Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Отчёт по лабораторной работе №1
Дисциплина: Телекоммуникационные технологии
Тема: Звуки и сигналы

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

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение 1.1

Загрузим файл `chap01.ipynb`, изучим его, прочитаем пояснения и запустим примеры. Проверим, что всё работает корректно.

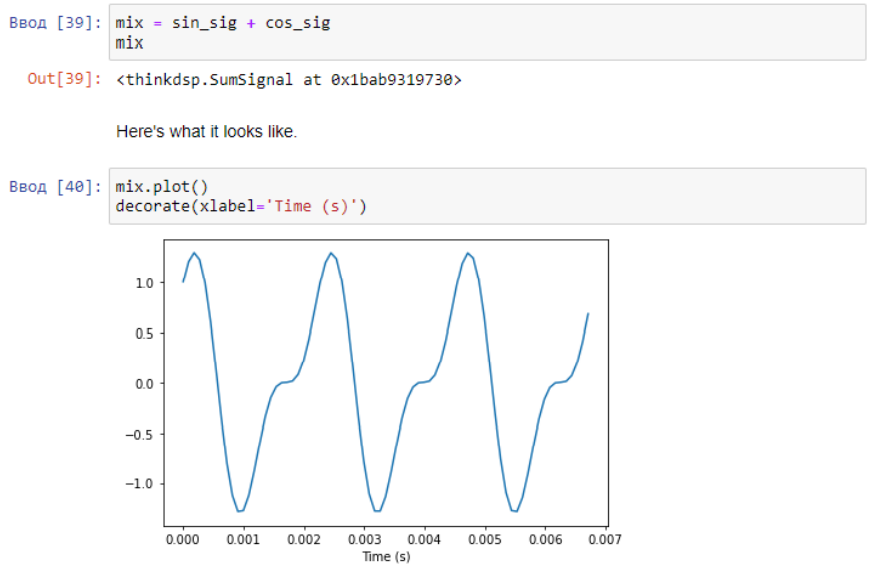

Рис. 1: Изучение и проверка примеров (часть 1).

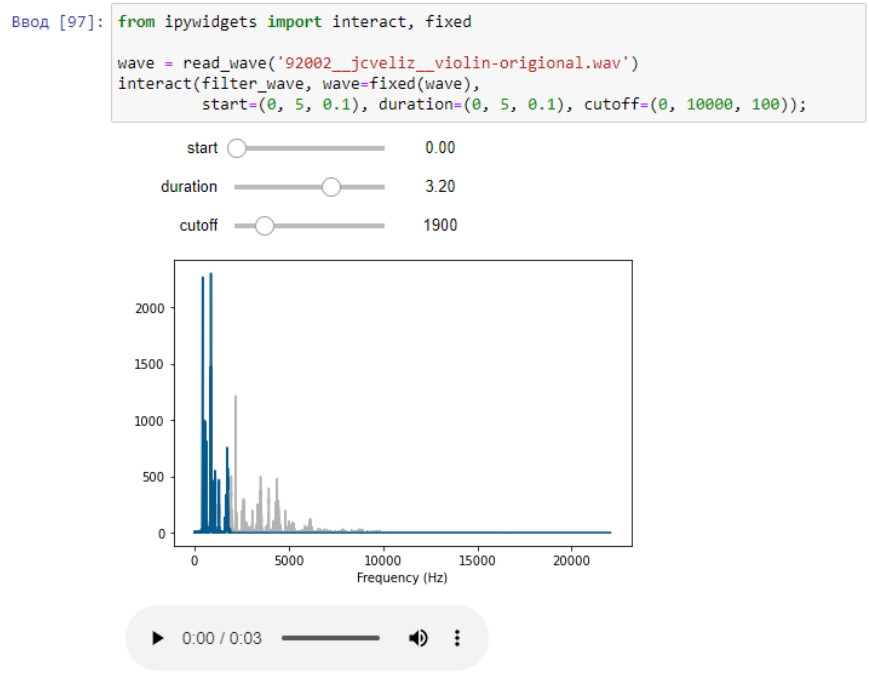

Рис. 2: Изучение и проверка примеров (часть 2).

Как мы видим из Рис.1 и Рис.2, все примеры были изучены и запущены.

2 Упражнение 1.2

Во втором упражнении происходит работа с образцом звука. Для этого необходимо скачать с freesound.org образец звука, имеющий четко выраженную высоту. После этого необходимо выделить полусекундный сегмент с постоянной высотой, вычислить и вывести его спектр.

Итак, был скачан этот образец звука с записью школьного звонка. Теперь считаем этот файл и выведем его график. Для чтения будем использовать `read_wave()`, а для построения графика - `plot()`.

```
from thinkdsp import read_wave

wave = read_wave('resources/Sounds/task2_school_rings.wav')
wave.normalize()
wave.make_audio()
wave.plot()

```

Листинг 1: Чтение файла звука и вывод графика.

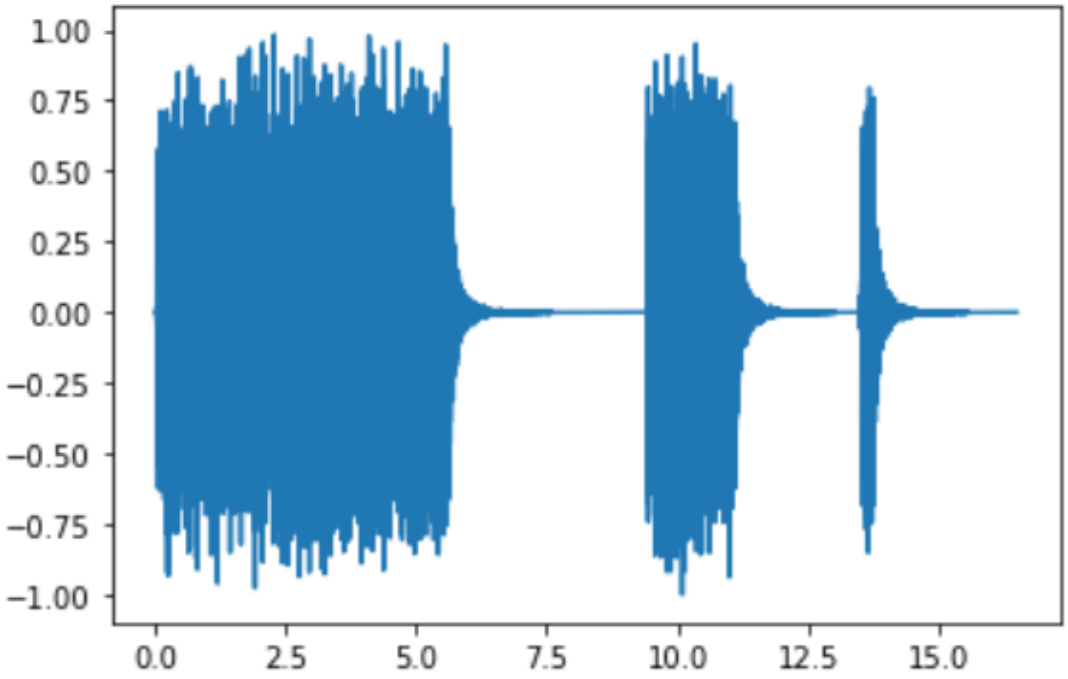

Рис. 3: График образца звука.

Теперь выделим сегмент с постоянной высотой и выведем его график. Для выделения сегмента был использован метод `segment()`.

```
segment = wave.segment(start=5.0, duration=0.3)
segment.make_audio()
segment.plot()

```

Листинг 2: Выделение сегмента и вывод его графика.

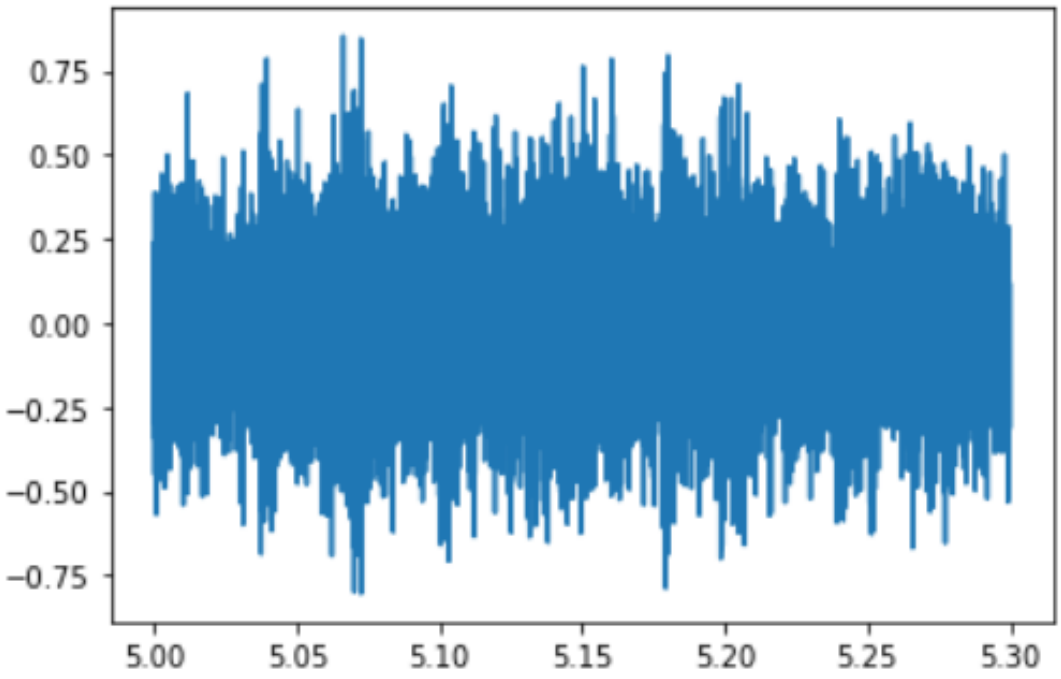

Рис. 4: График сегмента звука.

С помощью `make_spectrum()` создадим спектр нашего сегмента.

```
spectrum = segment.make_spectrum()
spectrum.plot(7000)

```

Листинг 3: Создание спектра сегмента.

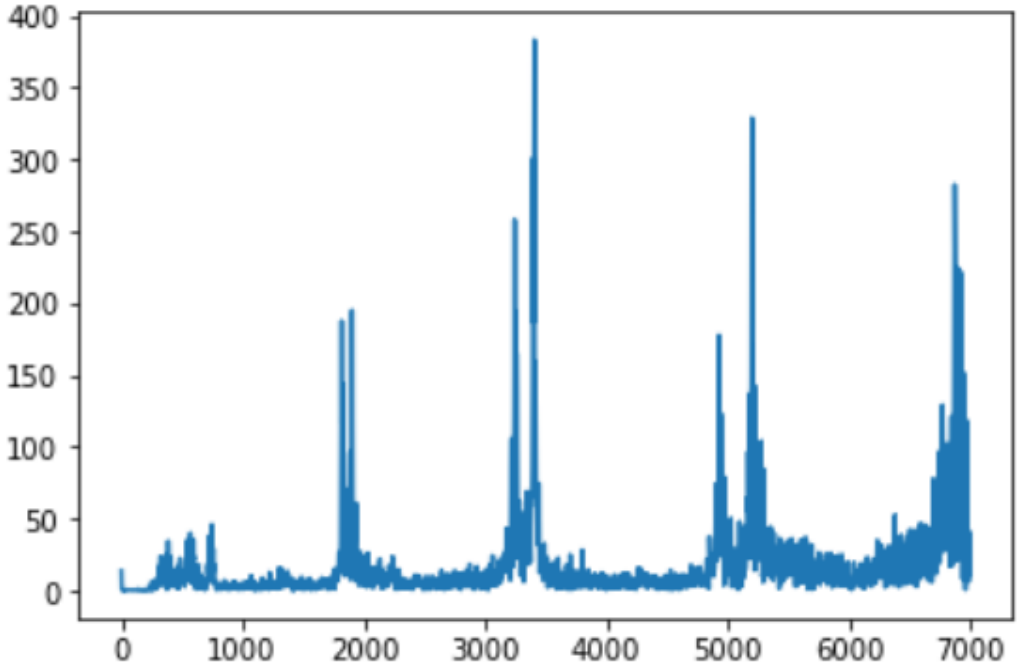

Рис. 5: Спектр сегмента записи.

Рассмотрим основную и доминирующую частоты, выведем спектр.

```
spectrum.plot(3500)

```

Листинг 4: Создание спектра сегмента.

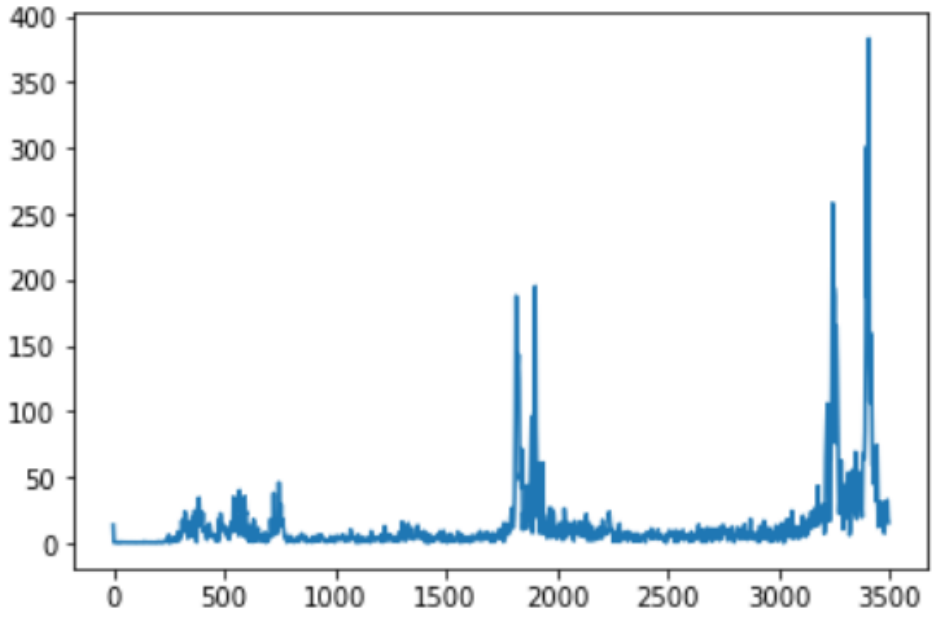

Рис. 6: Спектр сегмента записи (основная и доминирующая частоты).

Тембр - это одна из характеристик звука, его ”окраска”. Тембр зависит от частот в спектре и их интенсивностей. Также гармоники образуют тембр.

Теперь с помощью `low_pass()`, `high_pass()` и `band_stop()` произведём различную фильтрацию и преобразуем спектры обратно в сигнал.

```
# Filter out the high frequencies
spectrum.low_pass(2000)
spectrum.plot(7000)
spectrum.make_wave().make_audio()

# Filter out the low frequencies
spectrum = segment.make_spectrum()
spectrum.high_pass(2000)
spectrum.plot(7000)
spectrum.make_wave().make_audio()

# Selective filtering
spectrum = segment.make_spectrum()
spectrum.band_stop(1500, 6000)
spectrum.plot(7000)
spectrum.make_wave().make_audio()

```

Листинг 5: Фильтрация гармоник.

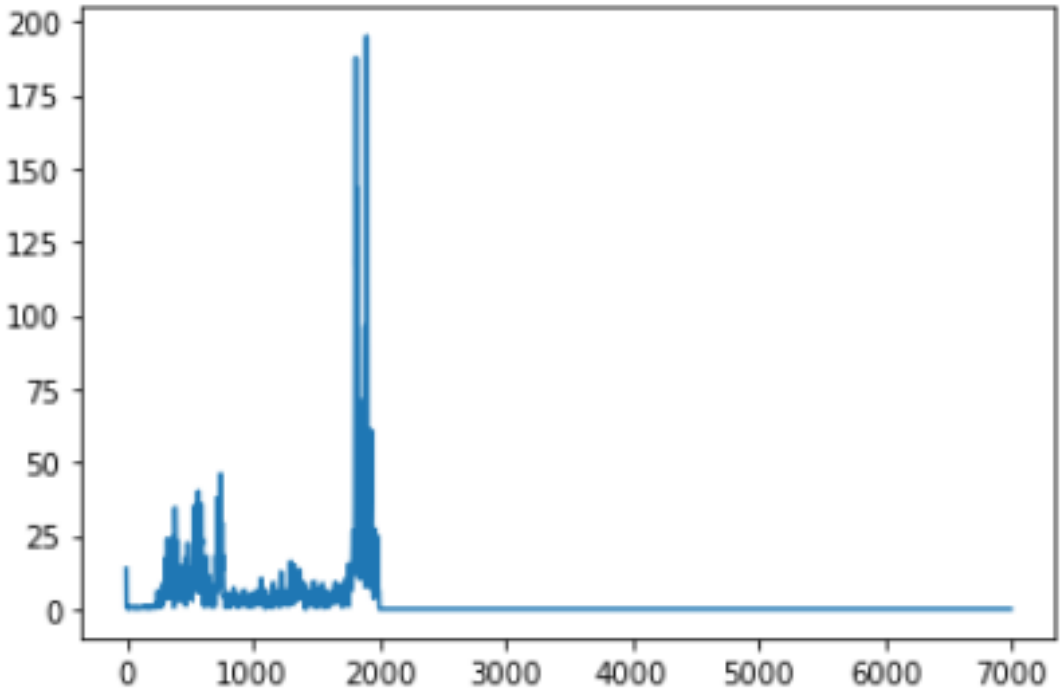

Рис. 7: Спектр, отфильтрованный по высоким частотам.

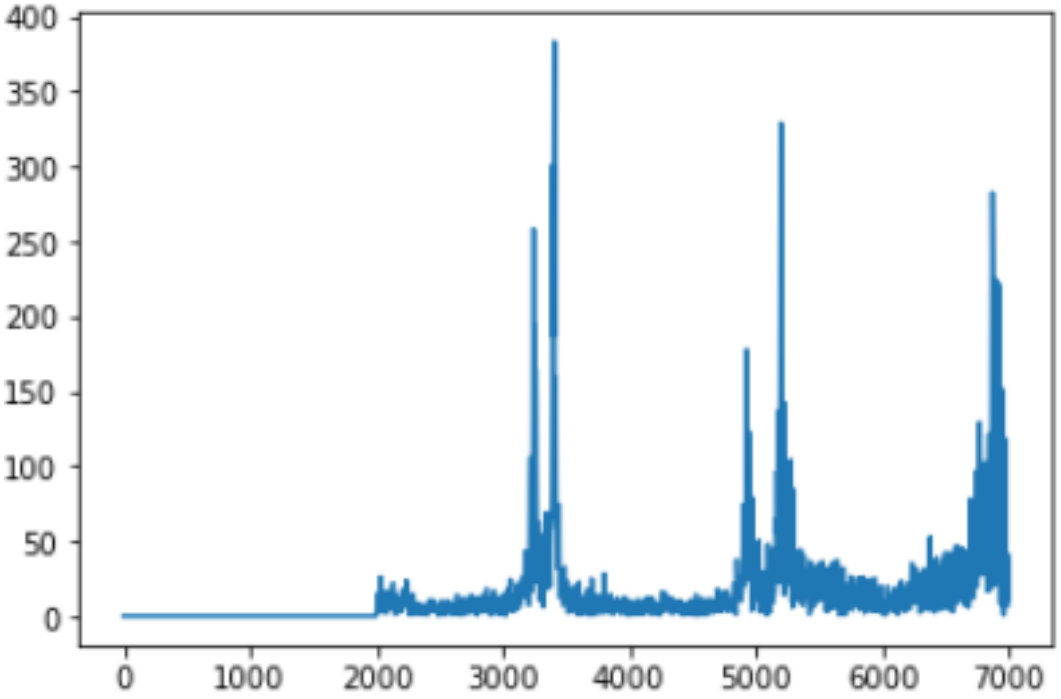

Рис. 8: Спектр, отфильтрованный по низким частотам.

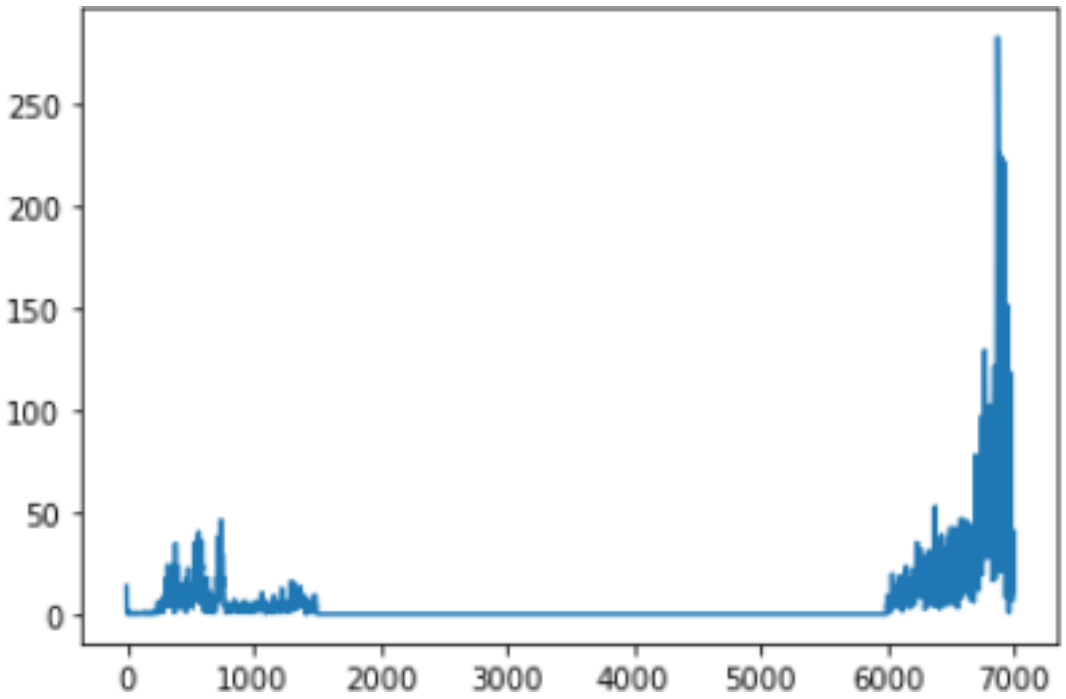

Рис. 9: Спектр, отфильтрованный выборочно.

Если же прослушать все полученные сигналы, то можно заметить, как отсутствие тех или иных частот влияет на звук. Например, отфильтрованный по высоким частотам сигнал вообще не похож на звук школьного звонка. Без низких же частот звук становится более сухим и тонким. При выборочной фильтрации звук вообще становится больше похож на стрекот.

В ходе выполнения данного упражнения была проведена работа с образцом звука. Из него был выделен сегмент с постоянной частотой. Был построен его спектр и произведена фильтрация гармоник. Полученные в ходе фильтрации спектры были преобразованы обратно в сигналы и прослушаны.

3 Упражнение 1.3

В этом упражнении нам предстоит создать составной сигнал из объектов `SinSignal` и `CosSignal`. Затем необходимо обработать сигнал для получения `wave` и прослушать его, после чего вычислить и построить `Spectrum`.

Итак, начнём с создания нескольких экземпляров `SinSignal` и `CosSignal`, после чего создадим результирующий сигнал и выведем его на экран. Тут же преобразуем его к `wave` и прослушаем.

```
from thinkdsp import SinSignal, CosSignal

sin1 = SinSignal(freq=150, amp=0.5)
sin2 = SinSignal(freq=800, amp=0.2)
cos1 = CosSignal(freq=500, amp=0.9)
cos2 = CosSignal(freq=1000, amp=1.0)

signal = sin1 + cos1 + sin2 + cos2
signal.plot()

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

```

Листинг 6: Создание сигнала, вывод графика и создание аудио объекта.

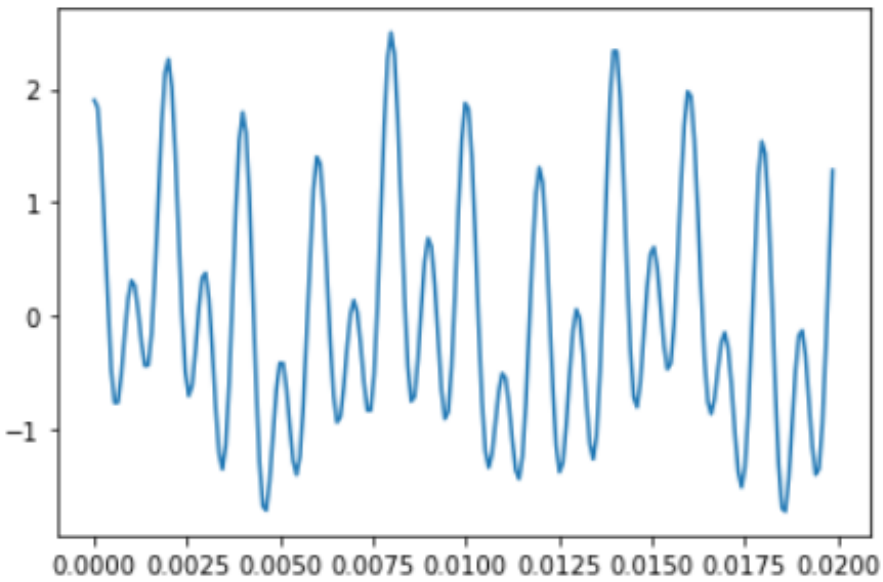

Рис. 10: Составной сигнал.

Получившийся составной сигнал похож на телефонный гудок. Теперь построим спектр и отобразим его.

```
spectrum = wave.make_spectrum()
spectrum.plot(2000)

```

Листинг 7: Создание спектра.

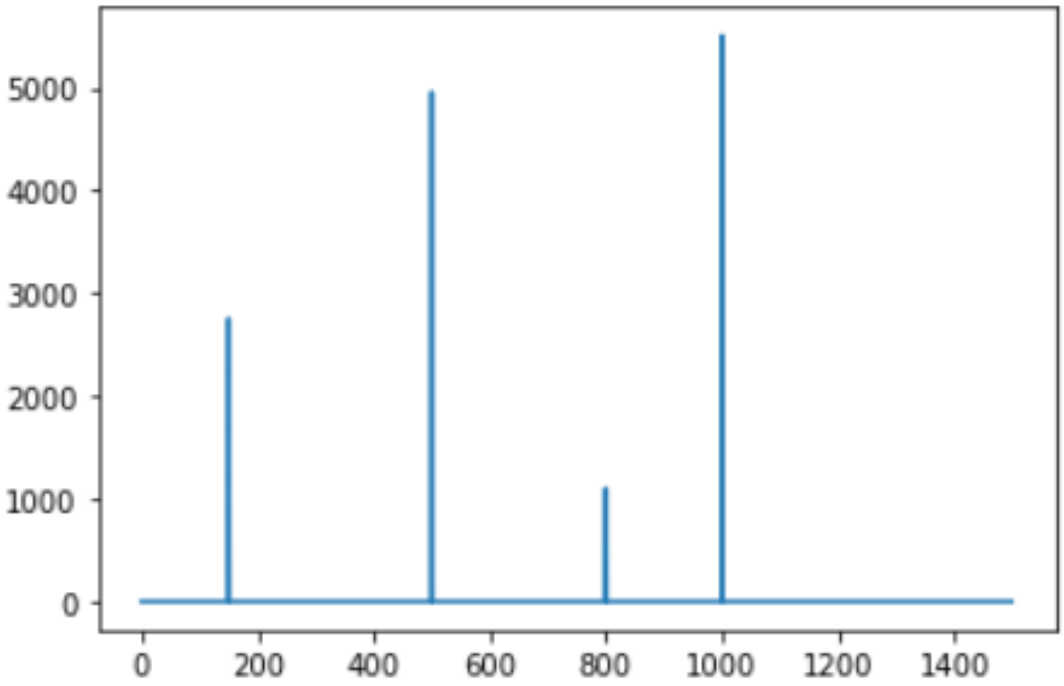

Рис. 11: Спектр созданного составного сигнала.

Все компоненты, из которых состоит полученный сигнал, кратны 50 Hz, а потому ”сливаются”в единый гудок.

Теперь проверим, что произойдёт, если добавить частотные компоненты, не кратные уже добавленным. Для этого прибавим к нашему сигналу ещё по одному `SinSignal` и `CosSignal` и прослушаем.

```
sin3 = SinSignal(freq=364, amp=0.8)
cos3 = CosSignal(freq=728, amp=0.7)

signal2 = signal + sin3 + cos3
signal2.make_wave(duration=1).make_audio()

```

Листинг 8: Добавление некратных компонент.

В полученном сигнале отчётливо слышны добавленные компоненты, потому что они не кратны 50 Hz. Мы слышим их как отдельные высоты, а не как ”дополнение”гудка. Кроме того, полученный звук стал менее приятным на слух.

В ходе выполнения данного упражнения был создан собственный составной сигнал из объектов `SinSignal` и `CosSignal`. Он был преобразован в `wave` и прослушан. Затем к нему были добавлены некратные компоненты, после чего сигнал снова был прослушан. Был сделан вывод, что компоненты, не кратные уже существующим, резко выделяются на фоне остальных.

4 Упражнение 1.4

Теперь нам необходимо написать функцию `stretch`, в которую передаётся `wave` и коэффициент изменения. Функция должна ускорять или замедлять сигнал изменением `ts` и `framerate`.

Итак, напишем саму функцию.

```
    def stretch(wave_stretch, factor):
        wave_stretch.ts /= factor
        wave_stretch.framerate *= factor

```

Листинг 9: Описание функции `stretch`.

При подаче `factor` меньше `1.0` сигнал будет замедляться, если же `factor` больше `1.0` - ускоряться.

Теперь необходимо проверить работу нашей функции. Для тестирования будем использовать звуковой файл, который использовался в Упражнении 1.2. Для начала создадим два одинаковых `wave` и прослушаем звук (для удобства сравнения).

```
        wave1 = read_wave('resources/Sounds/task2_school_rings.wav')
        wave2 = wave1
        wave1.make_audio()

```

Листинг 10: Считывание звукового файла.

Теперь замедлим один сигнал и ускорим второй, прослушаем результат и выведем графики сигналов.

```
        # Deceleration
        stretch(wave1, 0.3)
        wave1.make_audio()
        wave1.plot()
        # Acceleration
        stretch(wave2, 5.0)
        wave2.make_audio()
        wave2.plot()

```

Листинг 11: Редактирование сигналов с помощью `factor`.

Замедленный звук стал более низким и менее резким. Ускоренный же звук наоборот стал более звонким, высоким и резким. При сравнении полученных графиков с графиком нетронутого звука (Рис.3) можно заметить, что ускоренный сигнал сжался, а замеленный - растянулся.

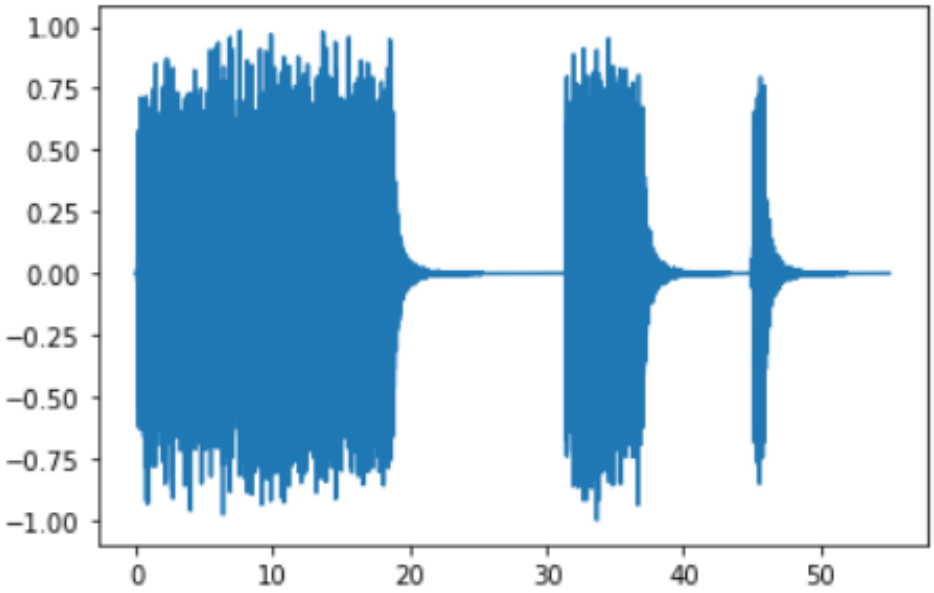

Рис. 12: Замедленный сигнал.

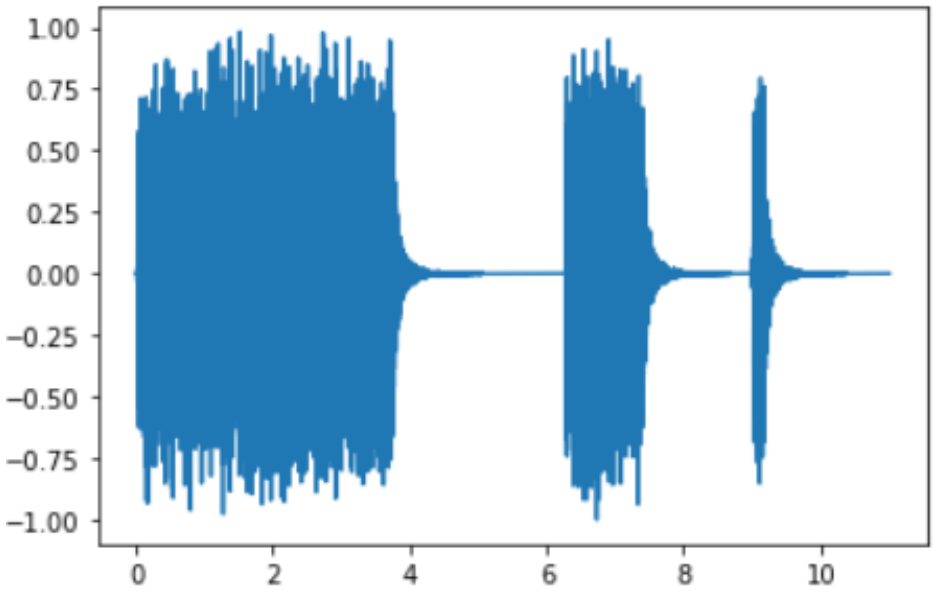

Рис. 13: Ускоренный сигнал.

Исходя из результатов тестирования, можно сделать вывод, что написанная функция `stretch` работает корректно.

В ходе выполнения данного упражнения была написана функция `stretch`. Она принимает на вход `wave` и коэффициент изменения. Функция ускоряет или замедляет сигнал (в зависимости от поданного `factor`) изменением `ts` и `framerate`. Функция была протестирована, и был сделан вывод, что она работает корректно.

5 Выводы

В ходе выполнения данной лабораторной работы были изучены объекты Signal, Wave и Spectrum. Также были получены навыки по работе с сигналами, их обработке и фильтрации, созданию спектров и собственных составных сигналов. Кроме того, были получены знания о том, как именно небольшое конкретное изменение сигнала может повлиять на звук.