Лабораторная работа 1

Луняк Николай

9 февраля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Проверка

Тут надо просто проверить, что ноутбуки запускаются. Да, запускаются.

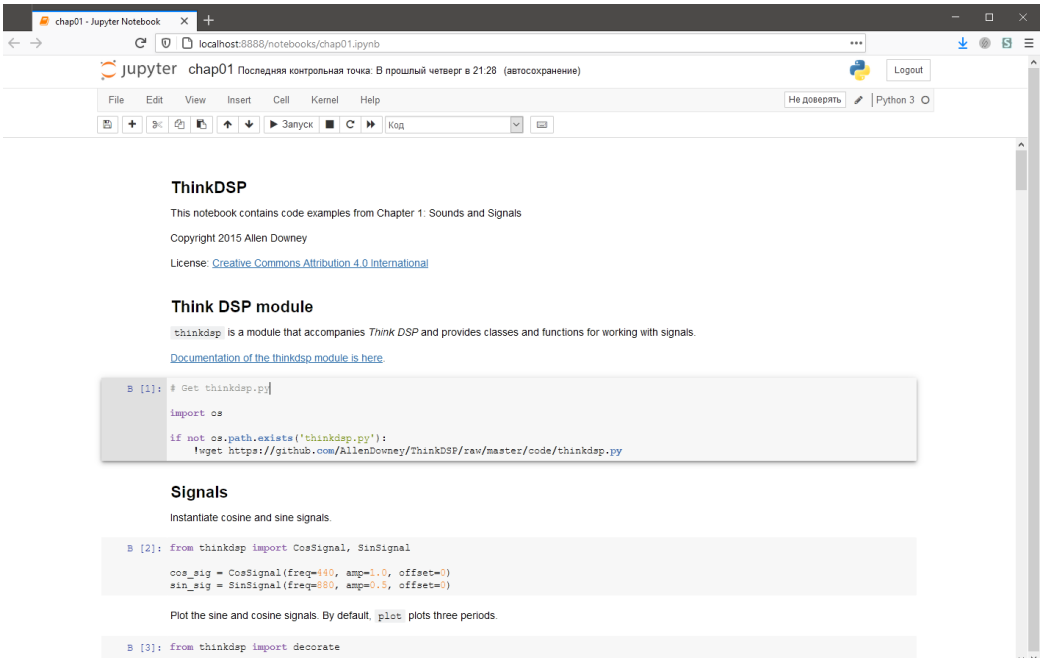

Рис. 1.1: Работает

Но не все так гладко... Например, в коде ниже используется `aplay`, которого у меня нет, потому что я работаю на Windows.

```
from thinkdsp import play_wave
play_wave(filename='temp.wav', player='aplay')
```

Листинг 1.1: Использование `aplay`

Я попробовал использовать `ffplay`, который у меня есть, и звук, действительно, стал проигрываться... но он так никогда и не закончил это делать и notebook завис на одной команде. Пришлось перезапускать.

Глава 2

Простая обработка

2.1 Получение звука

```
from thinkdsp import read_wave
wave = read_wave(
    'Sounds/557274__johnnie-holiday__dark-ambience_cut.wav'
)
```

Листинг 2.1: Загрузка звука

По какой-то причине скаченный файл содержал 3 канала, из-за чего `play_wave()` не могла его загрузить. Пришлось открыть Audacity и пересохранить файл руками (и там же я его и обрезал до половины секунды).

```
from thinkdsp import decorate
segment = wave.segment(0, wave.duration)
segment.plot()
decorate(xlabel='Time (s)')
```

Листинг 2.2: Визуализация

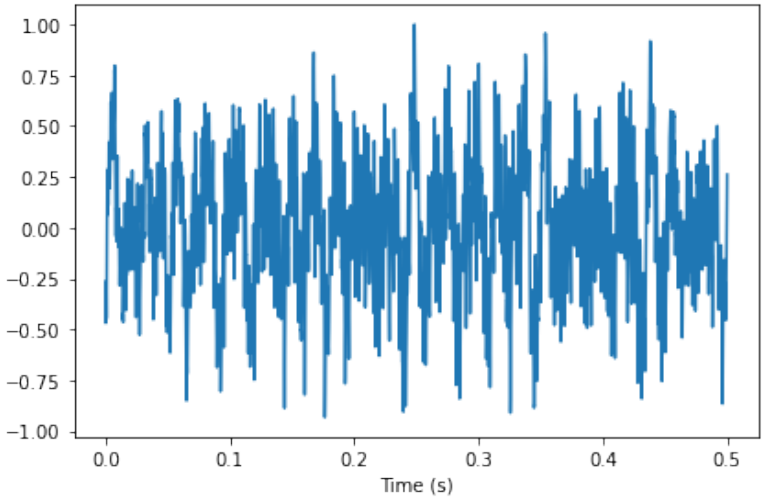

Рис. 2.1: Исходный звук

2.2 Спектр

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

Листинг 2.3: Спектр

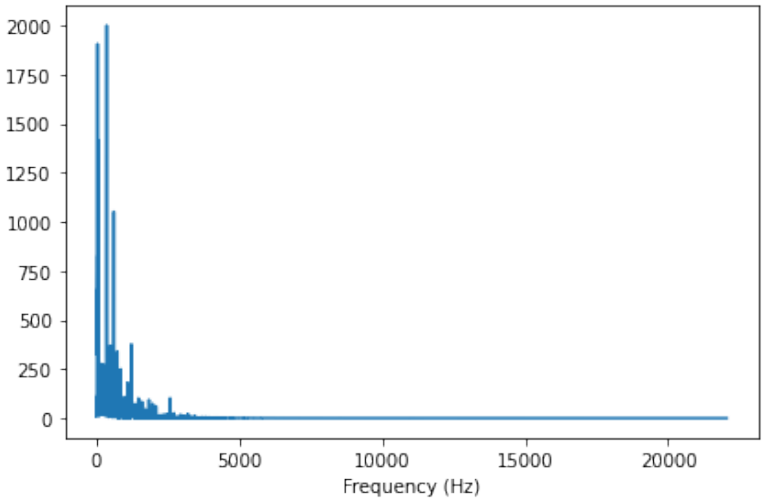

Рис. 2.2: Спектр

```
spectrum.plot(high=3000)
decorate(xlabel='Frequency (Hz)')
```

Листинг 2.4: Улучшаем масштаб

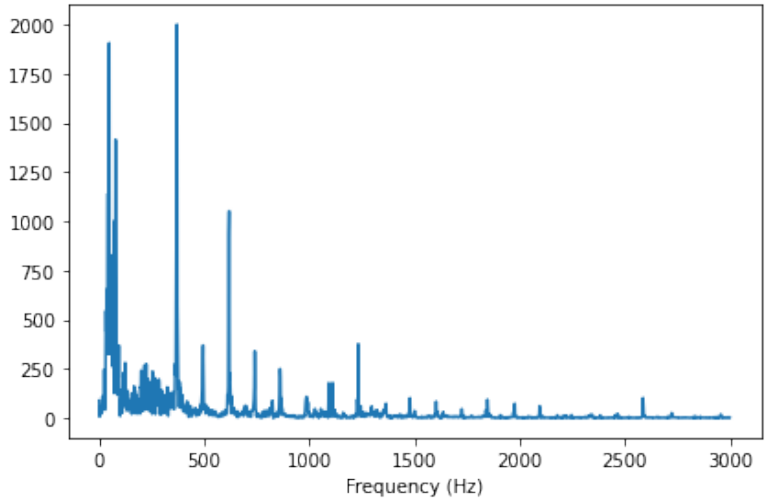

Рис. 2.3: Улучшили масштаб

Чем больше частот, тем более богатым считается тембр.

В первый раз я взял звук под названием 445999__breviceps__fart-2, и его спектр оказался чуть ли не шумом. После небольшой фильтрации я вообще получил тишину. Получается, подобные звуки - сами по себе шум, а в соответствии с определением выше, они могут считаться очень богатыми.

2.3 Фильтрация

```
spectrum.low_pass(1300)
spectrum.plot(high=1300)
decorate(xlabel='Frequency (Hz)')
```

Листинг 2.5: Делаем `low_pass`

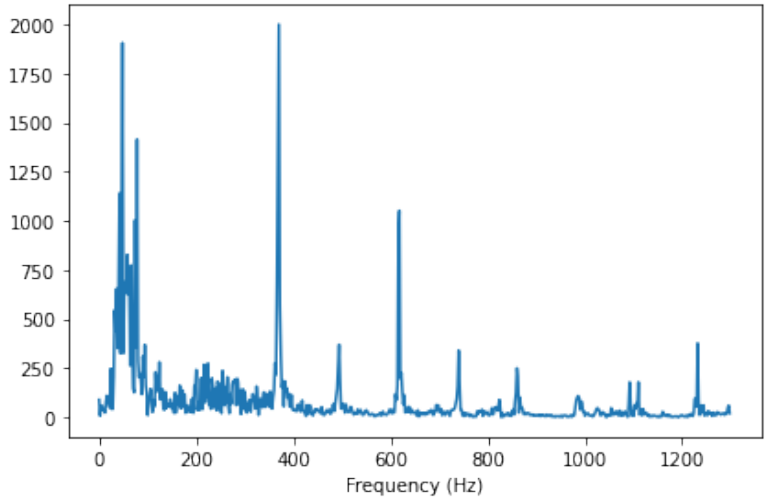

Рис. 2.4: После `low_pass`

```
filtered = spectrum.make_wave()
filtered.normalize()
filtered.plot()
decorate(xlabel='Time (s)')
```

Листинг 2.6: Финальный результат

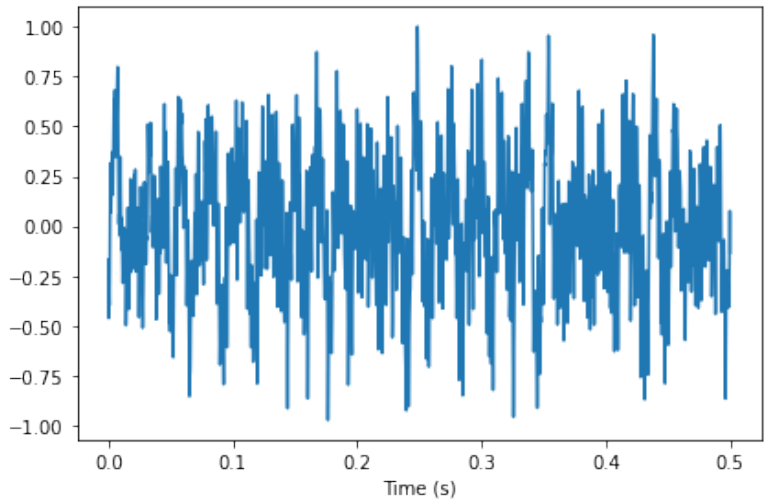

Рис. 2.5: Финальный результат

Звучит это все дело в сравнении с оригиналом так, как если бы на пути от источника звука до «нас» образовалась какая-то стена.

Глава 3

Комбинирование

3.1 Кратные частоты

```
from thinkdsp import CosSignal, SinSignal

cos_sig = CosSignal(freq=440,     amp=1.0, offset=0)
sin_sig = SinSignal(freq=440 * 3, amp=0.7, offset=0)

cos_sig.plot()
decorate(xlabel='Time (s)')

sin_sig.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.1: Создаем 2 сигнала

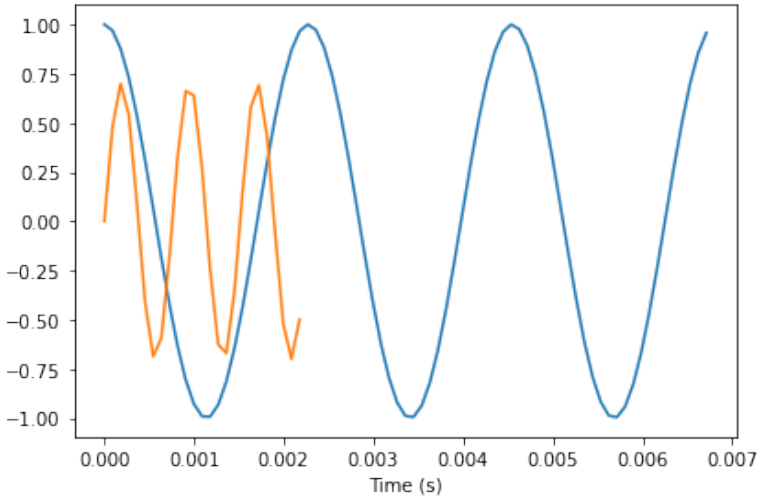

Рис. 3.1: Смотрим на 2 сигнала

```
mix = sin_sig + cos_sig
mix.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.2: Суммируем 2 сигнала

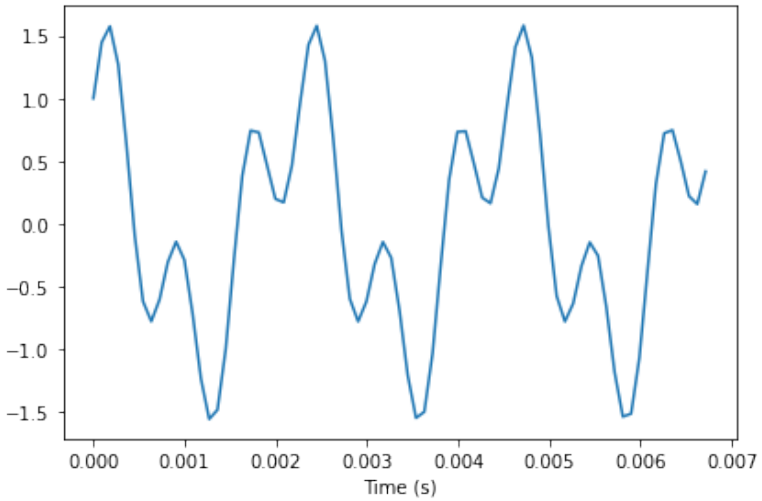

Рис. 3.2: Смотрим на сумму 2 сигналов

```
spectrum = wave.make_spectrum()
spectrum.plot(high=2000)
```

```
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.3: Смотрим на спектр 2 сигналов

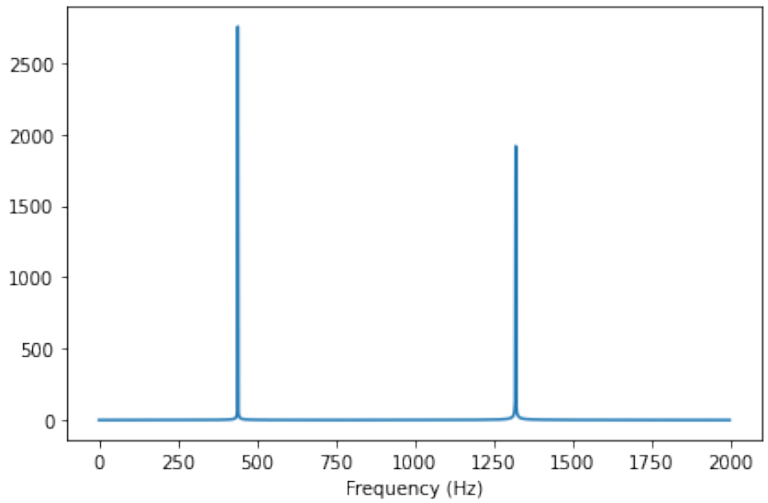

Рис. 3.3: Смотрим на спектр суммы

На слух это все звучит... ну нормально. Звук как звук. Относительно приятный.

3.2 Вносим что-то несочетающееся

```
one_more_cos_sig = CosSignal(
    freq=440 * 2.1315,
    amp=1.0,
    offset=0
)

one_more_mix = mix + one_more_cos_sig
one_more_mix.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.4: Примешиваем некратную частоту

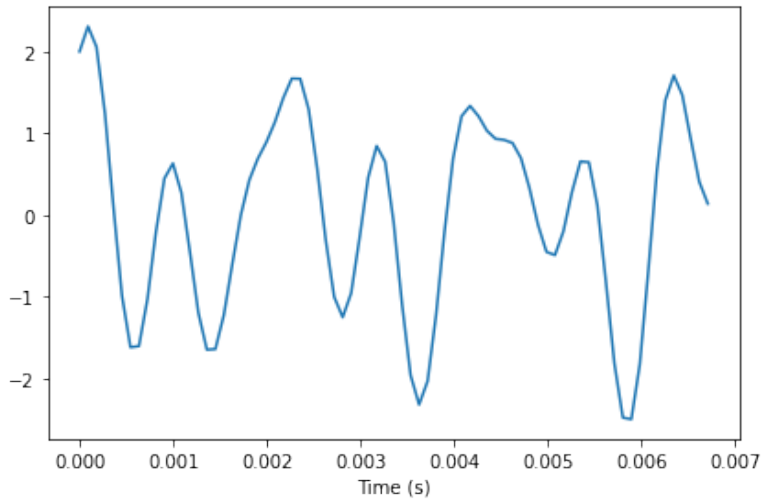

Рис. 3.4: Вместе с некратной

```
one_more_spectrum = one_more_wave.make_spectrum()
one_more_spectrum.plot(high=2000)
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.5: Смотрим спектр

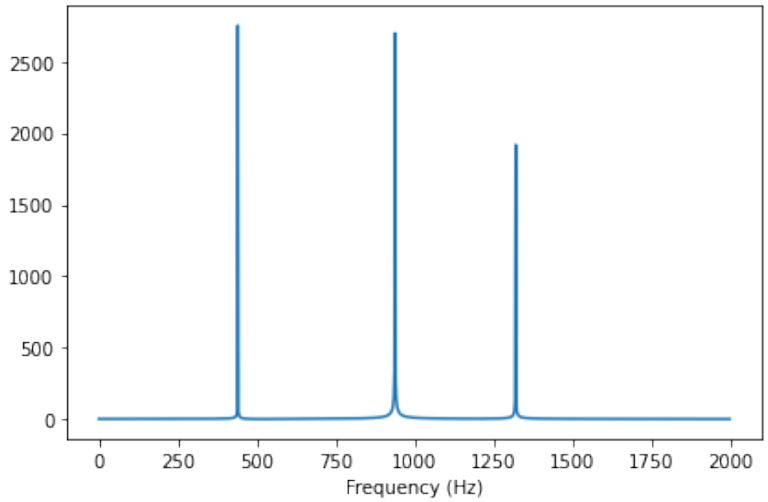

Рис. 3.5: Спектр

Этот звук уже больше походит на звук ошибки на каком-то космическом корабле.

3.3 Генерация

Хочется посмотреть, что будет, если взять много частот. Для этого я написал функцию `generate_compound()`, которой надо передать количество частот и правило генерации i-ой частоты.

```
def generate_compound(components_count, get_next_frequency):
    mix = CosSignal(
        freq=get_next_frequency(0),
        amp=1.0,
        offset=0
    )

    for it in range(2, components_count, 2):
        frequency = get_next_frequency(it)
        mix += CosSignal(
            freq=frequency,
            amp=1.0/(it + 1)**2,
            offset=0
        )

    for it in range(1, components_count, 2):
        frequency = get_next_frequency(it)
        mix += SinSignal(
            freq=frequency,
            amp=1.0/(it + 1)**2,
            offset=0
        )

    return mix.make_wave(
        duration=0.5,
        start=0,
        framerate=11025
    )
```

Листинг 3.6: Генерация

Пользоваться этим можно так:

```
import random

wave_multiples = generate_compound(
    10,
    lambda it: 440 * (it + 1)
)

wave_non_multiples = generate_compound(
    10,
    lambda it: 440 * random.uniform(1, 5)
```

```
)
```

Листинг 3.7: Создаем `Wave`'ы

```
wave_multiples.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.8: Смотрим кратные

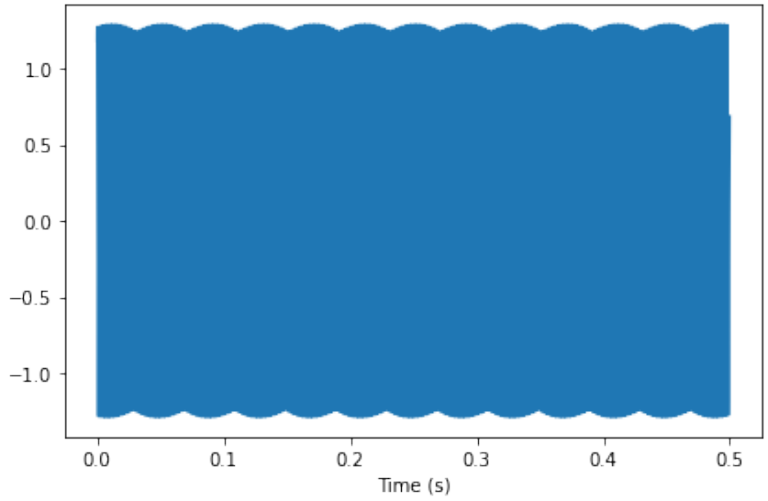

Рис. 3.6: Смотрим кратные

```
wave_non_multiples.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.9: Смотрим некратные

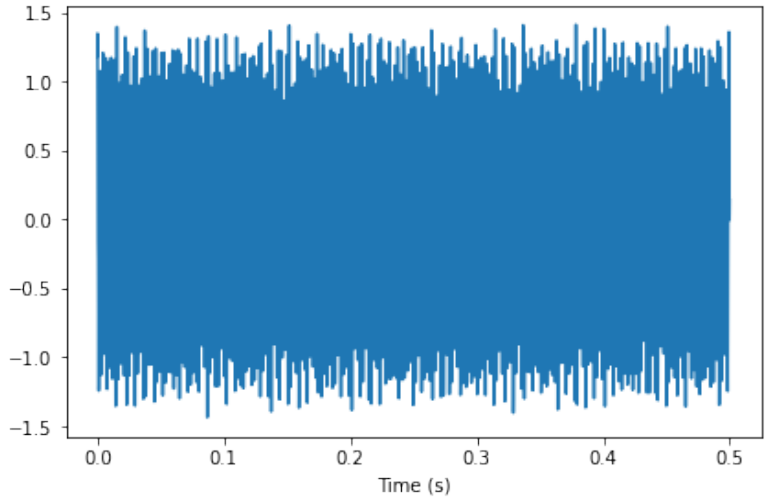

Рис. 3.7: Смотрим некратные

Ну и сами спектры.

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

Листинг 3.10: Смотрим спектр кратных

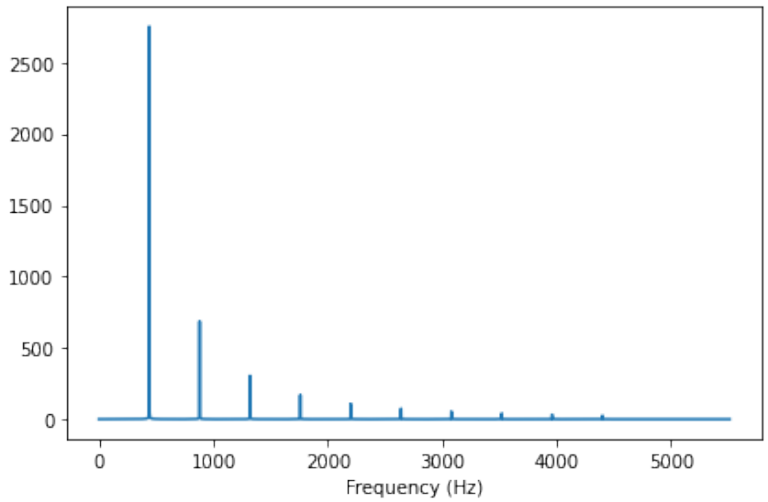

Рис. 3.8: Смотрим спектр кратных

```
spectrum_non_multiples = wave_non_multiples.make_spectrum()
```

```
spectrum_non_multiples.plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.11: Смотрим спектр некратных

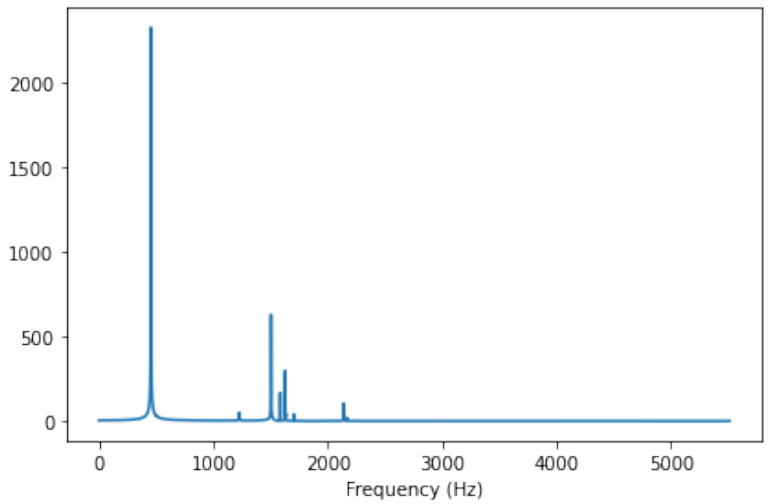

Рис. 3.9: Смотрим спектр некратных

Полученные звуки различаются, и природа этой разнецы та же, что и у полученых ранее звуков. Первый звук кажется более цельным (одна нота, но проигранная на неком инструменте со своим особым тембром), в то время как второй звук явно состоит из каких-то частей.

Глава 4

Растяжение

Ну, тут все понятно.

```
1 def stretch(wave, stretch_factor):
2     wave.framerate *= stretch_factor
```

Листинг 4.1: Растягиваем `Wave`