Лабораторная работа 2

Чернышев Ярослав

30 мая 2021 г.

Оглавление

Глава 1

Задание 2.1

В данном задании от меня требуется изучить готовые примеры из chap02.ipynb, а также ознакомиться с результатами работы с ним.

```
[26]     display(wave.make_audio())
```

```
[27] from ipywidgets import interact, interactive, fixed
     import ipywidgets as widgets

     slider1 = widgets.FloatSlider(min=100, max=10000, value=100, step=100)
     slider2 = widgets.FloatSlider(min=5000, max=40000, value=10000, step=1000)
     interact(view_harmonics, freq=slider1, framerate=slider2);
```

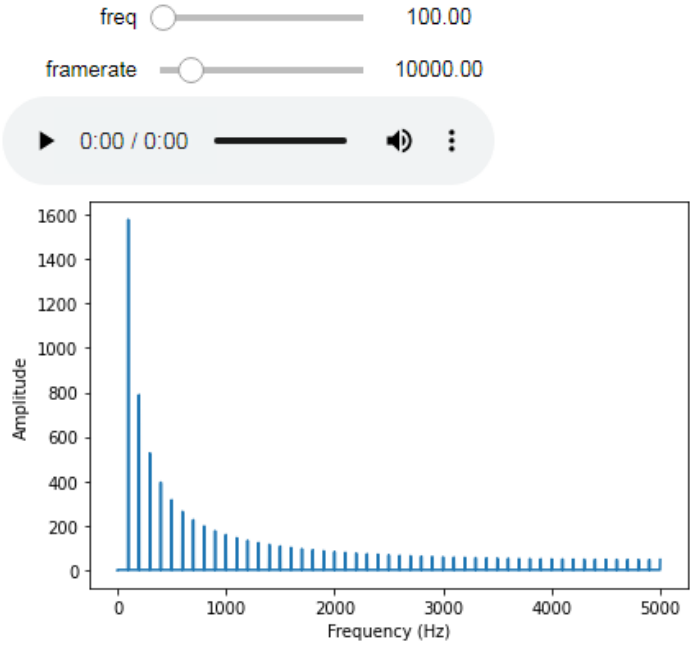

Рис. 1.1: Фрагмент работы

Глава 2

Задание 2.2

В этом задании от меня требуется поэксперементировать с пилообразным сигналом.

Первым делом, напишем соответствующий класс:

```
   from thinkdsp import Signal, Sinusoid, PI2
import numpy as np

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

```

Листинг 2.1: class Sawtooth

Теперь смотрим на волну и спектр получившегося сигнала:

```
from thinkdsp import decorate

saw = Sawtooth(100)
wave = saw.make_wave(saw.period*5, framerate=10000)
wave.plot()
decorate(xlabel='Time (s)')
```

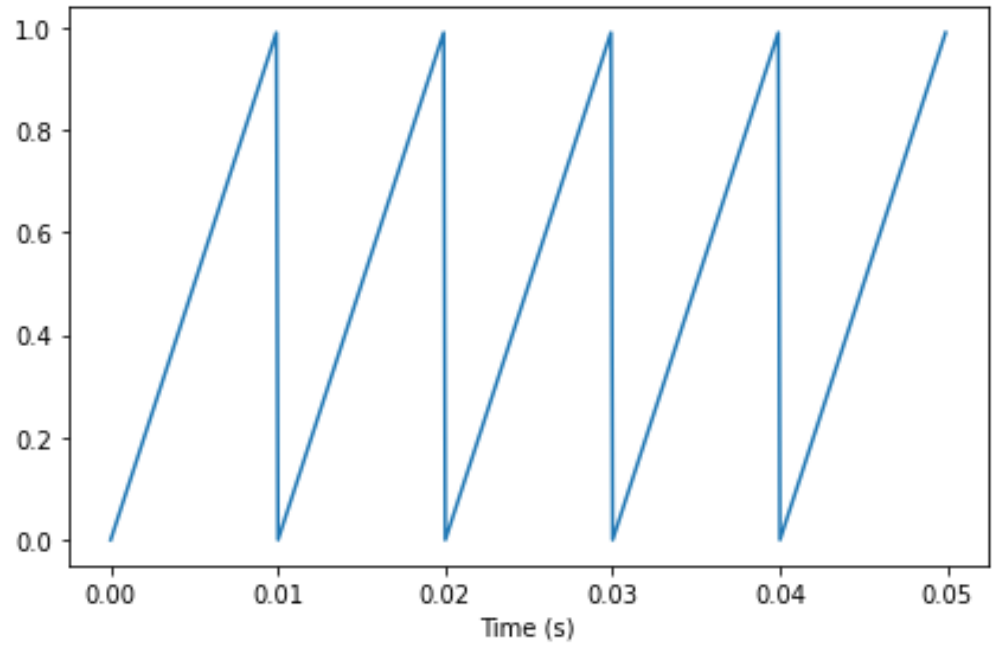

Рис. 2.1: Волны

```
wave.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)')
```

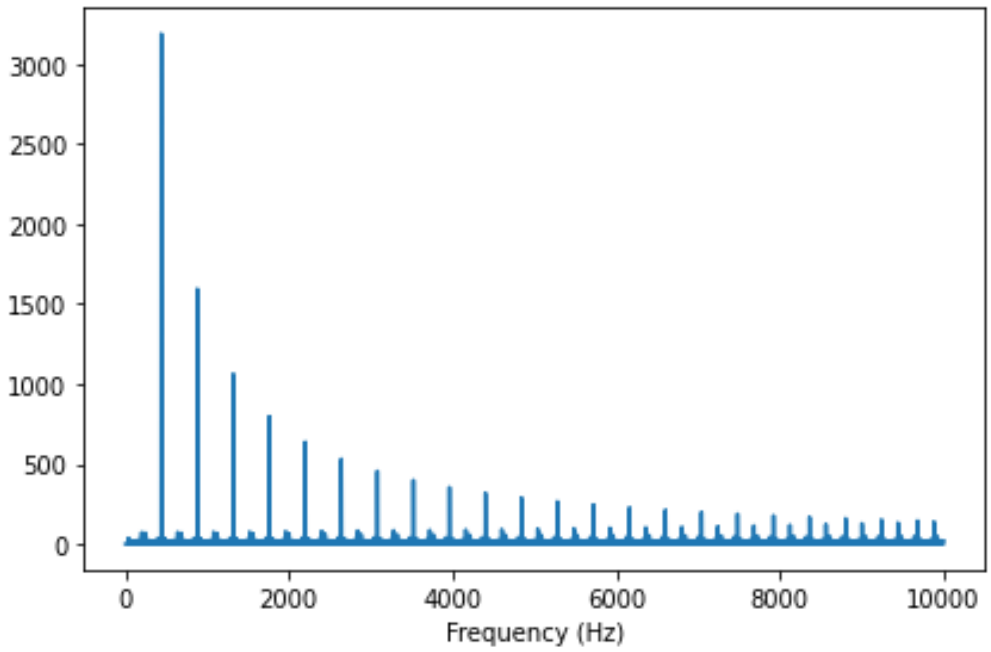

Рис. 2.2: Спектр

Как можно видеть, в данном сигнале присутствуют «четнократные» частоты, при этом и их амлитуды уменьшаются пропорционально самой частоте.

Ниже, на графике происходит сравнение квадратного и пилообразного сигналов:

```
from thinkdsp import SquareSignal

wave.make_spectrum().plot(color='gray')
square = SquareSignal(amp=0.5).make_wave(duration=0.5, framerate=10000)
square.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)')
```

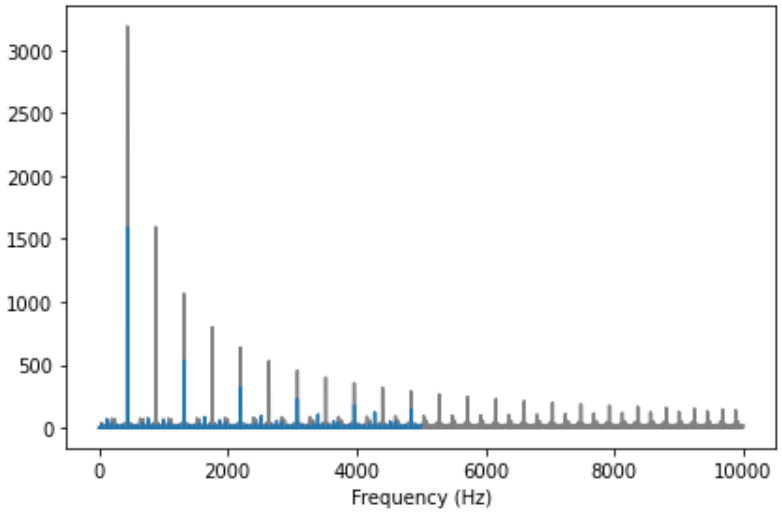

Рис. 2.3: Сравнение спектров

Глава 3

Задание 2.3

Требуется создать SquareSignal частотой 1100 Hz и проанализировать его.

```
from thinkdsp import SquareSignal

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

Листинг 3.1: Искомый сигнал

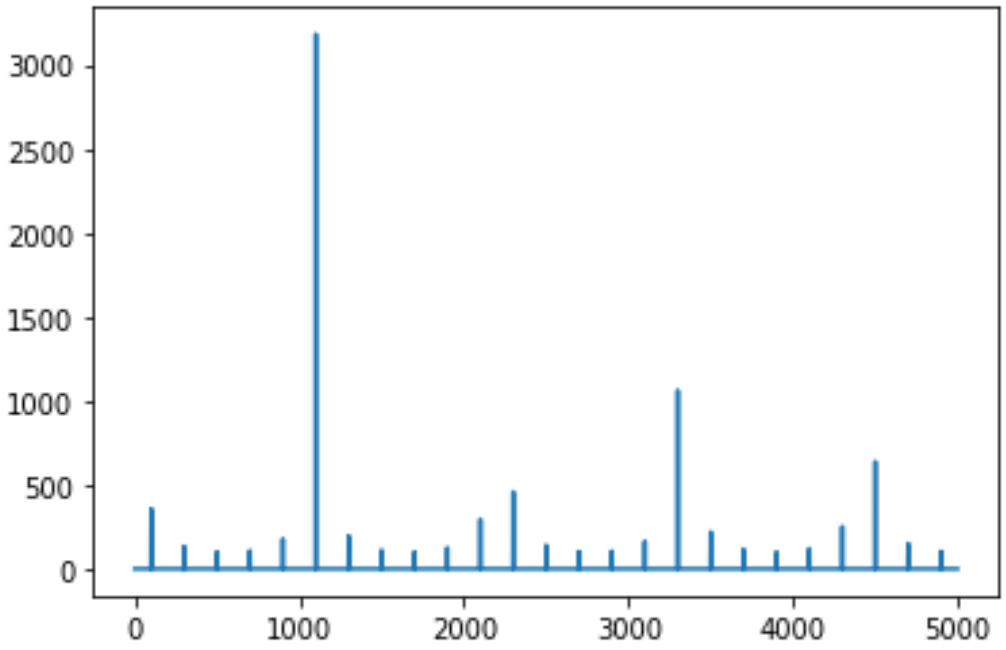

Рис. 3.1: Построенный сигнал

Для сравнения, можно создать новый сигнал с более хорошей частотой дискретизации:

```
wave2 = signal.make_wave(duration=0.5, framerate=signal.freq
    *10)
spectrum2 = wave2.make_spectrum()
spectrum2.plot()
```

Листинг 3.2: Более лучший сигнал

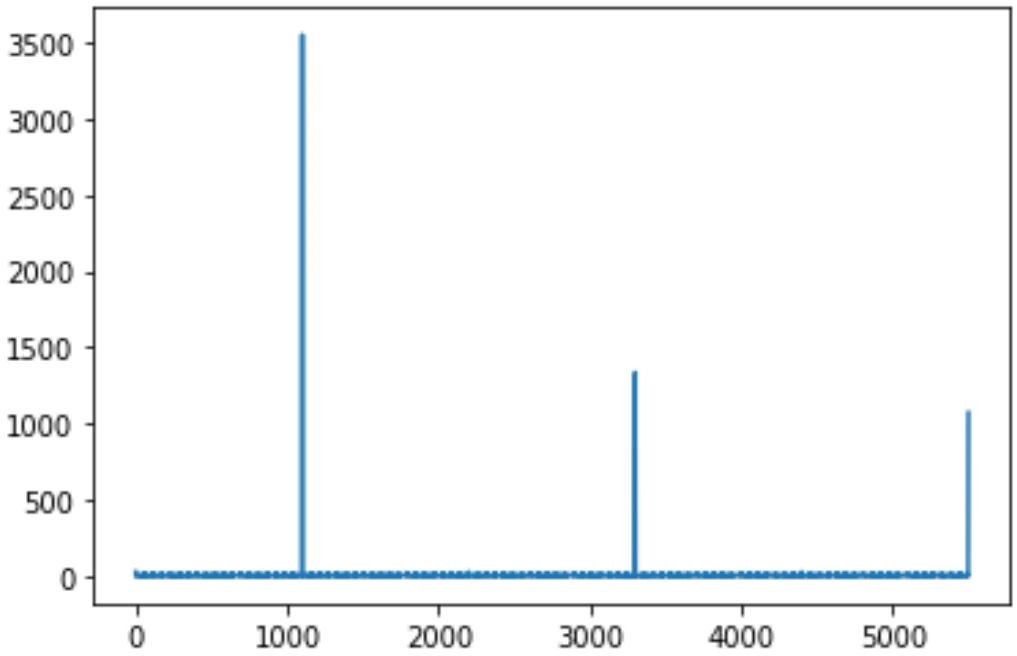

Рис. 3.2: Более лучший сигнал

Кроме того, разница ощутима на слух. Значит, человек способен услышать эти частоты.

Глава 4

Задание 2.4

Требуется поэксперементровать с высотами спектра. Создаём треугольный сигнал:

```
from thinkdsp import TriangleSignal

signal = TriangleSignal(440)
wave = signal.make_wave(duration=0.01, framerate=11025)
wave.plot()
```

Листинг 4.1: stretch

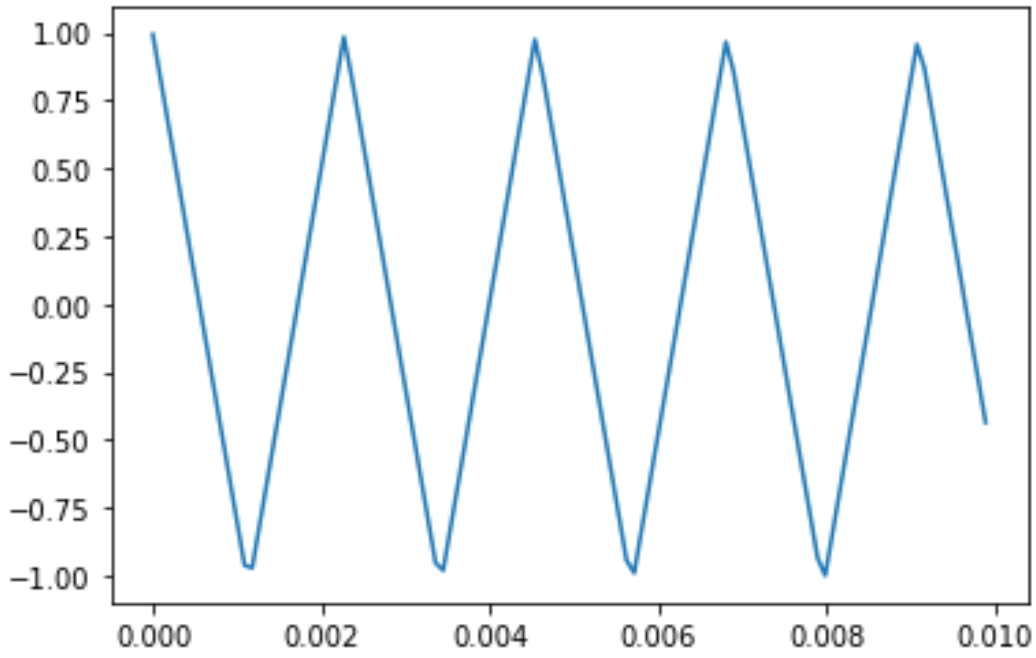

Рис. 4.1: Искомая волна

spectrum.hs[0] имеет значение (1.0436096431476471e-14+0j).
Устанавливаем spectrum.hs[0] = 100:

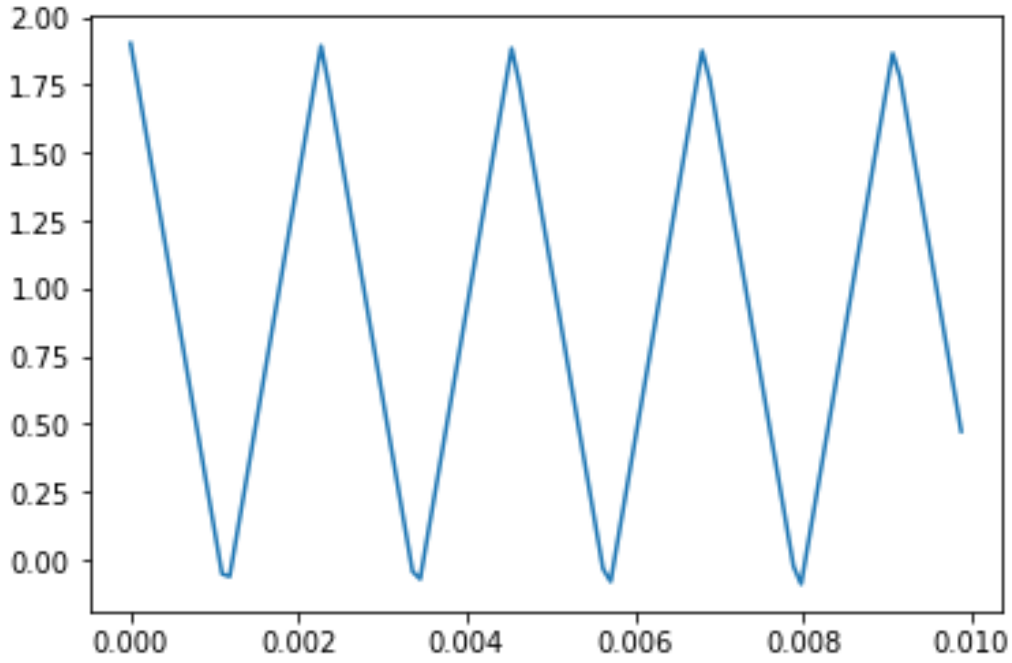

Рис. 4.2: Искомая волна

Как можно видеть, поменялась только амплитуда. Сам вид сигнала не изменился.

Глава 5

Задание 2.5

Требуется написать функцию, делящую амлитуду спектра на частоту. Сначала, создадим пилообразный сигнал из задания 1:

```
from thinkdsp import SawtoothSignal

signal = SawtoothSignal(440)
wave = signal.make_wave(duration=0.5, framerate=440 * 100)
spectrum = wave.make_spectrum()
spectrum.plot()
```

Листинг 5.1: пилообразный сигнал

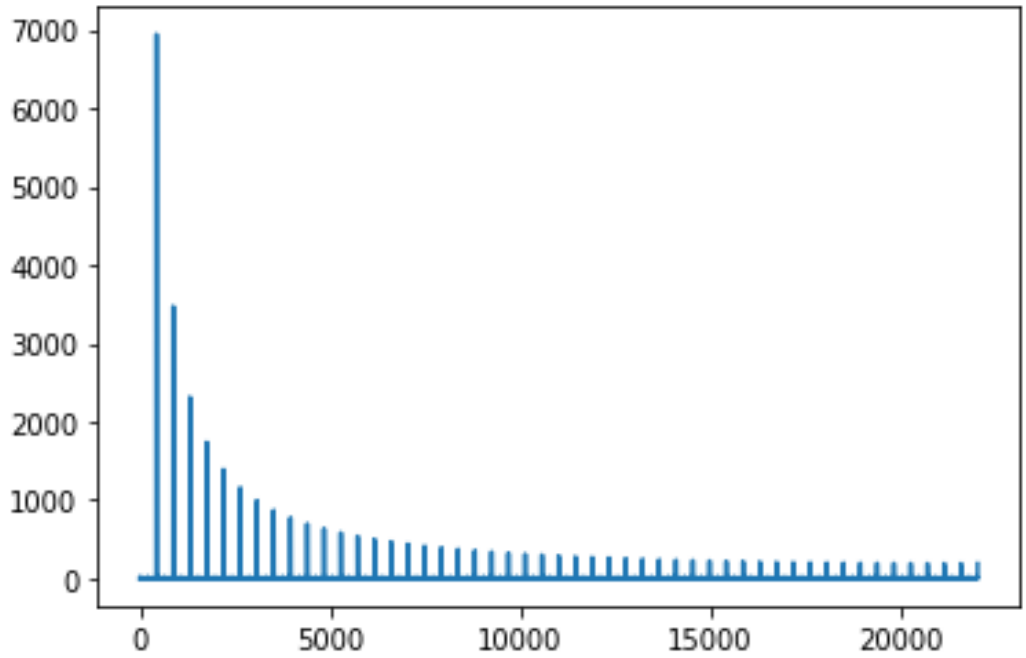

Рис. 5.1: Пилообразный сигнал

Затем, создадим искомую функцию:

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

modify(spectrum)
spectrum.plot(high=100)
```

Листинг 5.2: функция преобразования

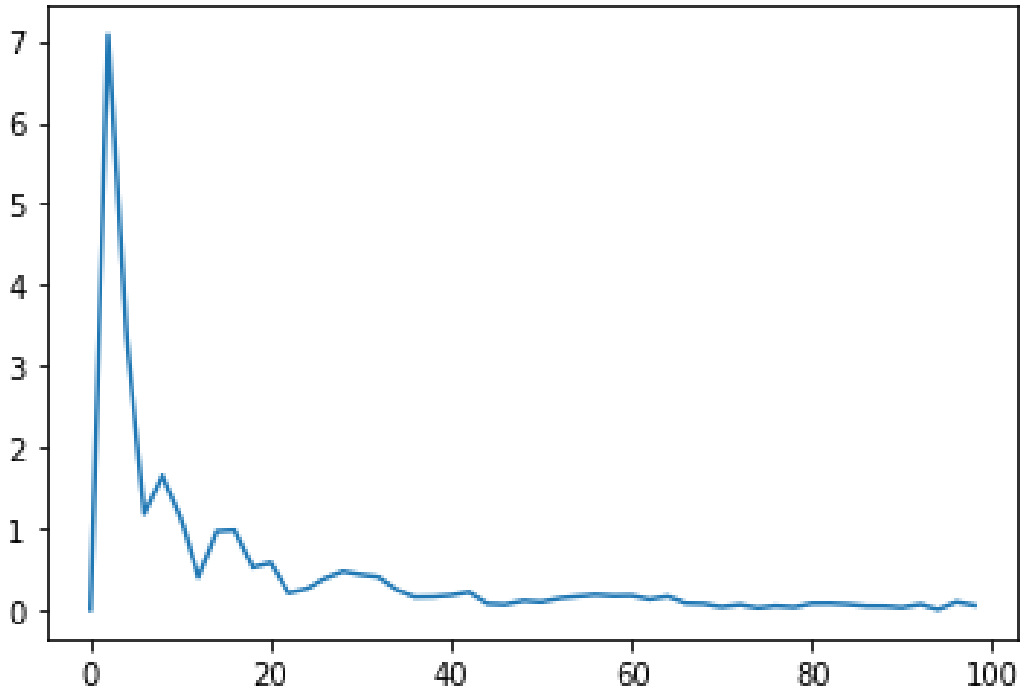

Рис. 5.2: Результат

Операция сделала звук более приглушенным.

Глава 6

Задание 2.6

Требуется найти и построить волну с указанными характеристиками.

Пойдём вторым предложенным путём и начнём изыскания с пилообраного сигнала:

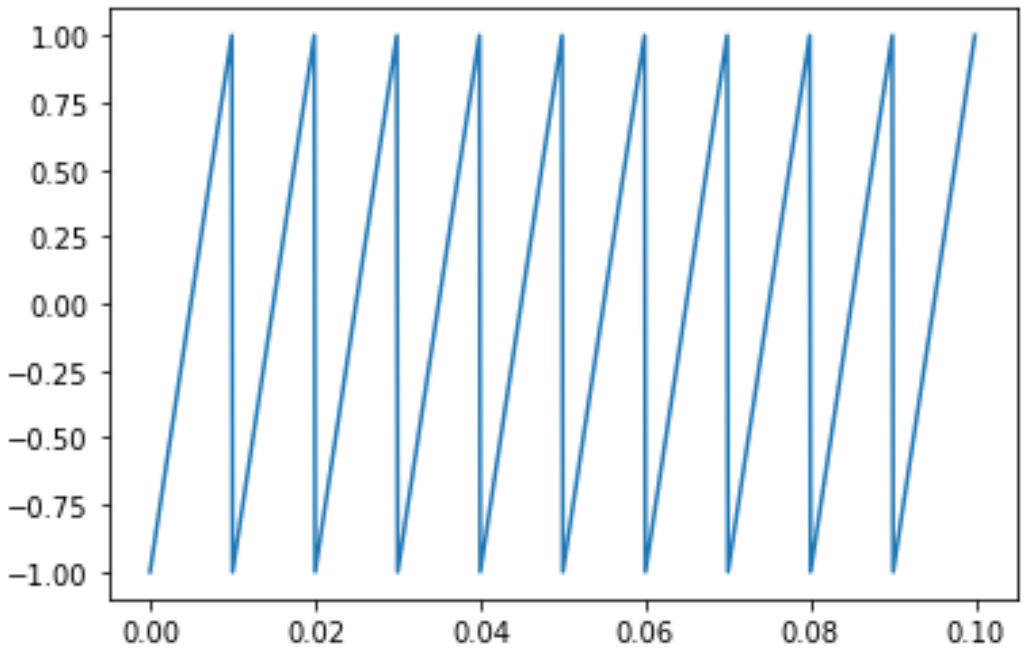

Рис. 6.1: Исходный пилообраный сигнал

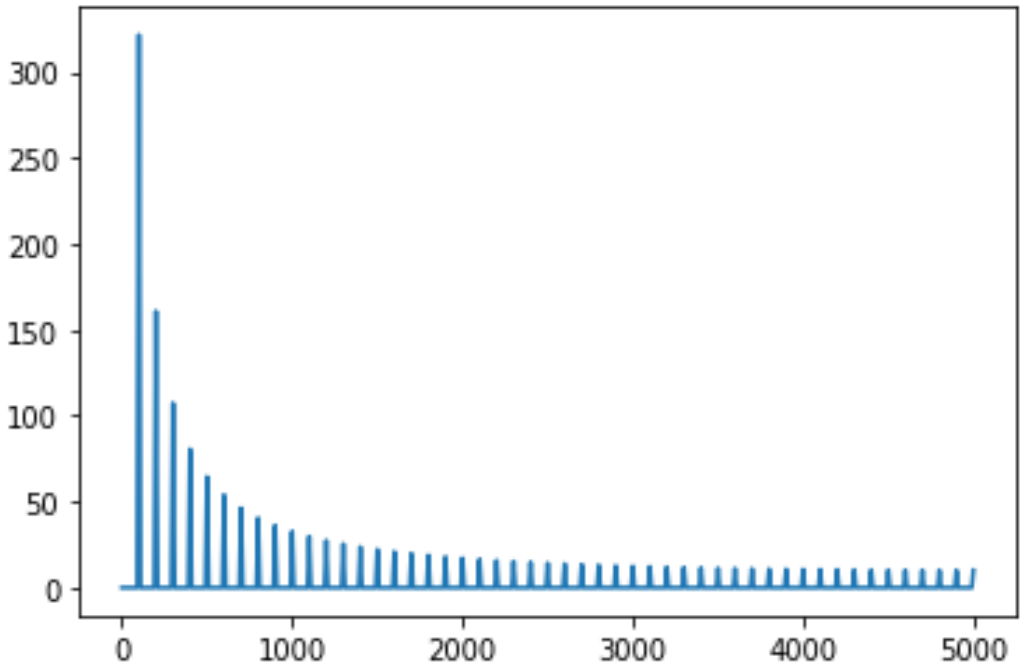

Рис. 6.2: Его спектр

Модифируем его с помощью той же самой функции modify:

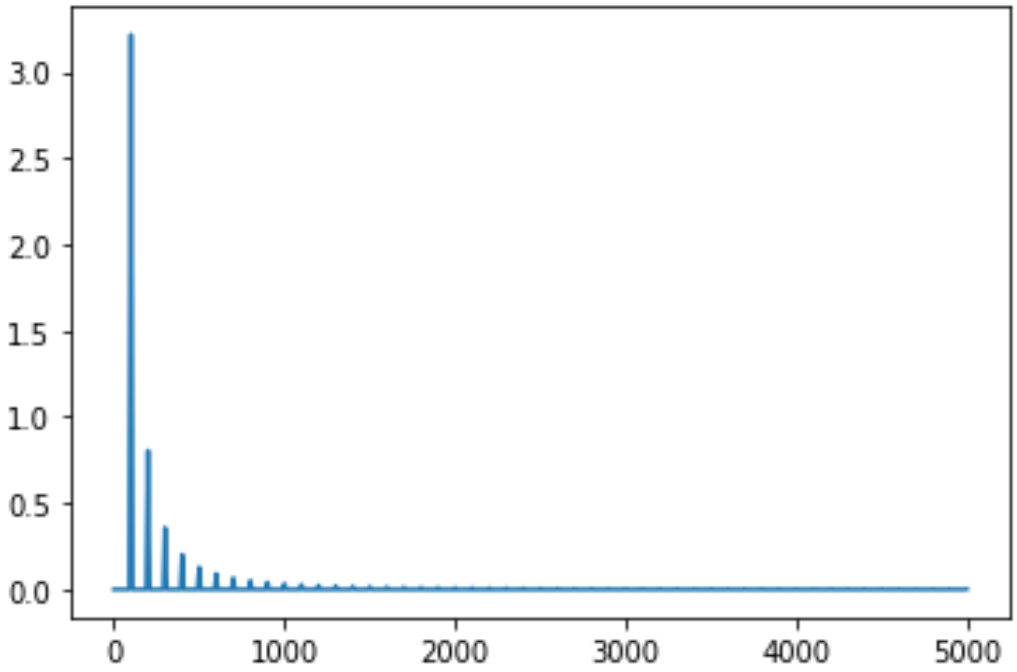

Рис. 6.3: Модифицированный спектр

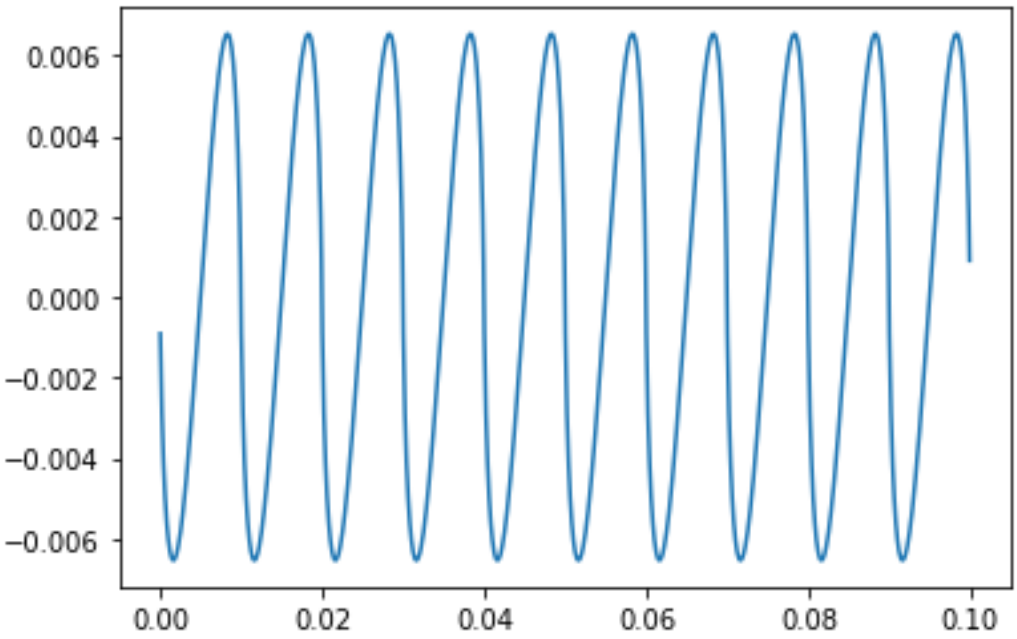

Рис. 6.4: Спектр, преобразованный в волну

Изучение официальной документации по библиотеке thinkdsp навело на мысль, что это может быть ParabolicSignal. Так и есть:

```
from thinkdsp import ParabolicSignal

signal_similar = ParabolicSignal(100)
wave_similar = signal_similar.make_wave(duration=
    signal_similar.period*4, framerate=10000)
wave_similar.plot()
```

Листинг 6.1: ParabolicSignal

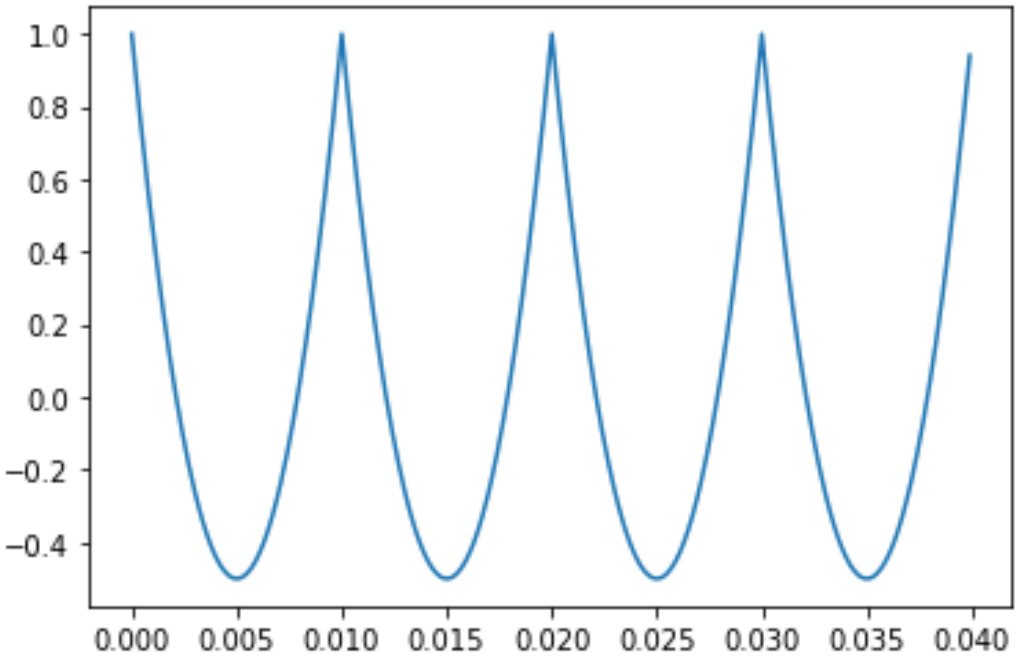

Рис. 6.5: Ответ, волна

Как можно видеть, убывание гармоник подобно квадратичному, присутствуют и чётные, и нечётные основной гармоники:

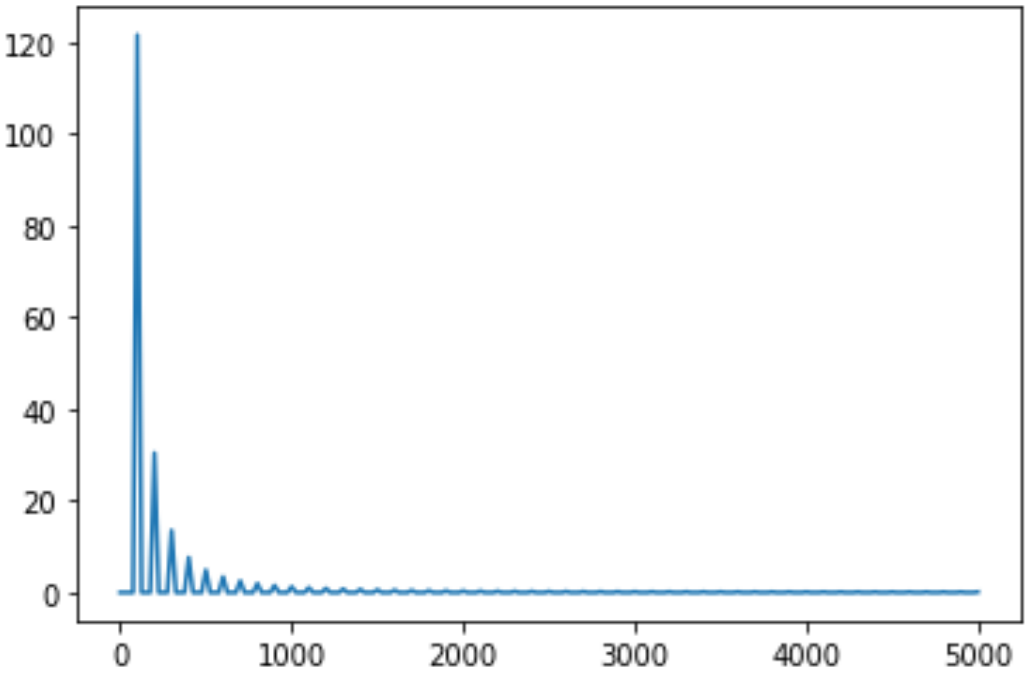

Рис. 6.6: Ответ, спектр