Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа №1

Сигналы и звуки

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

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург

2021

Содержание

Список иллюстраций

Листинги

1 Часть №1: Проверка

В первом пункте данной лабораторной работы нам необходимо проверить, что все листинги работают корректно. Для этого просто пройдемся по всем примерам и запустим их.

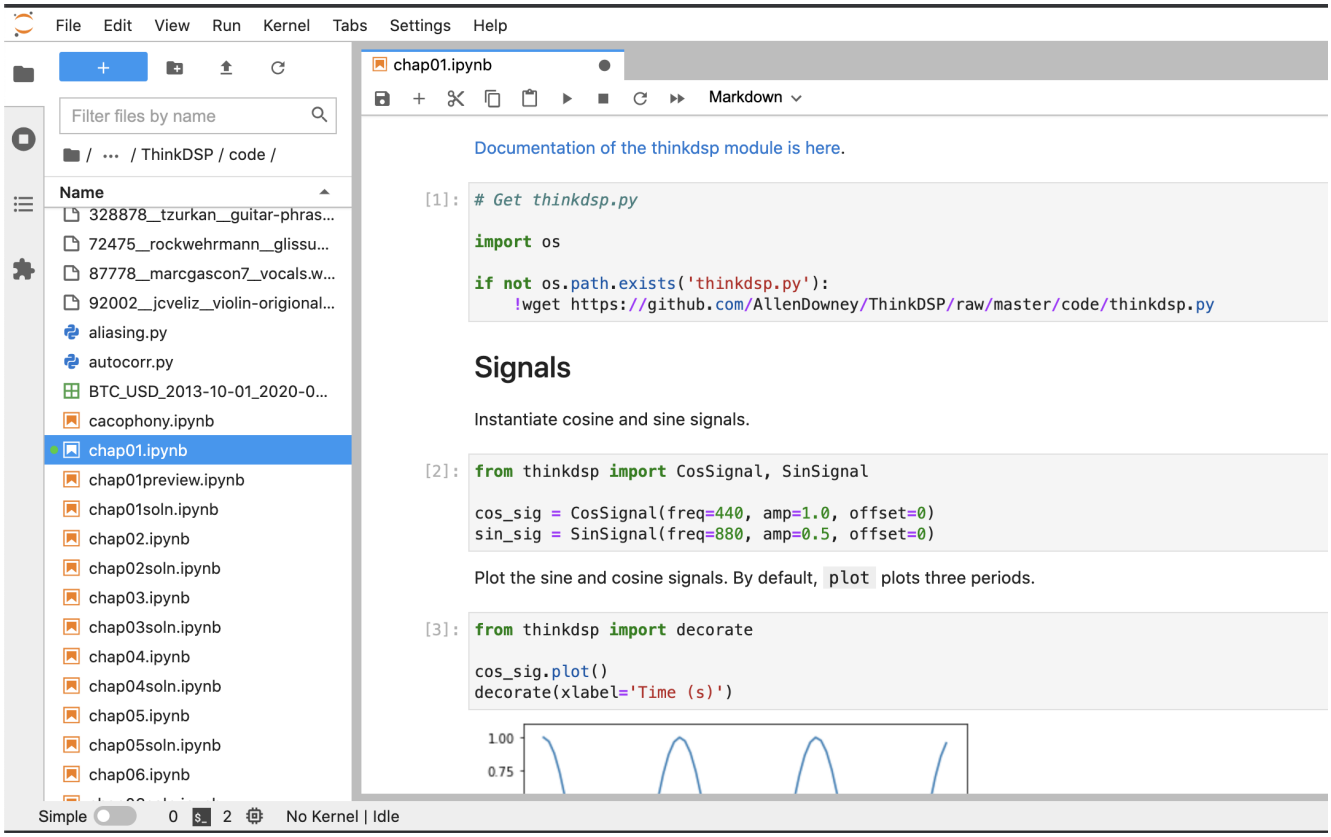

Рис. 1: Проверка работоспособности

После прохода и запуска каждого листинга становится ясно, что все работает как ожидалось и можно переходить к следующему пункту лабораторной работы

2 Часть №2: Обработка

Во втором пункте первой лабораторной работы необходимо сначала скачать с сайта любой образец звука, после чего выделить в нем полсуекундный фрагмент с постоянной высотой. Необходимо вычислить и рассчитать спектр. После этого необходимо произвести фильтрацию гармоник и преобразовать спектры обратно в сигнал.

Перейдем к выполнению лабораторной работы. Для начала подключим к нашему проекту `thinkdsp.py`, чтобы в дальнейшем работать с ней:

```
1    import os
2    if not os.path.exists('thinkdsp.py'):
3        !wget https://github.com/AllenDowney/ThinkDSP/raw/master/code/thinkdsp.
   py
```

Листинг 1: Подключение `thinkdsp.py`

Теперь подключим основные библиотеки, которые нам понадобятся:

```
1    from thinkdsp import read_wave
2    from thinkdsp import decorate
```

Листинг 2: Подключение библиотек

После этого прочтем скаченную нами ранее мелодию:

```
1    wave = read_wave('Sounds/564668__dingeaux__shwing.wav')
2    wave.normalize()
3    wave.make_audio()
```

Листинг 3: Прочтение скаченной мелодии

После прочтения мелодии выполним `wave.plot()`: и проверим результаты вывода

```
1    wave.plot()
2    decorate(xlabel='Time (s)')
```

Листинг 4: Просмотр `wave.plot()`

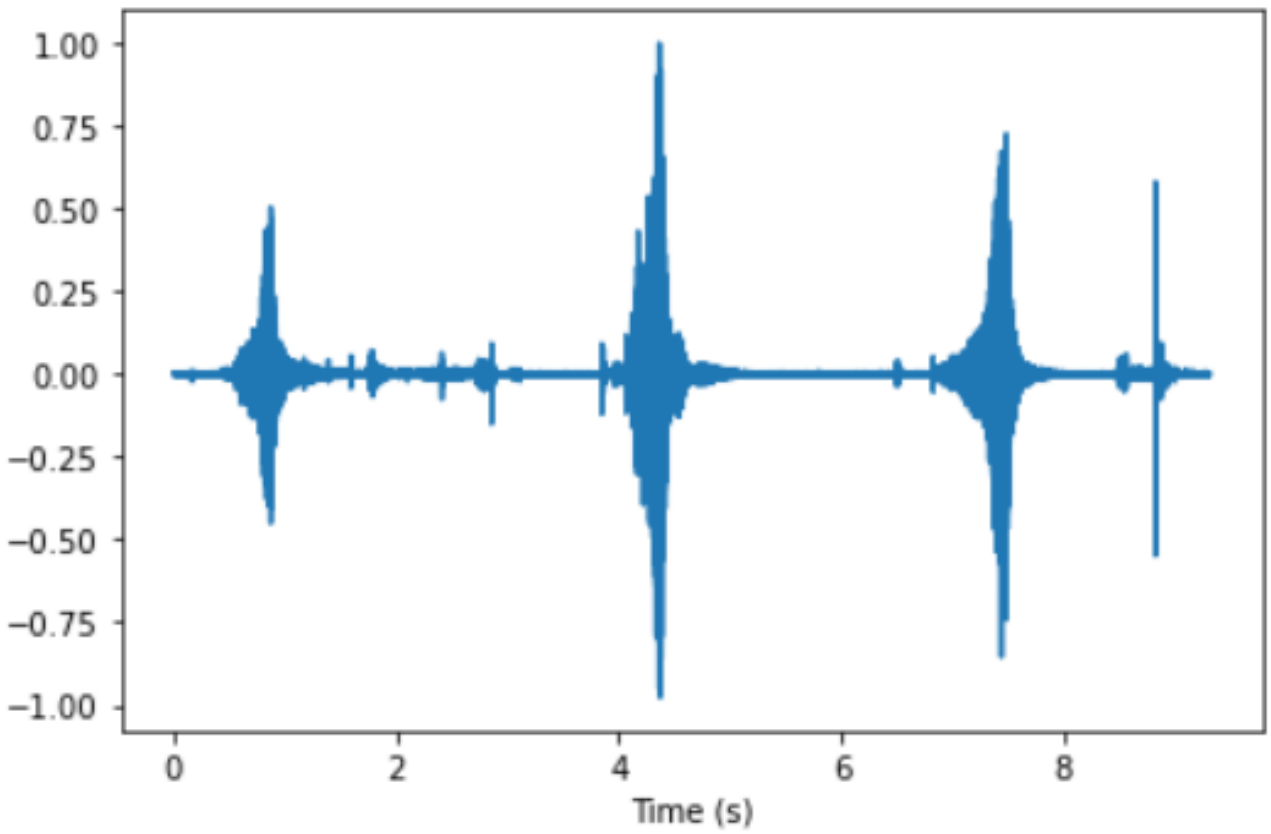

Рис. 2: Результат вывода `wave.plot()`

Далее нам необходимо выбрать сегмент из нашей мелодии. Был выбран сегмен с 3.9 секунды и длительностью 0.5 секунды:

```
segment = wave.segment(start=3.9, duration=0.5)
segment.make_audio()
```

Листинг 5: Выбор сегмента

Теперь выведем `segment.plot()` на экран:

```
segment.plot()
decorate(xlabel='Time (s)')
```

Листинг 6: Вывод `segment.plot()`

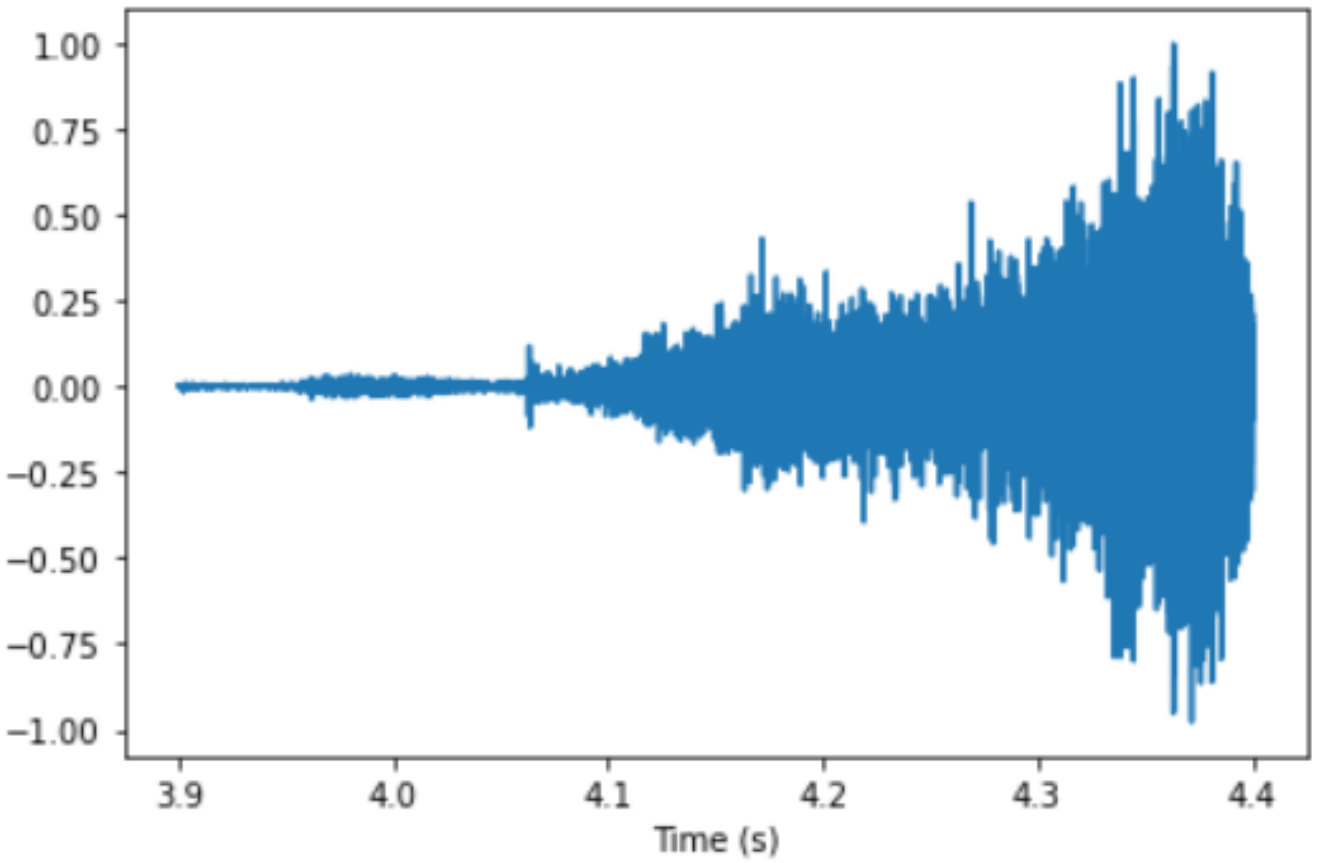

Рис. 3: Результат вывода `segment.plot()`

После этого посмотрим как выглядит наш спектр:

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

Листинг 7: Изображение спектра

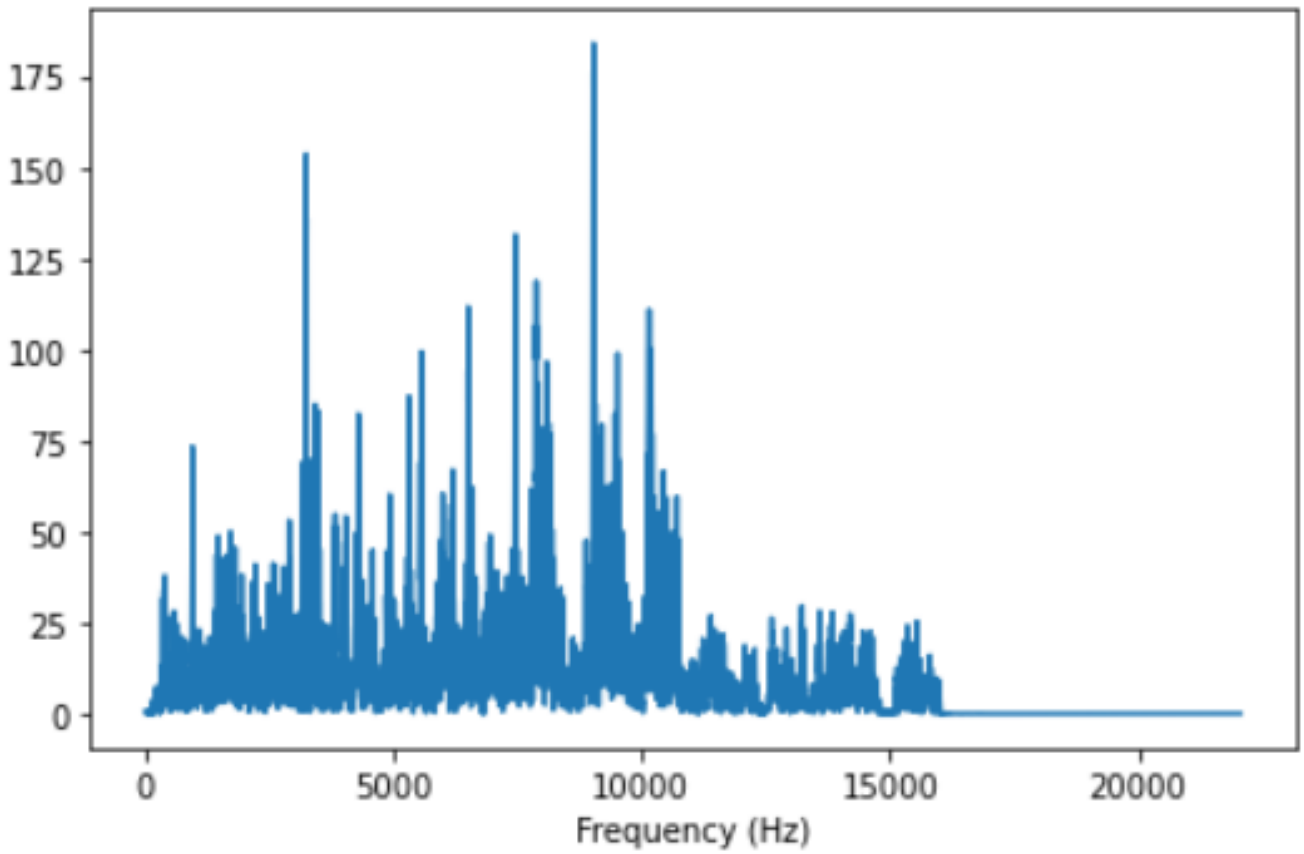

Рис. 4: Изображение спектра

Приблизим наш спектр, чтобы увидеть его более подробно:

```
spectrum.plot(high=16000)
decorate(xlabel='Frequency (Hz)')
```

Листинг 8: Приближенное изображение спектра

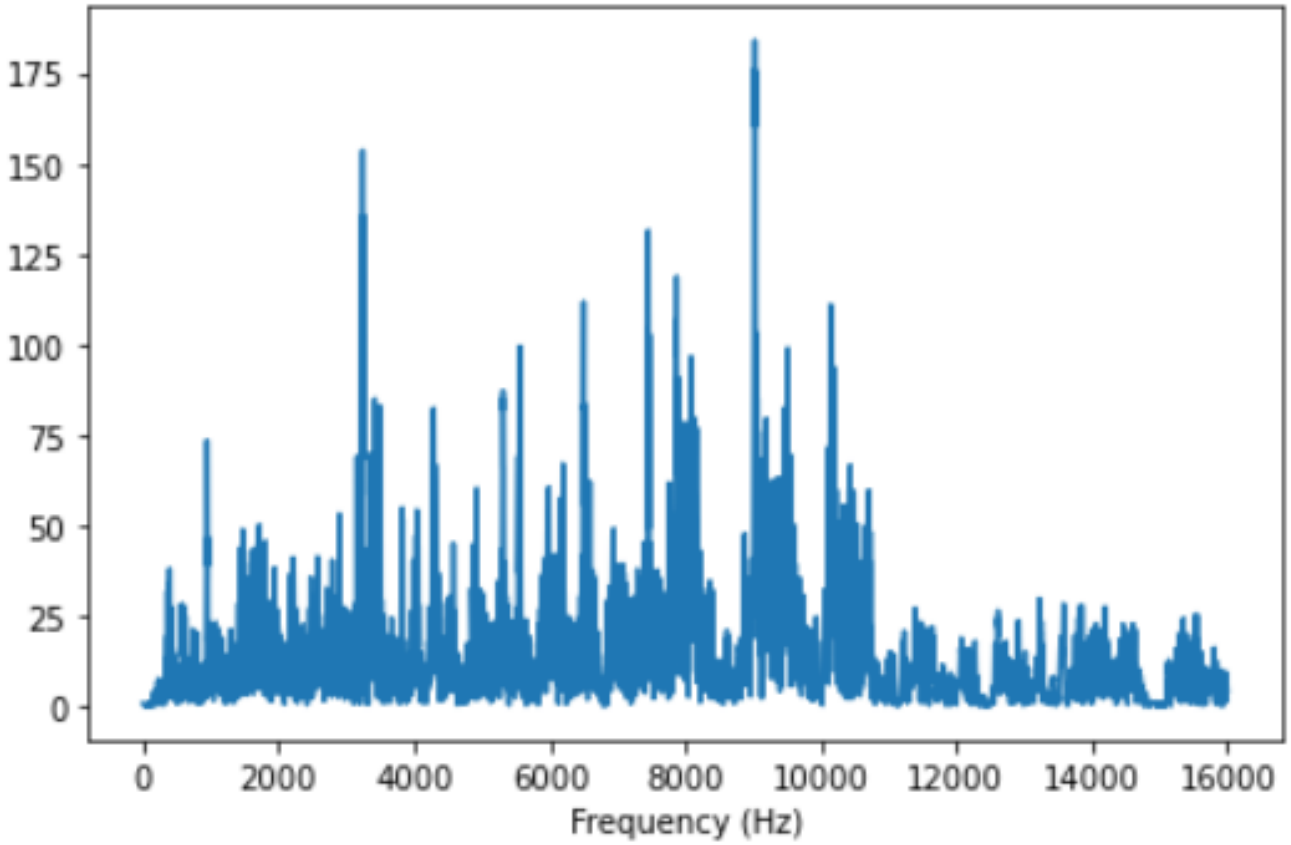

Рис. 5: Приближенное изображение спектра

Далее, полученный нам испектр необходимо отфильтровать:

```
filtered = spectrum.make_wave()
filtered.normalize()
filtered.plot()
decorate(xlabel='Time (s)')
```

Листинг 9: Фильтрация спектра

Для наглядности мы вывели полученный отфильтрованный спектр:

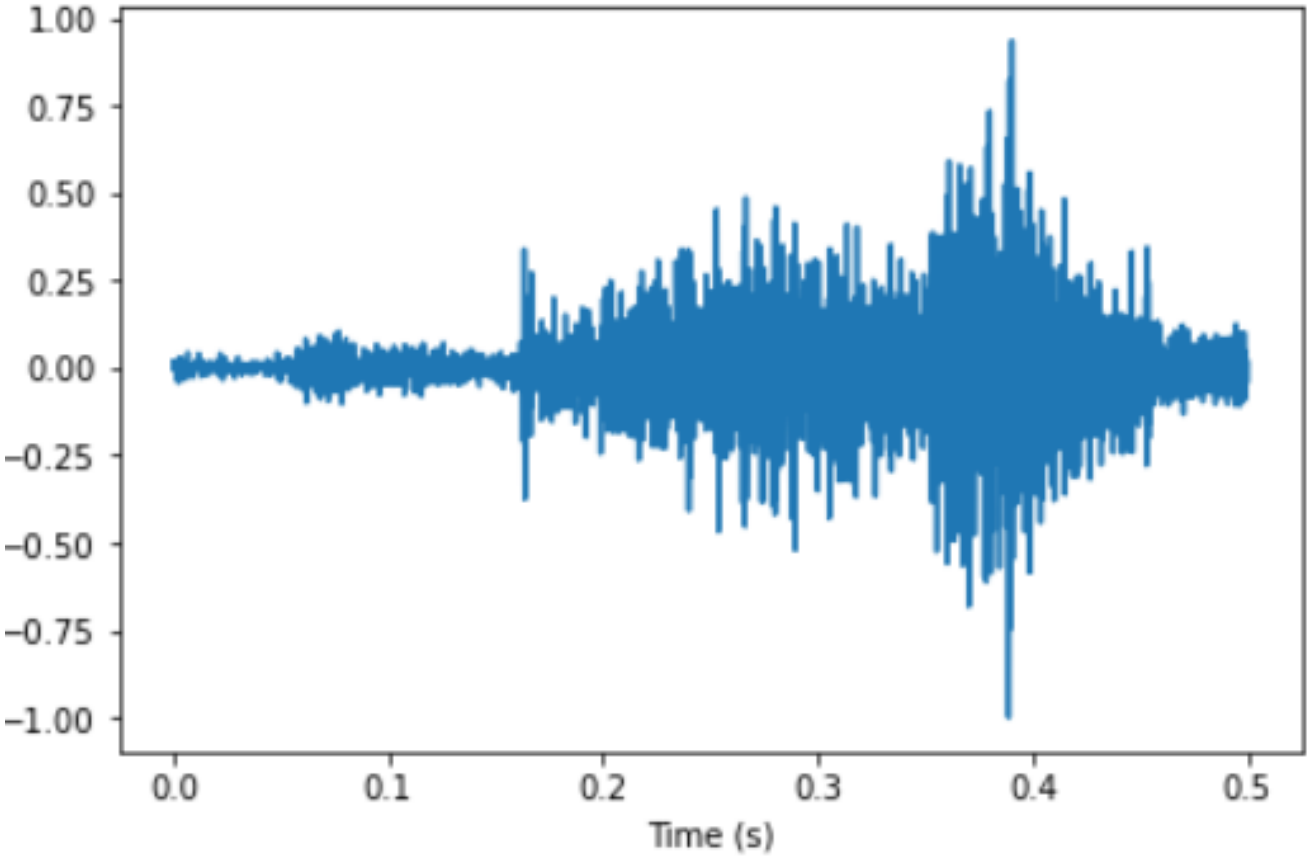

Рис. 6: Отфильтрованный спектр

После всего этого переводим полученный спектр обратно в аудио:

```
В [9]: spectrum.make_wave().make_audio()
Out[9]:
```

Рис. 7: Перевод спектра в аудио

По результатам выполнеиня данного пункта можно сделать вывод, что полученный звук похож на оригинальный, но более приглушен, будто источник теперь находится в другой комнате

3 Часть №3: Комбинирование

В третьем пункте первой лабораторной работы нам необходимо создать сложный сигнал из объектов `SinSignal` и `CosSignal` суммируя их. Также следует обработать полученный сигнал для получения волный, прослушать его, вычислить для него `Spectrum` и после вывести на экран. Для выполнения задания я взял синусоидальный канал с частотой `755 Hz` и косинусоидальный канал с частотой `400 Hz`:

```
from thinkdsp import CosSignal, SinSignal

cos_sig = CosSignal(freq=400, amp=1.0)
sin_sig = SinSignal(freq=755, amp=0.5)

cos_sig.plot()
decorate(xlabel='Time (s)')

sin_sig.plot()
decorate(xlabel='Time (s)')
```

Листинг 10: Создание каналов

В результате получаем такой вывод на экран:

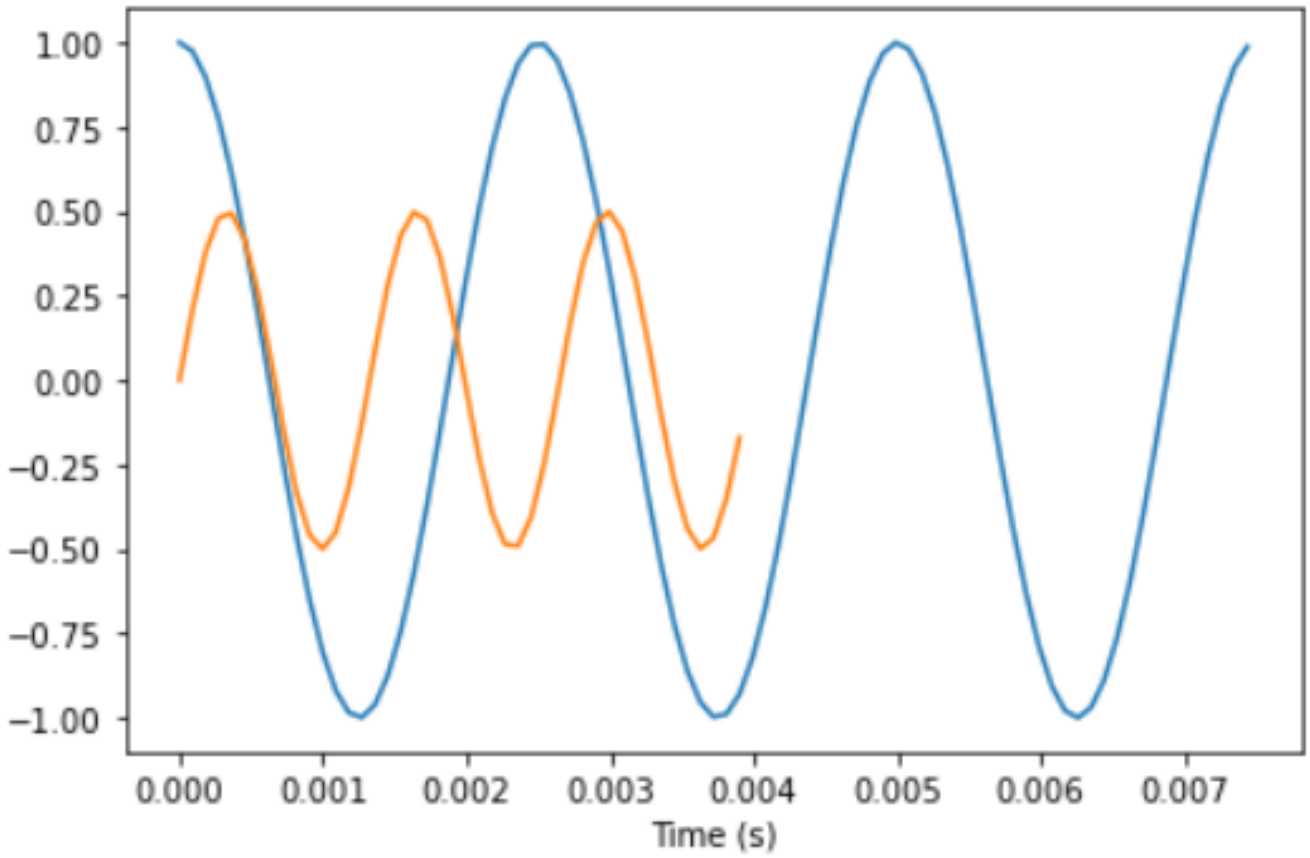

Рис. 8: Вывод спектров

Теперь просуммируем эти два канала и выведем полученный суммирующий сигнал:

```
signal = cos_sig + sin_sig
signal.plot()
decorate(xlabel='Time (s)')
```

Листинг 11: Суммирование каналов

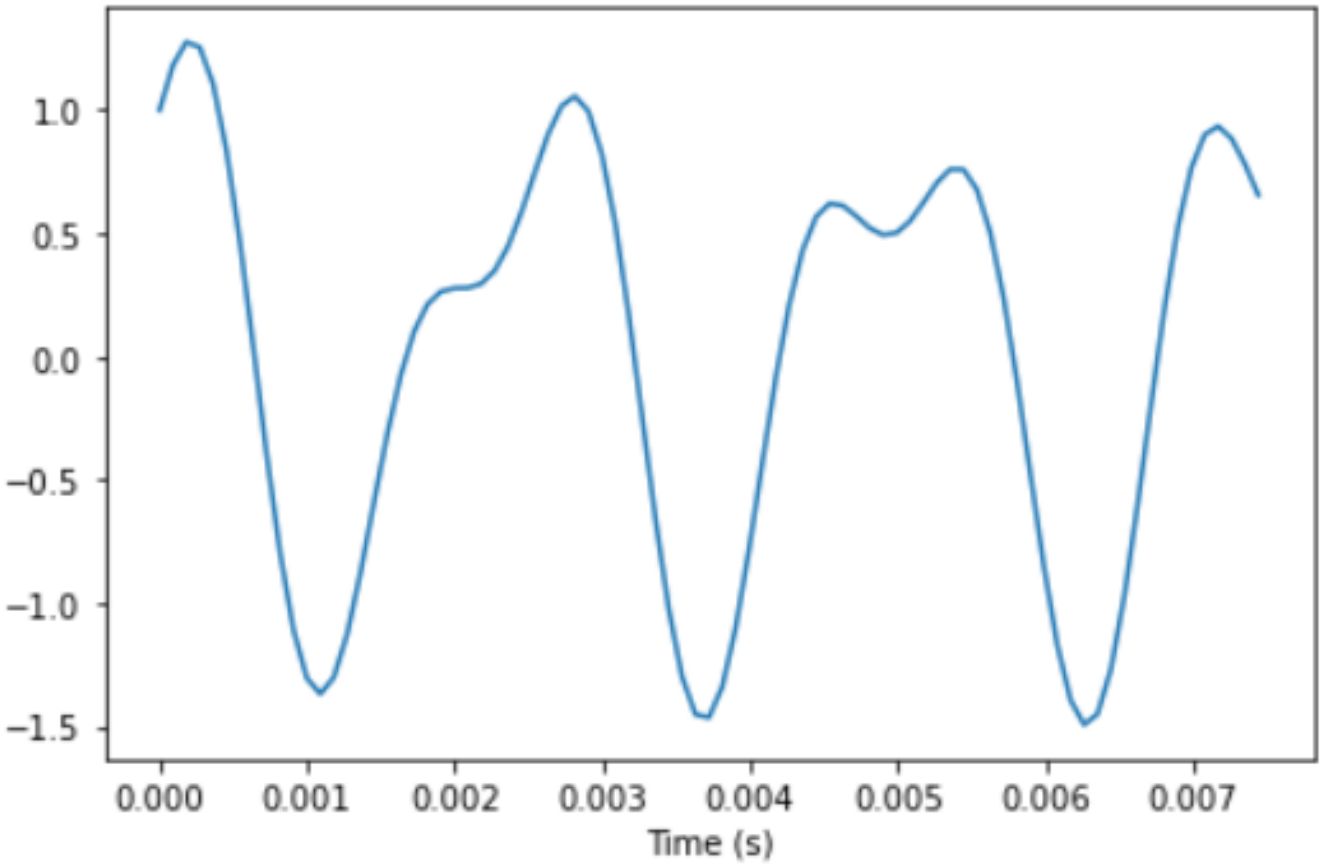

Рис. 9: Суммирование каналов

После этого получим аудио из суммирующего канала

```
B [14]: wave1 = signal.make_wave(duration=0.5, start=0, framerate=11025)
        wave1.make_audio()
Out[14]:
```

Рис. 10: Получение аудио из сумирующего канала

Наконец, получим спектр из полученного сигнала:

```
spectrum = wave1.make_spectrum()
```

```
    spectrum.plot(high=1000)
    decorate(xlabel='Frequency (Hz)')
```

Листинг 12: Получение спектра

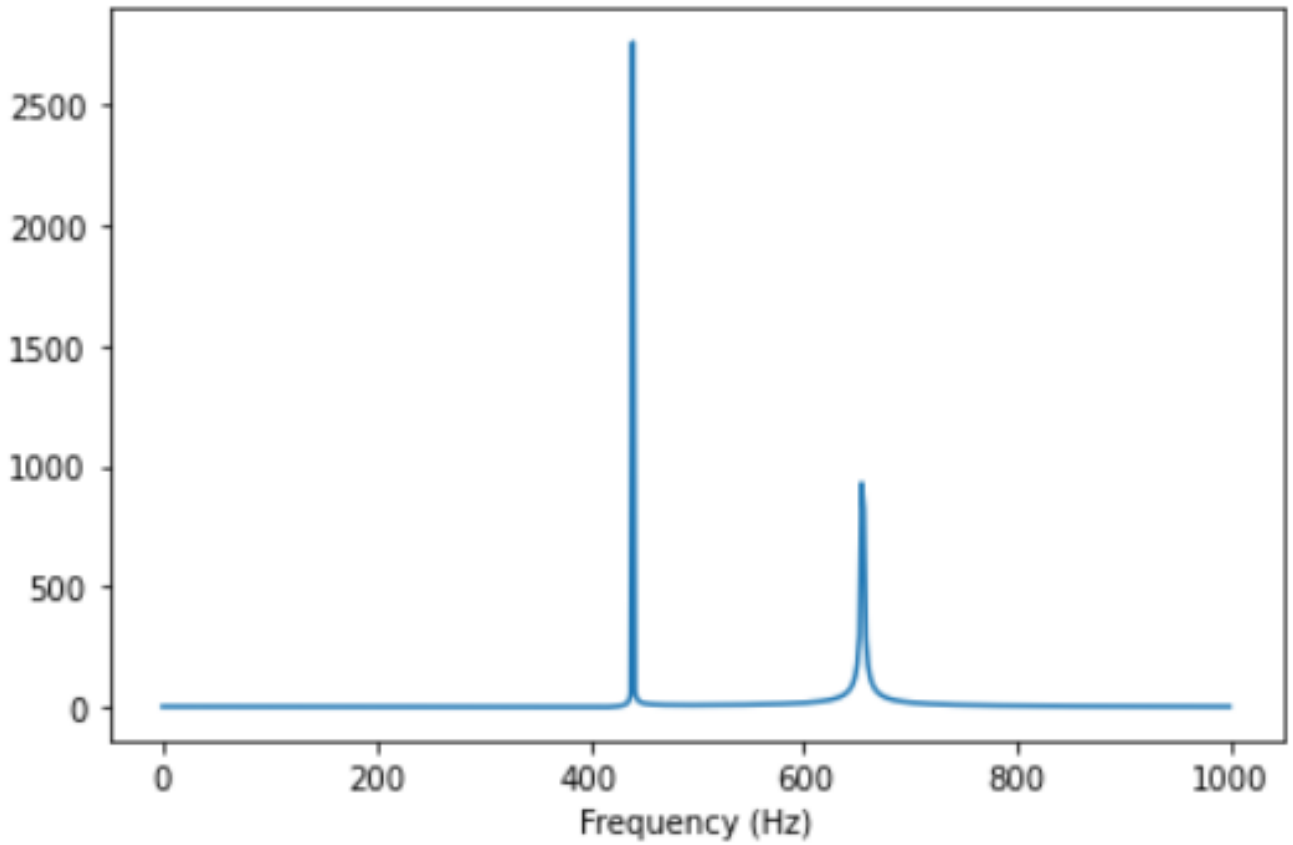

Рис. 11: Полученный спектр

Добавим к полученному сигналу синусоидальный сигнал с частотой 450 Hz иперевдем в аудио:

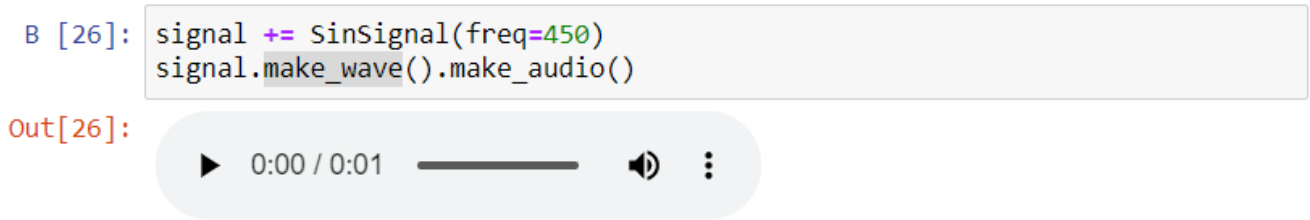

Рис. 12: Добавление к сигналу и перевод в аудио

После всего этого нам остается только получить спектр из сигнала и вывести все на экран:

```
    wave2 = signal.make_wave()
    wave2.apodize
    spectrum2 = wave2.make_spectrum()
```

```
4     spectrum2.plot(high=1000)
5     decorate(xlabel='Frequency (Hz)')
```

Листинг 13: Получение спектра и вывод на экран

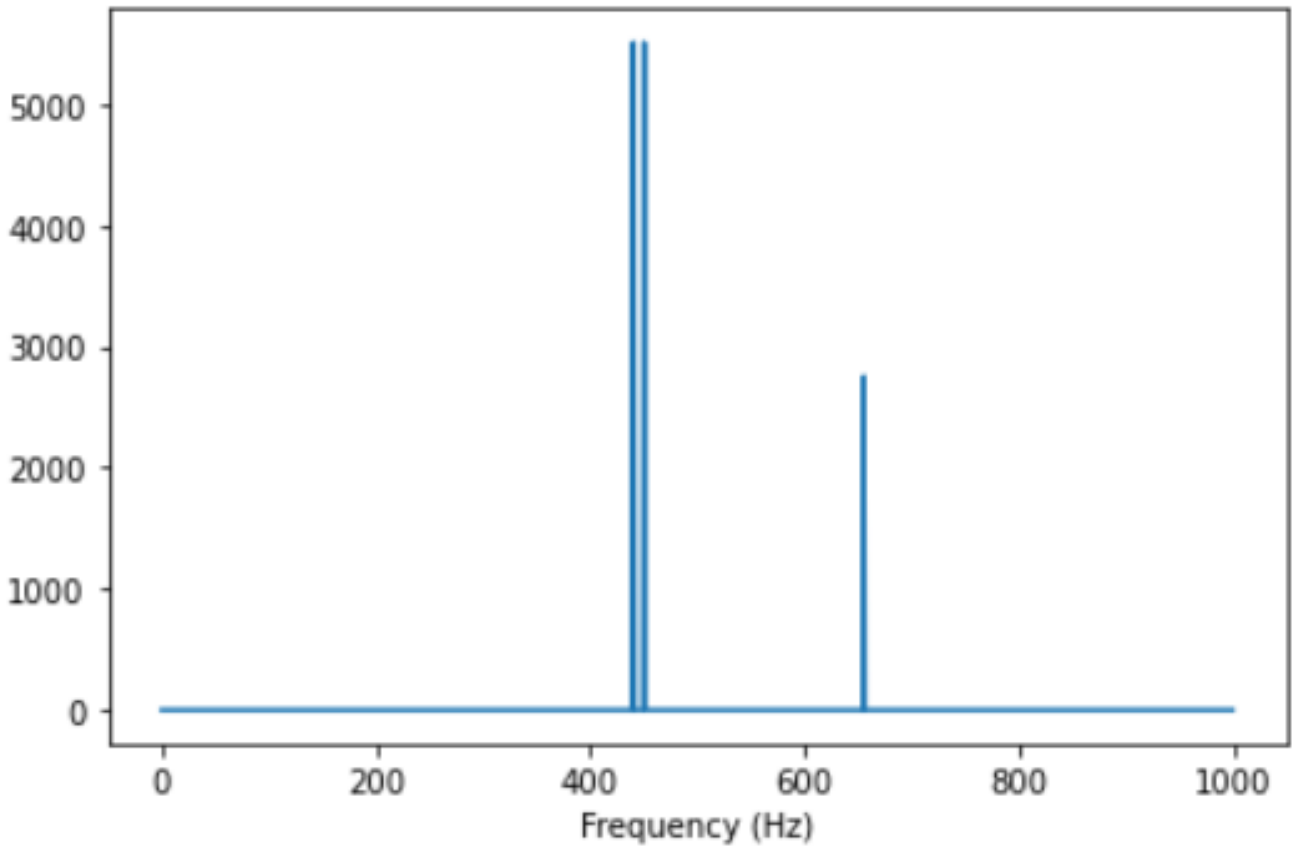

Рис. 13: Полученный спектр

По результатам выполнения данного пункта можно сделать вывод, что полученный в самом конце звук теперь звучит громче и появились колебания, теперь этот звук достаточно тяжело слушать.

4 Часть №4: Растяжение

В четвертом пункте первой лабораторной работы нам необходимо реализовать функцию `stretch`, которая принимает сигнал и коэффициент изменения, после чего в зависимости от коэффициента либо замедляет, либо ускоряет сигнал посредством изменения `ts` и `framerate`.

Перейдем к выполнению данного пункта. Сначала напишем саму функцию `stretch`:

```
1    def stretch(wave, factor):
2    wave.ts *= factor
3    wave.framerate /= factor
```

Листинг 14: Функция `stretch`

После этого вызовем эту функцию, подав нашу оригинальную аудиодорожку и коэффициент изменения равный 0.4, затем получим аудиодорожку:

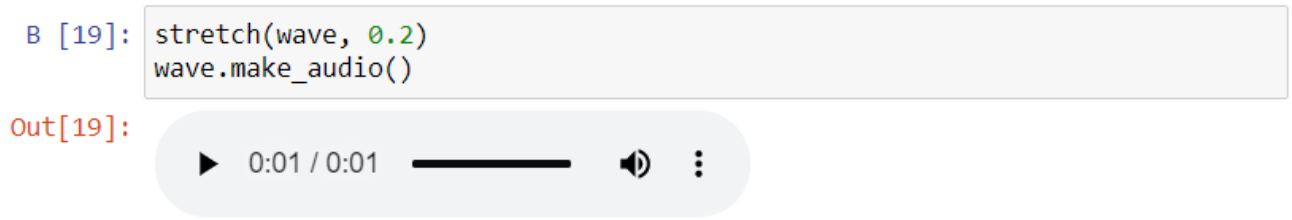

Рис. 14: Полученная в результате вызова функции `stretch` аудиодорожка

Затем вызовем отображение спектра полученной дорожки на экран:

```
1    wave.plot()
2    decorate(xlabel='Time (s)')
```

Листинг 15: Вывод на экран полученной после вызова функции `stretch` аудиодорожки

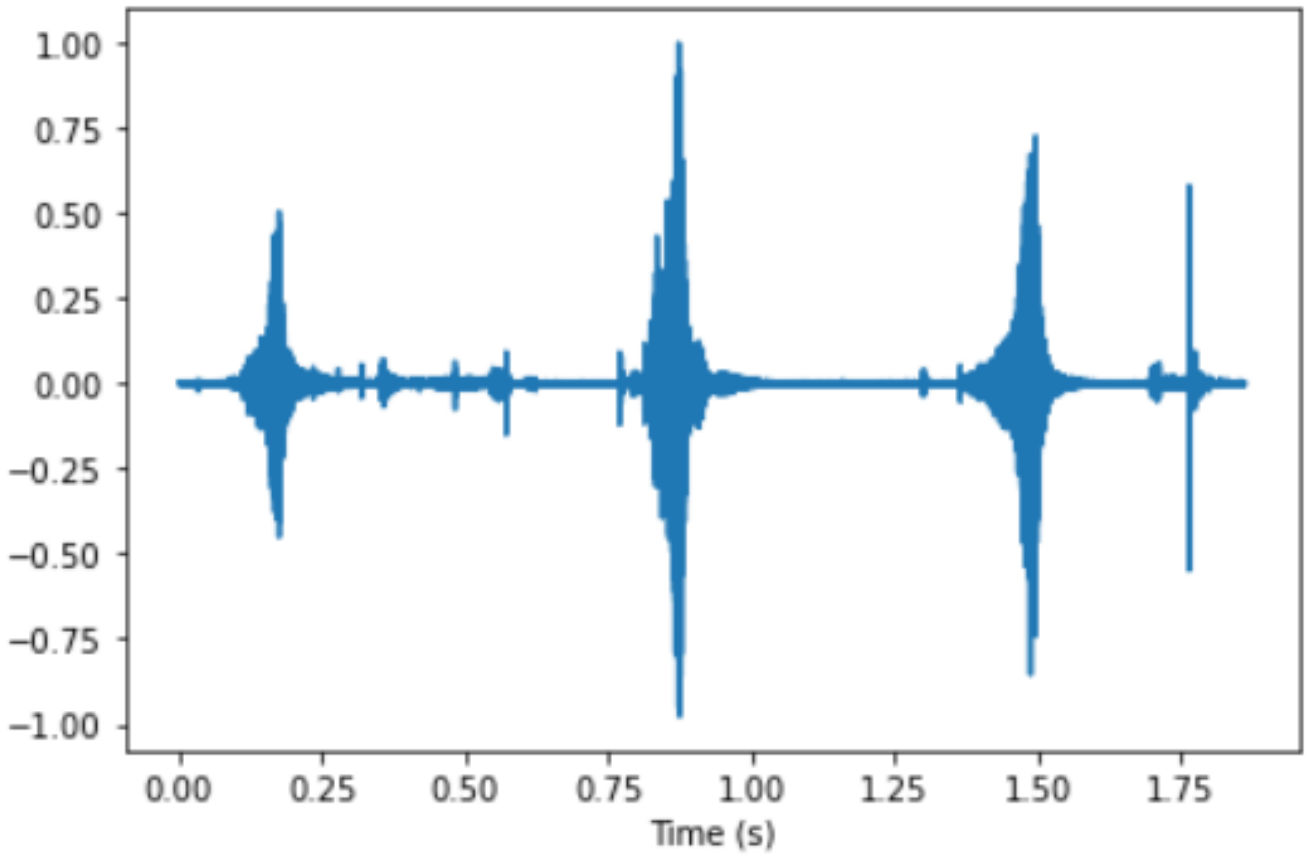

Рис. 15: Полученная в результате вызова функции `stretch` аудиодорожка

По результатам выполнения данного пункта можно сделать вывод, что полученная функция `stretch` действительно замедляет и ускоряет полученный сигнал.

5 Выводы

В ходе выполнения лабораторной работы мы изучили способы работы с сигналами и возможности их обработки, фильтрации и воспроизведения с помощью библиотеки Python. Кроме того мы получили сведения об основных понятиях и навыки работы с сигналами, а также реализовали и проверели функцию для преобразования полученного сигнала.