Лабораторная работа №2
Гармоники

Смирнов Никита

19 апреля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Упражнение 2.1

В данном упражнении нас просят открыть `chap02.ipynb`, прочитать пояснения и запустить примеры. Поэтому я просто изучил все примеры и комментарии к ним.

Глава 2

Упражнение 2.2

2.1 Написание класса и проверка сигнала

Для данного упражнения нужно написать класс под названием `SawtoothSignal` и за основу просят взять `Signal`, но в главе примеры на основе `Sinusoid`, поэтому я буду использовать `Sinusoid`.

```
import thinkdsp


class MySawtoothSignal(thinkdsp.Sinusoid):
    def evaluate(self, ts):
        cycles = self.freq * ts + self.offset / thinkdsp.PI2
        frac, _ = np.modf(cycles)
        ys = self.amp * frac
        return ys
```

Листинг 2.1: Класс SawtoothSignal

Убедимся, что все работает корректно.

```
sawtooth = MySawtoothSignal(200)
sawtooth_wave = sawtooth.make_wave(sawtooth.period*3,
    framerate=20000)
sawtooth_wave.plot()
```

Листинг 2.2: Визуализация пилообразного звука

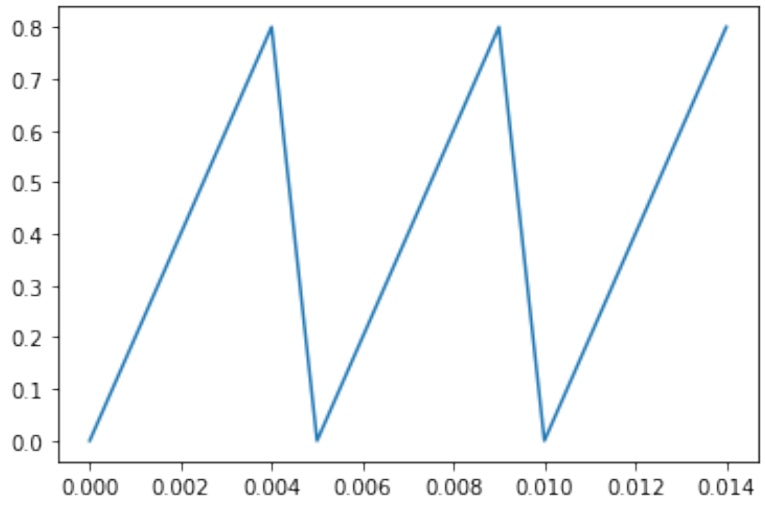

Рис. 2.1: Полученный пилообразный звук

2.2 Спектр звука

Теперь рассмотрим спектр нашего пилообразного звука.

```
spect = sawtooth_wave.make_spectrum()
spect.plot()
```

Листинг 2.3: Спектр звука

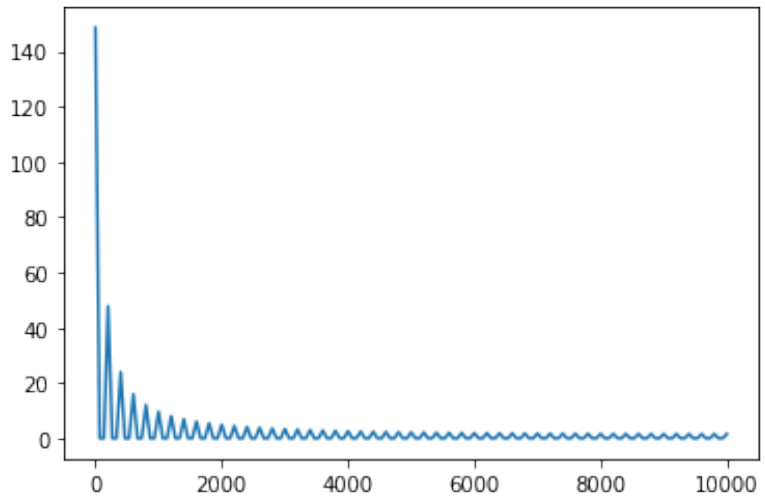

Рис. 2.2: Спектр сегмента звука

В сравнении с треугольной волной пилообразная форма уменьшается практически аналогично, но включает как четные, так и нечетные гармоники.

Глава 3

Упражнение 2.3

Нам нужно создать прямоугольный сигнал 1100 Гц и вычислить его спектр. Создадим прямоугольный сигнал 1100 Гц и вычислим его спектр.

```
signal = thinkdsp.SquareSignal(1100)
wave = signal.make_wave(duration=0.5, framerate=10000)
spect = wave.make_spectrum()
spect.plot()
```

Листинг 3.1: Создание прямоугольного сигнала

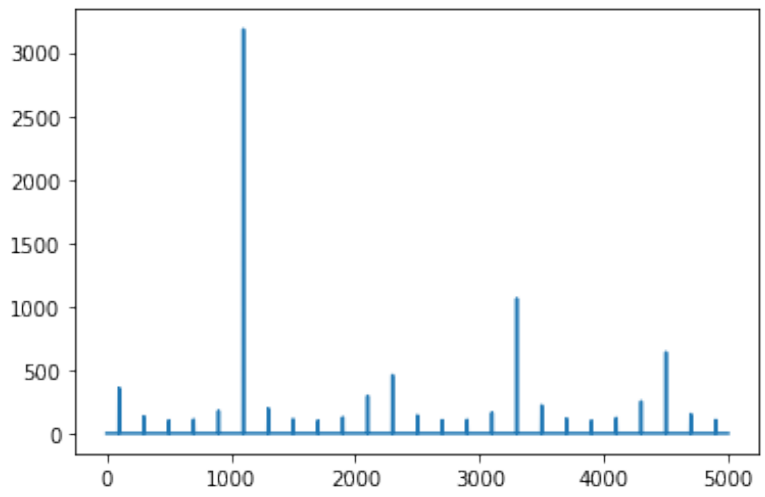

Рис. 3.1: Спектр сегмента звука

Основная и первая гармоника находятся в нужном месте, но вторая гармоника, которая должна быть 5500 Гц, смещается на 4500 Гц. Третья, которая должна быть 7700 Гц, находится на 2300 Гц и так далее.

Прослушаем полученный звук.

```
wave.make_audio()
```

Листинг 3.2: Воспроизведение прямоугольного сигнала

Когда мы слушаем полученную волну, можем слышать эти aliasing-гармоники, поскольку низкий тон имеет частоту 300 Гц. Создадим такой сигнал и прослушаем его. Разница присутствует и данные частоты можно расслышать.

```
signal = thinkdsp.SinSignal(300)
wave = signal.make_wave(duration=0.5, framerate=10000)
wave.make_audio()
```

Листинг 3.3: Создание и воспроизведение сигнала с пониженной частотой

Глава 4

Упражнение 2.4

4.1 Создание сигнала

Создадим треугольный сигнал с частотой 440 Гц и `wave` длительностью 0,01 секунды.

```
signal = thinkdsp.TriangleSignal(440)
wave = signal.make_wave(duration=0.01)
wave.plot()
```

Листинг 4.1: Создание треугольного сигнала

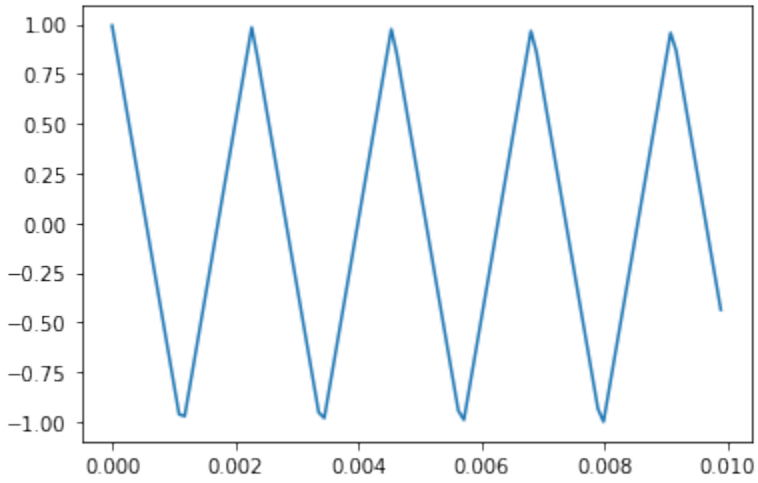

Рис. 4.1: Визуализация созданного сигнала

4.2 Нулевой компонент спектра

Первый элемент спектра - комплексное число, близкое к нулю. Если мы добавим в компонент нулевой частоты какое-то число, то это приведет к добавлению вертикального смещения спектра.

```
spectrum = wave.make_spectrum()
spectrum.hs[0]
```

Листинг 4.2: Вывод нулевого компонента

На выходе получили `(1.0436096431476471e-14+0j)`.

4.3 Изменение нулевого компонента

Теперь установим смещение равное 100 и увидим, что сигнал сместился по вертикали.

```
spectrum.hs[0] = 100
spectrum.make_wave().plot()
```

Листинг 4.3: Смещение спектра и его визуализация

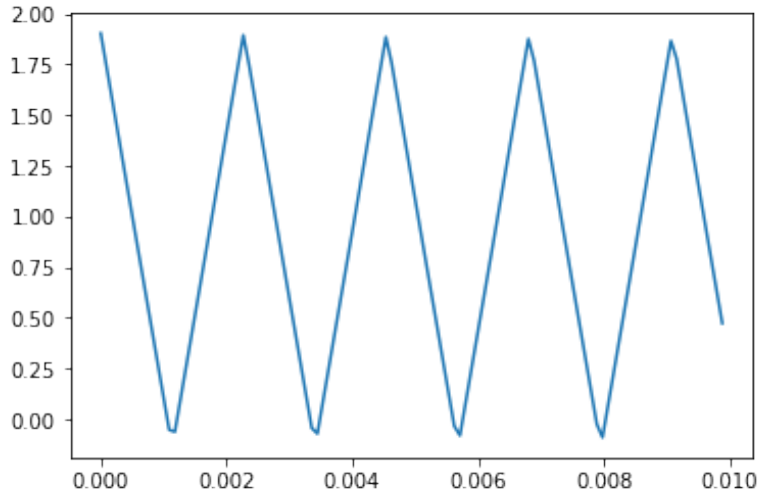

Рис. 4.2: Визуализация ускоренного звука

Глава 5

Упражнение 2.5

5.1 Создание функции

Напишем функцию, которая принимает `Spectrum` в качестве парметра и иизменяет его делением каждого элемента `hs` на соответсвующую частоту из `fs`.

```
def modify(spectrum):
  for it in range (1 , len(spectrum.hs)):
    spectrum.hs[it] /= spectrum.fs[it]
  spectrum.hs[0] = 0
```

Листинг 5.1: Создание функции

5.2 Проверка работоспособности функции

Создадим прямоугольный сигнал и прослушаем его.

```
wave = thinkdsp.SquareSignal(freq=440).make_wave(duration=0.5)
wave.make_audio()
```

Листинг 5.2: Создание сигнала и его воспроизведение

Теперь выведем только что созданный спектр, а также изменим его при помощи нашей функции и посмотрим на результат.

```
high = 10000
spectrum = wave.make_spectrum()
spectrum.plot(high=high, color='red')
modify(spectrum)
spectrum.scale(440)
```

```
spectrum.plot(high=high)
```

Листинг 5.3: Сравнение спектров

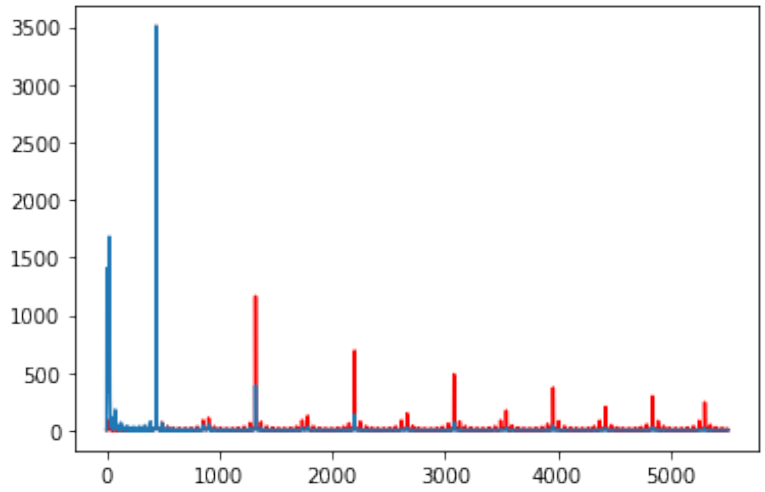

Рис. 5.1: Сравнение спектров

Фильтр подавляет гармоники, поэтому он действует как фильтр низких частот. Звук слышется почти как синусоида.

```
filtered = spectrum.make_wave()
filtered.make_audio()
```

Листинг 5.4: Воспроизведение отфильтрованного звука

Глава 6

Упражнение 2.6

Создадим пилообразный сигнал и выведем его спектр.

```
freq = 500
signal = thinkdsp.SawtoothSignal(freq=freq)
wave = signal.make_wave(duration=0.5, framerate=30000)
wave.make_audio()
spectrum = wave.make_spectrum()
spectrum.plot()
```

Листинг 6.1: Создание сигнала и визуализация его спектра

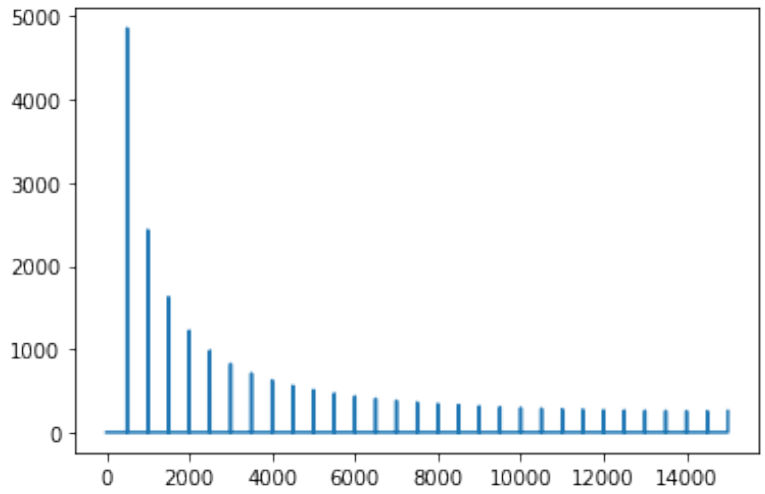

Рис. 6.1: Спектр сигнала

На рисунке видно, что гармоники уменьшаются как $1/f^2$.

```
spectrum.plot(color='red')
filter_spectrum(spectrum)
spectrum.scale(freq)
spectrum.plot()
```

Листинг 6.2: Сравнение гармоник

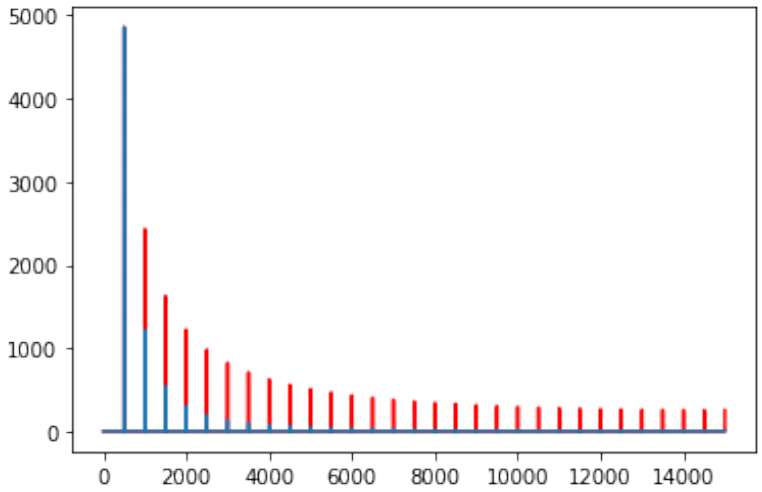

Рис. 6.2: Спектр сигнала

Теперь гармоника схожа с синусоидой.

```
wave.segment(duration=0.01).plot()
```

Листинг 6.3: Сегмент звука

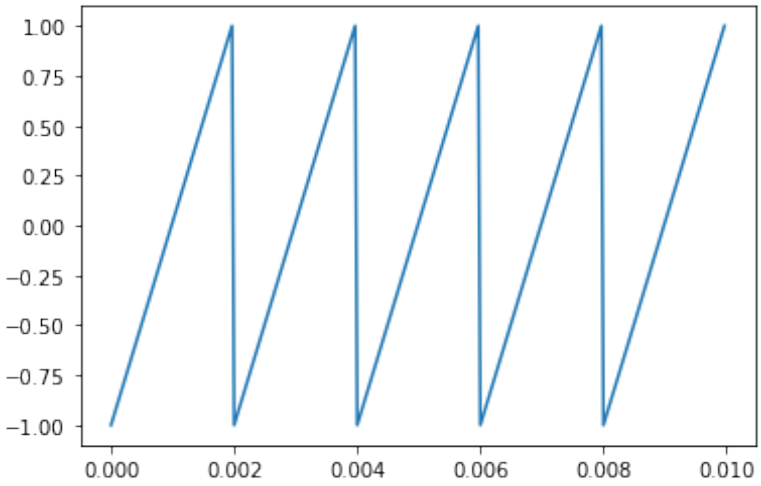

Рис. 6.3: Сравнение спектров

Глава 7

Выводы

Во время выполнения лабораторной работы получены навыки работы с новыми сигналами и их гармониками, а именно с треугольными, прямоугольными и пилообразными сигналами. Также рассмотрено одно из наиболее важных явлений в цифровой обработке сигналов - биения.