Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

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

Работу выполнил:
Ляшенко В.В.
Группа: 3530901/80201
Преподаватель:
Богач Н.В.

Санкт-Петербург
2021

Оглавление

Список иллюстраций

Листинги

Глава 1

Свойства преобразования Фурье

1.1 Преобразование Фурье

Преобразование Фурье – интегральное преобразование, которое является инструментом спектрального анализа непериодических сигналов. Для периодических сигналов используется дискретное преобразование Фурье.

Прямым преобразованием Фурье функции $f(t)$ называется следующая функция (при условии, что интеграл сходится):

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{1.1}$$

Обратное преобразование в этом случае задается формулой:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega \tag{1.2}$$

1.2 Свойства

1.2.1 Линейность

Преобразование Фурье относится к числу линейных интегральных операций, т.е. спектр суммы сигналов равен сумме спектров этих сигналов.

$$\sum_i a_i f_i(t) \Leftrightarrow \sum_i a_i F_i(\omega) \tag{1.3}$$

1.2.2 Смещение

При смещении функции по аргументу на t_0 её ПФ умножается на $e^{j\omega t}$.

$$F(\omega) = \int_{-\infty}^{\infty} f(t+t_0)e^{-j\omega t}dt = \int_{-\infty}^{\infty} f(t+t_0)e^{-j\omega(t+t_0)}d(t+t_0)e^{-j\omega(-t_0)} = e^{j\omega t_0}S(\omega) \tag{1.4}$$

1.2.3 Изменение масштаба

Если аргумент t функции $f(t)$ заменить на at, где a постоянный коэффициент, то ПФ функции с $F(\omega)$ изменится на $\frac{1}{|a|}F(\frac{\omega}{a})$.

$$F(\omega) = \int_{-\infty}^{\infty} f(at)e^{-j\omega t}dt = \frac{1}{a}\int_{-\infty}^{\infty} f(at)e^{-j\frac{\omega}{a}at}d(at) = \frac{1}{|a|}F(\frac{\omega}{a}) \tag{1.5}$$

Появление модуля коэффициента a вызвано тем, что при отрицательном значении коэффициента замена переменной приводит к изменению знаков у пределов интегрирования.

Из равенства следует, что сжатие функции $f(t)$ по времени в a приводит к расширению по частоте в a соответствующего ПФ и наоборот - расширение функции приводит к сжатию ПФ.

1.2.4 Дифференцирование

При дифференцировании функции $f(t)$ её ПФ умножается на $j\omega$.

Для доказательства используем определение понятия производной:

$$f(t) = \frac{df}{dt} = \lim \frac{f(t+\varepsilon) - f(t)}{\varepsilon} \tag{1.6}$$

Применим к этому выражению ПФ:

$$F(\omega) = \int_{-\infty}^{\infty} \lim_{\varepsilon \to 0} \frac{f(t+\varepsilon) - f(t)}{\varepsilon} e^{-j\omega t} dt = \lim_{\varepsilon \to 0} \frac{S(\omega)e^{j\omega\varepsilon} - S(\omega)}{\varepsilon} = S(\omega) \lim_{\varepsilon \to 0} \frac{e^{j\omega\varepsilon} - 1}{\varepsilon} = j\omega S(\omega) \tag{1.7}$$

При дифференцировании низкие частоты ослабляются, а высокие усиливаются. Фазовый спектр сигнала сдвигается на 90°для положительны частот и на -90°для отрицательных.

1.2.5 Интегрирование

Интегрирование является операцией обратной дифференцированию. Логично предположить, что при интегрировании функции $f(t)$ её ПФ делится на $j\omega$. Но это утверждение справедливо только для сигналов, не имеющих постоянной составляющей.

$$S(0) = \int_{-\infty}^{\infty} f(t)dt = 0 \tag{1.8}$$

В общем случае результат должен содержать дополнительное слагаемое в виде дельта-функции на нулевой частоте.

$$F(\omega) = \frac{S(\omega)}{j\omega} + \pi S(0)\delta(\omega) \tag{1.9}$$

При интегрировании исходного сигнала высокие частоты ослабляются, а низкие усиливаются. Фазовый спектр сигнала сдвигается на -90°для положительны частот и на 90°для отрицательных.

1.2.6 Свертывание

ПФ свертки двух функций равно произведению ПФ свертываемых функций.

$$f(t) = \int_{-\infty}^{\infty} s(f')g(t-t')dt' \tag{1.10}$$

Подвергнем такую конструкцию ПФ.

$$F(\omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(f')g(t-t')dt'e^{-j\omega t}dt = \int_{-\infty}^{\infty} s(f')e^{-j\omega t'}\int_{-\infty}^{\infty} g(t-t')e^{-j\omega(t-t')}d(t-t')dt' = S(\omega)G(\omega) \tag{1.11}$$

1.2.7 Произведение

Спектр произведения является сверткой спектров. Докажем это.

$$f(t) = s(t)g(t) \tag{1.12}$$

Тогда:

$$F(\omega) = \int_{-\infty}^{\infty} s(t)g(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty} \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega')e^{j\omega' t}d\omega' \right) g(t)e^{-j\omega t}dt =$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega') \int_{-\infty}^{\infty} g(t)e^{-j(\omega-\omega')t}dtd\omega' = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega')G(\omega-\omega')d\omega' \tag{1.13}$$

Глава 2

Упражнение 2.1

В начале мы должны для Jupyter загрузить `chap02.ipynb`, прочитать пояснения и запустить примеры.

Все примеры были успешно запущены. В последнем примере при низких значений `freq` звук похож на гудение, а при высоких - на скрежет. При изменениях параметра `framerate` звук получался более приглушённым или наоборот более слышимым (Рис.2.1).

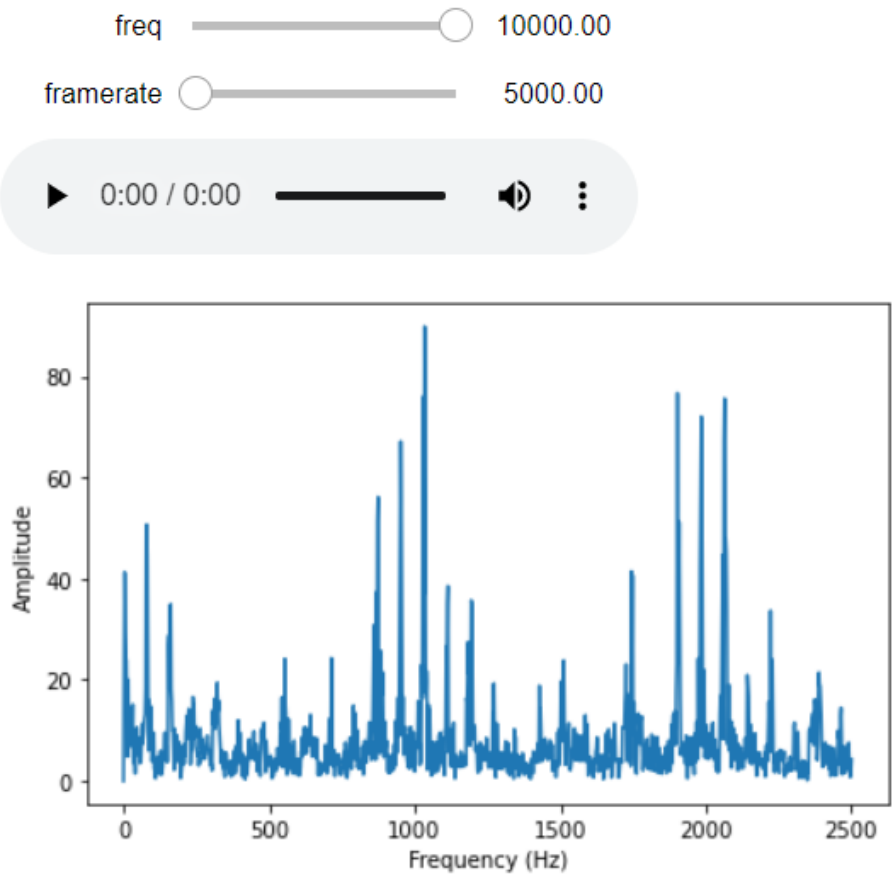

Рис. 2.1: Использование интерактивных виджетов IPython

Глава 3

Упражнение 2.2

3.1 Пилообразный сигнал

Напишем класс `SawtoothSignal`, расширяющий `Sinusoid` и предоставляющий `evaluate` для оценки пилообразного сигнала.

```
from thinkdsp import Sinusoid, normalize, unbias, PI2
import numpy as np
class SawtoothSignal(Sinusoid):

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

Листинг 3.1: Класс пилообразного сигнала

Проверим, что сигнал генерируется правильно (Рис.3.1).

```
signal = SawtoothSignal()
sawtooth_wave = signal.make_wave(signal.period*5, framerate=10000)
sawtooth_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.2: Генерация пилообразного сигнала

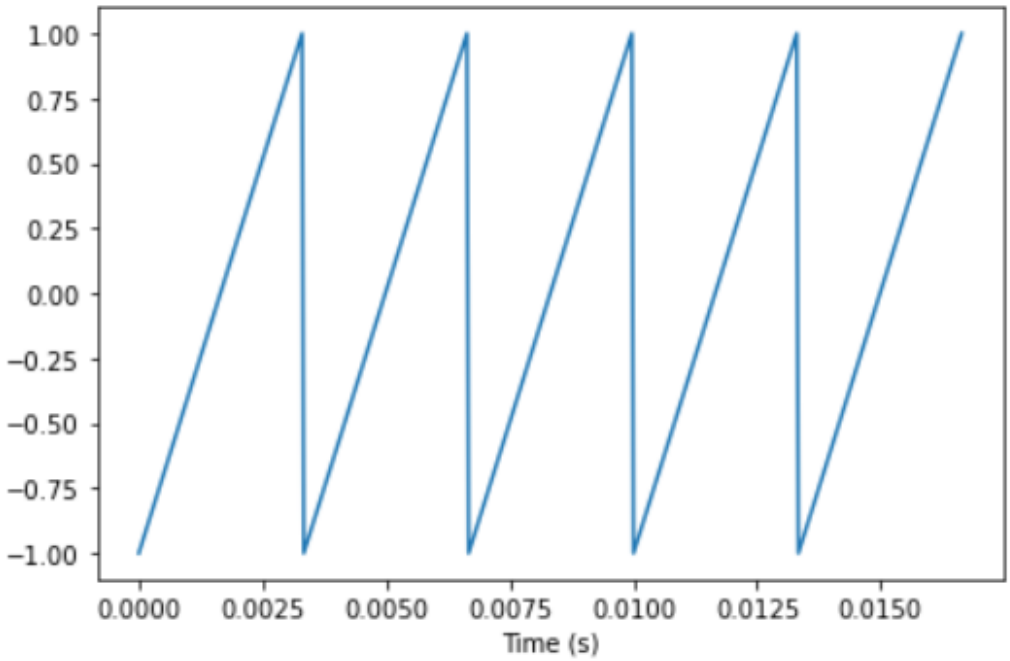

Рис. 3.1: Пилообразный сигнал

3.2 Спектр пилообразного сигнала

Теперь вычислим спектр пилообразного сигнала (Рис.3.2).

```
sawtooth_wave = signal.make_wave(duration=0.5, framerate=40000)
sawtooth_wave.apodize()
spectrum = sawtooth_wave.make_spectrum()
spectrum.plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.3: Вычисление спектра пилообразного сигнала

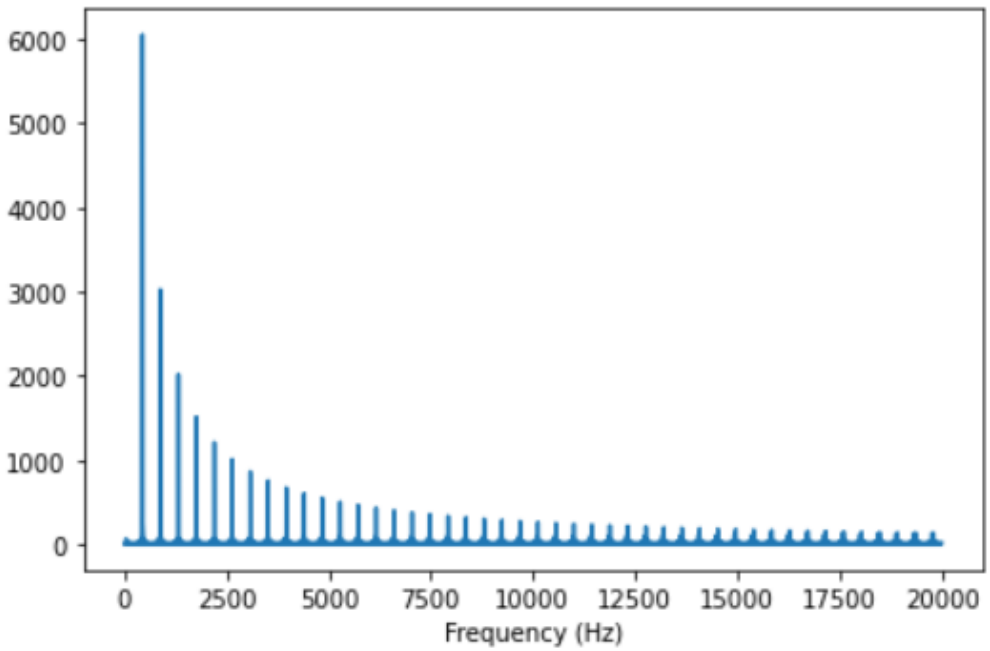

Рис. 3.2: Спектр пилообразного сигнала

Сравним гармоническую структуру пилообразного сигнала с треугольным и прямоугольным сигналами.

Сначала сравним с треугольным сигналом.

```
from thinkdsp import TriangleSignal

sawtooth_wave.make_spectrum().plot(color='gray')
triangle = TriangleSignal(amp=0.79).make_wave(duration=0.5, framerate=40000)
triangle.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.4: Сравнение с треугольным сигналосм

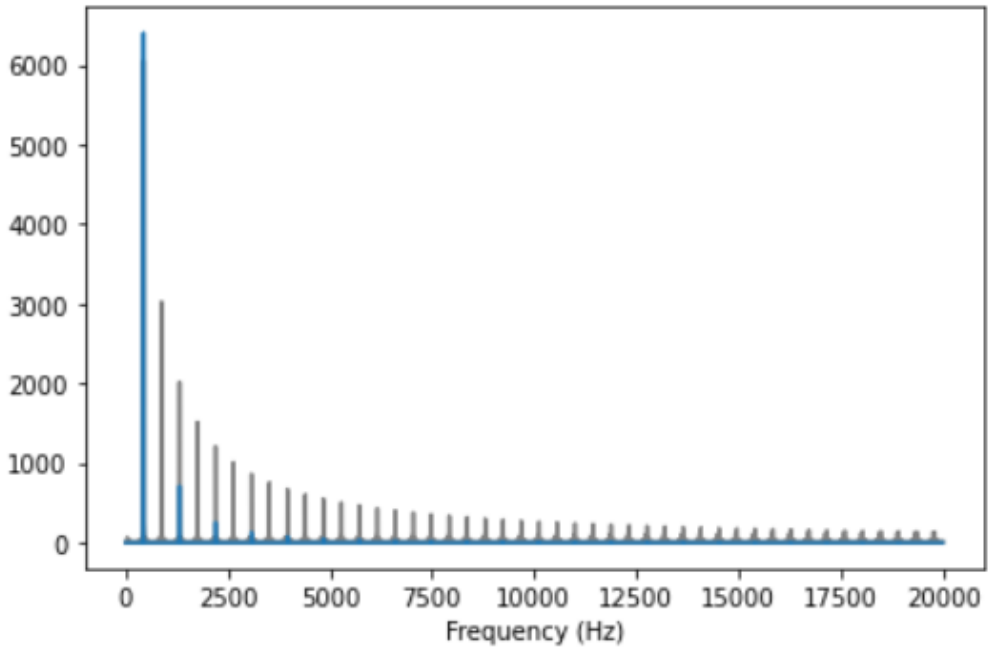

Рис. 3.3: Спектры треугольного и пилообразного сигналов

Из рис.3.3 видно, что пилообразный сигнал снижается медленнее, чем треугольный. Теперь посмотрим на прямоугольный сигнал.

```
from thinkdsp import SquareSignal

sawtooth_wave.make_spectrum().plot(color='gray')
square = SquareSignal(amp=0.5).make_wave(duration=0.5, framerate=40000)
square.make_spectrum().plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 3.5: Сравнение с прямоугольным сигналосм

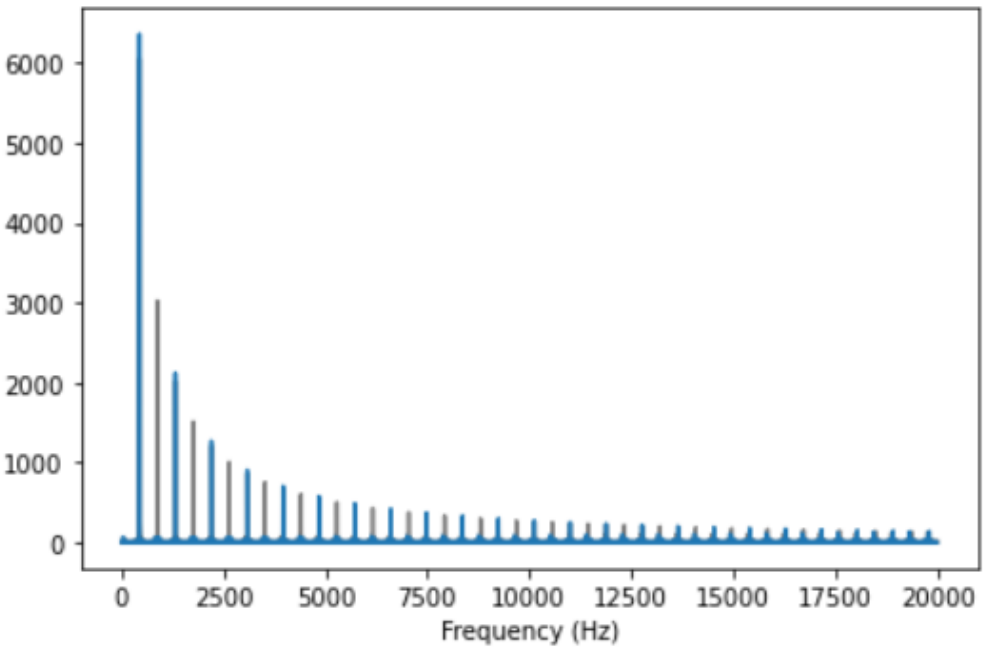

Рис. 3.4: Спектры прямоугольного и пилообразного сигналов

Из рис.3.4 видно, что пилообразный так же, как и прямоугольный, но у пилообразного есть и чётные, и нечётные гармоники.

Глава 4

Упражнение 2.3

Создадим прямоугольный сигнал 1100 Гц и вычислим `wave` с выборками 10000 кадров в секунду. Затем построим спектр этого сигнала.

```
from thinkdsp import SquareSignal

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

Листинг 4.1: Создание прямоугольного сигнала 1100 Гц

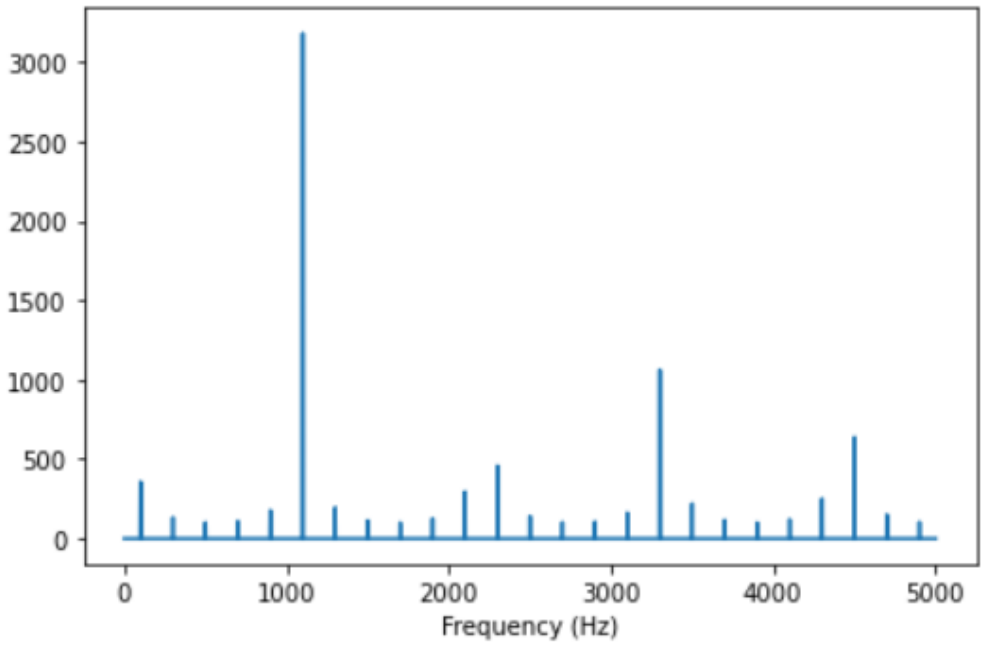

Рис. 4.1: Спектр прямоугольного сигнала

Из рис.4.1 видно, что основная гармоника находится на частоте 1100 Гц и первая на - 3300 Гц. Это верно. Но вот вторая гармоника, которая должна быть на 5500 Гц, располагается на 4500 Гц. Третья гармоника также не на месте. Она должна быть на частоте 7700 Гц,но находится на 2400 Гц. И т.д.

Таким образом, мы убедились, что гармоники "завернуты"из-за биения.

Воспроизведём полученный сигнал.В получившимся звуке мы можем услышать гармоники биения. Основной тон, который мы воспринимаем, является гармоникой биения на частоте 200 Гц.

Для того, чтобы в этом убедиться, сравним с с синусоидой 200 Гц.

```
from thinkdsp import SinSignal

SinSignal(200).make_wave(duration=0.5, framerate=10000).make_audio()
```

Листинг 4.2: Создание синусоиды 200 Гц

Глава 5

Упражнение 2.4

5.1 Треугольный сигнал

Проведём следующий эксперемент. Возьмём треугольный сигнал с частотой 440 Гц и `wave` длительностью 0,01 секунд. Выведем этот сигнал (Рис.5.1).

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

Листинг 5.1: Создание треугольного сигнала 440 Гц

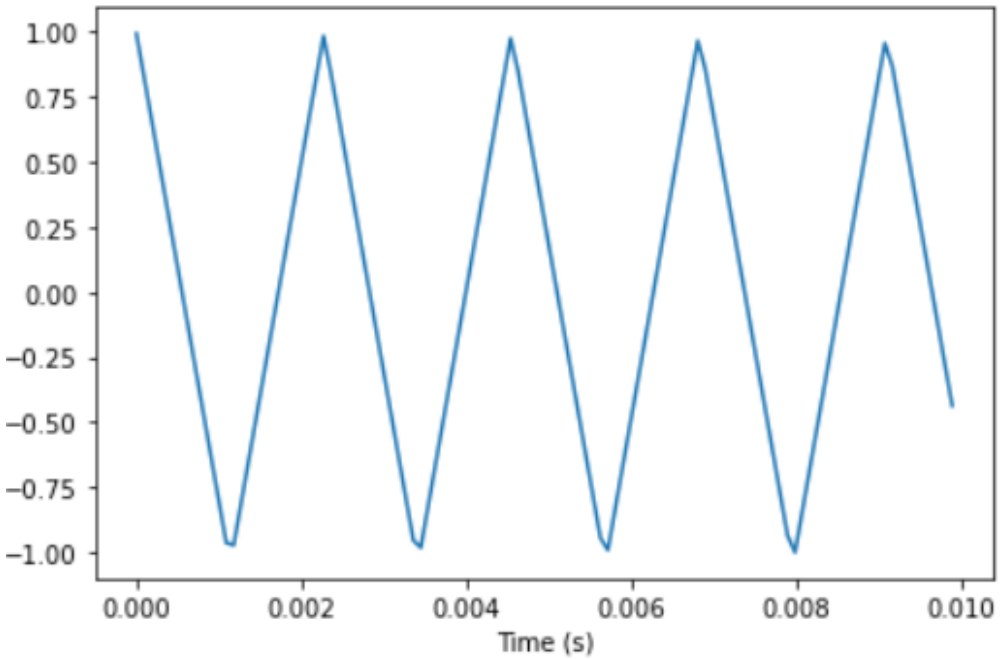

Рис. 5.1: Треугольный сигнал

5.2 Объект Spectrum

Создадим объект `Spectrum` и распечатаем `Spectrum.hs[0]`.

```
spectrum = triangle_wave.make_spectrum()
```

```
spectrum.hs[0]
```

Листинг 5.2: Вывод нулевой компоненты

В результате мы получаем `(1.0436096431476471e-14+0j)`. Это значение соответствует частотной компоненте: его размах пропорционален амплитуде соответствующей компоненты, а угол в степени числа e – это фаза.

5.3 Изменение компоненты

Установим `Spectrum.hs[0] = 100`.

```
triangle_wave.plot(color='gray')
spectrum.hs[0] = 100
spectrum.make_wave().plot()
decorate(xlabel='Time (s)')
```

Листинг 5.3: Изменение компоненты

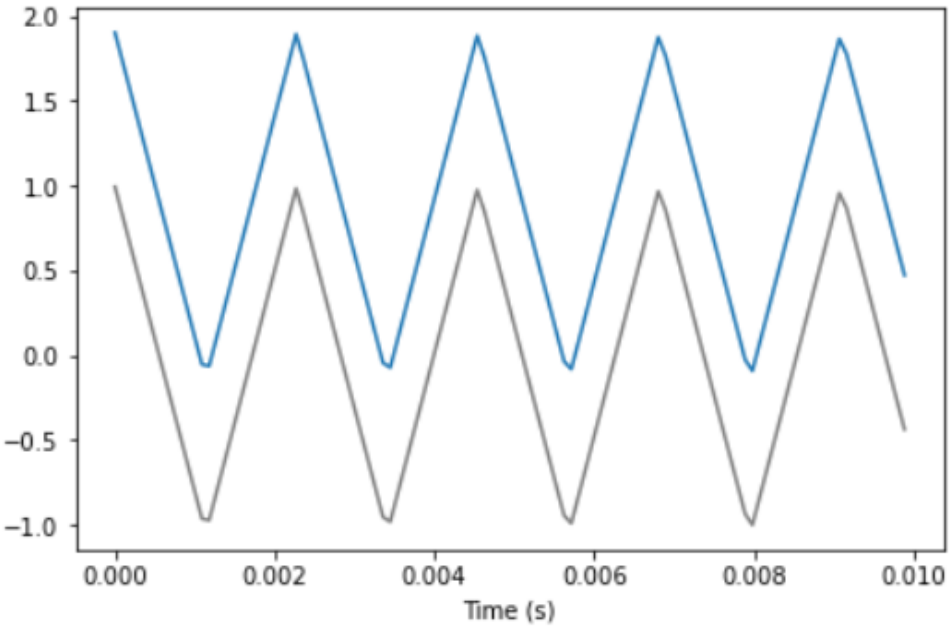

Рис. 5.2: Сравнение сигналов

Как можно видеть на рис.5.2 сигнал сместился вверх.

Глава 6

Упражнение 2.5

6.1 Создание функции

Напишем функцию, принимающую `Spectrum` как параметр и изменяющую его делением каждого элемента `hs` на соответствующую частоту `fs`.

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

Листинг 6.1: Функция изменения спектра

6.2 Использование функции

Проверим эту функцию, используя прямоугольный сигнал. Для этого вычислим `Spectrum` и распечатаем его (Рис.6.1).

```
wave = SquareSignal(freq=440).make_wave(duration=0.5)
spectrum = wave.make_spectrum()
spectrum.plot()
wave.make_audio()
```

Листинг 6.2: Вычисление спектра

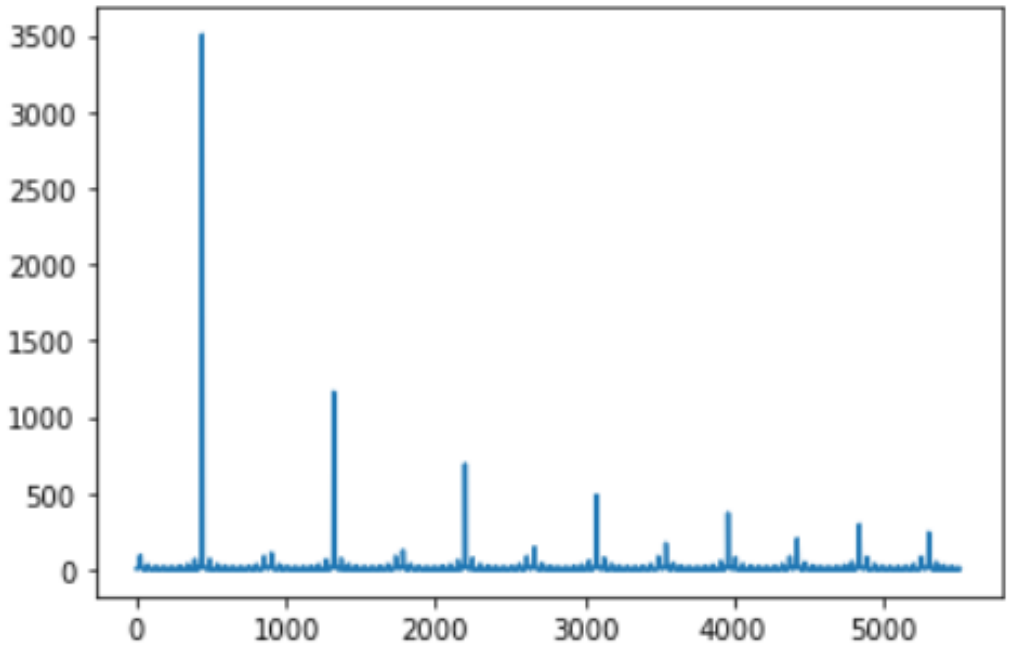

Рис. 6.1: Спектр прямоугольного сигнала

Изменим `Spectrum`, используя нашу функцию, и распечатаем его (Рис.6.2).

```
spectrum.plot(color='gray')
change_spectrum(spectrum)
spectrum.scale(440)
spectrum.plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 6.3: Вычисление нового спектра

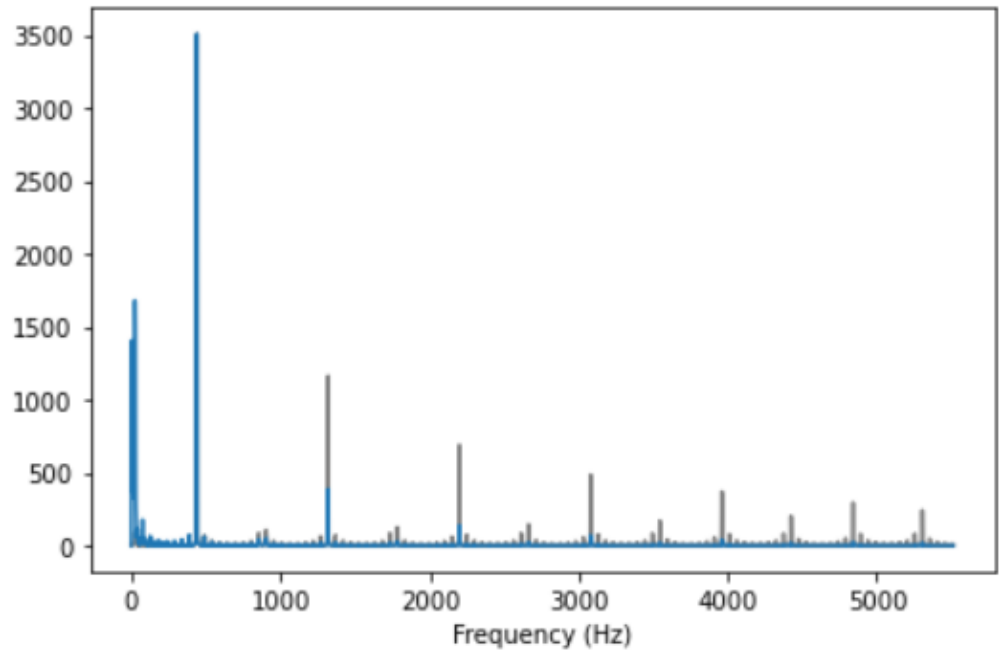

Рис. 6.2: Сравнение спектров

Используем `Spectrum.make_wave`, чтобы сделать `wave` из измененного `Spectrum` и послушаем его.

```
spectrum.make_wave().make_audio()
```

Листинг 6.4: Воспроизведение сигнала

Получившийся сигнал стал более приглушённый.

Глава 7

Упражнение 2.6

Составим сигнал, который будет состоять из чётных и нечётных гармоник, спадающих пропорционально $1/f^2$. Возьмём пилообразный сигнал, в котором есть все необходимые нам гармоники. Его гармоники спадают пропорционально $1/f$ (Рис.7.1).

```
signal = SawtoothSignal(500)
wave = signal.make_wave(duration=0.5, framerate=40000)
spectrum = wave.make_spectrum()
spectrum.plot()
decorate(xlabel='Frequency (Hz)')
wave.make_audio()
```

Листинг 7.1: Создание пилообразного сигнала

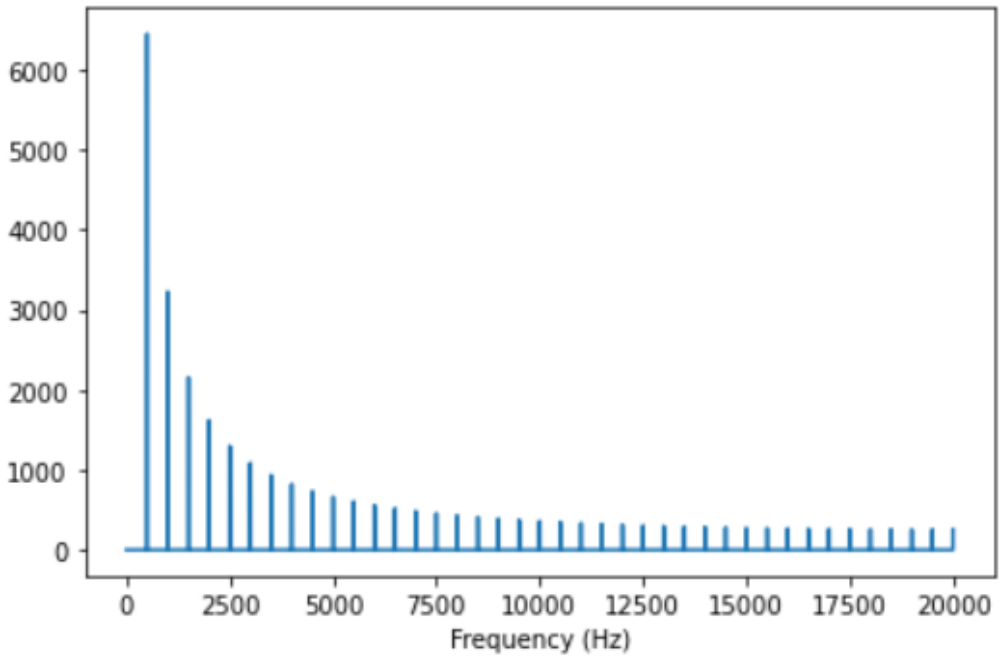

Рис. 7.1: Спектр пилообразного сигнала

Теперь используем функцию для изменения спектра, написанную в Упражнении 2.6. Если мы применим эту функцию, то мы можем заставить гармоники спадать как $1/f^2$ (Рис.7.2).

```
spectrum.plot(color='gray')
```

```
change_spectrum(spectrum)
spectrum.scale(500)
spectrum.plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 7.2: Применение функции change_spectrum к сигналу

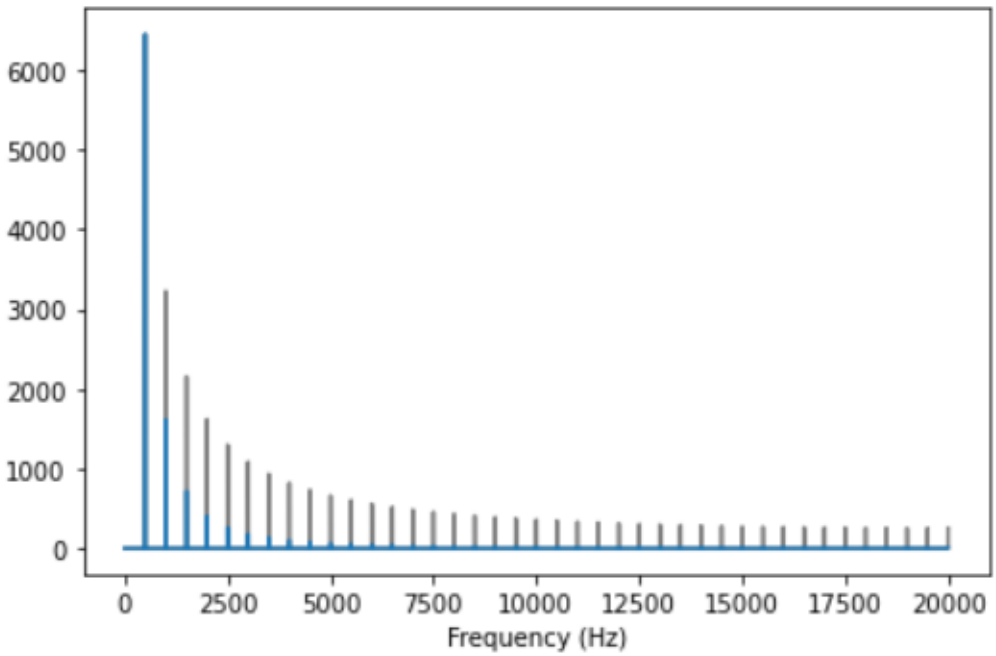

Рис. 7.2: Спектр полученного сигнала

Полученный сигнал похож на синусоиду (Рис.7.3).

```
wave = spectrum.make_wave()
wave.segment(duration=0.01).plot()
decorate(xlabel='Time (s)')
```

Листинг 7.3: Получение сигнала

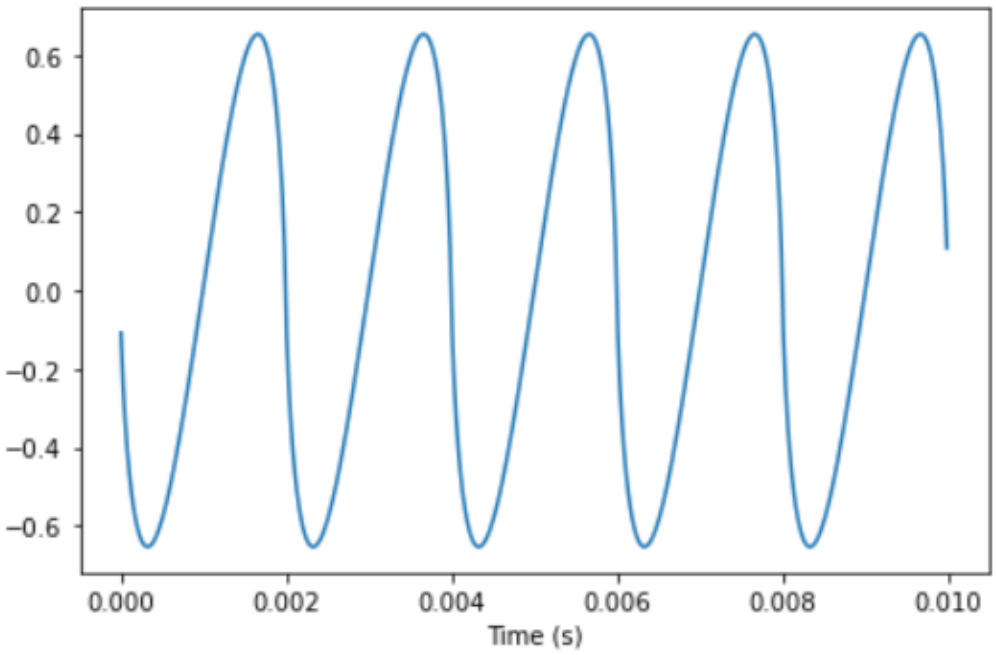

Рис. 7.3: Полученный сигнал

Глава 8

Выводы

В результате выполнения данный работы мы познакомились некоторыми видами сигналов: треугольным, пилообразным и прямоугольным. Мы получили навыки работы с ними. Также мы изучили биение — эффект, приводящий к наложению различных непрерывных сигналов при их дискретизации.

Литература

[1] Фильтрация измерительных сигналов / В.С.Гутников.– СПб.:Изд-во Энергоатомиздат, 1990.-192 с.

[2] Цифровая обработка сигналов / А.С.Сергиенко.– СПб.:Питер, 2003.-603 с.