

Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Сигналы и звуки

Работу выполнил студент
3-го курса, группа 3530901/80201
Солянкин Илья Андреевич

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург 2021

Содержание

1	Часть №1: Проверка	5
2	Часть №2: Обработка	6
3	Часть №3: Комбинирование	12
4	Часть №4: Растяжение	16
5	Выводы	18

Список иллюстраций

1	Проверка, что все работает	5
2	Результат вывода <code>wave.plot()</code>	7
3	Результат вывода <code>segment.plot()</code>	8
4	Изображение спектра	9
5	Приближенное изображение спектра	10
6	Отфильтрованный спектр	11
7	Перевод спектра в аудио	11
8	Вывод спектров	12
9	Суммирование каналов	13
10	Получение аудио из суммирующего канала	13
11	Полученный спектр	14
12	Добавление к сигналу и перевод в аудио	14
13	Полученный спектр	15
14	Полученная в результате вызова функции <code>stretch</code> аудиодорожка .	16
15	Полученная в результате вызова функции <code>stretch</code> аудиодорожка .	17

Листинги

1	Подключение <code>thinkdsp.py</code>	6
2	Подключение библиотек	6
3	Прочтение скаченной мелодии	6
4	Просмотр <code>wave.plot()</code>	6
5	Выбор сегмента	7
6	Вывод <code>segment.plot()</code>	7
7	Изображение спектра	8
8	Приближенное изображение спектра	9
9	Фильтрация спектра	10
10	Создание каналов	12
11	Суммирование каналов	13
12	Получение спектра	14
13	Получение спектра и вывод на экран	14
14	Функция <code>stretch</code>	16
15	Вывод на экран полученной после вызова функции <code>stretch</code> ауди- одорожки	16

1 Часть №1: Проверка

В первом пункте первой лабораторной работы нам необходимо проверить, что все листинги могут запускаться. Для этого просто пройдемся по всем примерам и запустим их.

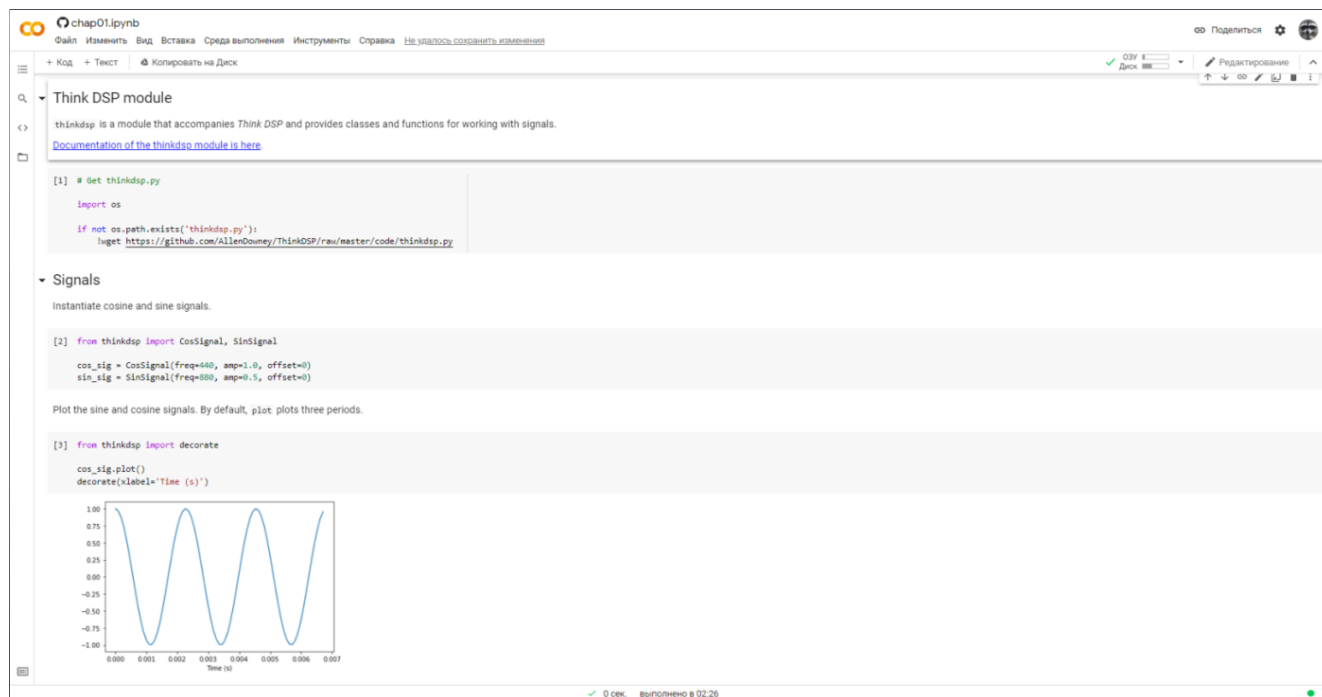


Рис. 1: Проверка, что все работает

После прохода и запуска каждого листинга стало понятно, что все работает как надо и теперь 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 полсекундный 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 посмотрим как выглядит `wave.plot()`:

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

Листинг 4: Просмотр `wave.plot()`

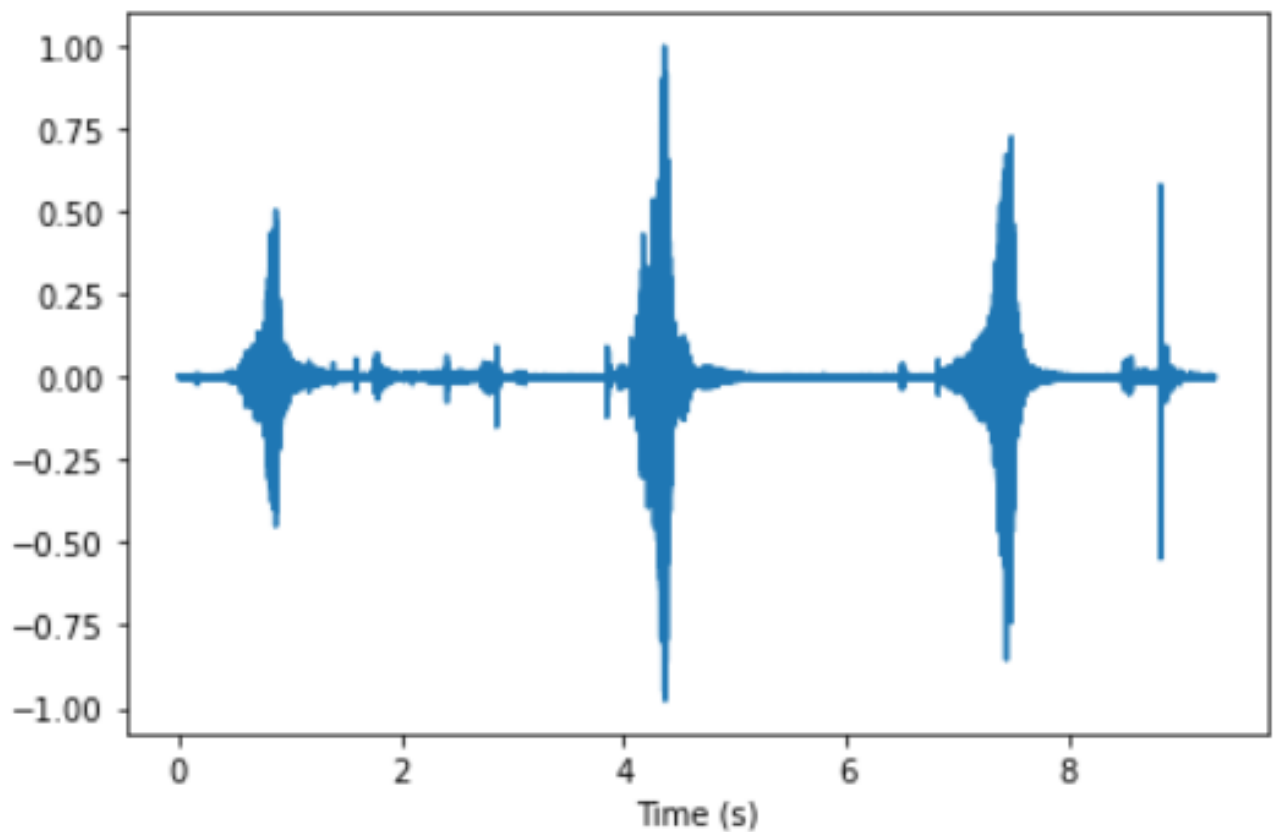


Рис. 2: Результат вывода `wave.plot()`

Далее нам необходимо выбрать сегмент из нашей мелодии. Я взял сегмент с 3.9 секунды и длительностью 0.5 секунды:

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

Листинг 5: Выбор сегмента

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

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

Листинг 6: Вывод `segment.plot()`

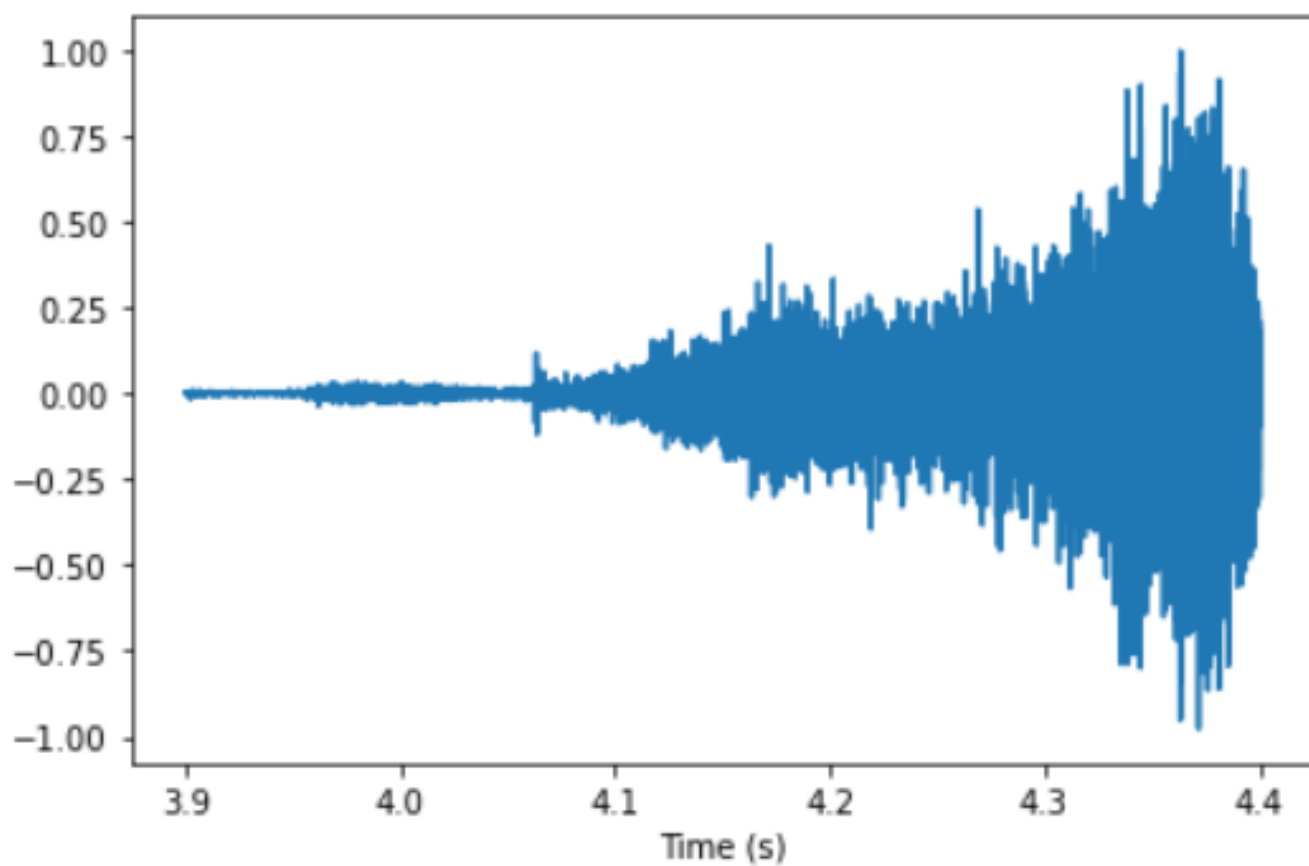


Рис. 3: Результат вывода `segment.plot()`

После этого посмотрим как выглядит наш спектр:

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

Листинг 7: Изображение спектра

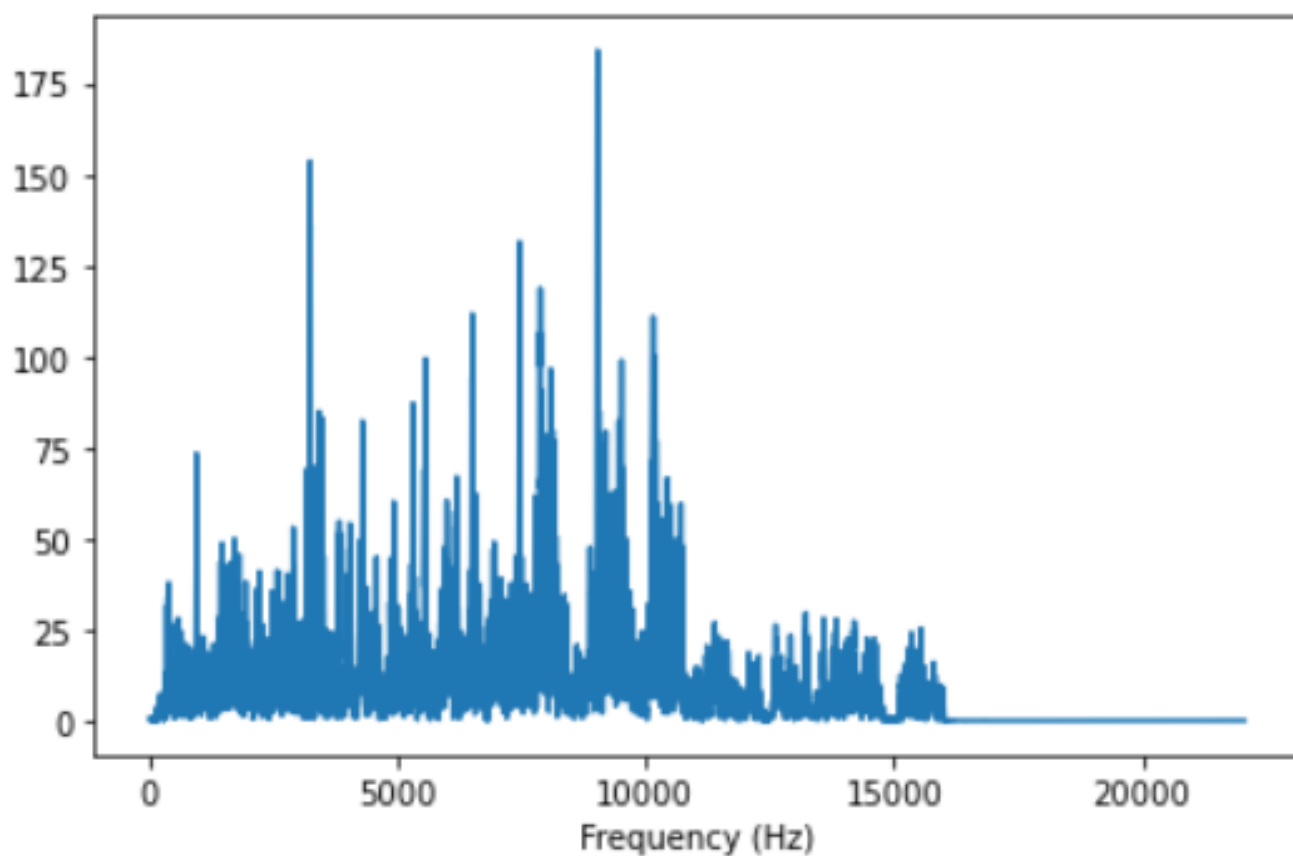


Рис. 4: Изображение спектра

Приблизим наш спектр, чтобы увидеть его более подробно:

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

Листинг 8: Приближенное изображение спектра

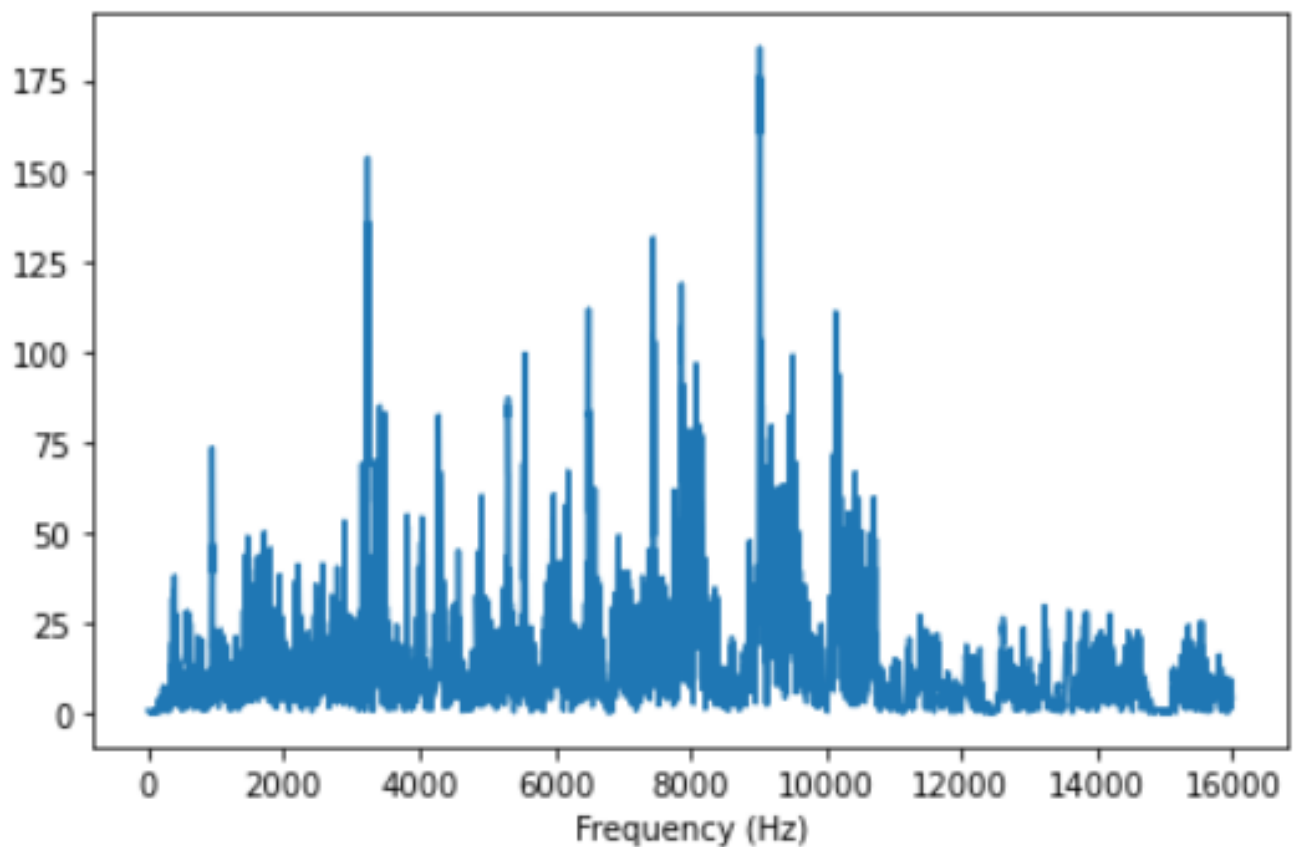


Рис. 5: Приближенное изображение спектра

Далее, полученный нам спектр необходимо отфильтровать:

```
1 filtered = spectrum.make_wave()  
2 filtered.normalize()  
3 filtered.plot()  
4 decorate(xlabel='Time (s)')
```

Листинг 9: Фильтрация спектра

Для наглядности мы вывели полученный отфильтрованный спектр:

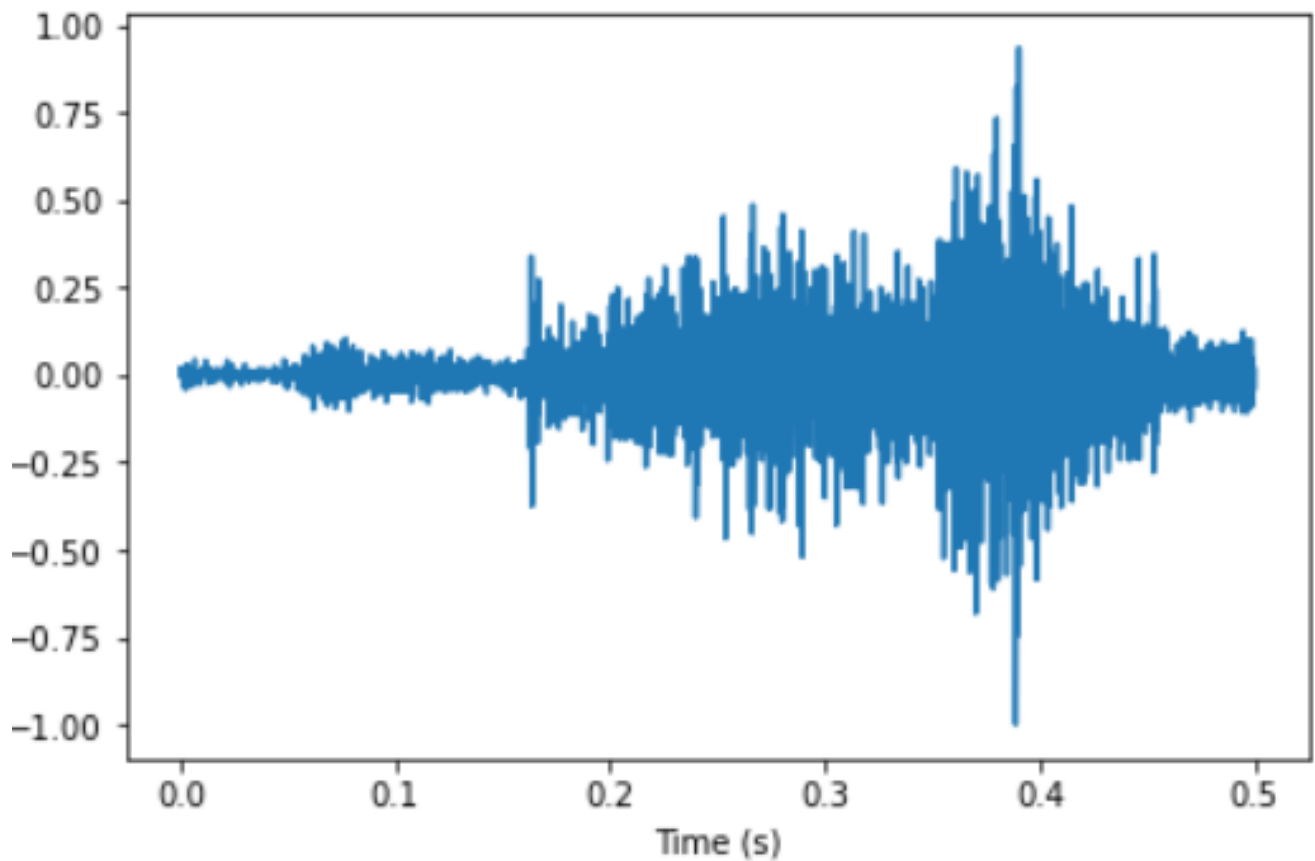


Рис. 6: Отфильтрованный спектр

После всего этого переводим полученный спектр обратно в аудио:

```
In [9]: spectrum.make_wave().make_audio()
```

Out[9]:



Рис. 7: Перевод спектра в аудио

По результатам выполнения данного пункта можно сделать вывод, что полученный звук похож на оригинальный, но более приглушенный, 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:

```
1 from thinkdsp import CosSignal, SinSignal
2
3 cos_sig = CosSignal(freq=400, amp=1.0)
4 sin_sig = SinSignal(freq=755, amp=0.5)
5
6 cos_sig.plot()
7 decorate(xlabel='Time (s)')
8
9 sin_sig.plot()
10 decorate(xlabel='Time (s)')
```

Листинг 10: Создание каналов

В результате получился такой вывод на экран:

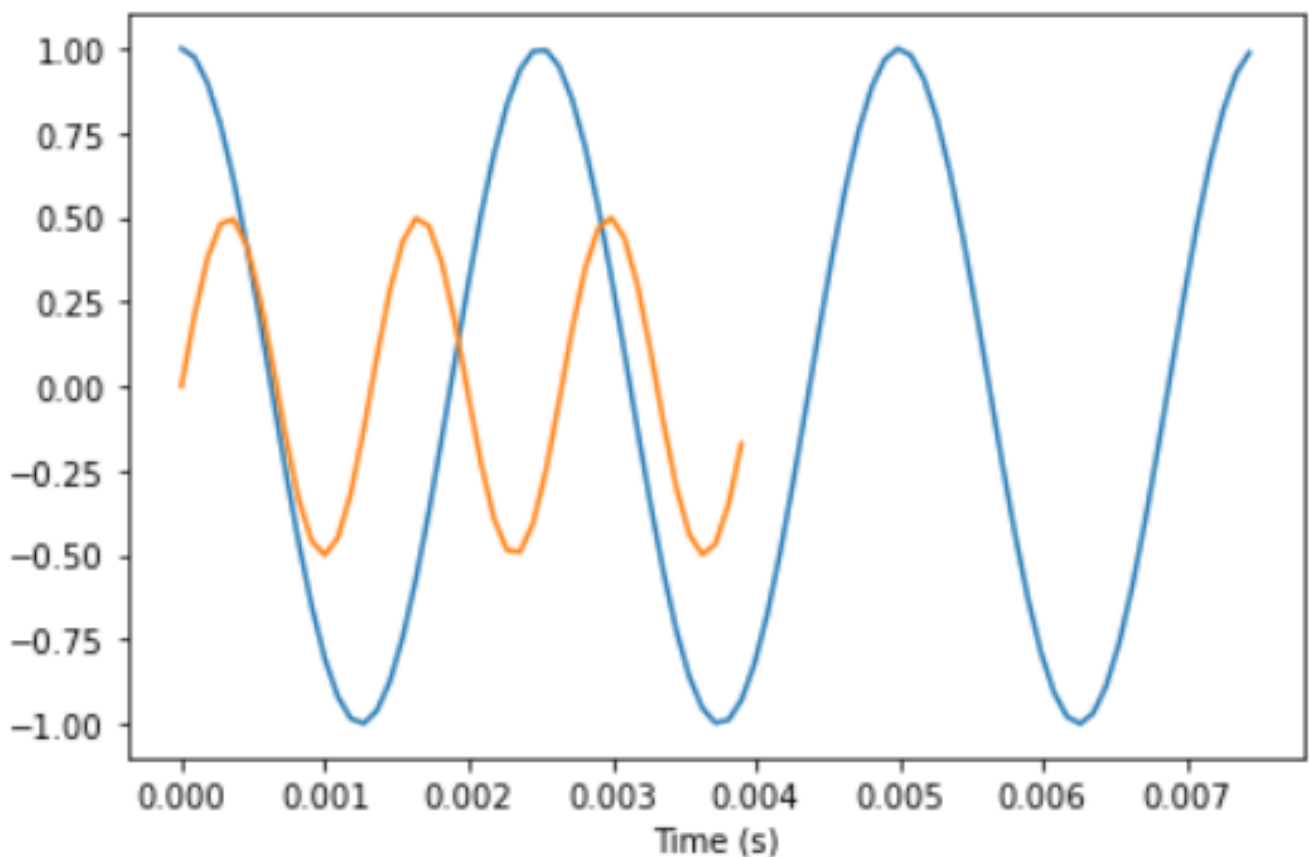


Рис. 8: Вывод спектров

Теперь просуммируем эти два канала и выведем полученный суммирующий сигнал:

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

Листинг 11: Суммирование каналов

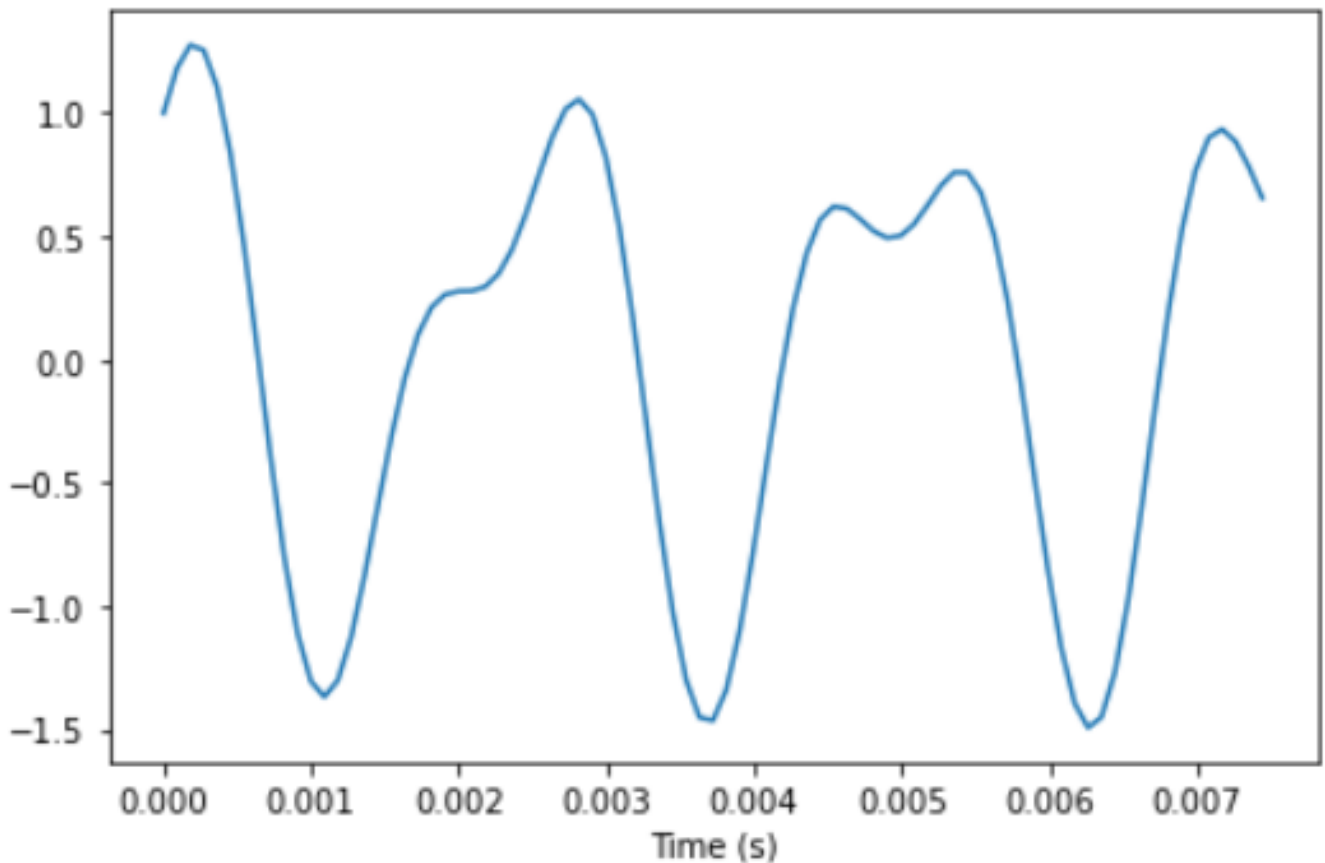


Рис. 9: Суммирование каналов

После этого было получено аудио из данного канала:

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

Out[14]:

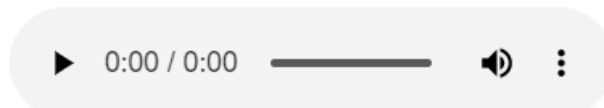


Рис. 10: Получение аудио из суммирующего канала

Наконец, получим спектр из полученного сигнала:

```

1 spectrum = wave1.make_spectrum()
2 spectrum.plot(high=1000)
3 decorate(xlabel='Frequency (Hz)')

```

Листинг 12: Получение спектра

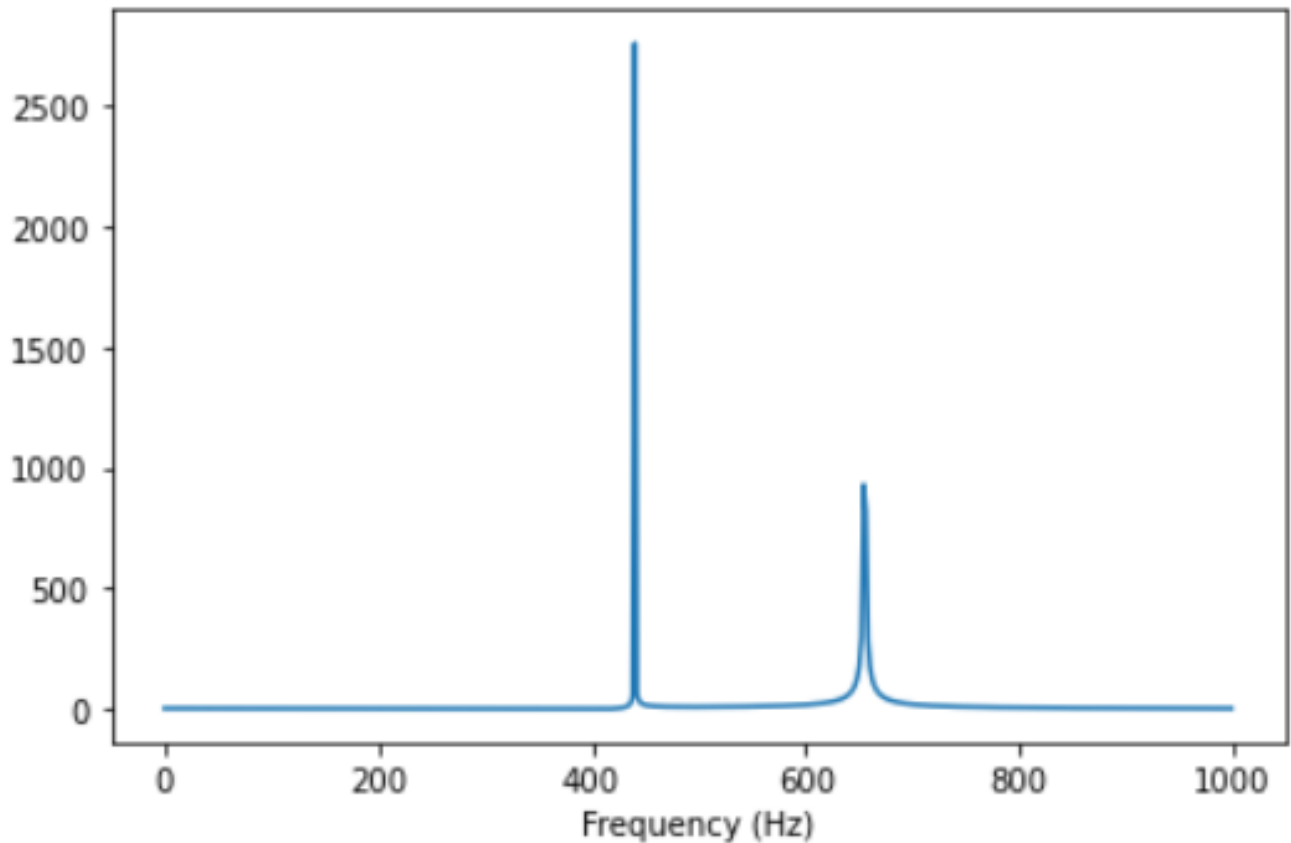


Рис. 11: Полученный спектр

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

```

В [26]: signal += SinSignal(freq=450)
        signal.make_wave().make_audio()

```

Out[26]:

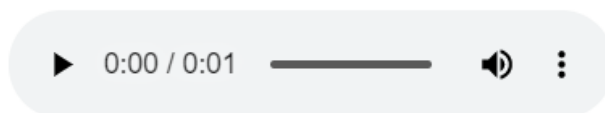


Рис. 12: Добавление к сигналу и перевод в аудио

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

```

1 wave2 = signal.make_wave()
2 wave2.apodize

```

```

3 spectrum2 = wave2.make_spectrum()
4 spectrum2.plot(high=1000)
5 decorate(xlabel='Frequency (Hz)')

```

Листинг 13: Получение спектра и вывод на экран

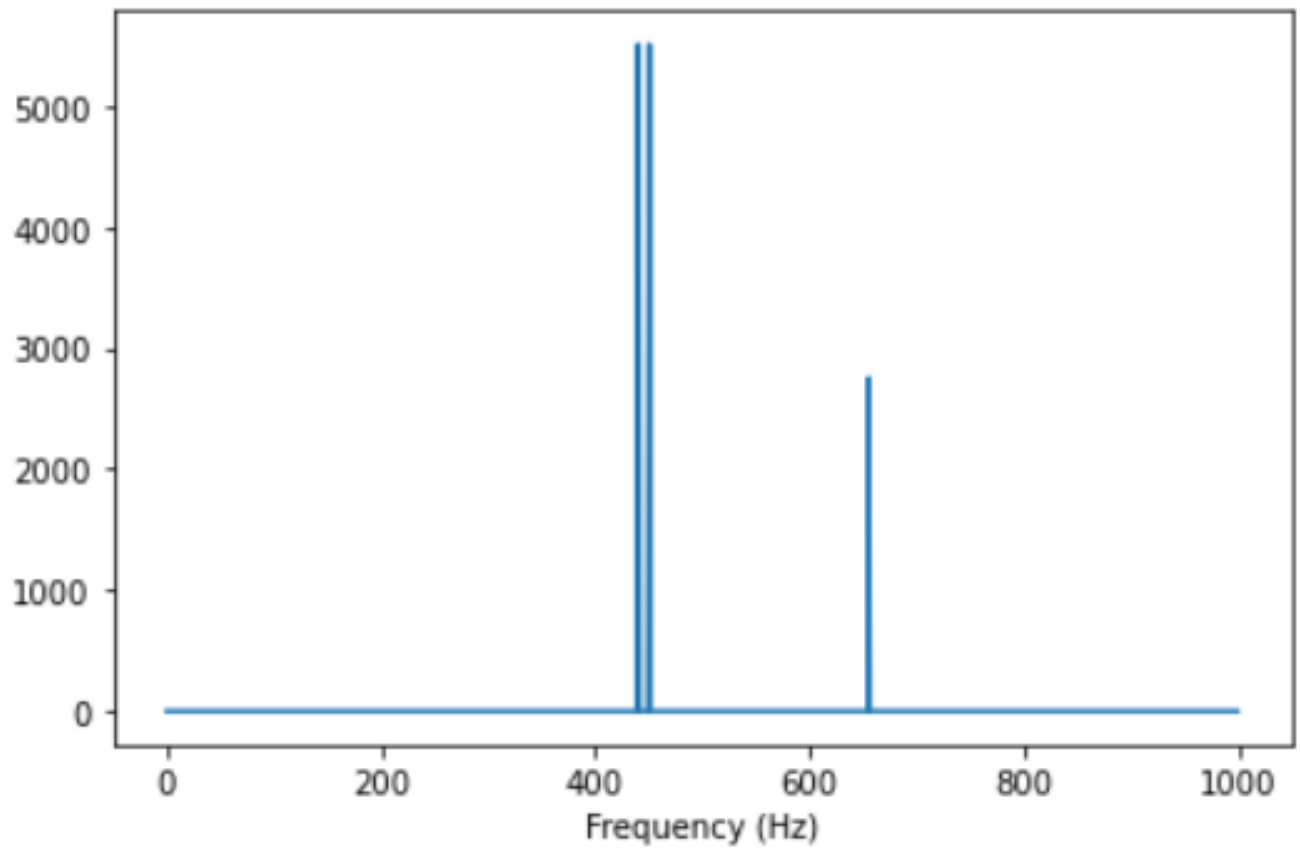


Рис. 13: Полученный спектр

По результатам выполнения данного пункта можно сделать вывод, что полученный в самом конце звук теперь звучит громче и появились колебания, теперь этот звук начинает резать уши во время прослушивания.

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:

```
In [19]: stretch(wave, 0.2)  
         wave.make_audio()
```

Out[19]:



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

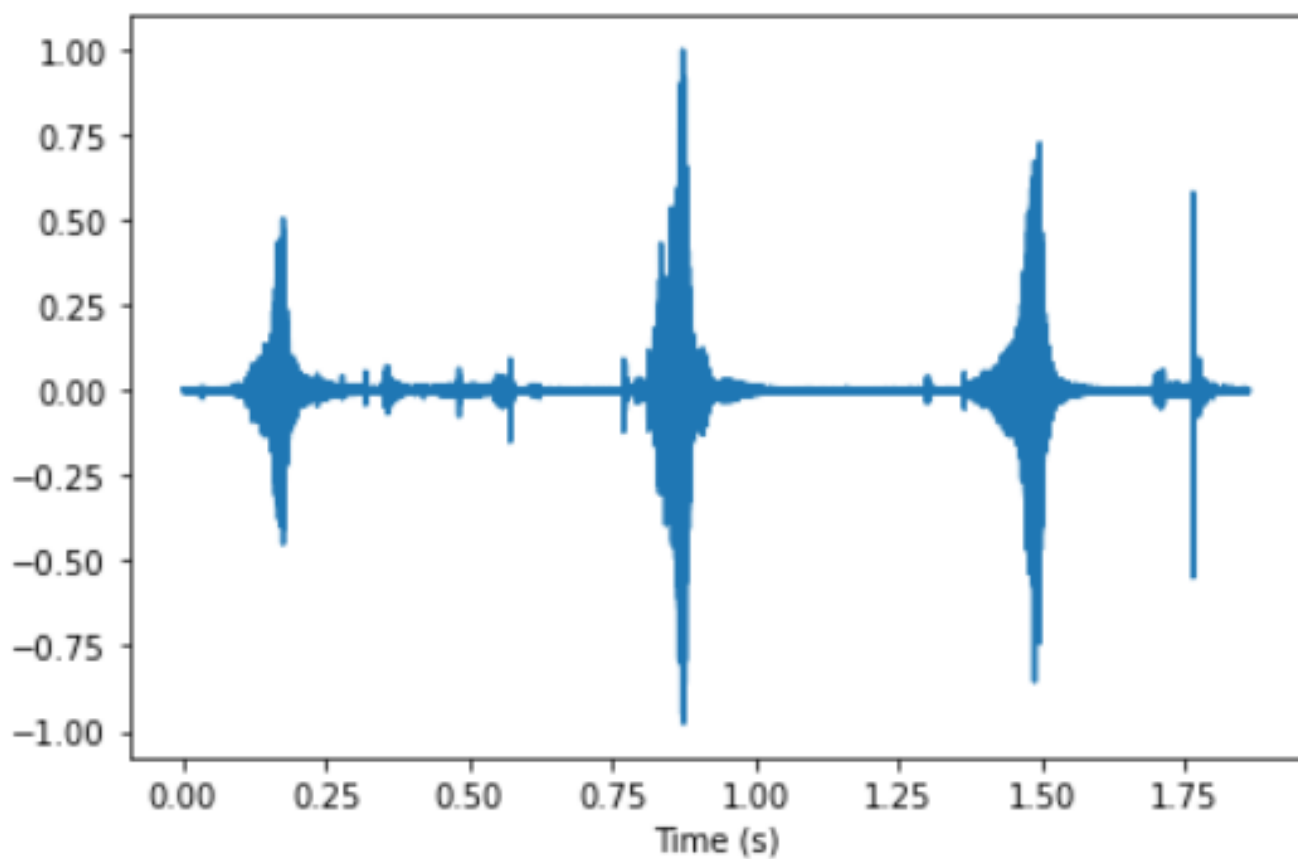


Рис. 15: Полученная в результате вызова функции `stretch` аудиодорожка

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

5 Выводы

В результате выполнения данной лабораторной работы мы изучили как надо работать с сигналами, как их обрабатывать, фильтровать и воспроизводить из с помощью библиотеки Python. Кроме того была реализована и проверена функция для преобразования полученного сигнала. Также нами были получены сведения об основных понятиях и навыки работы с сигналами.