Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Отчёт по лабораторной работе №2
Дисциплина: Телекоммуникационные технологии
Тема: Гармоники

Выполнил студент гр. 3530901/80201 В.А. Пучкина

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

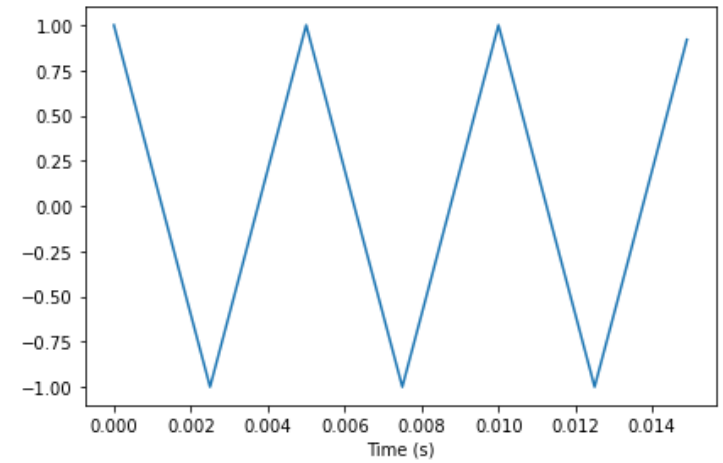
Содержание

Список иллюстраций

Листинги

1 Упражнение 2.1

В этому упражнении необходимо загрузить файл `chap02.ipynb`, изучить его, просмотреть пояснения и запустить примеры. Проверим, что всё работает корректно.



Make a wave and play it.

```
wave = signal.make_wave(duration=0.5, framerate=10000)
wave.apodize()
wave.make_audio()
```

Compute its spectrum and plot it.

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

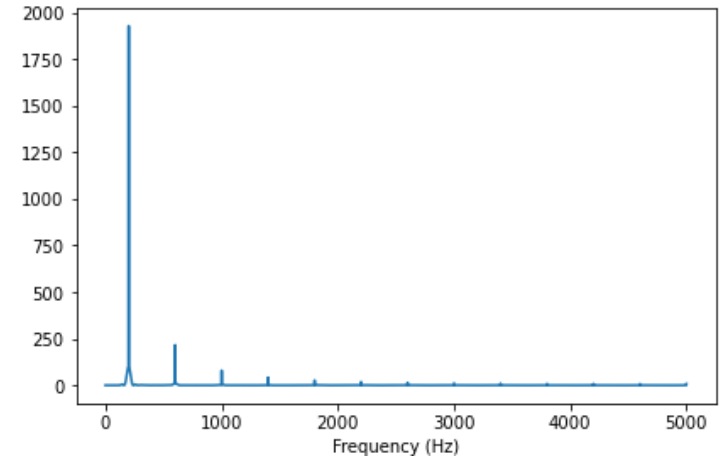
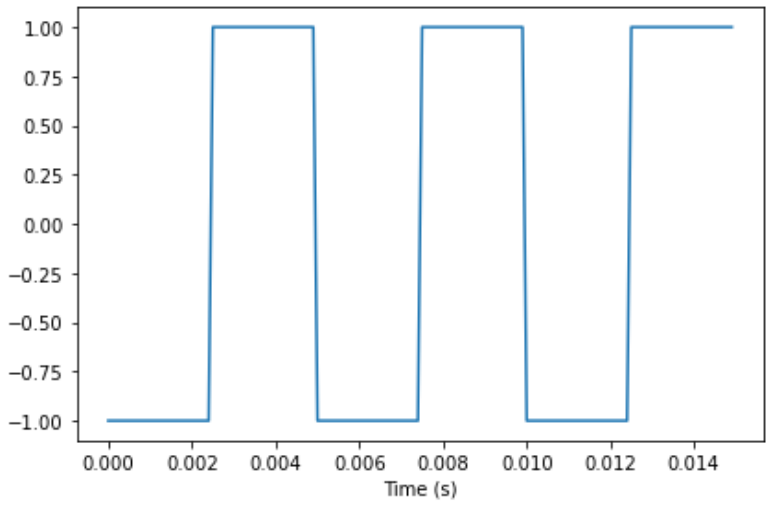


Рис. 1: Изучение и проверка примеров (треугольный сигнал).



Make a wave and play it.

```
wave = signal.make_wave(duration=0.5, framerate=10000)
wave.apodize()
wave.make_audio()
```

Compute its spectrum and plot it.

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

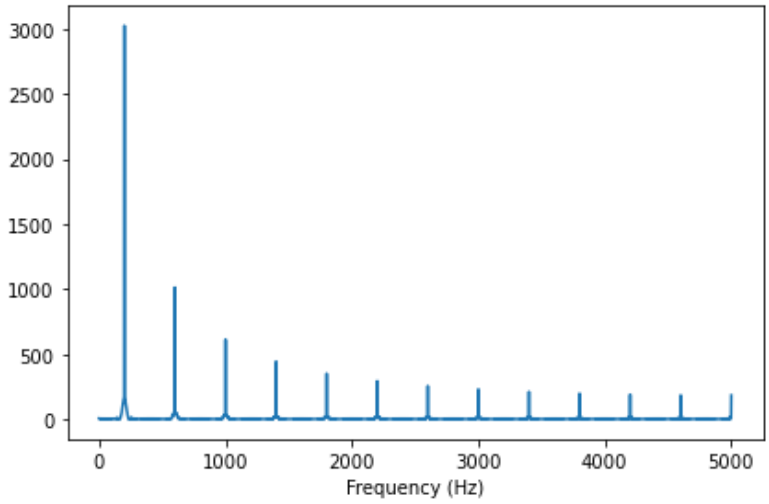


Рис. 2: Изучение и проверка примеров (прямоугольный сигнал).

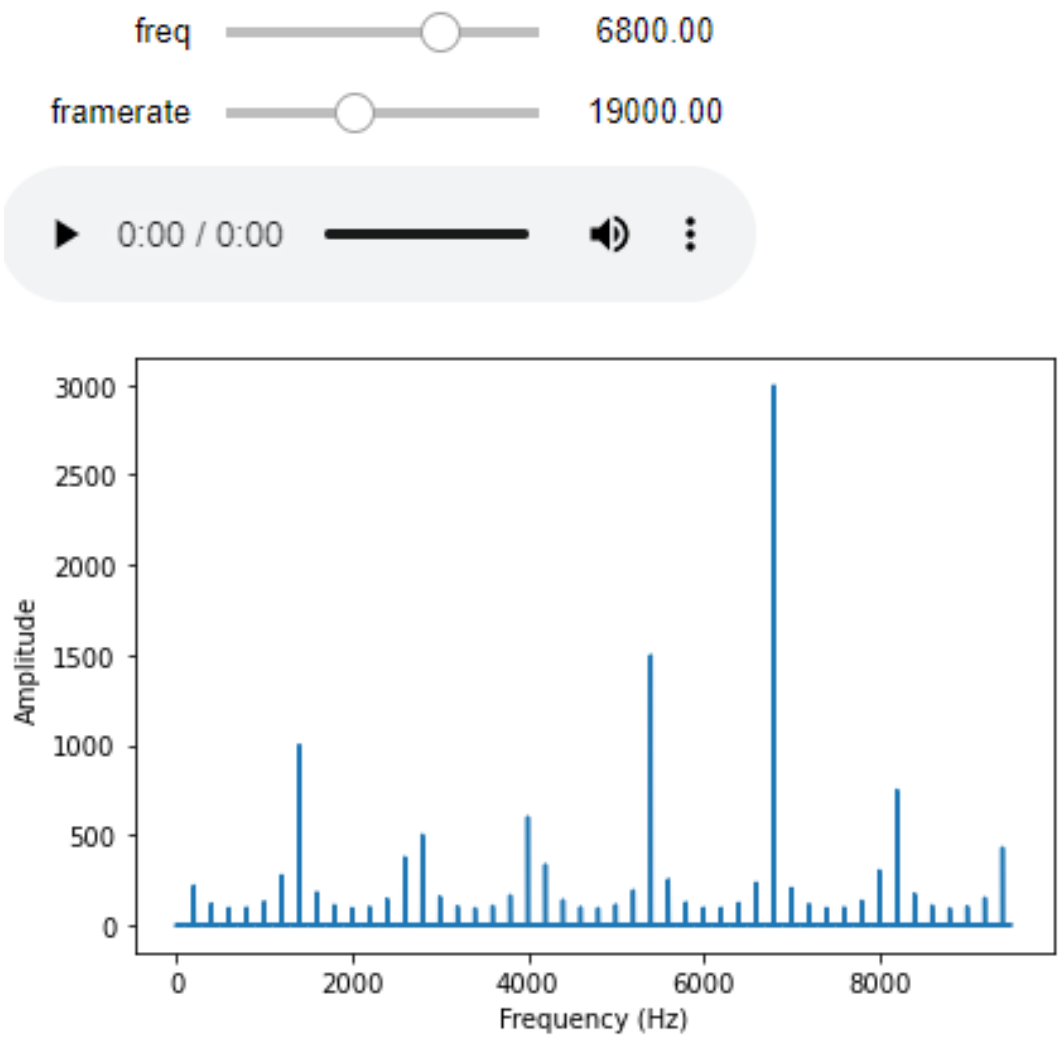


Рис. 3: Изучение и проверка примеров (влияние эффекта `aliasing`).

Все примеры были изучены и запущены.

В ходе выполнения данного упражнения были получены знания по созданию различных видов сигналов и работе с ними с помощью библиотеки для Python. Также было прослушаны звучания этих видов сигналов. Самым приятным из них на слух показался треугольный сигнал. Кроме того был рассмотрен эффект `aliasing`, из-за которого выборки из сигнала с высокой частотой кажутся выборками из сигнала с низкой. Были также рассмотрены примеры работы с фазой.

2 Упражнение 2.2

Во втором упражнении необходимо написать класс `SawtoothSignal`, расширяющий `signal` и предоставляющий `evaluate` для оценки пилообразного сигнала. Затем необходимо вычислить спектр пилообразного сигнала и соотнести его гармоническую структуру с треугольным и прямоугольным сигналами.

Итак, начнём с написания класса `SawtoothSignal`.

```
import numpy
from thinkdsp import decorate, Sinusoid, normalize, unbias, SquareSignal, \
    TriangleSignal, SinSignal

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

```

Листинг 1: Создание класса `SawtoothSignal`.

Данный класс расширяет класс `Sinusoid` (который, в свою очередь, расширяет класс `Signal`) и переопределяет его метод `evaluate`.

Теперь проверим работу нашего класса, создав пилообразный сигнал. Затем преобразуем его к `wave` и выведем график его небольшого сегмента, чтобы увидеть поведение сигнала. Тут же прослушаем полученный сигнал.

```
sawtooth_signal = SawtoothSignal()
sawtooth_wave = sawtooth_signal.make_wave(duration=0.5, framerate=60000)
sawtooth_wave.segment(start=0, duration=sawtooth_signal.period*4).plot()
decorate(xlabel='Time (s)')
sawtooth_wave.make_audio()

```

Листинг 2: Работа с пилообразным сигналом.

Как видно из Рис.4, полученный сигнал действительно пилообразный. Это означает, что написанный нами класс работает корректно. Полученный же звук напоминает резкий и неприятный гудок.

Теперь вычислим спектр полученного пилообразного сигнала.

```
sawtooth_wave.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 3: Вычисление и вывод спектра пилообразного сигнала.

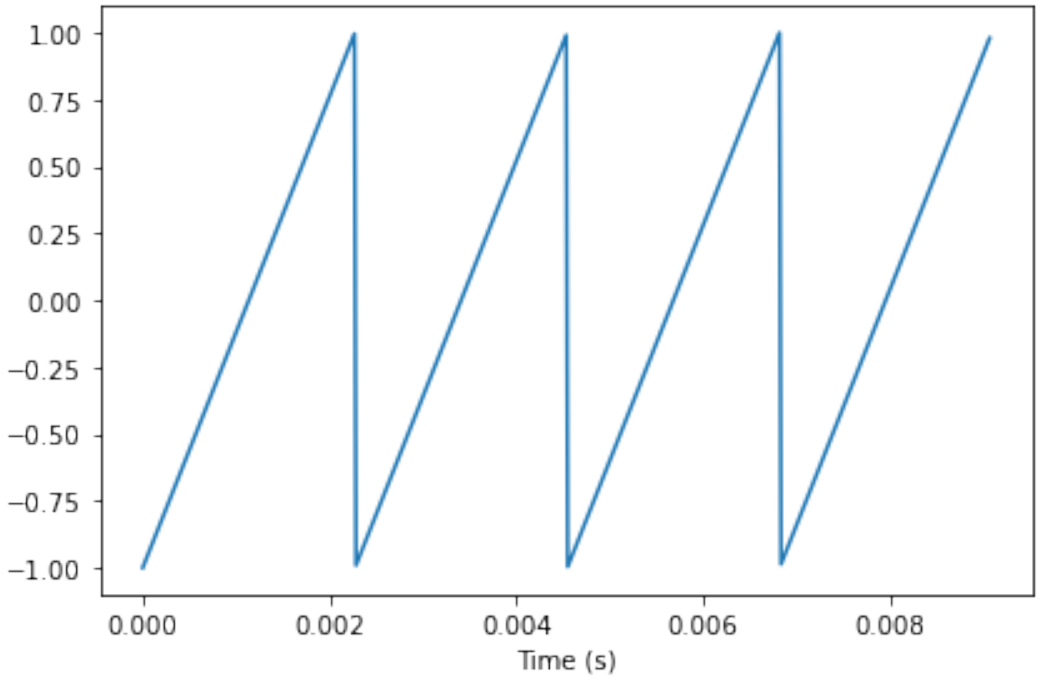


Рис. 4: Сегмент полученного сигнала.

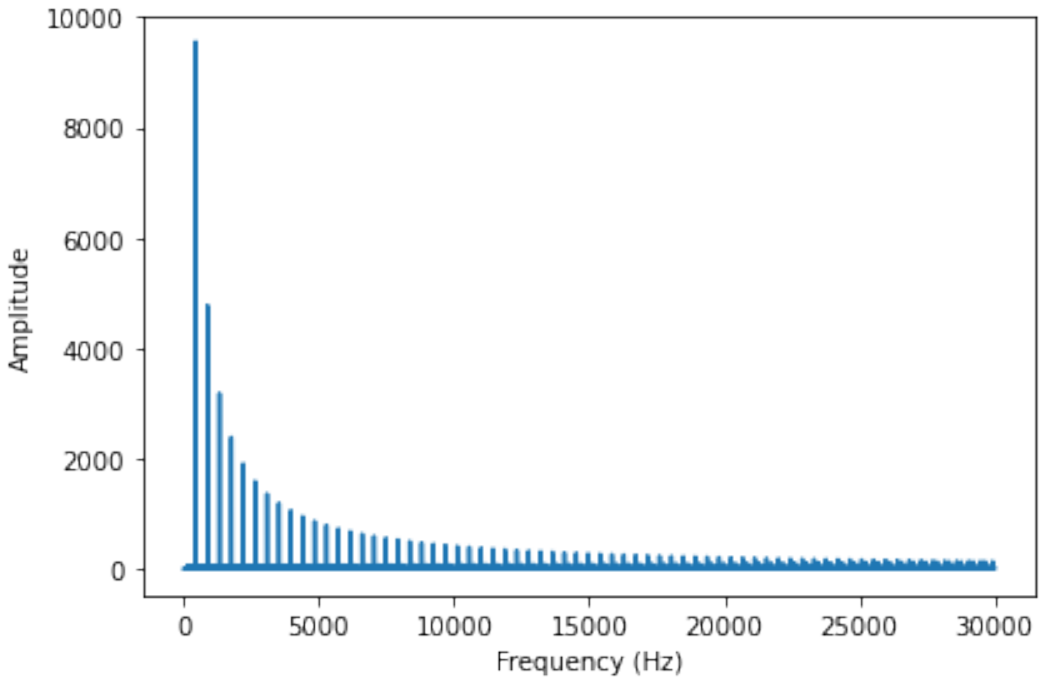


Рис. 5: Спектр пилообразного сигнала.

Далее сопоставим спектры пилообразного и треугольного сигналов. Для этого создадим треугольный сигнал и преобразуем его к `wave`, после чего выведем его спектр вместе со спектром пилообразного сигнала.

```
sawtooth_wave.make_spectrum().plot(color='red')
triangle_wave = TriangleSignal(amp=0.79).make_wave(duration=0.5, framerate=60000)
```

```
triangle_wave.make_audio()
triangle_wave.make_spectrum().plot(color='blue')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 4: Вывод спектров треугольного и пилообразного сигналов.

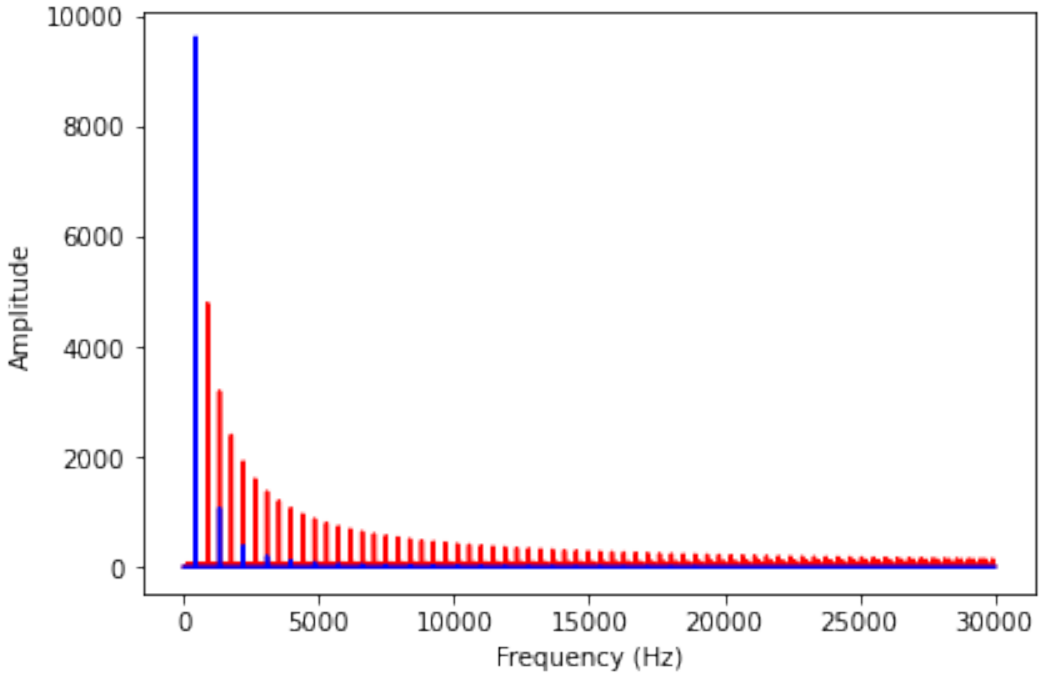


Рис. 6: Спектры пилообразного (красный) и треугольного (синий) сигналов.

Как видно из Рис.6, пилообразный сигнал уменьшается медленнее, чем треугольный. Гармоники треугольного сигнала убывают пропорционально $1/f^2$, а пилообразного как $1/f$. Звук же треугольного сигнала по сравнению с пилообразным более мягкий, без резких высоких частот.

Теперь сравним спектры пилообразного и прямоугольного сигналов.

```
sawtooth_wave.make_spectrum().plot(color='red')
square_wave = SquareSignal(amp=0.5).make_wave(duration=0.5, framerate=60000)
square_wave.make_audio()
square_wave.make_spectrum().plot(color='blue')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 5: Вывод спектров прямоугольного и пилообразного сигналов.

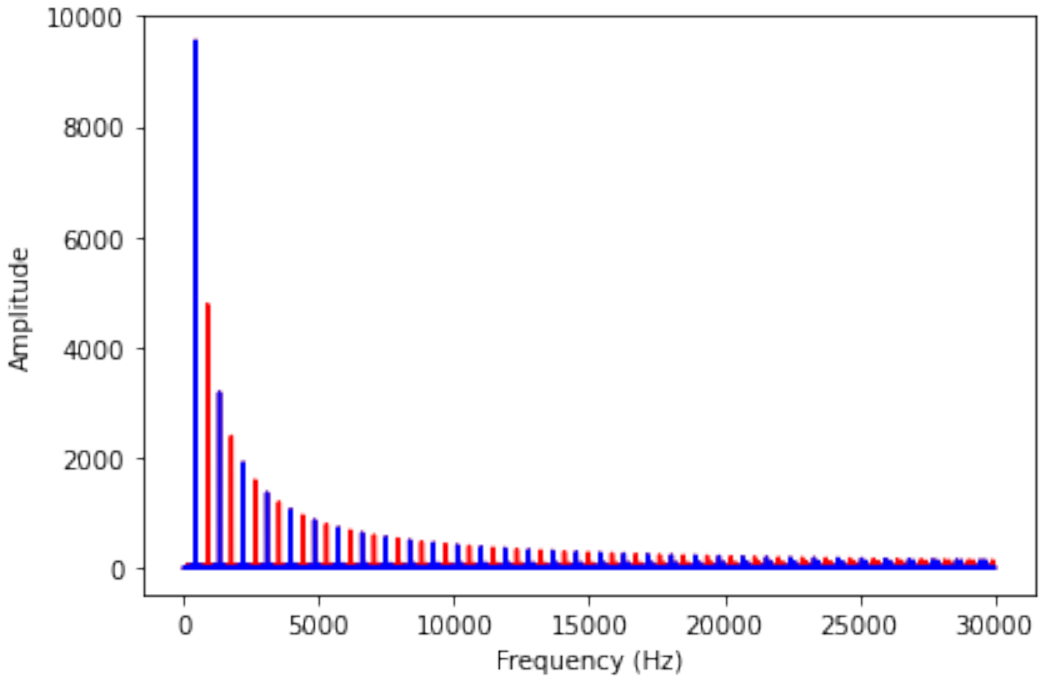


Рис. 7: Спектры пилообразного (красный) и прямоугольного (синий) сигналов.

Как видно из Рис.7, спектры пилообразного и прямоугольного сигналов уменьшаются одинаково, но пилообразный включает в себя чётные и нечётные гармоники (это хорошо заметно при сравнении Рис.7 и Рис.5). Звук же прямоугольного сигнала по сравнению с пилообразным кажется более низким и приглушённым.

В ходе выполнения данного задания был описан класс `SawtoothSignal`, формирующий пилообразный сигнал. Работа класса была проверена, был сделан вывод, что она работает корректно. Затем спектр сформированного пилообразного сигнала был сравнён со спектрами треугольного и прямоугольного сигналов. Был сделан вывод, что гармоники треугольного сигнала убывают как $1/f^2$, а пилообразного как $1/f$. А также, что спектры пилообразного и прямоугольного сигналов уменьшаются одинаково, но пилообразный включает в себя чётные и нечётные гармоники.

3 Упражнение 2.3

В этом упражнении необходимо создать прямоугольный сигнал 1100 Гц, вычислить его `wave` с выборками 10000 кадров в секунду. Затем необходимо построить спектр и убедиться, что большинство гармоник ”завёрнуты”из-за биений.

Итак, начнём с создания прямоугольного сигнала заданной частоты. Затем вычислим его `wave` с выборками 10000 кадров в секунду, после чего выведем его спектр (Рис.8).

```
square_wave = SquareSignal(1100).make_wave(duration=0.5, framerate=10000)
square_wave.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 6: Создание сигнала и получение его `wave`.

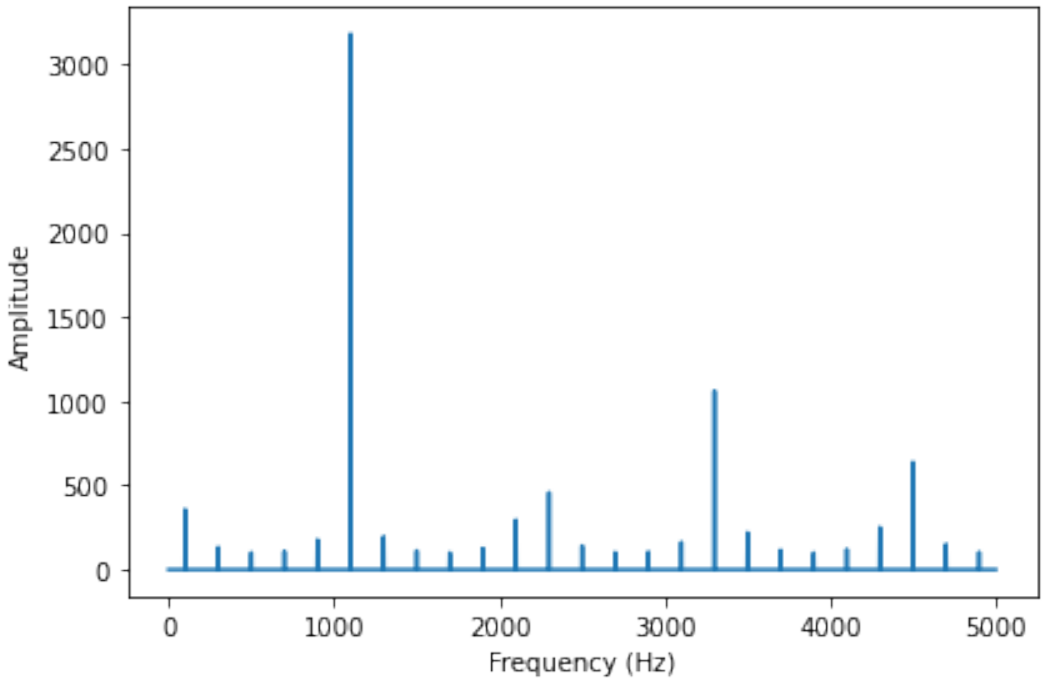


Рис. 8: Спектр прямоугольного сигнала.

Мы уже знаем, что прямоугольный сигнал содержит только нечётные гармоники. Таким образом, ожидалось увидеть гармоники на частотах 1100, 3300, 5500, 7700 и 9900 Гц. А теперь внимательно изучим Рис.8. Как и ожидалось, пики есть на 1100 и 3300 Гц, но третий пик находится на 4500, а не на 5500 Гц. Четвёртый же пик находится на 2300, а не на 7700 Гц, а пятый - на 100, а не на 9900 Гц.

Подобное поведение возникает из-за эффекта биений. Самой высокой частотой, которую можно обработать, могла бы быть 5000 Гц (половина частоты

дискретизации), а в нашем же случае - 3300 Гц. Частоты выше 5000 Гц ”заворачиваются”вокруг 5000 Гц (частота заворота). Именно поэтому мы видим ожидаемые пики на 1100 и 3300 Гц, а все остальные ”заворачиваются”из-за биений.

Теперь узнаем, слышны ли последствия этого эффекта при проигрывании. Для этого сначала преобразуем `wave` нашего сигнала в аудио и прослушаем.

```
square_wave.make_audio()

```

Листинг 7: Преобразование в аудио.

Также мы уже знаем, что воспринимаемые тон и высота звука обычно определяются основной частотой (компонента с самой низкой частотой), в нашем же случае это 100 Гц (”завёрнутая”четвёртая по счёту гармоника). А потому сравним полученный звук с синусоидой 100 Гц.

```
SinSignal(100).make_wave(duration=0.5, framerate=10000).make_audio()

```

Листинг 8: Получение аудио синусоиды 100 Гц.

Конечно, эти два звука не одинаковы. В первом (прямоугольный сигнал) очень заметен высокий звук, во втором (синусоида) же его нет. Однако, можно заметить, что именно фоновый ”гул”с первой записи воспринимается схожим со второй записью.

В ходе выполнения данного упражнения был создан прямоугольный сигнал 1100 Гц, затем вычислен его `wave` с выборками 10000 кадров в секунду. После этого был построен спектр, по которому видно, что большинство гармоник ”завёрнуты”из-за биений.

Кроме того, созданный сигнал был преобразован в аудио и сравнён с аудио синусоиды 100 Гц. При их прослушивании слышно, что полученные аудио сами по себе и не воспринимаются одинаковыми (или хотя бы похожими), однако фоновый звук первого аудио похож на аудио синусоиды. Таким образом, можно сделать вывод, что в данном примере последствия эффекта биения хоть и слышны при проигрывании, однако влияют не очень сильно.

4 Упражнение 2.4

В данном упражнении осуществляется работа с объектом `Spectrum`. При наличии объекта `Spectrum` мы можем вывести несколько первых значений `spectrum.fs`. В таком случае мы увидим, что они начинаются с нуля, то есть `spectrum.hs[0]` - амплитуда компоненты с частотой 0. Но что это значит? Данное упражнение направлено на получения ответа на этот вопрос и подразумевает проведение эксперимента со `Spectrum`.

Итак, для начала создадим треугольный сигнал частотой 440 Гц, получим и выведем его `wave`.

```
triangle_wave = TriangleSignal(440).make_wave(duration=0.01)
triangle_wave.plot()
decorate(xlabel='Time (s)')

```

Листинг 9: Создание сигнала и получение`wave`.

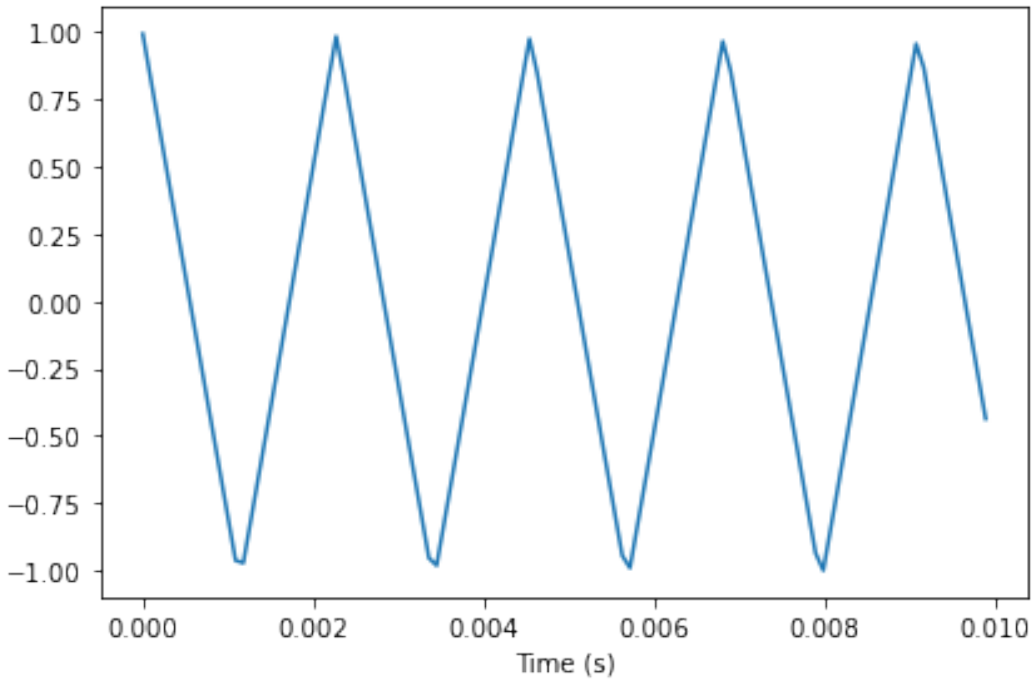


Рис. 9: `Wave` треугольного сигнала.

Теперь получим спектр и выведем `spectrum.hs[0]`.

```
triangle_spectrum = triangle_wave.make_spectrum()
triangle_spectrum.hs[0]

```

Листинг 10: Получение спектра и вывод `spectrum.hs[0]`.

Было получено значение первого элемента спектра (1.0436096431476471e-14+0j). Это комплексное число, близкое к 0. Это значение соответствует компо-

ненте нулевой частоты: размах пропорционален амплитуде компоненты, а угол - это фаза.

Теперь установим `spectrum.hs[0] = 100` и проверим, как это повлияет на сигнал. Для этого преобразуем изменённый спектр к `wave` и сравним его с неизменённым `wave`.

```
triangle_spectrum.hs[0] = 100
triangle_wave.plot(color='blue')
triangle_spectrum.make_wave().plot(color='red')
decorate(xlabel='Time (s)')

```

Листинг 11: Изменение первого элемента спектра и сравнение с первоначальным.

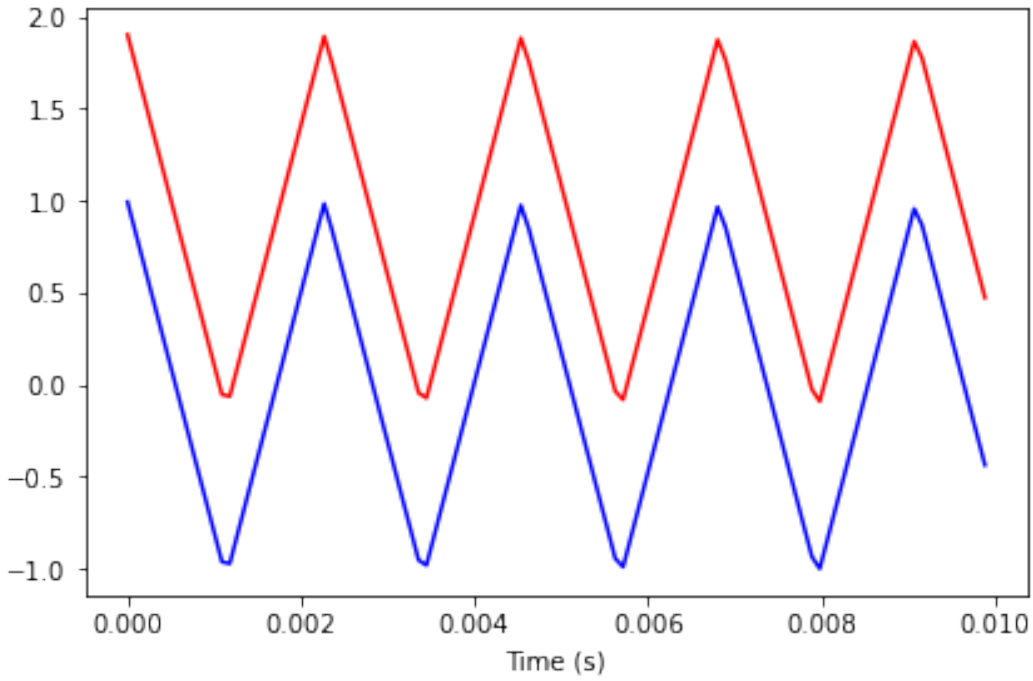


Рис. 10: Сравнение изменённого (красный) и неизменённого (синий) сигналов.

При сравнении можно заметить, что `wave` вертикально сместилась вверх. Такое изменение объясняется тем, что компонент нулевой частоты постоянен для всех значений в сигнале. Если сигнал несмещён, то компонент нулевой частоты будет равен 0. Таким образом, изменив компоненту нулевой частоты, мы сместили весь сигнал.

В ходе выполнения данного упражнения был проведён эксперимент со `Spectrum`. В ходе эксперимента была получена и изменена компонента нулевой частоты спектра треугольного сигнала. Это привело к смещению `wave` сигнала. Был сделан вывод, что изменение нулевой компоненты частоты влияет на все значения сигнала.

5 Упражнение 2.5

В данном упражнении необходимо написать функцию, принимающую в качестве параметра `Spectrum` и изменяющую его делением каждого элемента `hs` на соответствующую частоту из `fs`. Затем необходимо протестировать написанную функцию.

Сначала напишем функцию. Не забываем, что на нуль делить не стоит, а потому компоненту нулевой частоты обрабатываем отдельно.

```
def change_spectrum(spectrum):
    spectrum.hs[0] = 0
    spectrum.hs[1:] /= spectrum.fs[1:]

```

Листинг 12: Функция, изменяющая спектр.

Теперь протестируем нашу функцию на пилообразном сигнале. Для этого сначала создадим сигнал, после чего вычислим и распечатаем его спектр. Тут же получим аудио нашего сигнала для последующего сравнения.

```
sawtooth_wave = SawtoothSignal().make_wave(duration=0.5)
sawtooth_wave.make_audio()

sawtooth_spectrum = sawtooth_wave.make_spectrum()
sawtooth_spectrum.plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 13: Создание пилообразного сигнала и получение его спектра.

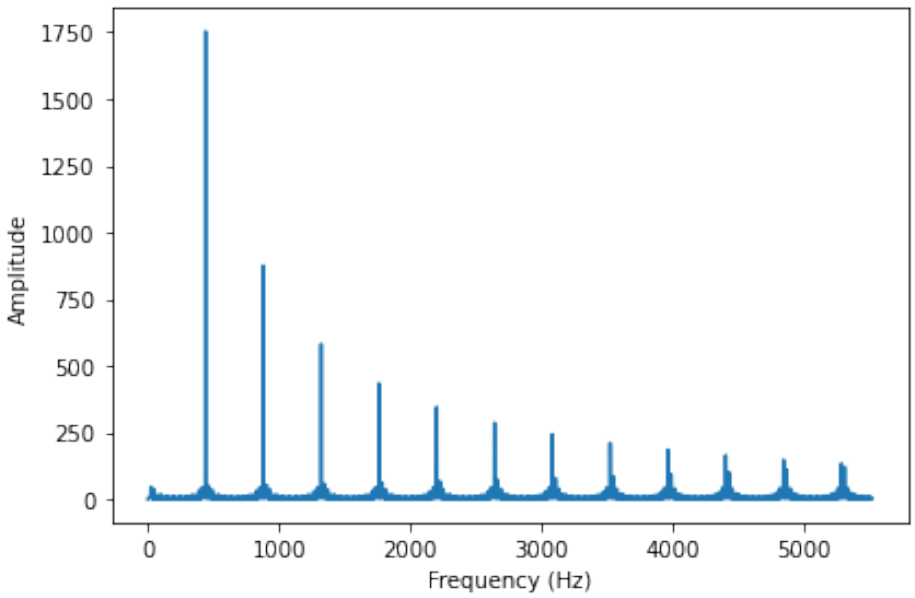


Рис. 11: Спектр пилообразного сигнала.

Теперь изменим спектр с помощью нашей функции и сравним его с первоначальным спектром.

```
sawtooth_spectrum.plot(color='blue')
change_spectrum(sawtooth_spectrum)
sawtooth_spectrum.scale(440)
sawtooth_spectrum.plot(color='red')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 14: Изменение спектра и его сравнине с первоначальным спектром.

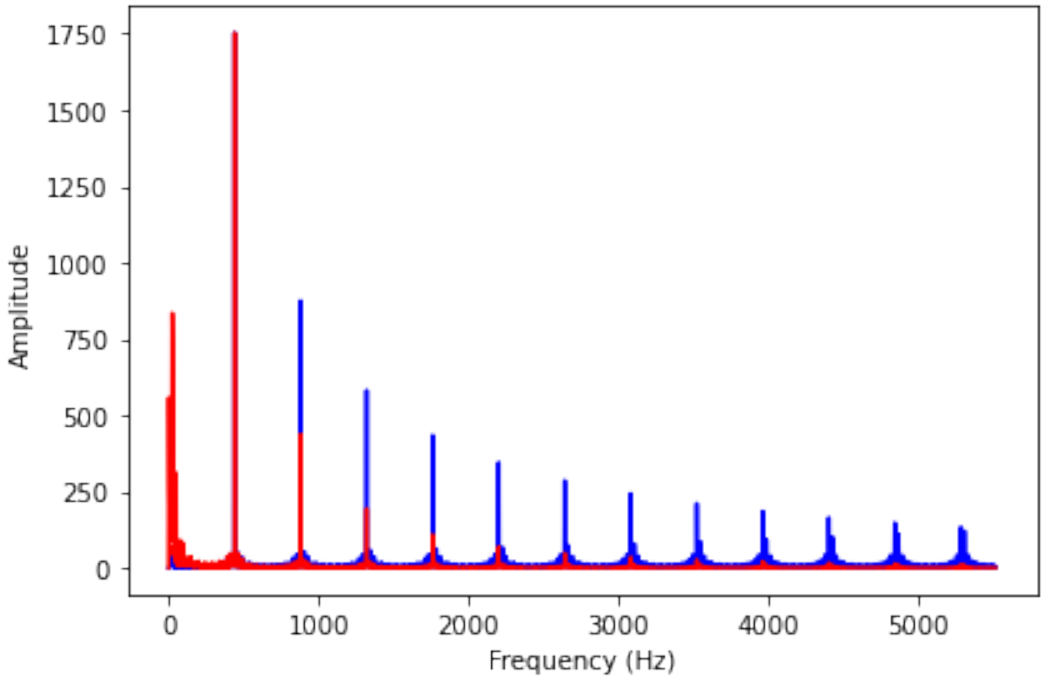


Рис. 12: Сравнение изменённого спектра (красный) с первоначальным (синий).

Ну и наконец получим аудио из изменённого спектра для сравнения с первоначальным.

```
sawtooth_spectrum.make_wave().make_audio()

```

Листинг 15: Получение аудио.

Полученный сигнал более приглушённый и имеет явно выраженные звуки ”бульканья”на фоне.

В ходе данного упражнения была написана функция, принимающая `Spectrum` как параметр и изменяющая его делением каждого элемента `hs` на соответствующую частоту из `fs`. Затем эта функция была протестирована на пилообразном сигнале. При сравнении звучаний первоначального и изменённого сигналов было замечено, что изменённый сигнал как будто имеет более плохое качество: он более тихий и имеет явно выраженные звуки ”бульканья”на фоне.

6 Упражнение 2.6

В этом упражнении предлагается составить сигнал, состоящий из чётных и нечётных гармоник, спадающих пропорционально $1/f^2$.

Попробуем взять сигнал с необходимым спектром и изменить его параметры для получения необходимого результата. Для этого нам подойдёт пилообразный сигнал.

Итак, создадим сигнал, получим его `wave` и аудио.

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

```

Листинг 16: Создание пилообразного сигнала и получение его `wave`.

Теперь вычислим его спектр и выведем его.

```
sawtooth_spectrum = sawtooth_wave.make_spectrum()
sawtooth_spectrum.plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 17: Вычисленине и вывод спектра.

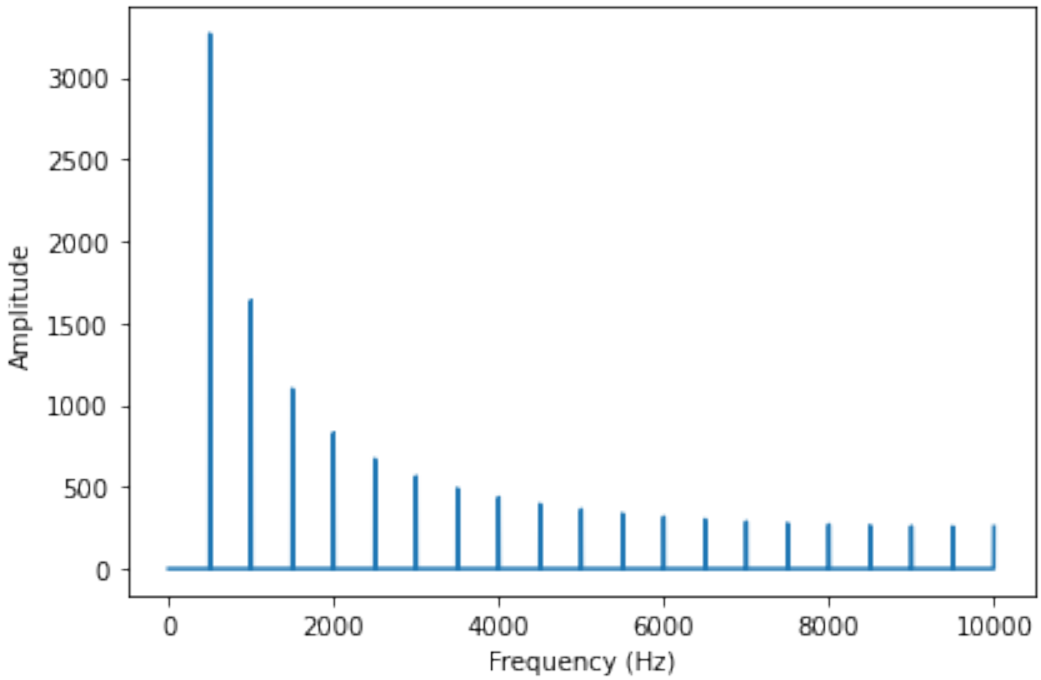


Рис. 13: Спектр пилообразного сигнала.

Как видно из спектра (Рис.13), у нас уже имеются все необходимые гармоники. Осталось только изменить спад гармоник с $1/f$ на $1/f^2$. Для этого можно использовать написанную в прошлом упражнении функцию `change_spectrum()`.

Итак, изменим спад гармоник и сравним изменённый спектр с первоначальным.

```
sawtooth_spectrum.plot(color='blue')
change_spectrum(sawtooth_spectrum)
sawtooth_spectrum.scale(400)
sawtooth_spectrum.plot(color='red')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')

```

Листинг 18: Изменение спада гармоник и сравнение спектров.

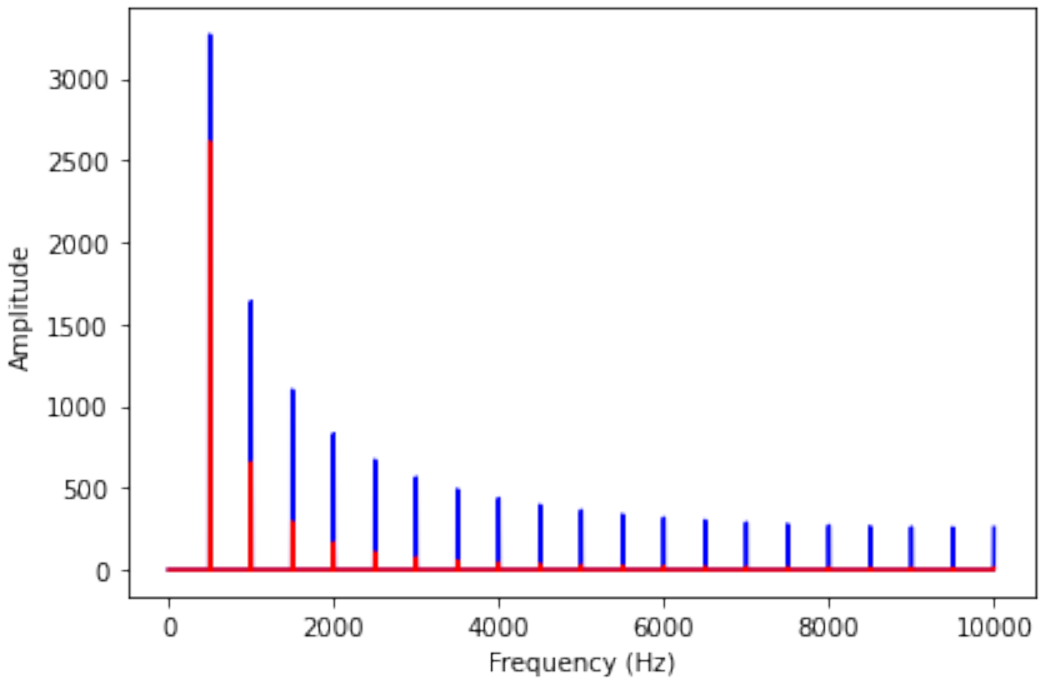


Рис. 14: Сравнение спектров пилообразного и полученного сигналов.

Как видно из Рис.14, теперь гармоники спадают как $1/f^2$. Преобразуем изменённый спектр в `wave`, выведем его сегмент и получим аудио.

```
sawtooth_wave_new = sawtooth_spectrum.make_wave()
sawtooth_wave_new.segment(duration=0.01).plot()
decorate(xlabel='Time (s)')
sawtooth_wave_new.make_audio()

```

Листинг 19: Получение аудио и `wave`.

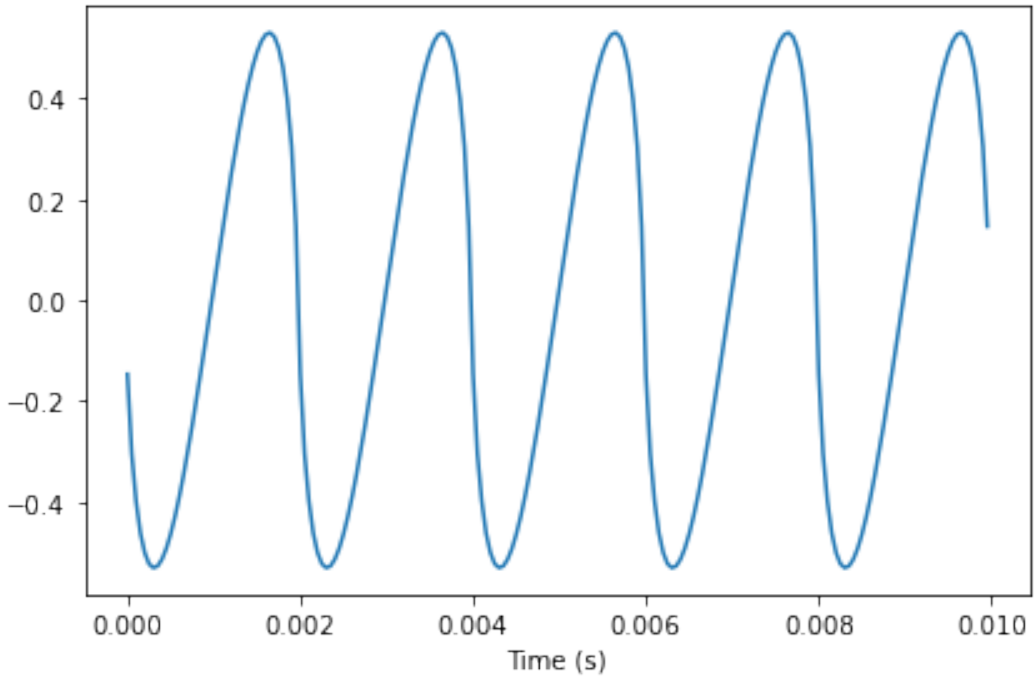


Рис. 15: Сегмент полученного сигнала.

Полученный сигнал имеет как будто промежуточный вид между синусоидой и пилообразным сигналом. При этом звучит он гораздо мягче и приятнее, чем первоначальный пилообразный сигнал.

В ходе выполнения данного упражнения на основе пилообразного сигнала был создан сигнал, состоящий из чётных и нечётных гармоник, спадающих пропорционально $1/f^2$. Полученный сигнал звучит мягче и приятнее пилообразного сигнала.

7 Выводы

В ходе выполнения данной лабораторной работы были изучены спектры, гармоники и некоторые виды сигналов: треугольный, прямоугольный и пилообразный. Был создан и протестирован класс `SawtoothSignal`, формирующий пилообразный сигнал. Также был изучен и эффект биения, из-за которого происходит ”заворот”гармоник. Кроме того, было изучено влияние компоненты нулевой частоты на сигнал. Затем была написана и протестирована функция, изменяющая спектр. И наконец, был составлен сигнал согласно требованиям.