

Лабораторная работа №5  
Автокорреляция

Смирнов Никита

25 апреля 2021 г.

# Оглавление

1	Упражнение 5.1	4
2	Упражнение 5.2	7
3	Упражнение 5.3	10
4	Упражнение 5.4	13
5	Выводы	14

# Список иллюстраций

1.1	Высота тона . . . . .	5
1.2	Высота тона . . . . .	6
2.1	Сегменты звуков . . . . .	8
2.2	Кривая отслеживания высоты тона на спектрограмме . . .	9
3.1	Визуализация данных . . . . .	11
3.2	Автокорреляция при помощи <code>autocorr</code> . . . . .	12

# Листинги

1.1	Вокальный чирп . . . . .	4
1.2	Первый сегмент . . . . .	4
1.3	Оценка высоты тона . . . . .	4
1.4	Нахождение <code>lag</code> . . . . .	5
1.5	Нахождение частоты . . . . .	5
1.6	Второй сегмент . . . . .	5
1.7	Оценка высоты тона . . . . .	5
1.8	Нахождение <code>lag</code> . . . . .	6
1.9	Нахождение частоты . . . . .	6
2.1	Загрузка звука . . . . .	7
2.2	Спектрограмма звука . . . . .	7
2.3	Инкапсуляция функции . . . . .	8
2.4	Пример работы функции . . . . .	8
2.5	Отслеживание высоты тона . . . . .	8
2.6	Кривая отслеживания высоты тона на спектрограмме . . . .	9
3.1	Таблица данных . . . . .	10
3.2	Визуализация данных . . . . .	10
3.3	Автокорреляция при помощи <code>autocorr</code> . . . . .	11
3.4	Вторая половина результата . . . . .	12

# Глава 1

## Упражнение 5.1

Я позаимствовал запись вокального чирпа, который использовался выше.

```
1 wave = thinkdsp.read_wave('28042__bcjordan__voicedownbew.wav')
2 wave.normalize()
3 wave.make_audio()
```

Листинг 1.1: Вокальный чирп

Далее я взял сегмент, который начинается с 0.3 секунды после начала и длительностью 0.01 секунды:

```
1 segment = wave.segment(start=0.3, duration=0.01)
```

Листинг 1.2: Первый сегмент

Применим автокорреляционную функцию, чтобы оценить высоту тона:

```
1 lags, corrs = autocorr(segment)
2 thinkplot.plot(lags, corrs)
3 thinkplot.config(xlabel='Lag (index)', ylabel='Correlation',
    ylim=[-2, 2])
```

Листинг 1.3: Оценка высоты тона

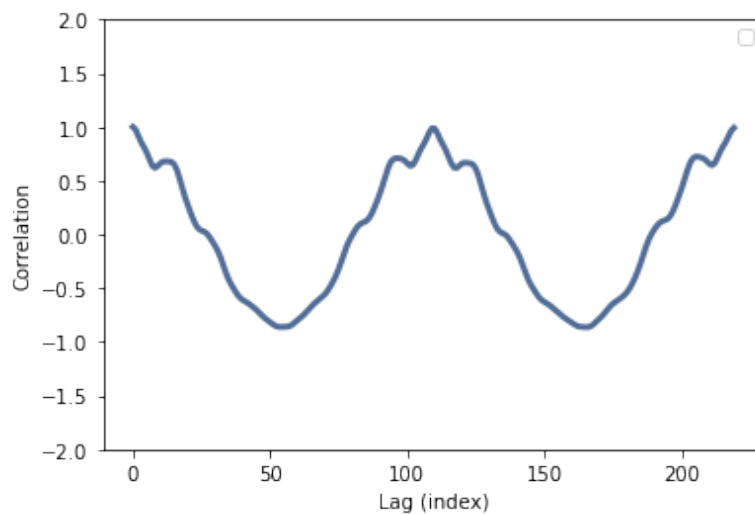


Рис. 1.1: Высота тона

Пик находится между 100 и 150. Используем `argmax`, чтобы уточнить значение `lag` для этого пика:

```

1 low, high = 100, 150
2 lag = np.array(corr[low:high]).argmax() + low
3 lag

```

Листинг 1.4: Нахождение `lag`

Находим соответствующую частоту для `lag = 109`:

```

1 period = lag / segment framerate
2 frequency = 1 / period
3 frequency

```

Листинг 1.5: Нахождение частоты

Частота равняется 404.5871559633028.

Теперь рассмотрим сегмент сигнала через 1 секунду и сделаем аналогичные действия.

```

1 segment = wave.segment(start=1.0, duration=0.01)

```

Листинг 1.6: Второй сегмент

Применим автокорреляционную функцию, чтобы оценить высоту тона:

```

1 lags, corrs = autocorr(segment)
2 thinkplot.plot(lags, corrs)

```

```

3 thinkplot.config(xlabel='Lag (index)', ylabel='Correlation',
  ylim=[-2, 2])

```

Листинг 1.7: Оценка высоты тона

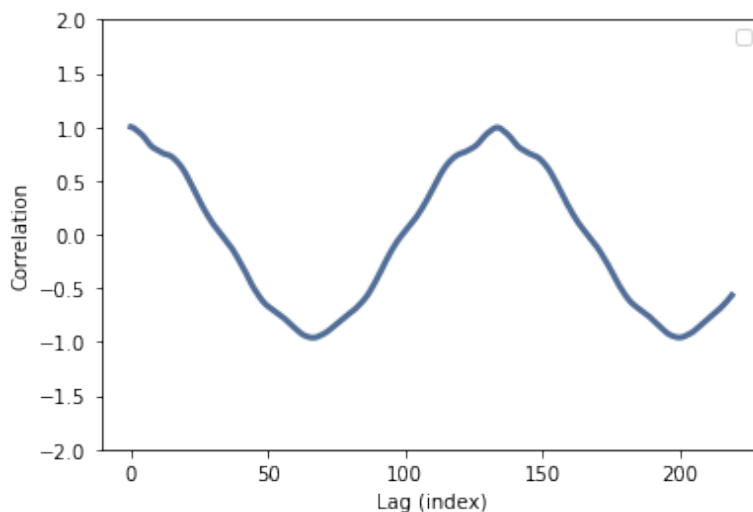


Рис. 1.2: Высота тона

Пик снова находится между 100 и 150. Используем `argmax`, чтобы уточнить значение `lag` для этого пика:

```

1 low, high = 100, 150
2 lag = np.array(corr[low:high]).argmax() + low
3 lag

```

Листинг 1.8: Нахождение `lag`

Находим соответствующую частоту для `lag = 134`:

```

1 period = lag / segment.framerate
2 frequency = 1 / period
3 frequency

```

Листинг 1.9: Нахождение частоты

Частота равняется 329.1044776119403. Отсюда можно сделать вывод, что основная частота ожидаемо уменьшается при увеличении времени начала сегмента.

## Глава 2

### Упражнение 5.2

Загрузим тот же вокальный чирп.

```
1 wave = thinkdsp.read_wave('28042__bcjordan__voicedownbew.wav')
2 wave.normalize()
3 wave.make_audio()
```

Листинг 2.1: Загрузка звука

Воспользуемся примером кода из `chap05.ipynb`. Вот его спектрограмма:

```
1 wave.make_spectrogram(2048).plot(high=4200)
2 thinkplot.config(xlabel='Time (s)',
3                  ylabel='Frequency (Hz)',
4                  xlim=[0, 1.4],
5                  ylim=[0, 4200])
```

Листинг 2.2: Спектрограмма звука



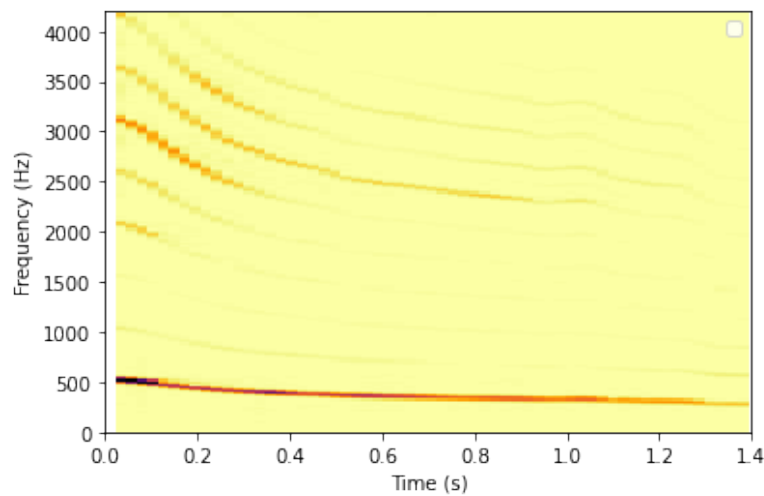


Рис. 2.1: Сегменты звуков

Инкапсулируем предлагаемый код в функцию. Найти первый самый высокий пик в автокорреляционной функции сложно. Поэтому мы просто укажем диапазон `lag` для поиска.

```

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 2.3: Инкапсуляция функции

Рассмотрим пример работы функции:

```

1 duration = 0.01
2 segment = wave.segment(start=0.2, duration=duration)
3 freq = estimate_fundamental(segment)
4 freq

```

Листинг 2.4: Пример работы функции

Результатом работы получилась частота `436.63366336633663`.

Используем написанную функцию для отслеживания высоты тона записанного звука. `ts` - средние точки каждого сегмента.

```

1 step = 0.05
2 starts = np.arange(0.0, 1.4, step)
3

```

```

4 ts = []
5 freqs = []
6
7 for start in starts:
8     ts.append(start + step/2)
9     segment = wave.segment(start=start, duration=duration)
10    freq = estimate_fundamental(segment)
11    freqs.append(freq)

```

Листинг 2.5: Отслеживание высоты тона

Рассмотрим кривую отслеживания высоты тона, наложенную на спектрограмму:

```

1 wave.make_spectrogram(2048).plot(high=900)
2 thinkplot.plot(ts, freqs, color='green')
3 thinkplot.config(xlabel='Time (s)',
4                  ylabel='Frequency (Hz)',
5                  xlim=[0, 1.4],
6                  ylim=[0, 900])

```

Листинг 2.6: Кривая отслеживания высоты тона на спектрограмме

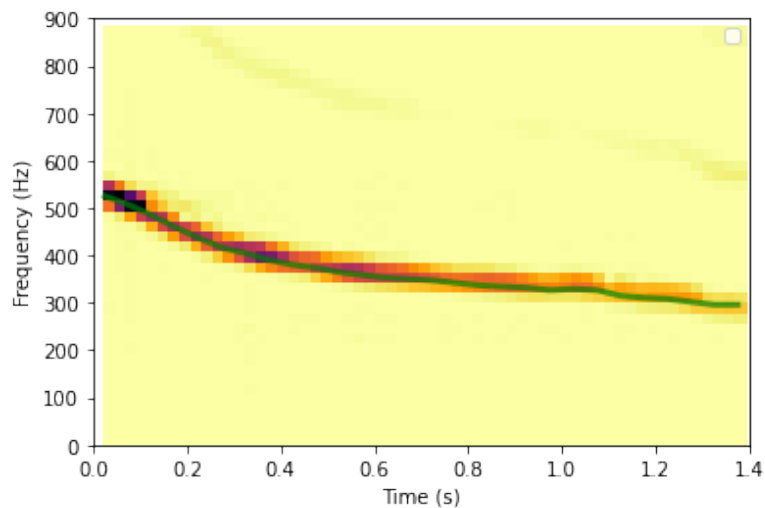


Рис. 2.2: Кривая отслеживания высоты тона на спектрограмме

Наложив оценки высоты тона на спектрограмму записи, можно видеть, что функция полностью справляется со своей задачей.

## Глава 3

### Упражнение 5.3

Воспользуемся данными о ежедневной цене BitCoin в течение года из прошлой лабораторной работы.

```
1 data = pd.read_csv('BTC_USD_2020-04-26_2021-04-25-CoinDesk.csv')
2 data
```

Листинг 3.1: Таблица данных

Визуализируем скачанные данные.

```
1 wave = thinkdsp.Wave(data['Closing Price (USD)'], data.index,
    framerate=1)
2 wave.plot()
3 thinkplot.config(xlabel='Time',
4                  ylabel='Price')
```

Листинг 3.2: Визуализация данных

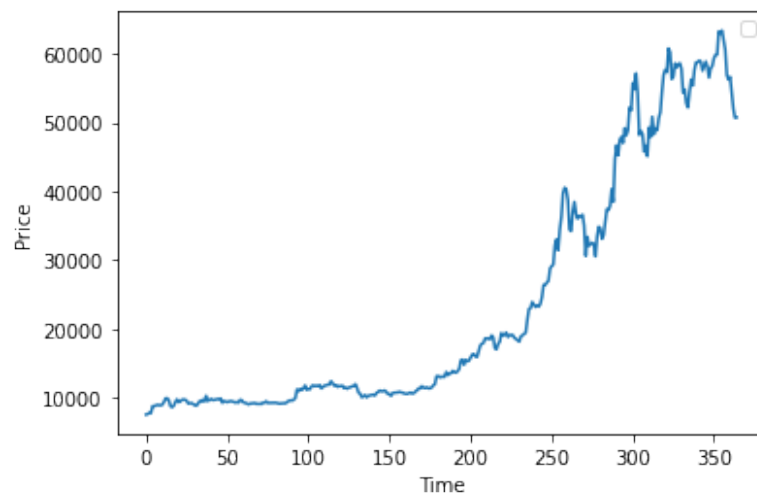


Рис. 3.1: Визуализация данных

Воспользуемся функцией автокорреляции, использующая статистическое определение, то есть она сдвигает среднее значение к нулю, делит на стандартное отклонение и делит сумму на  $N$ .

```
1 from autocorr import autocorr
2
3 lags, corrs = autocorr(wave)
4 thinkplot.plot(lags, corrs)
5 thinkplot.config(xlabel='Lag',
6                  ylabel='Correlation')
```

Листинг 3.3: Автокорреляция при помощи `autocorr`

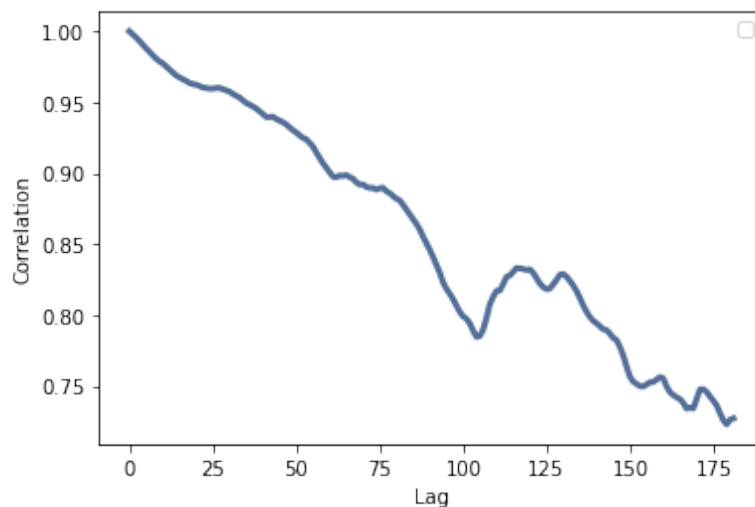


Рис. 3.2: Автокорреляция при помощи `autocorr`

Автокорреляционная функция падает медленно, поскольку задержка увеличивается, предполагая какой-то розовый шум.

Результат симметричен, потому что два сигнала идентичны, и отрицательный `lag` у одного даёт такой же эффект, как и положительный `lag` у другого. Вторая половина результата соответствует положительным `lag`:

```

1 N = len(corrs2)
2 half = corrs2[N//2:]
3 thinkplot.plot(half)
4 thinkplot.config(xlabel='Lag',
5                  ylabel='Dot product')
```

Листинг 3.4: Вторая половина результата

Для этого набора данных, вероятно, более уместно статистическое определение автокорреляционной функции.

## Глава 4

### Упражнение 5.4

В данном упражнении я просмотрел прикрепленный ролик на YouTube и изучил все примеры, которые были приведены в файле `saxophone.ipynb` для разных форматов записи.

## Глава 5

### Выводы

Во время выполнения лабораторной работы получены навыки работы с корреляцией, последовательной корреляцией, автокорреляцией. Также рассмотрена автокорреляционная функция (АКФ).