Лабораторная работа №4
Шум

Смирнов Никита

24 апреля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Теоретическая часть

1.1 Общие сведения

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

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

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

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

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

1.2.2 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:

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

1.2.3 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.

1.2.4 Перемножение функции

ПФ произведения двух функций - это свёртка их ПФ.

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

1.2.5 Свёртывание функции

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

1.2.6 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:

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

И из этого следует, что первое слагаемое равно 0.

1.2.7 Интегрирование функции

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

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

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

1.2.8 Обратимость

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

Глава 2

Упражнение 4.1

В этом упражнениии необходимо скачать звук источников шума. Определить похож ли спектр мощности скаченного звука на белый розовый или броуновский шумы.

Я нашел звук обстановки в ресторане.

```
import thinkdsp

wave = thinkdsp.read_wave('matias.wav')
wave.make_audio()
```

Листинг 2.1: Прослушивание скачанного шума

Выбрал отрезок:

```
segment = wave.segment(start=5, duration=1.0)
segment.make_audio()
```

Листинг 2.2: Выбор короткого отрезка

Посмотрим на спектр.

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

Листинг 2.3: Спектр звука

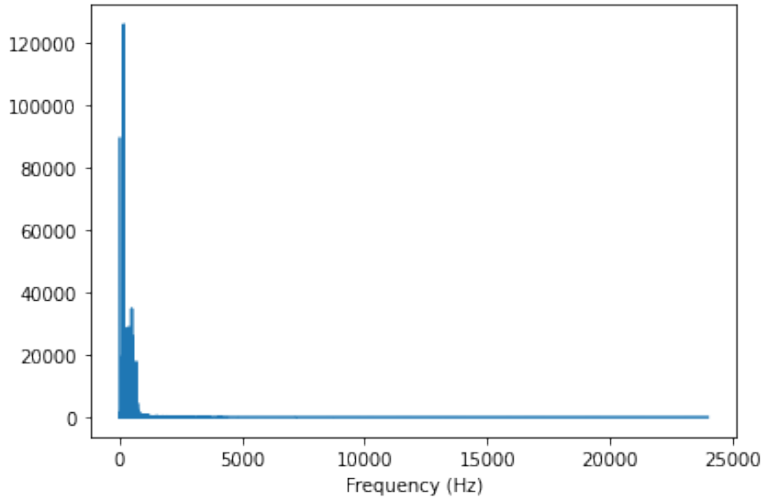

Рис. 2.1: Спектр звука

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

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

Листинг 2.4: Спектр мощности звука

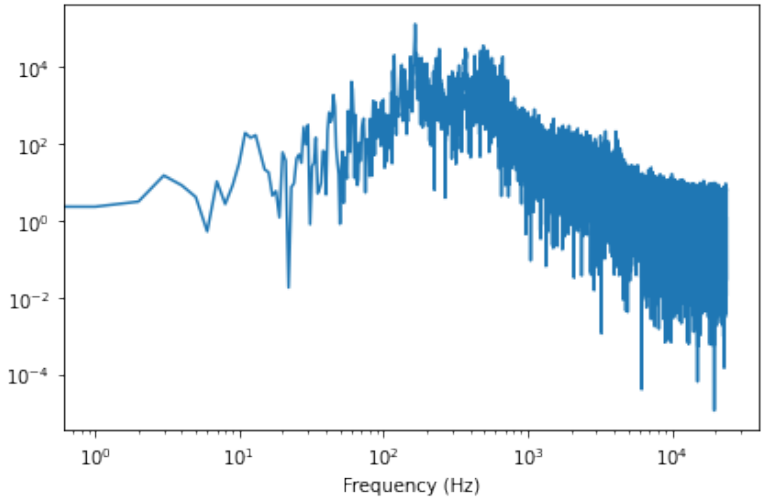

Рис. 2.2: Спектр мощности звука

Эта структура с увеличением, а затем и с уменьшением амплитуды кажется обычным явлением для естественных источников шума. Чтобы увидеть, как спектр меняется с течением времени, я выберу другой сегмент.

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

Листинг 2.5: Выбор другого сегмента звука

Теперь рассмотрим два спектра:

```
spectrum2 = segment2.make_spectrum()
spectrum.plot_power()
spectrum2.plot_power(color='#beaed4')
decorate(xlabel='Frequency (Hz)',
                ylabel='Amplitude')
```

Листинг 2.6: Спектр двух звуков

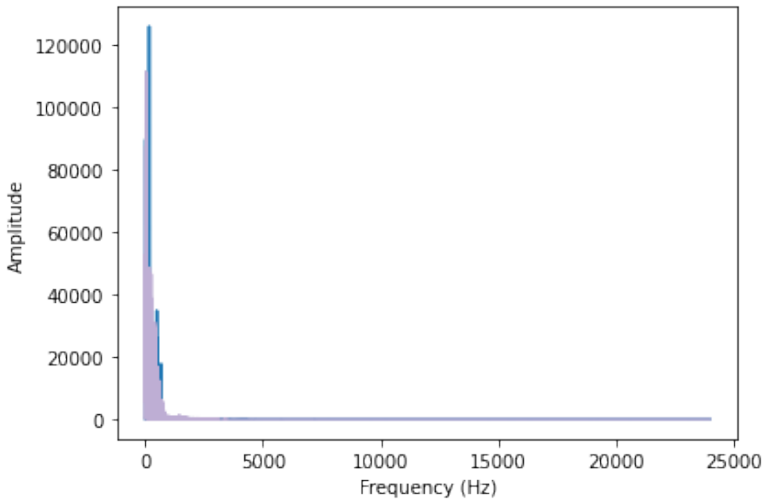

Рис. 2.3: Спектр двух звуков

Теперь рассмотрим график мощности в логарифмическом масштабе.

```
spectrum.plot_power()
spectrum2.plot_power(color='#beaed4')
decorate(xlabel='Frequency (Hz)',
                ylabel='Amplitude',
                xscale='log',
                yscale='log')
```

Листинг 2.7: Спектр мощности двух звуков

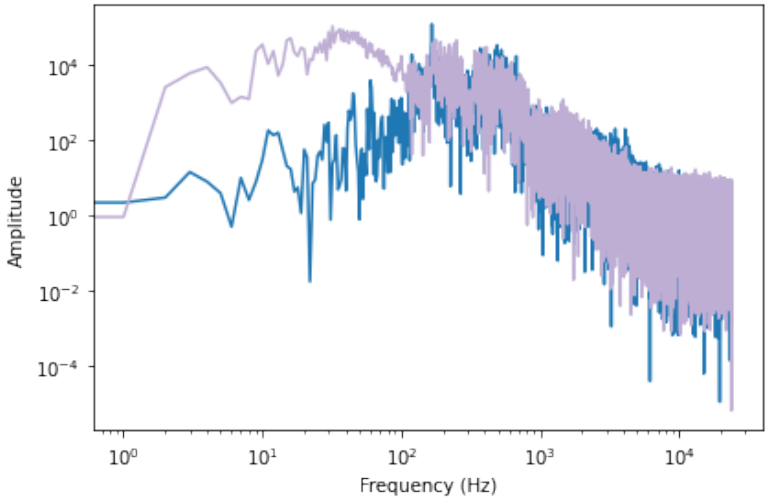

Рис. 2.4: Спектр мощности двух звуков

Таким образом, структура кажется неизменной с течением времени. Мы также можем посмотреть на спектрограмму:

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

Листинг 2.8: Спектрограмма звука

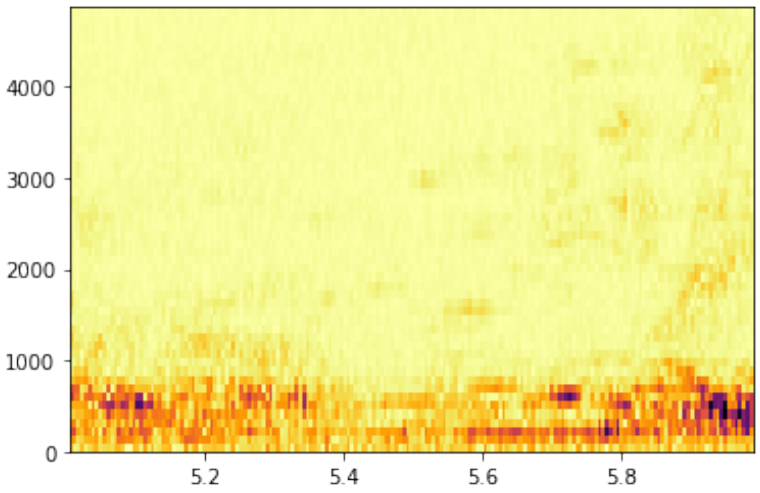

Рис. 2.5: Спектрограмма звука

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

Глава 3

Упражнение 4.2

`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.

```
def bartlett_method(wave, length=512, flag=True):
    spectro = wave.make_spectrogram(length, 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 = thinkdsp.Spectrum(hs, fs, wave.framerate)
    return spectrum
```

Листинг 3.1: Функция bartlett_method

Построим сегменты:

```
psd = bartlett_method(segment)
psd2 = bartlett_method(segment2)

psd.plot_power()
psd2.plot_power(color='#beaed4')

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

Листинг 3.2: Сегменты звуков

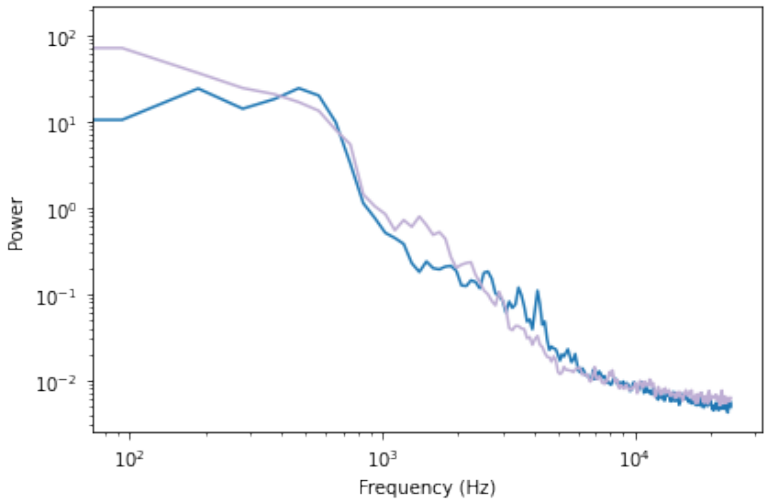

Рис. 3.1: Сегменты звуков

Теперь мы можем более чётко увидеть взаимосвязь между мощностью и частотой. Это не простая линейная зависимость, но она одинакова для разных сегментов, таких как около 1000 Гц, 6000 Гц и выше 10000 Гц.

Глава 4

Упражнение 4.3

На предложенной веб-странице я скачал данные о ежедневной цене BitCoin в течение года.

```
data = pd.read_csv('BTC_USD_2020-04-08_2021-04-07-CoinDesk.csv')
data
```

Листинг 4.1: Таблица данных

	Currency	Date	Closing Price (USD)	24h Open (USD)	24h High (USD)	24h Low (USD)
0	BTC	2020-04-26	7542.303401	7497.859881	7702.018980	7445.685898
1	BTC	2020-04-27	7624.853786	7542.303101	7707.594271	7503.770143
2	BTC	2020-04-28	7776.507543	7624.854338	7798.276656	7621.487082
3	BTC	2020-04-29	7761.758784	7788.574229	7793.636018	7677.178774
4	BTC	2020-04-30	8773.106488	7761.758619	8973.079277	7725.542654
...	...	...	...	...	...	...
360	BTC	2021-04-21	56608.769748	55723.227263	57121.943902	53442.851660
361	BTC	2021-04-22	54144.427476	56508.151244	56809.146393	53913.850515
362	BTC	2021-04-23	51965.059559	53830.823864	55471.076372	50500.731862
363	BTC	2021-04-24	50669.144382	51714.073970	52111.185068	47467.912032
364	BTC	2021-04-25	50733.769504	51217.172330	51253.442948	48932.158814

365 rows × 6 columns

Рис. 4.1: Таблица данных

Визуализируем скачанные данные.

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

Листинг 4.2: Визуализация данных

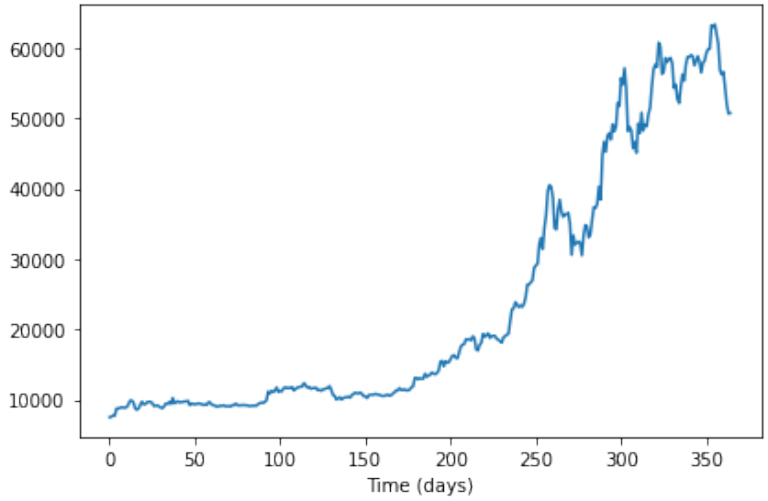

Рис. 4.2: Визуализация данных

Построим спектр искусственно созданного звука, где частотой будет выступать 1/дни.

```
spectrum = wave.make_spectrum()
spectrum.plot_power()
decorate(xlabel='Frequency (1/days)',
                 xscale='log', yscale='log')
```

Листинг 4.3: Спектр искусственного звука

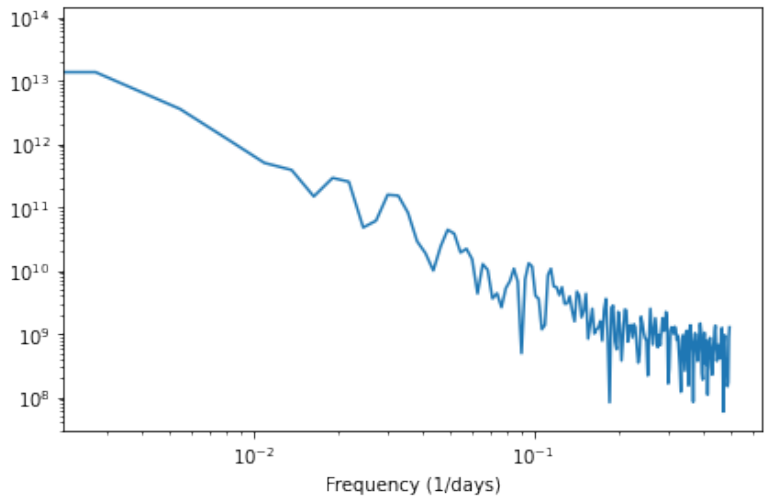

Рис. 4.3: Спектр искусственного звука

Спектр сход с прямой линией, поэтому можно предположить, что это "красный"или "розовый"шум. Проверим это, узнав наклон прямой.

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

Листинг 4.4: Наклон прямой

Наклон составляет `-1.8582636508741062`, что похоже на красный шум (который должен иметь наклон -2).

Глава 5

Упражнение 4.4

Созданный класс `UncorrelatedPoissonNoise` представляет некоррелированный пуассоновский шум. Оценивает сигнал в заданное время.

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

Листинг 5.1: Созданный класс UncorrelatedPoissonNoise

Рассмотрим как это звучит при низких уровнях «радиации».

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

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

Листинг 5.2: Создание звука

Звук действительно похож на звуки от счётчика Гейгера, будто бы находишься где-то в Припяти. Чтобы убедиться, что все работает, мы сравниваем ожидаемое количество частиц и фактическое количество:

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

Листинг 5.3: Создание звука

Количество частиц в обоих случаях равняется 10. Теперь рассмотрим

полученный звук:

```
wave.plot()
```

Листинг 5.4: Визуализация звука

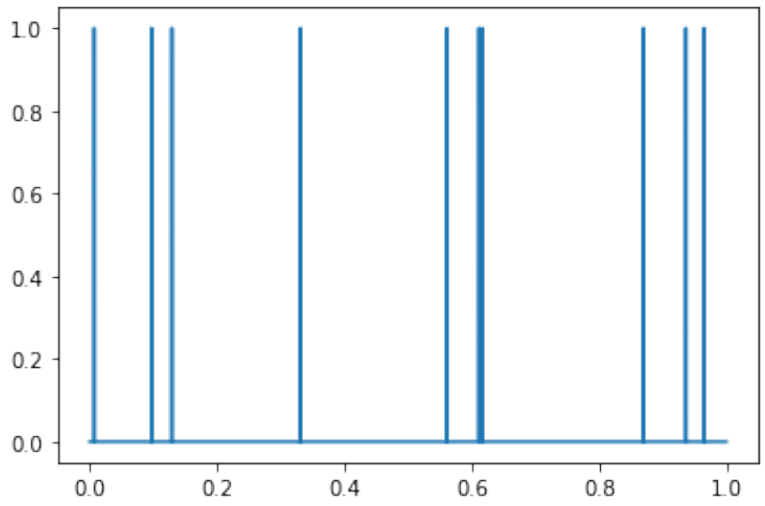

Рис. 5.1: Визуализация звука

Рассмотрим спектр мощности в логарифмическом масштабе:

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

Листинг 5.5: Спектр мощности звука

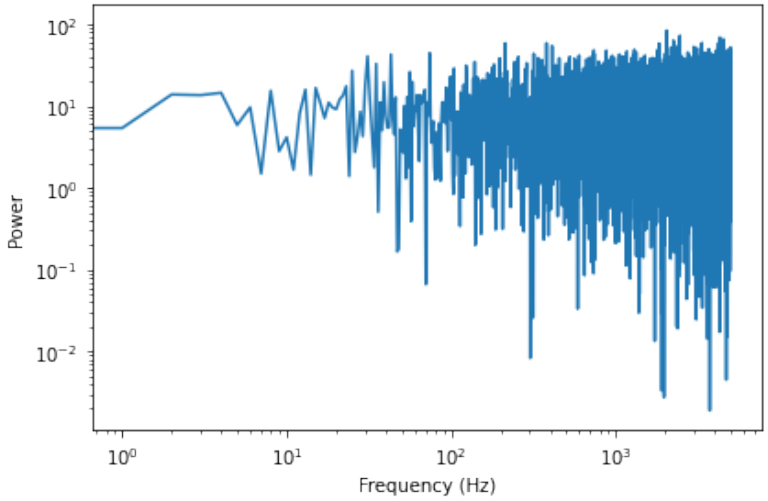

Рис. 5.2: Спектр мощности звука

Рассмотрим наклон:

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

Листинг 5.6: Наклон прямой

Похоже на белый шум, а крутизна близка к 0 (`-0.004513262607615736`).

При более высокой скорости поступления это больше похоже на белый шум:

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

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

Листинг 5.7: Создание нового звука

Построим его график.

```
wave.plot()
```

Листинг 5.8: Визуализация нового звука

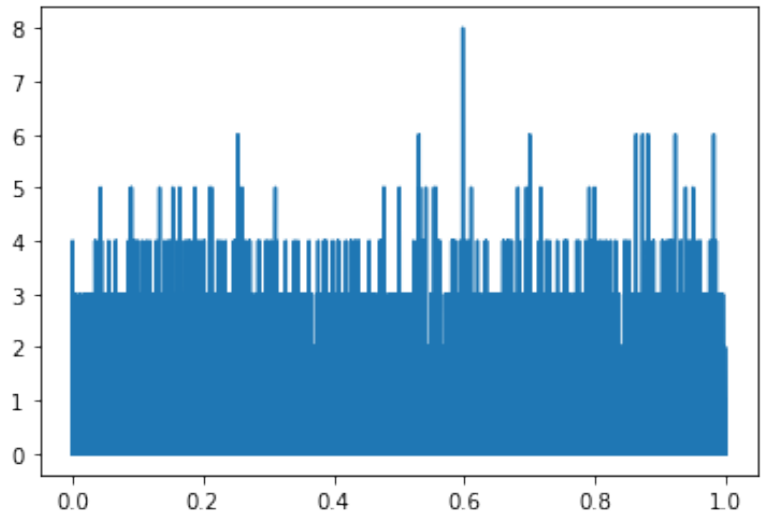

Рис. 5.3: Визуализация нового звука

И спектр сходится на гауссовском шуме.

```
spectrum = wave.make_spectrum()
spectrum.hs[0] = 0

thinkplot.preplot(2, cols=2)
thinkstats2.NormalProbabilityPlot(spectrum.real, label='real')
thinkplot.config(xlabel='Normal sample',
                 ylabel='Power',
                 legend=True,
                 loc='lower right')

thinkplot.subplot(2)
thinkstats2.NormalProbabilityPlot(spectrum.imag, label='imag')
thinkplot.config(xlabel='Normal sample',
                     loc='lower right')
```

Листинг 5.9: Сравнение спектров

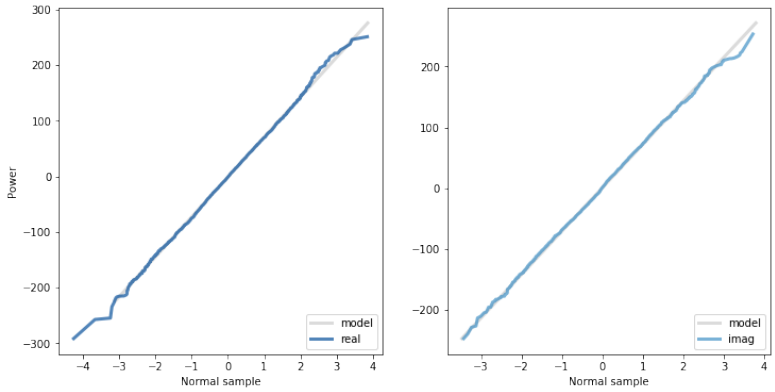

Рис. 5.4: Сравнение спектров

Глава 6

Упражнение 4.5

Вот весь процесс в функции: Создает розовый шум с помощью алгоритма Восса-Маккартни.

```
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)

    # the total number of changes is 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
```

Листинг 6.1: Создание функции

Чтобы проверить это, я сгенерирую 12005 значений:

```
ys = voss(12005)
ys
```

Листинг 6.2: Генерация значений

Теперь создадим из них звук:

```
wave = thinkdsp.Wave(ys)
wave.unbias()
wave.normalize()
```

Листинг 6.3: Создание звука

Теперь посмотрим на его визуализацию.

```
wave.plot()
```

Листинг 6.4: Визуализация звука

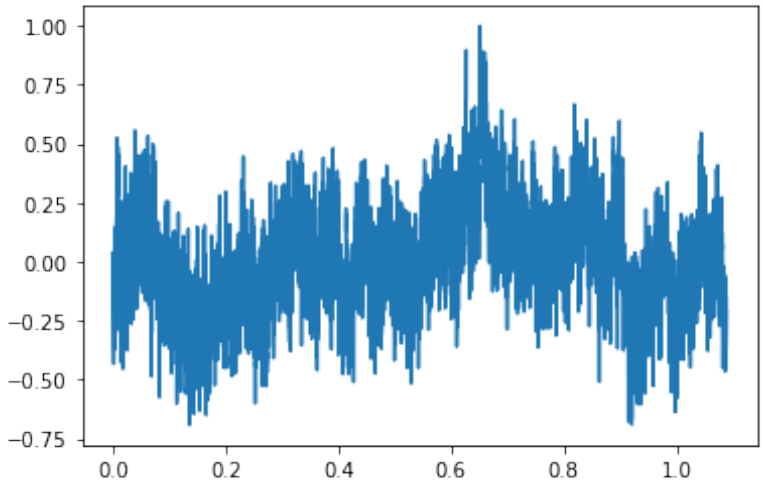

Рис. 6.1: Визуализация звука

Как и ожидалось, это больше похоже на случайное блуждание, чем на белый шум, но более случайное, чем на красный шум. Теперь рассмотрим спектр мощности:

```
spectrum = wave.make_spectrum()
spectrum.hs[0] = 0
spectrum.plot_power()
decorate(xlabel='Frequency (Hz)',
                xscale='log',
                yscale='log')
```

Листинг 6.5: Спектр мощности звука

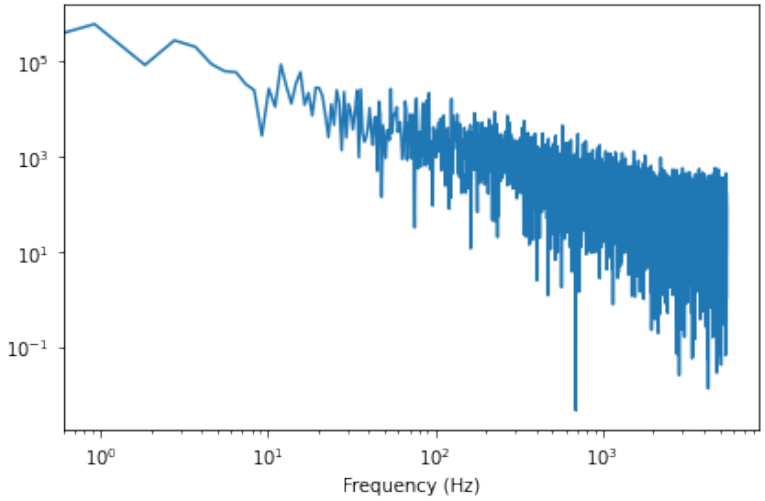

Рис. 6.2: Спектр мощности звука

Посмотрим на наклон:

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

Листинг 6.6: Наклон прямой

Расчетный наклон близок к -1 (`-1.0129573459835064`).

Мы можем лучше понять средний спектр мощности, сгенерировав более длинную выборку:

```
seg_length = 40 * 124
iters = 100
wave = thinkdsp.Wave(voss(seg_length * iters))
len(wave)
```

Листинг 6.7: Генерация более длинной выборки

И используя метод Барлетта для вычисления среднего.

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

Листинг 6.8: Использование метода Барлетта

Это довольно близко к прямой линии с некоторой кривизной на самых высоких частотах, если рассматривать спектр мощности звука.

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

```
3                 xscale='log',
4                 yscale='log')
```

Листинг 6.9: Спектр мощности звука

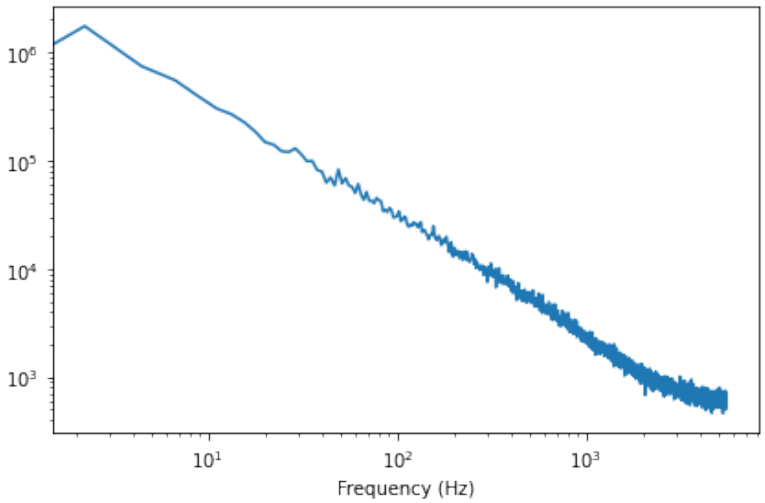

Рис. 6.3: Спектр мощности звука

Посмотрим на наклон:

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

Листинг 6.10: Наклон прямой

Наклон теперь более близок к -1 (`-1.0048368551745679`).

Глава 7

Выводы

Во время выполнения лабораторной работы получены навыки работы с различными видами шумов. Также получены навыки создания этих шумов через различные данные.