

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №5
Дисциплина: Телекоммуникационные технологии
Тема: Автокорреляция

Работу выполнил:
Ляшенко В.В.
Группа: 3530901/80201
Преподаватель:
Богач Н.В.

Санкт-Петербург
2021

Оглавление

1	Упражнение 5.1	4
2	Упражнение 5.2	6
2.1	Функция <code>estimate_fundamental</code>	6
2.2	Спектрограмма	6
3	Упражнение 5.3	9
4	Упражнение 5.4	11
5	Выводы	16

Список иллюстраций

1.1	Автокорреляция первого сегмента	4
1.2	Автокорреляция второго сегмента	5
2.1	Спектрограмма сегмента	7
2.2	Спектрограмма сегмента	8
3.1	Данные о BitCoin	9
3.2	Автокорреляция BitCoin	10
4.1	Спектр сегмента	11
4.2	Автокорреляция	12
4.3	Спектр без основной частоты	13
4.4	Автокорреляция	14
4.5	Фильтрованный спектр	15
4.6	Автокорреляция фильтрованного спектра	15

Листинги

1.1	Получение сегмента	4
1.2	Использование автокорреляцию	4
1.3	Уточнение значения lag	5
1.4	Нахождение частоты	5
1.5	Получение сегмента	5
2.1	Функция estimate_fundamental	6
2.2	Применение функции	6
2.3	Построение спектрограммы	6
2.4	Наложение оценки высоты на спектрограмму	7
3.1	Получение данных о BitCoin	9
3.2	Вычисление автокорреляции	10
4.1	Выделение сегмента	11
4.2	Построение спектра сегмента	11
4.3	Получение треугольной волны	12
4.4	Функция автокорреляции	12
4.5	Применение функции	12
4.6	Удаление основной частоты	13
4.7	Фильтрация гармоник	14

Глава 1

Упражнение 5.1

Блокнот Jupyter `chap05.ipynb` содержит приложение, в котором можно вычислить автокорреляции для различных `lag`. Воспользуемся им и оценим высоту тона вокального чирпа для нескольких времен начала сегмента.

Для начала скачаем запись с чирпом и выделим небольшой сегмент.

```
wave = read_wave('28042__bcjordan__voicedownbew.wav')
wave.normalize()
segment = wave.segment(start=0.1, duration=0.01)
```

Листинг 1.1: Получение сегмента

Теперь используем автокорреляцию (Рис.1.1).

```
lags, corrs = autocorr(segment)
plt.plot(lags, corrs)
decorate(xlabel='Lag (index)', ylabel='Correlation')
```

Листинг 1.2: Использование автокорреляцию

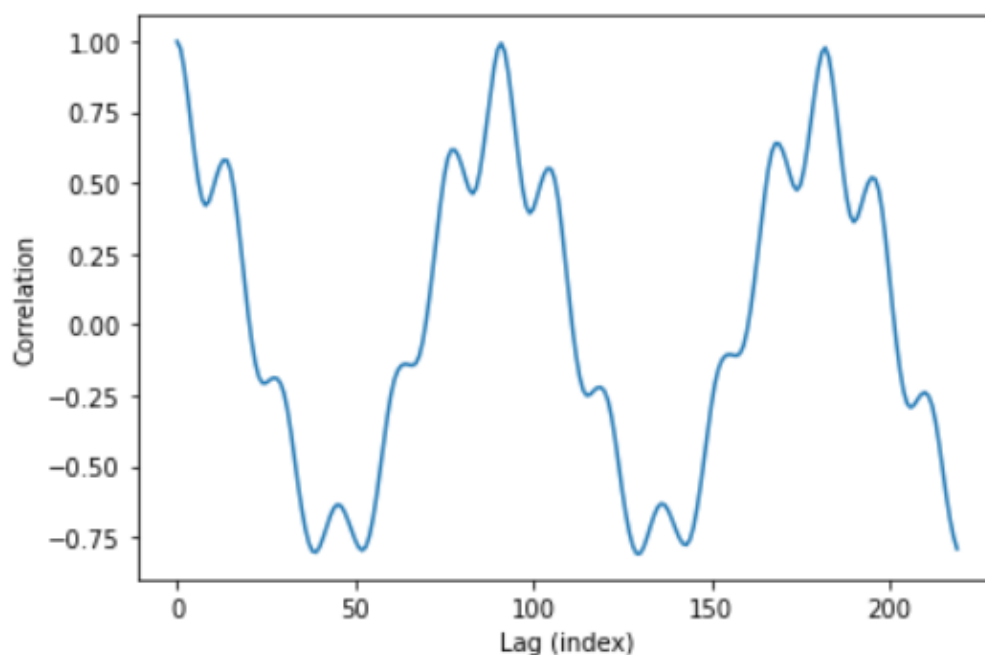


Рис. 1.1: Автокорреляция первого сегмента

Первый пик находится близко к `lag = 100`. Уточним значение, используя `argmax`.

```
low=70
high=150
lag = np.array(corr[low:high]).argmax() + low
lag
```

Листинг 1.3: Уточнение значения lag

Полученное значение lag = 91.

Затем вычислим частоту для найденного lag.

```
period = lag / segment.framerate
frequency = 1 / period
frequency
```

Листинг 1.4: Нахождение частоты

Полученная частота: 485 Гц.

Возьмем теперь другой сегмент того же сигнала и выполним те же действия (Рис.1.2).

```
segment2 = wave.segment(start=0.6, duration=0.01)
```

Листинг 1.5: Получение сегмента

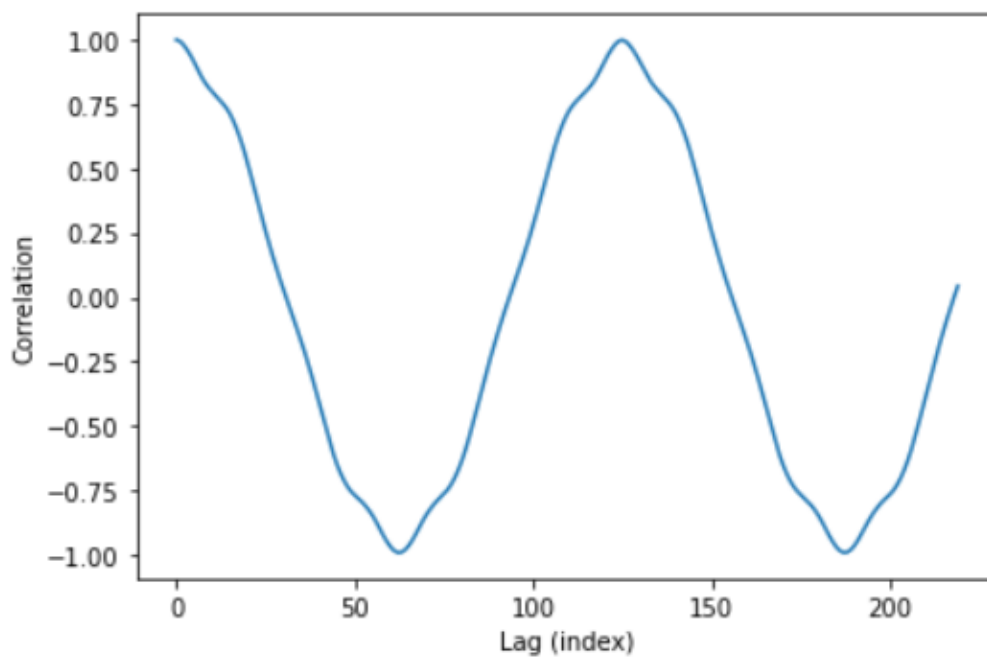


Рис. 1.2: Автокорреляция второго сегмента

Полученное значение lag = 125. Полученная частота: 353 Гц.

Таким образом, при увеличении времени начала сегмента частота уменьшается.

Глава 2

Упражнение 5.2

2.1 Функция `estimate_fundamental`

Пример кода в `chap05.ipynb` показывает, как использовать автокорреляцию для оценки основной частоты периодического сигнала. Инкапсулируем этот код в функцию, названную `estimate_fundamental`.

```
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 2.1: Функция `estimate_fundamental`

Используем ее для отслеживания высоты тона записанного звука.

```
wave = read_wave('28042__bcjordan__voicedownbew.wav')
wave.normalize()
segment = wave.segment(start=0.6, duration=0.01)
freq = estimate_fundamental(segment)
freq
```

Листинг 2.2: Применение функции

Полученная частота: 353 Гц.

2.2 Спектрограмма

Проверим ее работу функции.

В начале построим спектрограмму записи (Рис.2.1).

```
wave.make_spectrogram(2048).plot(high=4200)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 2.3: Построение спектрограммы

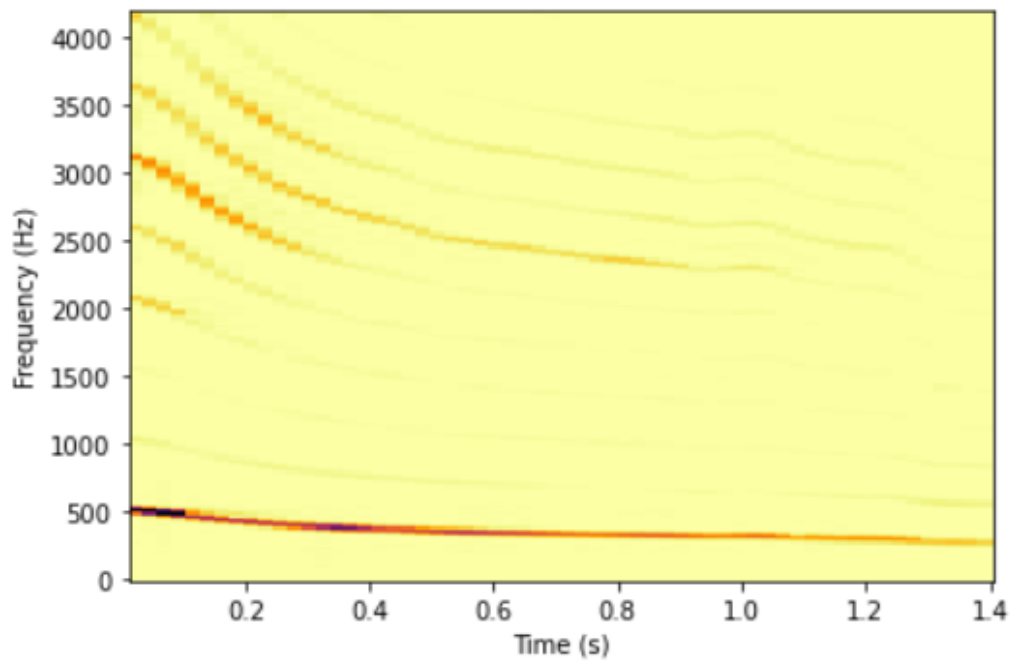


Рис. 2.1: Спектрограмма сегмента

Теперь будем накладывать оценки высоты на спектрограмму.

```
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')
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 2.4: Наложение оценки высоты на спектрограмму

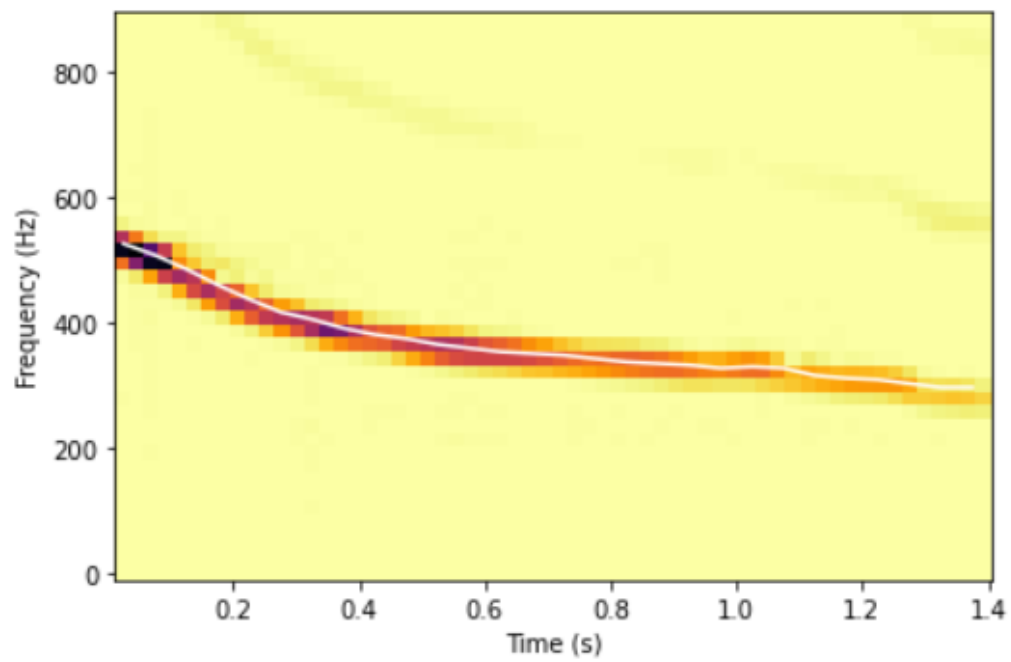


Рис. 2.2: Спектрограмма сегмента

Как мы можем видеть из рис.2.2, наложение оценки совпадает с кривой на спектрограмме, что говорит нам о правильной работе функции.

Глава 3

Упражнение 5.3

Используя данные о BitCoin из предыдущей лабораторной (Рис.3.1), вычислим автокорреляцию цен в платежной системе BitCoin.

```
import pandas as pd
from thinkdsp import Wave

df = pd.read_csv('files/BTC_USD_2020-04-12_2021-04-11-CoinDesk.csv',
                 parse_dates=[0])
ys = df['Closing Price (USD)']
ts = df.index
wave = Wave(ys, ts, framerate=1)
wave.plot()
decorate(xlabel='Time (days)')
```

Листинг 3.1: Получение данных о BitCoin

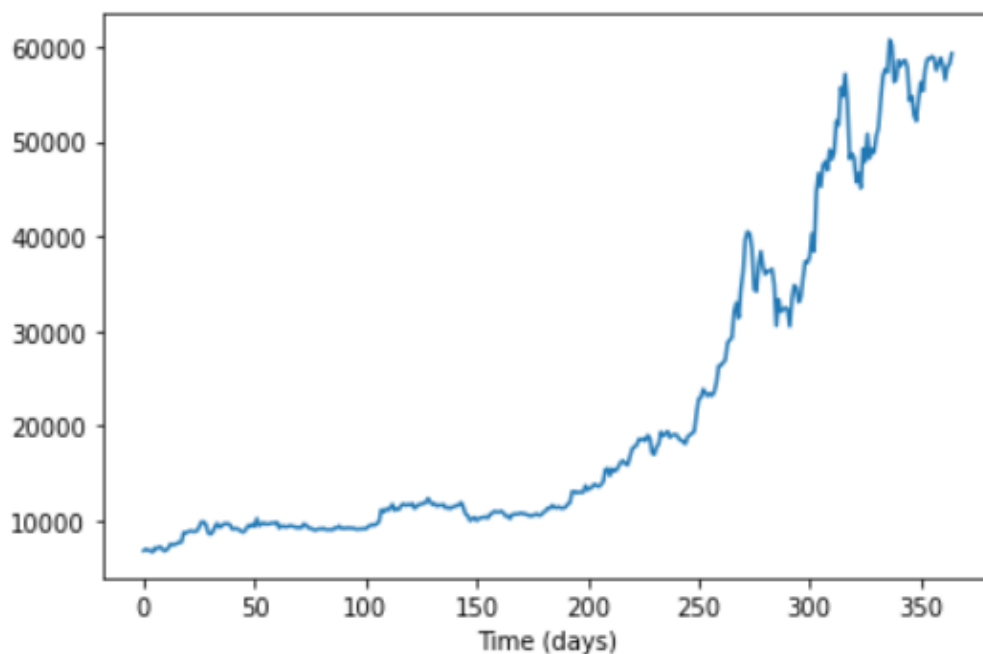


Рис. 3.1: Данные о BitCoin

Вычислим автокорреляцию.

```
lags, corrs = autocorr(wave)
plt.plot(lags, corrs)
decorate(xlabel='Lag', ylabel='Correlation')
```

Листинг 3.2: Вычисление автокорреляции

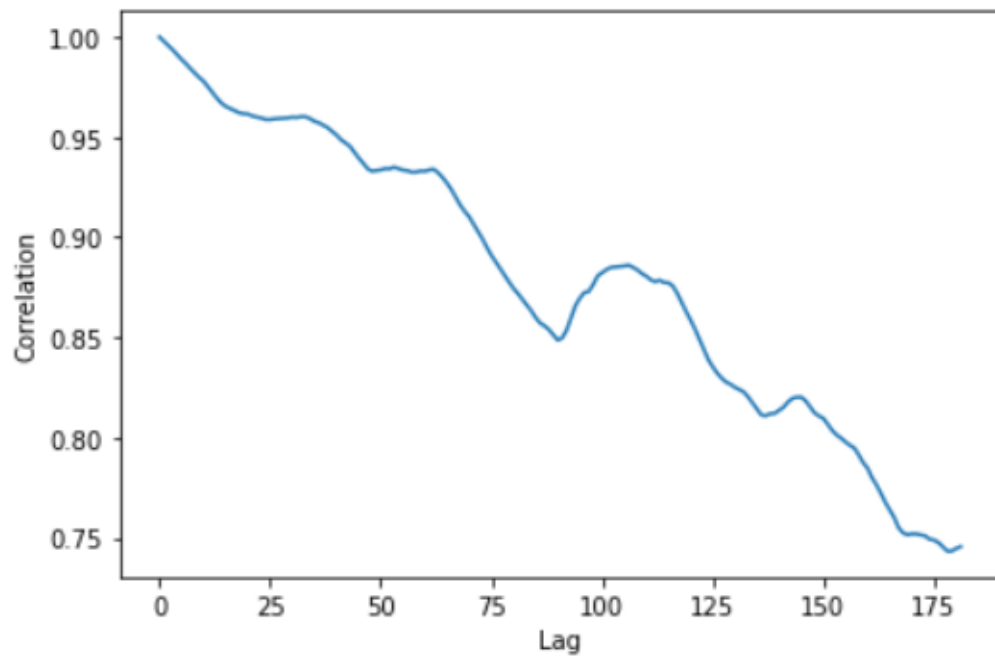


Рис. 3.2: Автокорреляция BitCoin

Как мы можем видеть на рис.3.2, автокорреляция спадает не очень быстро по мере увеличения задержки, что указывает на то, что перед нами розовый шум.

Глава 4

Упражнение 5.4

Воспользуемся блокнотом `saxophone.ipynb`, в котором исследуются автокорреляция, восприятие высоты тона и явление под названием «подавленная основная». Запустим примеры из него. Затем выберем другой сегмент записи и вновь поработаем с примерами. Пусть новый сегмент будет иметь ту же длину, но начнётся с 5 с.

```
start = 5.0
duration = 0.5
segment = wave.segment(start=start, duration=duration)
segment.make_audio()
```

Листинг 4.1: Выделение сегмента

Построим спектр этого сегмента (Рис.4.1).

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

Листинг 4.2: Построение спектра сегмента

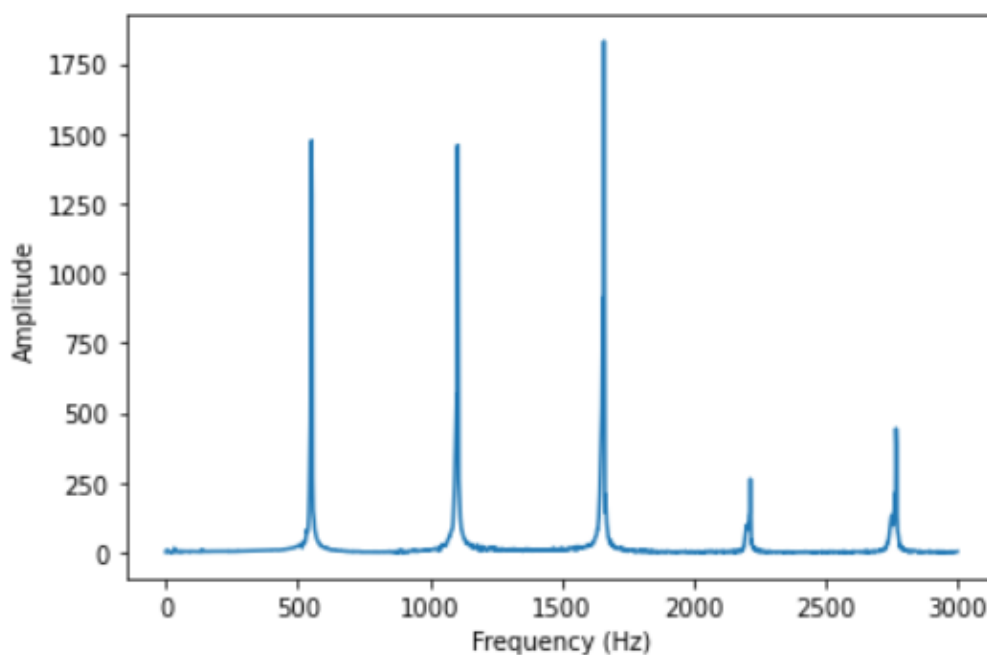


Рис. 4.1: Спектр сегмента

Затем получим все пики спектра. Пики в спектре находятся на частотах 552, 1106 и 1660 Гц.

Высота 552 Гц, которую мы воспринимаем, является основной, но не доминирующей. Сравним ее с треугольной волной на частоте 552 Гц.

```
from thinkdsp import TriangleSignal
TriangleSignal(freq=552).make_wave(duration=0.5).make_audio()
```

Листинг 4.3: Получение треугольной волны

У них одинаковая воспринимаемая высота звука.

Чтобы понять, почему мы воспринимаем основную частоту, даже если она не является доминирующей, используем автокорреляцию.

```
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 4.4: Функция автокорреляции

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

Листинг 4.5: Применение функции

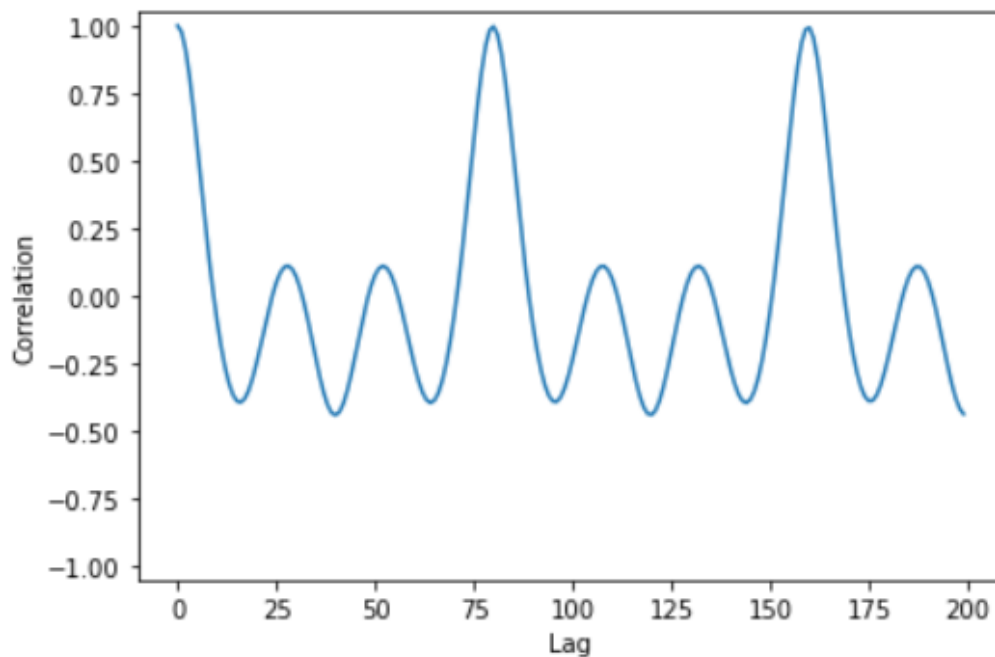


Рис. 4.2: Автокорреляция

Из рис.4.2 мы видим, что первый главный пик находится рядом с $\text{lag} = 75$.

Затем используем функцию `find_frequency`, которая находит самую высокую корреляцию в заданном диапазоне задержек и возвращает соответствующую частоту. Находим lag самого высокого пика и его частоту.

Полученный значения: lag = 80, частота = 551 Гц.

Высота звука, которую мы воспринимаем, соответствует наивысшему пику автокорреляционной функции, а не самому высокому компоненту спектра.

Теперь попробуем удалить основную частоту (Рис.4.3).

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

Листинг 4.6: Удаление основной частоты

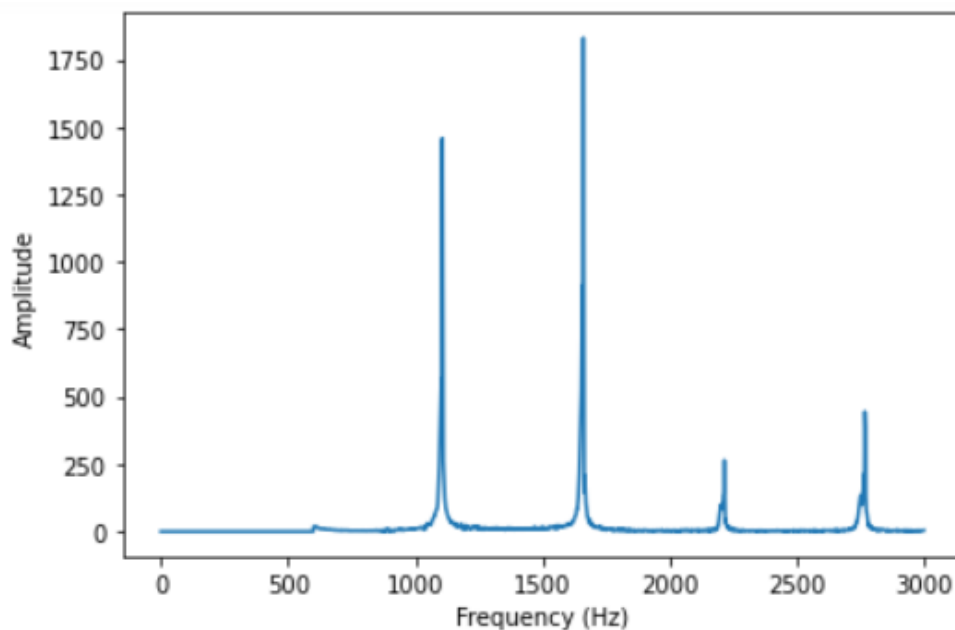


Рис. 4.3: Спектр без основной частоты

Послушаем полученный сегмент. Воспринимаемая высота звука по-прежнему составляет 551 Гц, хотя на этой частоте нет мощности. Это явление называют «подавленная основная».

Чтобы понять, почему мы слышим частоту, которой нет в сигнале, посмотрим на функцию автокорреляции (Рис.4.4).

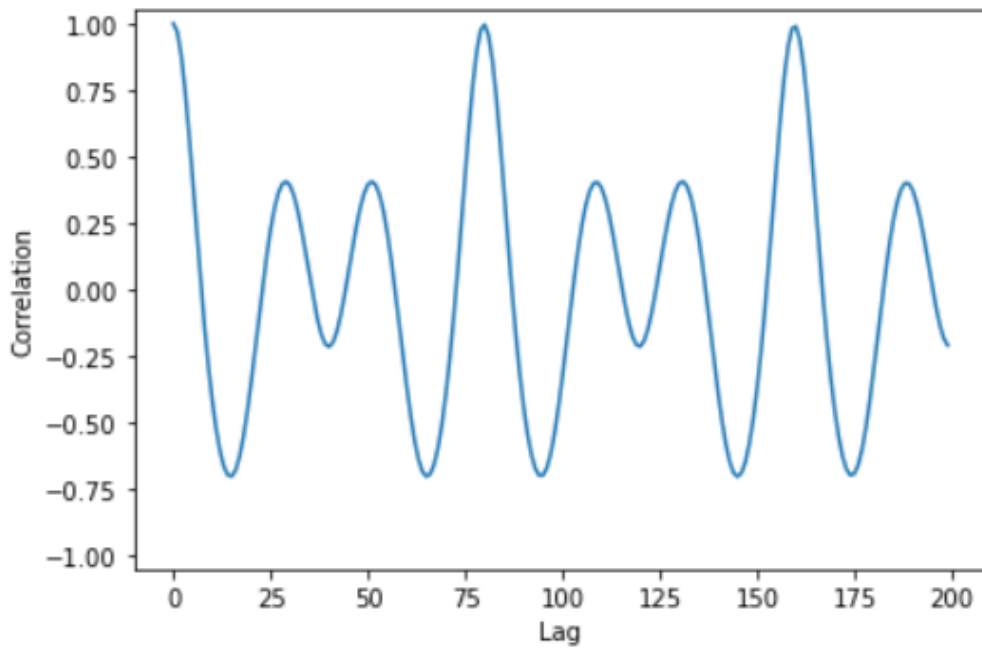


Рис. 4.4: Автокорреляция

Третий пик, соответствующий 551 Гц, по-прежнему самый высокий.

Но есть еще два пика, соответствующие 1521 Гц и 558 Гц. Но мы их не воспринимаем, так как высшие компоненты, которые присутствуют в сигнале, представляют собой гармоники 551 Гц, а не гармоники 558 или 1521 Гц.

Наше ухо интерпретирует высокие гармоники как свидетельство того, что «правильная» основная частота составляет 551 Гц.

Если избавиться от высоких гармоник, эффект исчезнет. Вот спектр с удаленными гармониками выше 1200 Гц (Рис.4.5).

```
spectrum4 = segment.make_spectrum()
spectrum4.high_pass(600)
spectrum4.low_pass(1200)
spectrum4.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 4.7: Фильтрация гармоник

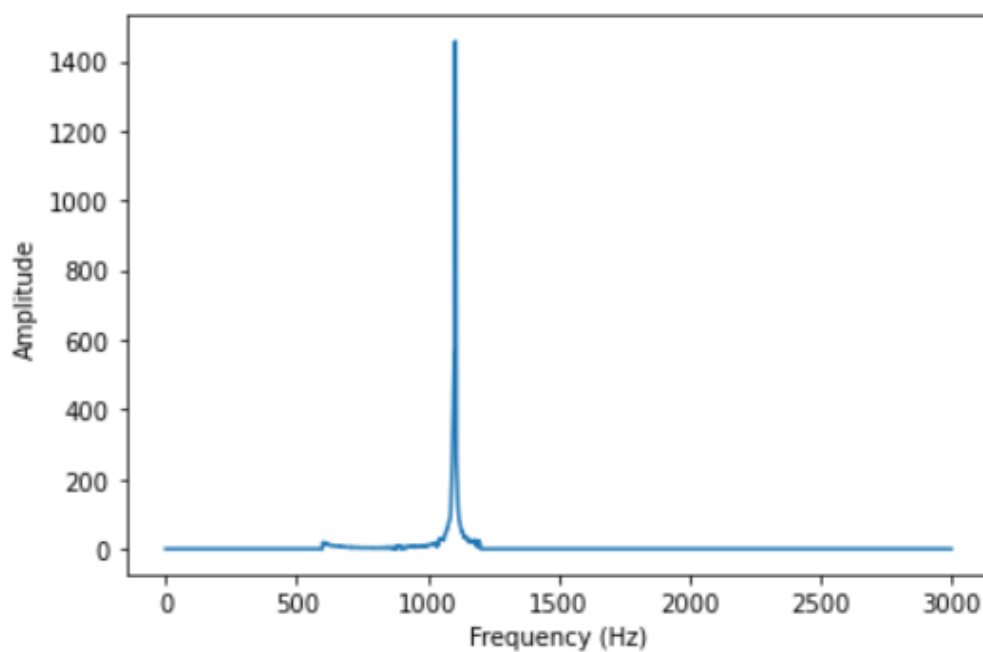


Рис. 4.5: Фильтрованный спектр

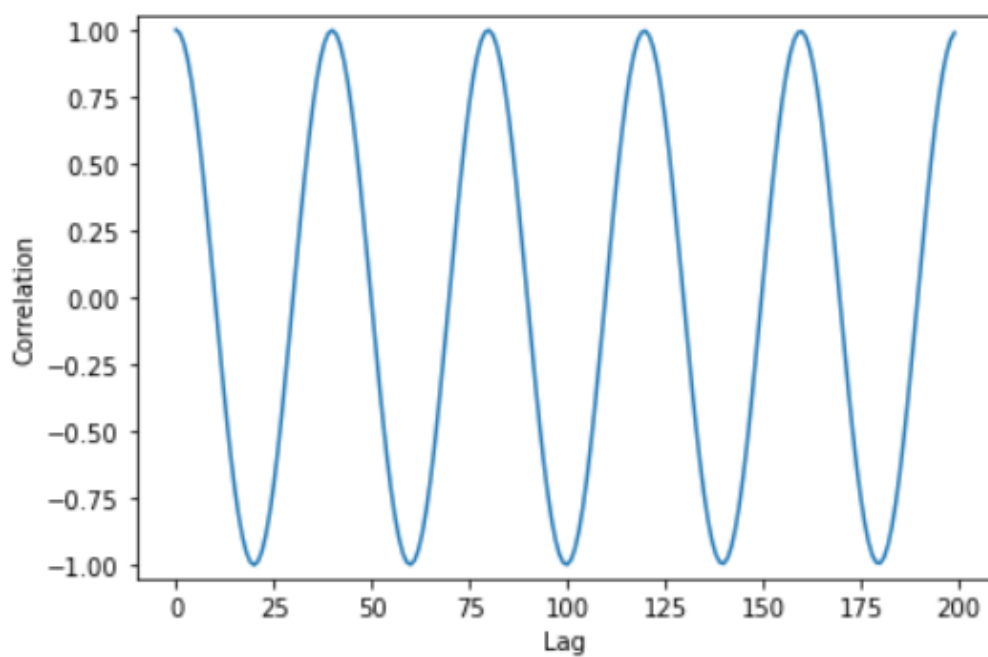


Рис. 4.6: Автокорреляция фильтрованного спектра

Теперь воспринимаемая высота звука 1106 Гц. Если мы посмотрим на функцию автокорреляции (Рис.4.6), то самый высокий пик будет на $\text{lag} = 40$, что соответствует 1102 Гц.

Таким образом, эти эксперименты показывают, что восприятие высоты звука не полностью основано на спектральном анализе, но также определяется автокорреляцией.

Глава 5

Выводы

В результате выполнения данной работы мы познакомились с понятием автокорреляции и научились использовать ее для оценки высоты тона.

Также мы исследовали понятие «подавленная основная».