Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

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

Выполнил студент гр. 3530901/80201 В.А. Пучкина

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение 5.1

Это упражнение направлено на знакомство с вычислением автокорреляции. Необходимо на основе информации из файла `chap05.ipynb` оценить высоты тона вокального `chirp` для нескольких времён начала сегмента.

Сначала определим методы, описанные в `chap05.ipnyb`.

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

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

Листинг 1: Методы `serial_corr` и `autocorr`.

Теперь считаем файл с вокальным `chirp` и выберем сегмент.

```
voice_wave = read_wave('resources/Sounds/task1_bcjordan__voicedownbew.wav')
voice_wave.normalize()
voice_segment = voice_wave.segment(start=0.3, duration=0.01)
voice_wave.make_audio()
```

Листинг 2: Чтение файла и выбор сегмента.

Получим автокорреляционную функцию выбранного сегмента с помощью `autocorr`.

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

Листинг 3: Получение автокорреляционной функции выбранного сегмента.

Из Рис.1 видно, что первый пик находится примерно при lag = 110. Найдем точный индекс и определим частоту.

```
lag = numpy.array(corrs[100:130]).argmax() + 100
period = lag / voice_segment.framerate
freq = 1 / period
print("lag =", lag, "\nfreq =", freq)
```

Листинг 4: Поиск пика и вычисление его частоты.

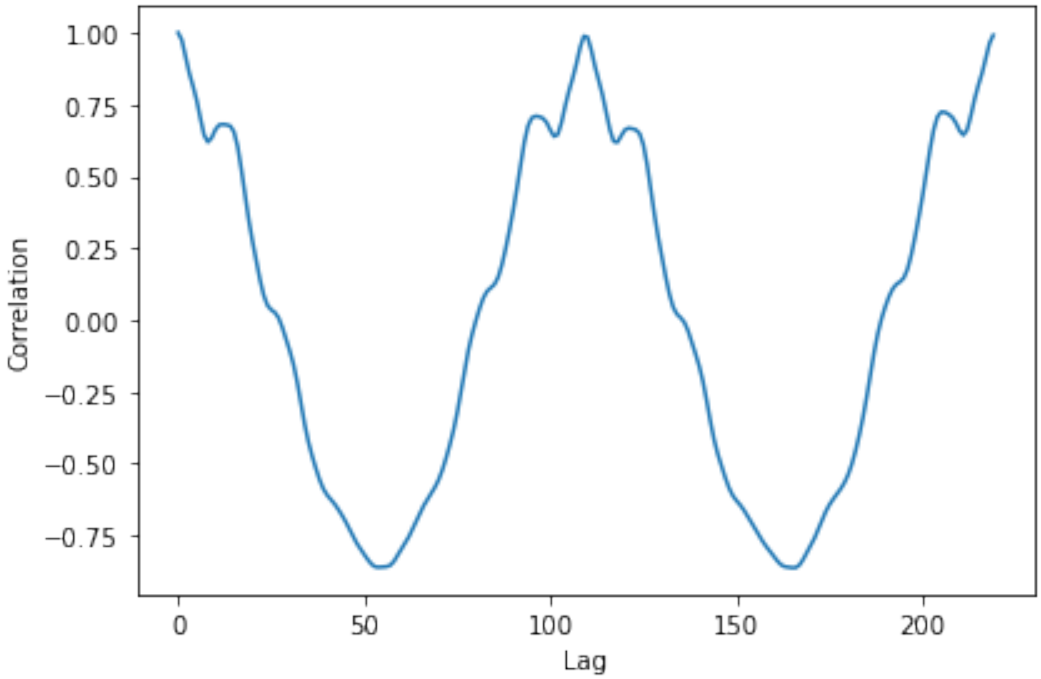

Рис. 1: Автокорреляционная функция для выбранного сегмента.

Были получены следующие результаты: lag = 109, частота = 404.6. Теперь выберем следующий сегмент и проделаем с ним те же действия.

```
voice_segment = voice_wave.segment(start=0.7, duration=0.01)
lag, corrs = autocorr(voice_segment)
plt.plot(lag, corrs)
decorate(xlabel='Lag', ylabel='Correlation')
```

Листинг 5: Получение автокорреляционной функции второго сегмента.

На Рис.2 видно, что первый пик находится примерно при lag = 130. Найдем точный индекс и определим частоту.

```
lag = numpy.array(corrs[120:140]).argmax() + 120
period = lag / voice_segment.framerate
freq = 1 / period
print("lag =", lag, "\nfreq =", freq)
```

Листинг 6: Поиск пика и вычисление его частоты.

Были получены следующие результаты: lag = 127, частота = 347.2. Таким образом, частота вокального `chirp` уменьшается с течением времени.

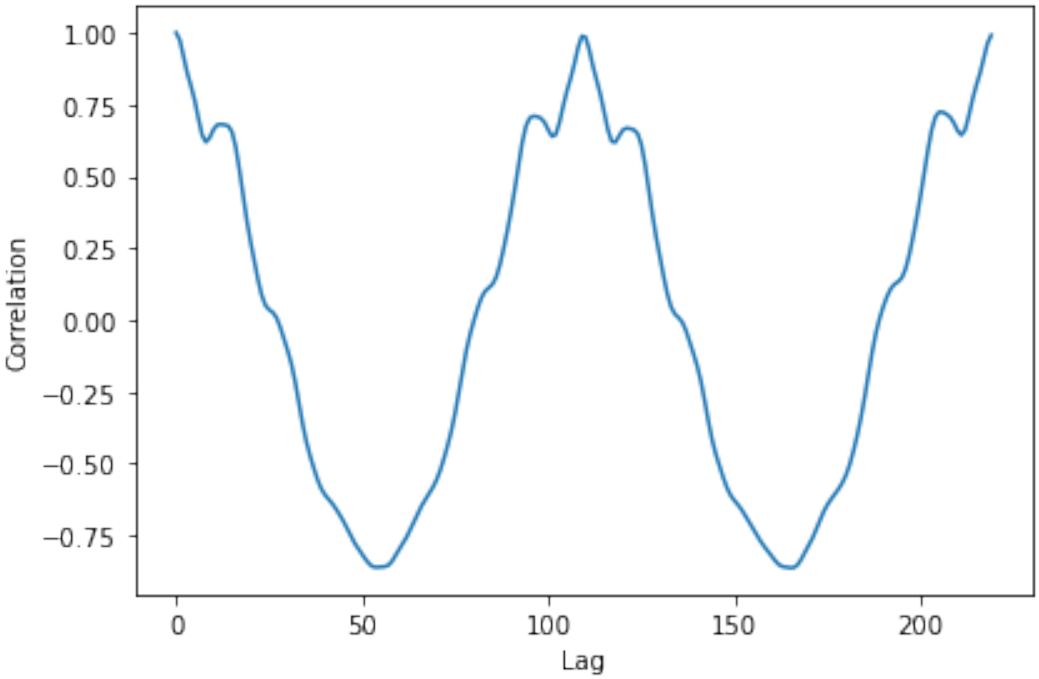

Рис. 2: Автокорреляционная функция для выбранного сегмента.

В ходе данного упражнения был получены автокорреляционные функции для двух сегментов записи вокального `chirp`. На них были выделены пики и получены их частоты. Был сделан вывод, что частота этого вокального `chirp` уменьшается с течением времени.

2 Упражнение 5.2

В этом упражнении необходимо выделить в отдельную функцию с названием `estimate_fundamental` код из `chap05.ipnyb`, использующий автокорреляцию для оценки основной частоты периодического сигнала. Затем следует использовать эту функцию для отслеживания высоты тона записанного звука и проверить качество её работы, накладывая оценки высоты тона на спектрограмму записи.

Итак, выделим необходимый код в функцию `estimate_fundamental`.

```
def estimate_fundamental(wave, low = 50, high = 150):
    lags, corrs = autocorr(wave)
    lag = numpy.array(corrs[low:high]).argmax() + low
    period = lag / wave.framerate
    freq = 1 / period
    return freq
```

Листинг 7: Функция `estimate_fundamental`.

Теперь проверим работу этой функции на примере записи из § 1. Для этого используем это функцию и будем накладывать её оценки высоты тона на спектрограмму записи.

Сначала посмотрим на спектрограмму записи.

```
voice_wave.make_spectrogram(2048).plot(5000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 8: Полуение спектрограммы.

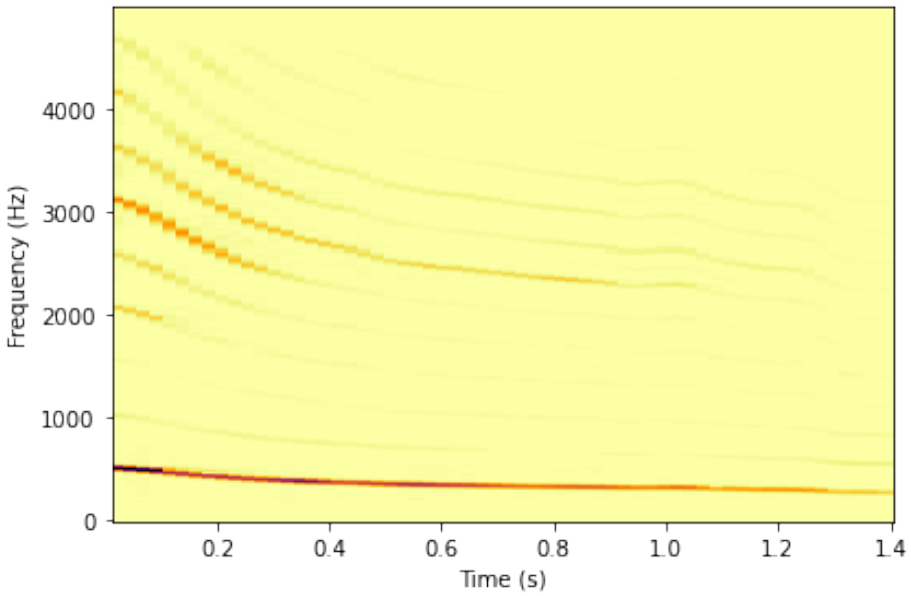

Рис. 3: Спектрограмма записи.

Теперь используем написанную функцию для получения оценок высоты.

```
ts = []
freqs = []
step = 0.05
starts = numpy.arange(0.0, 1.4, step)

for start in starts:
    ts.append(start + step / 2)
    segment = voice_wave.segment(start=start, duration=0.01)
    freq = estimate_fundamental(segment)
    freqs.append(freq)
```

Листинг 9: Полуение оценок высоты тона.

Теперь нанесём полученные оценки на спектрограмму и посмотрим на результат.

```
voice_wave.make_spectrogram(2048).plot(1000)
plt.plot(ts, freqs, color='white')
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 10: Сравнение полученных оценок со спектрограммой.

Из полученный спектрограммы (Рис.4) видно, что `estimate_fundamental` выдаёт корректные оценки основной частоты периодического сигнала.

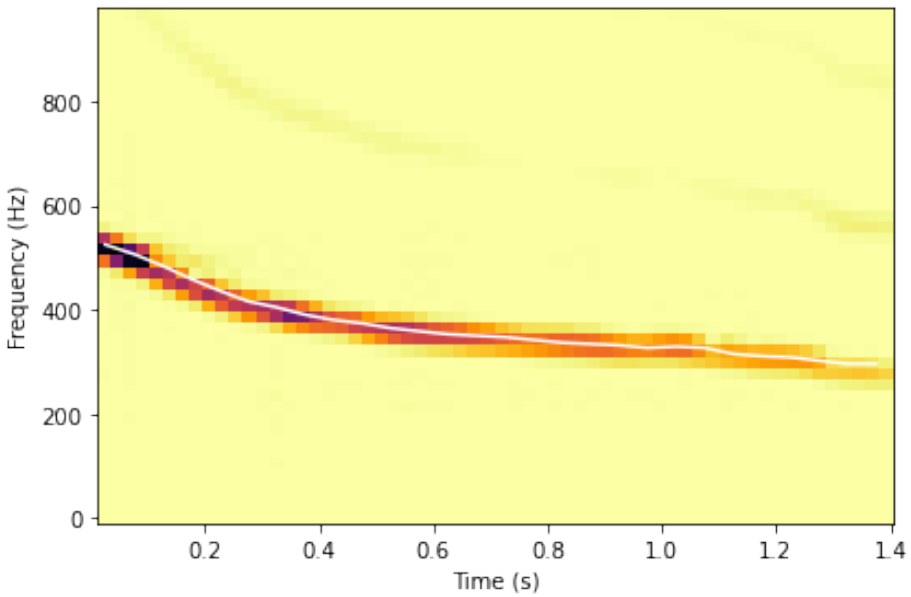

Рис. 4: Спектрограмма тона с наложенными оценками (белая линия).

В ходе выполнения данного упражнения на основе кода из `chap05.ipnyb` была написана функция `estimate_fundamental` для оценки основной частоты периодического сигнала. Она также была протестирована на записи вокального `chirp` для отслеживания высоты тона записанного звука. После чего полученные оценки высоты тона были наложены на спектрограмму. Из полученного результата был сделан вывод, что написанная функция работает корректно.

3 Упражнение 5.3

В данном упражнении необходимо использовать данные о курсе BitCoin из 4 лабораторной работы для вычисления автокорреляции цен BitCoin.

Итак, считаем файл и преобразуем данные в `wave`.

```
df = pandas.read_csv('resources/task3_BTC_USD_2013-10-01_2021-05-07.csv',
    parse_dates=[0])
ys = df['Closing Price (USD)']
ts = df.index

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

Листинг 11: Чтение файла и получение `wave`.

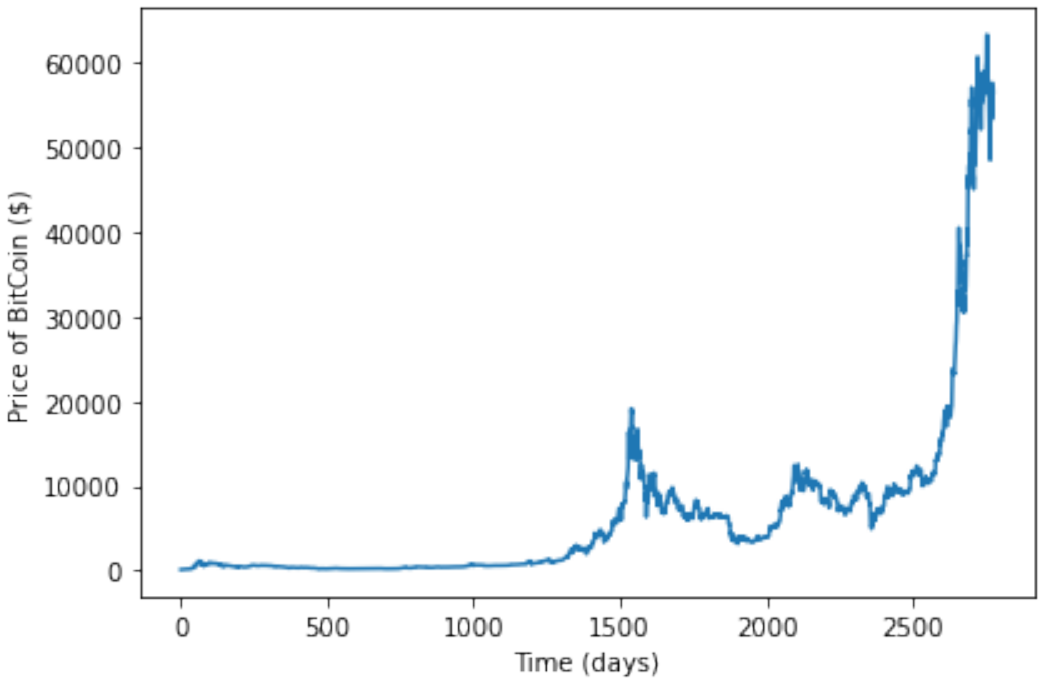

Рис. 5: Курс BitCoin.

Теперь получим автокорреляционную функцию и рассмотрим её.

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

Листинг 12: Получение автокорреляционной функции для курса BitCon.

По полученному графику (Рис.6) видно, что автокорреляционная функция не имеет периодичности. Кроме того, она имеет как спады, так и подъёмы.

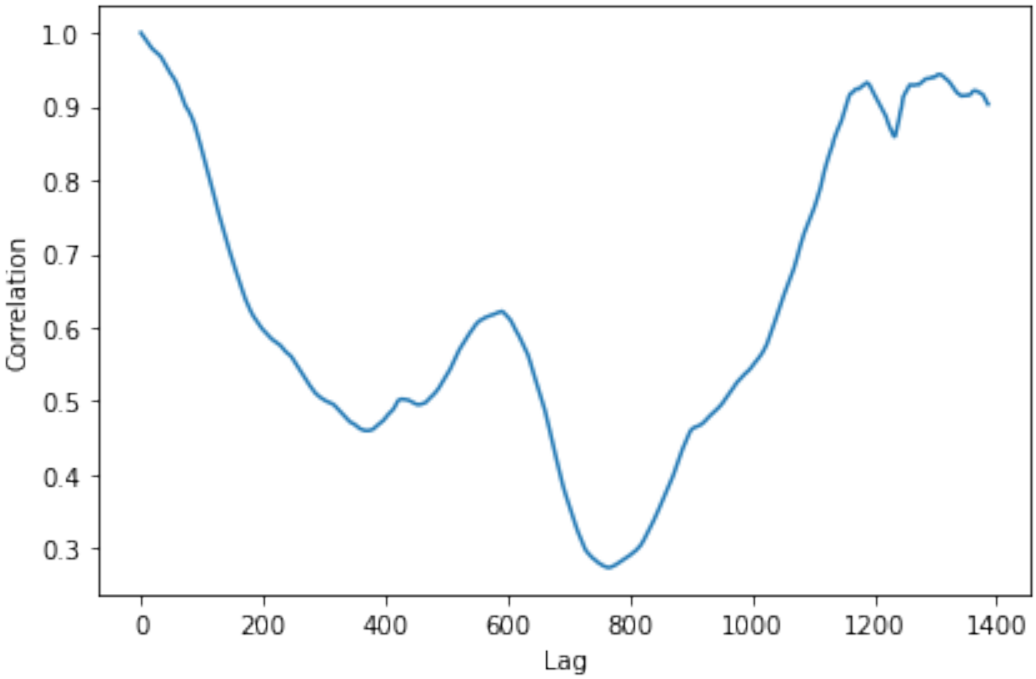

Рис. 6: Автокорреляционная функция курса BitCoin.

В ходе выполнения данного упражнения был проведён эксперимент с курсом BitCoin. На основе курса была получена `wave`, после чего вычислена автокорреляционная функция. Был сделан вывод, что автокорреляционная функция курса BitCoin не периодическая и имеет как спады, так и падения.

4 Упражнение 5.4

В данном упражнении необходимо изучить файл `saxophone.ipynb` и «погонять» примеры. В этом файле исследуются автокорреляция, восприятие высоты тона и явление «подавленная основная». После изучения примеров следует выбрать другой сегмент записи и вновь поработать с ним.

Итак, для начала считаем файл и получим его спектрограмму.

```
sax_wave = read_wave('resources/Sounds/task4_iluppai__saxophone_weep.wav')
sax_wave.normalize()
sax_wave.make_spectrogram(seg_length=1024).plot(high=3000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
sax_wave.make_audio()
```

Листинг 13: Чтение файла и получение спектрограммы.

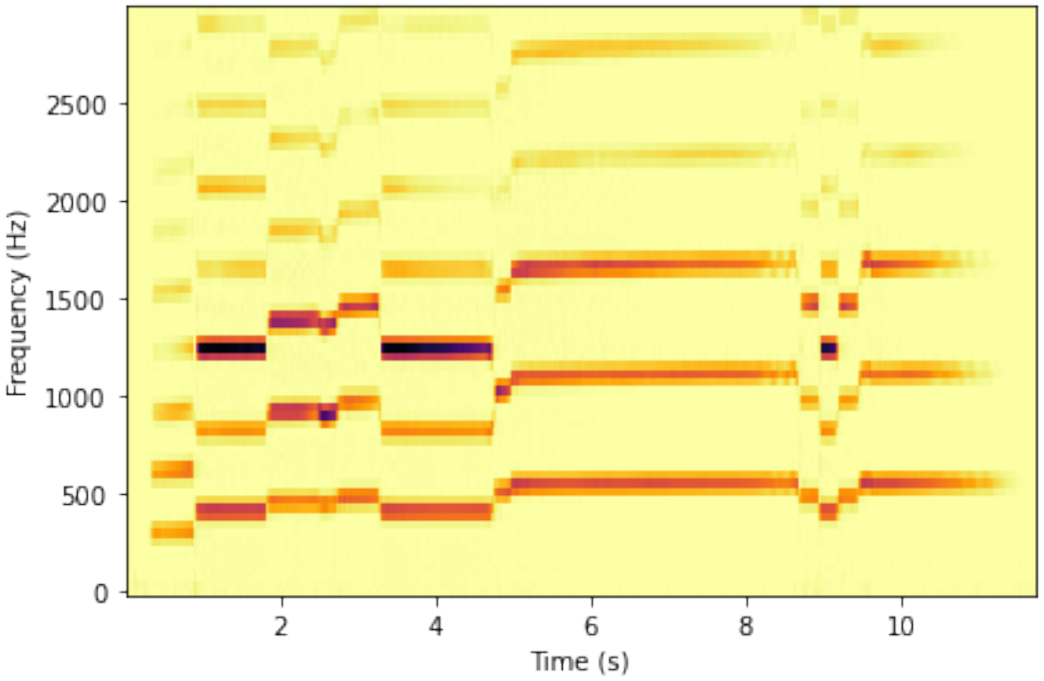

Рис. 7: Спектрограмма записи.

Теперь выберем сегмент записи, отличный от того, с которым происходит работа в примере.

```
sax_segment = sax_wave.segment(start=6, duration=0.5)
sax_spectrum = sax_segment.make_spectrum()
sax_spectrum.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
sax_segment.make_audio()
```

Листинг 14: Выбор сегмента и получение его спектра.

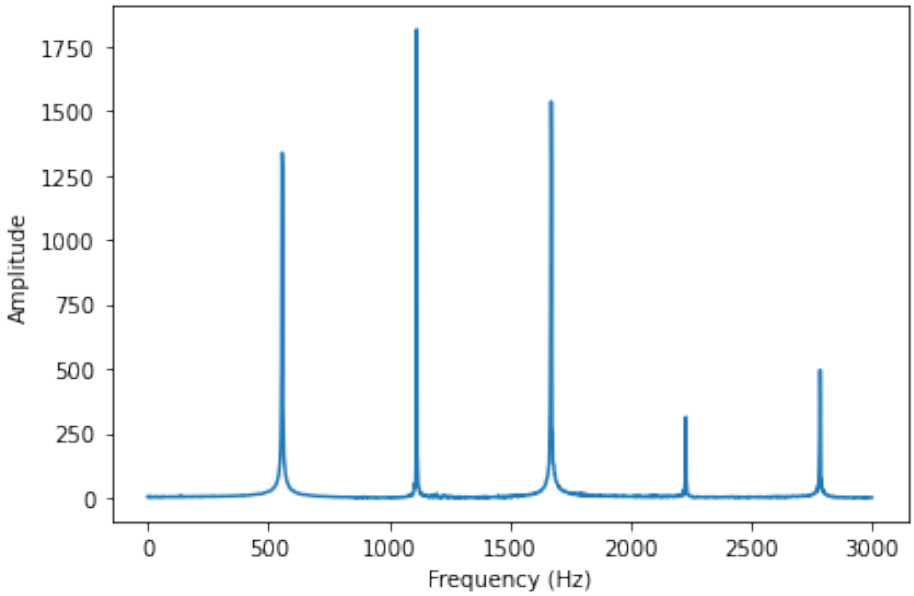

Рис. 8: Спектр выбранного сегмента.

Теперь узнаем все пики спектра.

```
sax_spectrum.peaks()[:10]
```

Листинг 15: Получение пиков спектра.

Три наивысших пика находятся на 1114, 1670 и 556 Гц. Из спектра (Рис.8) видно, что основная частота (556 Гц) не является доминирующей. Хотя именно её мы и воспринимаем. Для сравнения прослушаем звук треугольного сигнала с частотой 556 Гц.

```
TriangleSignal(freq=556).make_wave(duration=0.5).make_audio()
```

Листинг 16: Треугольный сигнал с частотой 556 Гц.

Если сравнить звук нашего сегмента и созданный треугольный сигнал, можно заметить, что воспринимаемая высота звука у них одинаковая.

Попробуем понять, почему мы воспринимаем основную частоту, даже если она не доминирующая. Для этого получим автокорреляционную функцию нашего сегмента.

```
def autocorr(segment):
    corrs = numpy.correlate(segment.ys, segment.ys, mode='same')
    N = len(corrs)
    lenght = range(N, N//2, -1)

    half = corrs[N//2:].copy()
    half /= lenght
    half /= half[0]
    return half
```

Листинг 17: Функция для получения автокорреляции.

```
sax_corrs = autocorr(sax_segment)
plt.plot(sax_corrs[:200])
decorate(xlabel='Lag', ylabel='Correlation')
```

Листинг 18: Получение автокорреляционной функции выбранного сегмента.

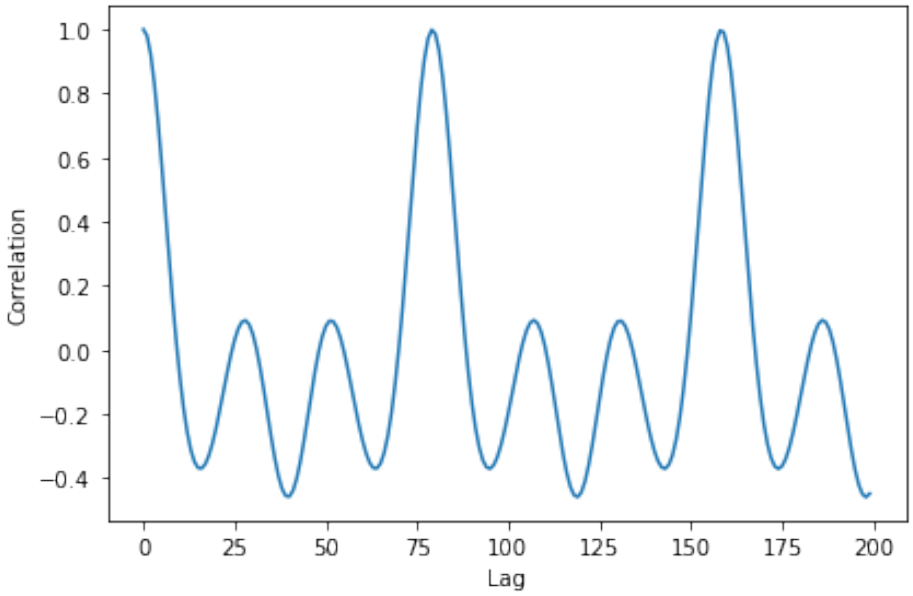

Рис. 9: Автокорреляционная функция сегмента.

На Рис.9 видно, что пик находится примерно при lag=75. Уточним это значение и получим соответствующую частоту.

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

Листинг 19: Функция для поиска самой высокой корреляции и её частоты.

```
find_frequency(sax_corrs, 70, 85)
```

Листинг 20: Поиск пика и его частоты.

Были получены значения: lag = 79, частота = 558 Гц.

Таким образом, воспринимаемая высота звука соответствует наивысшему пику автокорреляционной функции, а не доминирующей частоте спектра.

Попробуем убрать основную частоту.

```
sax_spectrum2 = sax_segment.make_spectrum()
sax_spectrum2.high_pass(700)
sax_spectrum2.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 21: Фильтрация частот ниже 700 Гц.

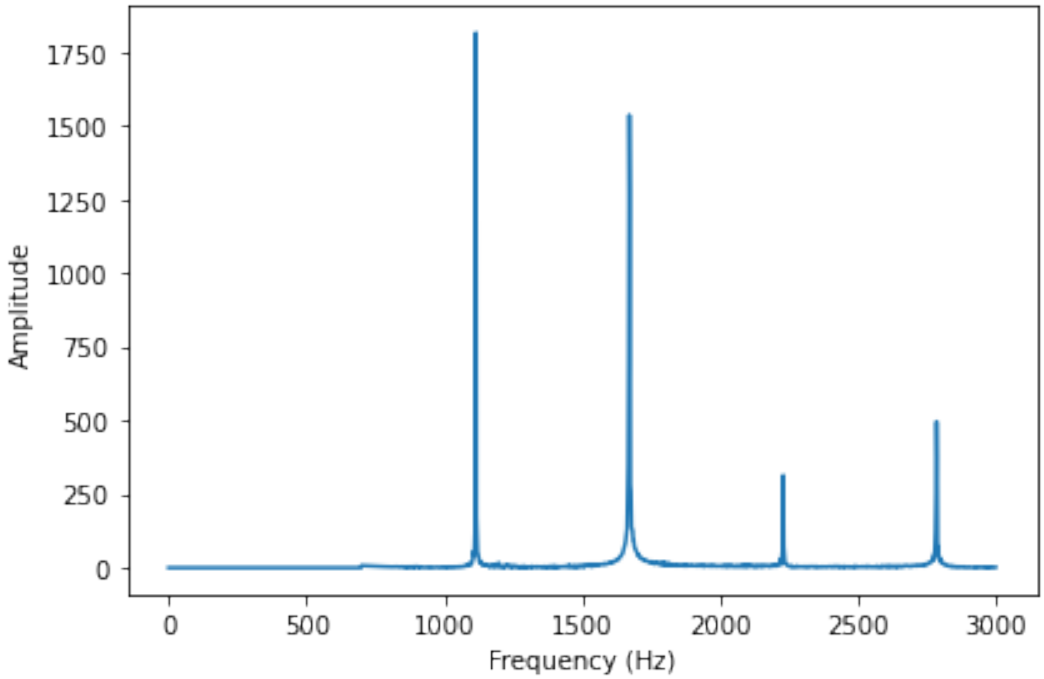

Рис. 10: Спектр сегмента после фильтрации.

Преобразуем полученный спектр в `wave` и прослушаем.

```
sax_segment2 = sax_spectrum2.make_wave()
sax_segment2.make_audio()
```

Листинг 22: Преобразование в `wave`.

Звук, конечно, стал более резким, однако воспринимаемая высота всё ещё совпадает с высотой оригинального сегмента, т.е. по прежнему составляет 556 Гц, несмотря на то что на этой частоте нет мощности. Это явление называется «Подавленная основная». Этот эффект может использоваться в звуковых системах для расширения области воспроизводимых низких частот, если нет возможности адекватно воспроизвести такие частоты напрямую (например, телефоны, наушники).

Попробуем понять, почему мы всё-таки слышим частоту, которой нет в сигнале. Для этого получим автокорреляционную функцию обновлённого сегмента.

```
sax_corrs = autocorr(sax_segment2)
plt.plot(sax_corrs[:200])
decorate(xlabel='Lag', ylabel='Correlation')
```

Листинг 23: Получение автокорреляционной функции обновлённого сегмента.

На Рис.11 опять можно наблюдать пик рядом с lag = 75. Уточним это значение и получим соответствующую частоту.

```
find_frequency(sax_corrs, 70, 85)
```

Листинг 24: Получение частоты пика.

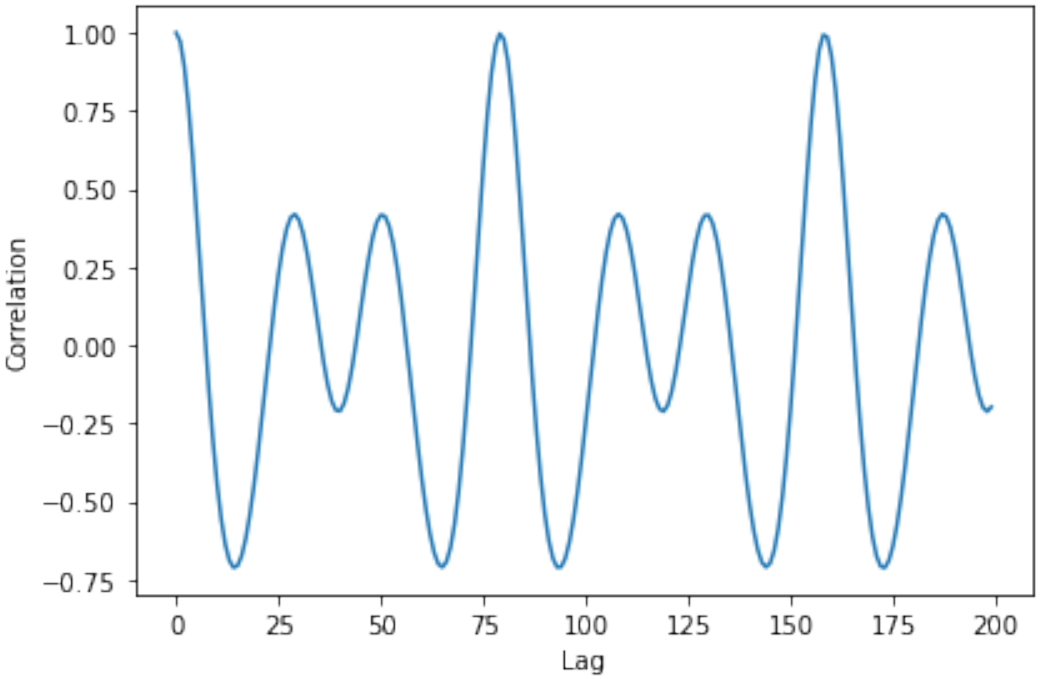

Рис. 11: Автокорреляционная функция обновлённого сегмента.

И мы вновь получили те же значения: lag = 79, частота = 558 Гц. На Рис.11 можно увидеть также ещё два пика: рядом с lag = 25 и lag = 50. Узнаем их частоты.

```
find_frequency(sax_corrs, 20, 40)
find_frequency(sax_corrs, 40, 60)
```

Листинг 25: Получение частот других пиков.

Были получены частоты 1520 и 882 Гц соответственно. Однако мы всё равно воспринимаем 558 Гц, а не их. Это связано с тем, что наивысшие пики, присутствующие в сигнале, являются гармониками 558 Гц, а не 1520 или 882 Гц. Наше ухо воспринимает высокие гармоники как подтверждение того, что «правильной» основной частотой является именно 558 Гц.

Попробуем избавиться от этих высоких гармоник.

```
sax_spectrum3 = sax_segment.make_spectrum()
sax_spectrum3.high_pass(700)
sax_spectrum3.low_pass(1400)
sax_spectrum3.plot(high=3000)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 26: Фильтрация высоких гармоник.

Теперь преобразуем отфильтрованный спектр в `wave` и прослушаем результат.

```
sax_segment3 = sax_spectrum3.make_wave()
sax_segment3.make_audio()
```

Листинг 27: Получение `wave`.

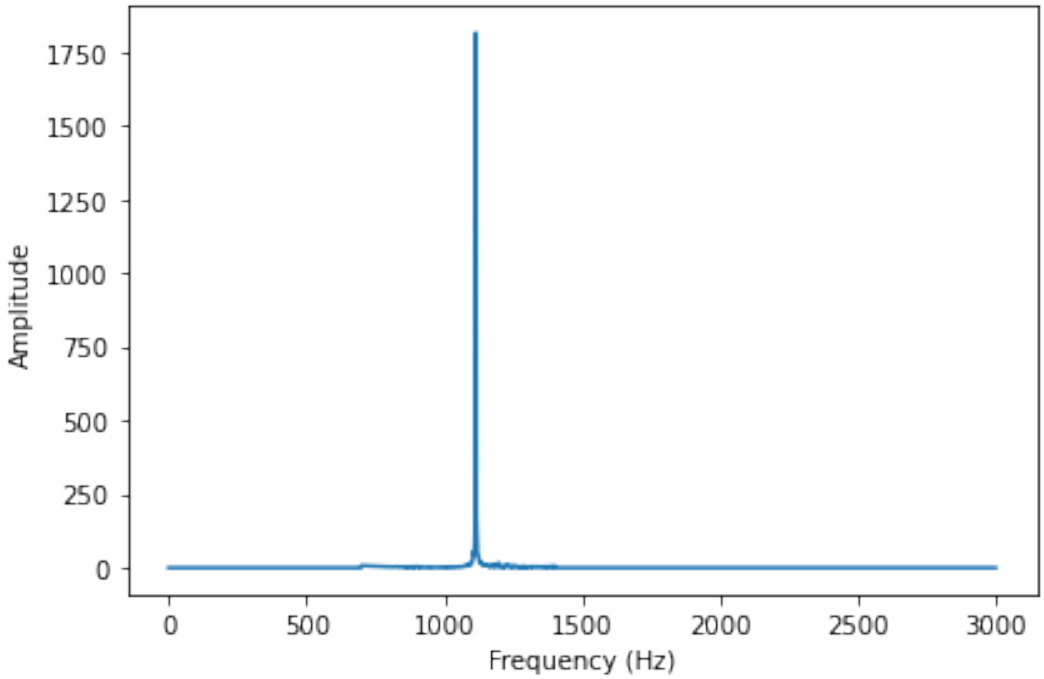

Рис. 12: Спектр сегмента без высоких гармоник.

Теперь воспринимаемая частота изменилась. Сравним с треугольным сигналом с частотой 1114 Гц.

```
TriangleSignal(1114).make_wave(duration=0.5).make_audio()
```

Листинг 28: Треугольный сигнал с частотой 1114 Гц.

Действительно, воспринимаемая высота звука теперь составляет 1114 Гц. Получим автокорреляционную функцию.

```
sax_corrs = autocorr(sax_segment3)
plt.plot(sax_corrs[:200])
decorate(xlabel='Lag', ylabel='Correlation')

```

Листинг 29: Получение автокорреляционной функции.

Самый высокий пик находится при lag = 40, что соответствует частоте 1102 Гц.

Таким образом, только удалив основную частоту и её гармоники, мы смогли изменить воспринимаемую высоту звука. Это доказывает, что восприятие высоты звука основано на спектральном анализе не полностью, а также определяется автокорреляцией.

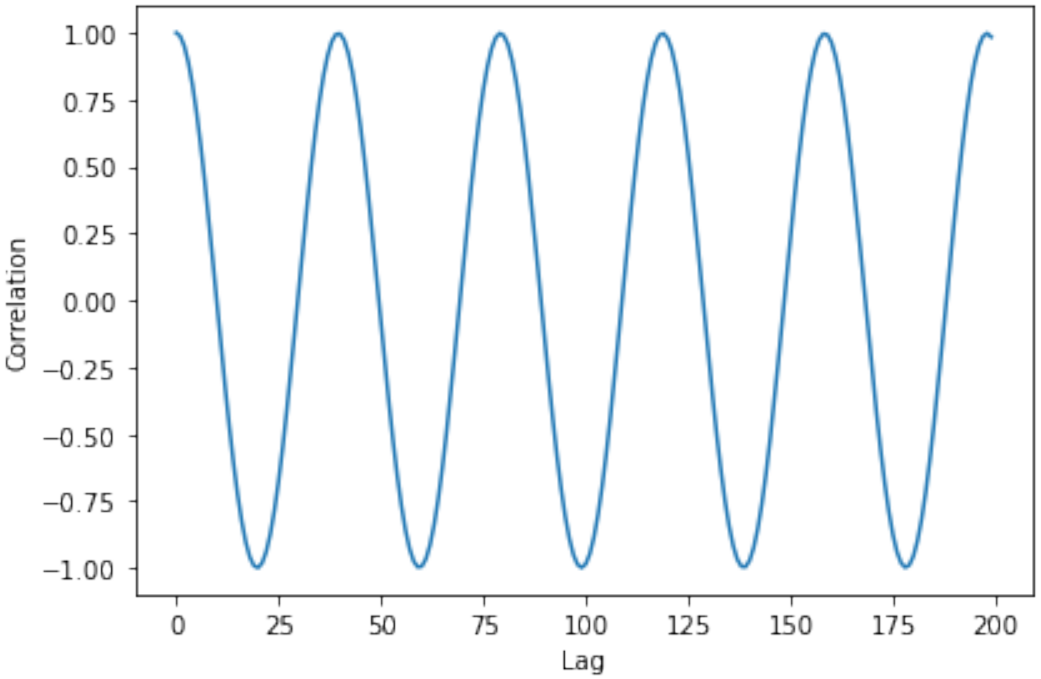

Рис. 13: Автокорреляционная функция сегмента с удалёнными гармониками.

В ходе данного упражнения было изучено явление «подавленная основная», из-за которой воспринимаемая высота тона может не меняться, если убрать основную частоту. Кроме того, для избавления от этого эффекта и смены воспринимаемой высоты звука были удалены и наивысшие гармоника. Также был сделан вывод, что восприятие высоты звука основано на спектральном анализе не полностью, а также определяется автокорреляцией.

5 Выводы

В ходе данной лабораторной работы были изучены возможности автокорреляции для оценки высоты тона звука. Кроме того, было исследование «подавленная основная», из-за которой воспринимаемая высота тона может не меняться, если убрать основную частоту. Этот эффект может использоваться для расширения области воспроизводимых низких частот, если нет возможности адекватно воспроизвести такие частоты напрямую (например, телефоны и наушники). Был сделан вывод, что восприятие высоты звука не полностью основано спектральном анализе, но и определяется автокорреляцией.