Лабораторная работа 2

Луняк Николай

16 февраля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Пробуем примеры из `chap02.ipynb`

Тут надо просто посмотреть на примеры из второй главы, послушать всякие звуки и убедиться, что все работает.

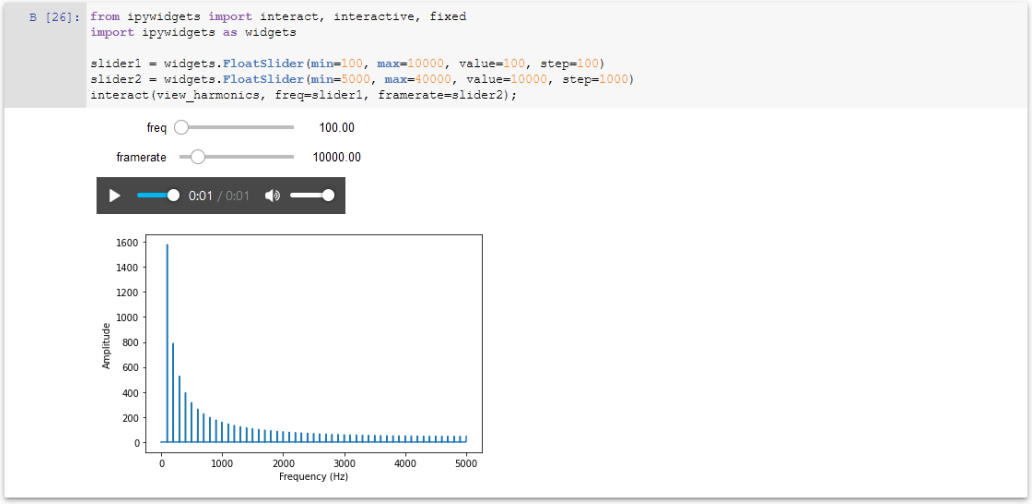

Рис. 1.1: Работает

Глава 2

Делаем `SawtoothSignal`

2.1 Основной код

В задании просят за основу взять `Signal`, но в самой главе приводят примеры создания треугольного и прямоугольного сигналов на основе `Sinusoid`. Скорее всего, в задании опечатка, поэтому я буду пользоваться так же `Sinusoid` (иначе придется реализовывать и конструктор, чтобы передавать ему частоту).

```
from thinkdsp import Signal, Sinusoid, normalize, unbias, PI2

import numpy as np

class MySawtooth(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: Реализация своего пилообразного сигнала

Надо еще убедиться, что он работает корректно.

```
from thinkdsp import decorate

test_saw = MySawtooth(200)
test_wave = test_saw.make_wave(
    test_saw.period*3,
    framerate=10000
)
test_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 2.2: Проверка

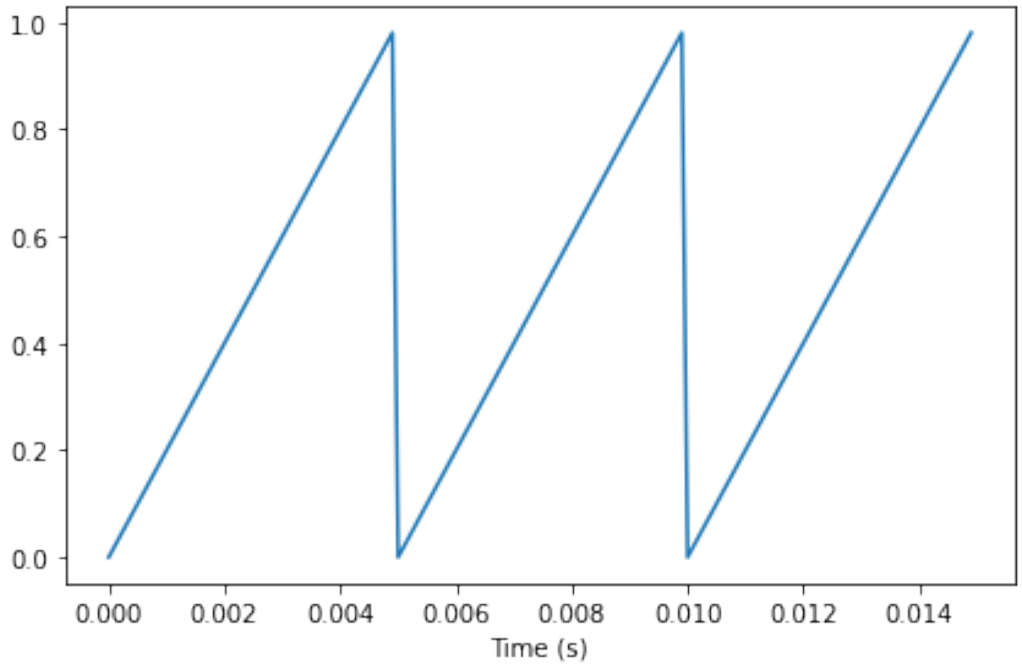

Рис. 2.1: Результат

2.2 Строим спектр

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

Листинг 2.3: Построение спектра

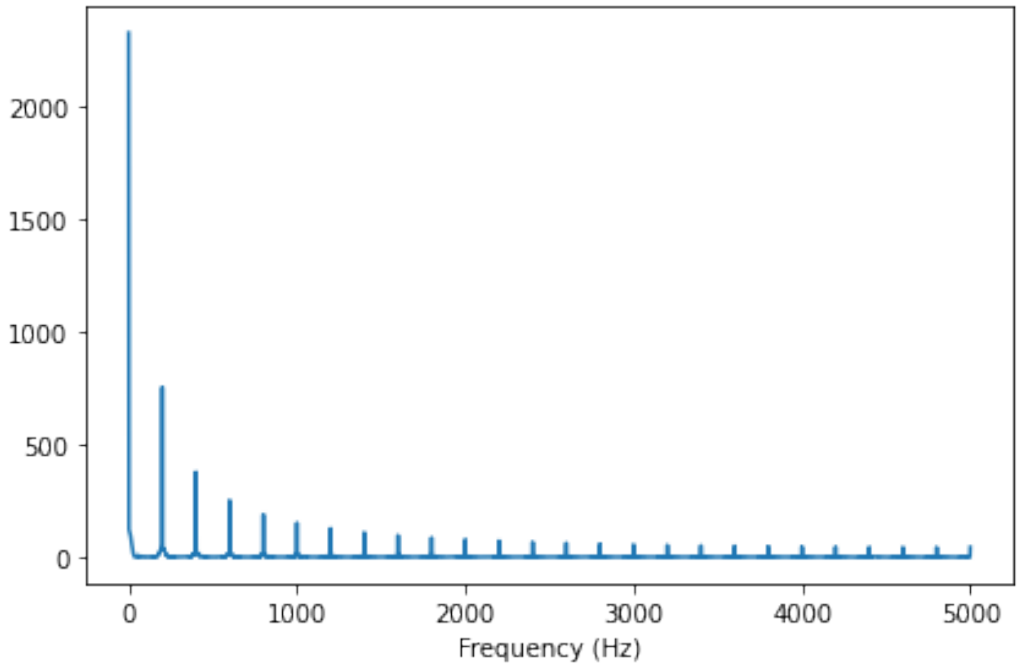

Рис. 2.2: Спектр

В данном сигнале у нас как присутствуют «четнократные» частоты, так и их амлитуды уменьшаются пропорционально самой частоте, а не ее квадрату.

Глава 3

Проверка квадратного сигнала

```
from thinkdsp import SquareSignal

signal = SquareSignal(1100)
wave = signal.make_wave(duration=0.5, framerate=10_000)
spectrum = wave.make_spectrum()
spectrum.plot()
```

Листинг 3.1: Строим квадратный сигнал

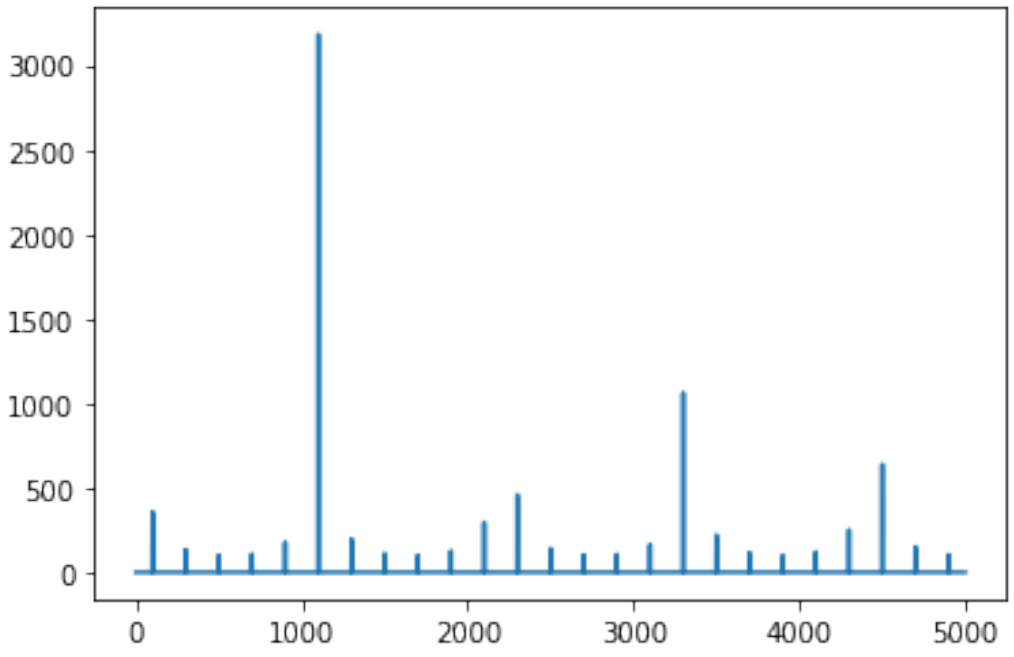

Рис. 3.1: Спектр

Чтобы понять, слышно ли эти «aliased-частоты», можно создать новый сигнал с более хорошей частотой дискретизации и сравнить.

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

Листинг 3.2: Строим квадратный сигнал

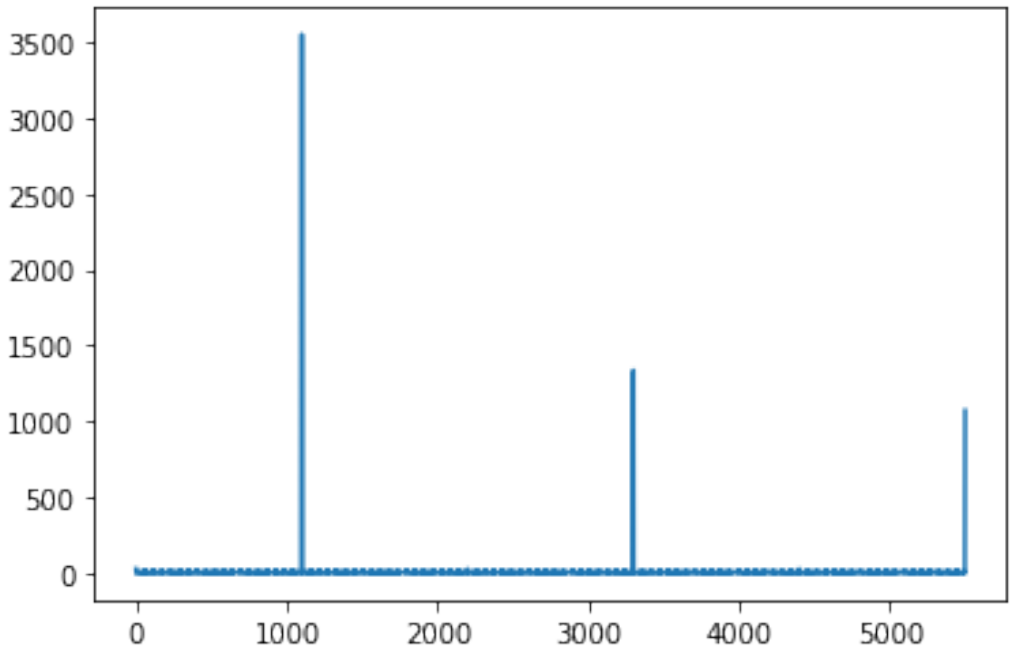

Рис. 3.2: Спектр почище

На слух разница заметна, а значит мы действительно можем расслышать эти частоты.

Глава 4

Нулевая частота

4.1 Создаем сигнал

```
from thinkdsp import TriangleSignal

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

Листинг 4.1: Создаем сигнал

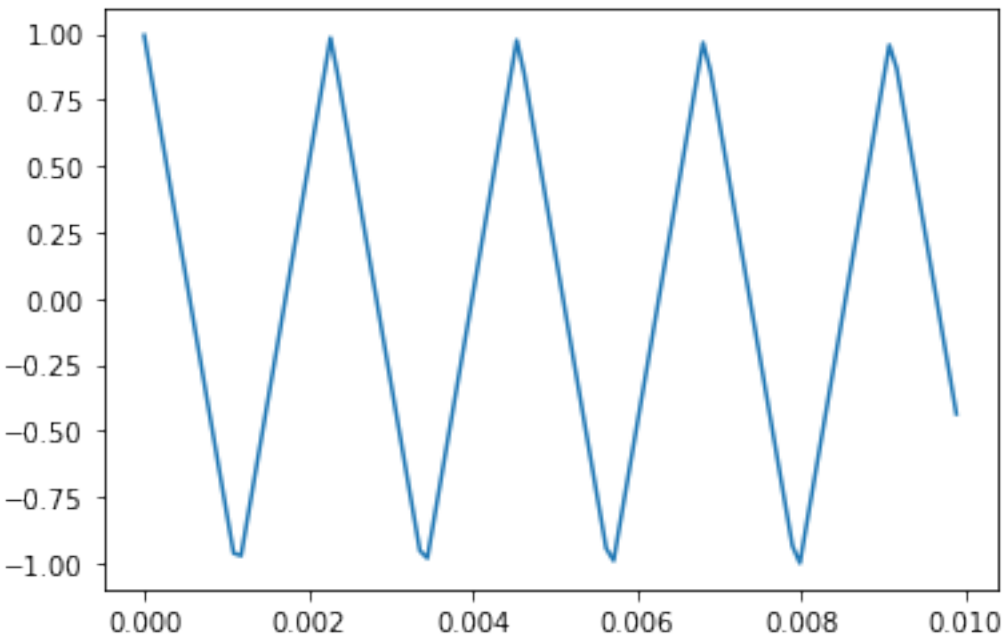

Рис. 4.1: Первый сигнал

4.2 Смотрим на `spectrum.hs[0]`

Видим: `1.0436096431476471e-14+0j`. «Длинна» этого комплексного числа описывает амплитуду нулевой компоненты разложения, а угол с Re - сдвиг по фазе, то есть, частоту.

4.3 Зануляем

```
spectrum.hs[0] = 0
wave2 = spectrum.make_wave()
wave2.plot()
```

Листинг 4.2: «Зануленный» сигнал

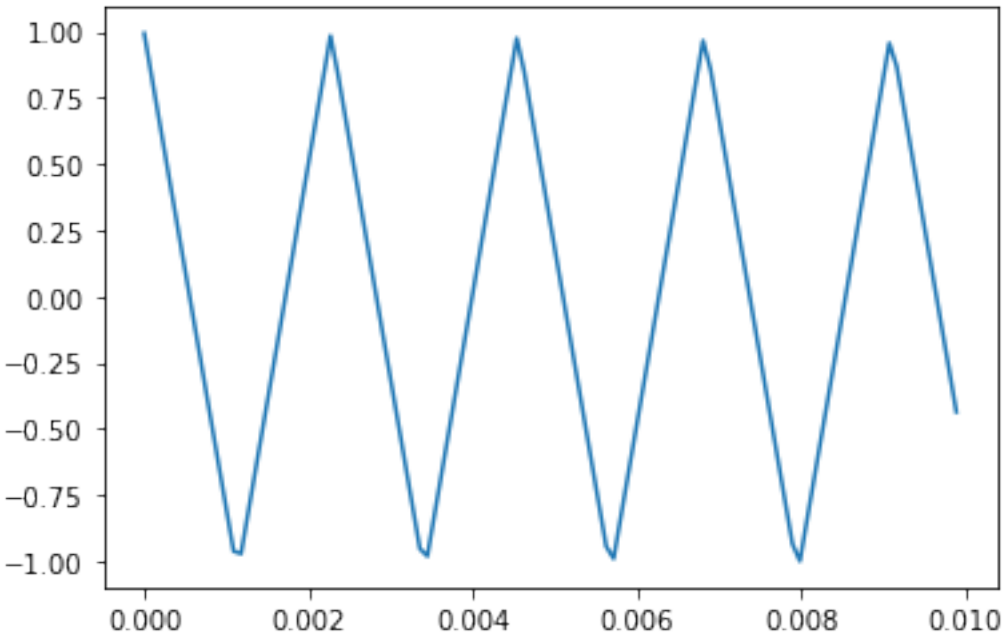

Рис. 4.2: Новый сигнал

Как можно видеть, никакой разницы нет.

Глава 5

Делим амплитуду на частоту

Сама функция имеет следующий вид.

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

Листинг 5.1: Функция трансформации компонент

Для ее проверки я создам пилообразный сигнал.

```
from thinkdsp import SawtoothSignal

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

Листинг 5.2: Исходный сигнал

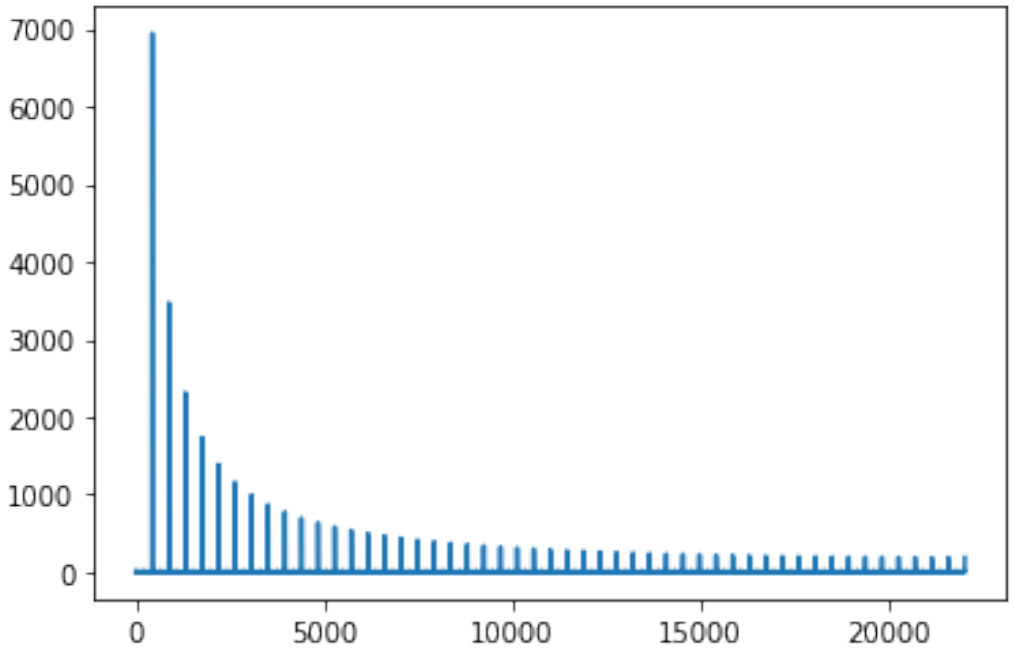

Рис. 5.1: Исходный сигнал

Можно заметить, что амплитуды соответствующих компонент итак уменьшаются достаточно быстро, а, учитывая что внутри `modify()` будет производиться деление на $440k$ ($k \in \mathbb{Z}$), очень многие компоненты окажутся сильно подавленными.

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

Листинг 5.3: После `modify()`

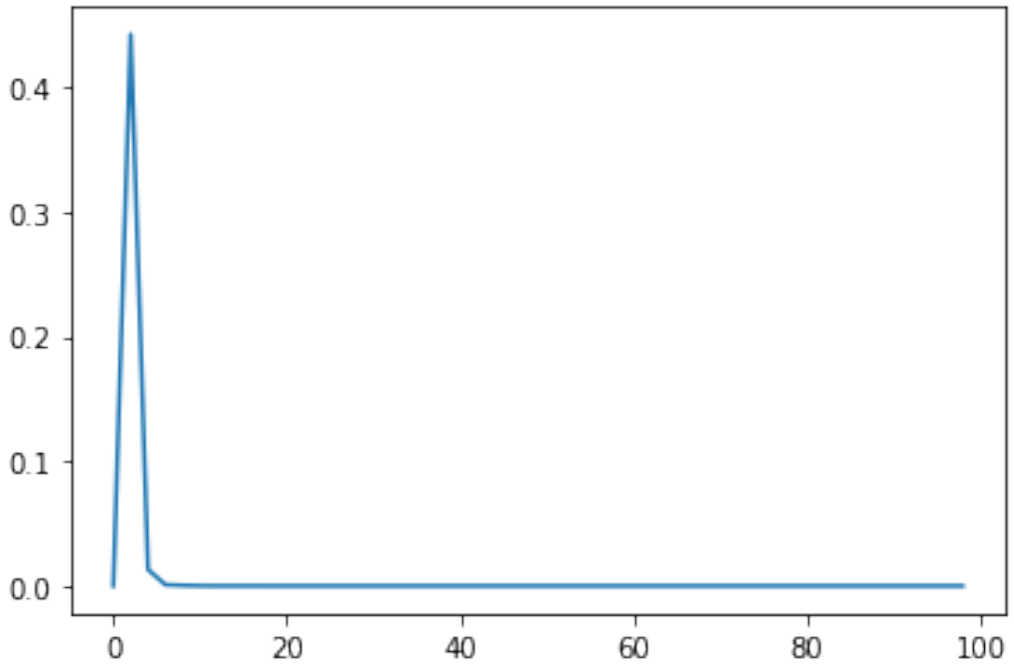

Рис. 5.2: Измененный сигнал

Если прослушать этот сигнал через `spectrum.make_wave().make_audio()`, мы, действительно, ничего не услышим. Возможно, стоило попробовать с `SawtoothSignal`, ведь умплитуды компонент у него спадают обратно частоте а не ее квадрату.

Глава 6

Ищем загадочный сигнал

Возьмем пилообразный сигнал, применив написанную ранее `modify()`.

```
signal = SawtoothSignal(100)
wave = signal.make_wave(
    duration=signal.period*10,
    framerate=10_000
)
wave.plot()
```

Листинг 6.1: Новый сигнал

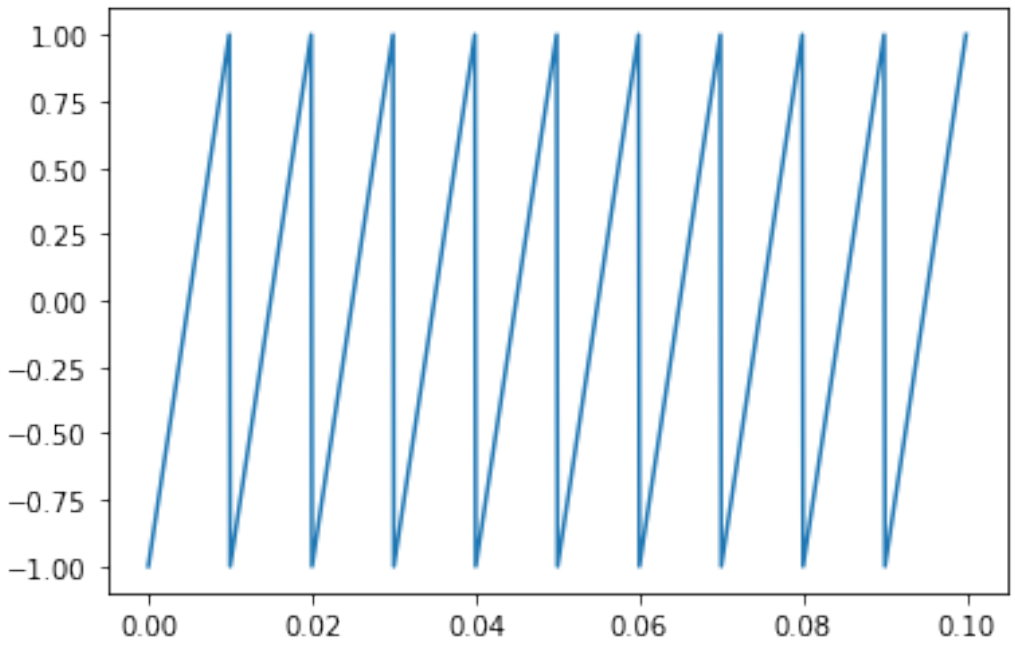

Рис. 6.1: Пилообразный сигнал

```
spectrum = wave.make_spectrum()
spectrum.plot()
```

Листинг 6.2: Его спектр

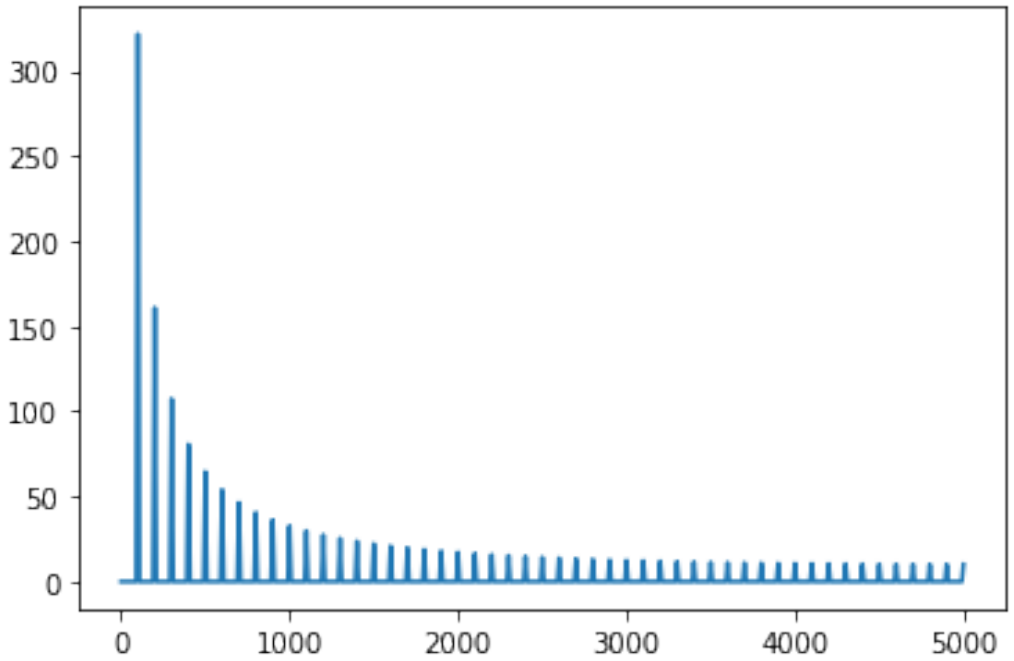

Рис. 6.2: Спектр

Применим `modify()`.

```
modify(spectrum)
spectrum.plot()
```

Листинг 6.3: Снова применяем `modify()`

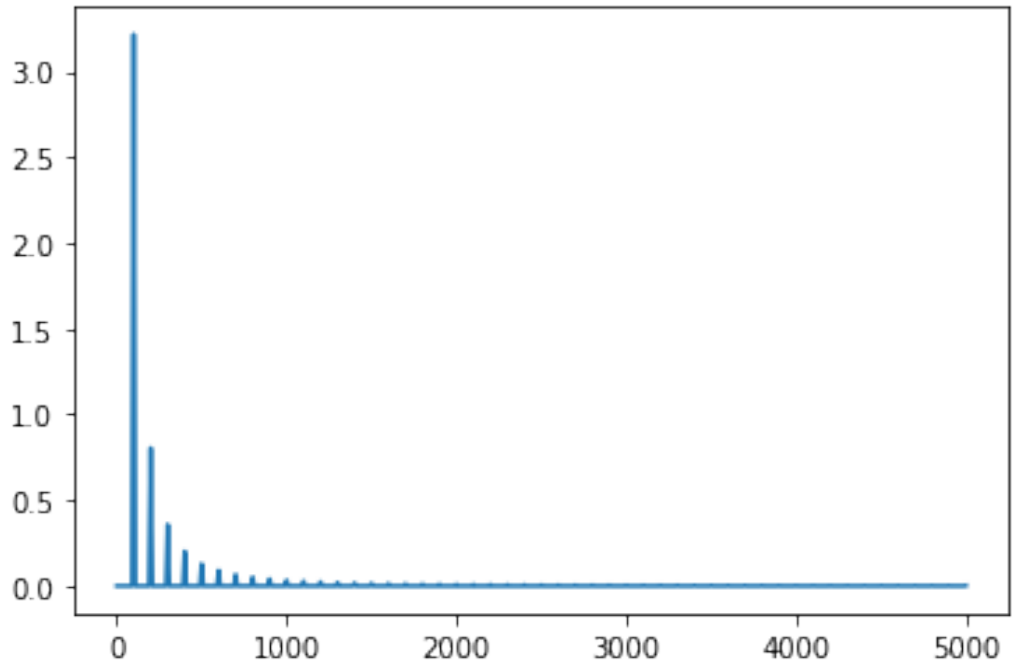

Рис. 6.3: Модифицированный спектр

Посмотрим, что там вышло.

```
wave2 = spectrum.make_wave()
wave2.plot()
```

Листинг 6.4: Смотрим

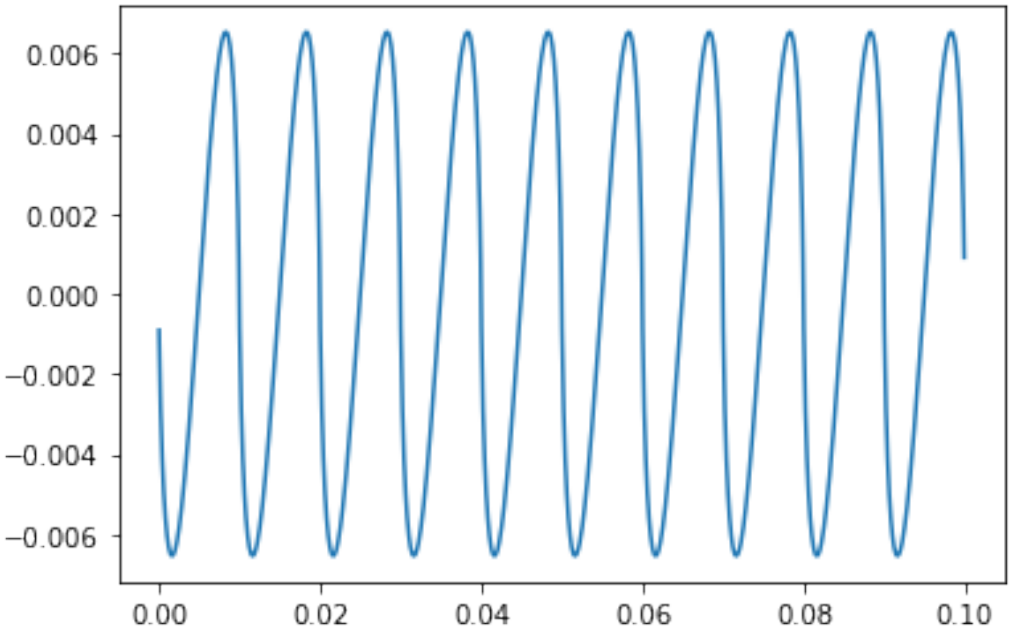

Рис. 6.4: Видим

Похоже на синусоиду, но это точно не она. Иначе бы спектр содержал лишь одну компоненту. Признаюсь, я так и не догадался сам, поэтому посмотрел в решение - там написано, что это `ParabolicSignal`.

```
from thinkdsp import ParabolicSignal

signal4 = ParabolicSignal(100)
wave4 = signal4.make_wave(duration=signal4.period*4,
    framerate=10_000)
wave4.plot()
```

Листинг 6.5: Что же это за сигнал такой...

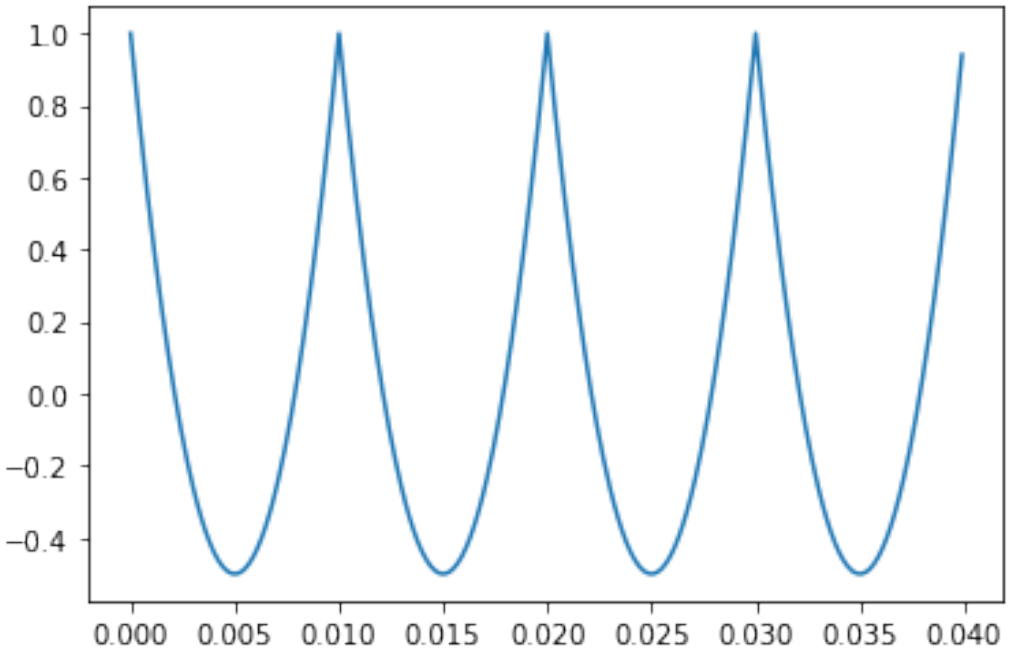

Рис. 6.5: Вот это я понимаю сигнал