Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Отчёт по лабораторной работе №3
Дисциплина: Телекоммуникационные технологии
Тема: Апериодические сигналы

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

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

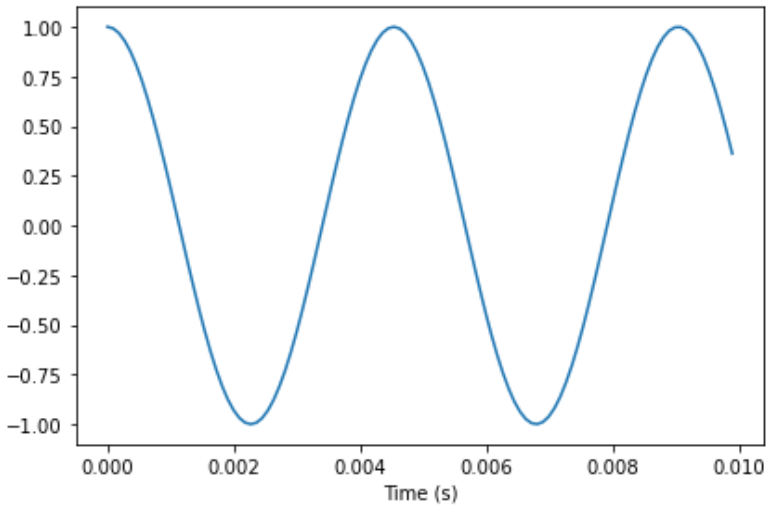
1 Упражнение 3.1

В этому упражнении необходимо загрузить файл `chap03.ipynb`, изучить его, просмотреть пояснения и запустить примеры. Кроме того, в примере с утечкой предлагается заменить окно Хэмминга на любое другое, предоставляемое NumPy.

Итак, изучим примеры.

Here's what the waveform looks like near the beginning.

```
wave1.segment(start=0, duration=0.01).plot()
decorate(xlabel='Time (s)')
```



And near the end.

```
wave1.segment(start=0.9, duration=0.01).plot()
decorate(xlabel='Time (s)')
```

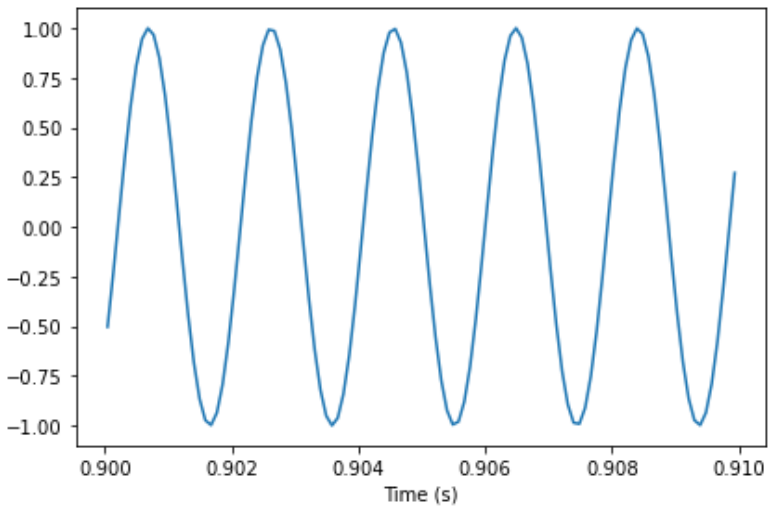
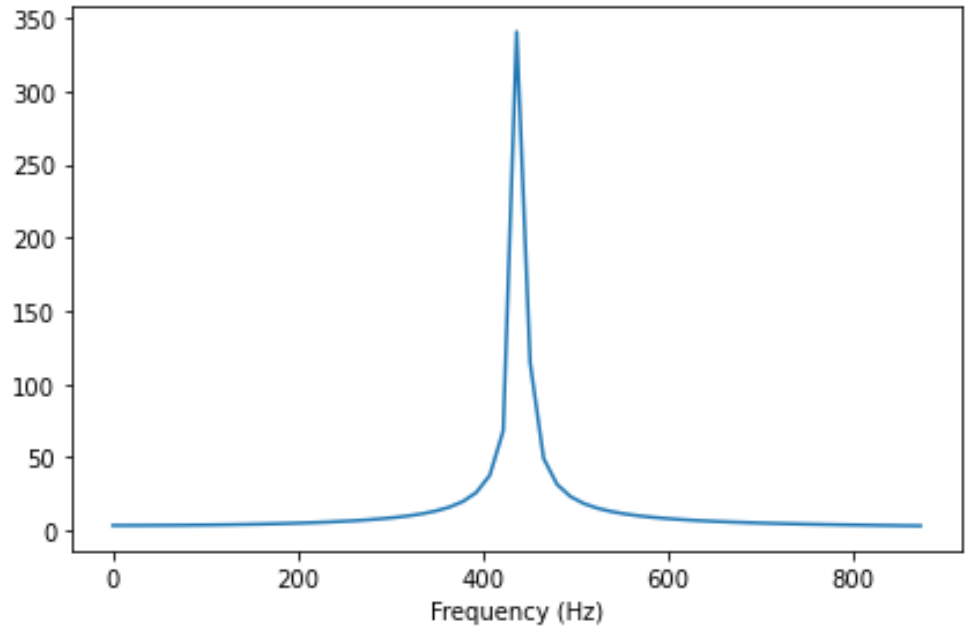


Рис. 1: Изучение и проверка примеров (Chirp).

```
spectrum = wave.make_spectrum()
spectrum.plot(high=880)
decorate(xlabel='Frequency (Hz)')
```



Windowing helps (but notice that it reduces the total energy).

```
wave.hamming()
spectrum = wave.make_spectrum()
spectrum.plot(high=880)
decorate(xlabel='Frequency (Hz)')
```

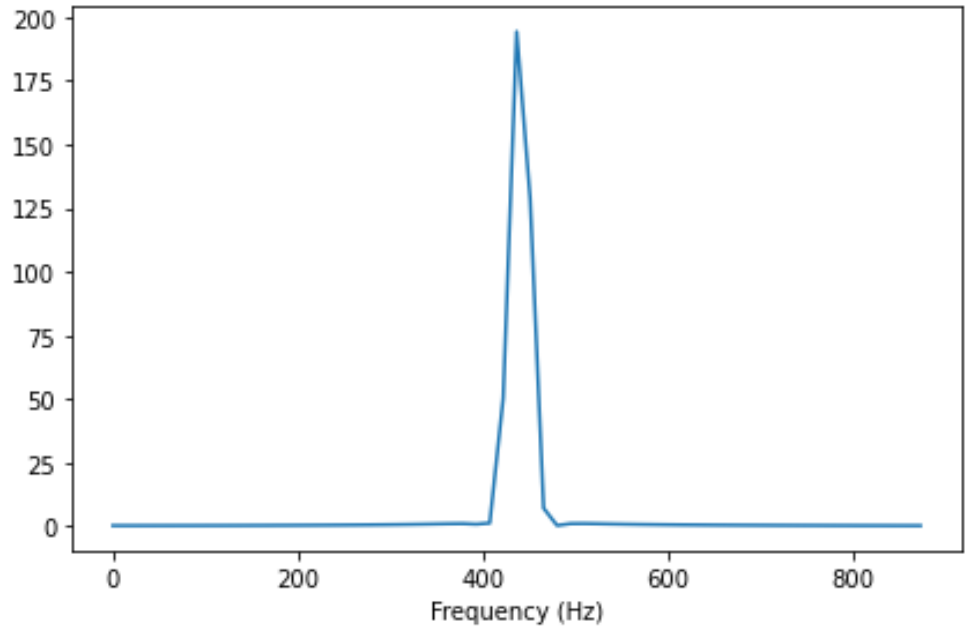


Рис. 2: Изучение и проверка примеров (Влияние использования окон на утечку).

A spectrogram is a visualization of a short-time DFT that lets you see how the spectrum varies over time.

```
def plot_spectrogram(wave, seg_length):
    """
    """
    spectrogram = wave.make_spectrogram(seg_length)
    print('Time resolution (s)', spectrogram.time_res)
    print('Frequency resolution (Hz)', spectrogram.freq_res)
    spectrogram.plot(high=700)
    decorate(xlabel='Time(s)', ylabel='Frequency (Hz)')
```

```
signal = Chirp(start=220, end=440)
wave = signal.make_wave(duration=1, framerate=11025)
plot_spectrogram(wave, 512)
```

```
Time resolution (s) 0.046439909297052155
Frequency resolution (Hz) 21.533203125
```

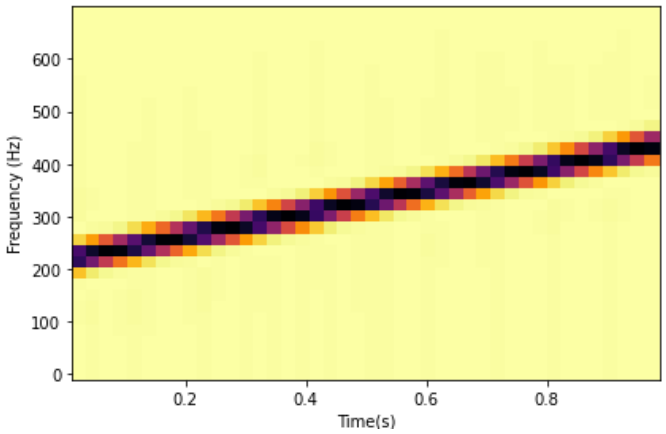


Рис. 3: Изучение и проверка примеров (Спектрограмма).

Все примеры были изучены и запущены. Теперь попробуем использовать другие окна в примере с Хэммингом. Для этого сначала создадим аналогичный сигнал, получим его `wave` и вычислим спектр с использованием окна Бартлетта.

```
signal = SinSignal(freq=440)
wave = signal.make_wave(signal.period * 30.25)

wave.window(numpy.bartlett(len(wave)))
spectrum = wave.make_spectrum()
spectrum.plot(high=880)
decorate(xlabel='Frequency (Hz)')

```

Листинг 1: Создание сигнала и получение спектра с использованием `bartlett`.

Теперь получим спектр, используя окно Блэкмана.

```
wave.window(numpy.blackman(len(wave)))
spectrum = wave.make_spectrum()
spectrum.plot(high=880)
decorate(xlabel='Frequency (Hz)')

```

Листинг 2: Получение спектра с использованием `blackman`.

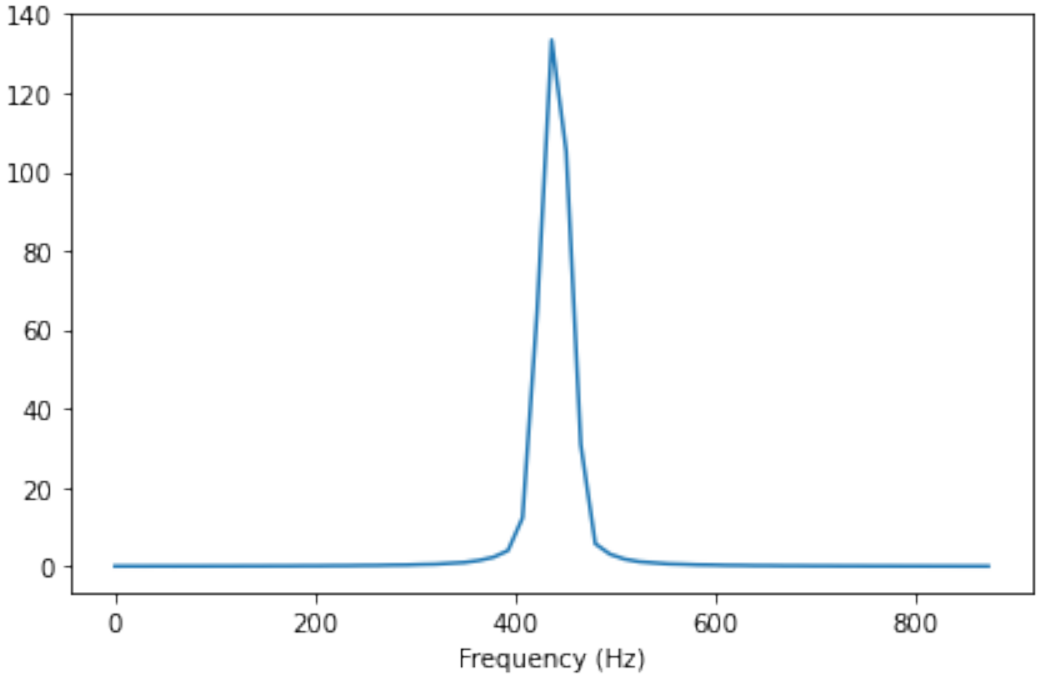


Рис. 4: Спектр, полученный с помощью окна bartlett.

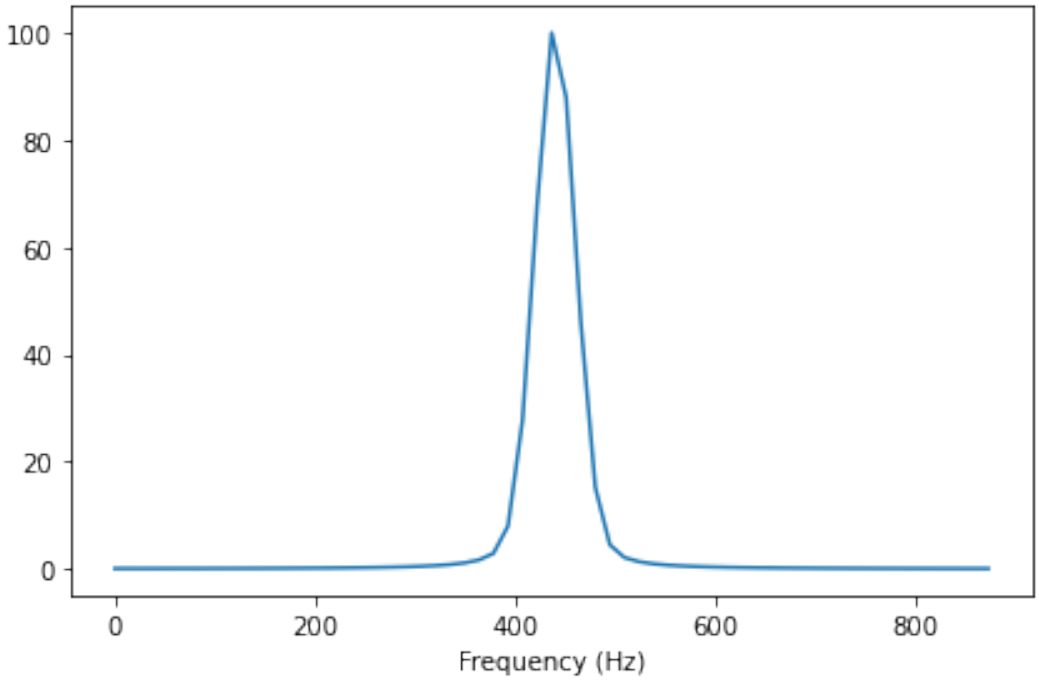


Рис. 5: Спектр, полученный с помощью окна blackman.

Если сравнить полученные спектры со спектром, полученным без использования оконных функций (Рис.2, верхний график), можно заметить, что утечки действительно уменьшились. Однако, при сравнении этих результатов с результатом после использования окна Хэмминга (Рис.2, нижний график) видно, что общая энергия становится ещё ниже. Это связано с тем, что окно Хэмминга - это

окно-«вездеход», а вот окна Бартлетта и Блэкмана являются оптимальными для некоторых случаев.

В ходе выполнения данного упражнения были получены знания работе с `Chirp`, утечками и оконными функциями. Также были рассмотрены спектрограммы и их возможности. Кроме того, были сравнены результаты применения различных оконных функций для получения спектра. Был сделан вывод, что окна Бартлетта и Блэкмана оптимальны в некоторых случаях, а окно Хэмминга является универсальным.

2 Упражнение 3.2

Во втором упражнении необходимо написать класс `SawtoothChirp`, расширяющий `Chirp` и переопределяющий `evaluate` для генерации пилообразного сигнала с линейно увеличивающейся (или уменьшающейся) частотой.

Итак, начнём с написания класса `SawtoothChirp`.

```
PI2 = numpy.pi * 2

class SawtoothChirp(Chirp):
    def evaluate(self, ts):
        freqs = numpy.linspace(self.start, self.end, len(ts))
        dts = numpy.diff(ts, prepend=0)
        dphis = PI2 * freqs * dts
        phases = numpy.cumsum(dphis)
        cycles = phases / PI2
        # From SawtoothSignal
        frac, _ = numpy.modf(cycles)
        ys = normalize(unbias(frac), self.amp)
        return ys

```

Листинг 3: Создание класса `SawtoothChirp`.

Данный класс расширяет класс `Chirp` и переопределяет его метод `evaluate`.

Теперь проверим работу нашего класса, создав пилообразный `chirp`. Затем преобразуем его к `wave` и получим аудио.

```
sawtooth_signal = SawtoothChirp(start=900, end=350)
sawtooth_wave = sawtooth_signal.make_wave(duration=1, framerate=4000)
sawtooth_wave.apodize()
sawtooth_wave.make_audio()

```

Листинг 4: Создание и работа с сигналом.

Звук очень необычный и похож на включенный задом наперёд звук сирены. Теперь построим спектрограмму нашего сигнала.

```
spectrogram = sawtooth_wave.make_spectrogram(200)
spectrogram.plot()
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')

```

Листинг 5: Создание и вывод спектрограммы.

На полученной спектрограмме хорошо видно, как гармоники с наложенными частотами отражаются от частоты заворота. Именно так на спектрограмме можно увидеть эффект биений. Кроме того, если вслушаться в полученный звук, то биения можно услышать как фоновое шипение.

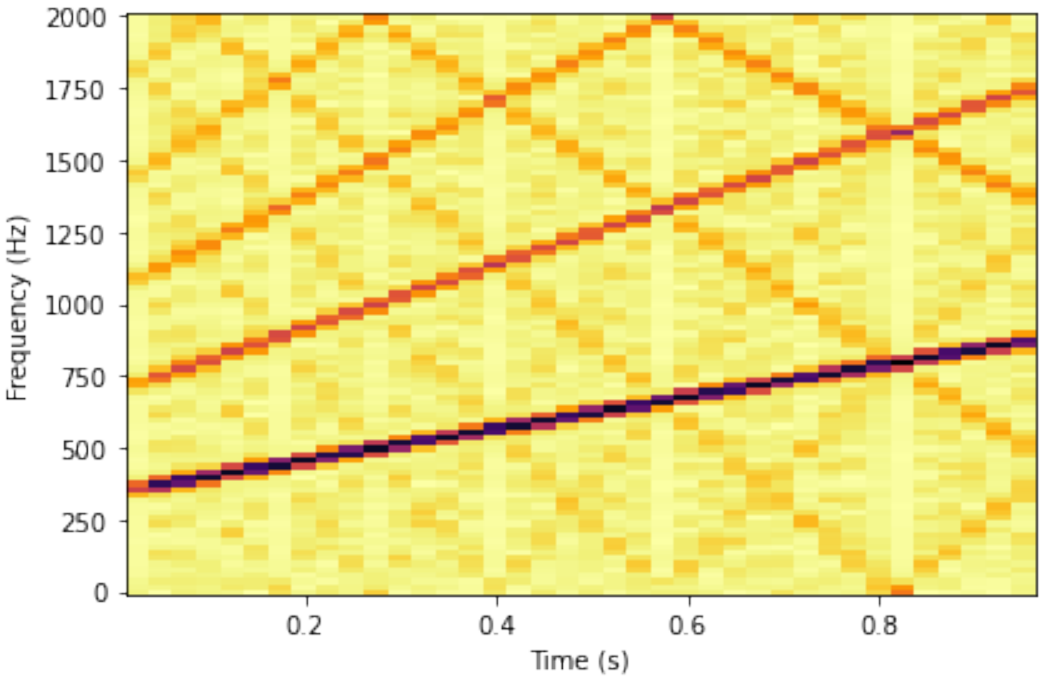


Рис. 6: Спектрограмма полученного сигнала.

В ходе выполнения данного задания был описан класс `SawtoothChirp`, генерирующий пилообразный сигнал с линейно увеличивающейся (или уменьшающейся) частотой. С помощью этого класса был создан сигнал, после чего была получена его спектрограмма. На спектрограмме хорошо виден эффект биений.

3 Упражнение 3.3

В третьем упражнении необходимо создать пилообразный `chirp` с частотой, меняющейся от 2500 до 3000 Гц. Затем на основе этого сигнала сгенерировать сигнал длительностью 1 с и частотой кадров 20 кГц. После этого необходимо предположить, каким будет его спектр, и сравнить предположения с самим спектром.

Итак, создадим пилообразный `chirp` с требуемой частотой и сгенерируем сигнал длительностью 1 с и частотой кадров 20 кГц. Тут же прослушаем его.

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

```

Листинг 6: Создание сигнала, получение `wave` и аудио.

Звук этого сигнала сильно режет слух и кажется слишком резким и пищащим.

Предположим, какой спектр будет иметь этот сигнал. Так как частота изменяется от 2500 до 3000 Гц, то в этом диапазоне ожидается увидеть всплеск. Кроме того, мы помним, что пилообразный сигнал имеет и чётные, и нечётные гармоники. Первая гармоника ожидается в диапазоне от 5000 до 6000 Гц, однако она должна быть поменьше предыдущего всплеска. Вторая гармоника тогда должна быть ещё меньше находиться в диапазоне от 7500 до 9000 Гц. Следующие гармоники накладываются друг на друга, а потому на всех остальных частотах должны быть небольшие колебания.

Теперь, наконец, получим спектр нашего сигнала и сравним его с ожидаемым спектром.

```
sawtooth_wave.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)')

```

Листинг 7: Создание и вывод спектра.

Как видно из Рис.7 наши ожидания совпали с реальным спектром.

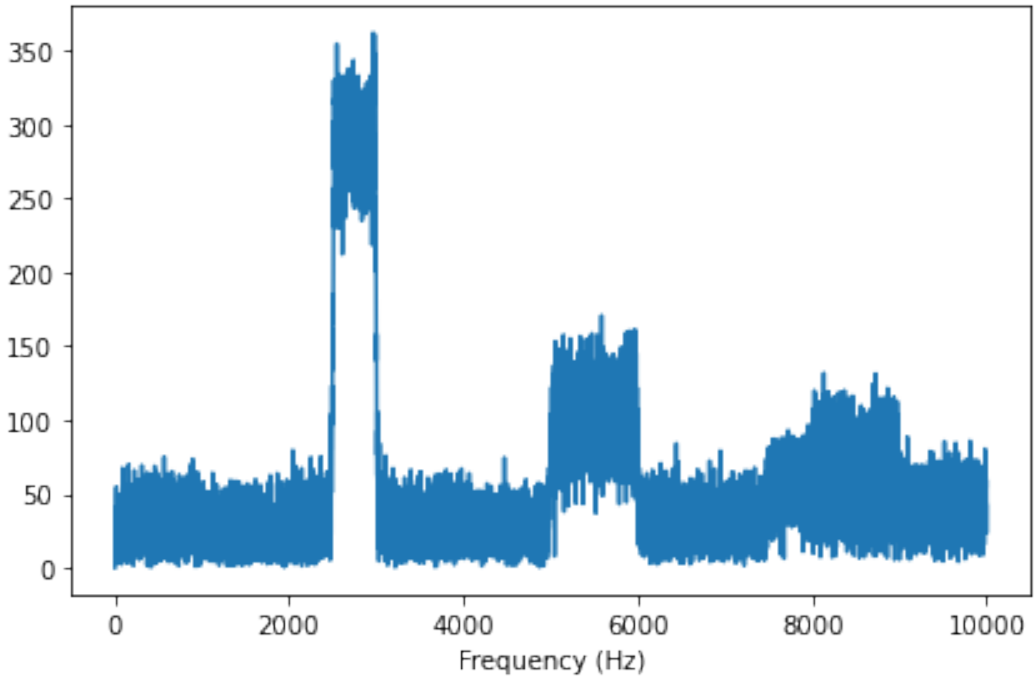


Рис. 7: Спектр полученного сигнала.

В ходе выполнения данного упражнения был создан пилообразный `chirp` с частотой, меняющейся от 2500 до 3000 Гц. После этого был сгенерированы его аудио и `wave`. Полученный сигнал режет слух и кажется очень неприятным и пищащим. Далее было сделано предположение о том, как будет выглядеть его спектр. После чего был получен спектр и сравнён с ожиданиями. Наше ожидания совпали с самим спектром.

4 Упражнение 3.4

В этом упражнении происходит работа с глиссандо. Глиссандо - это нота, меняющаяся от одной высоты до другой, т.е. своеобразный `chirp`. Необходимо найти звук глиссандо и изучить спектрограмму его нескольких секунд.

Итак, была выбрана эта запись глиссандо скрипки. Теперь считаем файл и выведем аудио.

```
wave = read_wave('resources/Sounds/task4_violin_glissando.wav')
wave.make_audio()

```

Листинг 8: Считывание и воспроизведение файла.

Теперь выделим небольшой сегмент и получим его аудио.

```
segment = wave.segment(start=0.17, duration=1)
segment.make_audio()

```

Листинг 9: Работа с сегментом.

И наконец, получим спектрограмму этого сегмента.

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

```

Листинг 10: Получение спекограммы.

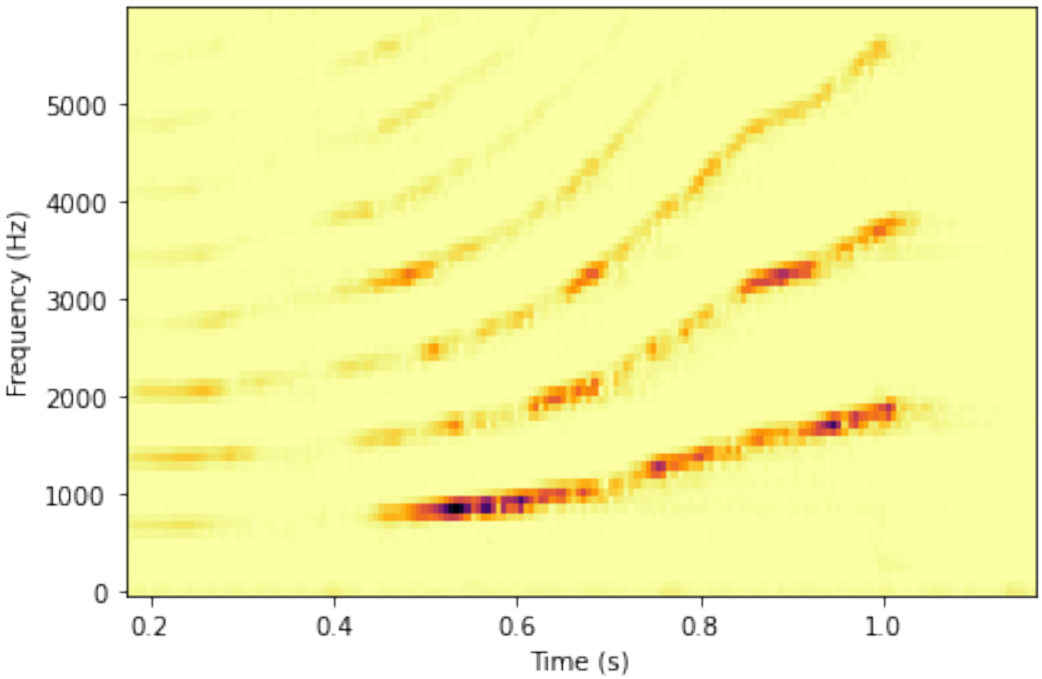


Рис. 8: Спектрограмма звука глиссандо.

Полученная спектрограмма выглядит очень интересно, на ней хорошо заметно глиссандо.

В ходе выполнения данного упражнения была получена спектрограмма глиссандо скрипки.

5 Упражнение 3.5

В этом упражнении необходимо написать класс `TromboneGliss`, расширяющий `Chirp` и переопределяющий `evaluate`. Затем необходимо создать сигнал, имитирующий глиссандо на тромбоне от C3 (262 Гц) до F3 (349 Гц) и обратно до C3. После этого нужно изучить спектрограмму полученного сигнала.

Итак, сначала напишем класс `TromboneGliss`.

```
class TromboneGliss(Chirp):
    def evaluate(self, ts):
        l1, l2 = 1.0 / self.start, 1.0 / self.end
        lengths = numpy.linspace(l1, l2, len(ts))
        freqs = 1 / lengths
        dts = numpy.diff(ts, prepend=0)
        dphis = PI2 * freqs * dts
        phases = numpy.cumsum(dphis)
        ys = self.amp * numpy.cos(phases)
        return ys

```

Листинг 11: Класс `TromboneGliss`.

Этот класс расширяет `Chirp` и переопределяет `evaluate`. Теперь создадим первую половину сигнала: глиссандо от C3 (262 Гц) до F3 (349 Гц). Тут же прослушаем его.

```
signal_up = TromboneGliss(262, 349)
wave_up = signal_up.make_wave(duration=1)
wave_up.make_audio()

```

Листинг 12: Создание сигнала глиссандо от C3 до F3.

Полученный звук действительно напоминает глиссандо. Теперь создадим сигнал глиссандо от F3 до C3 и прослушаем его.

```
signal_down = TromboneGliss(349, 262)
wave_down = signal_down.make_wave(duration=1)
wave_down.make_audio()

```

Листинг 13: Создание сигнала глиссандо от F3 до C3.

Осталось только склеить оба этих сигнала, вывести спектрограмму и прослушать результат.

```
wave_up_down = wave_up | wave_down
spectrogram = wave_up_down.make_spectrogram(1024)
spectrogram.plot(high=1000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
wave_up_down.make_audio()

```

Листинг 14: Создание итогового сигнала и получение его спектрограммы.

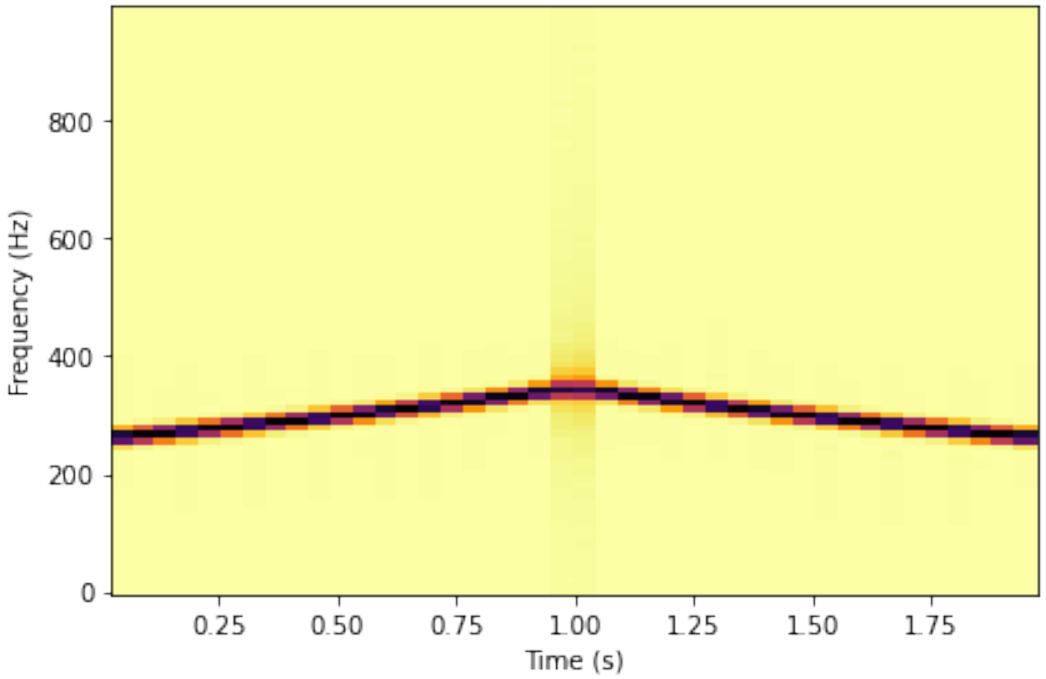


Рис. 9: Спектрограмма итогового сигнала.

Как видно из спектрограммы (Рис.9), полученный сигнал больше похож на линейный `chirp`. Его же звук действительно напоминает глиссандо от C3 до F3 и обратно в C3.

В ходе выполнения данного упражнения был написан класс `TromboneGliss`, генерирующий сигнал глиссандо на тромбоне. С помощью этого класса был создан сигнал, имитирующий глиссандо на тромбоне от C3 до F3 и обратно до C3. У этого сигнала была получена спектрограмма, изучив которую, был сделан вывод, что этот сигнал больше похож на линейный `chirp`.

6 Упражнение 3.6

В этом упражнении необходимо найти запись серии гласных звуков и посмотреть на их спектрограмму. Эксперимент заключается в попытке определения различных гласных звуков на спектрограмме.

Был выбран первый отрывок гласных из этой записи. На этом отрывке ребёнок произносит звуки «А», «Э», «И», «О» и «У». Теперь считаем файл, получим необходимый сегмент, прослушаем его и вывеем спектрограмму.

```
w = read_wave('resources/Sounds/task6_vowels.wav')
wave = w.segment(start=0, duration=4.4)
wave.make_spectrogram(1024).plot(high=1000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
wave.make_audio()

```

Листинг 15: Формирование нужного сгемента и вывод спектрограммы.

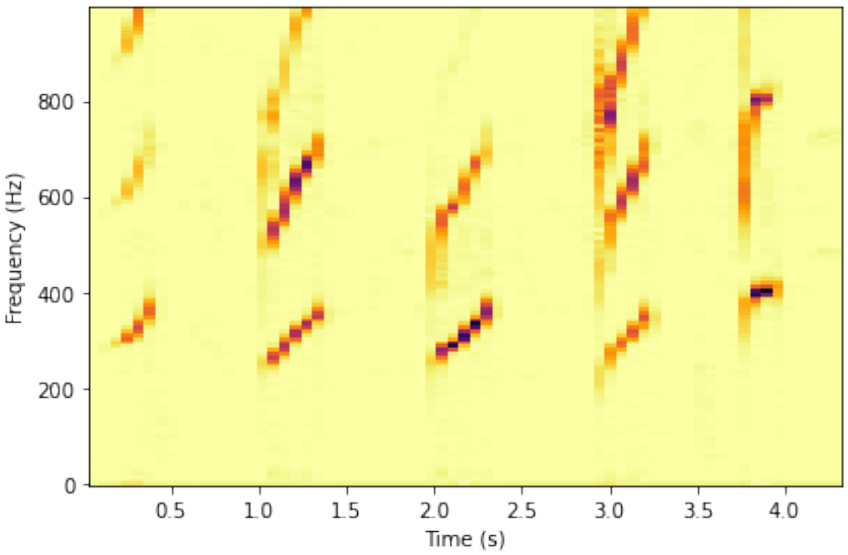


Рис. 10: Спектрограмма серии гласных звуков.

Итак, мы получили спектрограмму гласных звуков. Догадаться, какой именно звук представлен на спектрограмме, я могу только по порядку. Также из спектрограммы видно, что звук «И» имеет самую малую частоту, остальные же звуки на спектрограмме очень похожи.

Теперь получим сегмент со звуком «А» и изучим его спектр.

```
segment_a = wave.segment(start=0.25, duration=0.7)
segment_a.make_spectrum().plot(high=1000)
segment_a.make_audio()

```

Листинг 16: Вывод спектра звука «А».

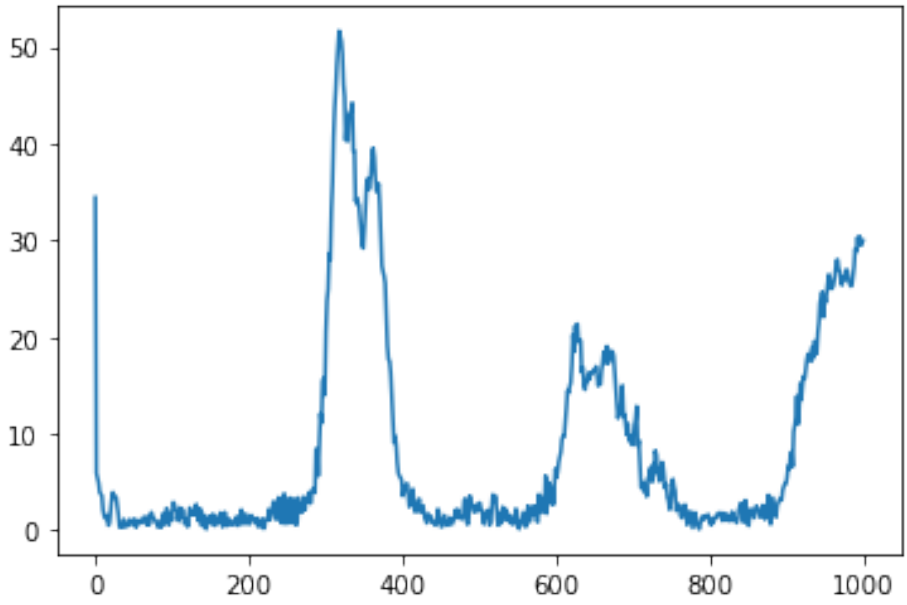


Рис. 11: Спектр звука «А».

Из спектра видно, что самая высокоамплитудная частота находится примерно на 300 Гц. Теперь получим сегмент со звуком «Э» и изучим его спектр.

```
segment_e = wave.segment(start=0.85, duration=0.7)
segment_e.make_spectrum().plot(high=1000)
segment_e.make_audio()

```

Листинг 17: Вывод спектра звука «Э».

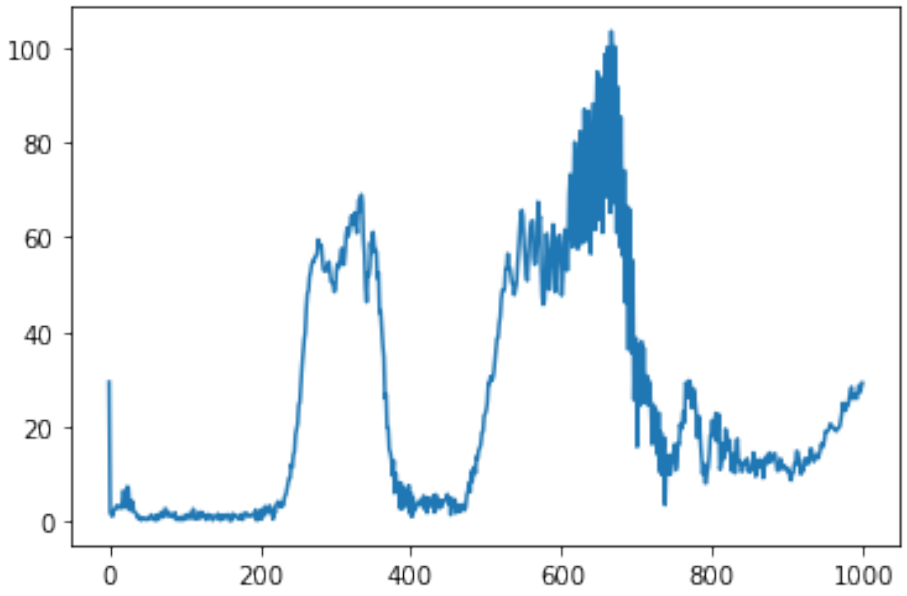


Рис. 12: Спектр звука «Э».

Сегмент «Э» имеет самые высокие пики примерно на 700 и 350 Гц. Теперь

получим сегмент со звуком «И» и изучим его спектр.

```
segment_i = wave.segment(start=1.8, duration=0.7)
segment_i.make_spectrum().plot(high=1000)
segment_i.make_audio()

```

Листинг 18: Вывод спектра звука «И».

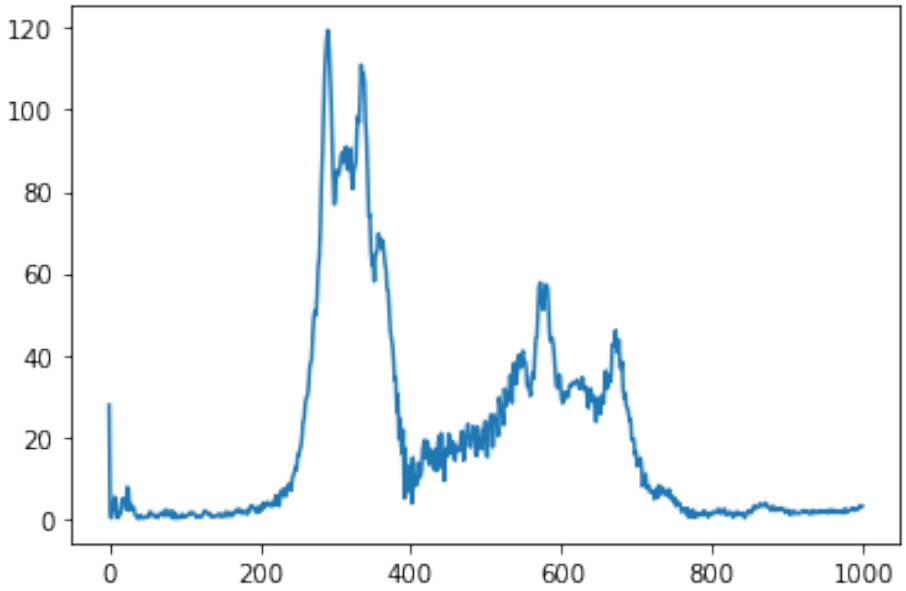


Рис. 13: Спектр звука «И».

Сегмент «И» имеет высокие пики примерно на 300 и 600 Гц. Теперь получим сегмент со звуком «О» и изучим его спектр.

```
segment_o = wave.segment(start=2.6, duration=0.7)
segment_o.make_spectrum().plot(high=1000)
segment_o.make_audio()

```

Листинг 19: Вывод спектра звука «О».

Сегмент «О» имеет высокий пик примерно на 750 Гц. Теперь получим сегмент со звуком «У» и изучим его спектр.

```
segment_u = wave.segment(start=3.7, duration=0.5)
segment_u.make_spectrum().plot(high=1000)
segment_u.make_audio()

```

Листинг 20: Вывод спектра звука «У».

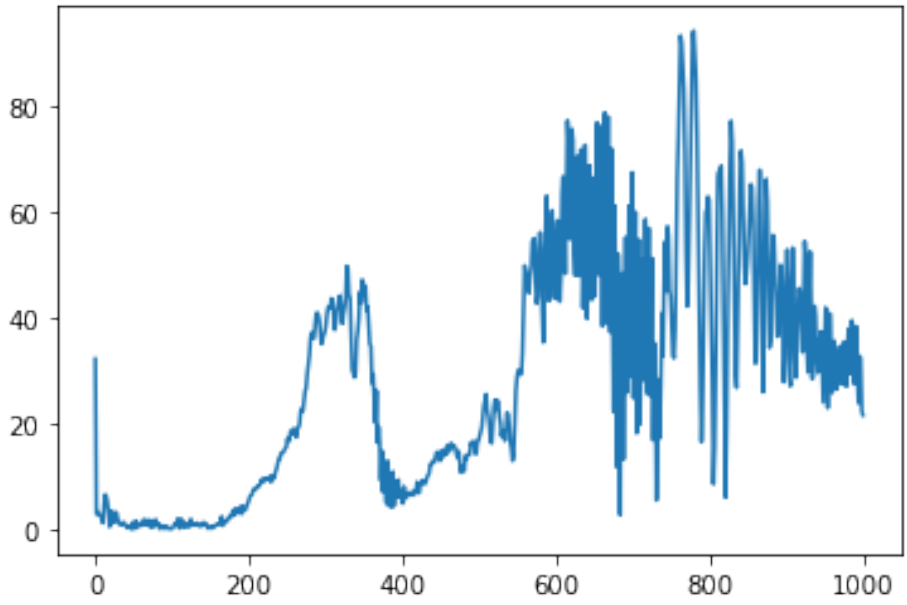


Рис. 14: Спектр звука «О».

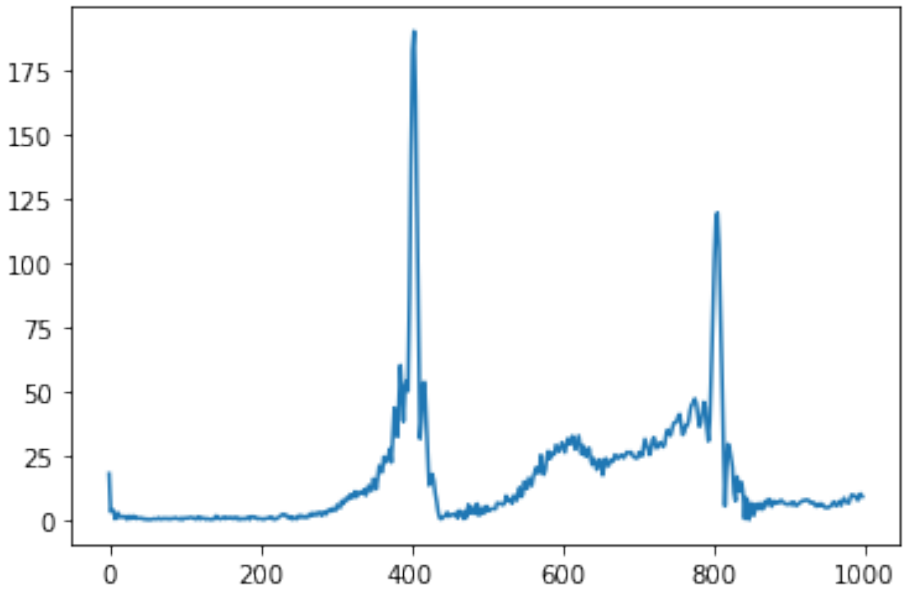


Рис. 15: Спектр звука «У».

Сегмент «У» имеет самые высокие пики на 400 и 800 Гц.

В ходе этого упражнения была найдена и проанализирована запись серии гласных звуков, а также изучена её спектрограмма. Кроме того, были изучены спектры отдельных гласных звуков.

7 Выводы

В ходе выполнения данной лабораторной работы были изучены спектры, спектрограммы, `chirp` и глиссандо. Был создан класс `SawtoothChrip`, генерирующий пилообразный сигнал с линейно увеличивающейся (или уменьшающейся) частотой, после чего этот класс был протестирован. Также был создан и протестирован класс `TromboneGliss`, генерирующий сигнал глиссандо на тромбоне. Кроме того, были изучены эффект утечек и оконные функции, сформирована и изучена спектрограмма гласных звуков и спектры отдельных звуков.