Лабораторная работа №3
Апериодические сигналы

Кобыжев Александр

14 марта 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Упражнение 3.1

1.1 Пример утечки

В данном упражнении нас просят открыть `chap03.ipynb`, запустить и прослушать примеры из блокнота. Также в задании прописано, что нужно заменить в примере с утечкой окно Хэмминга одним из окон, которое предоставляется `NumPy`. Для этого сначала рассмотрим пример утечки.

```
signal = thinkdsp.SinSignal(freq=440)
duration = signal.period * 30.25
wave = signal.make_wave(duration)
spectrum = wave.make_spectrum()

spectrum.plot(high=880)
thinkplot.config(xlabel='Frequency (Hz)')
```

Листинг 1.1: Пример утечки

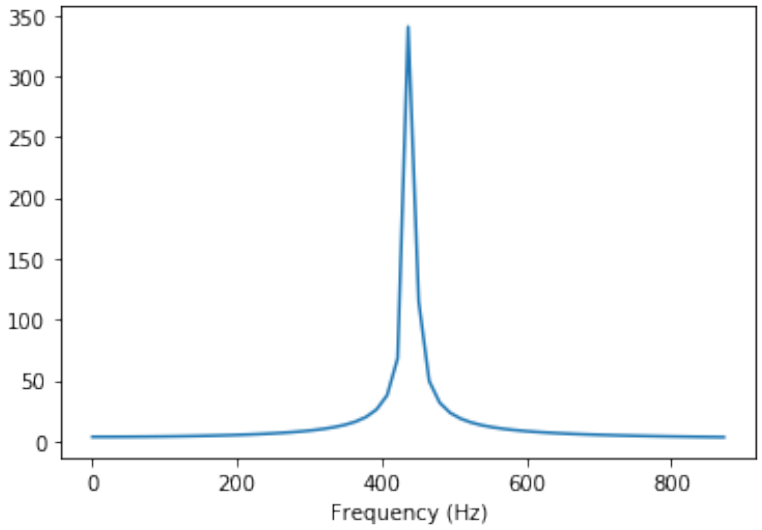

Рис. 1.1: Спектр утечки

1.2 Замена окна Хэмминга

Заменим окно Хэмминга на другие 4 окна.

```
for window_func in [np.bartlett, np.blackman, np.kaiser,
    np.hanning]:
    wave = signal.make_wave(duration)
    if window_func.__name__ == "kaiser":
        wave.ys *= window_func(len(wave.ys), 10)
    else:
        wave.ys *= window_func(len(wave.ys))
    spectrum = wave.make_spectrum()
    spectrum.plot(high=880, label=window_func.__name__)

thinkplot.config(xlabel='Frequency (Hz)', legend=True)
```

Листинг 1.2: Создание других окон

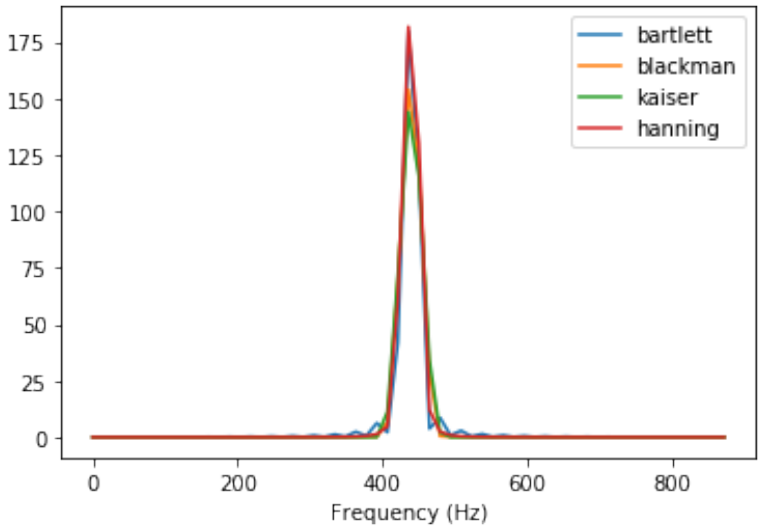

Рис. 1.2: Спектр созданных окон

Все четыре окна хорошо справляются с задачей уменьшения утечки.

Глава 2

Упражнение 3.2

2.1 Написание класса SawtoothChirp

Напишем класс `SawtoothChirp`, который переопределяет `evaluate` для генерации пилообразного сигнала.

```
import math
PI2 = 2 * math.pi

class SawtoothChirp(thinkdsp.Chirp):
    def evaluate(self, ts):
        freqs = np.linspace(self.start, self.end, len(ts) - 1)
        dts = np.diff(ts)
        dphis = PI2 * freqs * dts
        phases = np.cumsum(dphis)
        phases = np.insert(phases, 0, 0)
        cycles = phases / PI2
        frac, _ = np.modf(cycles)
        ys = self.amp * frac
        return ys
```

Листинг 2.1: Класс SawtoothChirp

2.2 Проверка работоспособности

После написания класса удостоверимся, что переопределённый метод действительно генерирует пилообразный сигнал с линейно увеличивающейся или уменьшающейся частотой путём прослушивания получившегося звука и рассмотрения спектрограммы.

```
signal = SawtoothChirp(start=300, end=900)
wave = signal.make_wave(duration=1, framerate=10000)
wave.apodize()
wave.make_audio()
```

Листинг 2.2: Создание и воспроизведение звука

На полученной спектрограмме эффект биений очевиден, а прослушав полученный до этого звук внимательно, можно даже их и услышать.

```
sp = wave.make_spectrogram(1024)
sp.plot()
thinkplot.config(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 2.3: Спектрограмма звука

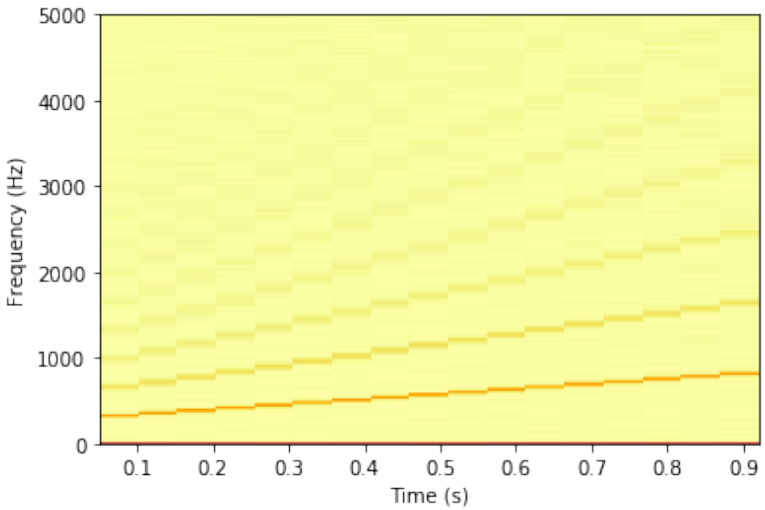

Рис. 2.1: Спектр сегмента звука

Глава 3

Упражнение 3.3

Поскольку основная частота колеблется от 2500 до 3000 Гц, я ожидаю увидеть что-то вроде высокой башни в этом диапазоне. Первая гармоника колеблется от 5000 до 6000 Гц, поэтому я ожидаю более короткую башню в данном диапазоне. Вторая гармоника колеблется от 7500 до 9000 Гц, поэтому я ожидаю что-то еще более короткое в этом диапазоне.

Другие гармоники повсюду накладываются друг на друга, поэтому я ожидаю увидеть некоторую энергию на всех других частотах. Эта распределённая энергия создает интересные звуки.

```
signal = SawtoothChirp(start=2500, end=3000)
wave = signal.make_wave(duration=1, framerate=20000)
wave.make_audio()
```

Листинг 3.1: Создание сигнала

Теперь посмотрим на получившийся спектр.

```
wave.make_spectrum().plot()
```

Листинг 3.2: Визуализация спектра

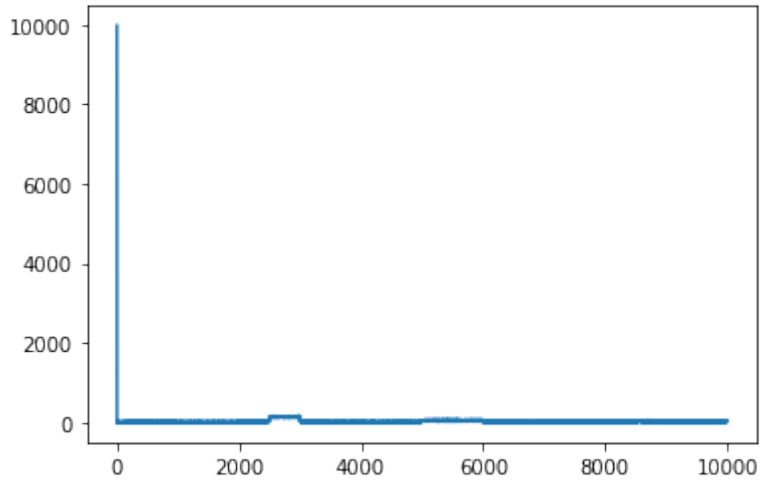

Рис. 3.1: Спектр созданного звука

Получился похожий график спектра, как я и предполагал, но нам мешает большая частота в самом начале, поэтому уберём её для более детального просмотра спектра звука.

```
cut_wave = wave.make_spectrum()
cut_wave.high_pass(10)
cut_wave.plot()
```

Листинг 3.3: Удаление частоты в начале

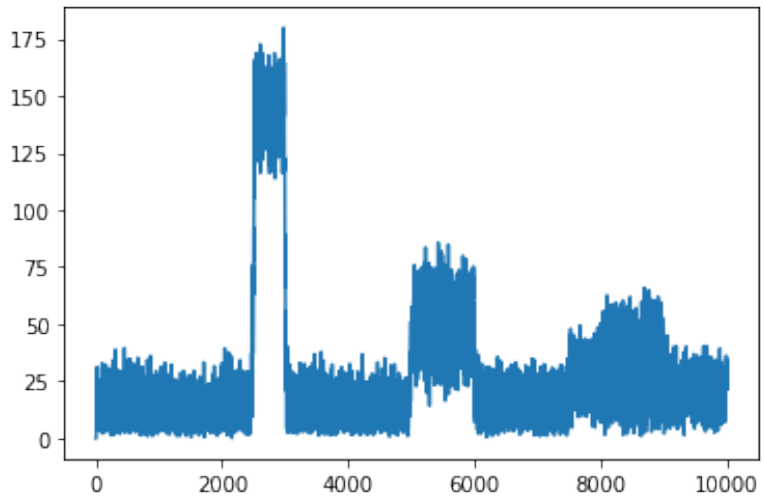

Рис. 3.2: Улучшенный спектр созданного звука

Глава 4

Упражнение 3.4

Для данного задания я нашёл в интернете глиссандо на скрипке.

```
wave =
    thinkdsp.read_wave('411728__inspectorj__violin-glissando-ascending-a-h1.wav')
wave.make_audio()
```

Листинг 4.1: Загрузка и прослушивание звука

Теперь рассмотрим спектрограмму звука.

```
wave.make_spectrogram(512).plot(high=5000)
```

Листинг 4.2: Спектрограмма звука

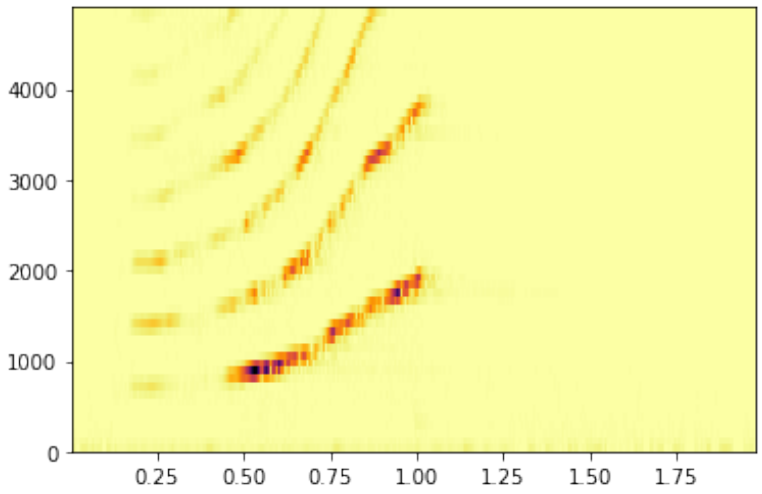

Рис. 4.1: Спектрограмма звука

Теперь мы можем наблюдать глиссандо.

Глава 5

Упражнение 3.5

5.1 Создание класса

Написанный класс представляет собой тромбоноподобный сигнал с переменной частотой.

```
class TromboneGliss(thinkdsp.Chirp):
    def _evaluate(self, ts):
        l1, l2 = 1.0 / self.start, 1.0 / self.end
        lengths = np.linspace(l1, l2, len(ts)-1)
        freqs = 1 / lengths
        return self._evaluate(ts, freqs)
```

Листинг 5.1: Создание класса

5.2 Создание звука и его спектрограмма

Создадим первую часть звука от C3 до F3, где C3 - 262 Гц, а F3 - 349 Гц.

```
low = 262
high = 349
signal = TromboneGliss(low, high)
wave1 = signal.make_wave(duration=1)
wave1.apodize()
wave1.make_audio()
```

Листинг 5.2: Создание первой части звука

Теперь создадим вторую часть звука от F3 до C3.

```
signal = TromboneGliss(high, low)
wave2 = signal.make_wave(duration=1)
wave2.apodize()
wave2.make_audio()
```

Листинг 5.3: Создание второй части звука

Затем соединим обе части в один полноценный звук.

```
wave = wave1 | wave2
wave.make_audio()
```

Листинг 5.4: Соединение двух частей звука

Теперь сделаем спектрограмму и расммотрим её.

```
sp = wave.make_spectrogram(1024)
sp.plot(high=1000)
```

Листинг 5.5: Спектрограмма звука

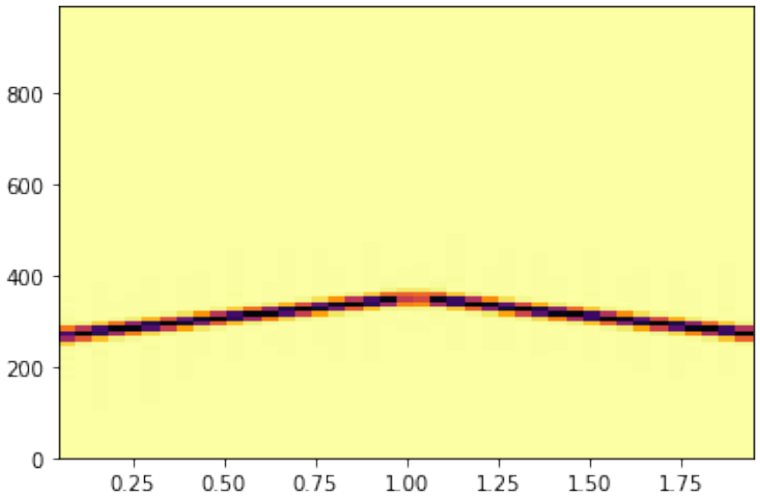

Рис. 5.1: Спектрограмма глиссандо на трамбоне

Глиссандо на тромбоне похоже на линейный чирп.

Глава 6

Упражнение 3.6

В интернете я нашёл гласные звуки, записанные женщиной, голос которой похож на голос бабы-яги из советских фильмов.

```
wave =
    thinkdsp.read_wave('337631__anaphy__female-creak-vowels-a-e-i-o-u.wav')
wave.make_audio()
```

Листинг 6.1: Загрузка и прослушивание звука

Теперь сделаем спектрограмму.

```
wave.make_spectrogram(1024).plot(high=1000)
```

Листинг 6.2: Создание спектрограммы

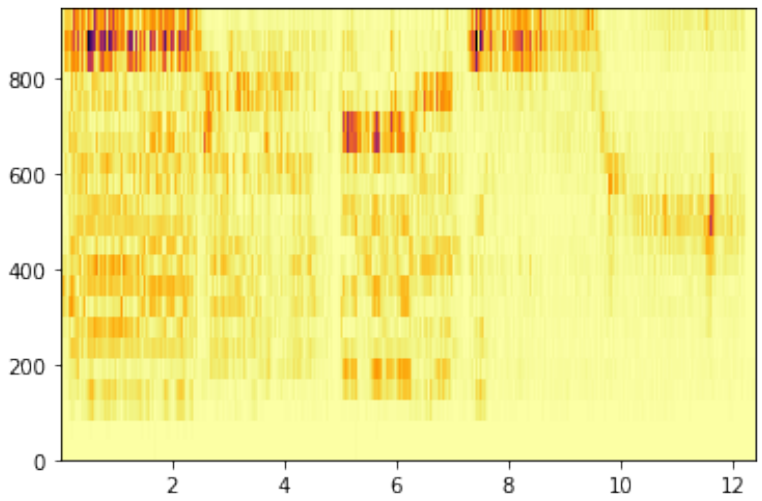

Рис. 6.1: Спектрограмма звука

Пики на спектрограмме называются формантами. В общем, гласные звуки различаются соотношением амплитуд первых двух формант относительно основного тона.

Мы можем увидеть форманты более чётко, выбрав сегмент во время «a».

```
high = 1000
thinkplot.preplot(5)

segment = wave.segment(start=0, duration=2)
segment.make_spectrum().plot(high=high)
```

Листинг 6.3: Спектр звука «a»

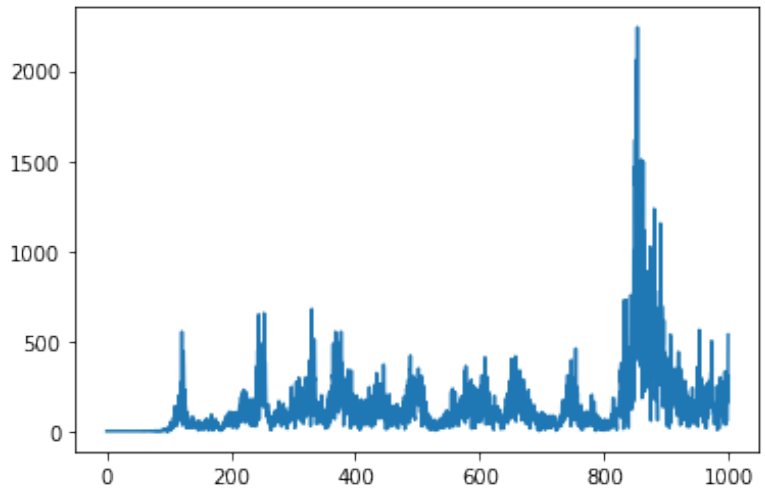

Рис. 6.2: Спектр звука «a»

Основная частота около 100 Гц. Следующие самые высокие пики находятся на 300 Гц и 900 Гц.

Выберем сегмент «e».

```
segment = wave.segment(start=2.5, duration=2)
segment.make_spectrum().plot(high=high)
```

Листинг 6.4: Спектр звука «e»

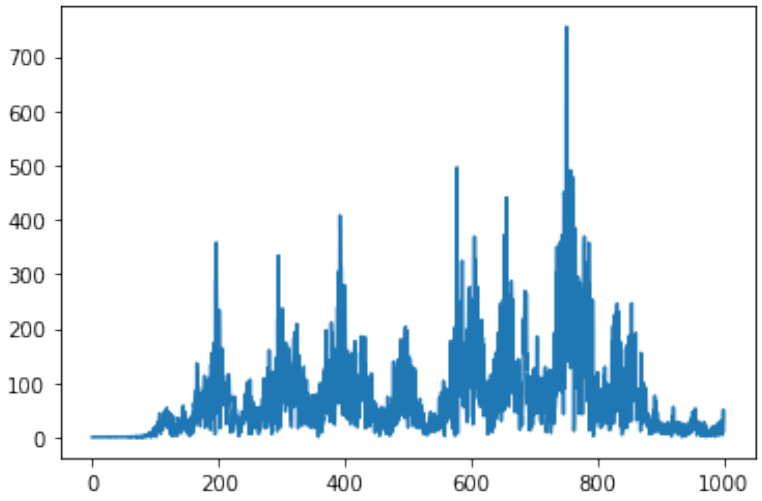

Рис. 6.3: Спектр звука «e»

Основная частота составляет 200 Гц. Следующие самые высокие пики находятся на 400 Гц и 600 Гц.

Выберем сегмент «i».

```
segment = wave.segment(start=4.5, duration=2.5)
segment.make_spectrum().plot(high=high)
```

Листинг 6.5: Спектр звука «i»

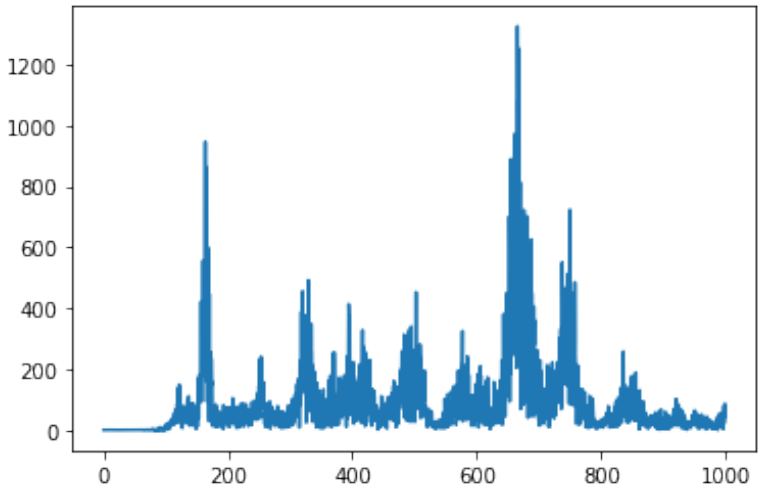

Рис. 6.4: Спектр звука «i»

Основная частота составляет 150 Гц. Следующие самые высокие пики находятся на 300 Гц и 750 Гц.

Это сегмент «o»:

```
segment = wave.segment(start=7.5, duration=2)
segment.make_spectrum().plot(high=high)
```

Листинг 6.6: Спектр звука «o»

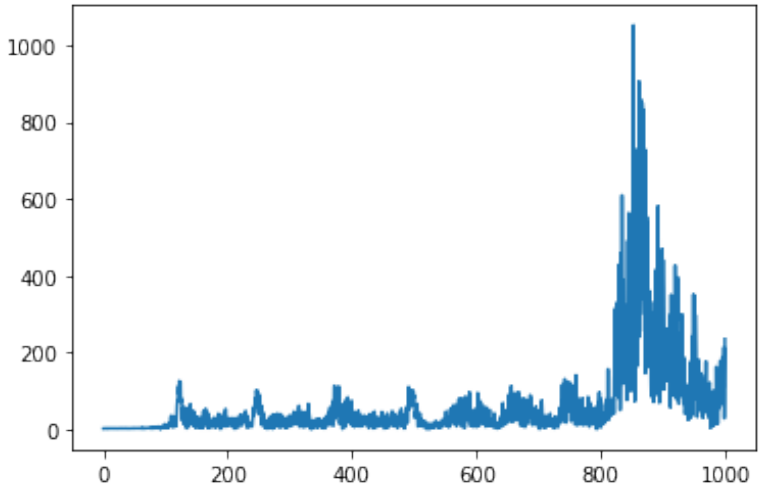

Рис. 6.5: Спектр звука «o»

Основная частота составляет 100 Гц. Следующие самые высокие пики находятся на частоте 900 Гц.

Это сегмент «u»:

```
segment = wave.segment(start=10.2, duration=2)
segment.make_spectrum().plot(high=high)
```

Листинг 6.7: Спектр звука «u»

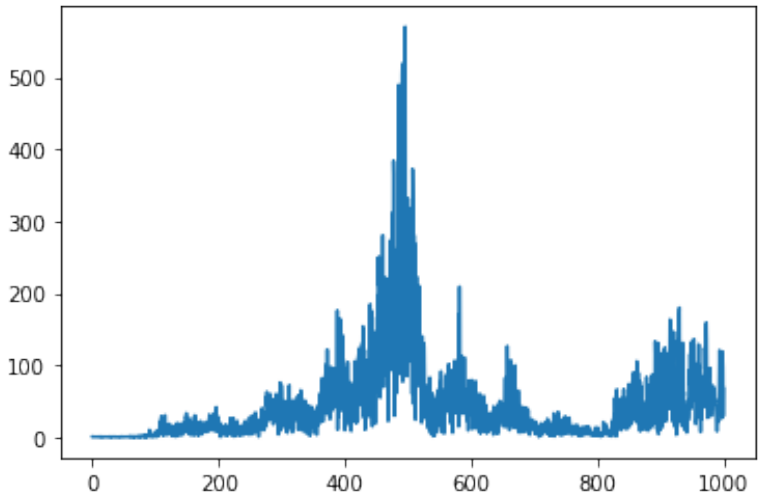

Рис. 6.6: Спектр звука «u»

Основной и самый высокий пик приходится на 500 Гц.

Глава 7

Выводы

Во время выполнения лабораторной работы получены навыки работы с апериодическими сигналами, частотные компоненты которых изменяются во времени, то есть практически все звуковые сигналы. Также рассмотрены спектрограммы - распространённый способ визуализации апериодических сигналов.