

Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Автокорреляция

Работу выполнил студент
3-го курса, группа 3530901/80201
Солянкин Илья Андреевич

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург 2021

Содержание

1	Часть №1: Оценка высоты тона вокального чирпа	5
2	Часть №2: Функция estimate-fundamental	7
3	Часть №3: BitCoin	10
4	Часть №4: saxophone.ipynb	12
5	Выводы	17

Список иллюстраций

1	Получение чирпа	5
2	Полученный график автокорреляции	6
3	Получение чирпа	7
4	Построенная спектограмма	8
5	Полученная частота минимального сегмента	8
6	Построенная спектограмма сегмента	9
7	Данные из файла в виде графика	10
8	Полученный график после автокорреляции	11
9	Получение звука саксофона	12
10	Полученная спектограмма	12
11	Получение сегмента из записи	13
12	Полученный спектр сегмента	13
13	Пики полученного спектра	13
14	Треугольный сигнал	14
15	Полученный график автокорреляции	14
16	Использование написанной функции	15
17	Полученный график после фильтрации	16
18	Получение аудио из отфильтрованного сегмента	16

Листинги

1	Функция <code>serial-corr</code>	5
2	Функция <code>autocorr</code>	5
3	Построение графика автокорреляции	5
4	Функция <code>estimate-fundamental</code>	7
5	Построение спектограммы	7
6	Нахождение нужного сегмента	8
7	Вывод спектограммы сегмента	8
8	Представление данных файла в виде графика	10
9	Получение нового графика после автокорреляции	10
10	Построение спектограммы	12
11	Получение спектра сегмента	13
12	Функция для автокорреляции	14
13	Получение графика автокорреляции	14
14	Функция для нахождения частоты	15
15	Фильтрация сегмента	15

1 Часть №1: Оценка высоты тона вокального чирпа

В первом пункте пятой лабораторной работы нам необходимо вычислить автокорреляцию для различных Lag и оценить высоты тона вокального чирпа.

Начнем с написания функции `serial_corr` и `autocorr`:

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

Листинг 1: Функция `serial_corr`

```
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
```

Листинг 2: Функция `autocorr`

После этого прочитам чирп с записью голоса и для проверки выведем его:

```
B [42]: wave = read_wave('28042_bcjordan_voicedownbew.wav')
        wave.normalize()
        wave.make_audio()
```

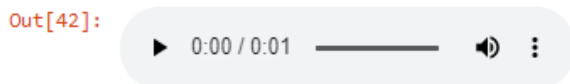


Рис. 1: Получение чирпа

После этого построим график автокорреляции:

```
1 segment = wave.segment(0, 0.01)
2 lags, corrs = autocorr(segment)
3 lagx = np.array(corrs[90:110]).argmax() + 90
4 thinkplot.plot(lags, corrs, color = 'blue')
5 thinkplot.config(xlabel='Lag', ylabel='Correlation')
```

Листинг 3: Построение графика автокорреляции

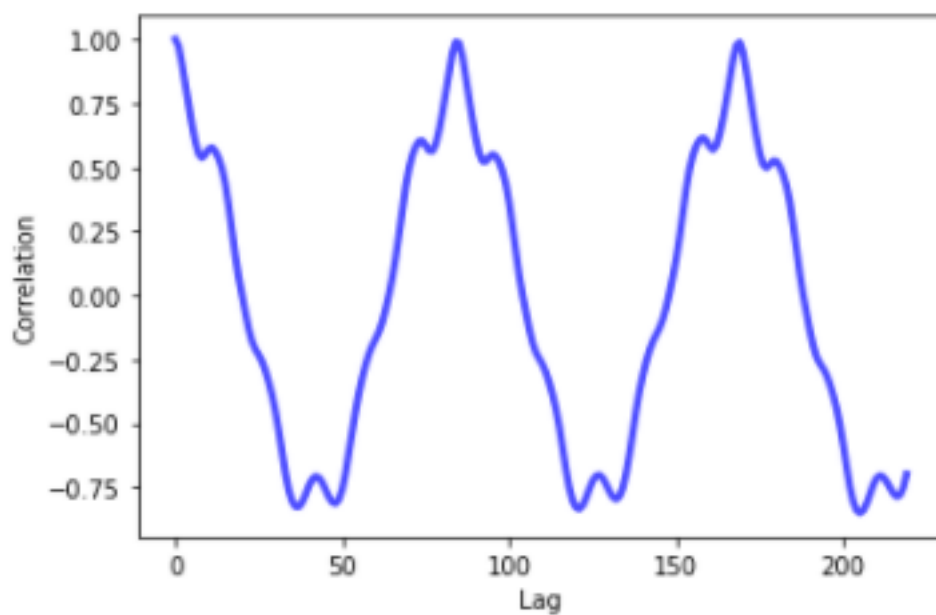


Рис. 2: Полученный график автокорреляции

По графику видно, что он периодический, период равен 90 Lag

2 Часть №2: Функция estimate-fundamental

Во втором пункте пятой лабораторной работы нам необходимо написать функцию `estimate-fundamental`, отслеживающую высоту тона записанного звука. Также необходимо проверить ее работоспособность.

Для начала напишем функцию `estimate-fundamental`, которая с помощью функции автокорреляции может помочь в отслеживании высоты тона:

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

Листинг 4: Функция `estimate-fundamental`

После этого возьмем запись чирпа:

```
B [7]: wave = read_wave('28042__bcjordan__voicedownbew.wav')
       wave.normalize()
       wave.make_audio()
```

Out[7]:



Рис. 3: Получение чирпа

Теперь построим спектограмму полученной записи:

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

Листинг 5: Построение спектограммы

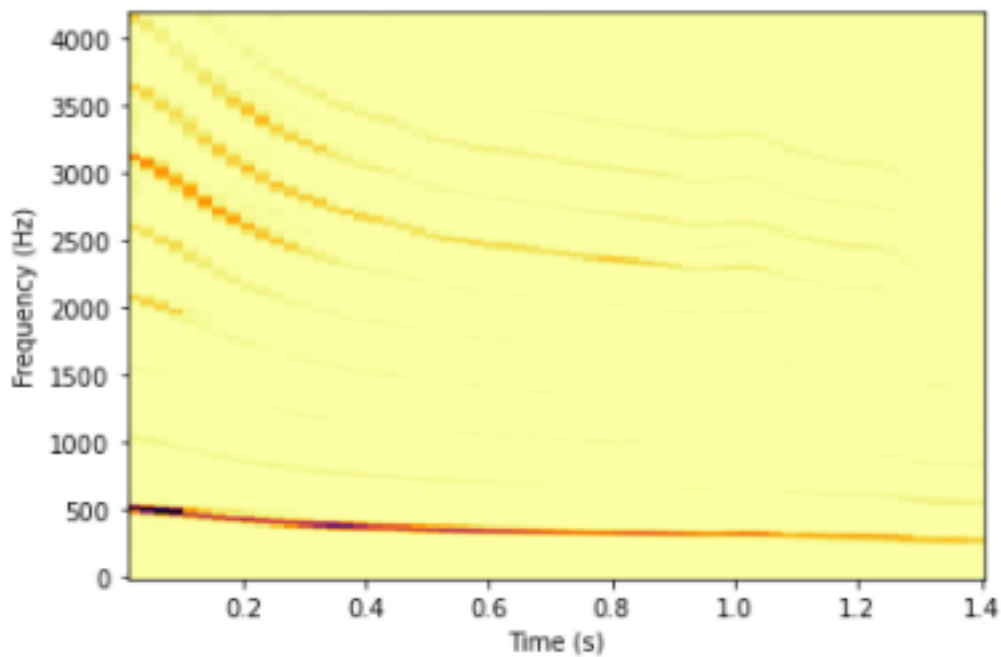


Рис. 4: Построенная спектограмма

Затем с помощью написанной ранее функции `estimate-fundamental` получим частоту минимального сегмента:

```
B [9]: segment = wave.segment(start=0.2, duration=0.01)
      freq = estimate_fundamental(segment)
      freq
Out[9]: 436.63366336633663
```

Рис. 5: Полученная частота минимального сегмента

После этого с шагом 0,05 найдем этот сегмент:

```
1 starts = np.arange(0.0, 1.4, 0.05)
2 ts = []
3 freqs = []
4 for start in starts:
5     ts.append(start + 0.05/2)
6     segment = wave.segment(start=start, duration=0.01)
7     freq = estimate_fundamental(segment)
8     freqs.append(freq)
```

Листинг 6: Нахождение нужного сегмента

И, наконец, выведем его спектограмму на экран:

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

Листинг 7: Вывод спектограммы сегмента

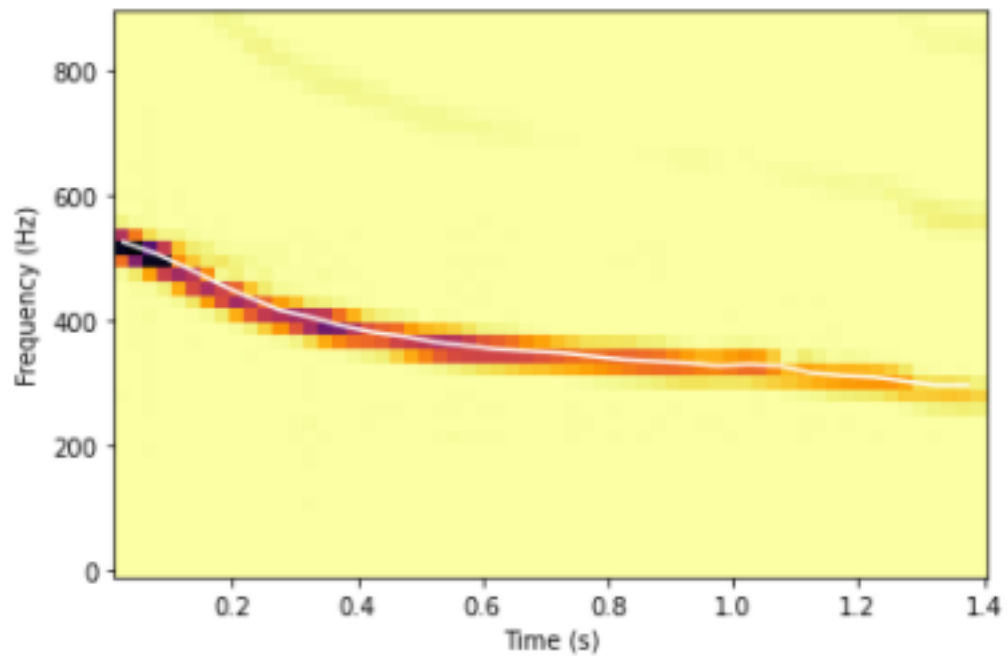


Рис. 6: Построенная спектограмма сегмента

В результате на полученной спектограмме сегмента можно увидеть частоту тона в каждый момент времени

3 Часть №3: BitCoin

В третьем пункте пятой лабораторной работы нам необходимо использовать данные цен BitCoin из прошлой лабораторной работы вычислить автокорреляцию цен на BitCoin.

Возьмем тот же файл, что и в прошлой лабораторной работе и так же представим данные из него в виде графика:

```
1 df = pd.read_csv('BTC_USD_2013-10-01_2020-03-26-CoinDesk.csv', parse_dates=[0])
2 ys = df['Closing Price (USD)']
3 ts = df.index
4 wave = Wave(ys, ts, framerate=1)
5 wave.plot()
6 decorate(xlabel='Time (days)', ylabel='Price of BitCoin ($)')
```

Листинг 8: Представление данных файла в виде графика

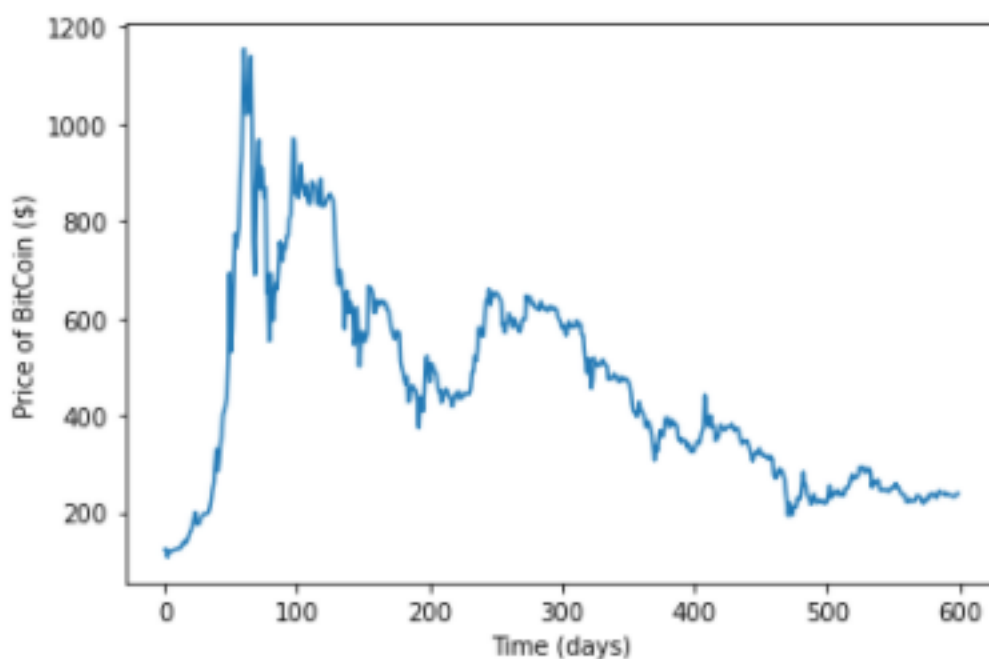


Рис. 7: Данные из файла в виде графика

После этого с помощью функции автокорреляции получим еще один график:

```
1 lags, corrs = autocorr(wave)
2 thinkplot.plot(lags, corrs)
3 decorate(xlabel='Lag', ylabel='Correlation')
```

Листинг 9: Получение нового графика после автокорреляции

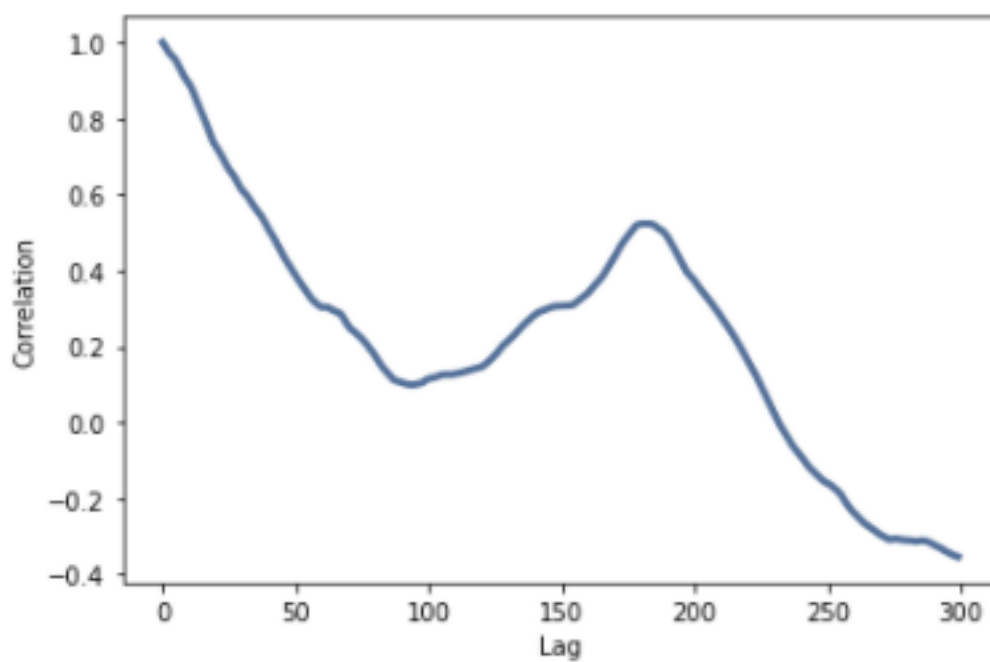


Рис. 8: Полученный график после автокорреляции

В результате можн осказать о том, что полученный график повторяет график цены BitCoin. Периодичность в графике не наблюдается.

4 Часть №4: saxophone.ipynb

В четвертом пункте пятой лабораторной работы нам необходимо прочитать блокнот `saxophone.ipynb`, пройтись по всем примерам, затем выбрать другой сегмент записи и поработать с ним.

Будем использовать звук саксофона, который уже был использован:

```
In [14]: wave = read_wave('100475_iluppai_saxophone-weep.wav')
         wave.normalize()
         wave.make_audio()
```

Out[14]:

▶ 0:00 / 0:11 🔊 ⋮

Рис. 9: Получение звука саксофона

После этого построим спектограмму данного звука:

```
1 gram = wave.make_spectrogram(seg_length=1024)
2 gram.plot(high=3000)
3 decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 10: Построение спектограммы

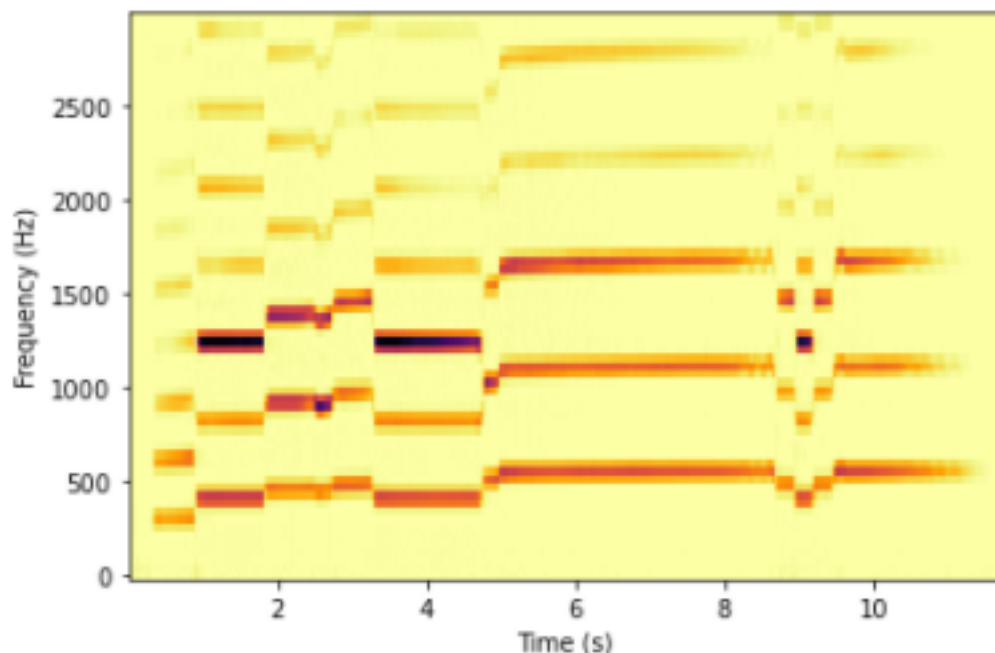


Рис. 10: Полученная спектограмма

После этого выделим сегмент из исходной записи с 6 секунды длительностью 0,5 секунды:

```
B [16]: segment = wave.segment(start=6.0, duration=0.5)
segment.make_audio()
```

Out[16]:

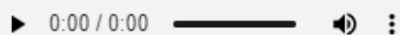


Рис. 11: Получение сегмента из записи

Далее получим спектр из данного сегмента:

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

Листинг 11: Получение спектра сегмента

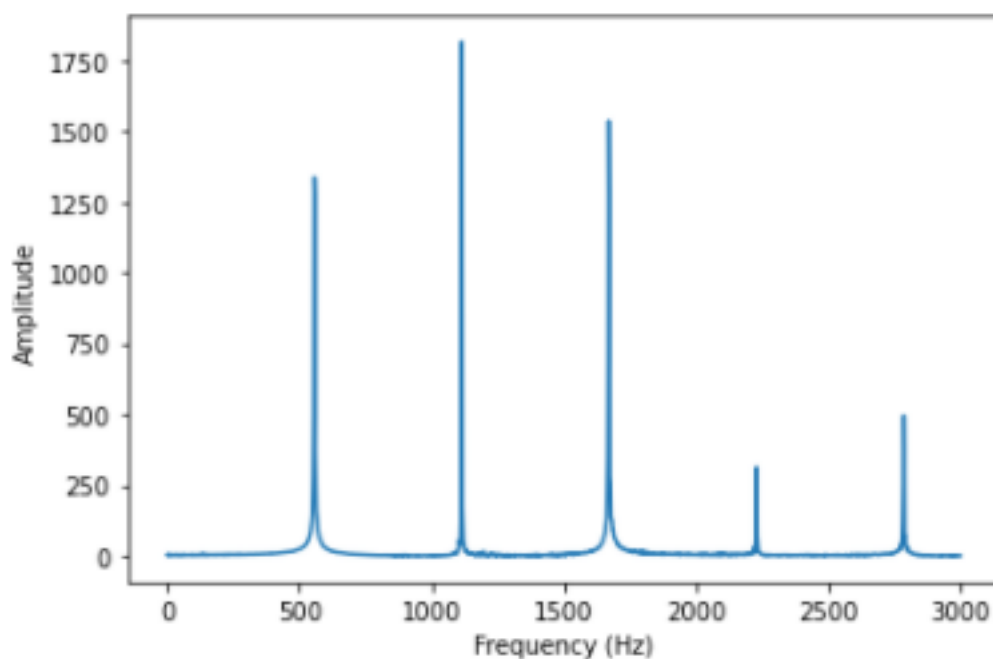


Рис. 12: Полученный спектр сегмента

Сразу же получим все пики полученного спектра:

```
B [18]: spectrum.peaks()[:10]
```

```
Out[18]: [(1815.189472282375, 1114.0),
(1536.1422239239648, 1670.0),
(1337.2214321092208, 556.0),
(1276.178380563102, 1672.0),
(1274.2203137022834, 558.0),
(706.3189084856051, 1668.0),
(498.87612764141124, 554.0),
(496.1139335893734, 2784.0),
(375.3414270174492, 2786.0),
(366.17346455840516, 560.0)]
```

Рис. 13: Пики полученного спектра

Для сравнения получим треугольный сигнал той же частоты:

```
In [19]: TriangleSignal(freq=560).make_wave(duration=0.5).make_audio()  
Out[19]:
```



Рис. 14: Треугольный сигнал

Теперь воспользуемся автокорреляцией для нашего сегмента. Для этого сначала напишем функцию:

```
1 def autocorr(segment):  
2     corrs = np.correlate(segment.ys, segment.ys, mode='same')  
3     N = len(corrs)  
4     lengths = range(N, N//2, -1)  
5  
6     half = corrs[N//2:].copy()  
7     half /= lengths  
8     half /= half[0]  
9     return half
```

Листинг 12: Функция для автокорреляции

Теперь обратимся к этой функции и выведем на экран полученный график:

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

Листинг 13: Получение графика автокорреляции

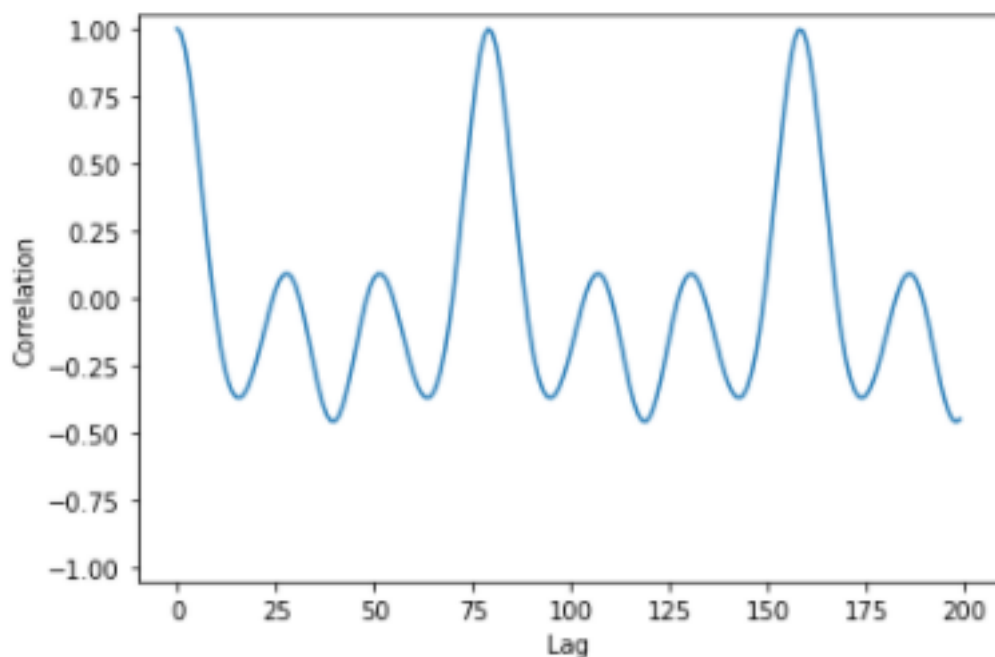


Рис. 15: Полученный график автокорреляции

По графику видно, что первый наибольший пик начинается примерно около $\text{lag} = 65$. Для нахождения частоты в данном lag напомним функцию:

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

Листинг 14: Функция для нахождения частоты

После этого вызовем ее, передав на вход примерные значения начала и конца lag :

```
B [23]: find_frequency(corrs, 65, 95)
79
Out[23]: 558.2278481012657
```

Рис. 16: Использование написанной функции

Как можно увидеть, пик находится на $\text{lag} = 79$

Теперь нам необходимо отфильтровать наш сегмент через фильтр низких частот:

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

Листинг 15: Фильтрация сегмента

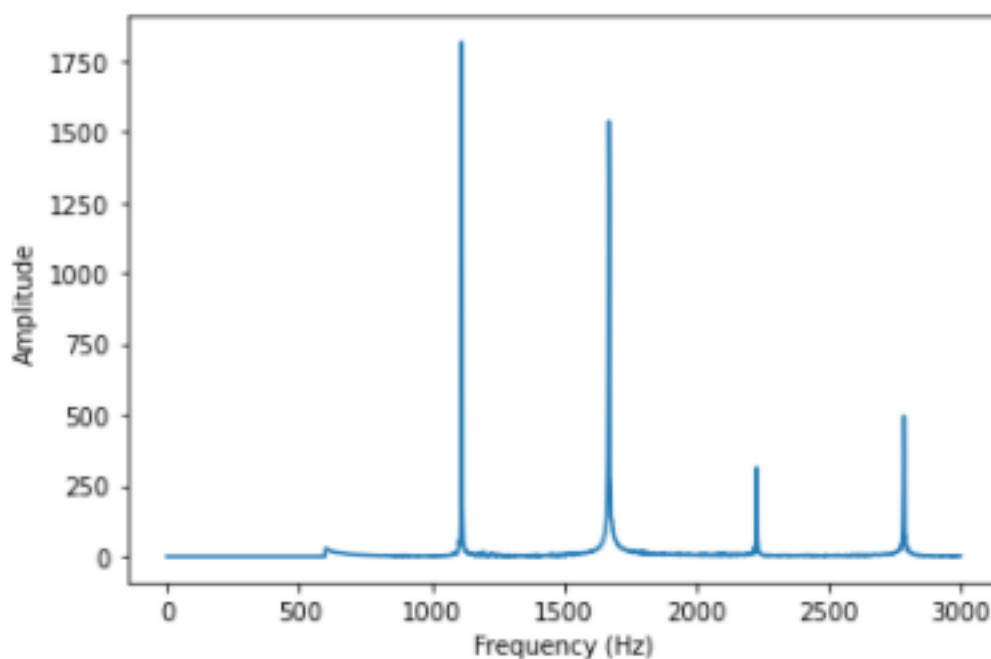


Рис. 17: Полученный график после фильтрации

Как можно увидеть по графику, главная частота была убрана.

Теперь нам осталось преобразовать сегмент в аудио для сравнения с исходным сегментом:

```
In [25]: segment2 = spectrum2.make_wave()  
         segment2.make_audio()
```

Out[25]:

▶ 0:00 / 0:00 🔊 ⋮

Рис. 18: Получение аудио из отфильтрованного сегмента

Наконец, прослушав исходный сегмент и отфильтрованный можно сказать, что эти сигналы очень похожи, но отфильтрованный звучит приглушеннее.

5 Выводы

В результате выполнения данной лабораторной работы мы изучили, что такое автокорреляция. Научились вычислять ее для различных lag . Была создана функция `estimate-fundamental` для отслеживания высоты тона звука. Также вычислили автокорреляцию цен за Bitcoin из прошлой лабораторной работы. Наконец, мы поработали с сегментами записи звуков саксофона и вычислили для них автокорреляцию.