Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Сигналы и звуки

Выполнил студент гр. 3530901/80201 И.С. Иванов

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение №1: Изучить примеры из chap01

Для выполнения первого пункта необходимо изучить и выполнить примеры из файла chap01.ipynb. Запустим все примеры из этого файла.

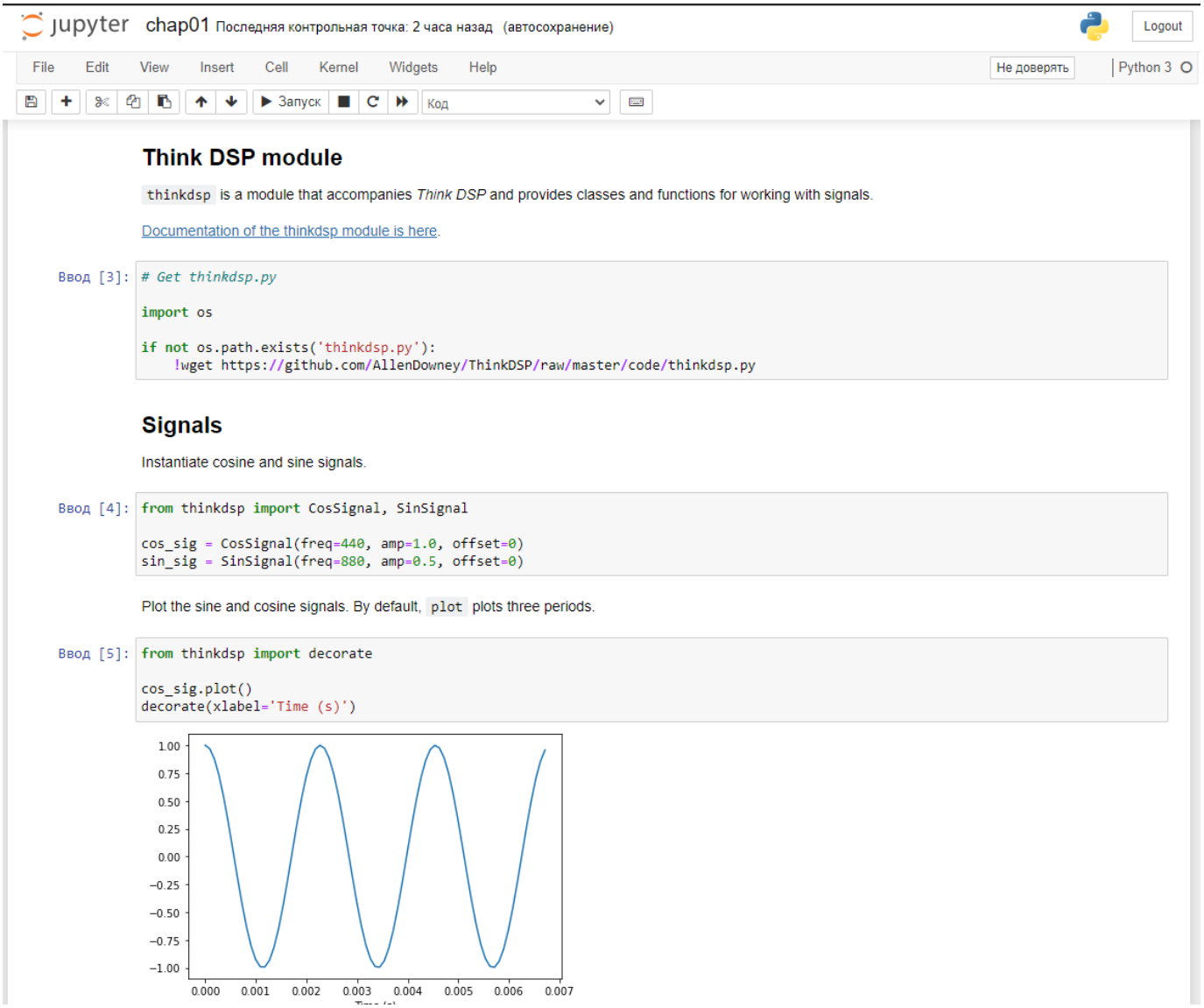
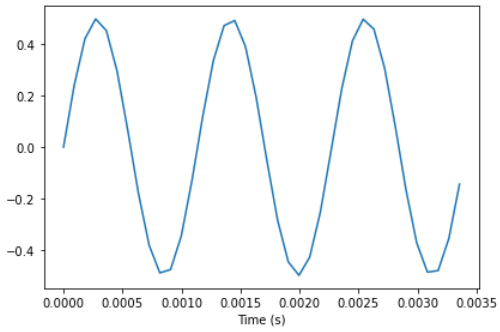


Рис. 1: Изучение и проверка примеров из файла (1)

Here's the sine signal.

Ввод [6]:
```
sin_sig.plot()
decorate(xlabel='Time (s)')
```



Notice that the frequency of the sine signal is doubled, so the period is halved.

The sum of two signals is a SumSignal.

Ввод [7]:
```
mix = sin_sig + cos_sig
mix
```

Out[7]:
```
<thinkdsp.SumSignal at 0x186cb61ce80>
```

Here's what it looks like.

Ввод [8]:
```
mix.plot()
decorate(xlabel='Time (s)')
```

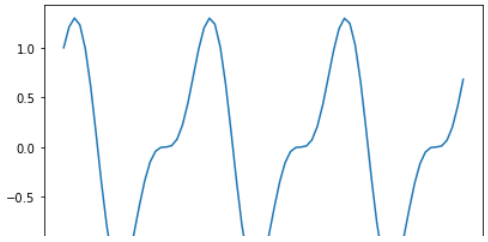


Рис. 2: Изучение и проверка примеров из файла (2)

2 Упражнение №2: Обработка сигналов

Во втором упражнении необходимо сначала скачать с сайта https://freesound.org/ любой образец звука, имеющий четко выраженную высоту и выделить в нем полсуекундный фрагмент с постоянной высотой. Необходимо вычислить и рассчитать спектр выбранного фрагмента. Далее нужно произвести фильтрацию гармоник и преобразовать спектры обратно в сигнал.

Необходимо проверить наличие файла `thinkdsp.py` и скачать его при отсутствии:

```
1         import os
2         if not os.path.exists('thinkdsp.py'):
3             from urllib.request import urlretrieve
4             urlretrieve("https://github.com/AllenDowney/ThinkDSP/raw/master/code/
    thinkdsp.py", "thinkdsp.py")
```

Листинг 1: Проверка наличия `thinkdsp.py`

Далее импортируем из раннее проверенного файла функции, и прочитаем файл с образцом звука:

```
from thinkdsp import read_wave
wave = read_wave('Sounds/105384__drzoom__hitting-pan-top-large.wav')
wave.normalize()
wave.make_audio()
```

Листинг 2: Чтение скаченного образца звука

После чтения образца звука выведем график:

```
wave.plot()
```

Листинг 3: Вывод графика образца

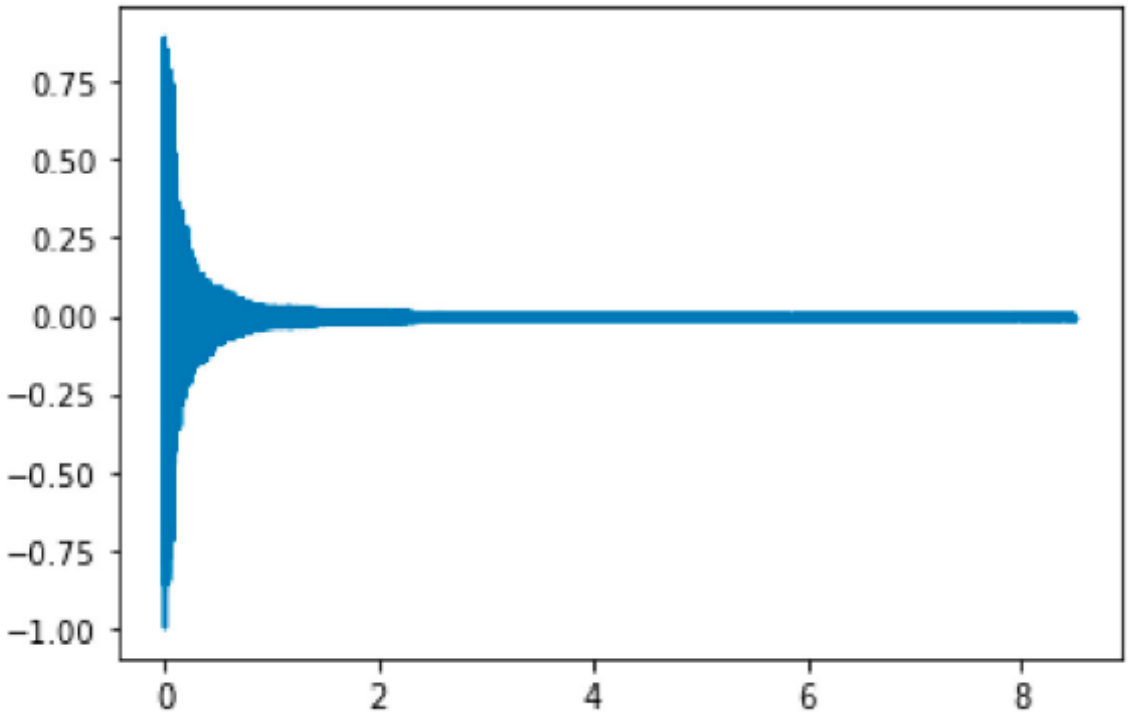


Рис. 3: График образца звука

Далее необходимо выбрать сегмент из нашего образца звука. Был взят сегмент от нулевой секунды длительностью 0.3 секунды:

```
segment = wave.segment(start=0, duration=0.3)
segment.make_audio()
```

Листинг 4: Выбор сегмента

Теперь выведем график на экран:

```
segment.plot()
```

Листинг 5: Вывод графика

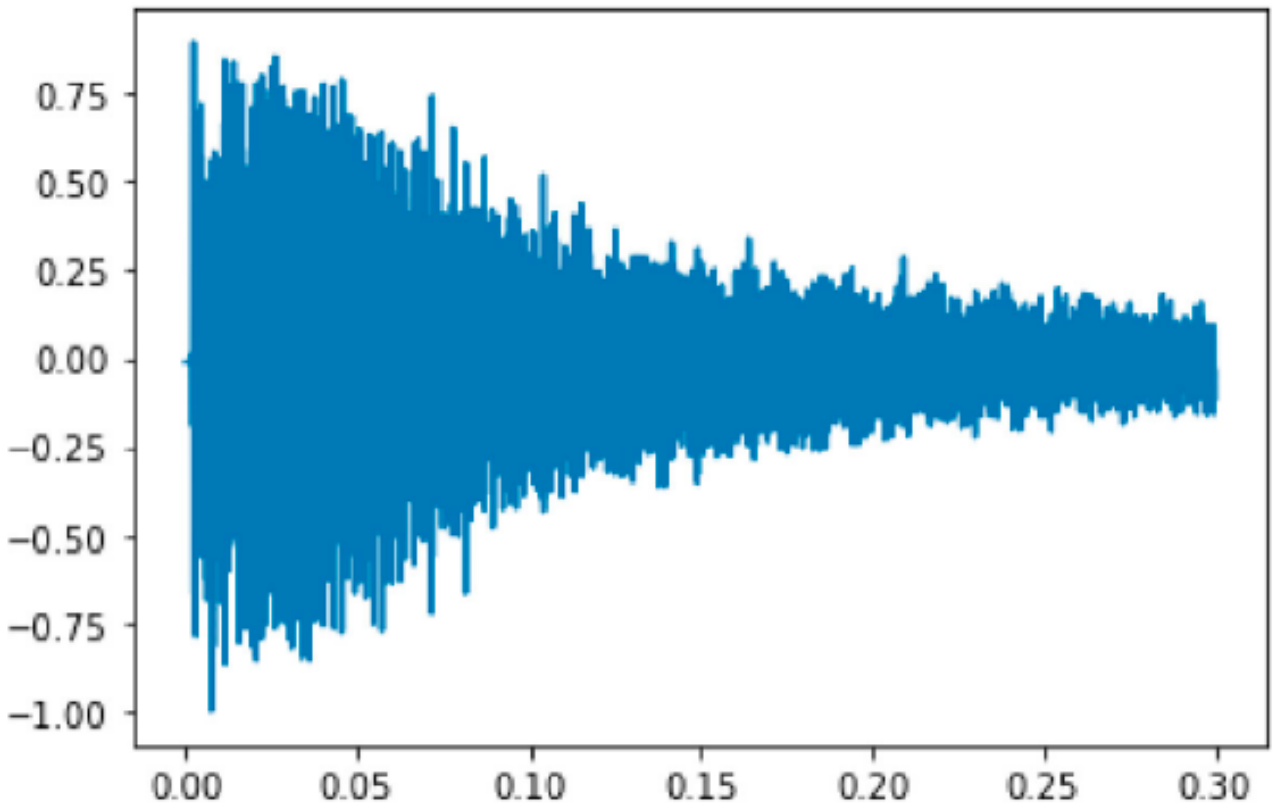


Рис. 4: График сегмента

После этого посмотрим как выглядит наш спектр:

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

Листинг 6: Изображение спектра

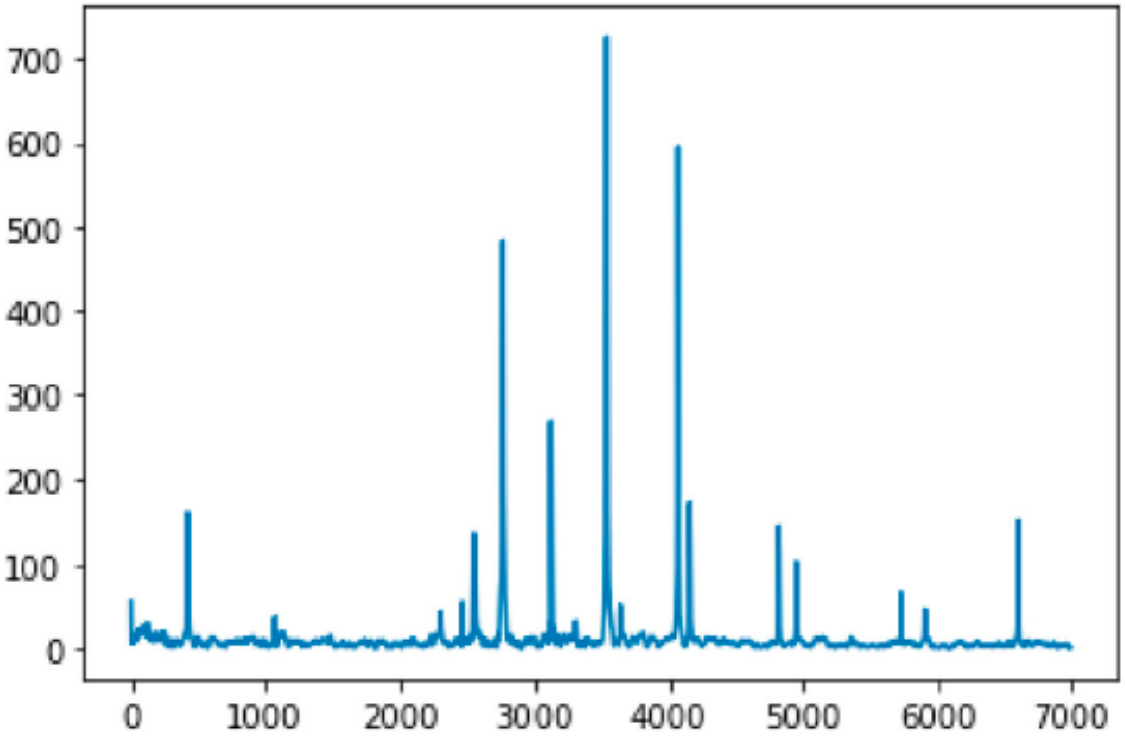


Рис. 5: Изображение спектра

Увеличим спектр

```
spectrum = segment.make_spectrum()
spectrum.plot(high=4500)

```

Листинг 7: Приближенное изображение спектра

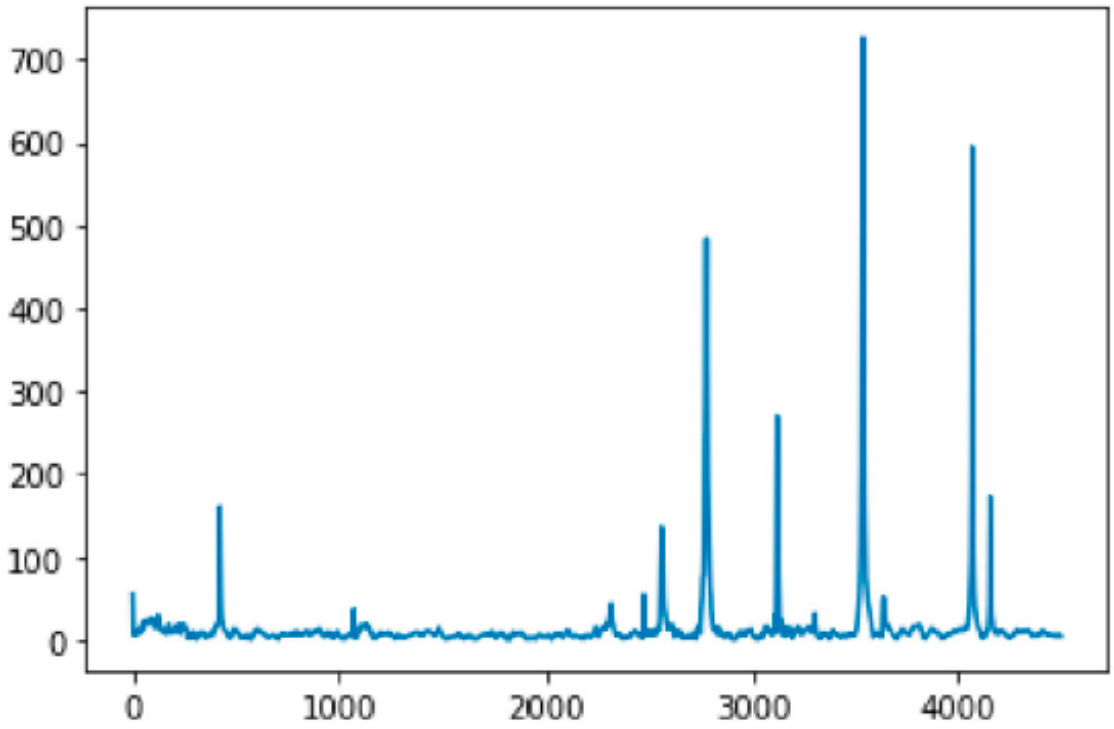


Рис. 6: Увеличенное изображение спектра

Далее, фильтрация спектра:

```
spectrum.low_pass(2000)
spectrum.plot(high=4000)

```

Листинг 8: Фильтрация спектра

Даллее вывод графика отфильтрованного спектра:

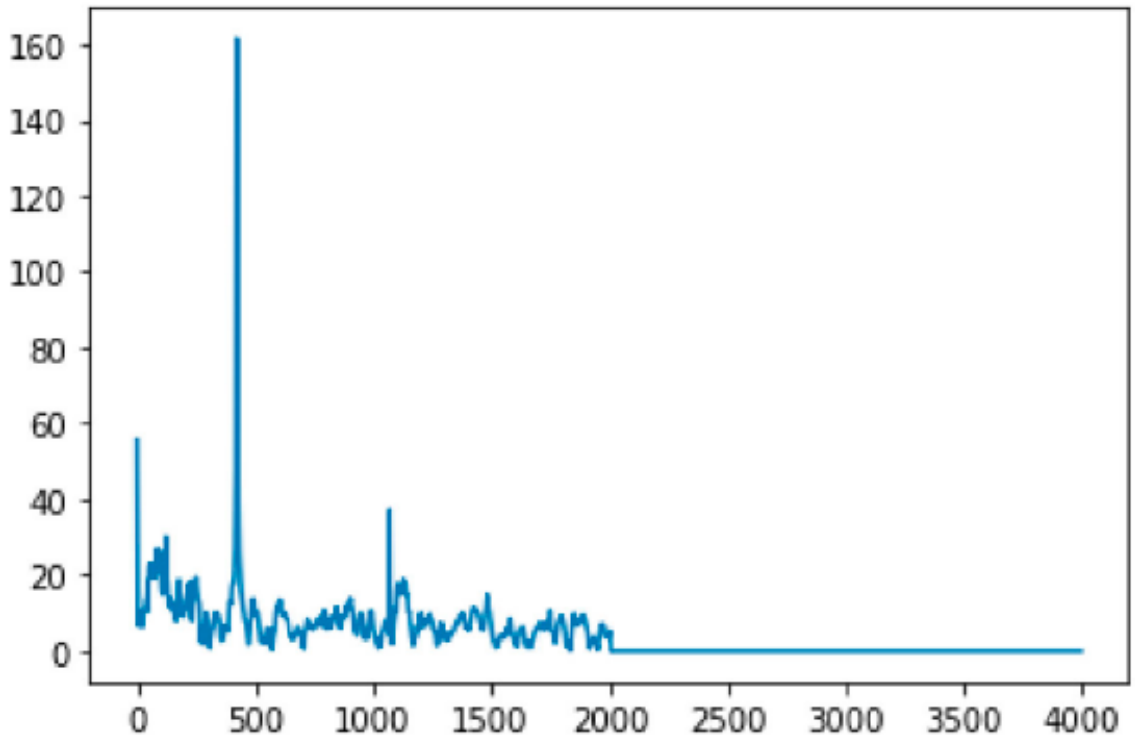


Рис. 7: Отфильтрованный спектр

Переводи отфильтрованный спектр обратно в аудио:

```
spectrum.make_wave().make_audio()
```

Листинг 9: Перевод в аудио

По результатам выполнеия данного упражнения можно сделать вывод, что в полученном звуке отсутствуют частоты выше 2000 Гц. Звук стал похожим на звук из низкокачественный динамика .

3 Упражнение №3: Сложение сигналов

В третьем упражнении нам необходимо создать сложный сигнал из объектов `SinSignal` и `CosSignal` суммируя их. Также следует обработать полученный сигнал для получения wave, прослушать его, вычислить для него `Spectrum` и после вывести на экран.

```
from thinkdsp import SinSignal, CosSignal

sin_signal = (SinSignal(freq=400, amp=1.0) +
          SinSignal(freq=600, amp=0.5) +
          SinSignal(freq=800, amp=0.25))

cos_signal = (CosSignal(freq=400, amp=1.0) +
          CosSignal(freq=600, amp=0.5) +
          CosSignal(freq=800, amp=0.25))

sin_signal.plot()
cos_signal.plot()

```

Листинг 10: Создание сигналов

В результате получился такой вывод на экран:

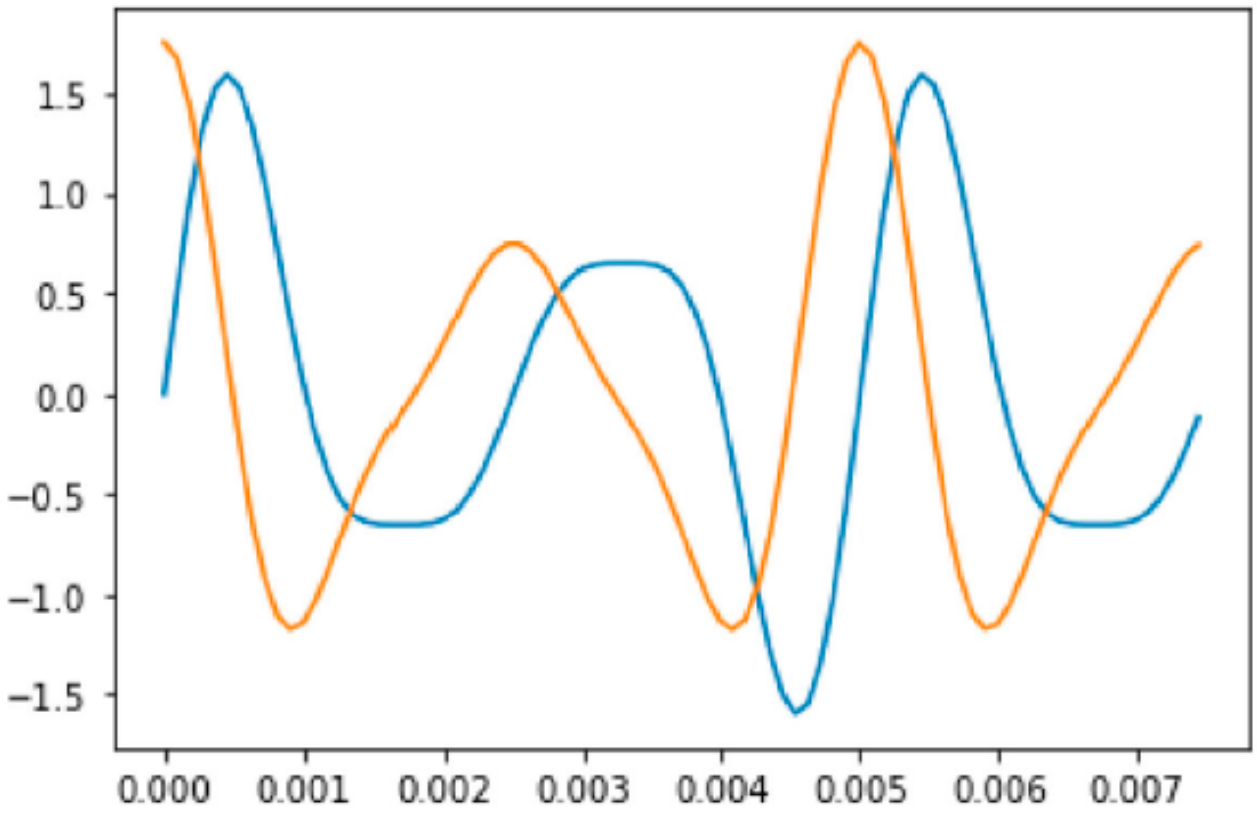


Рис. 8: График сигналов

Теперь сложим два канала и выведем полученный график:

```
signal = sin_signal + cos_signal
signal.plot()
```

Листинг 11: Суммирование каналов

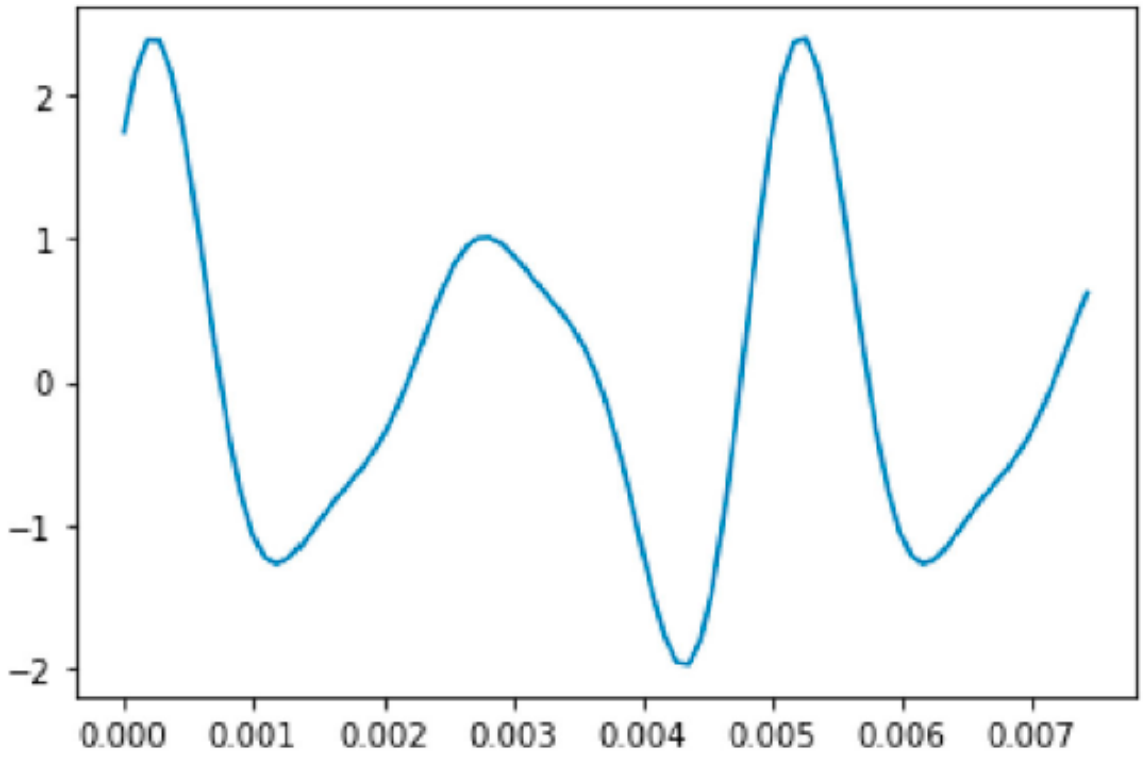


Рис. 9: Суммированый сигнал

После этого было получено аудио из данного сигнала:

```
wave2 = signal.make_wave(duration=0.5)
wave2.make_audio()
```

Листинг 12: Получение аудио

Получим спектр из полученного сигнала:

```
spectrum2 = wave2.make_spectrum()
spectrum2.plot(high=1000)
```

Листинг 13: Получение спектра

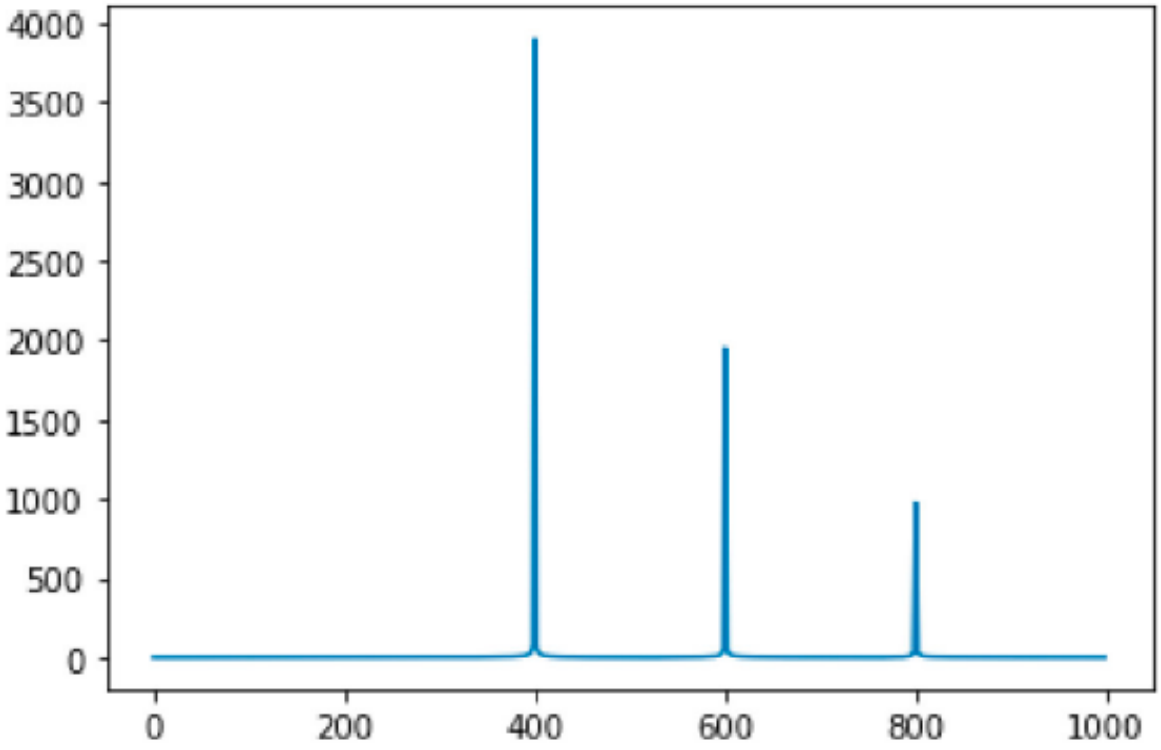


Рис. 10: Полученный спектр

Добавим к сигналу синусоидальный сигнал с частотой 133 Hz и переведем в аудио:

```
signal += SinSignal(freq=133, amp=0.75)
wave3 = signal.make_wave()
wave3.make_audio()

```

Листинг 14: Получение спектра

Получим спектр из сигнала:

```
wave3.apodize()
spectrum3 = wave3.make_spectrum()
spectrum3.plot(high=1000)

```

Листинг 15: Полученый спектр

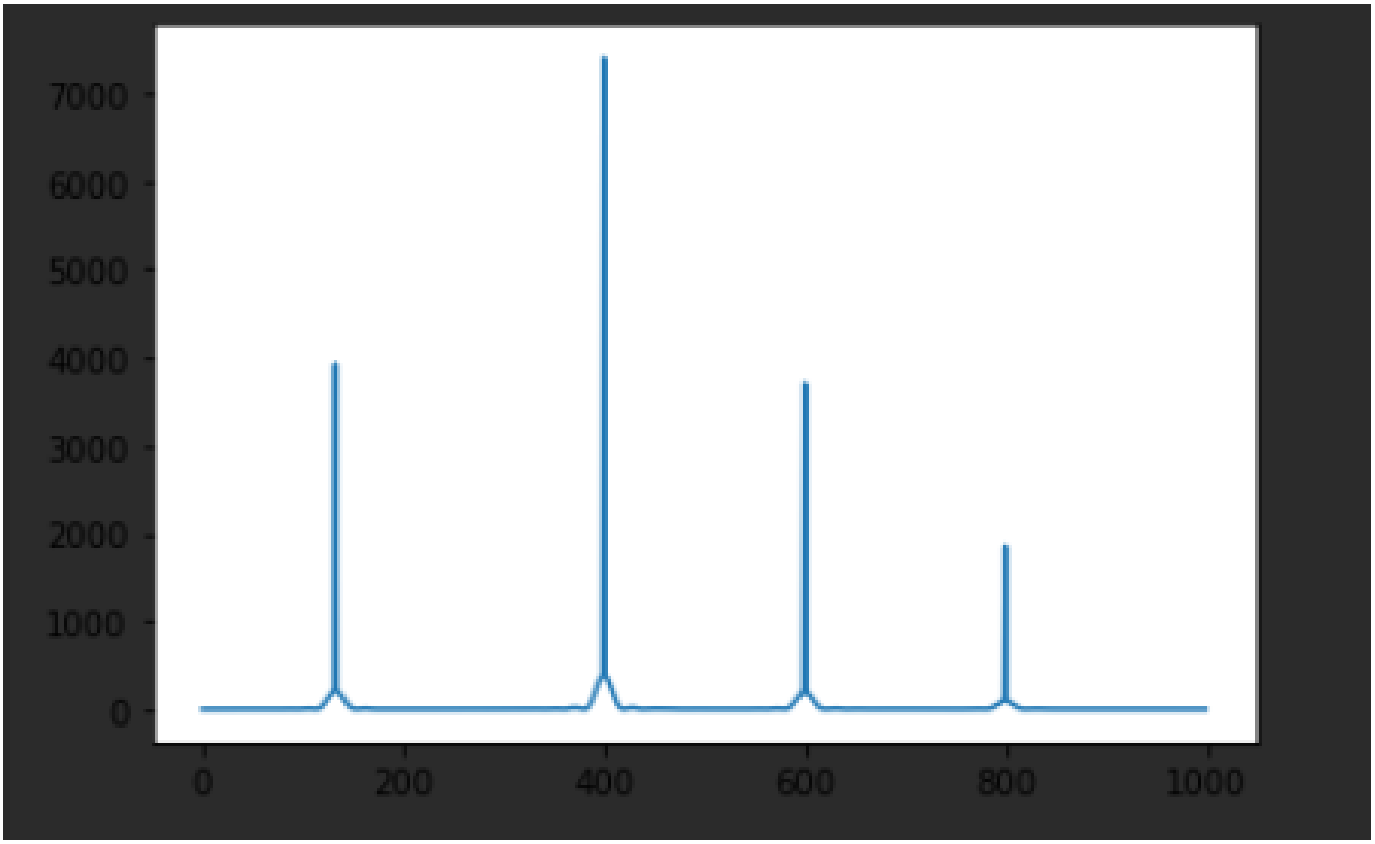


Рис. 11: Полученный спектр

По результатам выполнеия данного упражнения можно сделать вывод, что в полученном в конце звуке появились колебания.

4 Упражнение №4: Изменение длительности сигнала

В четвертом упражнении 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`.

Напишем саму функцию `stretch`:

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

```

Листинг 16: Функция `stretch`

Прочитаем раннее скачанный звуковой файл:

```
wave4 = read_wave('Sounds/105384__drzoom__hitting-pan-top-large.wav')
wave4.normalize()
wave4.make_audio()

```

Листинг 17: Чтение файла

Вызов функции `stretch`. Параметры wav = wave4 и factor = 0.5, затем получим аудио:

```
stretch(wave4, 0.5)
wave4.make_audio()

```

Листинг 18: Вызов функции `stretch`

Спектр полученной дорожки:

```
wave4.plot()

```

Листинг 19: Спектр после вызова функции `stretch`

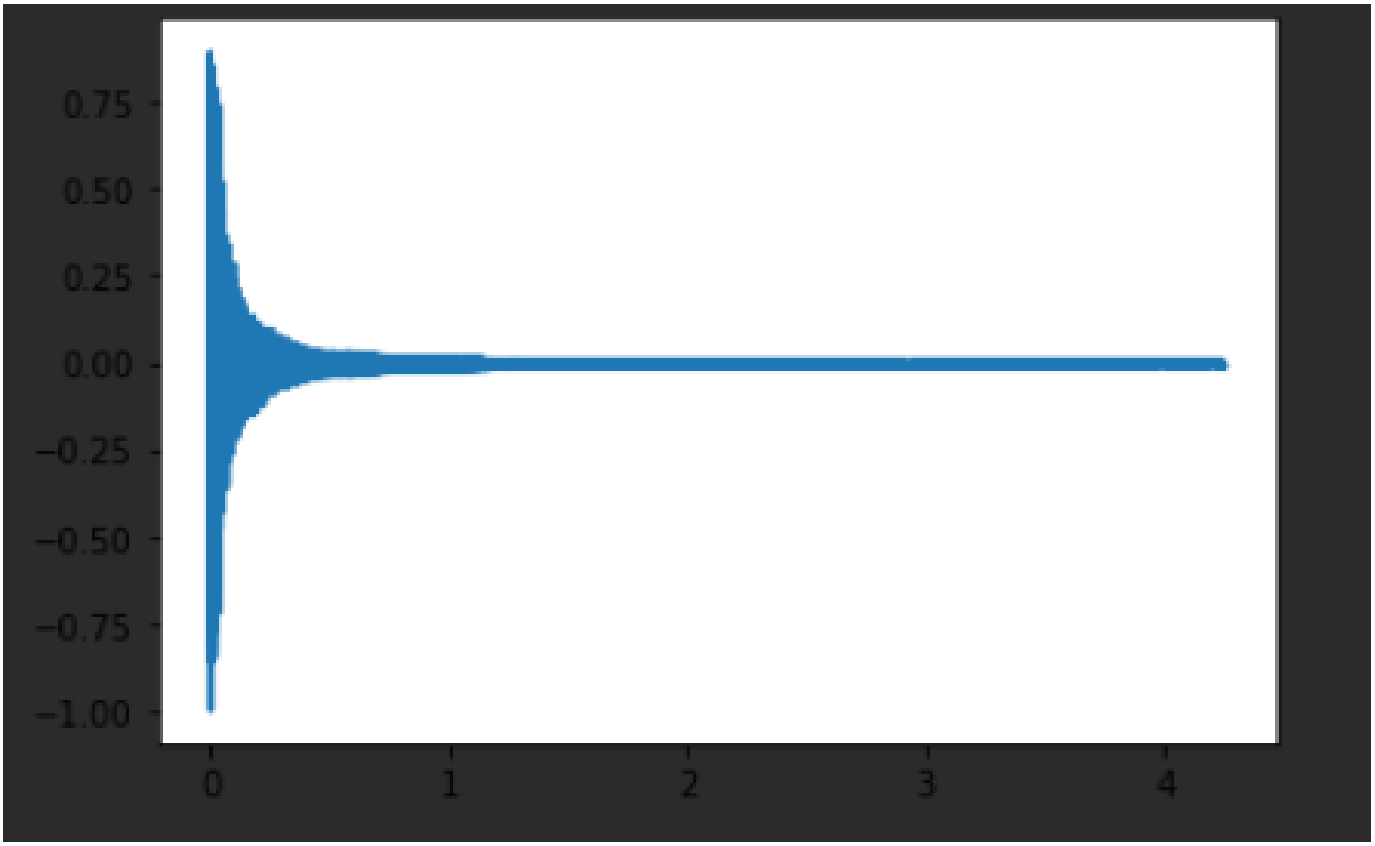


Рис. 12: Спектр аудиодорожки

По результатам выполнения данного упражнения 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.