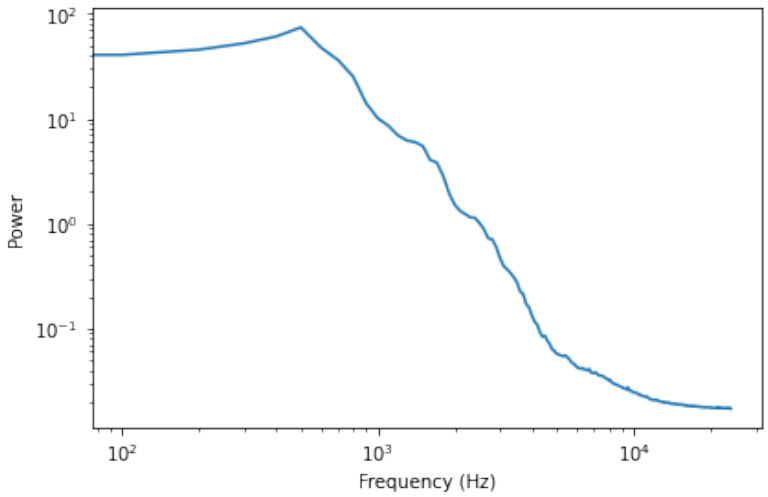

Рис. 3.2: Логарифмический масштаб

Видно, что график не похож на горизонтальную прямую или на какую-либо линейную зависимость, при этом сигнал обладает определенной стабильностью во времени (разные частоты неравнозначны, и это сохраняется на протяжении всего времени сигнала).

Глава 4

BitCoin

```
import pandas
data = pandas.read_csv('Data/BTC_USD_2020-12-31_2021-03-30-CoinDesk.csv')
data
```

Листинг 4.1: Загрузка `DataFrame`

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2020-12-31	28768.836208	27349.327233	28928.214391	27349.283204
1	BTC	2021-01-01	29111.521567	28872.829775	29280.045328	27916.625059
2	BTC	2021-01-02	29333.605121	28935.810981	29601.594898	28753.412314
3	BTC	2021-01-03	32154.167363	29353.640608	33064.673534	29012.927887
4	BTC	2021-01-04	33002.536427	32074.106611	34452.080337	31885.581619
...	...	...	...	...	...	...
85	BTC	2021-03-26	52173.867980	52335.565034	53209.406384	50458.099965
86	BTC	2021-03-27	54483.045732	51344.048980	54806.881514	51286.291177
87	BTC	2021-03-28	56234.356105	55067.824399	56520.307287	54022.076941
88	BTC	2021-03-29	55343.925815	55843.748916	56541.006527	54741.626373
89	BTC	2021-03-30	57627.679249	55786.546075	58353.529393	54929.680588

90 rows × 6 columns

Рис. 4.1: `DataFrame`

Я скачал данные не за весь период, а только за последний год.

```
from thinkdsp import Wave
wave = Wave(data['Closing Price (USD)'], data.index,
    framerate=1)
wave.plot()
```

Листинг 4.2: Визуализация

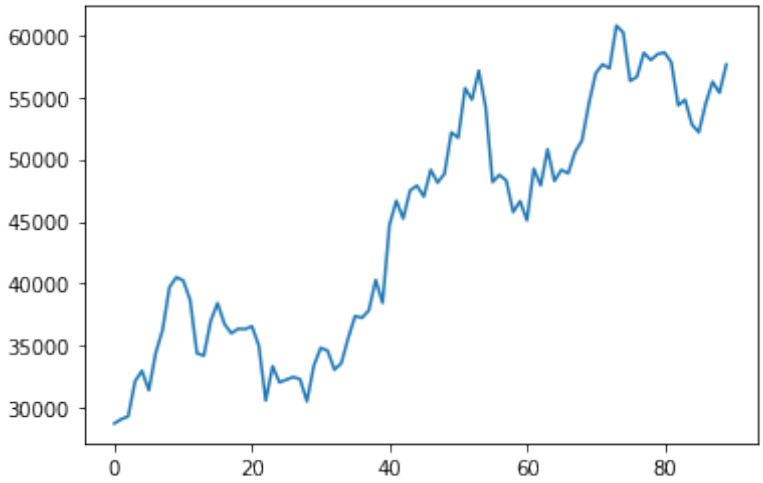

Рис. 4.2: Визуализация

Выше мы превратили данные в сигнал, означающий зависимость цены от номера дня.

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

Листинг 4.3: Спектр

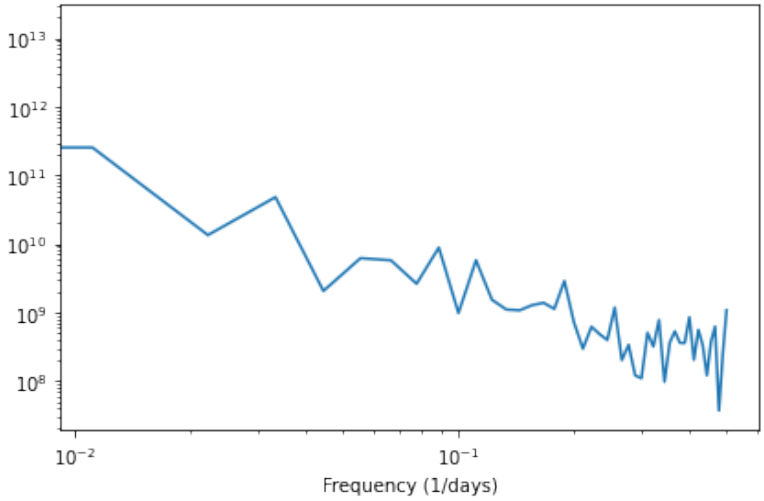

Рис. 4.3: Спектр

Полученный спектр походит на прямую линию, что намекает на воз-

можный «красный» или «розовый» шум. Для более точной оценки можно оценить наклон прямой более формально:

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

Листинг 4.4: Спектр

Для «красного» шума характерен наклон -2, поэтому предположу, что это все-таки «розовый». Физический же смысл данного спектра в том, что частота изменения стоимости обратно пропорциональна изначимости этих изменений. Другими словами, серьезные изменения остаются достаточно редкими в сравнении с ежедневными флуктуациями.

Глава 5

Счетчик Гейгера

Нам необходимо реализовать сигнал в соответствии с распределением Пуассона, которое имеет следующий вид:

$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Парамет λ соответствует мат. ожиданию и вариации случайной величины ($\lambda = E\xi = E(\xi - E\xi)$). В нашем случае логично говорить об ожидаемом числе попавшихся частиц в неком промежутке времени. Или даже лучше сказать, о среднем количестве встреченных частиц на одно изменение (и измерений в секунду у нас `framerate` штук).

```
from thinkdsp import Noise

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

signal = MyUncorrelatedPoissonNoise(amp=0.001)
wave = signal.make_wave(duration=10, framerate=10_000)
wave.make_audio()
```

Листинг 5.1: Реализация

Прослушать это в LaTeXнельзя, но звучит правдоподобно.

Можно посмотреть, сколько на самом деле нам прилетело частиц:

```
sum(wave.ys), 0.001 * 10_000 * 10
```

Листинг 5.2: Подсчитываем частицы

Получаем на выходе `(108, 100.0)`, что означает, что примерно столько и прилетело частиц, сколько мы хотели.

Можно еще посчитать пики на вейвформе.

```
wave.plot()
```

Листинг 5.3: Wave

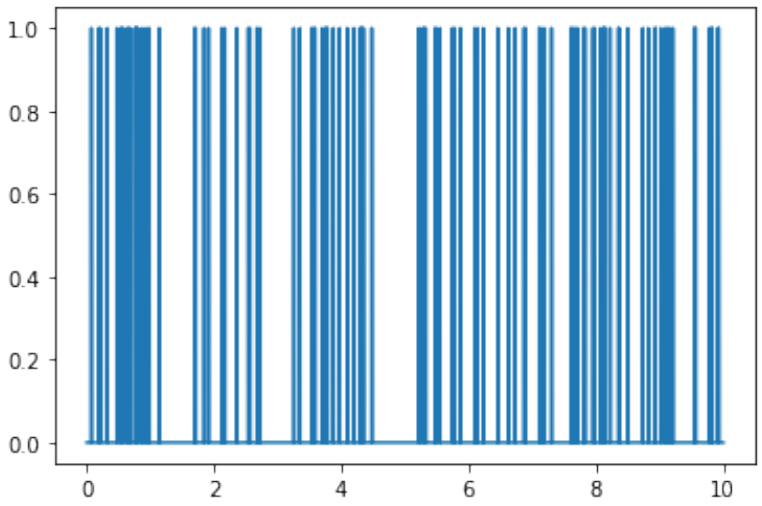

Рис. 5.1: Wave

А вот и спектр:

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

Листинг 5.4: Спектр

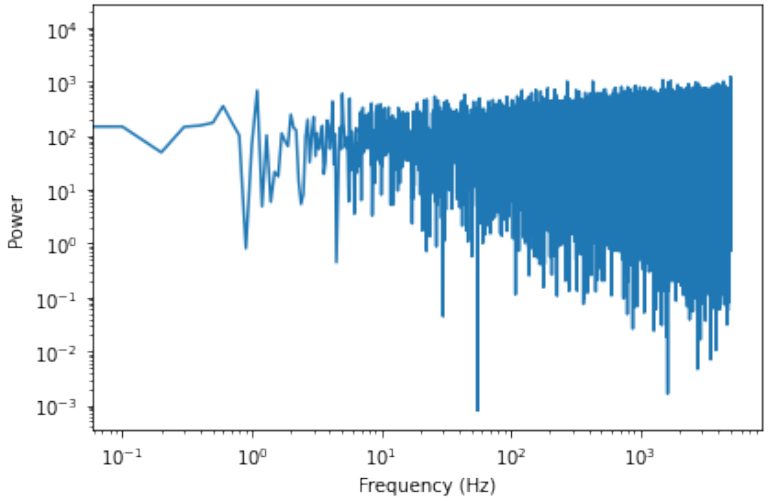

Рис. 5.2: Спектр

Он походит на прямую горизонтальную линию, а оценка его наклона показывает величину, близкую к 0. Похоже, что это «белый» шум.

Глава 6

Алгоритм Восса-МакКартни

Это просто более хороший алгоритм для генерации «розового» шума.

```
def voss(rows, columns=16):
    array = numpy.empty((rows, columns))
    array.fill(numpy.nan)

    array[0, :] = numpy.random.random(columns)
    array[:, 0] = numpy.random.random(rows)

    target_columns = numpy.random.geometric(0.5, rows)
    target_columns[target_columns >= columns] = 0

    target_rows = numpy.random.randint(rows, size=rows)

    array[target_rows, target_columns] = numpy.random.random(
    rows)

    dataframe = pandas.DataFrame(array)
    dataframe.fillna(method='ffill', axis=0, inplace=True)

    total = dataframe.sum(axis=1)
    return total.values

wave = Wave(voss(11025))
wave.unbias()
wave.plot()
wave.make_audio()
```

Листинг 6.1: Алгоритм

Идея алгоритма заключается в сложении нескольких последовательностей случайных чисел, изменяющихся с разной частотой (эти числовые последоваьельности представлены столбцами, вдоль которых потом производится суммирование).

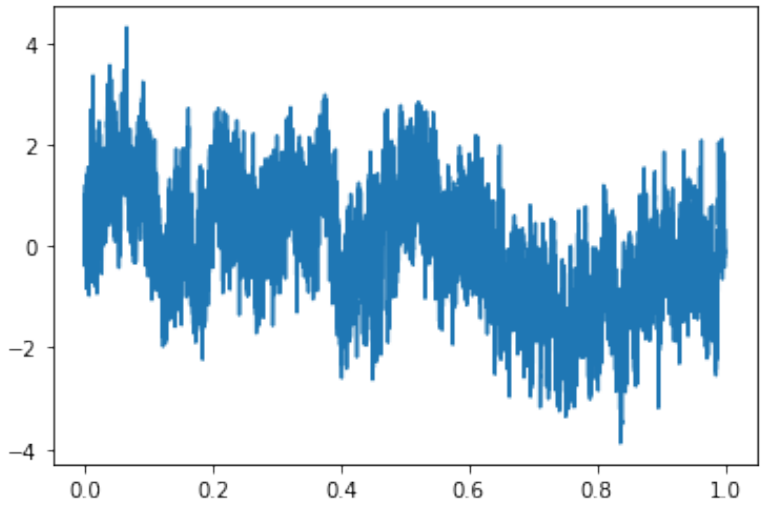

Рис. 6.1: Сигнал

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

Листинг 6.2: Спектр

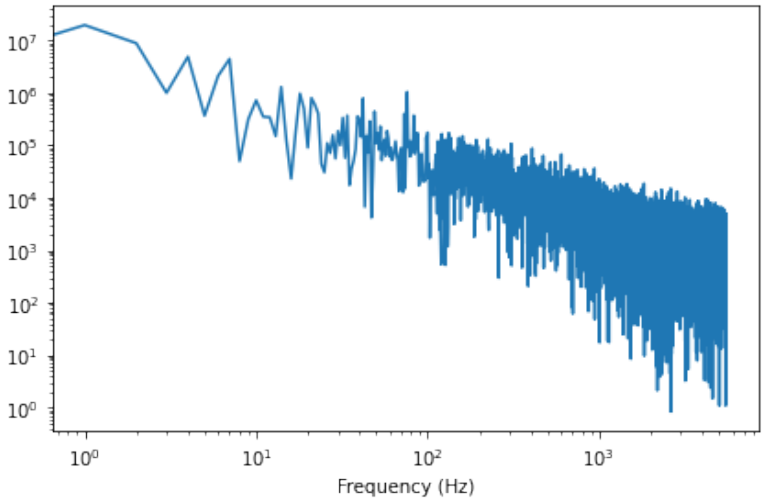

Рис. 6.2: Спектр

Виднеется прямая. Оценка ее наклона дает величину, близкую к -1, что говорит о том, что перед нами точно «розовый» шум.

Список литературы

[3Bl] 3Blue1Brown. *But what is the Fourier Transform? A visual introduction.* URL: https://youtu.be/spUNpyF58BY.