Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Гармоники

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 chap02

Для выполнения первого пункта необходимо изучить и выполнить примеры из файла chap02.ipynb. Запустим все примеры из этого файла.

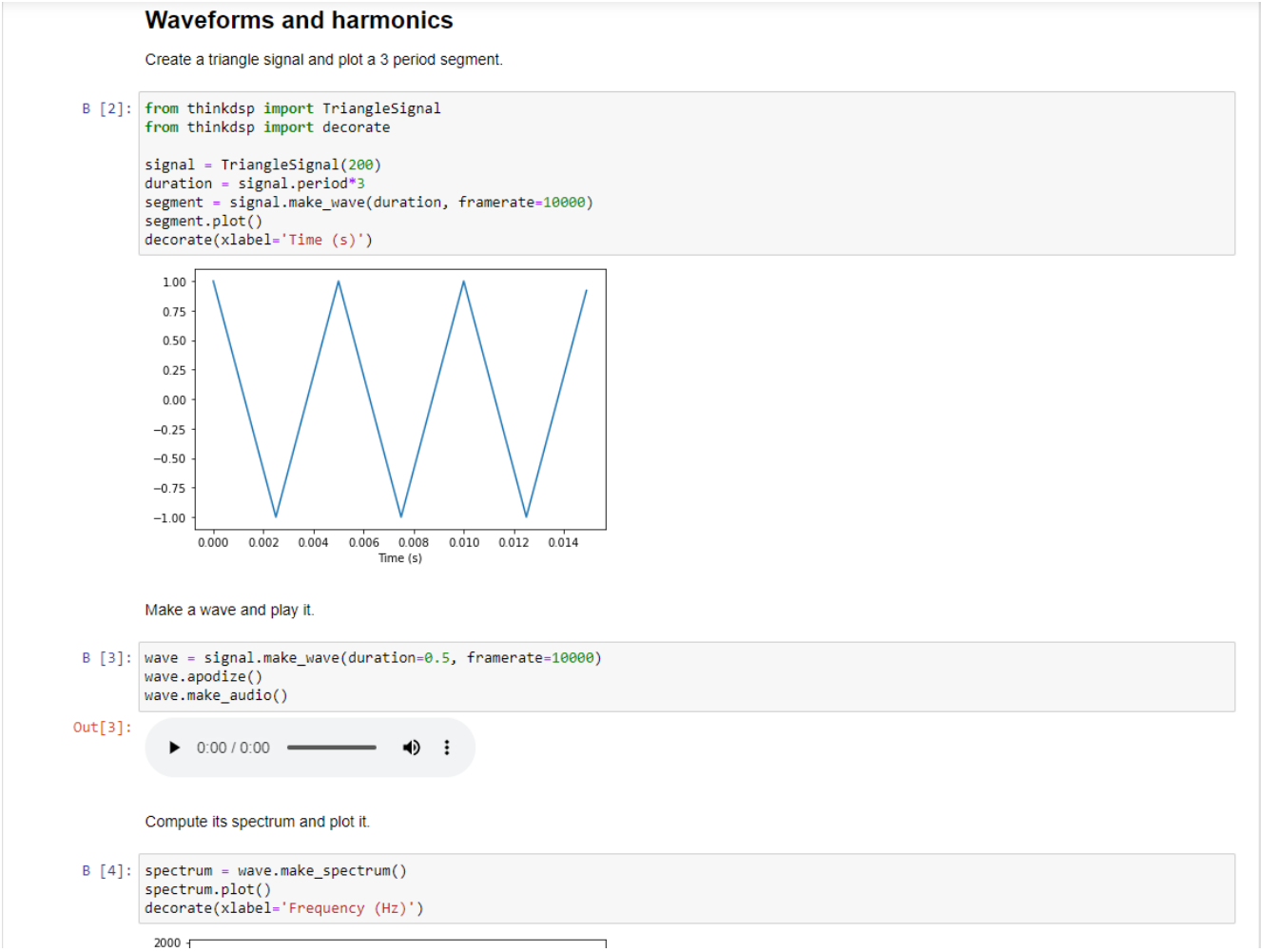
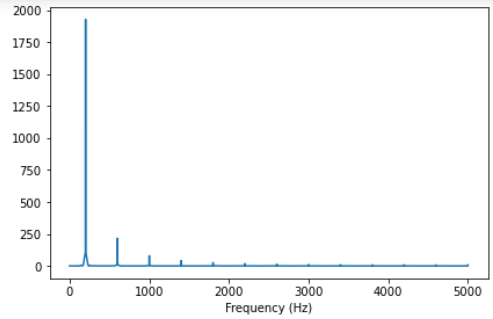


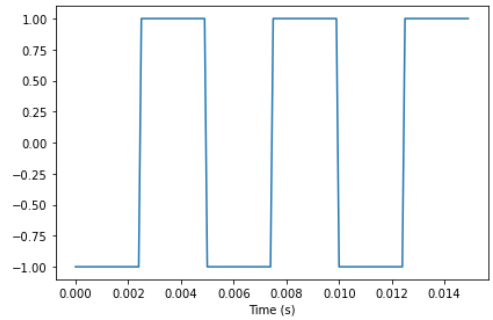
Рис. 1: Изучение и проверка примеров из файла (1)



Make a square signal and plot a 3 period segment.

```
B [5]: from thinkdsp import SquareSignal

signal = SquareSignal(200)
duration = signal.period*3
segment = signal.make_wave(duration, framerate=10000)
segment.plot()
decorate(xlabel='Time (s)')
```



Make a wave and play it.

```
B [6]: wave = signal.make_wave(duration=0.5, framerate=10000)
```

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

2 Упражнение №2: SawtoothSignal

Во втором упражнении необходимо написать класс SawtoothSignal, расширяющий signal и предоставляющий evaluate для оценки пилообразного сигнала

Подключение зависимостей и написание класса SawtoothSignal:

```
from thinkdsp import Sinusoid
from thinkdsp import normalize, unbias
import numpy as np

class SawtoothSignal(Sinusoid):

    def evaluate(self, ts):
        cycles = self.freq * ts + self.offset / np.pi / 2
        frac, _ = np.modf(cycles)
        ys = normalize(unbias(frac), self.amp)
        return ys


```

Листинг 1: Класс SawtoothSignal

Проверим работу класса. Выведем график сигнала и послушаем получившийся wav:

```
saw_wav = SawtoothSignal().make_wave(duration=0.5, framerate=40000)
saw_wav.segment(start=0, duration=0.01).plot()
saw_wav.make_audio()
```

Листинг 2: Вывод графика и создание wav

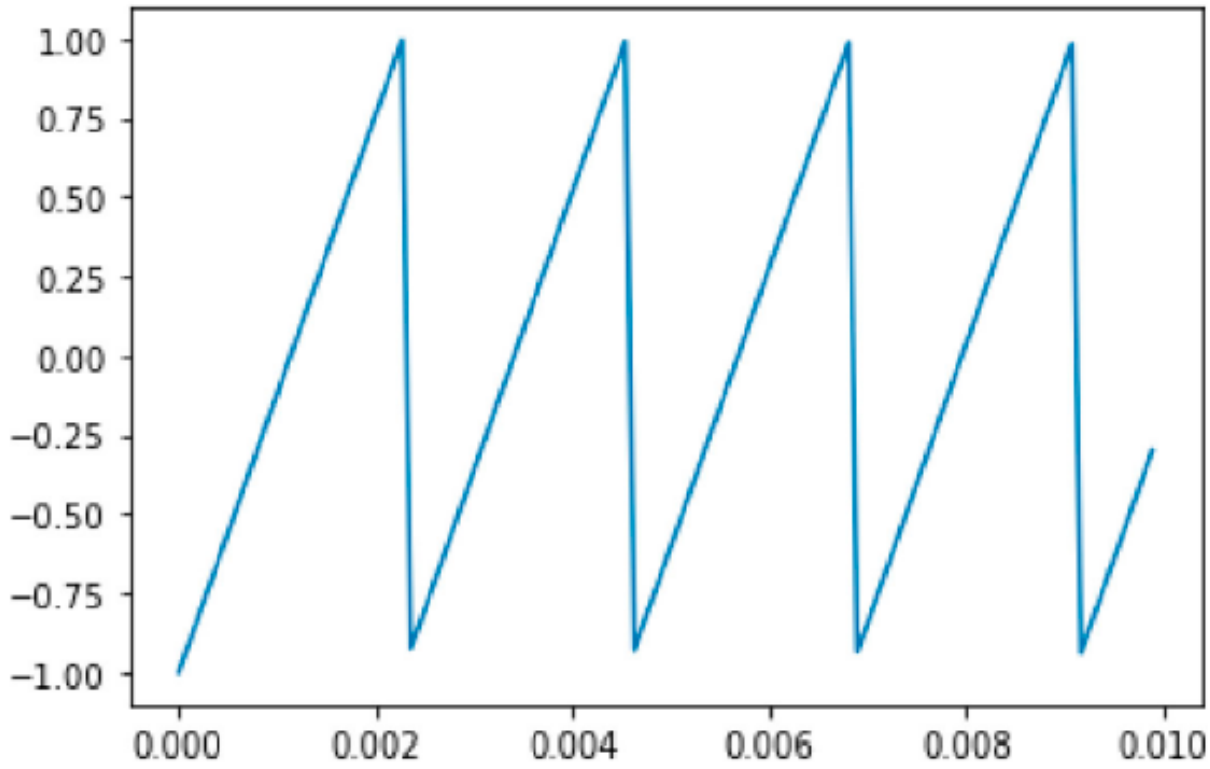


Рис. 3: График sawtooth

Выведем спектр:

```
saw_wav.make_spectrum().plot()
```

Листинг 3: Вывод спектра sawtooth

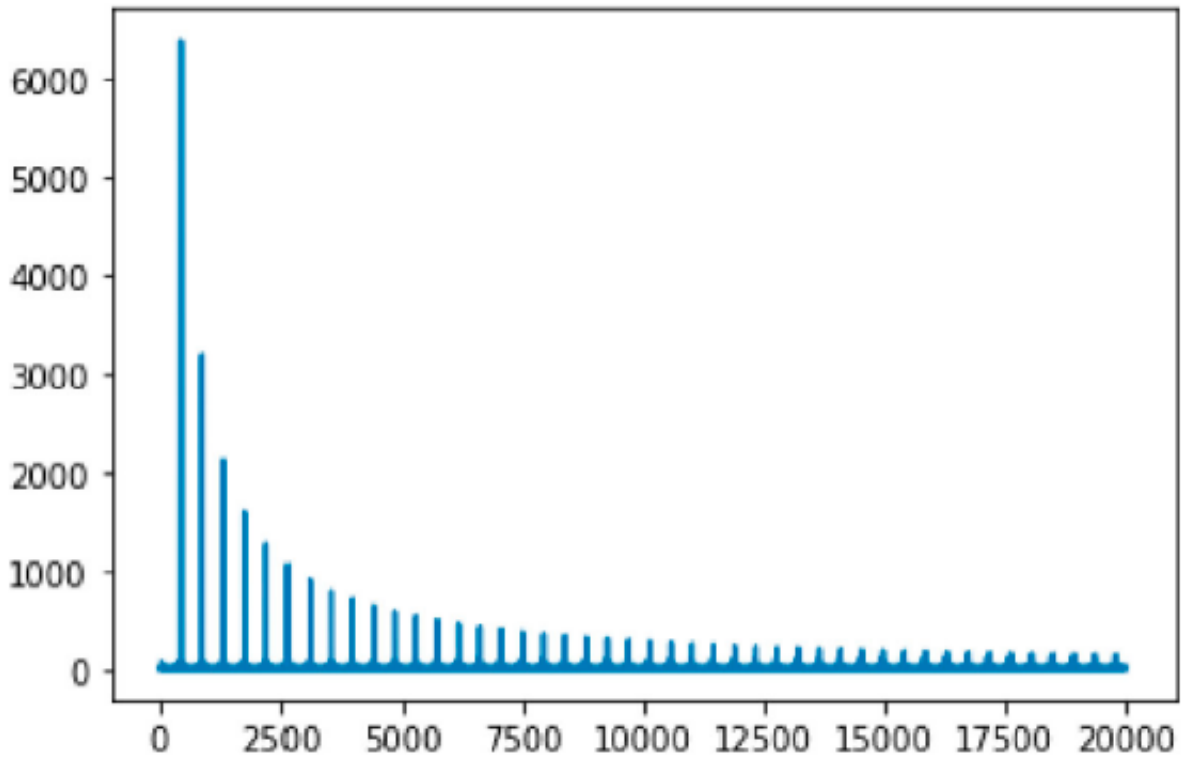


Рис. 4: Спектр sawtooth

Сравним наш пилообразный сигнал с прямоугольным

```
    from thinkdsp import SquareSignal

    saw_wav.make_spectrum().plot(color='gray')
    square_wav = SquareSignal(amp=0.5).make_wave(duration=0.5, framerate
=40000)
    square_wav.make_spectrum().plot()

```

Листинг 4: Изображение спектра

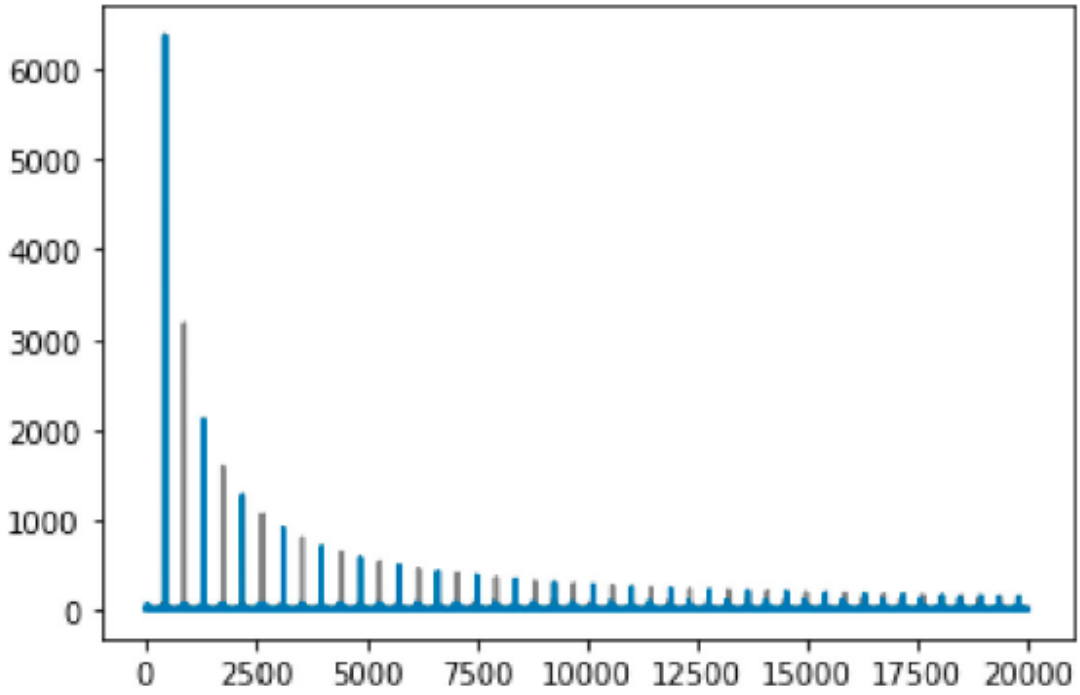


Рис. 5: Прямоугольный и пилообразный спектр

Прослушаем прямоугольный сигнал:

```
square_wav.make_audio()
```

Листинг 5: Создание звука прямоугольного сигнала

Сравним наш пилообразный сигнал с треугольным:

```
from thinkdsp import TriangleSignal

saw_wav.make_spectrum().plot(color='gray')
triangle_wav = TriangleSignal(amp=0.79).make_wave(duration=0.5, framerate
=40000)
triangle_wav.make_spectrum().plot()
```

Листинг 6: Фильтрация спектра

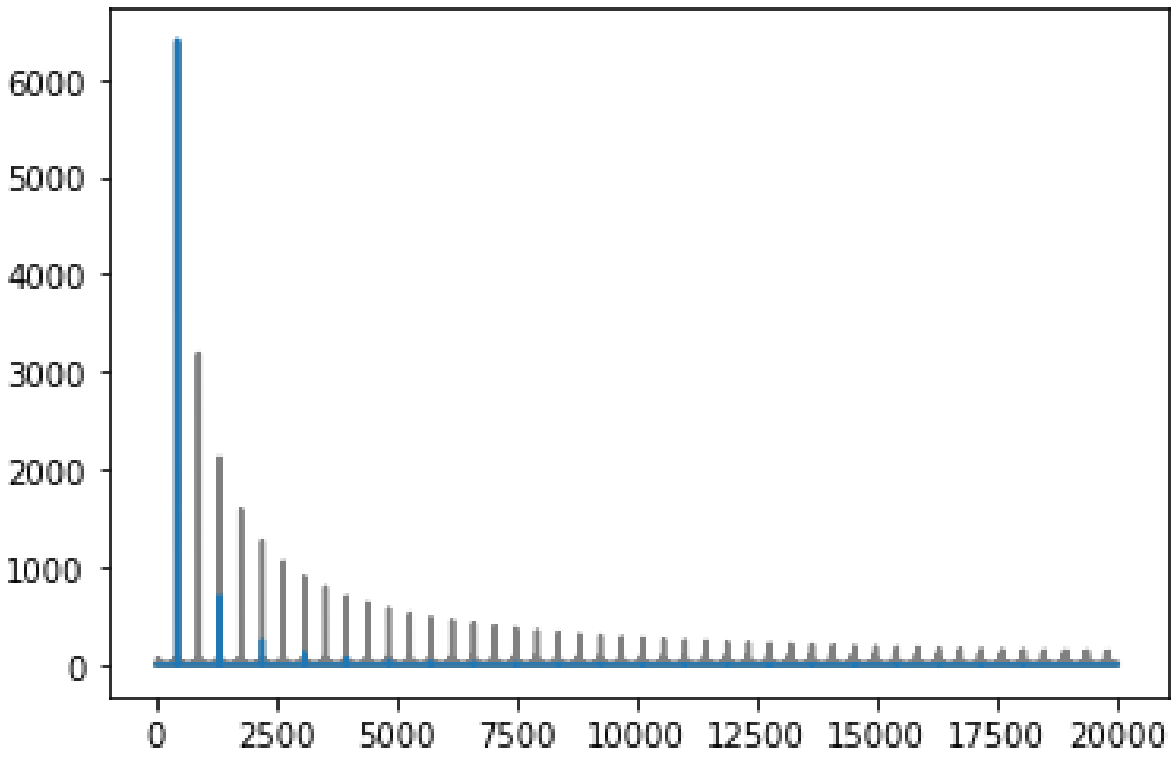


Рис. 6: Отфильтрованный спектр

Прослушаем треугольный сигнал:

```
triangle_wav.make_audio()
```

Листинг 7: Перевод в аудио

По результатам выполнения данного упражнения можно сделать вывод, что по сравнению с прямоугольным сигналом, пилообразный уменьшается аналогично, но включает четные и нечетные гармоники. Гармоники треугольного сигнала уменьшаются пропорционально $1/f^2$, а пилообразный уменьшается $1/f$.

3 Упражнение №3: Квадратный сигнал

В третьем упражнении нам необходимо создать квадратный сигнал частотой 1100 Гц и вычислить wave с выборками 10000 кадров в секунду. Построив спектр, убедимся, что большинство гармоник ”завернуты”из-за биения.

Создадим прямоугольный сигнал и выведем его спектр:

```
square_wav1 = SquareSignal(1500).make_wave(duration=0.5, framerate=10000)
square_wav1.make_spectrum().plot()

```

Листинг 8: Создание сигнала и вывод его спектра

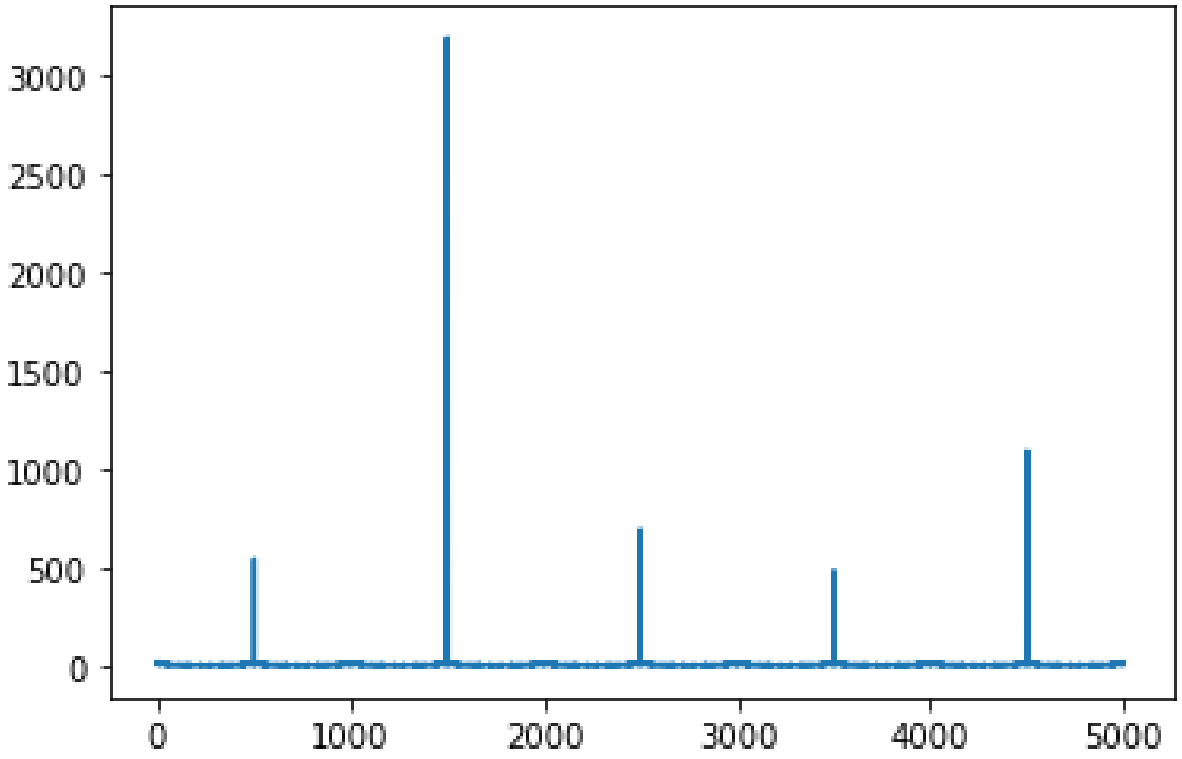


Рис. 7: Спектр прямоугольного сигнала

После этого было получено аудио из данного сигнала:

```
square_wav1.make_audio()

```

Листинг 9: Получение аудио

Для сравнения создадим синусоидальный сигнал частотой 500 Гц и выведем его спектр:

```
from thinkdsp import SinSignal

sin_wav = SinSignal(500).make_wave(duration=0.5, framerate=10000)
sin_wav.make_spectrum()

```

Листинг 10: Создание синусоидального сигнала и получение его спектра

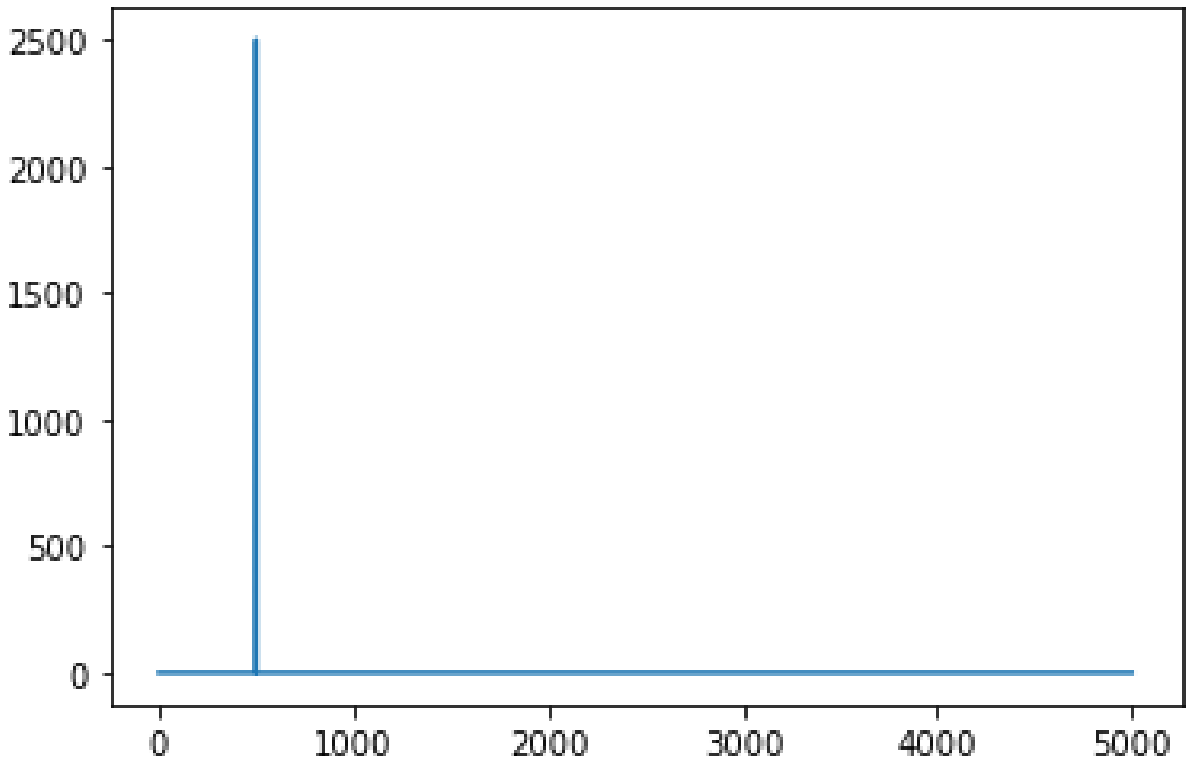


Рис. 8: Полученный спектр синусоидального сигнала

Получим аудио из данного сигнала:

```
sin_wav.make_audio()

```

Листинг 11: Получение аудио

По результатам выполнения данного упражнения, сравнивая два сигнала, можно сделать вывод, что они похожи. Прослушивая первый сигнал и сравнивая со вторым, слышно что основной тон в первом сигнале, который мы слышим это ”alias”частоты 500 Гц.

4 Упражнение №4: Нулевая частота

В четвертом упражнении необходимо взять объект `spectrum` и распечатать несколько первых значений `spectrum.hs`. Необходимо также убедиться, что они начинаются с нуля, т.е. `spectrum.hs[0]` - амплитуда компонента с частотой 0.

1. Создать треугольный сигнал с частотой 440Гц и `wave` длительностью 0.01с после чего вывести график сигнала.
2. Создать объект `spectrum` и вывести `spectrum.hs[0]`. Определить какая амплитуда и фаза у компонента.
3. Установить `spectrum.hs[0]` = 100 и определить как данная операция влияет на сигнал.

Создадим треугольный сигнал и выведем его график:

```
triangle = TriangleSignal().make_wave(duration=0.01)
triangle.plot()

```

Листинг 12: Создание и вывод на экран треугольного сигнала

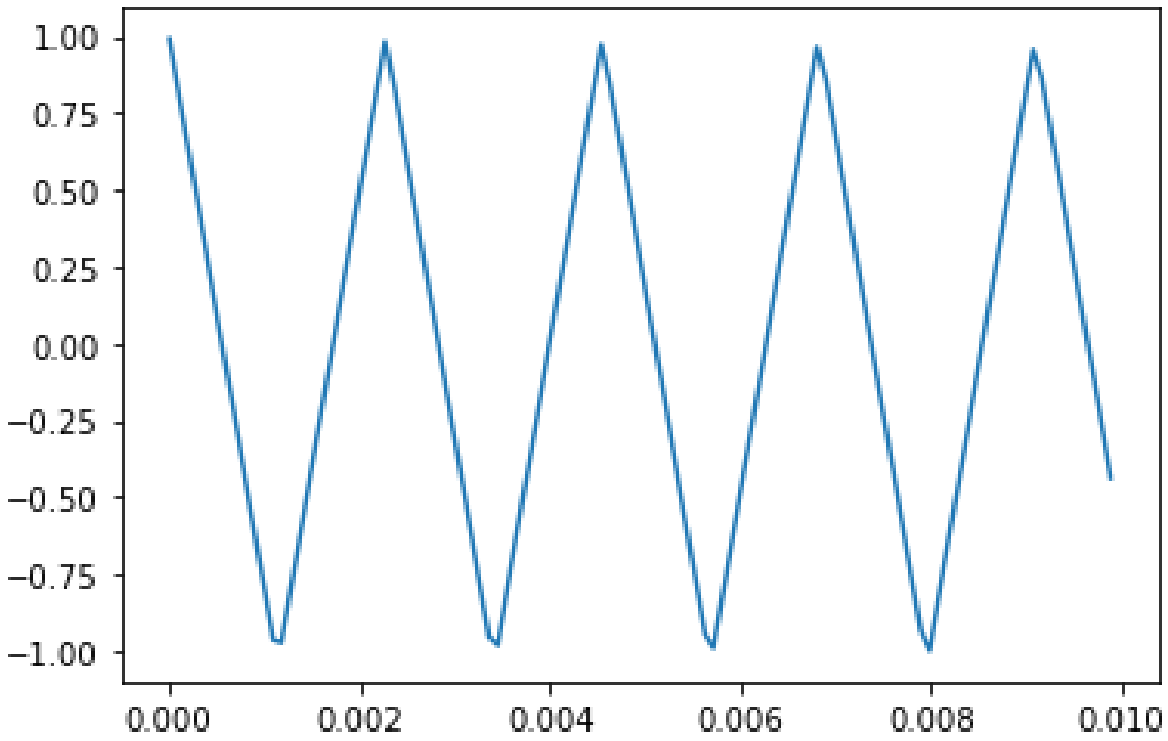


Рис. 9: График треугольного сигнала

Создадим спектр сигнала и выведем spectrum.hs[0]

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

Листинг 13: Создание спектра и вывод spectrum.hs

Получившийся элемент спектра `(1.0436096431476471e-14+0j)` комплексное число, близкое к нулю.

Установим `spectrum.hs[0]` = 100 и выведем сигнал на экран:

```
spectrum.hs[0] = 100
triangle.plot(color='gray')
spectrum.make_wave().plot()
```

Листинг 14: Изменение spectrum.hs и вывод сигналов на экран

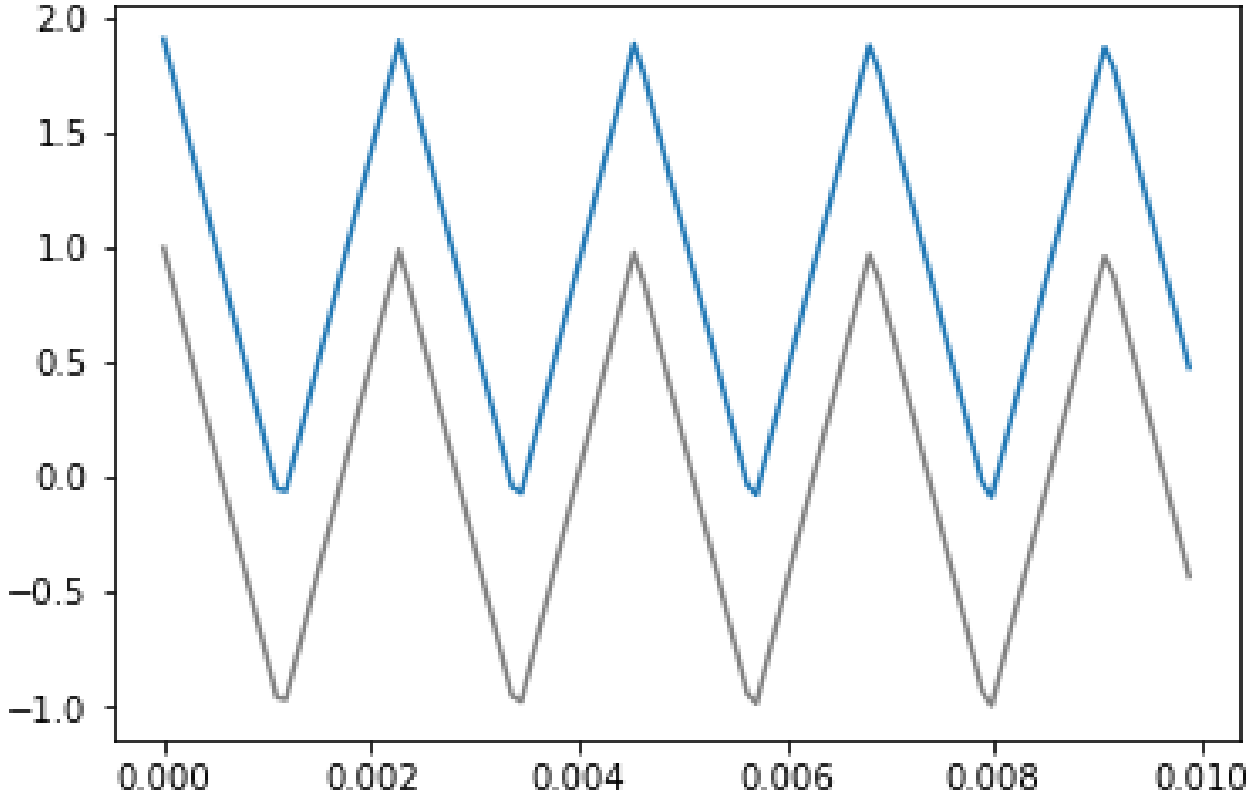


Рис. 10: График сигнала после изменения

По результатам выполнения данного упражнения можно сделать вывод, что два сигнала не отличаются друг от друга, кроме уровня громкости.

5 Упражнение №5: Фильтрация спектра

В пятом упражнении необходимо написать функцию, принимающую `spectrum` как параметр и изменяющую его делением каждого элемента `hs` на соответствующую частоту из `fs`. Необходимо также проверить данную функцию на прямоугольном, треугольном или пилообразном сигнале.

1. Вычислить спектр и вывести его.
2. Изменить спектр, используя функцию, и вывести его.
3. Использовать `spectrum.make_wave`, чтобы сделать `wave` из измененного спектра и прослушать его.

Создадим функцию filter_spectrum:

```
def filter_spectrum(s):
    s.hs[1:] /= s.fs[1:]
    s.hs[0] = 0

```

Листинг 15: Функция filter_spectrum

Создадим прямоугольный сигнал и переведем его в аудио:

```
square_wav2 = SquareSignal(freq=440).make_wave(duration=0.5)
square_wav2.make_audio()

```

Листинг 16: Создание прямоугольного сигнала и создание аудио

Выведем на экран исходный спектр. Изменим спектр с помощью нашей функции. Выведем измененный спектр.

```
spectrum_square = square_wav2.make_spectrum()
spectrum_square.plot(high=10000, color='gray')
filter_spectrum(spectrum_square)
spectrum_square.scale(440)
spectrum_square.plot(high=10000)

```

Листинг 17: Изменение и вывод спектров на экран

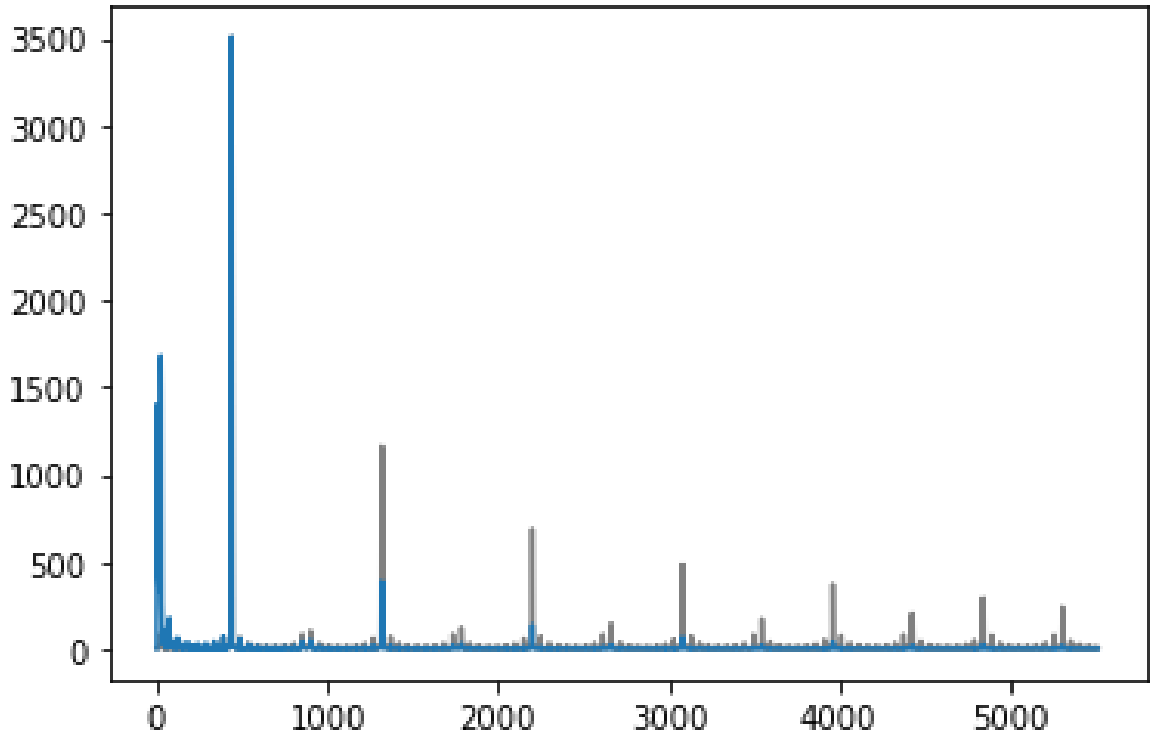


Рис. 11: Исходный и измененный спектр

Переведем отфильтрованный спектр в аудио и прослушаем.

По результатам выполнения данного упражнения можно сделать вывод, что полученный сигнал тише исходного и высокие частоты были отфильтрованы.

6 Упражнение №6: Создание сигнала

В шестом упражнении необходимо определить, можно ли создать сигнал, состоящий из четных и нечетных гармоник, спадающих пропорционально $1/f^2$.

Создадим пилообразный сигнал и прослушаем его:

```
freq = 500
signal = SawtoothSignal(freq=freq)
sawtooth_wave = signal.make_wave(duration=0.5, framerate=20000)
sawtooth_wave.make_audio()

```

Листинг 18: Создание пилообразного сигнала

Выведем его спектр:

```
sawtooth_spectrum = sawtooth_wave.make_spectrum()
```

```
sawtooth_spectrum.plot()
```

Листинг 19: Спектр пилообразного сигнала

Необходимо отфильтровать спектр с помощью функции написанной раннее. После фильтрации выведем спектр на экран

```
sawtooth_spectrum.plot(color='gray')
filter_spectrum(sawtooth_spectrum)
sawtooth_spectrum.scale(freq)
sawtooth_spectrum.plot()
```

Листинг 20: Фильтрация и вывод спектра

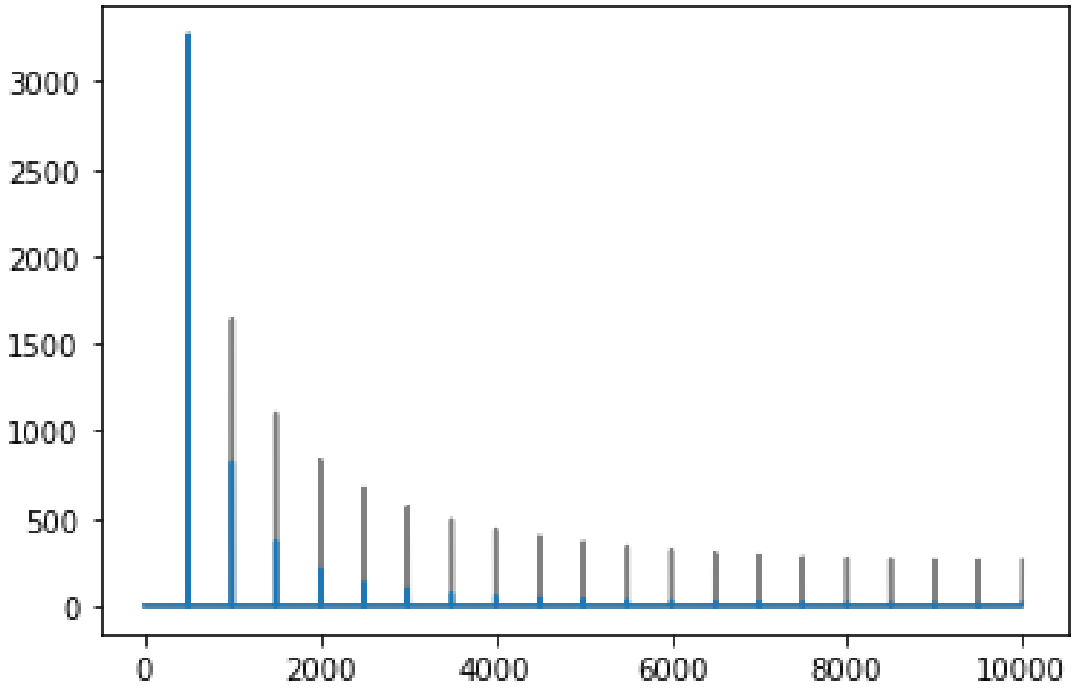


Рис. 12: Исходный и отфильтрованный спектр

Далее переведем отфильтрованный спектр в аудио и прослушаем.

После этого выделим сегмент из полученного сигнала отфильтрованного спектра длительностью 0.01 с. Выведем сигнал сегмента на экран.

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

Листинг 21: Фильтрация и вывод спектра

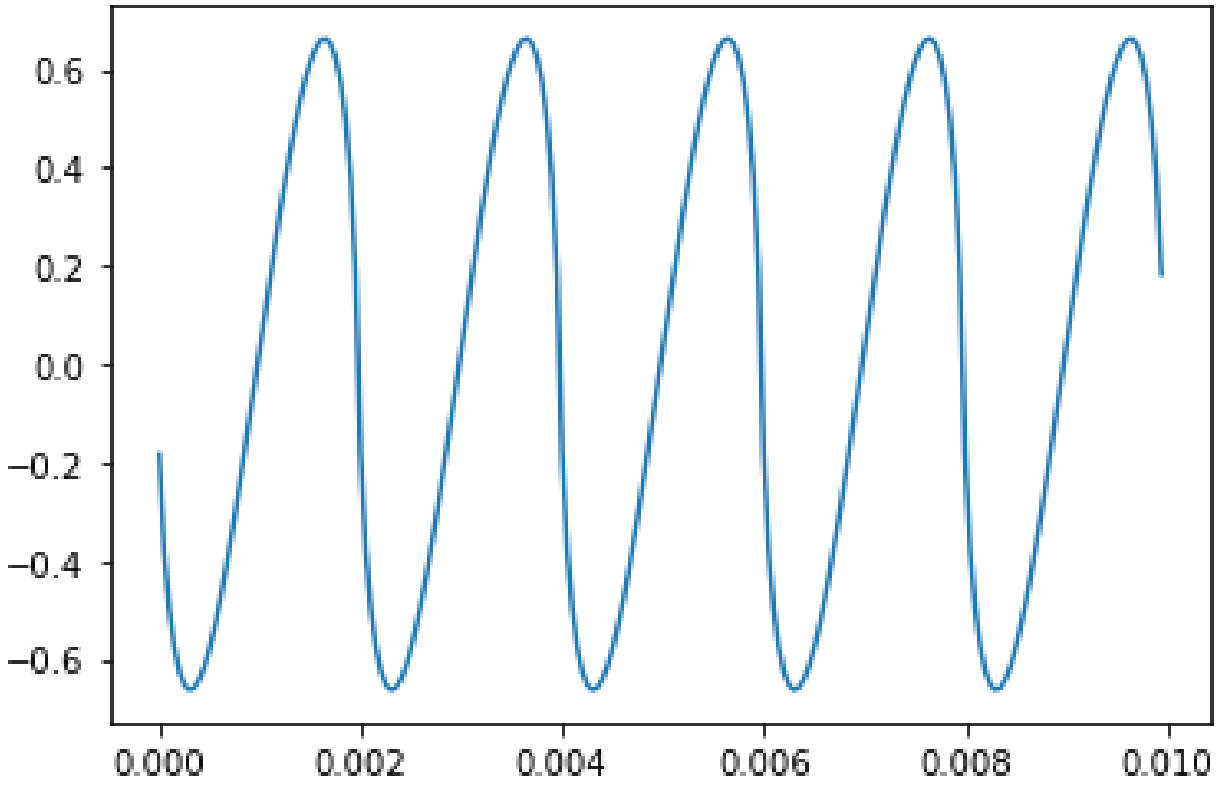


Рис. 13: Сигнал сегмента

Данный сигнал не особо похож на какую либо математическую функцию. Поэтому выполним другой подход. Сложим серию косинусоидальных сигналов с правильными частотами и амплитудами. Отобразим спектр полученного сигнала:

```
from thinkdsp import CosSignal

freqs = np.arange(500, 9500, 500)
amps = 1 / freqs**2
cos_signal = sum(CosSignal(freq, amp) for freq, amp in zip(freqs, amps))

cos_signal.make_wave().make_spectrum().plot()

```

Листинг 22: Получение сигнала

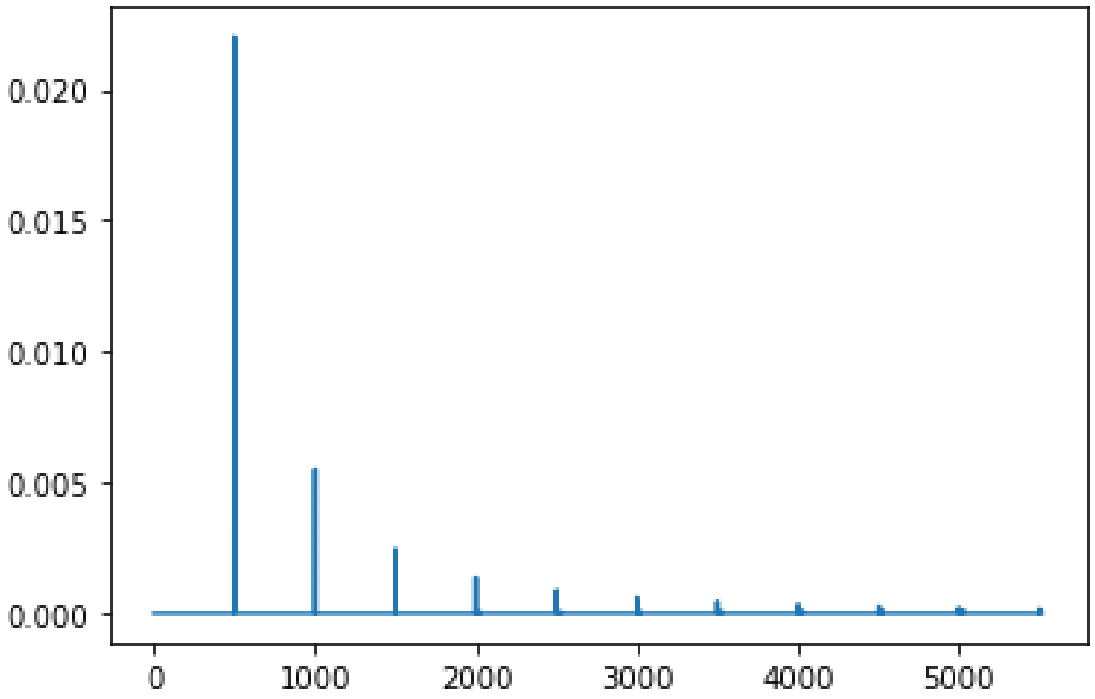


Рис. 14: Спектр косинусоидального сигнала

Прослушаем данный сигнал.

Выделим сегмент сигнала и выведем на экран.

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

Листинг 23: Получение сигнала

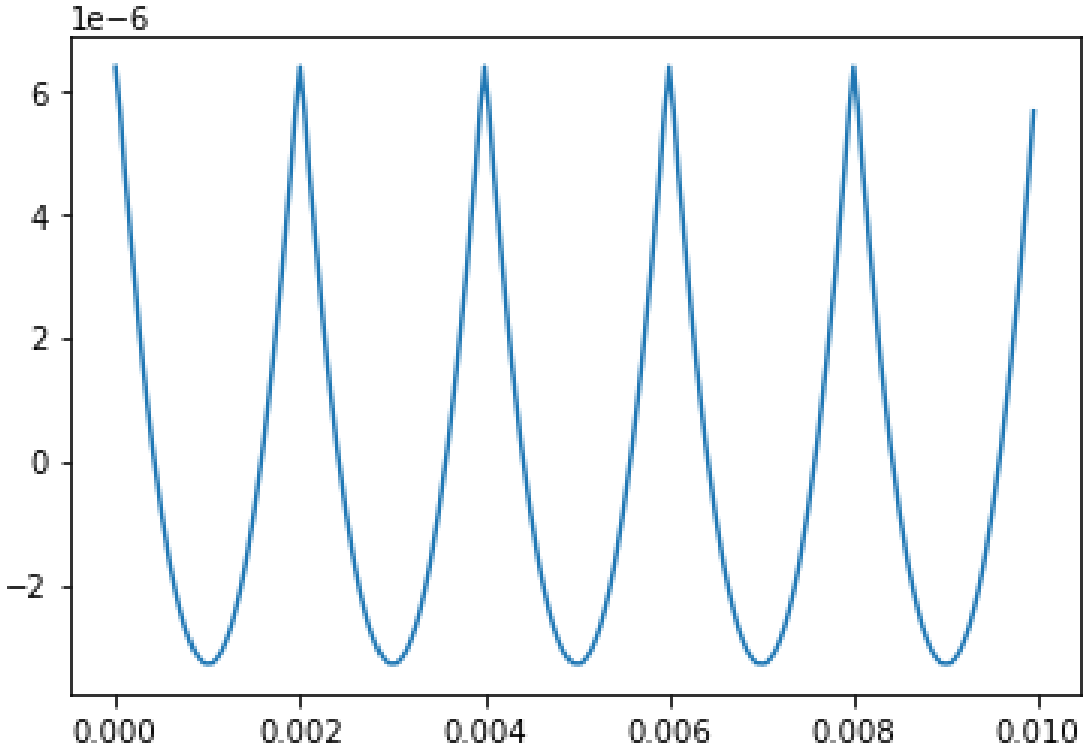


Рис. 15: Сегмент косинусоидального сигнала

Полученный сигнал похож на параболы. Это верно лишь отчасти. Такого же результата можно добиться используя `ParabolicSignal` из `thinkdsp.py`. Создадим новый сигнал на основе `ParabolicSignal`.

```
1        from thinkdsp import ParabolicSignal
2
3        par_wave = ParabolicSignal(freq=500).make_wave(duration=0.5, framerate
    =20000)
4        par_wave.make_audio()
5
```

Листинг 24: Получение параболического сигнала используя ParabolicSignal

Послушаем сигнал и посмотрим на его график.

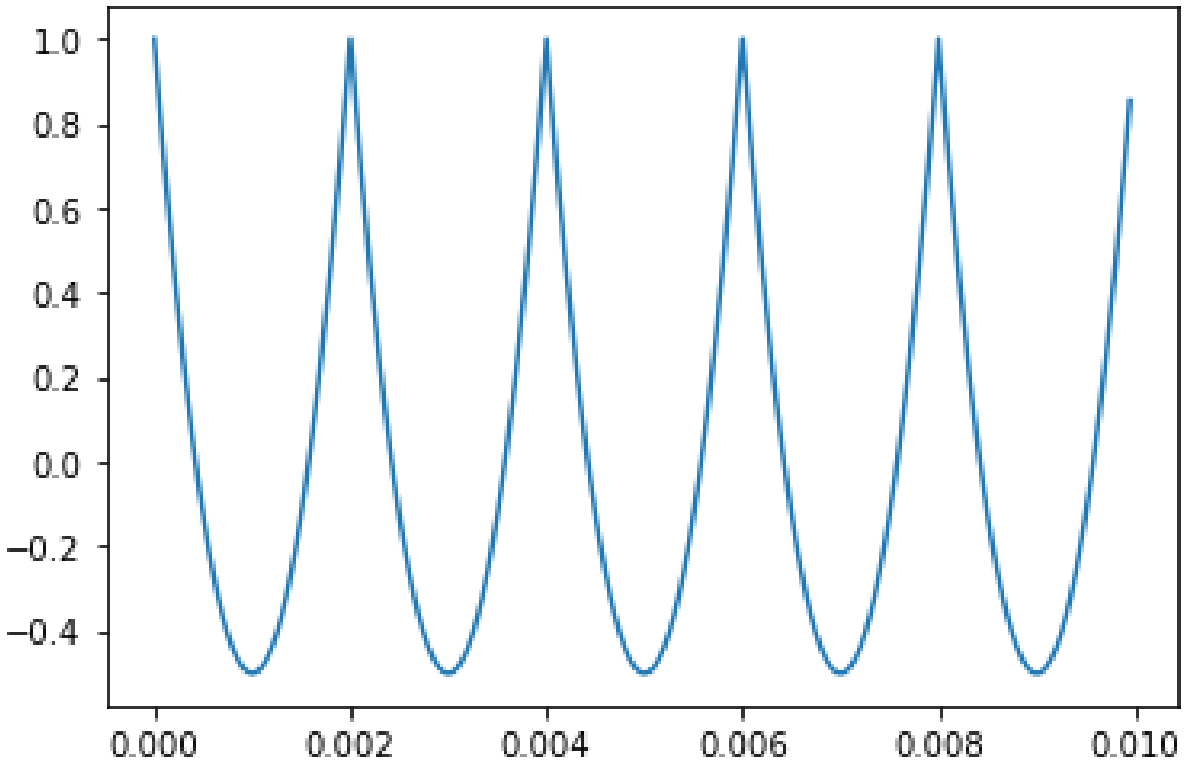


Рис. 16: Параболический сигнал

Также посмотрим на его спектр.

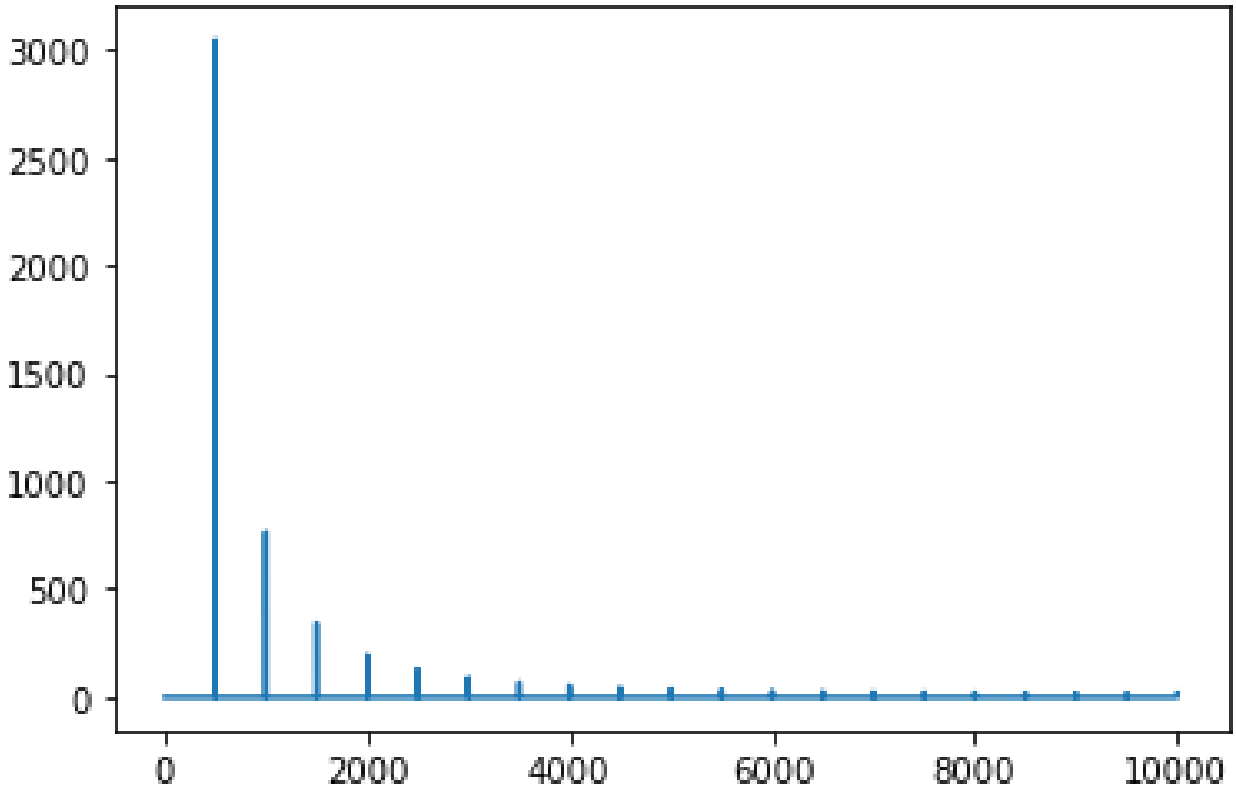


Рис. 17: Спектр параболического сигнала

По результатам выполнения данного упражнения можно сделать вывод, что два последних полученных сигнала идентичны.

Первый сигнал по звуку такой же как последние два, но график сигнала говорит об обратном.

7 Выводы

В результате выполнения данной лабораторной работы мы изучили как надо работать с гармониками, как их обрабатывать, изменять параметры и т.д. Кроме того была реализован и проверен класс для получения пилообразного сигнала. Была создана функция для изменения спектра путем деления каждого элемента hs на соответствующую частоту fs. Был выведен параболический сигнал с помощью сложения косинусоидальных сигналов.