Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Автокорреляция

Выполнил студент гр. 3530901/80201 И.С. Иванов

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение №1

Во первом упражнении необходимо вычислить автокорреляцию для различных `lag` и оценить высоту тонна локального `chirp`.

Создадим функции `autocorr` и `serial_corr`

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

```

Листинг 1: Функция serial_corr

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

```

Листинг 2: Функция autocorr

Прочитаем файл. Построим график автокорреляции.

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

segment = wave.segment(0, 0.01)
lags, corrs = autocorr(segment)
low, high = 90, 110
lag = np.array(corrs[low:high]).argmax() + low
plt.plot(lags, corrs, color='blue')
decorate(xlabel='Lag', ylabel='Correlation')

```

Листинг 3: Чтение файла, построение графика автокорреляции

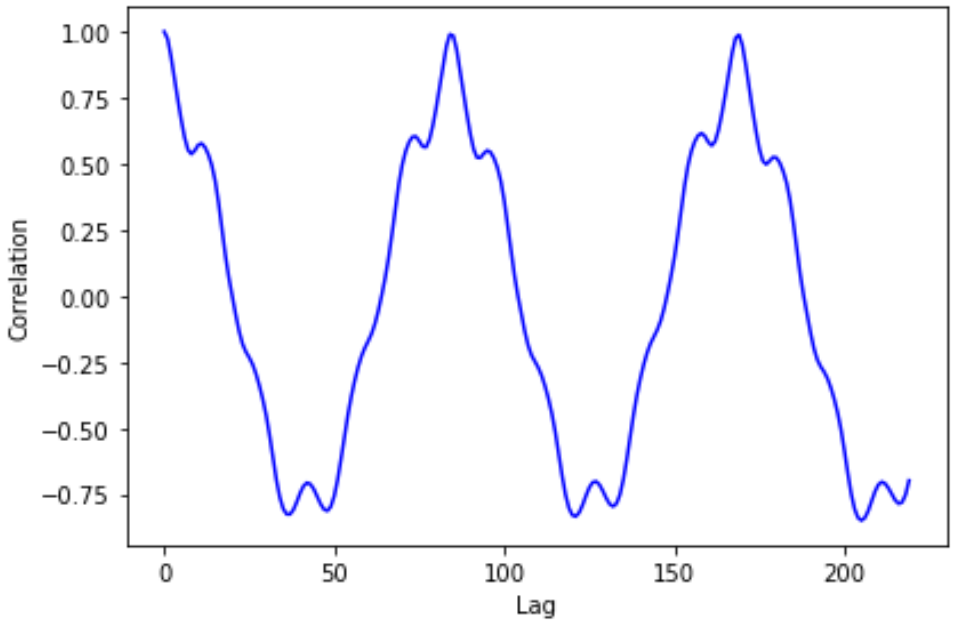

Рис. 1: График автокорреляции

На графике видно, что он периодический. Период равен 90.

2 Упражнение №2

Во втором упражнении необходимо написать функцию `estimate_fundamental`, отслеживающую высоту тона записанного звука. Также необходимо проверить ее работоспособность.

Напишем функцию:

```
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 4: Функция estimate_fundamental

Посмотрим на спектрограмму аудио файла из предыдущего упражнения.

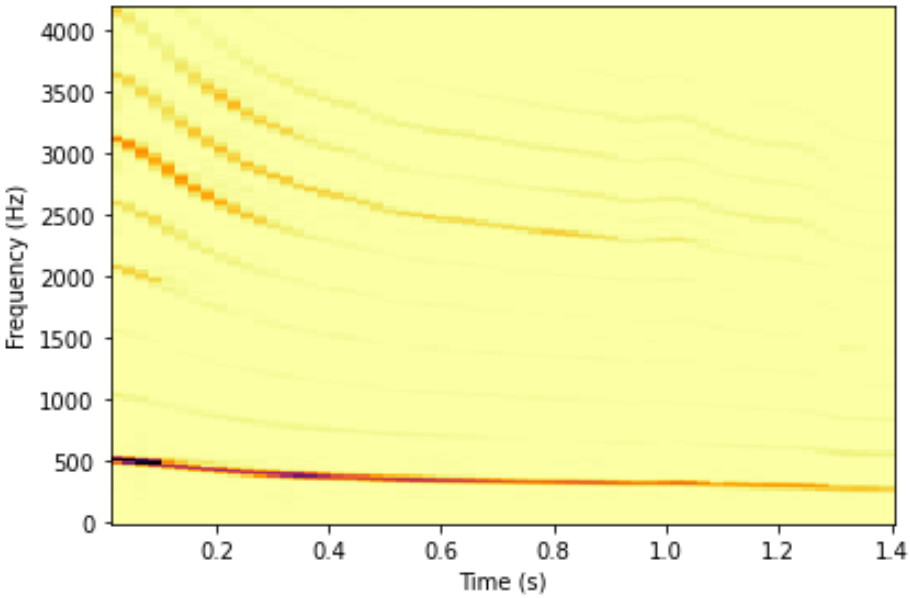

Рис. 2: Спектрограмма аудио файла

С помощью написанной ранее функции получим минимальную частоту сегмента.

```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
freq = estimate_fundamental(segment)
freq

```

Листинг 5: Получение минимальной частоты сегмента

Минимальная частота равна 436.63366336633663.

Найдем этот сегмент с шагом 0.05.

```
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)

```

Листинг 6: Поиск сегмента с минимальной частотой

Выведем спектрограмму найденного сегмента.

```
wave.make_spectrogram(2048).plot(high=900)
plt.plot(ts, freqs, color='white')
decorate(xlabel='Time (s)',
                      ylabel='Frequency (Hz)')

```

Листинг 7: Спектрограмма сегмента

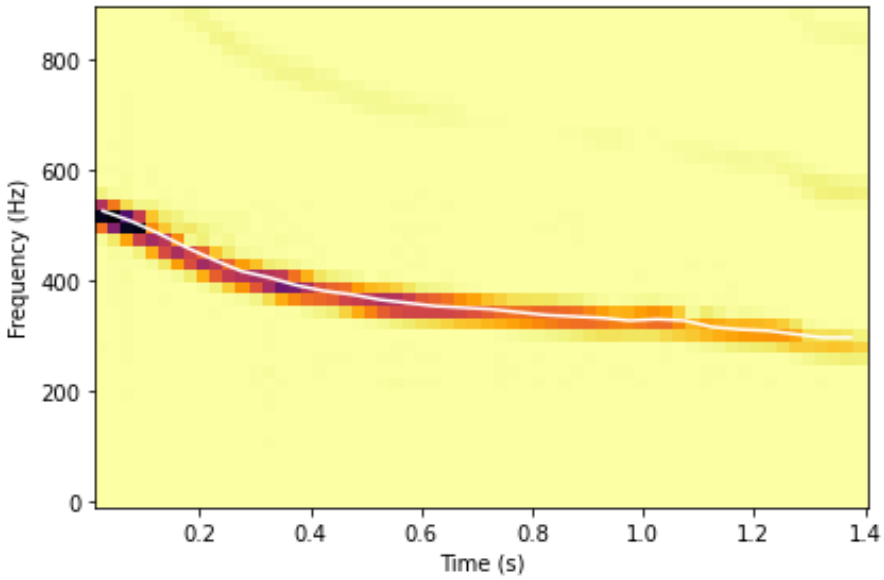

Рис. 3: Спектрограмма сегмента

На полученной спектрограмме можно увидеть искомую частоту.

3 Упражнение №3

В третьем упражнении нам необходимо используя данные курса Bitcoin из предыдущей лабораторной работы вычислить автокорреляцию курса.

Считаем файл и выведем график:

```
import pandas as pd

df = pd.read_csv('Res/BTC_USD_2013-10-01_2021-05-04-CoinDesk.csv',
                 parse_dates=[0])

ys = df['Closing Price (USD)']
ts = df.index

from thinkdsp import Wave

wave = Wave(ys, ts, framerate=1)
wave.plot()
decorate(xlabel='Time (days)',
         ylabel='Price of BitCoin ($)')

```

Листинг 8: Считывание файла

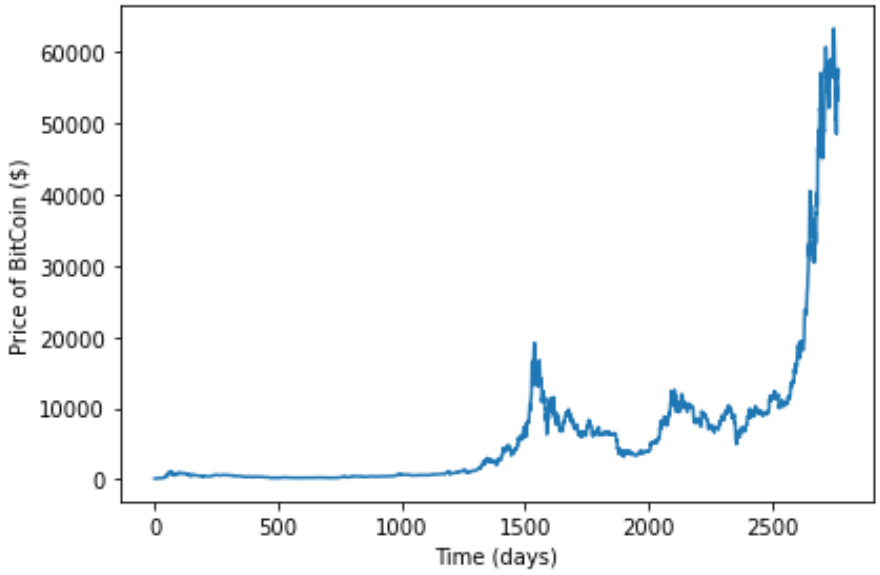

Рис. 4: График курса Bitcoin

Посмотрим на график функции автокорреляции.

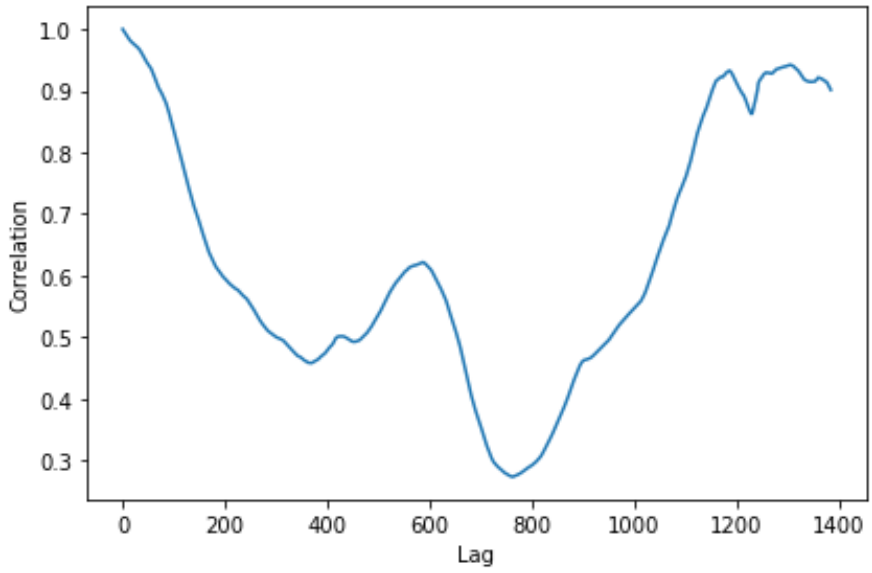

Рис. 5: Автокорреляционный график курса Bitcoin

Периодичности на графике не наблюдается.

4 Упражнение №4

В четвертом упражнении необходимо просмотреть файл `saxophone.ipynb`, пройтись по всем примерам, затем выбрать сегмент записи и поработать с ним.

Построим спектрограмму данного в задании файла

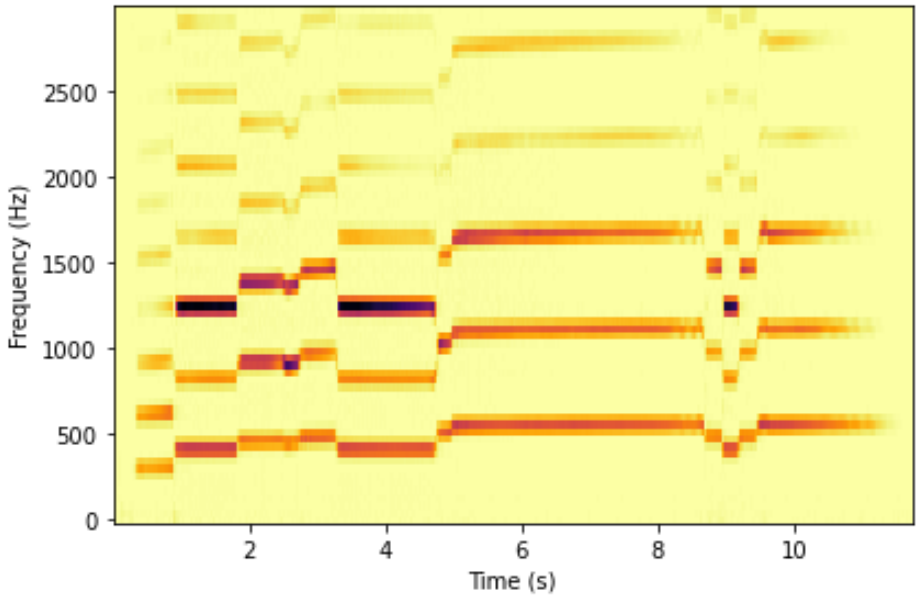

Рис. 6: Спектрограмма звука саксофона

Выделим сегмент отличный от исходного. Сегмент с 8 секунды длительностью 0.5 секунд.

Посмотрим на спектр данного сегмента.

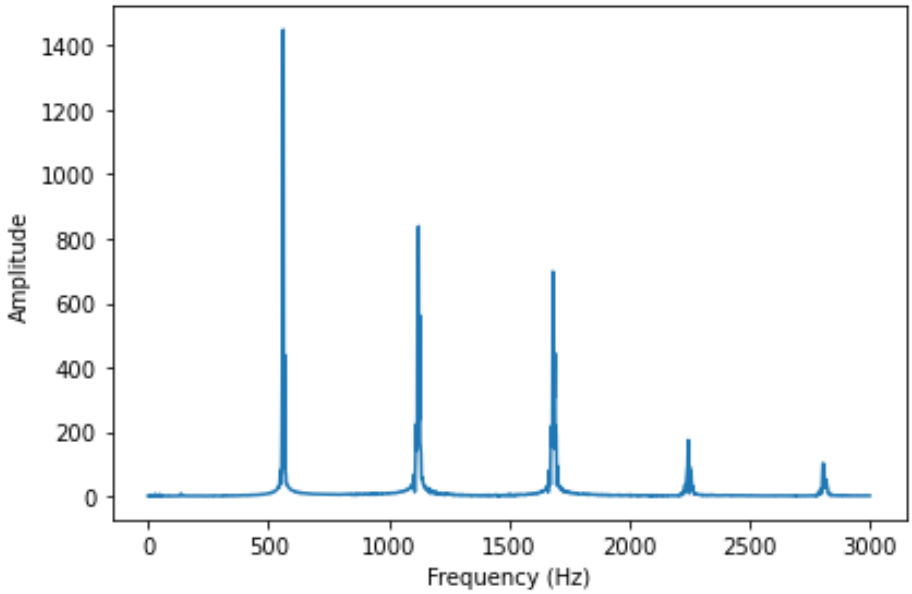

Рис. 7: Спектр сегмента саксофона

Получим ”Peaks”сегмента.

```
[(1449.7324059418409, 562.0),
 (838.6796549940561, 1124.0),
 (760.4866790380277, 1120.0),
 (697.9180739586471, 1682.0),
 (562.4385733324308, 1128.0),
 (548.2317169052268, 1684.0),
 (490.9435713084754, 1122.0),
 (479.5263999004895, 558.0),
 (444.5487141915133, 1690.0),
 (439.8651374537576, 566.0)]
```

Листинг 9: Peaks

Построим треугольный сигнал основной частоты.

```
from thinkdsp import TriangleSignal

TriangleSignal(freq=562).make_wave(duration=0.5).make_audio()
```

Листинг 10: Создание треугольного сигнала

Обновим функцию автокорреляции

```
def autocorr(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

```

Листинг 11: Обновление функции autocorr

Выведем на экран график автокорреляции звукового сегмента.

```
corrs = autocorr(segment)
plt.plot(corrs[:200])
decorate(xlabel='Lag', ylabel='Correlation', ylim=[-1.05, 1.05])

```

Листинг 12: Вывод графика автокорреляции

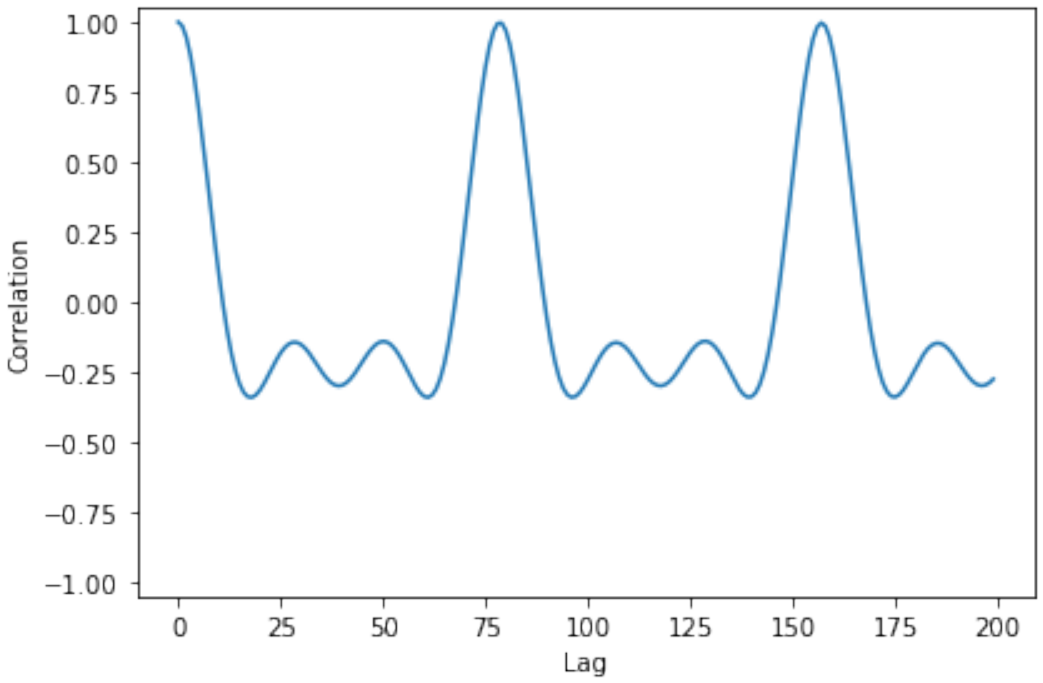

Рис. 8: График автокорреляции сегмента звука саксофона

На графике видно, что первый наибольший пик на значении 80. Для нахождения частоты напишем функцию.

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

```

Листинг 13: Функция find_frequency

```
find_frequency(corrs, 70, 95)

```

Листинг 14: Вызов find_frquency

Самый большой `lag` = 79. Частота 558.2278481012657.

Необходимо отфильтровать сегмент с помощью фильтра низких частот.

```
spectrum2 = segment.make_spectrum()
spectrum2.high_pass(600)
spectrum2.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 15: Фильтрация сегмента

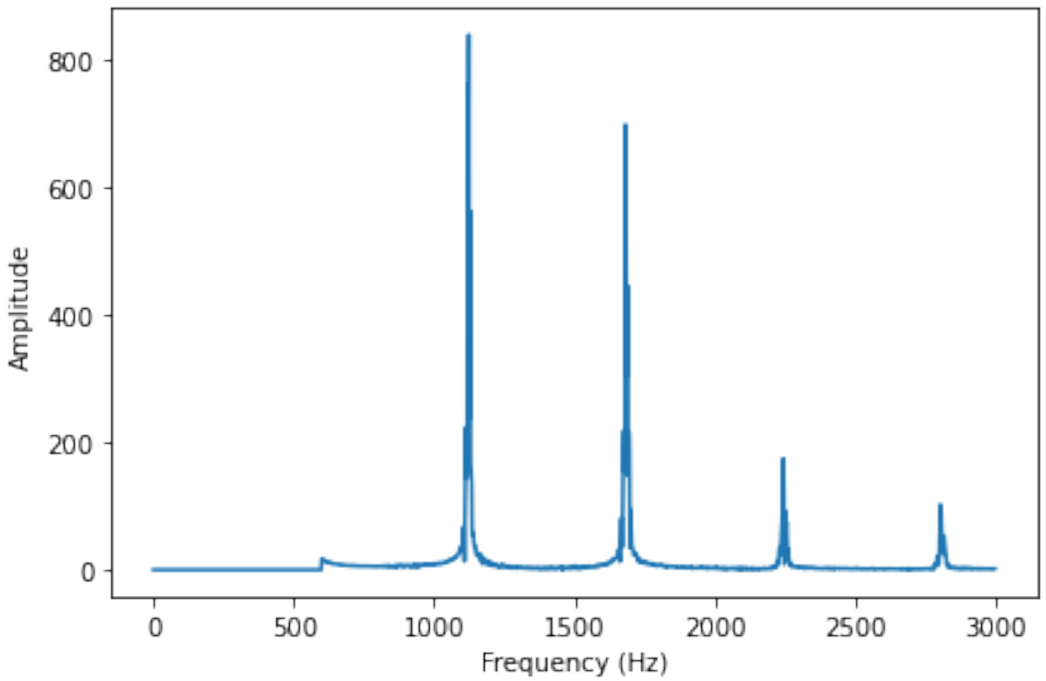

Рис. 9: Спектр отфильтрованного сегмента саксофона

На спектре можно увидеть, что основная частота была убрана.

Сравнив звук исходного фрагмента и отфильтрованного можно сказать, что отфильтрованный звучит приглушеннее, так как убраны низкие частоты.

5 Выводы

В результате выполнения данной лабораторной работы мы изучили, понятие автокорреляции. Была создана функция `estimate_fundamental` для отслеживания высоты тона звука. Была вычислена автокорреляция курса валют `Bitcoin` из прошлой лабораторной работы. Была произведена работа по вычислению значения автокорреляции для сегмента записи саксофона.