

Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа №3

Апериодические сигналы

Выполнил студент 3-го курса
группа 3530901/80201
Матвеец Андрей Вадимович

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург

2021

Содержание

1	Часть №1: Запуск примеров из chap03.ipynb	7
2	Часть №2: Создание SawtoothChirp	12
3	Часть №3: Создание меняющегося пилообразного сигнала	15
4	Часть №4: Глиссандро	17
5	Часть №5: Создание TromboneFliss	19
6	Часть №6: Анализ букв	22
7	Выводы	28

Список иллюстраций

1	Проверка работоспособности	7
2	Замена на окно Бартлетта	8
3	Замена на окно blackman	9
4	Замена на окно hanning	9
5	Замена на окно kaiser	10
6	Совмещение всех графиков	11
7	Начальный сегмент	13
8	Сегмент в аудио	13
9	Конечный сегмент	14
10	Спектограмма сигнала	14
11	Сегмент сигнала	15
12	Сигнал в аудио	15
13	Спектр сигнала	16
14	Полученный сегмент	17
15	Файл в аудио	17
16	Полученный спектр	18
17	Полученная спектограмма	18
18	Сигнал up в аудио	19
19	Полученный сегмент up	19
20	Сигнал down в аудио	20
21	Полученный сегмент down	20
22	Объединение up и down в аудио	20

23	Полученный спектр up и down	21
24	Полученная спектограмма up и down	21
25	Полученный сегмент	22
26	Представление сигнала в аудио	22
27	Полученная спектограмма для сегмента	23
28	Сегмент с буквой A	23
29	Представление сегмента с буквой A в аудио	24
30	Сегмент с буквой B	24
31	Представление сегмента с буквой B в аудио	24
32	Сегмент с буквой C	25
33	Представление сегмента с буквой C в аудио	25
34	Сегмент с буквой D	26
35	Представление сегмента с буквой D в аудио	26
36	Сегмент с буквой E	27
37	Представление сегмента с буквой E в аудио	27

Листинги

1	Окно Бартлетта	7
2	Окно <code>blackman</code>	8
3	Окно <code>hanning</code>	9
4	Окно <code>kaiser</code>	9
5	Совмещение всех графиков	10
6	Импорт библиотек	12
7	Класс <code>SawtoothChirp</code>	12
8	Получение начального сегмента	12
9	Получение конечного сегмента	13
10	Спектограмма сигнала	14
11	Создание сигнала согласно заданию	15
12	Сегмент сигнала	15
13	Спектр сигнала	16
14	Получение сегмента из аудиофайла	17
15	Получение спектра из сегмента	17
16	Получение спектограммы	18
17	Класс <code>TromboneFliss</code>	19
18	Сегмент сигнала <code>up</code>	19
19	Сегмент сигнала <code>down</code>	20
20	Спектр сигнала <code>up</code> и <code>down</code>	21
21	Спектограмма сигнала <code>up</code> и <code>down</code>	21
22	Вывод сегмента	22

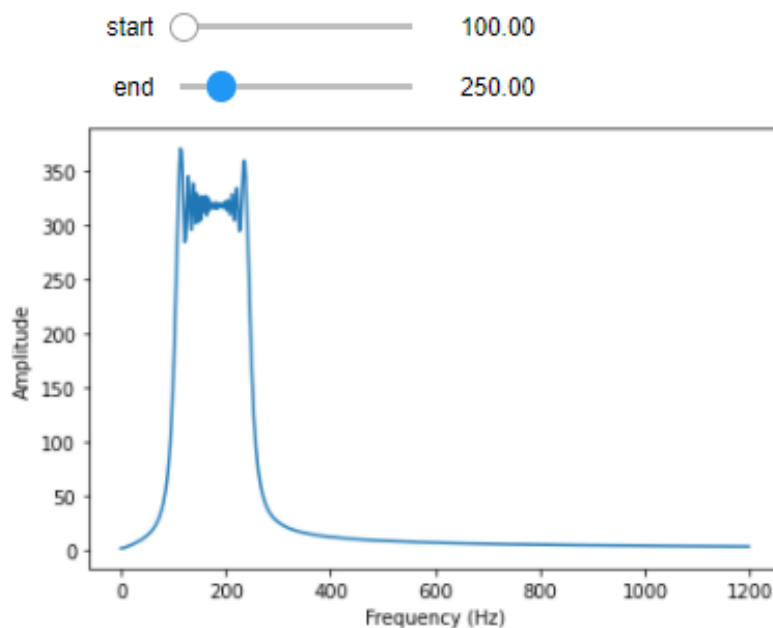
23	Спектограмма для нашего сегмента	22
24	Сегмент с буквой А	23
25	Сегмент с буквой В	24
26	Сегмент с буквой С	25
27	Сегмент с буквой D	25
28	Сегмент с буквой Е	26

1 Часть №1: Запуск примеров из chap03.ipynb

В первом пункте лабораторной работы нам необходимо запустить все файлы из chap03.ipynb, а также в примере с утечкой заменить окно Хэмминга одним из окон, представляемых NumPy. Начнем с запуска всех программ.

```
[19] def eye_of_sauron(start, end):  
    """Plots the spectrum of a chirp.  
  
    start: initial frequency  
    end: final frequency  
    """  
  
    signal = Chirp(start=start, end=end)  
    wave = signal.make_wave(duration=0.5)  
    spectrum = wave.make_spectrum()  
  
    spectrum.plot(high=1200)  
    decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

```
▶ slider1 = widgets.FloatSlider(min=100, max=1000, value=100, step=50)  
slider2 = widgets.FloatSlider(min=100, max=1000, value=200, step=50)  
interact(eye_of_sauron, start=slider1, end=slider2);
```



```
[20]
```

Рис. 1: Проверка работоспособности

Теперь перейдем к написанию разных окон в примерах с утечкой. Начем с окна Бартлетта:

```
1 wave = signal.make_wave(duration)
```

```

2 wave.window(np.bartlett(len(wave)))
3 spectrum = wave.make_spectrum()
4 spectrum.plot(high=880)
5 decorate(xlabel='Frequency (Hz)')

```

Листинг 1: Окно Бартлетта

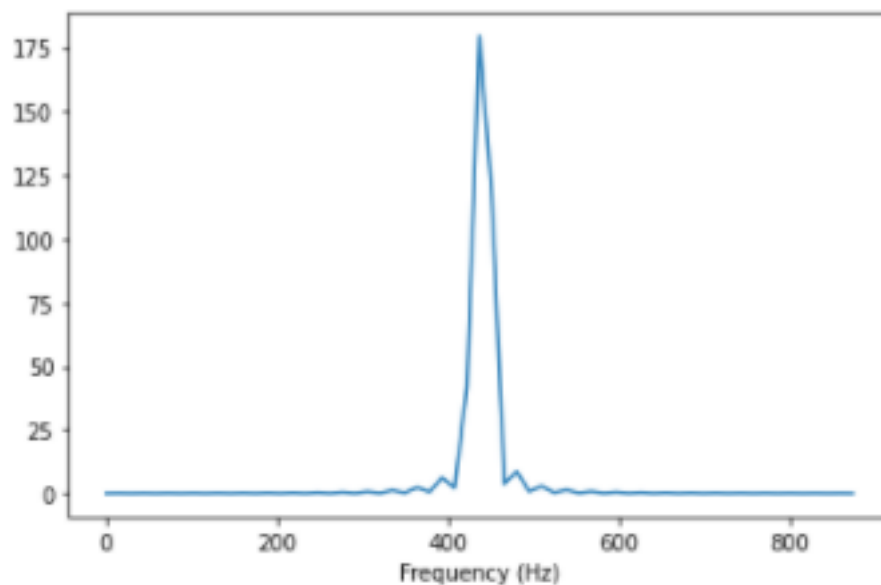


Рис. 2: Замена на окно Бартлетта

После этого напишем окно **blackman**:

```

1 wave = signal.make_wave(duration)
2 wave.window(np.blackman(len(wave)))
3 spectrum = wave.make_spectrum()
4 spectrum.plot(high=880)
5 decorate(xlabel='Frequency (Hz)')

```

Листинг 2: Окно blackman

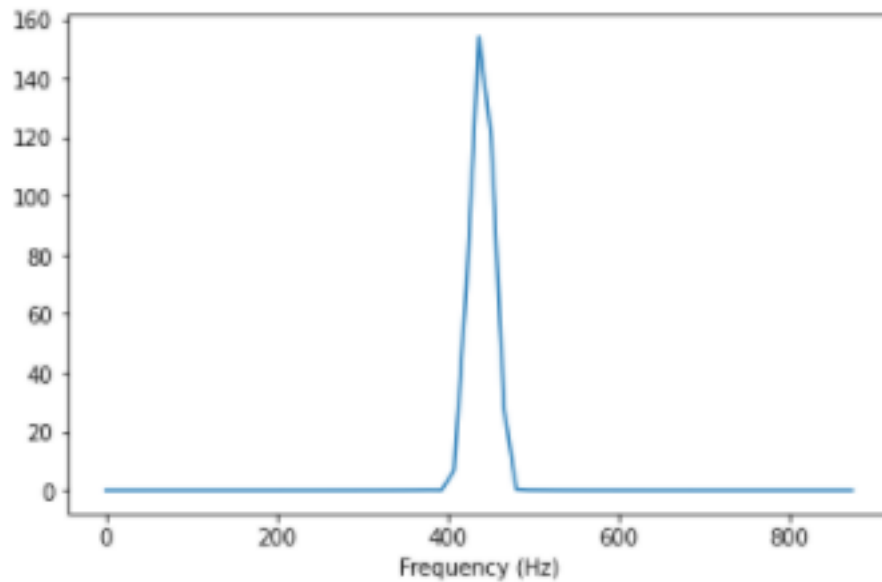


Рис. 3: Замена на окно blackman

Теперь напишем окно `hanning`:

```

1 wave = signal.make_wave(duration)
2 wave.window(np.hanning(len(wave)))
3 spectrum = wave.make_spectrum()
4 spectrum.plot(high = 880)
5 decorate(xlabel = 'Frequency (Hz)')
```

Листинг 3: Окно `hanning`

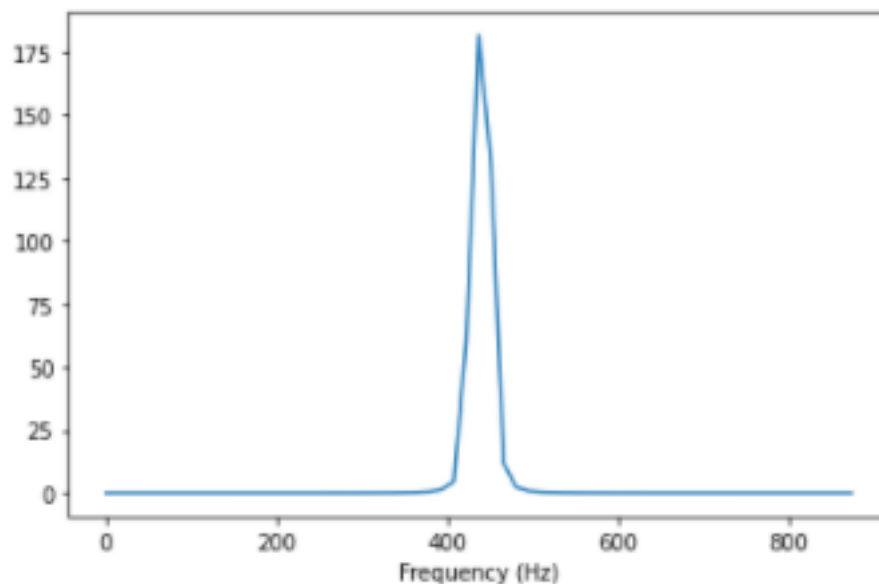


Рис. 4: Замена на окно `hanning`

И, наконец, создадим окно `kaiser`

```

1 wave = signal.make_wave(duration)
```

```

2 wave.window(np.hanning(len(wave)))
3 spectrum = wave.make_spectrum()
4 spectrum.plot(high = 880)
5 decorate(xlabel = 'Frequency (Hz)')

```

Листинг 4: Окно kaiser

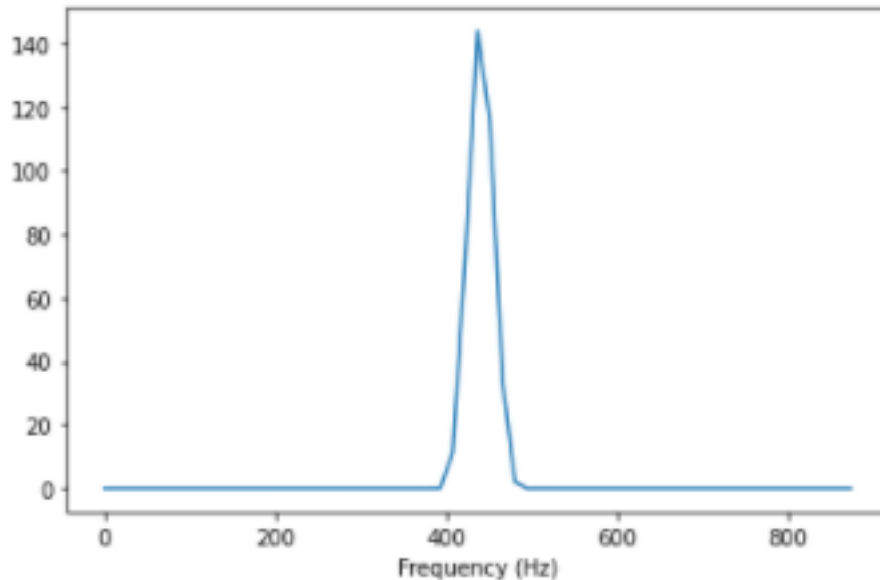


Рис. 5: Замена на окно kaiser

Для большей наглядности совместим все графики в одном:

```

1 wave = signal.make_wave(duration)
2 wave.window(np.bartlett(len(wave)))
3 spectrum = wave.make_spectrum()
4 spectrum.plot(high=880)
5 decorate(xlabel='Frequency (Hz)')
6
7 wave = signal.make_wave(duration)
8 wave.window(np.kaiser(len(wave), 10))
9 spectrum = wave.make_spectrum()
10 spectrum.plot(high = 880, color = 'green')
11 decorate(xlabel = 'Frequency (Hz)')
12
13 wave = signal.make_wave(duration)
14 wave.window(np.hanning(len(wave)))
15 spectrum = wave.make_spectrum()
16 spectrum.plot(high = 880, color = 'orange')
17 decorate(xlabel = 'Frequency (Hz)')
18
19 wave = signal.make_wave(duration)
20 wave.window(np.blackman(len(wave)))
21 spectrum = wave.make_spectrum()
22 spectrum.plot( high=880, color = 'black')
23 decorate(xlabel = 'Frequency (Hz)')

```

Листинг 5: Совмещение всех графиков

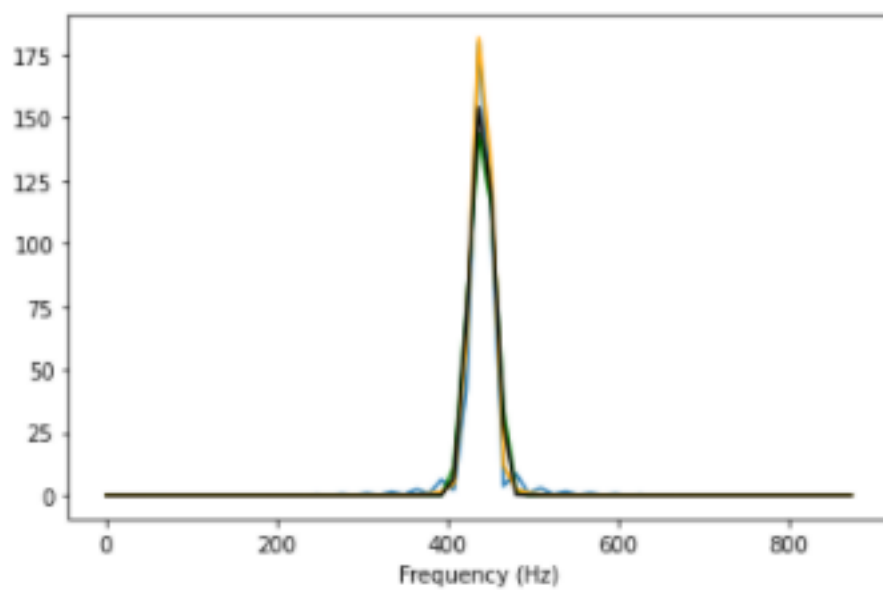


Рис. 6: Совмещение всех графиков

2 Часть №2: Создание SawtoothChirp

Во втором пункте лабораторной работы нам необходимо создать класс `SawtoothChirp`, который бы расширял `Chirp` и переопределял `evaluate` для генерации пилообразного сигнала с линейно увеличивающейся частотой.

Для начала импортируем все необходимые нам для выполнения библиотеки:

```
1 from thinkdsp import Signal, Sinusoid, SquareSignal, TriangleSignal,
  SawtoothSignal, ParabolicSignal
2 from thinkdsp import normalize, unbias, PI2, decorate
3 from thinkdsp import Chirp
4 import numpy as np
```

Листинг 6: Импорт библиотек

После чего создадим сам класс:

```
1 class MySawtoothChirp(Chirp):
2     def evaluate(self, ts):
3         freqs = np.linspace(self.start, self.end, len(ts) - 1)
4
5         dts = np.diff(ts)
6         dphis = PI2 * freqs * dts
7
8         phases = np.cumsum(dphis)
9
10
11         cycles = phases / PI2
12         frac, _ = np.modf(cycles)
13
14         ys = normalize(unbias(frac), self.amp)
15         return ys
```

Листинг 7: Класс `SawtoothChirp`

Теперь выведем начальный сегмент полученного сигнала:

```
1 signal = MySawtoothChirp(start=220, end=440)
2 wave = signal.make_wave(duration=1, framerate=4025)
3 wave.segment(start=0, duration=0.02).plot()
4 decorate(xlabel='Time (s)')
```

Листинг 8: Получение начального сегмента

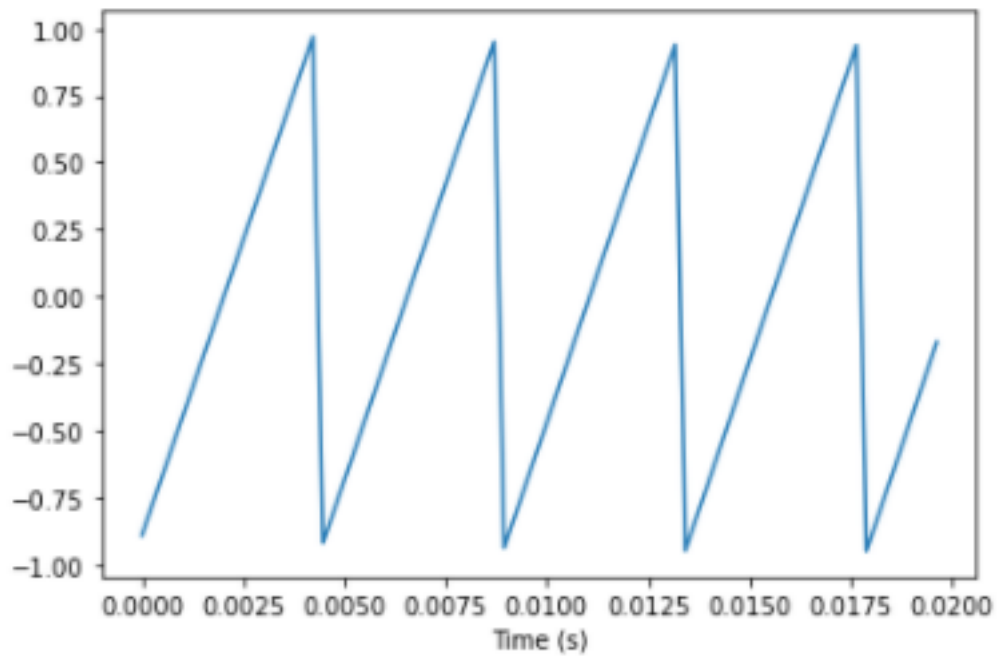


Рис. 7: Начальный сегмент

После чего переведём изначальный сигнал в аудио:

```
In [3]: wave.apodize()
        wave.make_audio()
```

Out[3]:



Рис. 8: Сегмент в аудио

Затем выведем конечный участок сегмента:

```
1 wave.segment(start=1-0.02, duration=0.02).plot()
2 decorate(xlabel='Time (s)')
```

Листинг 9: Получение конечного сегмента

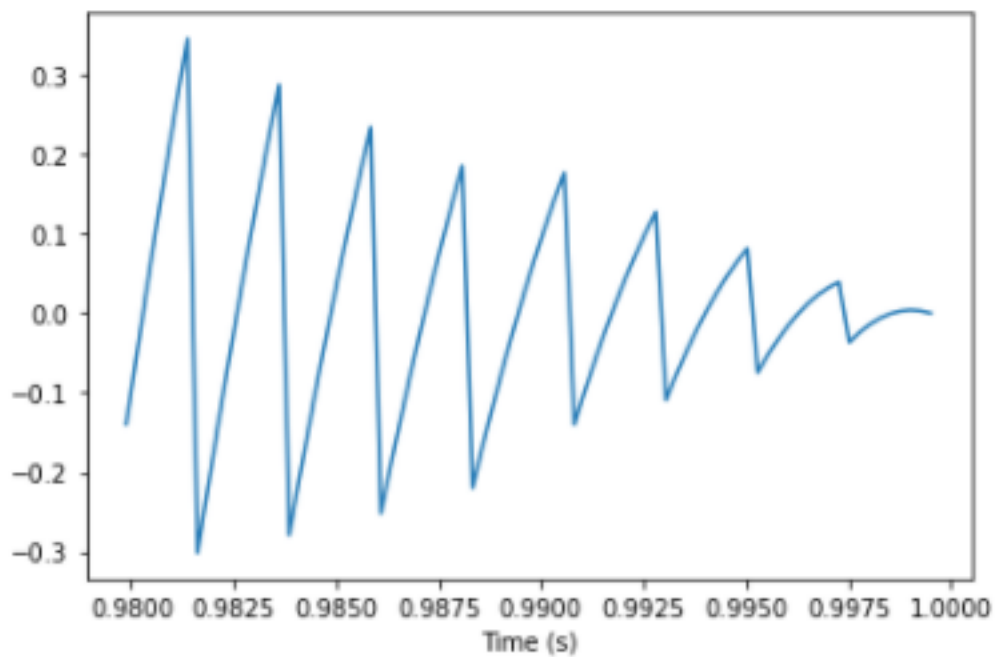


Рис. 9: Конечный сегмент

Наконец, выведем на экран спектрограмму для нашего сигнала:

```

1 sp = wave.make_spectrogram(256)
2 sp.plot()
3 decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 10: Спектограмма сигнала

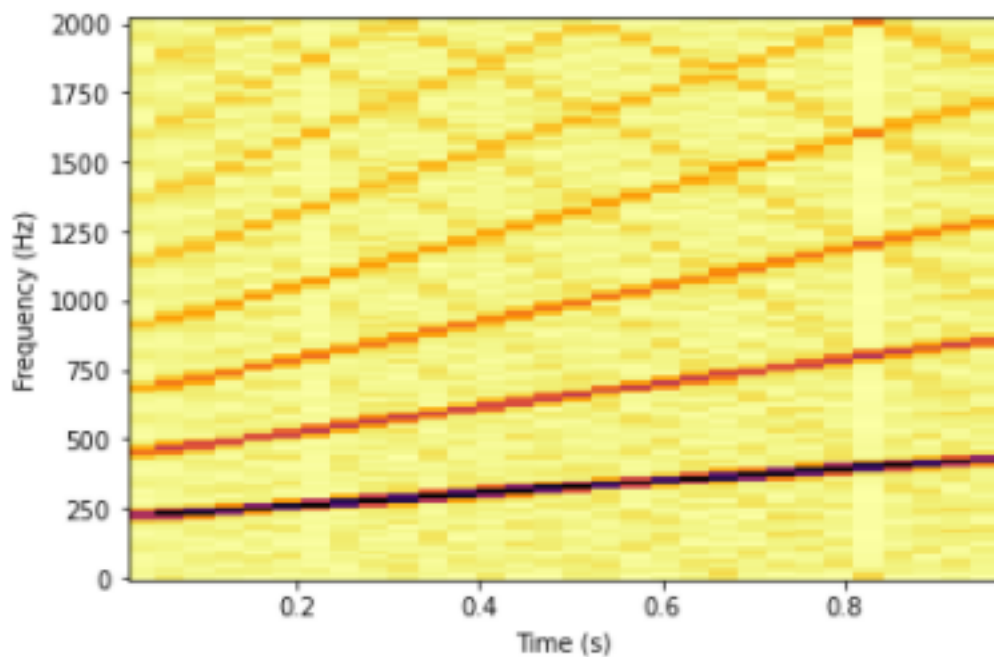


Рис. 10: Спектограмма сигнала

3 Часть №3: Создание меняющегося пилообразного сигнала

В третьей части лабораторной работы там необходимо создать пилообразный чирп, меняющийся от 2500 до 3000 Гц и на его основе сгенерировать сигнал длительностью 1с и частотой кадров в 20кГц. Нарисовать Spectrum.

Для начала создадим сигнал согласно заданию:

```
1 signal = MySawtoothChirp(start=2500, end=3000)
2 wave = signal.make_wave(duration=1, framerate=20_000)
```

Листинг 11: Создание сигнала согласно заданию

После этого посмотрим на сегмент этого сигнала:

```
1 wave.segment(start=0.9, duration=0.02).plot()
2 decorate(xlabel='Time')
```

Листинг 12: Сегмент сигнала

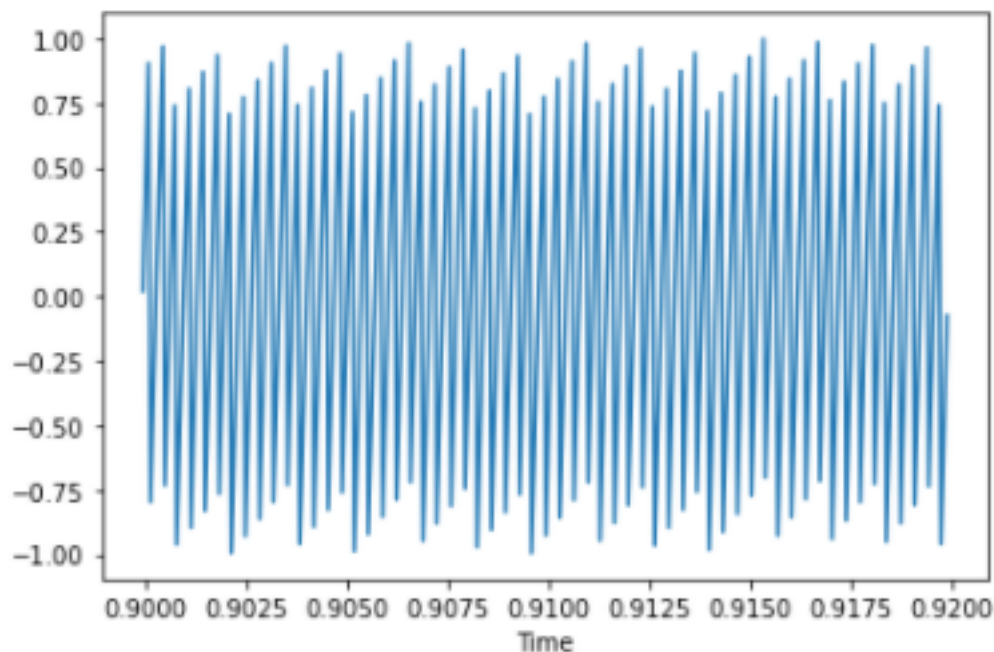


Рис. 11: Сегмент сигнала

Затем представим сигнал в виде аудио:

```
B [7]: wave.make_audio()
```

Out[7]:

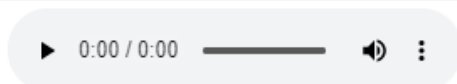


Рис. 12: Сигнал в аудио

И, наконец, построим и выведем на экран спектр нашего сигнала:

```
1 spectrum = wave.make_spectrum()  
2 spectrum.plot()  
3 decorate(xlabel='Frequency')
```

Листинг 13: Спектр сигнала

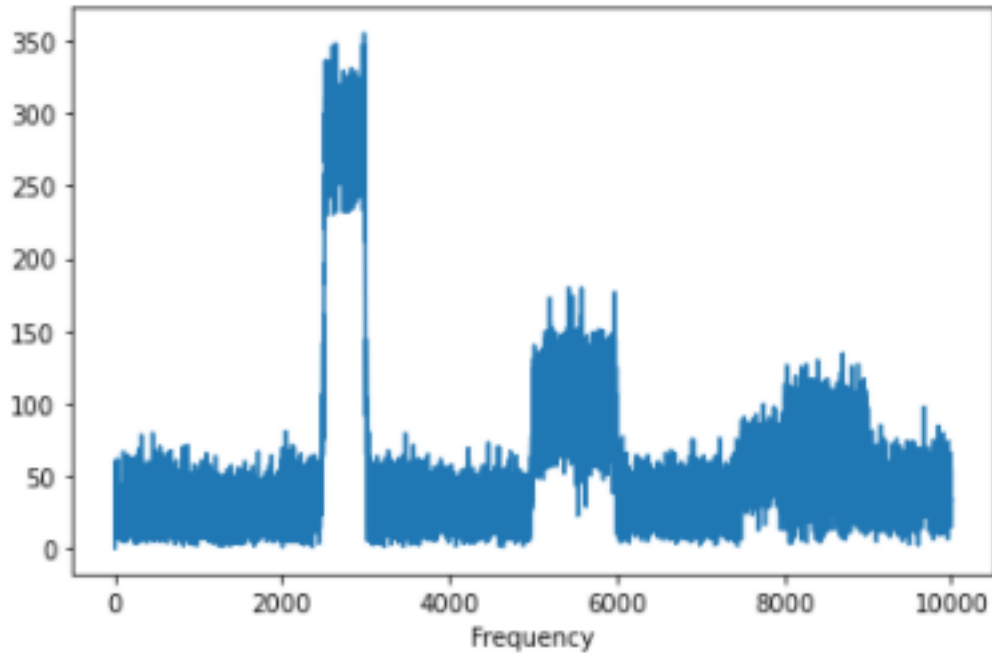


Рис. 13: Спектр сигнала

В заключение можно сказать, что полученный сигнал очень сильно режет слух, а проанализировав спектр можно сделать вывод, что он содержит большое количество частотных компонент.

4 Часть №4: Глиссандо

В четвертом пункте лабораторной работы нам необходимо сначала скачать звук глиссандо и распечатать спектограмму первых секунд.

В качестве аудиодорожки я выбрал «Rhapsody in Blue» (George Gershwin), как посоветовали в ThinkDSP.

```
1 from thinkdsp import read_wave
2
3 wave = read_wave('rhapblue11924.wav')
4 segment = wave.segment(start=1.35, duration=1.8-1.35)
5 segment.plot()
```

Листинг 14: Получение сегмента из аудиофайла

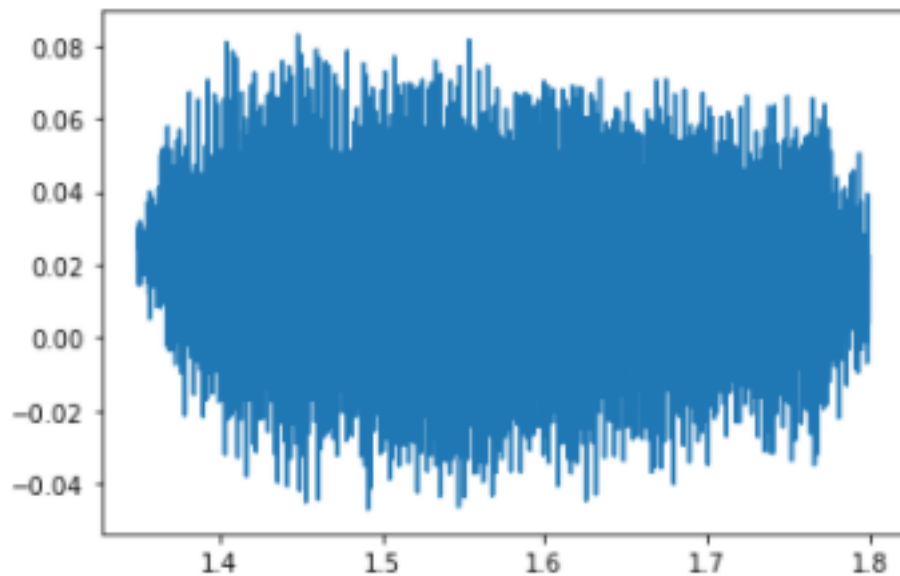


Рис. 14: Полученный сегмент

Затем для проверки выведем наш файл в аудио:

```
В [10]: wave.make_audio()
```

Out[10]:

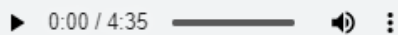


Рис. 15: Файл в аудио

После этого получим спектр из нашего сегмента:

```
1 spectrum = segment.make_spectrum()
2 spectrum.plot()
```

Листинг 15: Получение спектра из сегмента

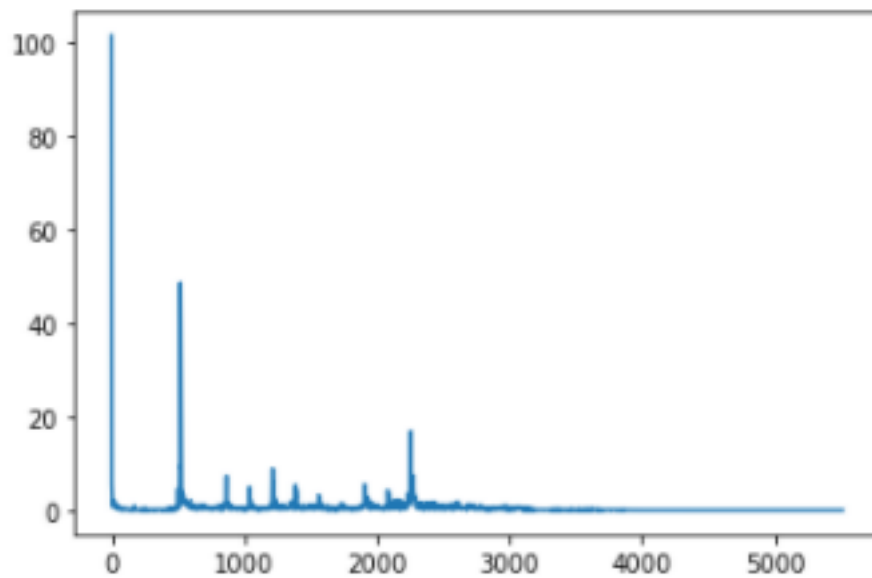


Рис. 16: Полученный спектр

И, наконец, получим спектограмму для нашего сигнала:

```
1 wave.make_spectrogram(512).plot(high=5000)
2 decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 16: Получение спектограммы

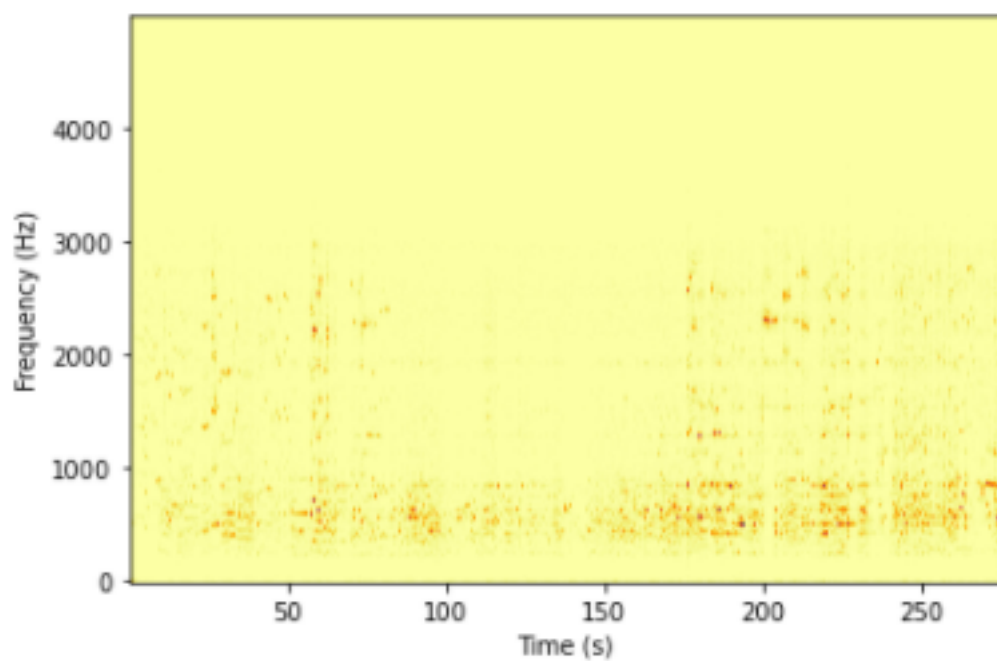


Рис. 17: Полученная спектограмма

5 Часть №5: Создание TromboneFliss

В пятом пункте лабораторной работы нам необходимо написать класс `TromboneFliss`, расширяющий `Chirp` и предоставляющий `evaluate`. Создать сигнал, имитирующий глиссандо на тромбоне от C3 до F3 и обратно.

Для начала напишем класс `TromboneFliss`:

```
1 spectrum = segment.make_spectrum()  
2 spectrum.plot()
```

Листинг 17: Класс `TromboneFliss`

Теперь получим сигнал от 262 до 349 Гц и представим его в аудио формате:

```
B [14]: signal = MyTromboneGliss(262, 349)  
wave1 = signal.make_wave(duration=1)  
wave1.apodize()  
wave1.make_audio()
```

Out[14]:

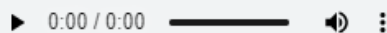


Рис. 18: Сигнал `up` в аудио

После этого посмотрим на сегмент этого сигнала:

```
1 wave1.segment(start=0, duration=0.04).plot()
```

Листинг 18: Сегмент сигнала `up`

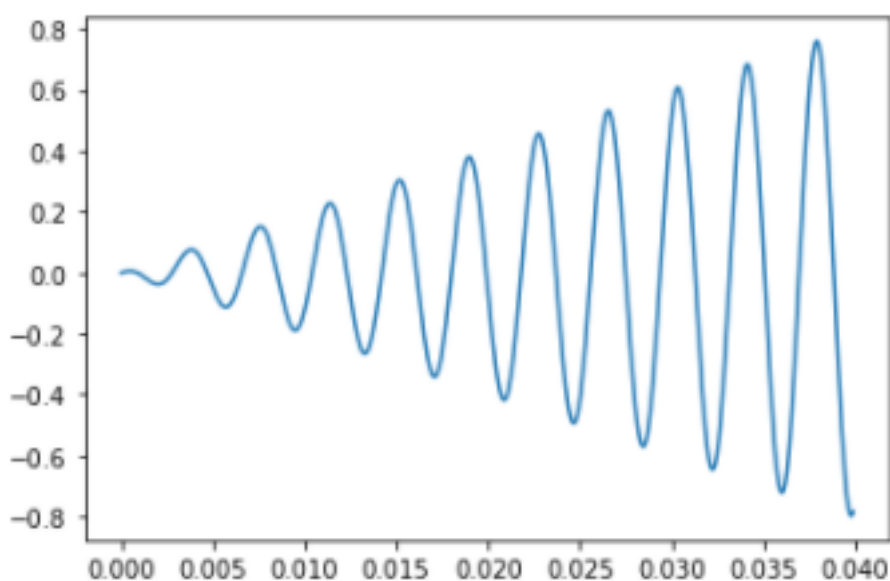


Рис. 19: Полученный сегмент `up`

Теперь получим обратный сигнал, от 349 до 262 Гц и так же представим его в виде аудио:

```
In [16]: signal = MyTromboneGliss(349, 262)
         wave2 = signal.make_wave(duration=1)
         wave2.apodize()
         wave2.make_audio()
```


Out[16]: 

Рис. 20: Сигнал **down** в аудио

И затем так же посмотрим на сегмент этого сигнала:

```
1 wave2.segment(start=0, duration=0.04).plot()
```

Листинг 19: Сегмент сигнала **down**

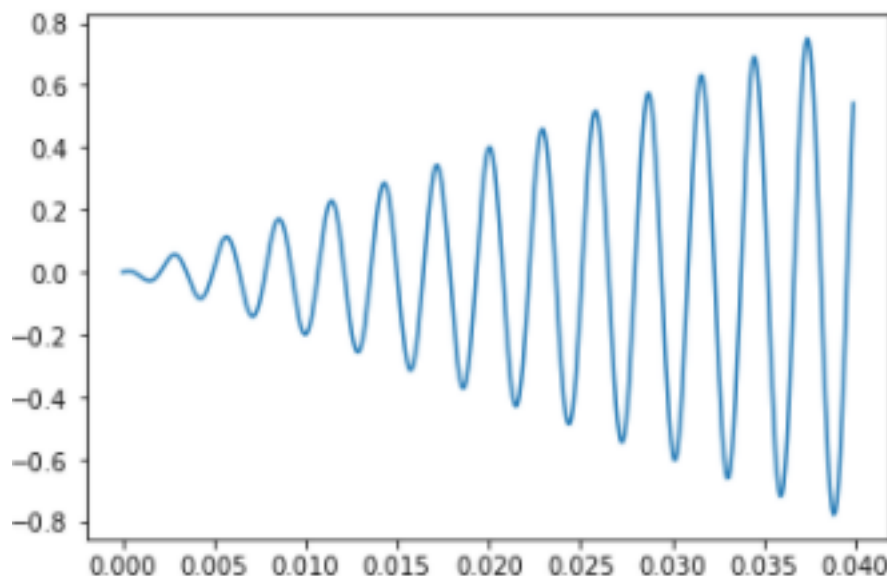


Рис. 21: Полученный сегмент **down**

После всего этого объединим эти два сигнала в один и получим аудиофайл:

```
In [18]: wave = wave1 | wave2
         wave.make_audio()
```


Out[18]: 

Рис. 22: Объединение **up** и **down** в аудио

Затем получим спектр полученного сигнала:

```
1 wave.make_spectrum().plot(high=500)
```

Листинг 20: Спектр сигнала up и down

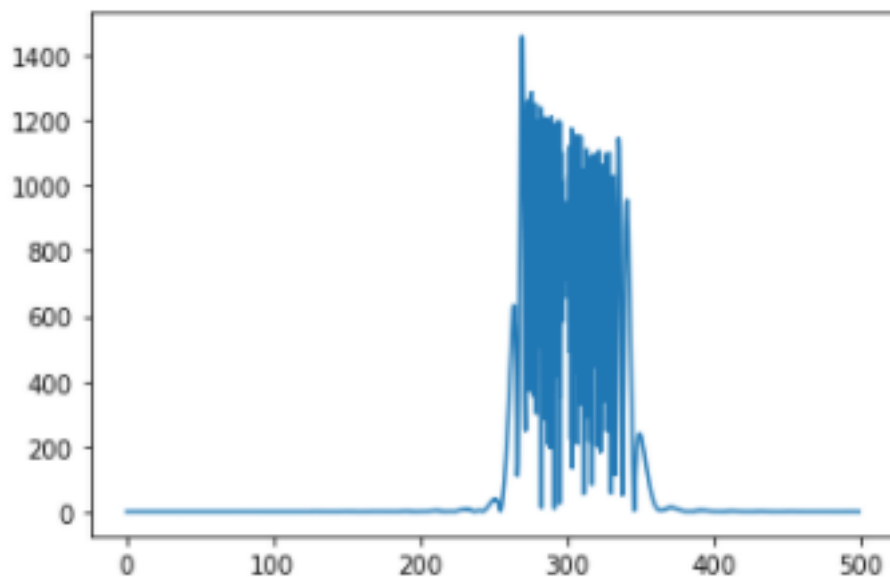


Рис. 23: Полученный спектр up и down

И, наконец, получим спектограмму полученного сигнала:

```
1 wave.make_spectrogram(1024).plot(high=600)
2 decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 21: Спектограмма сигнала up и down

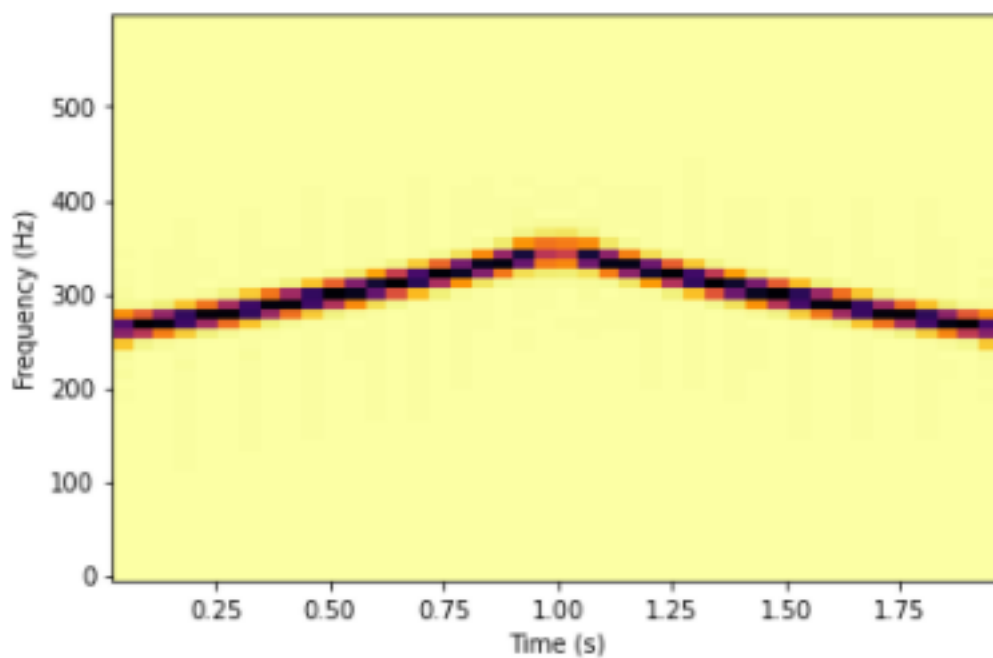


Рис. 24: Полученная спектограмма up и down

6 Часть №6: Анализ букв

В шестой части лабораторной работы нам необходимо найти запись серии звуков букв алфавита. Необходимо построить спектограммы для этих звуков.

В качестве аудиодорожки я взял запись с сайта FreeSounds, в которой проговариваются все буквы латинского алфавита, взял первый фрагмент с буквами а-е и после чего вывел данный сегмент:

```
1 wave = read_wave('67703__acclivity__alphabet-male.wav')
2 segment = wave.segment(start=1, duration=7)
3 segment.plot()
```

Листинг 22: Вывод сегмента

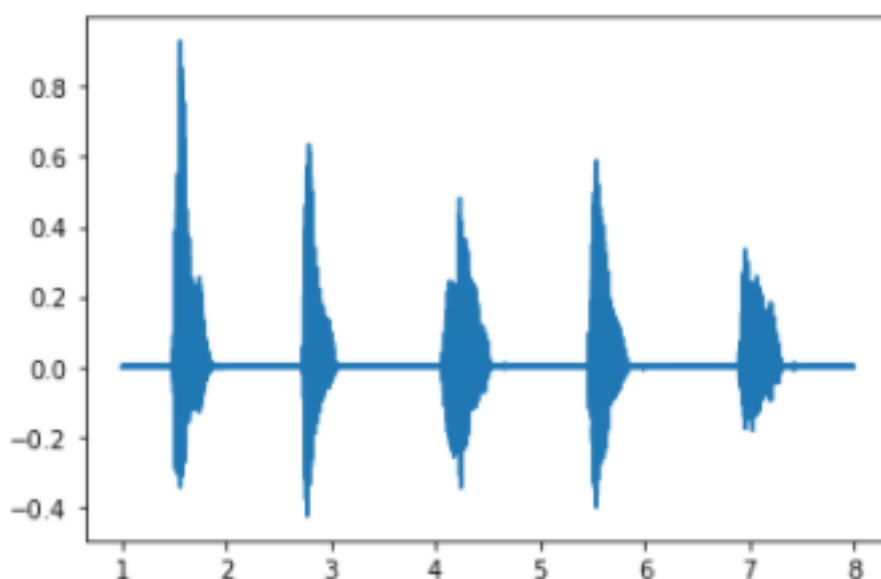


Рис. 25: Полученный сегмент

После этого проверим, что сигнал не поврежден, выведя его в видео аудио:

```
B [22]: wave.make_audio()
```

Out[22]:

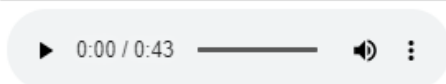


Рис. 26: Представление сигнала в аудио

Затем получим спектограмму для нашего сегмента:

```
1 segment.make_spectrogram(1024).plot(high=1000)
2 decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

Листинг 23: Спектограмма для нашего сегмента

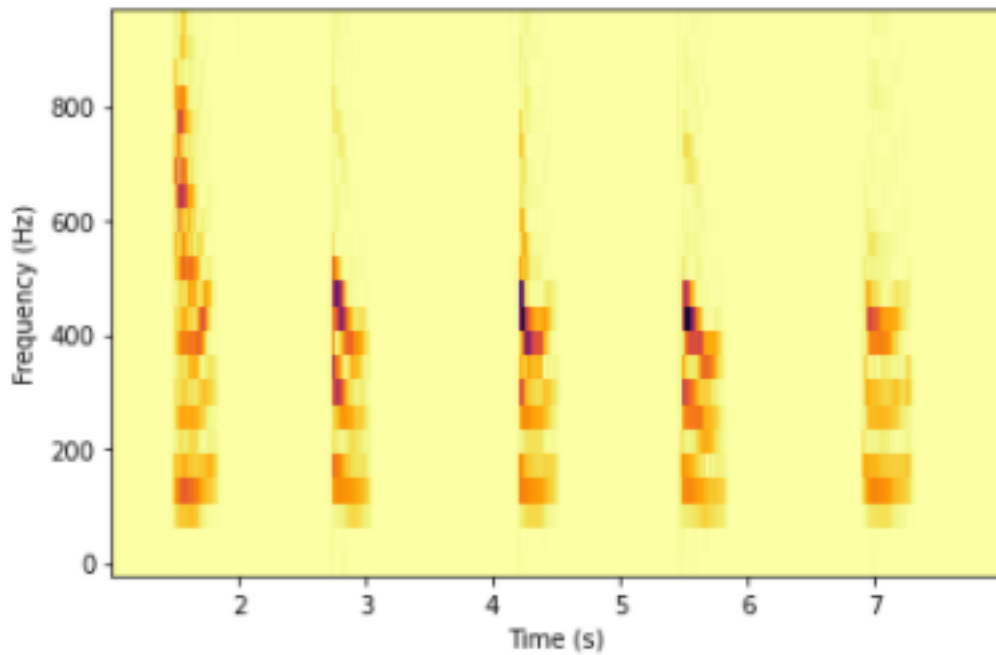


Рис. 27: Полученная спектограмма для сегмента

Теперь пройдемся по всем буквам из нашего сегмента, выводя сегменты на экран и представляя их в виде аудио. Начнем с первой буквы:

```

1  high = 1000
2  segment = wave.segment(start=1, duration=1)
3  segment.make_spectrum().plot(high=high)
4  decorate(xlabel='Frequency (Hz)')
```

Листинг 24: Сегмент с буквой А

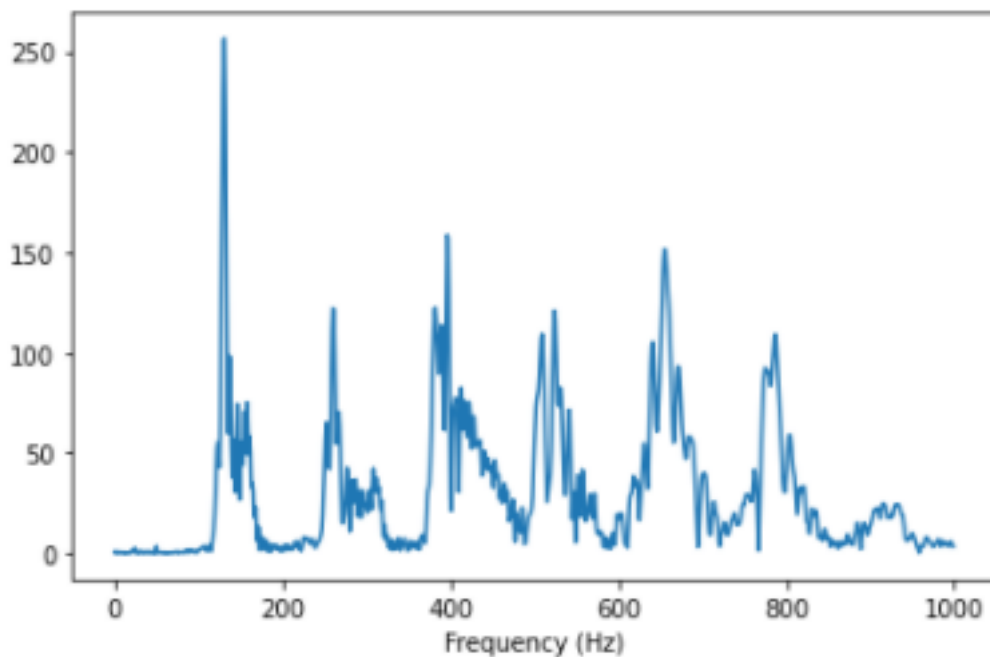


Рис. 28: Сегмент с буквой А

Представим сегмент с буквой А в виде аудиодорожки:

```
B [25]: segment.make_audio()
```

Out[25]:



Рис. 29: Представление сегмента с буквой А в аудио

Перейдем к сегменту с буквой В:

```
1 segment = wave.segment(start=2, duration=1)
2 segment.make_spectrum().plot(high=high)
3 decorate(xlabel='Frequency (Hz)')
```

Листинг 25: Сегмент с буквой В

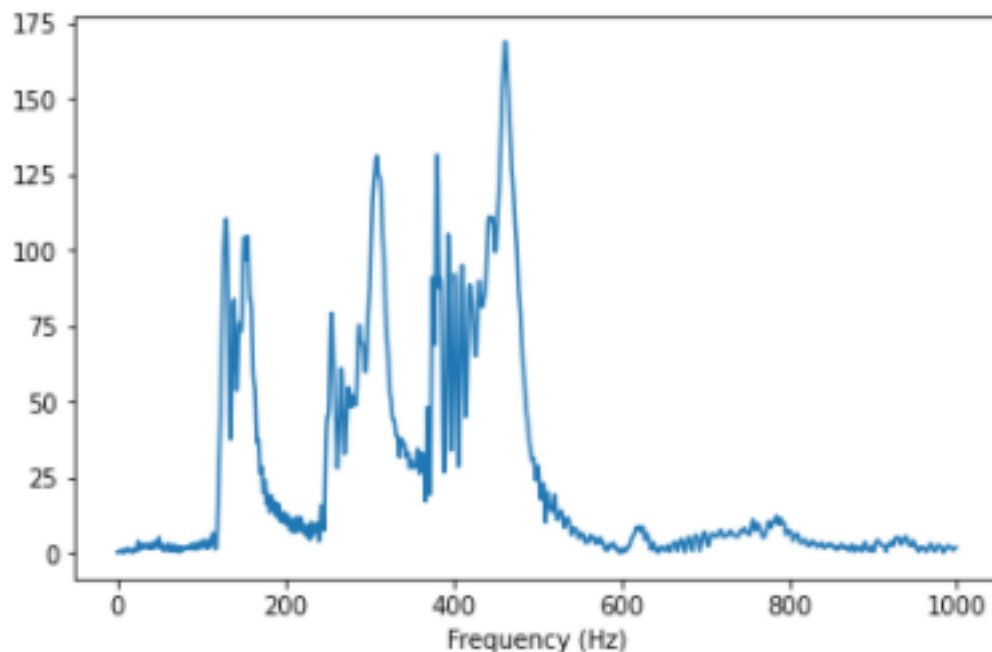


Рис. 30: Сегмент с буквой В

Представим сегмент с буквой В в виде аудиодорожки:

```
B [27]: segment.make_audio()
```

Out[27]:



Рис. 31: Представление сегмента с буквой В в аудио

Перейдем к сегменту с буквой С:


```

1 segment = wave.segment(start=3.5, duration=1)
2 segment.make_spectrum().plot(high=high)
3 decorate(xlabel='Frequency (Hz)')

```

Листинг 26: Сегмент с буквой С

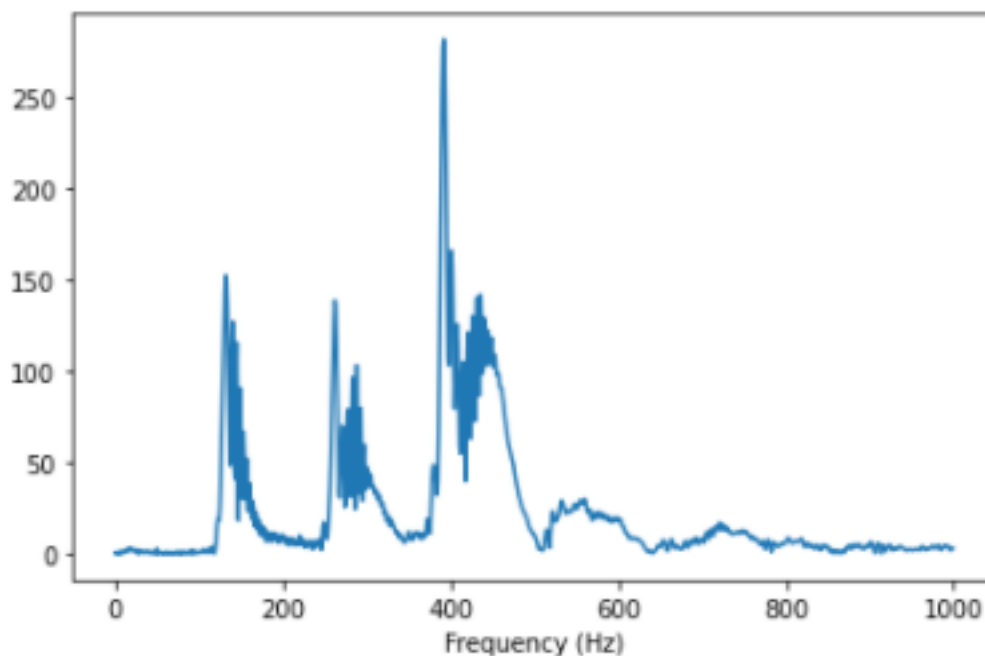


Рис. 32: Сегмент с буквой С

Представим сегмент с буквой С в виде аудиодорожки:

```

In [29]: segment.make_audio()

```

Out[29]:

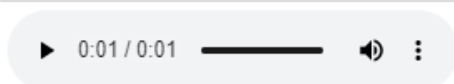


Рис. 33: Представление сегмента с буквой С в аудио

Перейдем к сегменту с буквой D:

```

1 segment = wave.segment(start=5, duration=1)
2 segment.make_spectrum().plot(high=high)
3 decorate(xlabel='Frequency (Hz)')

```

Листинг 27: Сегмент с буквой D

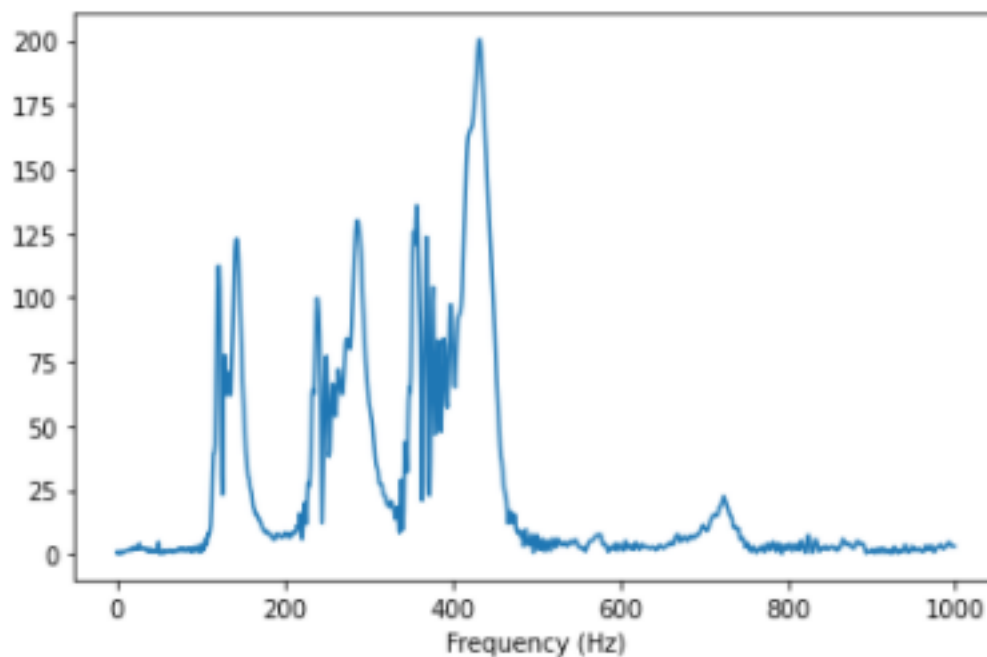


Рис. 34: Сегмент с буквой D

Представим сегмент с буквой D в виде аудиодорожки:

```
In [31]: segment.make_audio()
```

Out[31]:



Рис. 35: Представление сегмента с буквой D в аудио

Перейдем к сегменту с буквой E:

```
1 segment = wave.segment(start=6.5, duration=1)
2 segment.make_spectrum().plot(high=high)
3 decorate(xlabel='Frequency (Hz)')
```

Листинг 28: Сегмент с буквой E

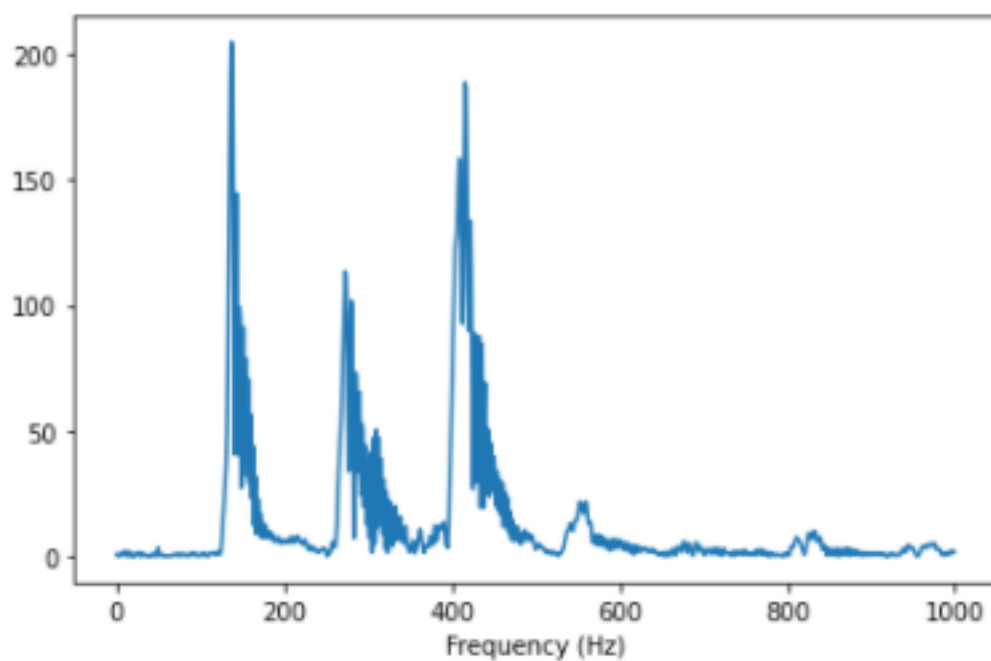


Рис. 36: Сегмент с буквой E

Представим сегмент с буквой E в виде аудиодорожки:

```
B [33]: segment.make_audio()
```

Out[33]:



Рис. 37: Представление сегмента с буквой E в аудио

В результате выполнения данного пункта можно сделать вывод о том, что гласные звуки при представлении их в виде сегмента имеют более выраженный пик и большую частоту, чем согласные звуки.

7 Выводы

В результате выполнения лабораторной работы мы изучили методы работы с апериодическими сигналами, понятие чирпа, каков его спектр. Также научились строить спектограммы чирпов, работать с утечками. Кроме того были реализованы и проверены классы для реализации чирпа, и для создания имитации глиссандо на тромбоне.