Лабораторная работа 5

Луняк Николай

8 апреля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Подбор высоты

В этом задании надо воспользоваться готовым материалом из `chap05.ipynb` и, двигая слайдеры, подобрать период сигнала на основе автокорреляции.

Будем последовательно выставлять разные значения для `start`, а затем, начиная с `offset` = 0, подбирать период.

Так как на слайдере не удается различить столь малые числа (а как настроить его по-другому мне не особо хочется разбираться), текущее значение `offset` буду запрашивать в отдельной ячейке как `slider1.value`.

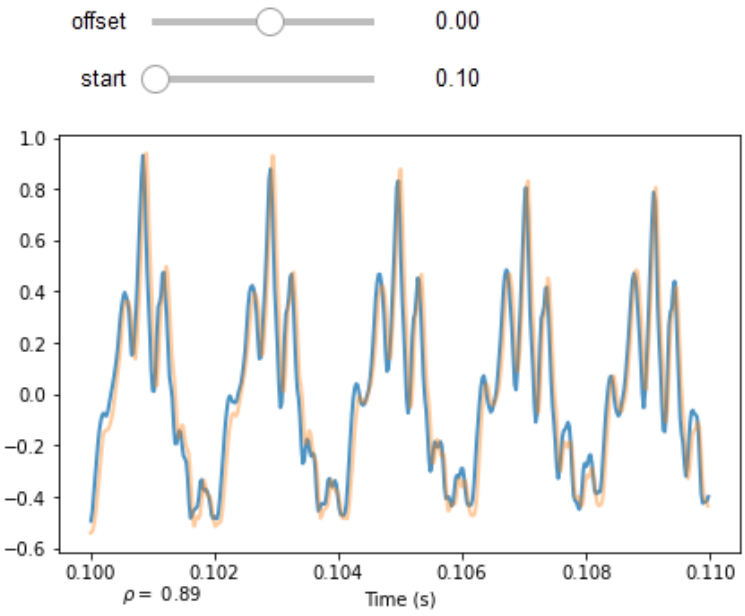

Рис. 1.1: Для `offset` = 0.0021

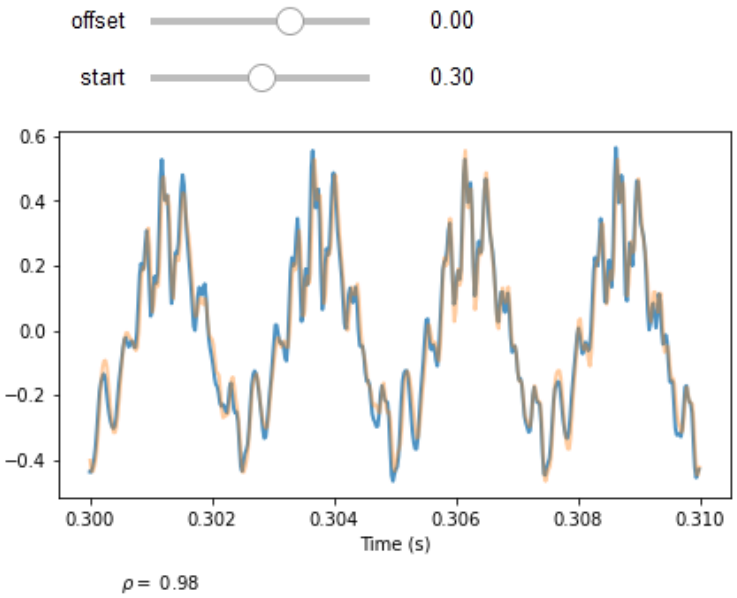

Рис. 1.2: Для `offset` = 0.0025

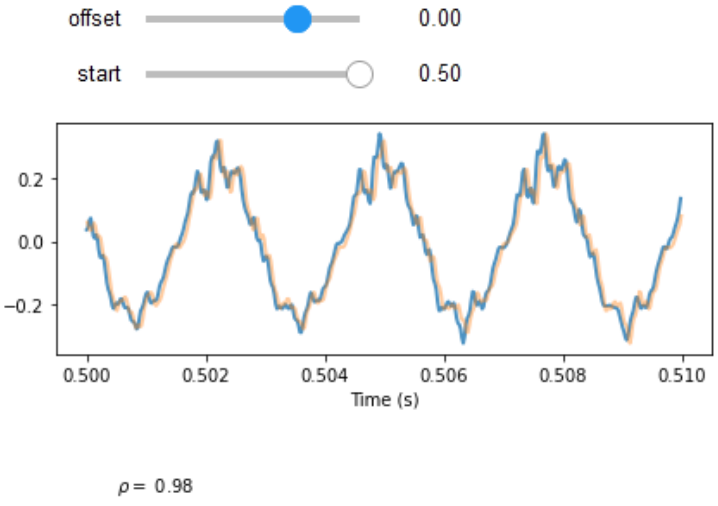

Рис. 1.3: Для `offset` = 0.0028

Как мы можем видеть, `offset` меняется нелинейно, что логично, учитывая особенности исследуемого сигнала.

Глава 2

Автоматизируем процесс

```
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
from thinkdsp import Spectrum, Wave

import numpy as np
import pandas as pd

from matplotlib import pyplot

import thinkstats2

def serial_corr(wave, lag=1):
    """Computes serial correlation with given lag.

    wave: Wave
    lag: integer, how much to shift the wave

    returns: float correlation coefficient
    """
    n = len(wave)
    y1 = wave.ys[lag:]
    y2 = wave.ys[:n-lag]
    corr_mat = np.corrcoef(y1, y2)
    return corr_mat[0, 1]

def autocorr(wave):
    """Computes and plots the autocorrelation function.

    wave: Wave
    """
```

```
    lags = np.arange(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs

wave = read_wave('Sounds/28042__bcjordan__voicedownbew.wav')
wave.normalize()
wave.make_audio()
```

Листинг 2.1: Берем готовый код и загружаем звук

Теперь реализуем функцию `estimate_fundamental()` и сравним ее результат со спектрограммой (будем вызывать ее периодически, а потом соединим эти результаты одной линией поверх спектрограммы).

```
def estimate_fundamental(segment, start=70, end=150):
    lags, correlations = autocorr(segment)
    lag = np.array(correlations[start:end]).argmax() + start
    period = lag / segment.framerate
    return 1 / period

ts = []
frequencies = []

for it in np.arange(0.0, 1.4, 0.05):
    ts.append(it + 0.025)
    segment = wave.segment(start=it, duration=0.01)
    frequency = estimate_fundamental(segment)
    frequencies.append(frequency)

wave.make_spectrogram(2048).plot(high=2000)
pyplot.plot(ts, frequencies, color='blue')
decorate(
    xlabel='Time (s)',
    ylabel='Frequency (Hz)'
)
```

Листинг 2.2: Сравниваем

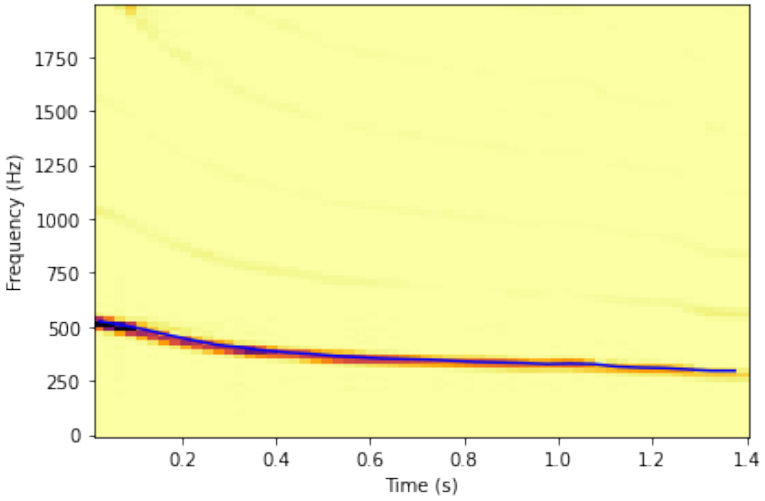

Рис. 2.1: Спектрограмма

Синяя линия ложится ровно поверх красной (основной частоты), что дает понять, что наше вычисление работает правильно.

Глава 3

Без BinCoin'а уже никуда

Загрузка данных аналогична прошлой лабораторной работе.

```
data = pd.read_csv('Data/BTC_USD_2020-12-31_2021-03-30-CoinDesk.csv')
wave = Wave(data['Closing Price (USD)'], data.index, framerate=1)
wave.plot()
```

Листинг 3.1: Загрузка датасета

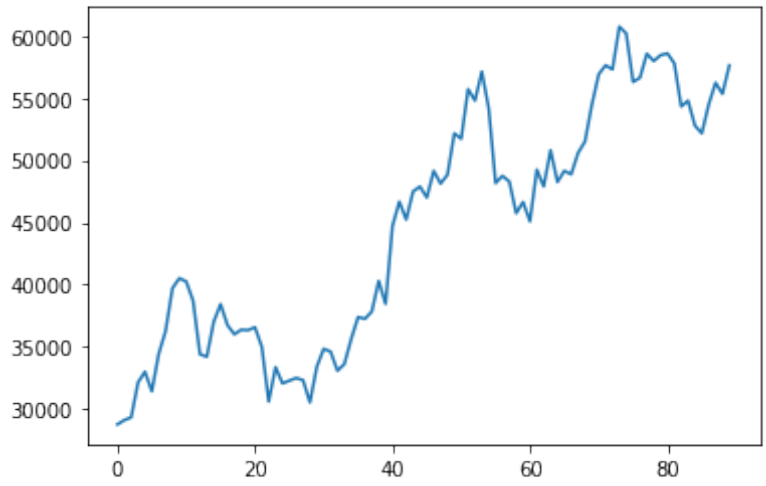

Рис. 3.1: Курс битка

```
lags, correlations = autocorr(wave)
pyplot.plot(lags, correlations)
decorate(
    xlabel='Lag',
```

```
5       ylabel='Correlation'
6   )
```

Листинг 3.2: Автокорреляция

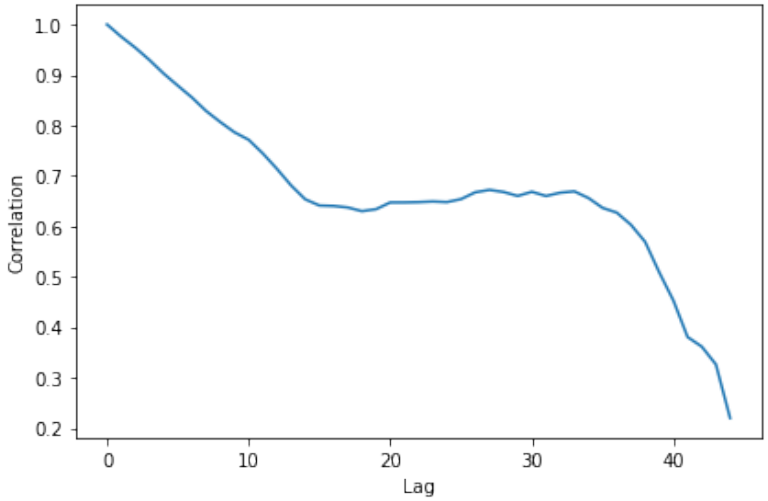

Рис. 3.2: Автокорреляция

Спадает она не быстро, больше походит на «розовый» шум, что согласуется с результатом предыдущей лабораторной работы.

Глава 4

Исследуем `saxophone.ipynb`

В рамках данного раздела нам предложен к рассмотрению файл `saxophone.ipynb`, в котором рассматривается феномент «отсутствующего основного тона». От нас же требуется потыкать разные кнопки и посмотреть на разные результаты при выборе разных сегментов. И YouTube [Vih] посмотреть еще.

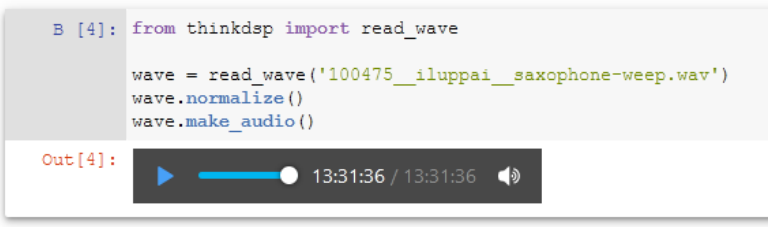

Рис. 4.1: КРАСИВО

Если мы посмотрим на спектр участка $[2; 2.5]$ секунд, то увидим следующую картину.

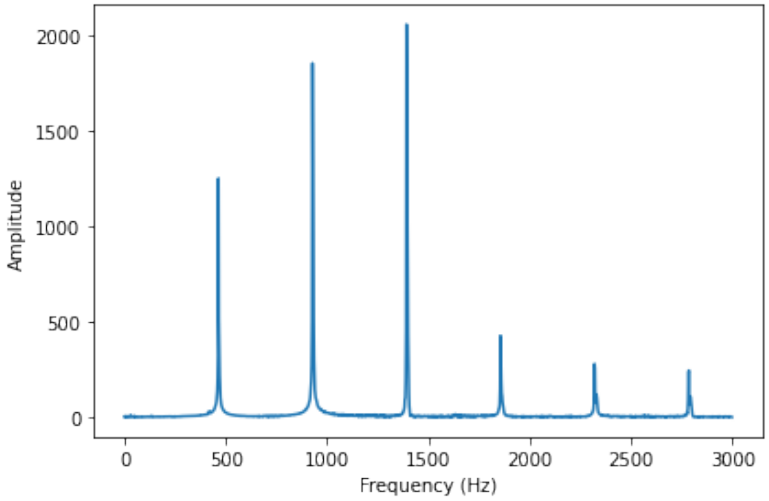

Рис. 4.2: Спектр вблизи 2 сек.

Хотя основной частотой и воспринимается 464 Гц, на самом деле ей должна была бы быть 1392 Гц. Объяснить этот эффект может помочь автокорреляция.

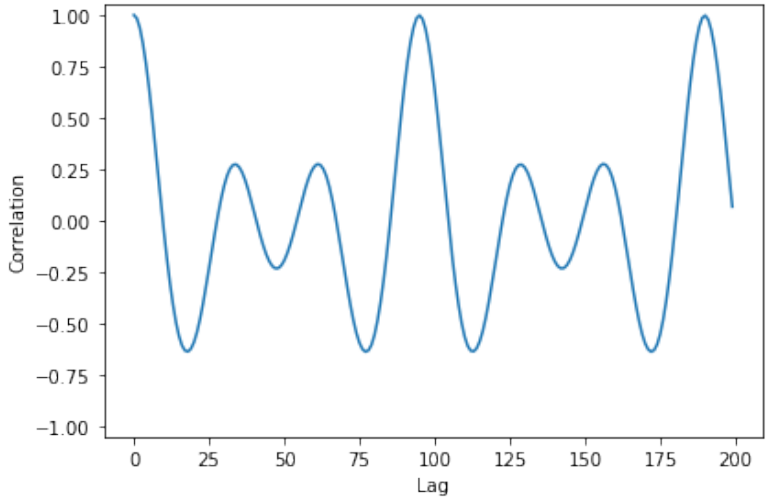

Рис. 4.3: Автокорреляция

Первый большой пик расположен вблизи 100 лага. Для того, чтобы вычислить точный лаг и соответствующую частоту, автор прибегает к коду, аналогичному `estimate_fundamential()`. В итоге получается $(95, 464\text{Hz})$.

Если удалить компоненту 464 Гц из спектра, то на слух все равно будет казаться, что основной тон 464 Гц. Этот эффект и называется «пропавшей» основной частотой (ладно, признаюсь, я читал оригинал ThinkDSP, а как по-русски это называется, я не знаю).

Если снова нарисовать функцию автокорреляции, то окажется, что она совпадает с той, что мы видели ранее. Впрочем, на этом рисунке есть и другие пики, поменьше. Почему ухо воспринимает именно 464 Гц как основной тон? Оказывается, все остальные компоненты спектра - это гармоники 464 Гц, и именно на эту информацию опирается мозг при попытке оценить «настоящую» основную частоту.

Если избавиться от гармоник (`high_pass(600)`, `low_pass(1200)`), то и сам эффект пропадает.

Видео, кстати, тоже интересное.

Список литературы

[Vih] Vihart. *What is up with Noises? (The Science and Mathematics of Sound, Frequency, and Pitch)*. URL: https://www.youtube.com/watch?v=i_0DXxNeaQ0.