Лабораторная работа № 5. Автокорреляция.

3530901/80201, Шелаев Н. Р.

30 мая 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Корреляция

Посмотрим на значение корреляции между двумя одинаковыми сигналами, сдвинутыми по фазе.

```
        from thinkdsp import SinSignal

        def make_sine(offset):
            signal = SinSignal(freq = 440, offset = offset)
            wave = signal.make_wave(duration=0.5, framerate
    =10000)
            return wave

        def compute_corr(offset):
            wave1 = make_sine(offset = 0)
            wave2 = make_sine(offset = -offset)
            wave1.segment(duration = 0.01).plot()
            wave2.segment(duration = 0.01).plot()
            corr = wave1.corr(wave2)
            print('corr =', corr)
            decorate(xlabel='Time (s)')

        PI2 = np.pi * 2
        slider = widgets.FloatSlider(min = 0, max = PI2,
    value = 1)
        interact(compute_corr, offset = slider);

```

Листинг 1.1: Два сдвинутые по фазе сигнала

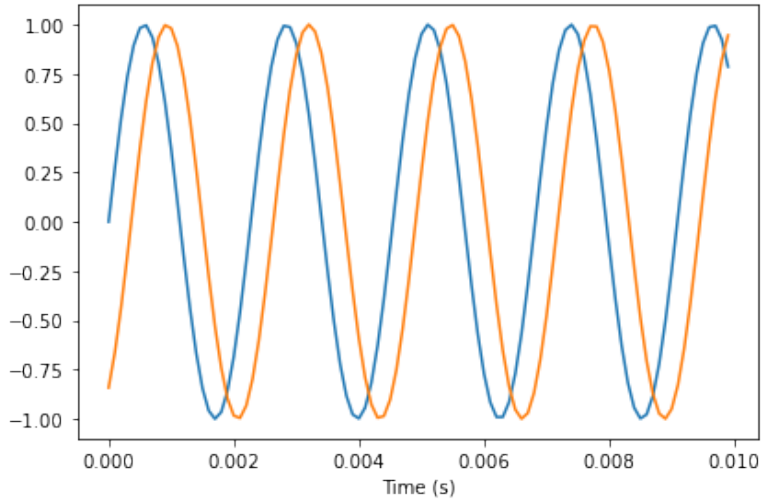

Рис. 1.1: Изменение фазы между двумя одинаковыми сигналами

```
        offsets = np.linspace(0, PI2, 101)
        corrs = []
        for offset in offsets:
            wave2 = make_sine(offset)
            corr = np.corrcoef(wave1.ys, wave2.ys)[0, 1]
            corrs.append(corr)

        plt.plot(offsets, corrs)

```

Листинг 1.2: Корреляция между сигналами в зависимости от фазы

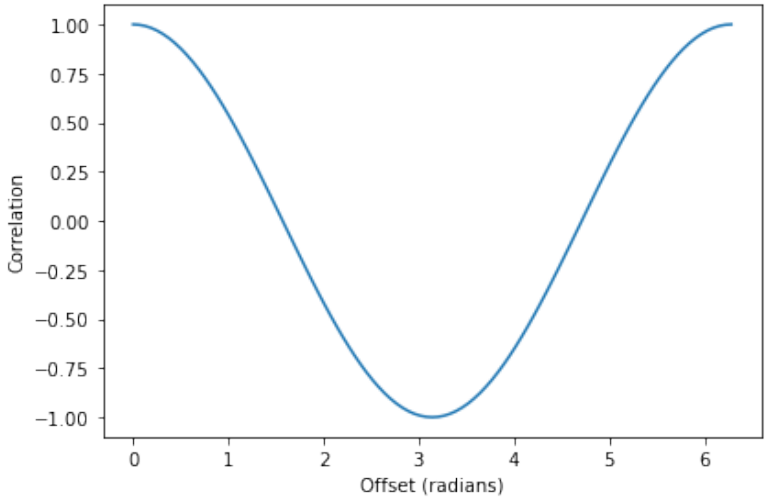

Рис. 1.2: И получили ... косинусоиду

Корреляция между сигналами в одной фазе равна 1, в противофазе она становится равна -1.

Глава 2

Последовательная корреляция

Построим последовательную корреляцию для различных видов шума.

```
        from thinkdsp import UncorrelatedGaussianNoise

        def serial_corr(wave, lag = 1):
                N = len(wave)
                y1 = wave.ys[lag:]
                y2 = wave.ys[:N-lag]
                corr = np.corrcoef(y1, y2)[0, 1]
            return corr

        signal = UncorrelatedGaussianNoise()
        wave = signal.make_wave(duration=0.5, framerate
    =11025)
        serial_corr(wave)

```

Листинг 2.1: Некоррелированный шум

Значение корреляции близко к 0 (кто бы мог подумать?).

```
        from thinkdsp import BrownianNoise

        signal = BrownianNoise()
        wave = signal.make_wave(duration=0.5, framerate
    =11025)
        serial_corr(wave)

```

Листинг 2.2: Броуновский шум

Значение корреляции близко к 1 (тоже ничего удивительного).

```
        from thinkdsp import PinkNoise

        signal = PinkNoise(beta = 1)
```

```
4           wave = signal.make_wave(duration=0.5, framerate
    =11025)
5           serial_corr(wave)
6
```

Листинг 2.3: Розовый шум

В этот раз значение корреляции близко к 0,8.

```
1           np.random.seed(25)
2           betas = np.linspace(0, 2, 21)
3           corrs = []
4
5           for beta in betas:
6               signal =  PinkNoise(beta = beta)
7               wave = signal.make_wave(duration=1.0, framerate
    =11025)
8               corr = serial_corr(wave)
9               corrs.append(corr)
10
11          plt.plot(betas, corrs)
12
```

Листинг 2.4: Немного более подробно изучим Розовый шум

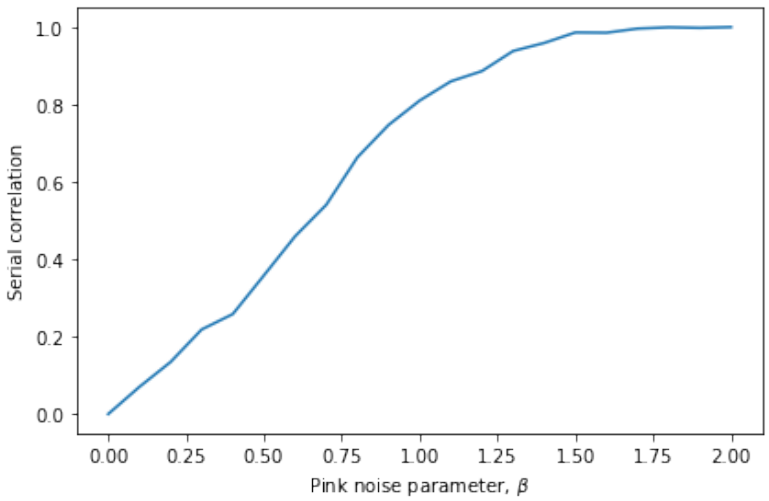

Рис. 2.1: Зависимость значения корреляции от значения параметра

```
1           def autocorr(wave):
2               lags = np.arange(len(wave.ys) // 2)
3               corrs = [serial_corr(wave, lag) for lag in lags]
4               return lags, corrs
5
```

```
6           def plot_pink_autocorr(beta, label):
7               signal = PinkNoise(beta = beta)
8               wave = signal.make_wave(duration=1.0, framerate
    =10000)
9               lags, corrs = autocorr(wave)
10              plt.plot(lags, corrs, label = label)
11
12          np.random.seed(25)
13
14          for beta in [1.7, 1.0, 0.3]:
15              label = r'$\beta$ = %.1f' % beta
16              plot_pink_autocorr(beta, label)
17
```

Листинг 2.5: Функция автокорреляции для Розового шума

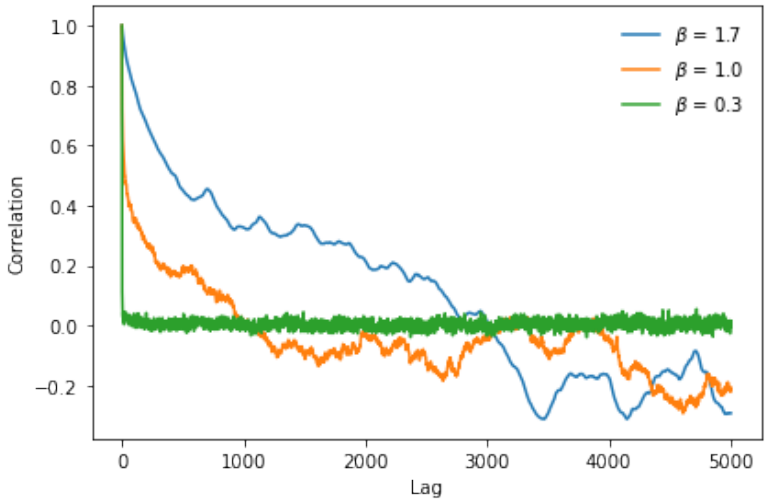

Рис. 2.2: Зависимость корреляции от начальной фазы сигнала

При низких значениях β корреляция быстро падает. По мере увеличения β эта зависимость длится дольше.

Глава 3

Автокорреляция периодических сигналов

Теперь изучим автокорреляцию разных периодических сигналов.

```
from thinkdsp import read_wave

wave = read_wave('28042__bcjordan__voicedownbew.wav')
wave.normalize()
spectrum = wave.make_spectrum()
spectrum.plot()
spectro = wave.make_spectrogram(seg_length = 1024)
spectro.plot(high = 4200)

```

Листинг 3.1: Запись вокального исполнения чирпов

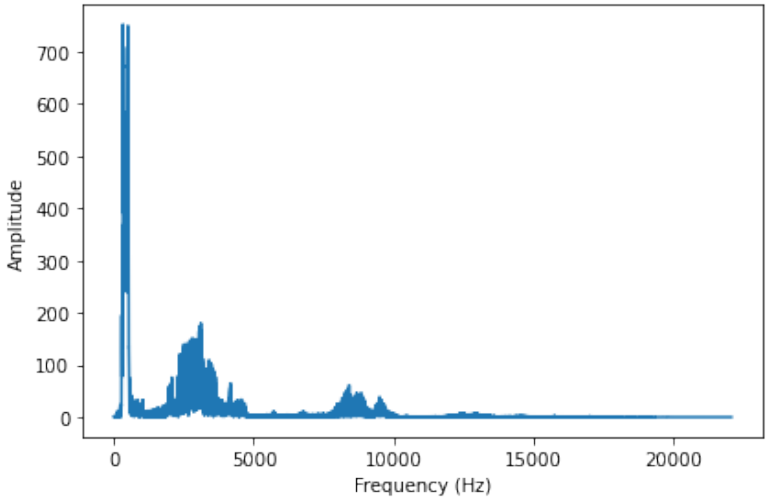

Рис. 3.1: Спектр данного сигнала

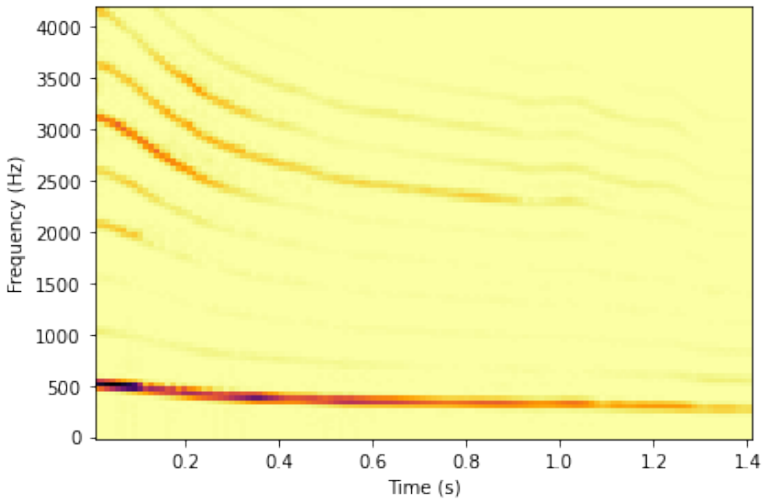

Рис. 3.2: Спектрограмма сигнала

```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
segment.plot()
spectrum = segment.make_spectrum()
spectrum.plot(high = 1000)

```

Листинг 3.2: Уменьшение ширины сегмента

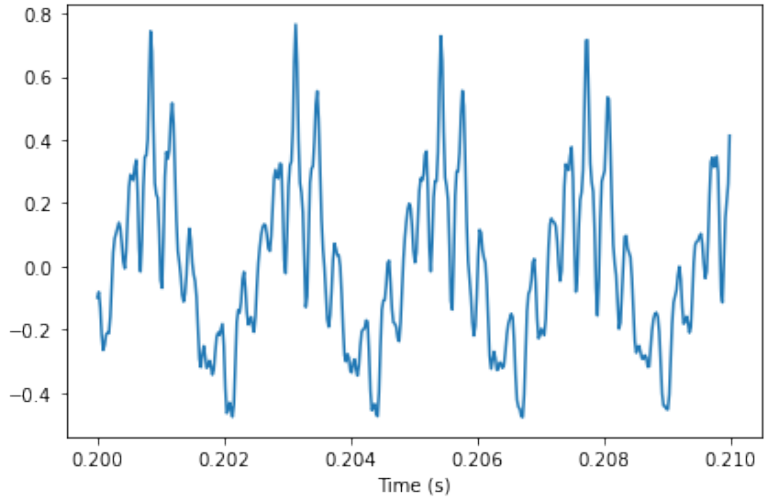

Рис. 3.3: Полученный сегмент сигнала

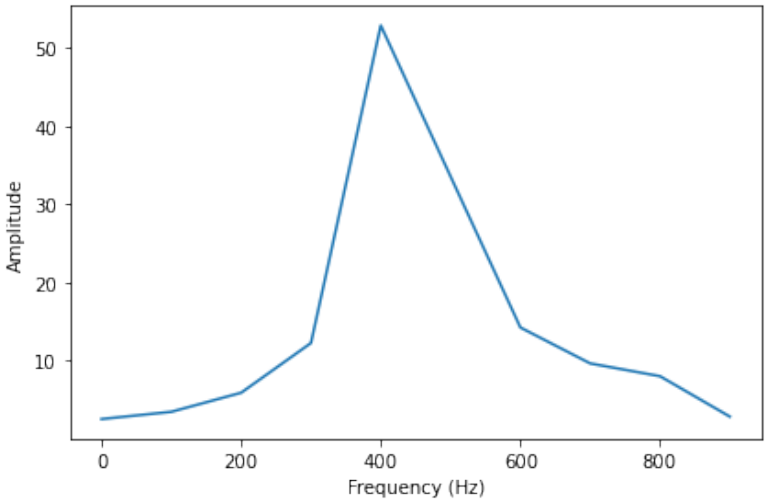

Рис. 3.4: Спектр этого сегмента

```
        def plot_shifted(wave, offset = 0.001, start = 0.2):
            segment1 = wave.segment(start=start, duration
=0.01)
            segment1.plot(linewidth = 2, alpha = 0.8)
            segment2 = wave.segment(start=(start - offset),
duration=0.01)
            segment2.shift(offset)
            segment2.plot(linewidth = 2, alpha = 0.4)
            corr = segment1.corr(segment2)
```

```
8               text = r'$\rho =$ %.2g' % corr
9               plt.text((segment1.start + 0.0005), -0.8, text)
10              decorate(xlabel = 'Time (s)')
11
12          slider1 = widgets.FloatSlider(min = 0, max = 0.004,
    step = (0.004 / 40), value = 0)
13          slider2 = widgets.FloatSlider(min = 0.1, max = 0.5,
    step = 0.05, value = 0.2)
14          interact(plot_shifted, wave = fixed(wave), offset =
    slider1, start = slider2);
15
```

Листинг 3.3: Функция для нахождения корреляции между сигналами

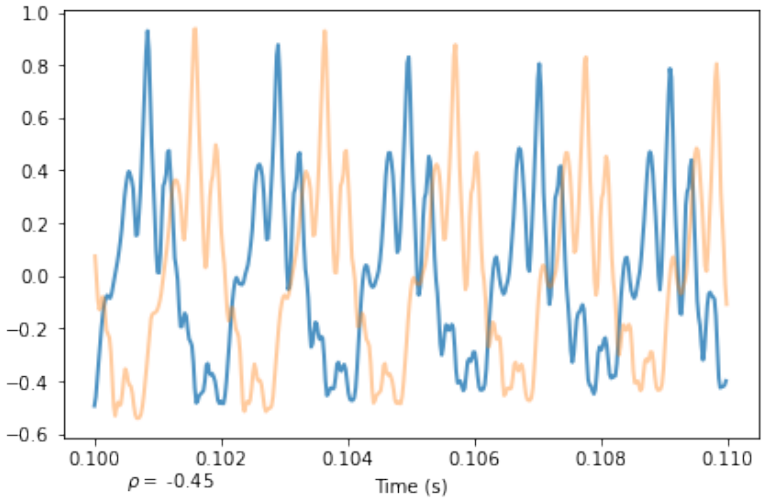

Рис. 3.5: Зависимость значения корреляции от фазы

```
1           wave = read_wave('28042__bcjordan__voicedownbew.wav')
2           wave.normalize()
3           duration = 0.01
4           segment = wave.segment(start=0.2, duration=duration)
5           lags, corrs = autocorr(segment)
6           plt.plot(lags, corrs)
7
```

Листинг 3.4: Автокорреляция чирпа

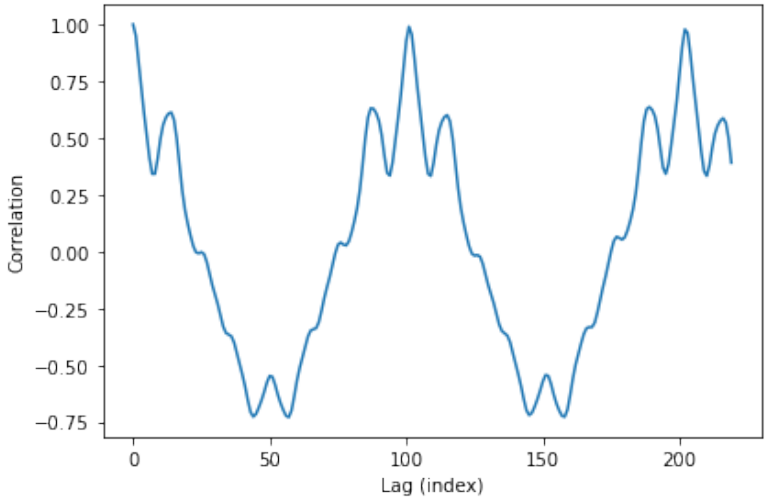

Рис. 3.6: Зависимость корреляции от начальной фазы сигнала

Глава 4

Встроенная функция автокорреляции

Поспользуемся встроенной функцией автокорреляции в модуле NumPy.

```
1         N = len(segment)
2         corrs2 = np.correlate(segment.ys, segment.ys, mode =
      'same')
3         lags = np.arange(-N // 2, N // 2)
4         plt.plot(lags, corrs2)
5
```

Листинг 4.1: Использование встроенной функции для автокорреляции

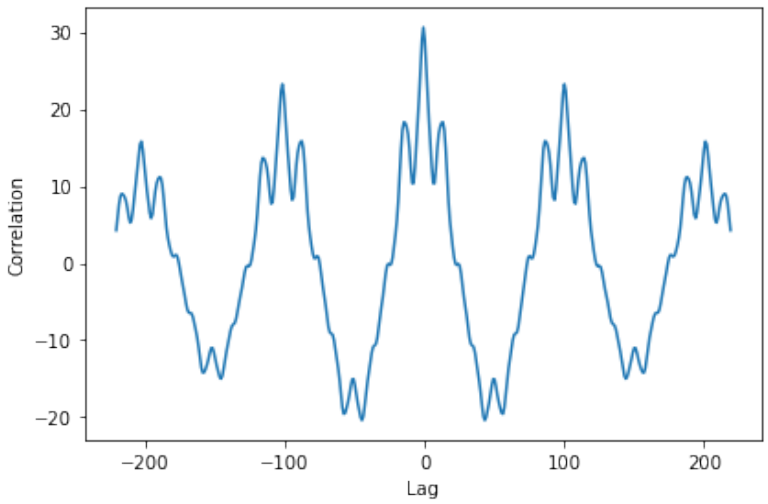

Рис. 4.1: Полученная зависимость

```
N = len(corrs2)
lengths = range(N, N // 2, -1)
half = corrs2[N // 2:].copy()
half /= lengths
half /= half[0]
plt.plot(half)

```

Листинг 4.2: Правая часть сигнала

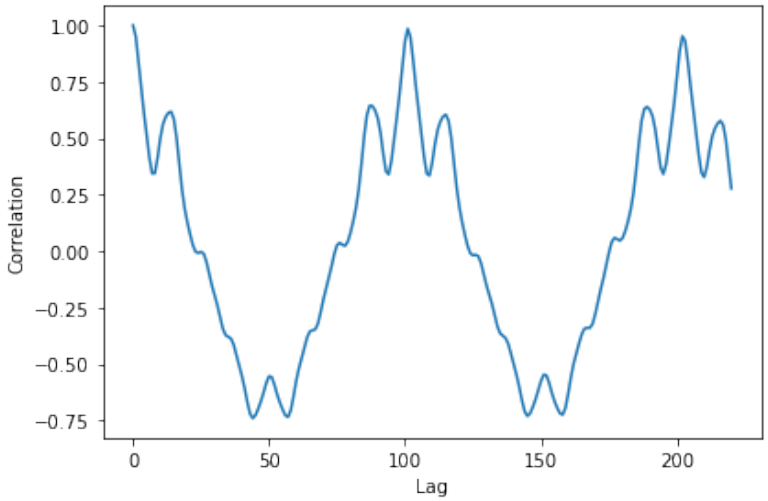

Рис. 4.2: Взяли правую часть сигнала (фаза > 0)

```
plt.plot(half)
plt.plot(corrs)
diff = corrs - half[:-1]
plt.plot(diff)

```

Листинг 4.3: Сравнение двух функций

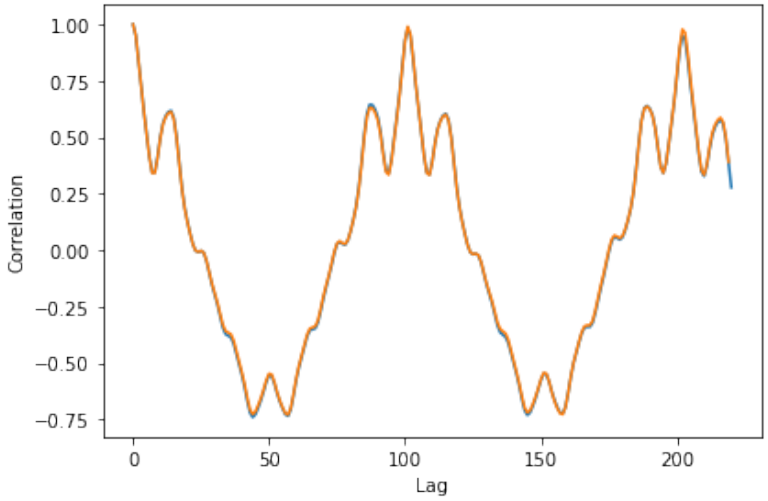

Рис. 4.3: Сравнение полученных результатов двух различных функций автокорреляции

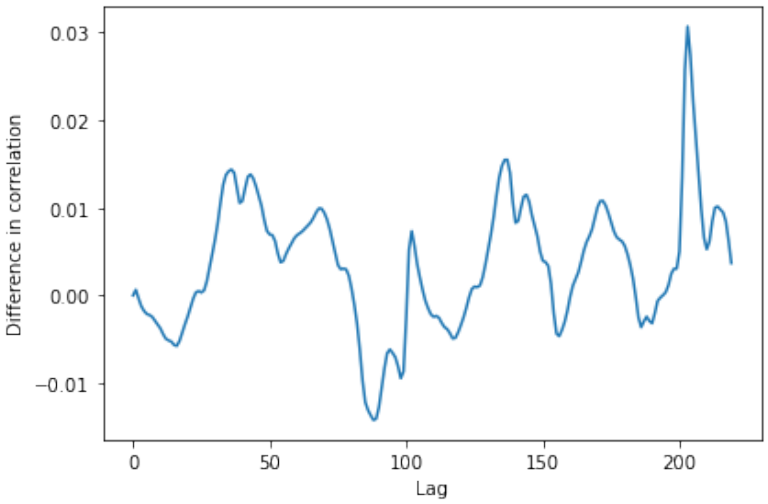

Рис. 4.4: Разница между результатами двух функций

Результаты работы этих двух функций очень похожи, но встроенная функция работает быстрее.

Глава 5

Упражнения

5.1 Задание 2

Создание итоговой функции для оценки основной частоты периодического сигнала.

```
        from thinkdsp import read_wave

        wave = read_wave('28042__bcjordan__voicedownbew.wav')
        wave.normalize()
        wave.make_spectrogram(2048).plot(high = 4200)

        def estimate_fundamental(segment, low=70, high=150):
            lags, corrs = autocorr(segment)
            lag = np.array(corrs[low:high]).argmax() + low
            period = lag / segment.framerate
            frequency = 1 / period
            return frequency

```

Листинг 5.1: Итоговая функция

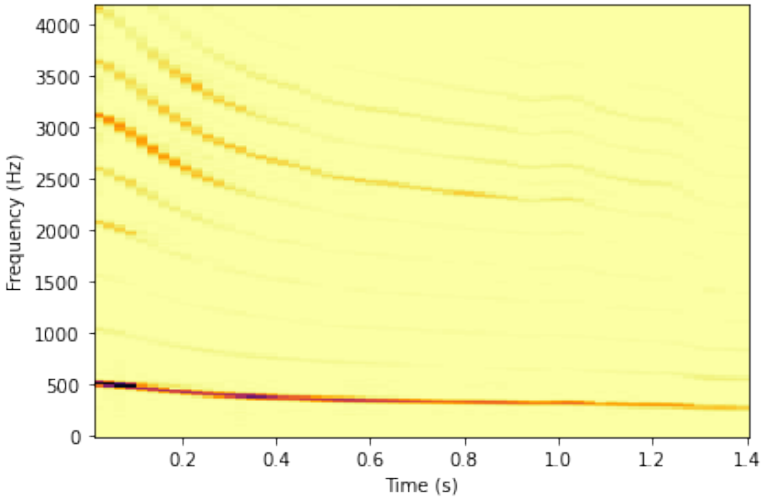

Рис. 5.1: Спектрограмма сигнала

```
        step = 0.05
        starts = np.arange(0.0, 1.4, step)
        ts = []
        freqs = []
        for start in starts:
            ts.append(start + step / 2)
            segment = wave.segment(start=start, duration=
duration)
            freq = estimate_fundamental(segment)
            freqs.append(freq)
        wave.make_spectrogram(2048).plot(high = 900)
        plt.plot(ts, freqs, color = 'white')

```

Листинг 5.2: Проверка работы функции

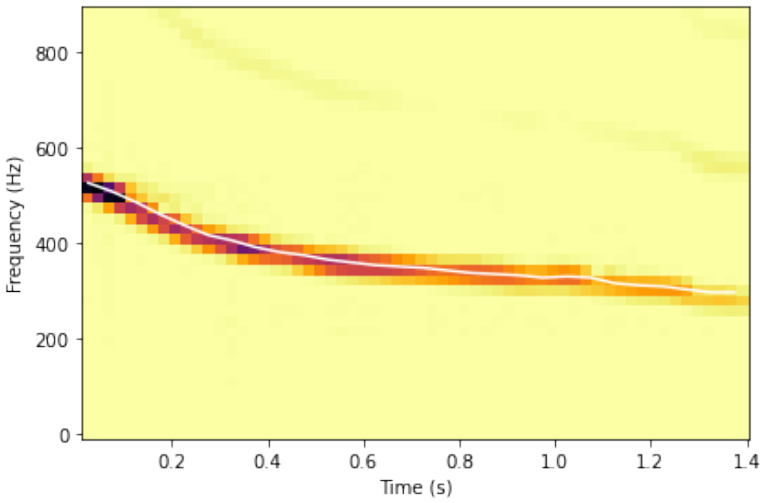

Рис. 5.2: Полученная спектрограмма

5.2 Задание 3

Автокорреляция цен в платежной системе `BitCoins`.

```
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         lags, corrs = autocorr(wave)
10        plt.plot(lags, corrs)
11
```

Листинг 5.3: Метод Бартлетта

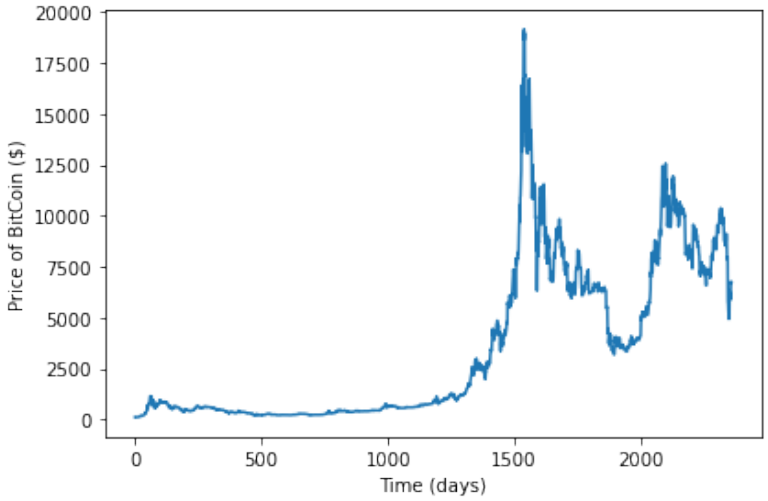

Рис. 5.3: Сигнал BitCoins

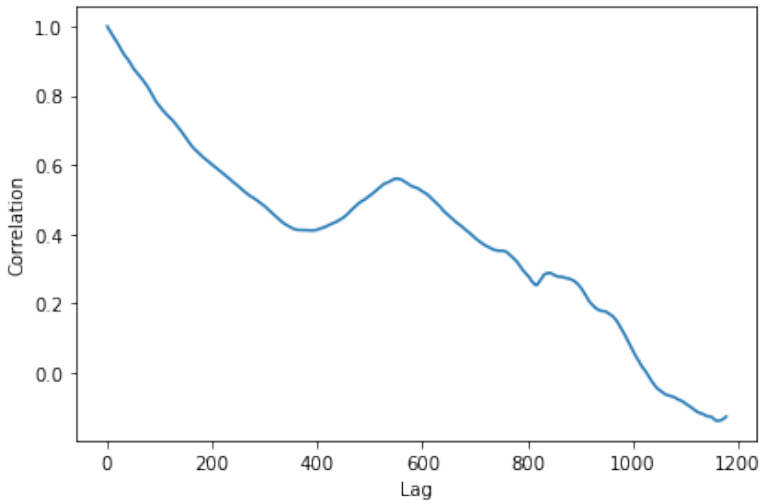

Рис. 5.4: Автокорреляция этого сигнала

```
N = len(wave)
corrs2 = np.correlate(wave.ys, wave.ys, mode='same')
lags = np.arange(-N // 2, N // 2)
N = len(corrs2)
half = corrs2[N // 2:]
lengths = range(N, N // 2, -1)
half /= lengths
half /= half[0]
plt.plot(corrs, label = 'autocorr')
```

```
10          plt.plot(half, label = 'correlate')
11
```

Листинг 5.4: Снова сравним две функции

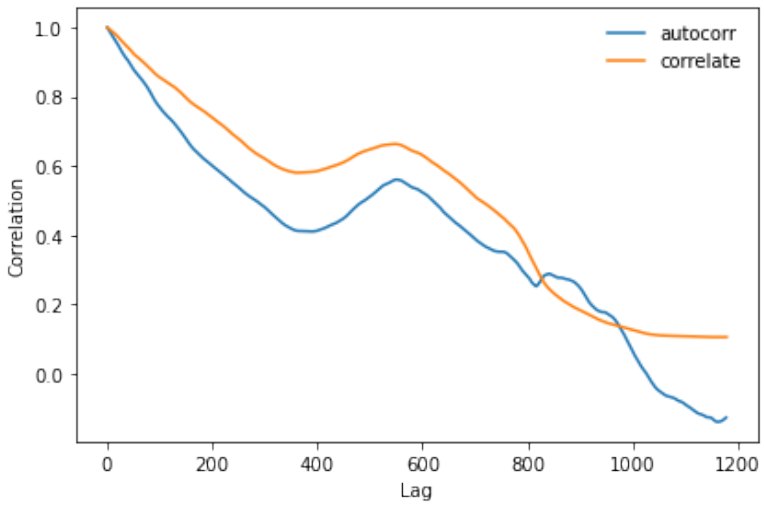

Рис. 5.5: Сравнение результатов работы двух функций

5.3 Задание 4

Исследуем код в файле `saxophone.ipynb`.

```
1           wave=read_wave('100475__iluppai__saxophone-weep.wav')
2           wave.normalize()
3           gram = wave.make_spectrogram(seg_length = 1024)
4           gram.plot(high = 3000)
5
```

Листинг 5.5: Получение сигнала

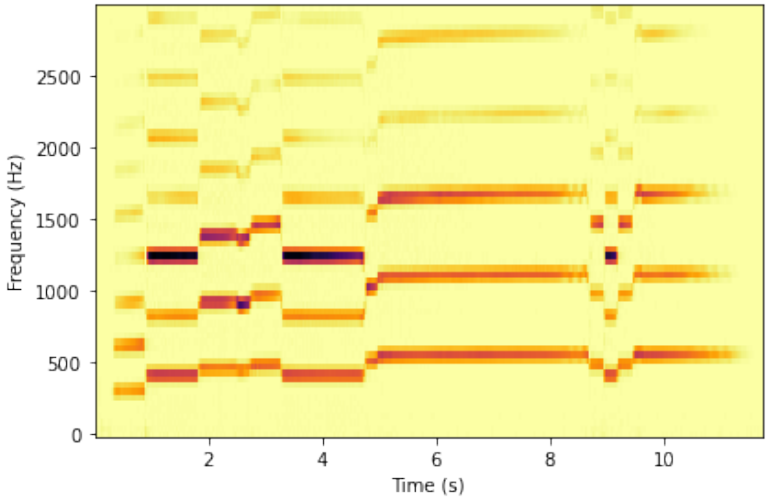

Рис. 5.6: Спектрограмма сигнала

```
        start = 2.0
        duration = 0.5
        segment = wave.segment(start = start, duration =
duration)
        spectrum = segment.make_spectrum()
        spectrum.plot(high = 3000)

```

Листинг 5.6: Продолжаем исследование

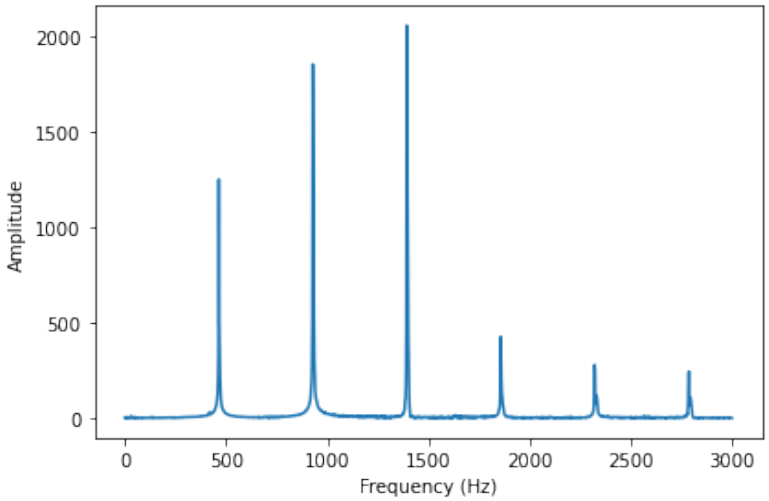

Рис. 5.7: Спектр сигнала

Пики спектра находятся на частотах 1392, 928 и 464 Гц.

```
        def autocorr_new(segment):
            corrs = np.correlate(segment.ys, segment.ys, mode
    = 'same')
            N = len(corrs)
            lengths = range(N, N // 2, -1)
            half = corrs[N // 2:].copy()
            half /= lengths
            half /= half[0]
            return half

        corrs = autocorr_new(segment)
        plt.plot(corrs[:200])

```

Листинг 5.7: Новая функция для автокорреляции

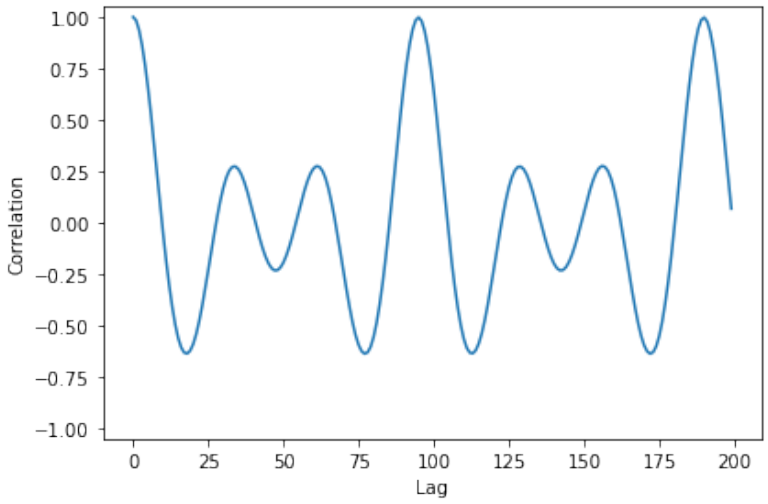

Рис. 5.8: График корреляции

```
        def find_frequency(corrs, low, high):
            lag = np.array(corrs[low:high]).argmax() + low
            period = lag / segment.framerate
            frequency = 1 / period
            return frequency

        find_frequency(corrs, 80, 100)
        spectrum2 = segment.make_spectrum()
        spectrum2.high_pass(600)
        spectrum2.plot(high = 3000)
        corrs = autocorr_new(segment2)
```

```
plt.plot(corrs[:200])

```

Листинг 5.8: Применили ФВЧ

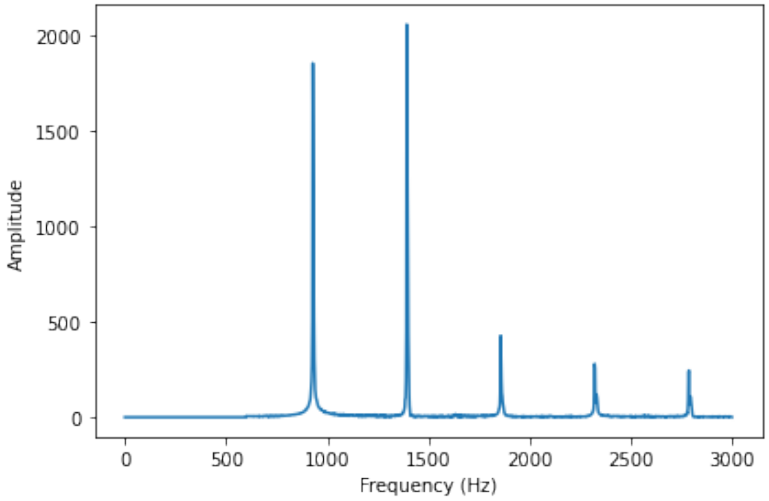

Рис. 5.9: Спектр сигнала после применения ФВЧ

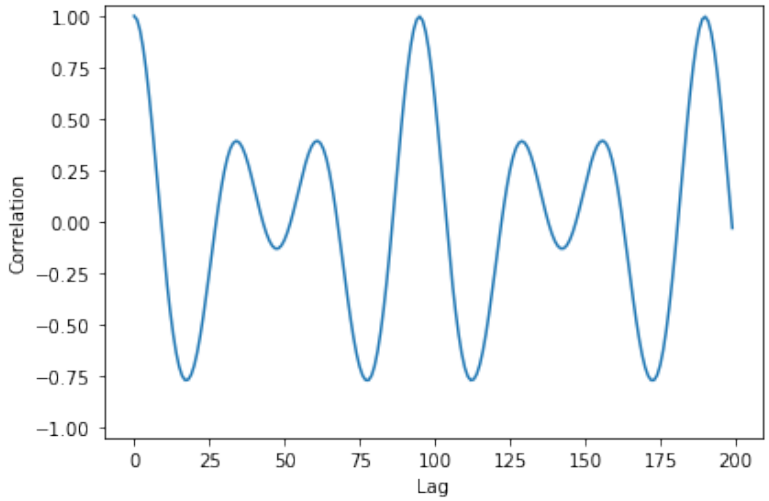

Рис. 5.10: Снова применили функцию автокорреляции

```
spectrum4 = segment.make_spectrum()
spectrum4.high_pass(600)
spectrum4.low_pass(1200)
```

```
spectrum4.plot(high = 3000)
corrs = autocorr_new(segment4)
plt.plot(corrs[:200])
```

Листинг 5.9: Применили ФВЧ и ФНЧ

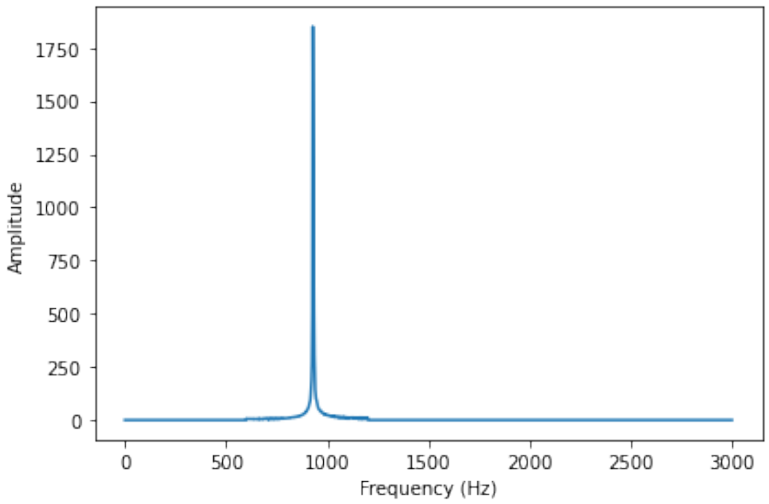

Рис. 5.11: Спектр сигнала после применения ФВЧ и ФНЧ

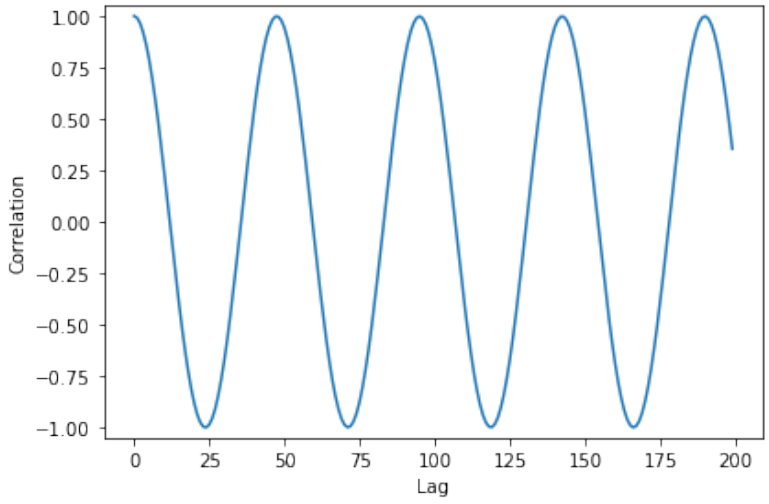

Рис. 5.12: Применили функцию автокорреляции

В результате, можно сделать вывод, что восприятие высоты тона не пол-

ностью основано на спектральном анализе, но также зависит и от автокорреляции.

Глава 6

Вывод

В данной работе мы изучили изменение корреляции от фазы одинаковых сигналов и посмотрели влияние функции автокорреляции на звучание различных сигналов.