

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

1	Упражнение №1: Изучить примеры из chap01	5
2	Упражнение №2: Обработка сигналов	6
3	Упражнение №3: Сложение сигналов	12
4	Упражнение №4: Изменение длительности сигнала	17
5	Выводы	19

Список иллюстраций

1	Изучение и проверка примеров из файла (1)	5
2	Изучение и проверка примеров из файла (2)	6
3	График образца звука	7
4	График сегмента	8
5	Изображение спектра	9
6	Увеличенное изображение спектра	10
7	Отфильтрованный спектр	11
8	График сигналов	13
9	Суммированный сигнал	14
10	Полученный спектр	15
11	Полученный спектр	16
12	Спектр аудиодорожки	18

Листинги

1	Проверка наличия thinkdsp.py	6
2	Чтение скаченного образца звука	7
3	Вывод графика образца	7
4	Выбор сегмента	8
5	Вывод графика	8
6	Изображение спектра	8
7	Приближенное изображение спектра	9
8	Фильтрация спектра	10
9	Перевод в аудио	11
10	Создание сигналов	12
11	Суммирование каналов	13
12	Получение аудио	14
13	Получение спектра	14
14	Получение спектра	15
15	Полученный спектр	15
16	Функция stretch	17
17	Чтение файла	17
18	Вызов функции stretch	17
19	Спектр после вызова функции stretch	17

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.

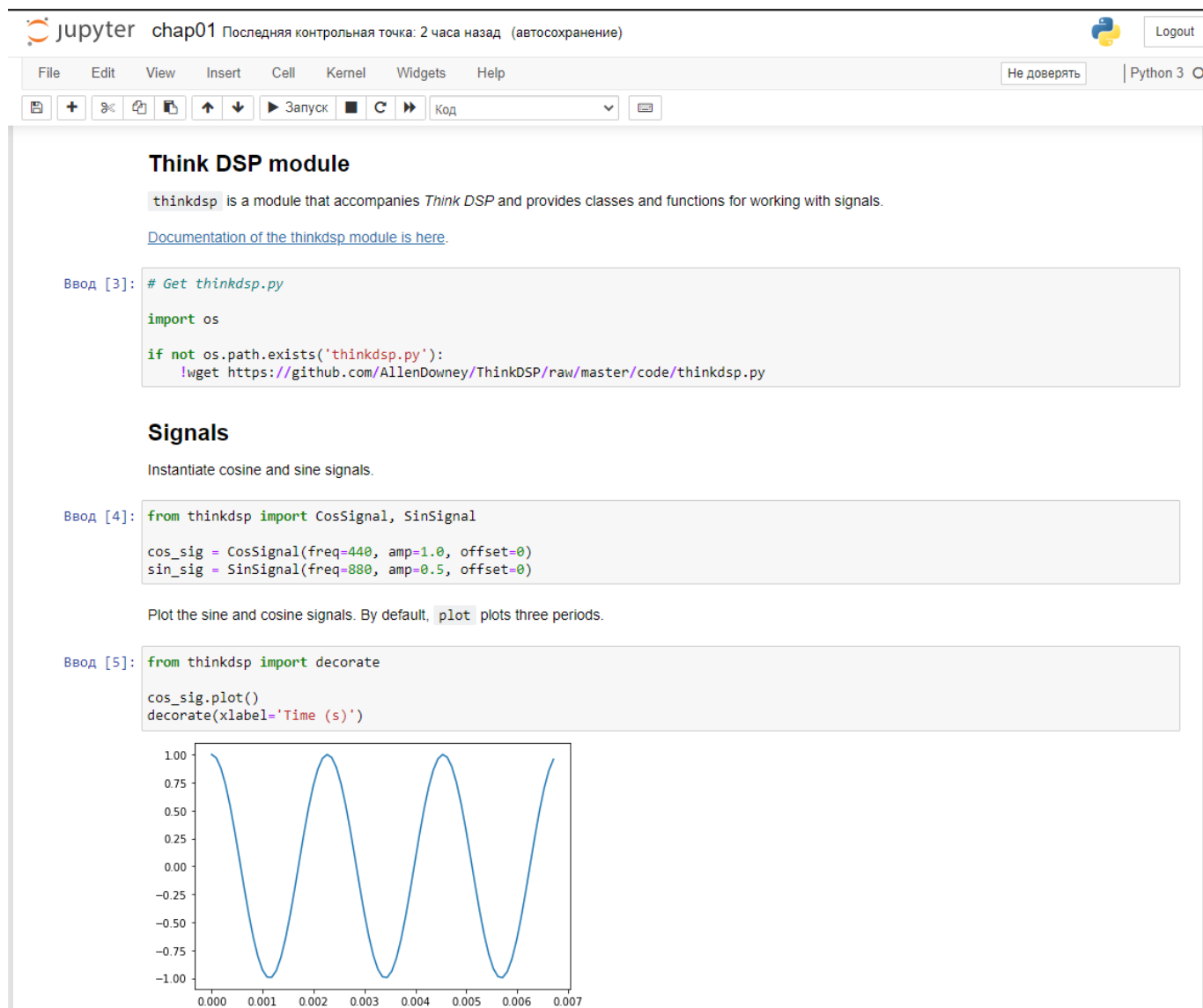
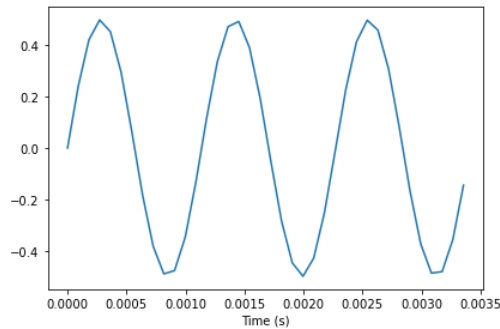


Рис. 1: Изучение и проверка примеров из файла (1)

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)')
```

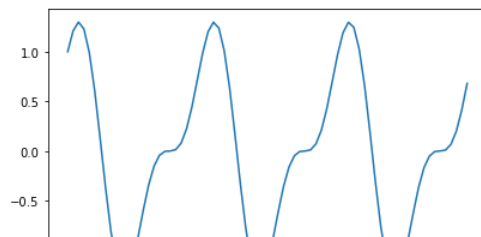


Рис. 2: Изучение и проверка примеров из файла (2)

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 полусекундный 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 читаем файл с образцом звука:

```

1     from thinkdsp import read_wave
2     wave = read_wave('Sounds/105384__drzoom__hitting-pan-top-large.wav')
3     wave.normalize()
4     wave.make_audio()
5

```

Листинг 2: Чтение скаченного образца звука

После чтения образца звука выведем график:

```

1     wave.plot()
2

```

Листинг 3: Вывод графика образца

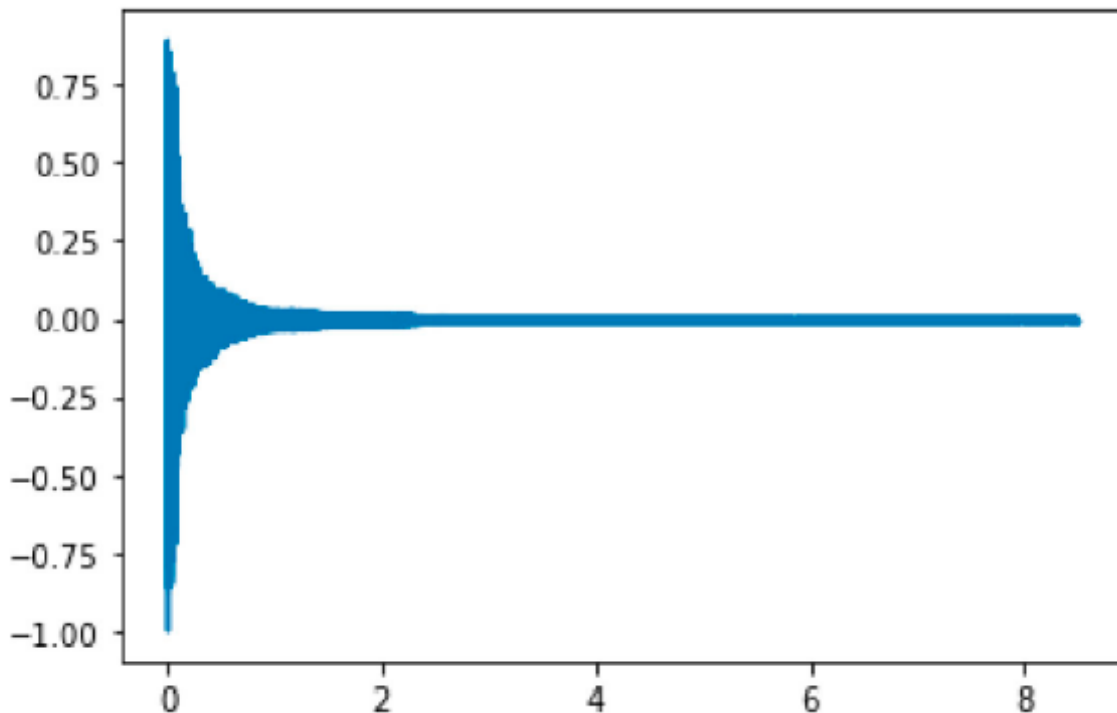


Рис. 3: График образца звука

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

```

1     segment = wave.segment(start=0, duration=0.3)
2     segment.make_audio()
3

```

Листинг 4: Выбор сегмента

Теперь выведем график на экран:

```

1     segment.plot()
2

```

Листинг 5: Вывод графика

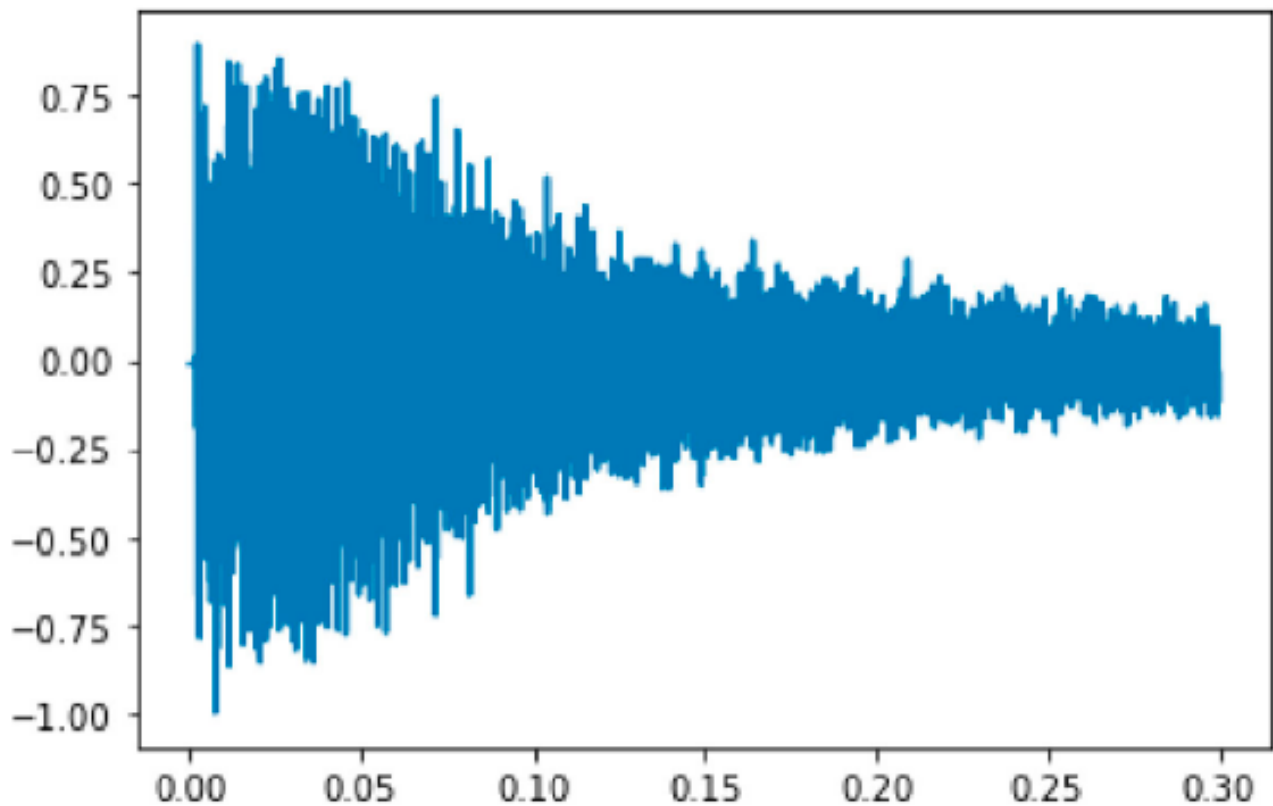


Рис. 4: График сегмента

После этого посмотрим как выглядит наш спектр:

```

1     spectrum = segment.make_spectrum()
2     spectrum.plot(high=7000)
3

```

Листинг 6: Изображение спектра

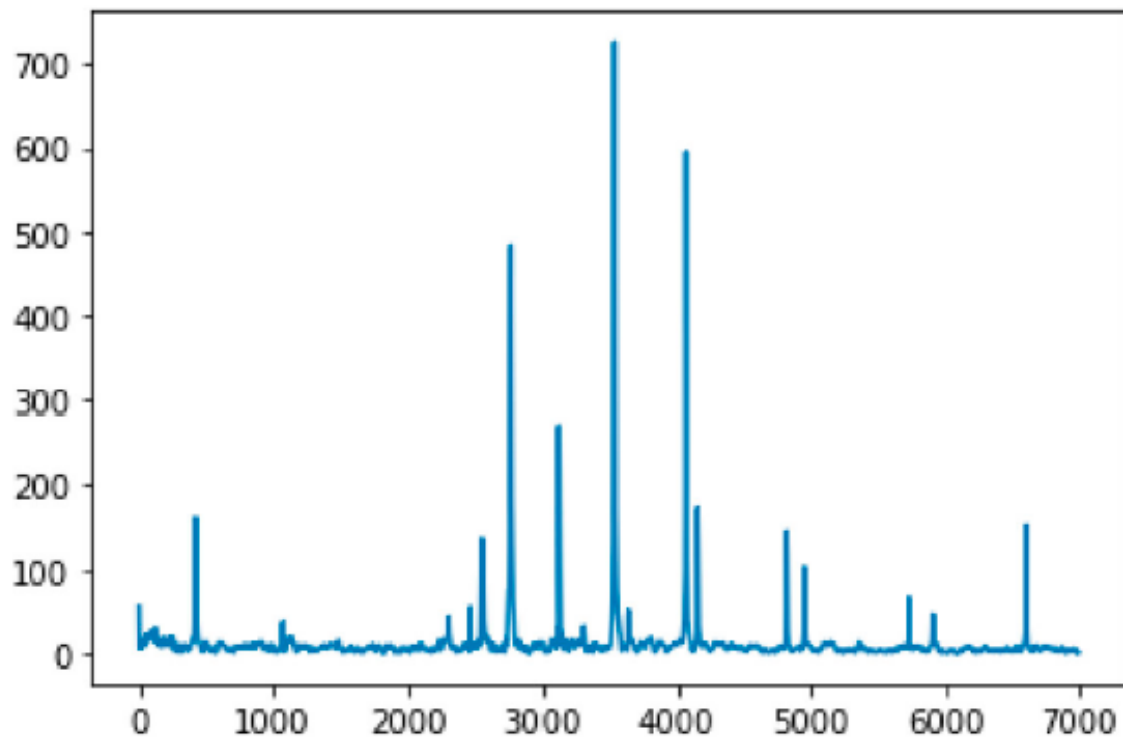


Рис. 5: Изображение спектра

Увеличим спектр

```
1 spectrum = segment.make_spectrum()  
2 spectrum.plot(high=4500)  
3
```

Листинг 7: Приближенное изображение спектра

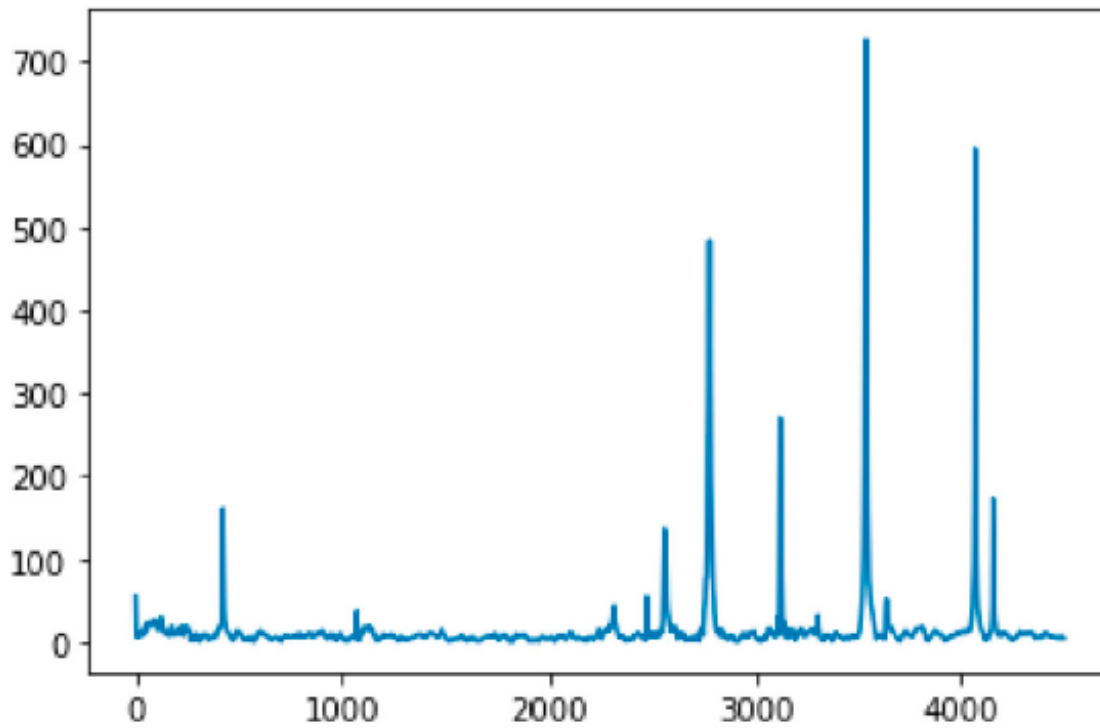


Рис. 6: Увеличенное изображение спектра

Далее, фильтрация спектра:

```
1 spectrum.low_pass(2000)
2 spectrum.plot(high=4000)
3
```

Листинг 8: Фильтрация спектра

Далее вывод графика отфильтрованного спектра:

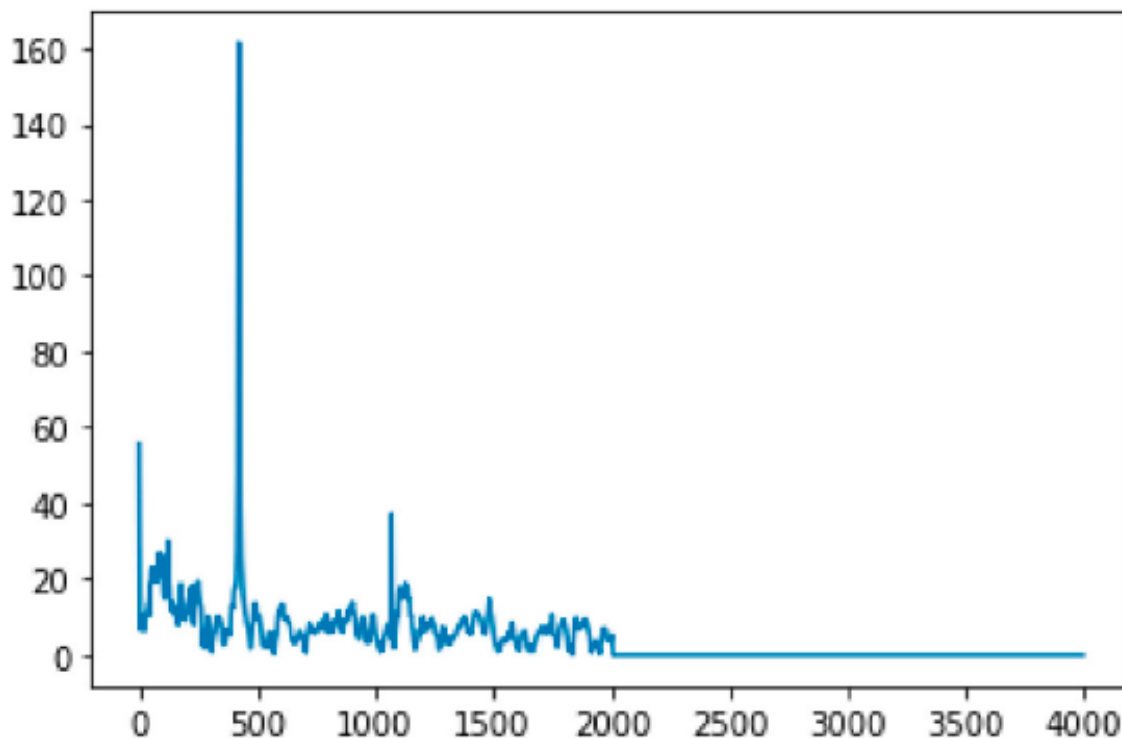


Рис. 7: Отфильтрованный спектр

Переводи отфильтрованный спектр обратно в аудио:

```
1 spectrum.make_wave().make_audio()
2
```

Листинг 9: Перевод в аудио

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

```
1      from thinkdsp import SinSignal, CosSignal
2
3      sin_signal = (SinSignal(freq=400, amp=1.0) +
4                    SinSignal(freq=600, amp=0.5) +
5                    SinSignal(freq=800, amp=0.25))
6
7      cos_signal = (CosSignal(freq=400, amp=1.0) +
8                    CosSignal(freq=600, amp=0.5) +
9                    CosSignal(freq=800, amp=0.25))
10
11     sin_signal.plot()
12     cos_signal.plot()
13
```

Листинг 10: Создание сигналов

В результате получился такой вывод на экран:

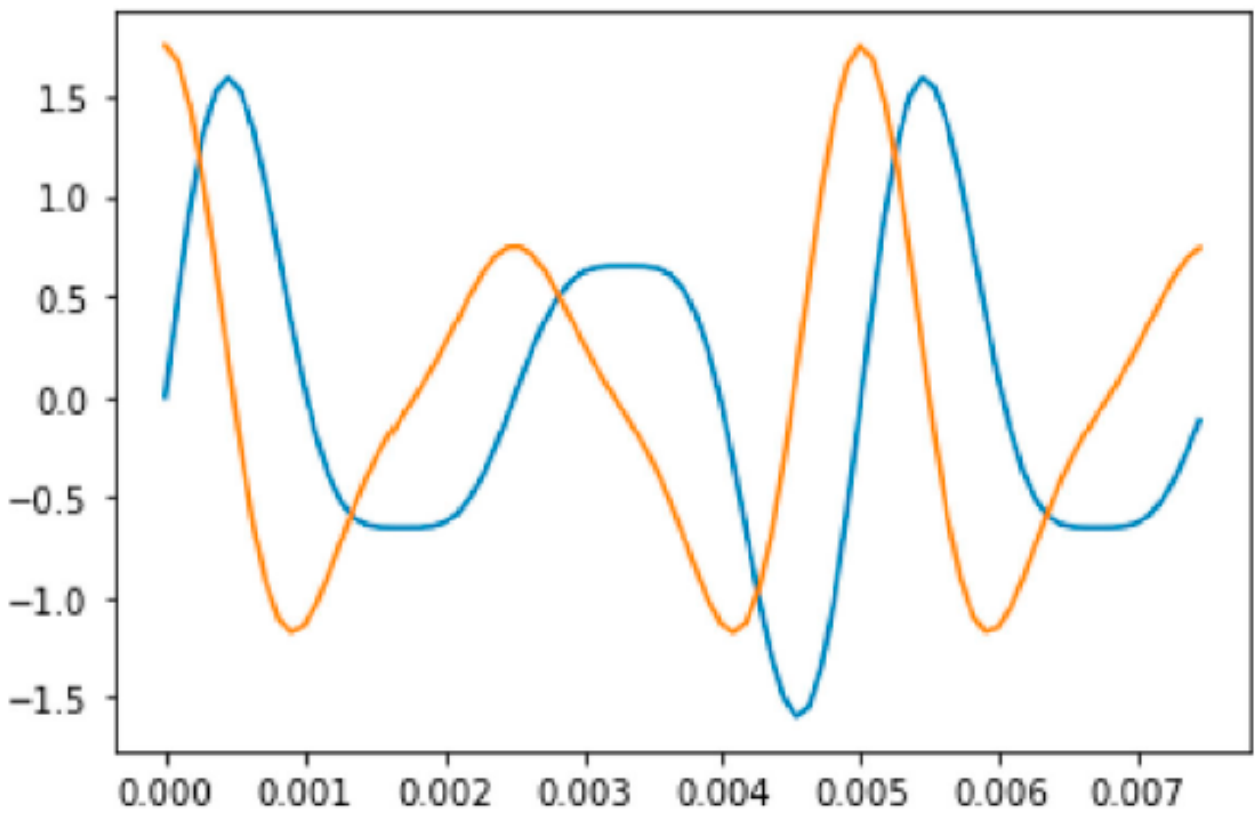


Рис. 8: График сигналов

Теперь сложим два канала и выведем полученный график:

```
1 signal = sin_signal + cos_signal  
2 signal.plot()  
3
```

Листинг 11: Суммирование каналов

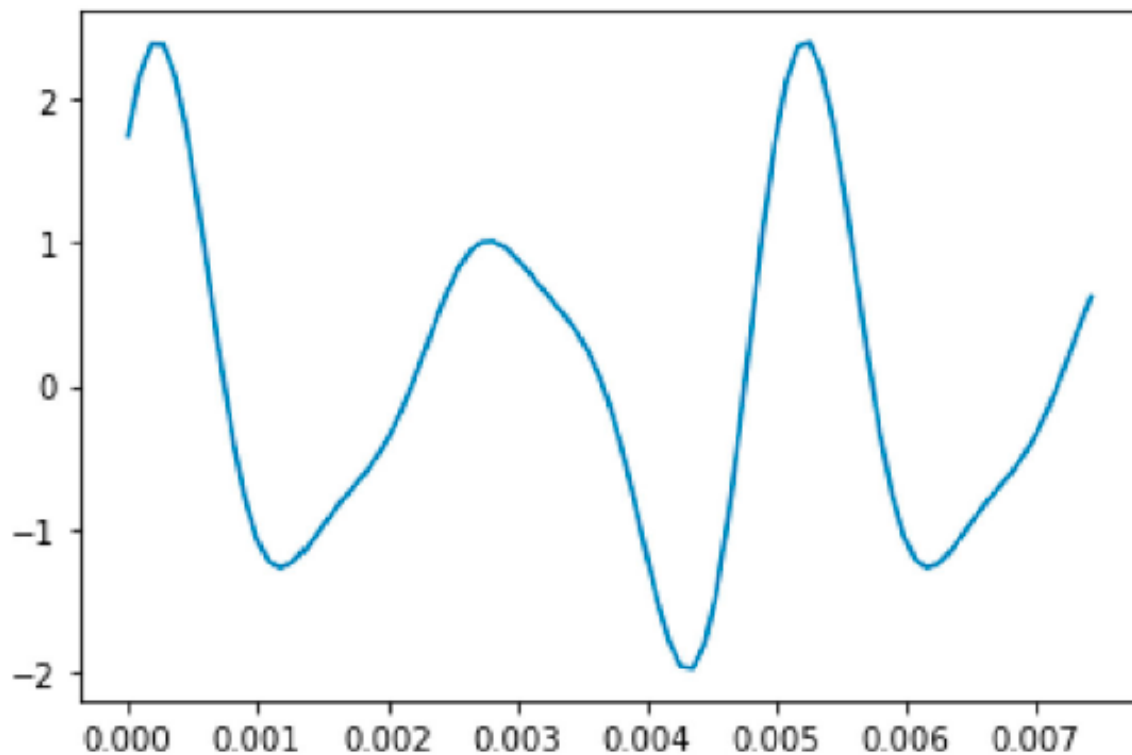


Рис. 9: Суммированный сигнал

После этого было получено аудио из данного сигнала:

```

1 wave2 = signal.make_wave(duration=0.5)
2 wave2.make_audio()
3

```

Листинг 12: Получение аудио

Получим спектр из полученного сигнала:

```

1 spectrum2 = wave2.make_spectrum()
2 spectrum2.plot(high=1000)
3

```

Листинг 13: Получение спектра

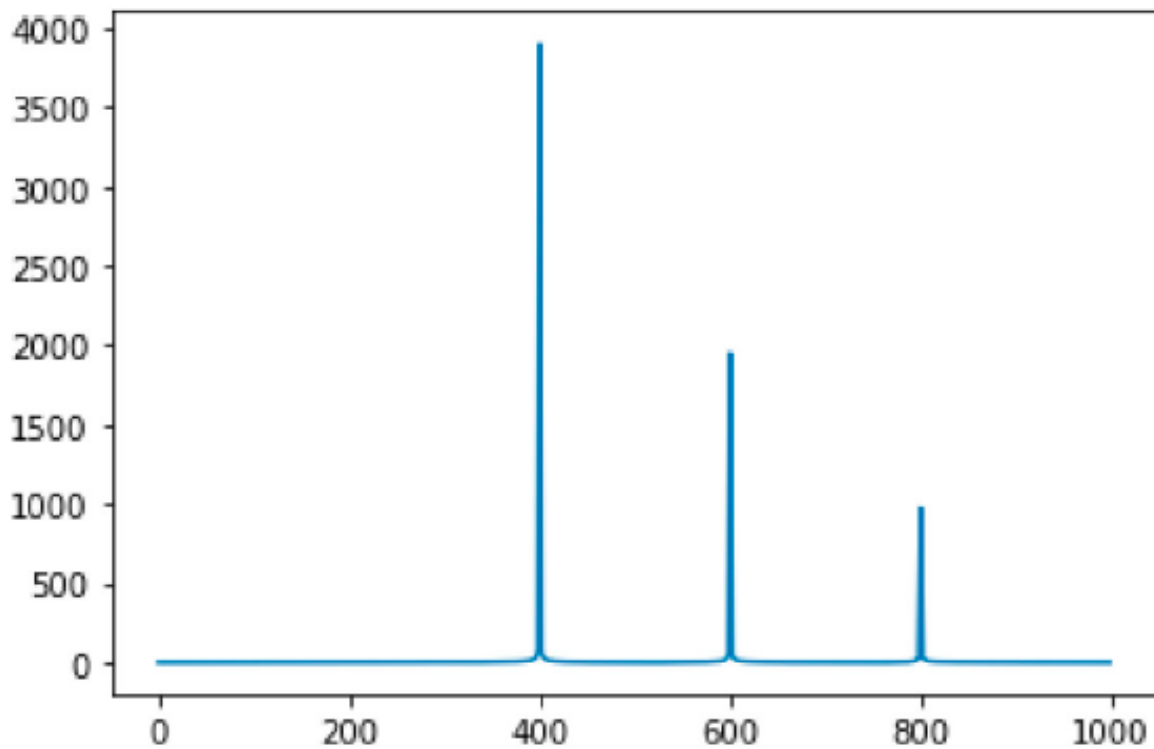


Рис. 10: Полученный спектр

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

```

1     signal += SinSignal(freq=133, amp=0.75)
2     wave3 = signal.make_wave()
3     wave3.make_audio()
4

```

Листинг 14: Получение спектра

Получим спектр из сигнала:

```

1     wave3.apodize()
2     spectrum3 = wave3.make_spectrum()
3     spectrum3.plot(high=1000)
4

```

Листинг 15: Полученный спектр

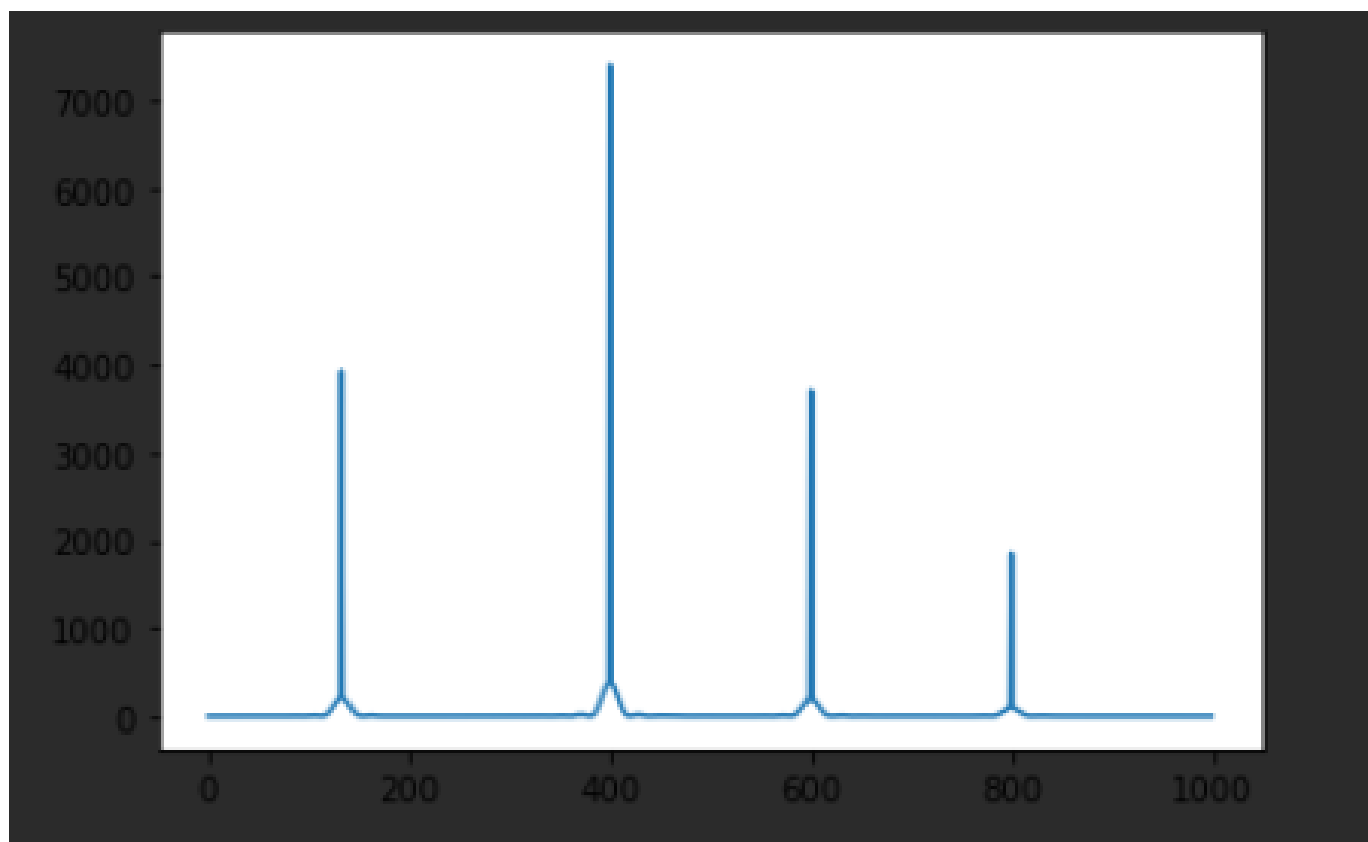


Рис. 11: Полученный спектр

По результатам выполнения 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`:

```
1     def stretch(wav, factor):
2         wav.ts *= factor
3         wav.framerate /= factor
4
```

Листинг 16: Функция `stretch`

Прочитаем ранее скачанный звуковой файл:

```
1     wave4 = read_wave('Sounds/105384__drzoom__hitting-pan-top-large.wav')
2     wave4.normalize()
3     wave4.make_audio()
4
```

Листинг 17: Чтение файла

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

```
1     stretch(wave4, 0.5)
2     wave4.make_audio()
3
```

Листинг 18: Вызов функции `stretch`

Спектр полученной дорожки:

```
1     wave4.plot()
2
```

Листинг 19: Спектр после вызова функции `stretch`

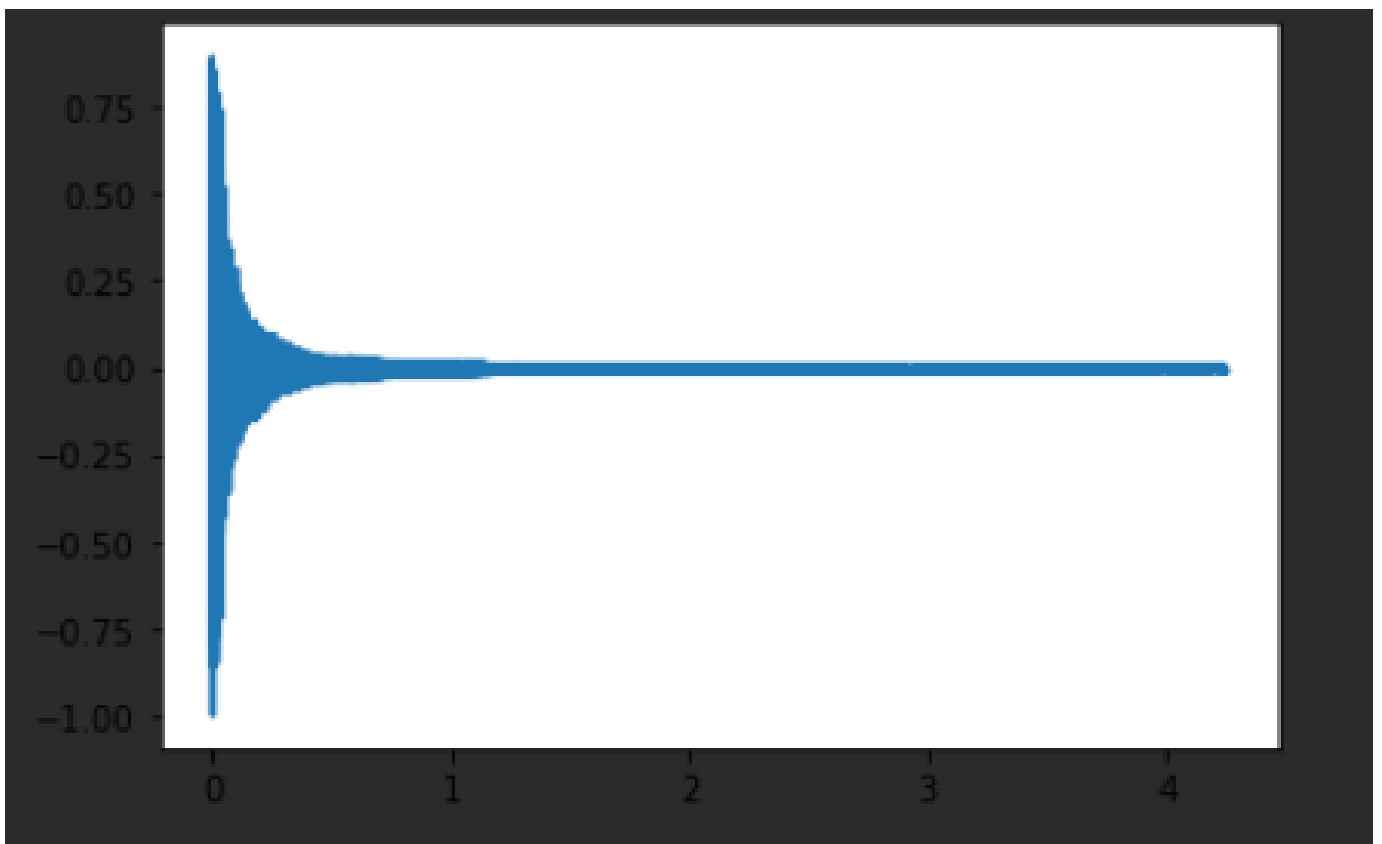


Рис. 12: Спектр аудиодорожки

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.