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

Отчёт по лабораторной работе №9
Дисциплина: Телекоммуникационные технологии
Тема: Дифференцирование и интегрирование

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

Санкт-Петербург
2021

Оглавление

1 Упражнение 9.1 **4**

2 Упражнение 9.2 **9**

3 Упражнение 9.3 **12**

4 Упражнение 9.4 **16**

5 Упражнение 9.5 **18**

6 Выводы **21**

Список иллюстраций

1.1 Сигнал Facebook 4
1.2 Спектр Facebook 5
1.3 Выходной сигнал 5
1.4 Спектр выходного сигнала 6
1.5 Отношение входных и выходных данных 6
1.6 Фильтры нарастающей суммы и интегрирования 7
1.7 Сравнение отношения и фильтра 7
1.8 Сравнение суммирования и фильтрации 8

2.1 Треугольный сигнал 9
2.2 Результат diff 10
2.3 Результат differentiate 10
2.4 Спектр в сигнал 11

3.1 Прямоугольный сигнал 12
3.2 Результат cumsum 13
3.3 Результат integrate 13
3.4 Спектр в сигнал 14
3.5 Сравнение функций 15

4.1 Результирующий сигнал 16
4.2 Спектр результирующего сигнала 17

5.1 Кубический сигнал 18
5.2 Вторая разность кубического сигнала 19
5.3 Вторая производная кубического сигнала 19
5.4 Сравнение фильтров 20

Листинги

1.1 Создание сигнала Facebook 4
1.2 Построение спектра Facebook 5
2.1 Создание треугольного сигнала 9
2.2 Использование diff 9
2.3 Использование differentiate 10
2.4 Преобразование спектра в сигнал 11
3.1 Создание прямоугольного сигнала 12
3.2 Использование cumsum 12
3.3 Использование integrate 13
3.4 Преобразование спектра в сигнал 14
3.5 Сравнение cumsum и integrate 14
4.1 Создание и работа с пилообразным сигналом 16
4.2 Создание спектра результирующего сигнала 16
5.1 Создание кубического сигнала 18
5.2 Вычисление второй разности 18
5.3 Вычисление второй производной 19

Глава 1

Упражнение 9.1

В начале мы запустим примеры из `chap09.ipynb`.

В пособии сказано, что некоторые примеры не работают с апериодическими сигналами. Заменим периодический пилообразный сигнал на непериодические данные Facebook и посмотрим, что случится.

Сначала создадим сигнал (Рис.1.1).

```
import pandas as pd
from thinkdsp import Wave

df = pd.read_csv('FB_2.csv', header=0, parse_dates=[0])
ys = df['Close']
in_wave = Wave(ys, framerate=1)
in_wave.plot()
decorate(xlabel='Time (days)', ylabel='Price ($)')
```

Листинг 1.1: Создание сигнала Facebook

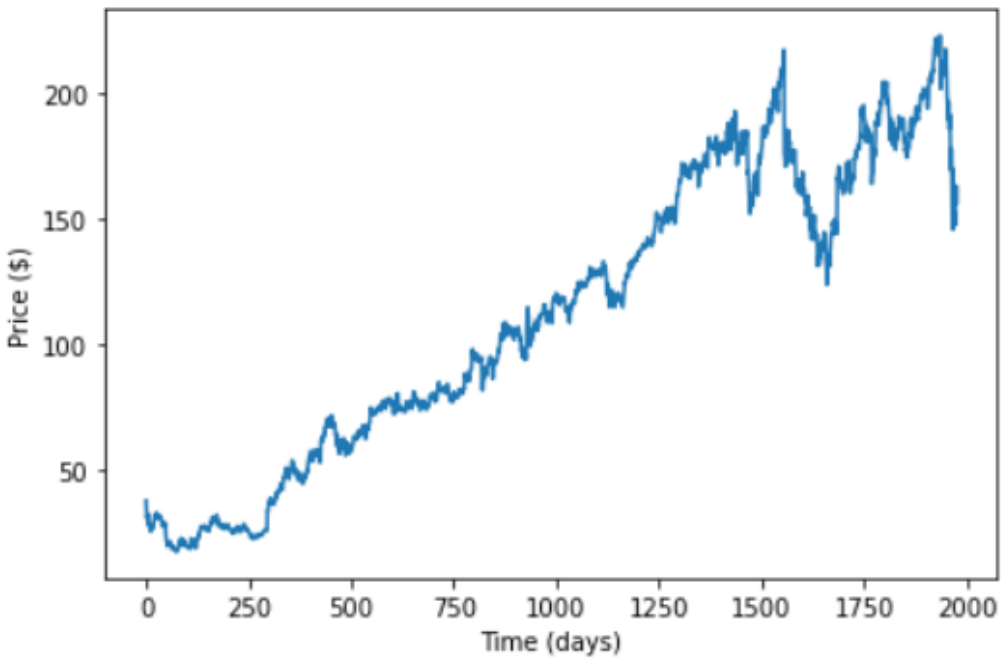

Рис. 1.1: Сигнал Facebook

Построим его спектр (Рис.1.2).

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

Листинг 1.2: Построение спектра Facebook

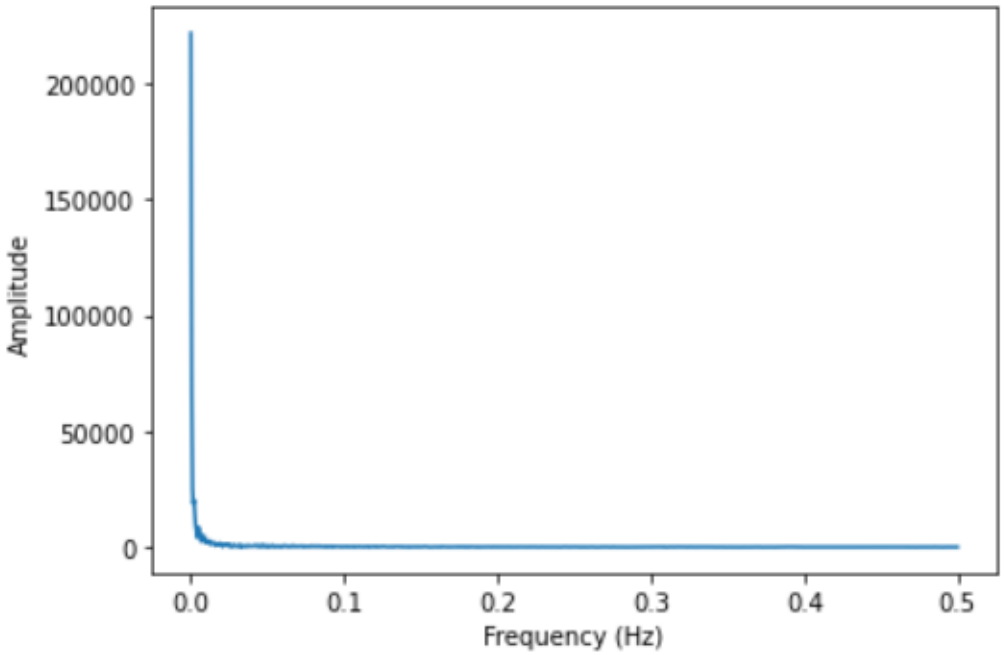

Рис. 1.2: Спектр Facebook

Теперь получим выходной сигнал, который является совокупной суммой входных сигналов (Рис.1.3), и его спектр (Рис.1.4).

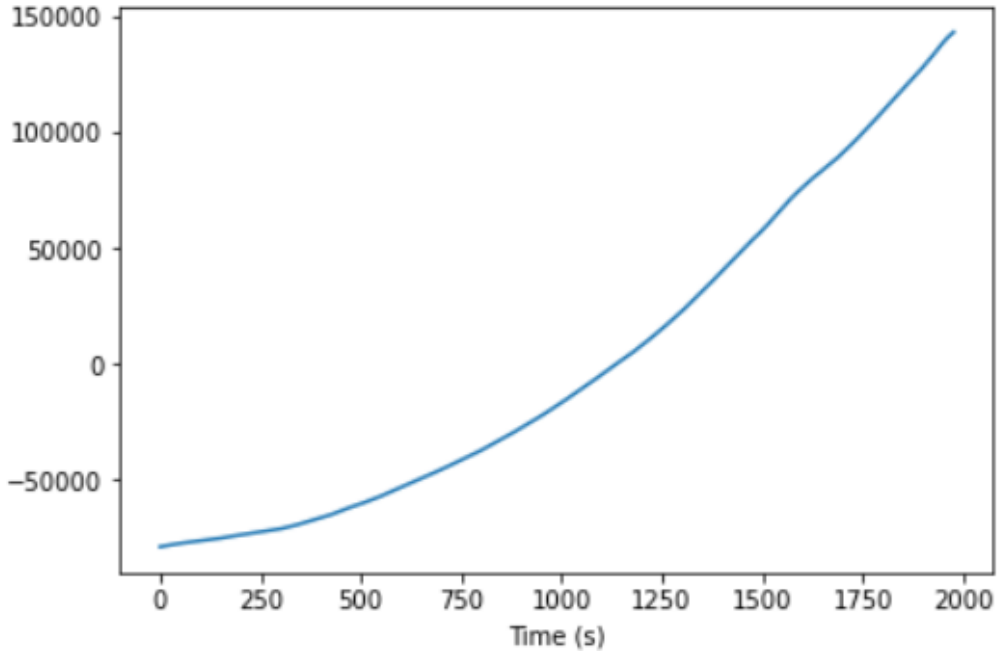

Рис. 1.3: Выходной сигнал

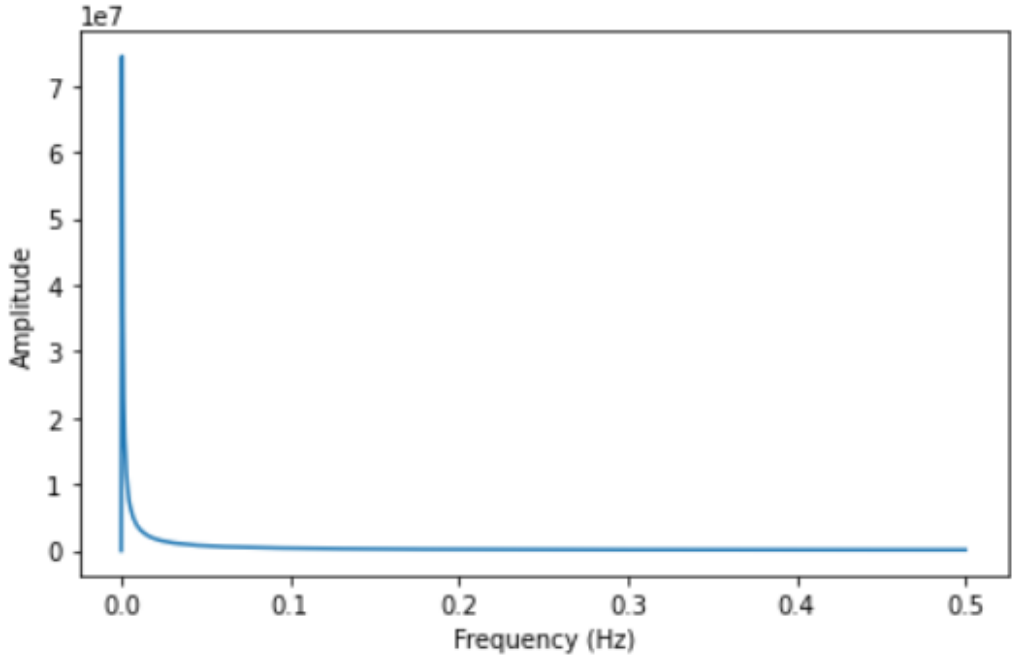

Рис. 1.4: Спектр выходного сигнала

Теперь посмотрим на отношение между входными и выходными данными (Рис.1.5).

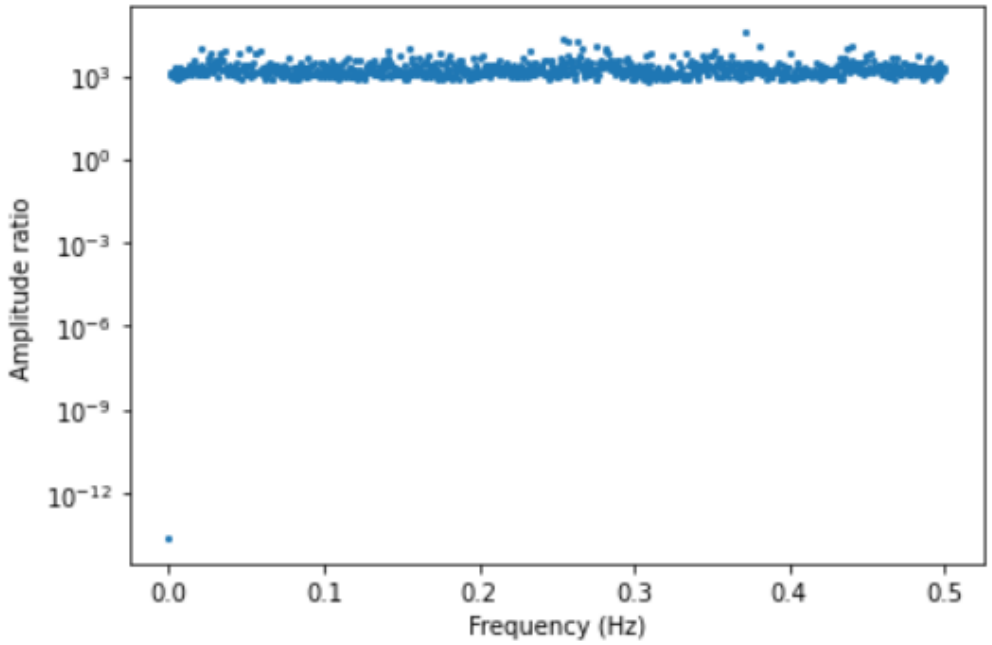

Рис. 1.5: Отношение входных и выходных данных

Построим фильтр для нарастающей суммы и сравним его с фильром интегрирования.

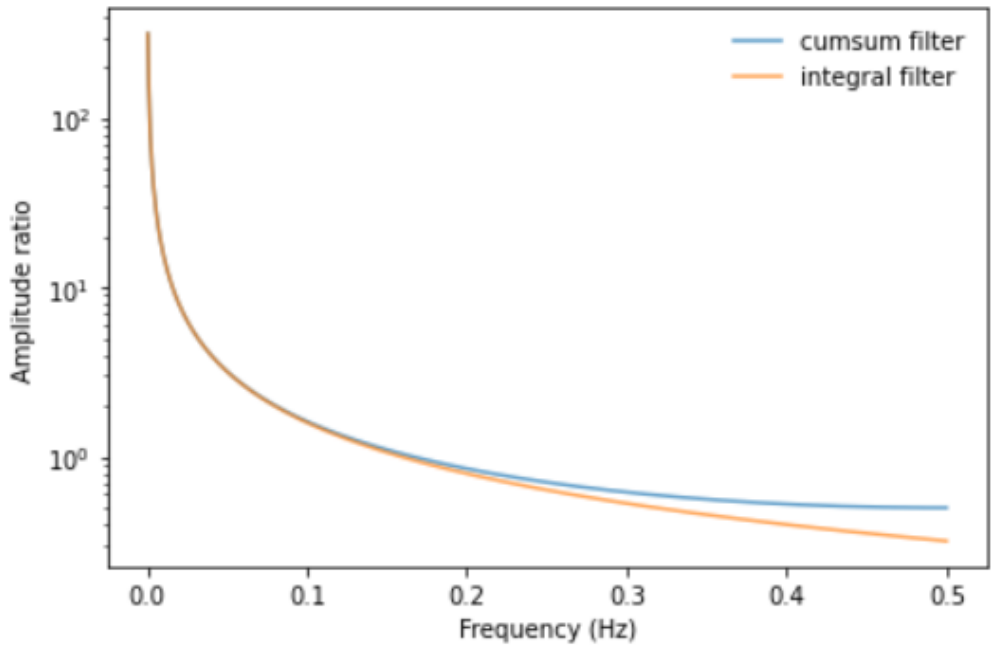

Рис. 1.6: Фильтры нарастающей суммы и интегрирования

Как мы видим на рис.1.6, графики сначала полностью совпадают, а под конец немного расходятся.

Затем мы можем сравнить вычисленное отношение с фильтром.

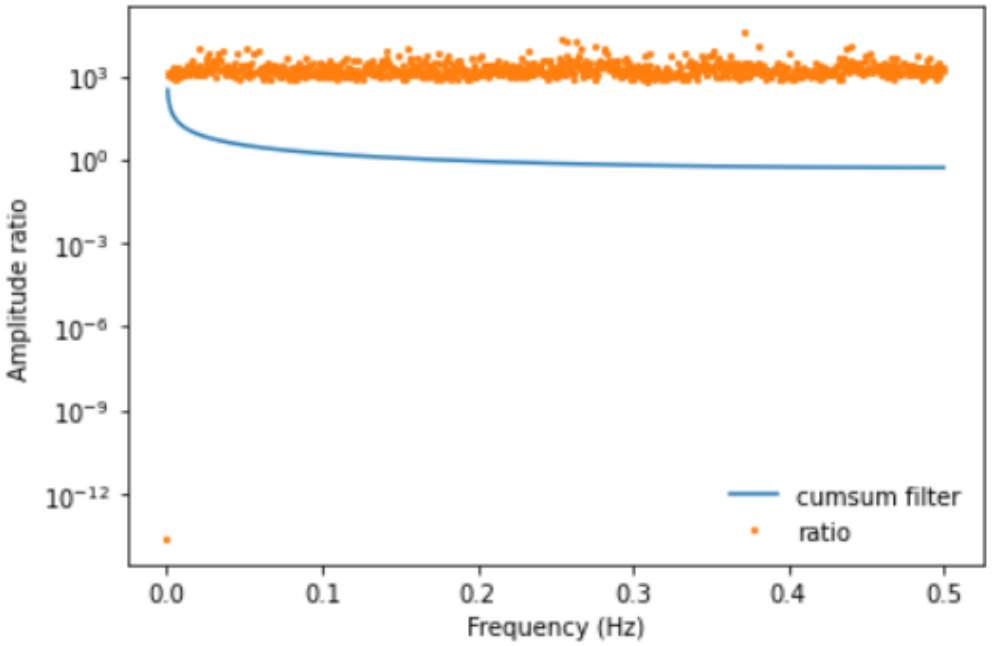

Рис. 1.7: Сравнение отношения и фильтра

Здесь случается первое расхождение (Рис.1.7). Данные графики должны совпадать. Это означало бы что фильтр `cumsum` является обратным фильтру `diff`. Но этого не происходит.

Теперь применим фильтр `cumsum` в частотной области.

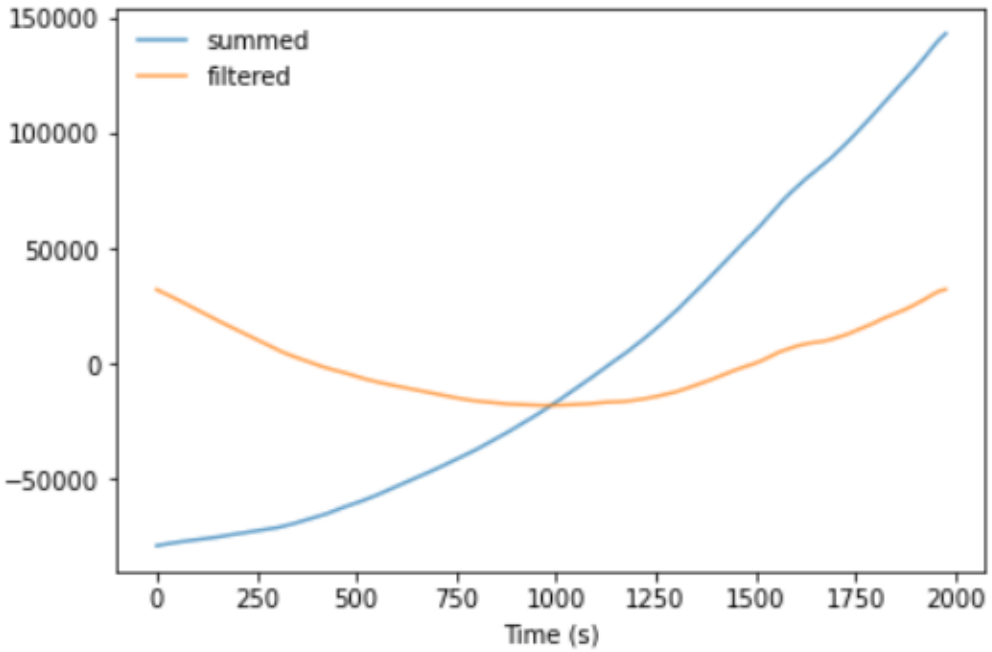

Рис. 1.8: Сравнение суммирования и фильтрации

На рис.1.8 вновь видим пример неправильной работы. Графики не совпадают.

Глава 2

Упражнение 9.2

В этом упражнении изучается влияние `diff` и `differentiate` на сигнал. Создадим треугольный сигнал и напечатаем его (Рис.2.1).

```
from thinkdsp import TriangleSignal

triangle = TriangleSignal(freq=50).make_wave(duration=0.1, framerate=44100)
triangle.plot()
decorate(xlabel='Time (s)')
```

Листинг 2.1: Создание треугольного сигнала

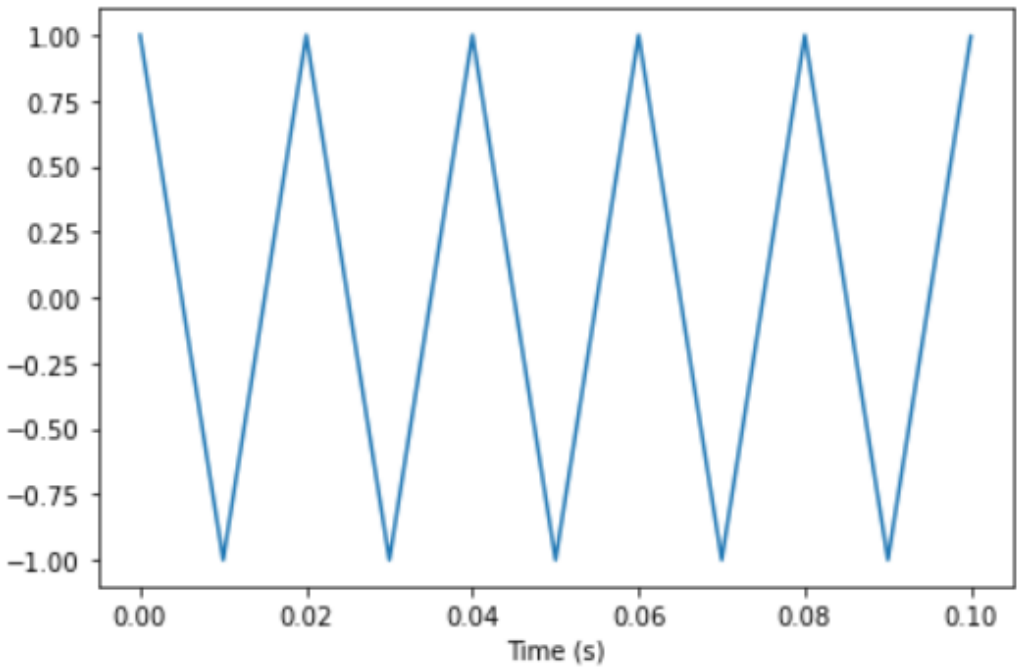

Рис. 2.1: Треугольный сигнал

Применим к нему `diff` и напечатаем результат (Рис.2.2).

```
out_wave = triangle.diff()
out_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 2.2: Использование diff

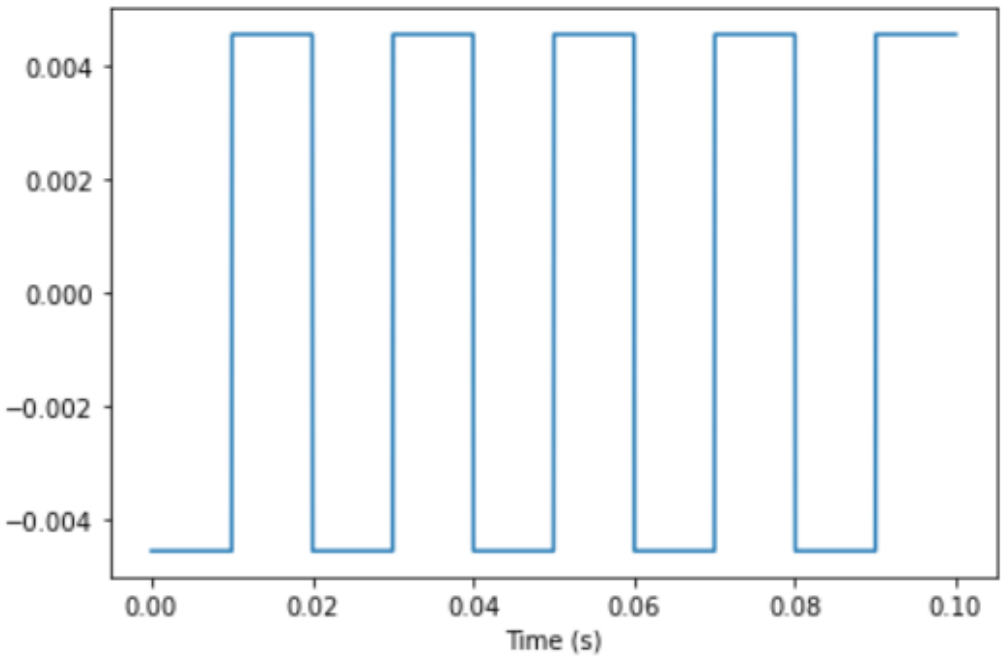

Рис. 2.2: Результат diff

Вычислим спектр треугольного сигнала, применим к нему `differentiate` и напечатаем (Рис.2.3).

```
out_wave2 = triangle.make_spectrum().differentiate().make_wave()
out_wave2.plot()
decorate(xlabel='Frequency (Hz)')
```

Листинг 2.3: Использование differentiate

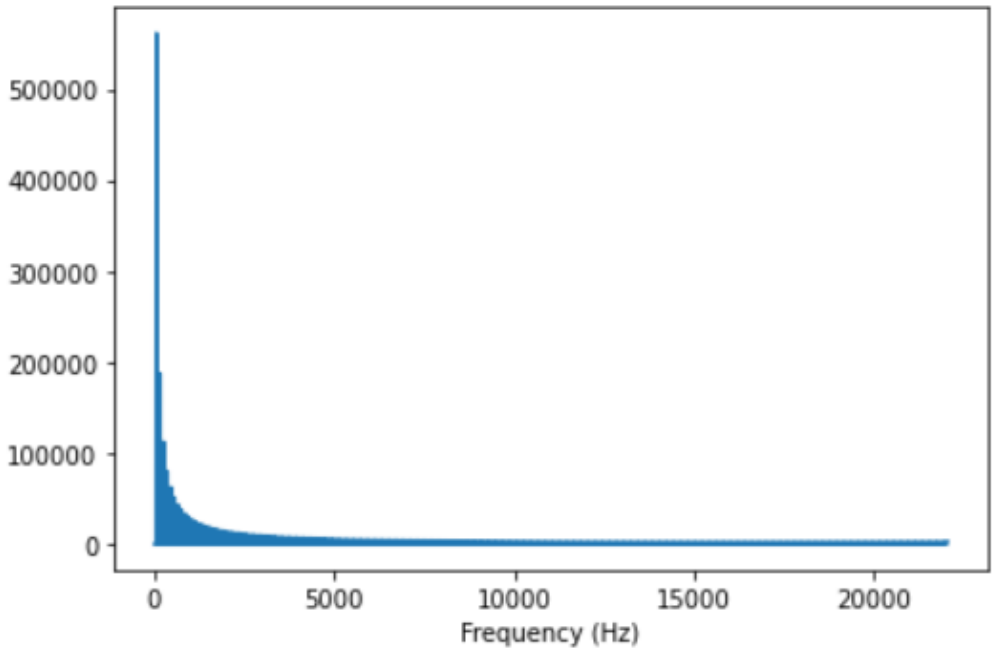

Рис. 2.3: Результат differentiate

Теперь преобразуем спектр обратно в сигнал и напечатем его (Рис.2.4).

```
out_wave2.make_wave().plot()
decorate(xlabel='Time (s)')
```

Листинг 2.4: Преобразование спектра в сигнал

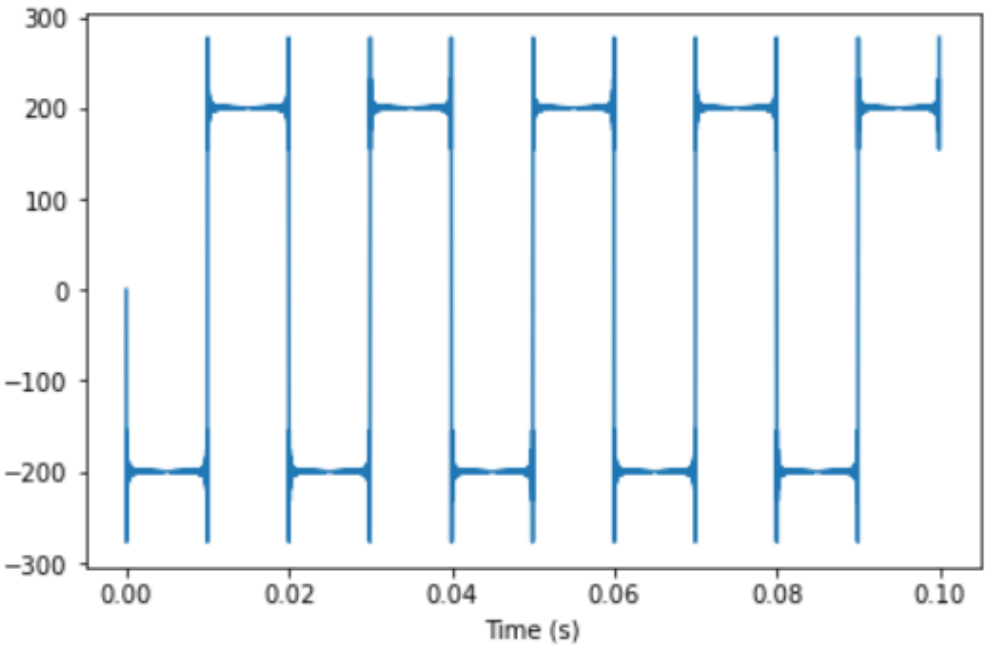

Рис. 2.4: Спектр в сигнал

Когда мы берём спектральную производную, мы получаем "звон"вокруг разрывов.

С математической точки зрения это происходит, потому что производная треугольного сигнала не определена в вершинах треугольников.

Глава 3

Упражнение 9.3

В данном упражнении изучается влияние `cumsum` и `integrate` на сигнал. Создадим прямоугольный сигнал и напечатаем его (Рис.3.1).

```
from thinkdsp import SquareSignal

square = SquareSignal(freq=50).make_wave(duration=0.1, framerate=44100)
square.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.1: Создание прямоугольного сигнала

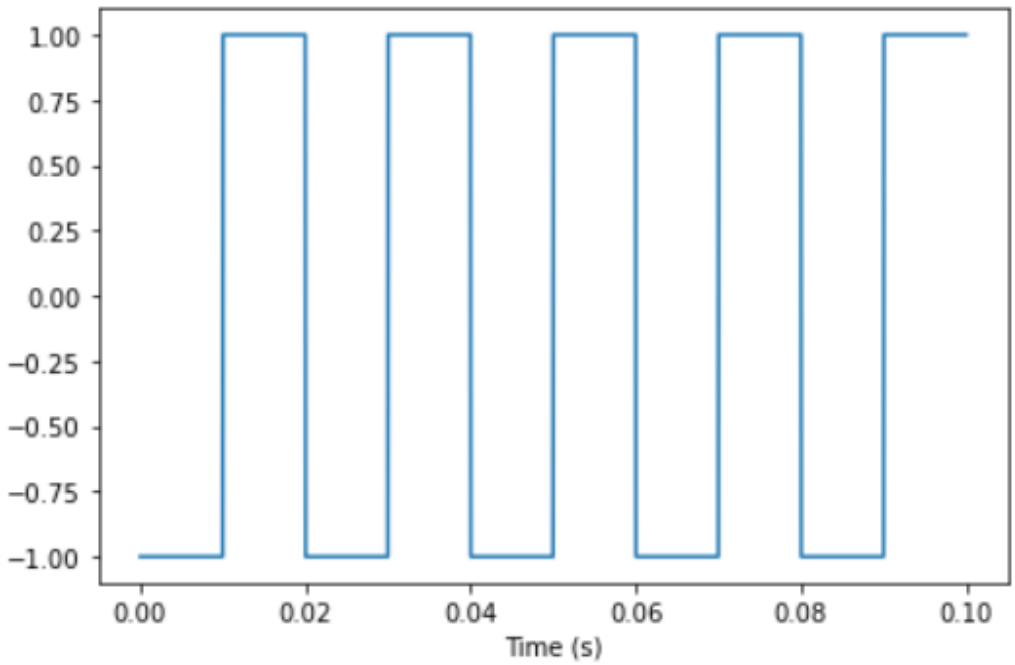

Рис. 3.1: Прямоугольный сигнал

Применим к нему `cumsum` и напечатаем результат (Рис.3.2).

```
out_wave = square.cumsum()
out_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.2: Использование cumsum

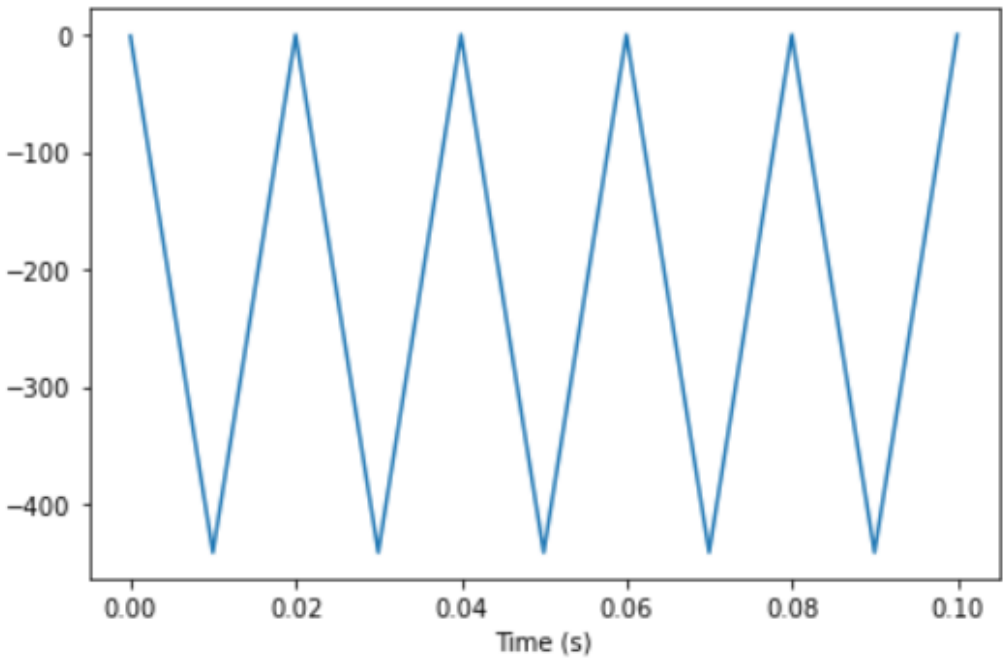

Рис. 3.2: Результат cumsum

Вычислим спектр прямоугольного сигнала, применим к нему `integrate` и напечатаем (Рис.3.3).

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

Листинг 3.3: Использование integrate

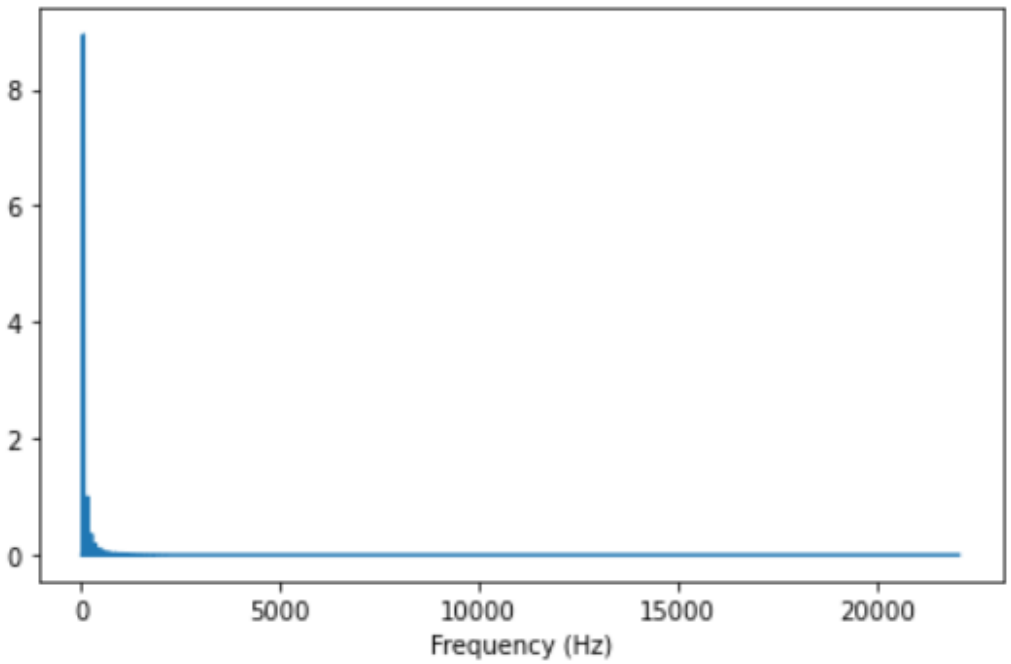

Рис. 3.3: Результат integrate

Теперь преобразуем спектр обратно в сигнал и напечатем его (Рис.3.4).

```
spectrum.hs[0] = 0
out_wave2 = spectrum.make_wave()
out_wave2.plot()
decorate(xlabel='Time (s)')
```

Листинг 3.4: Преобразование спектра в сигнал

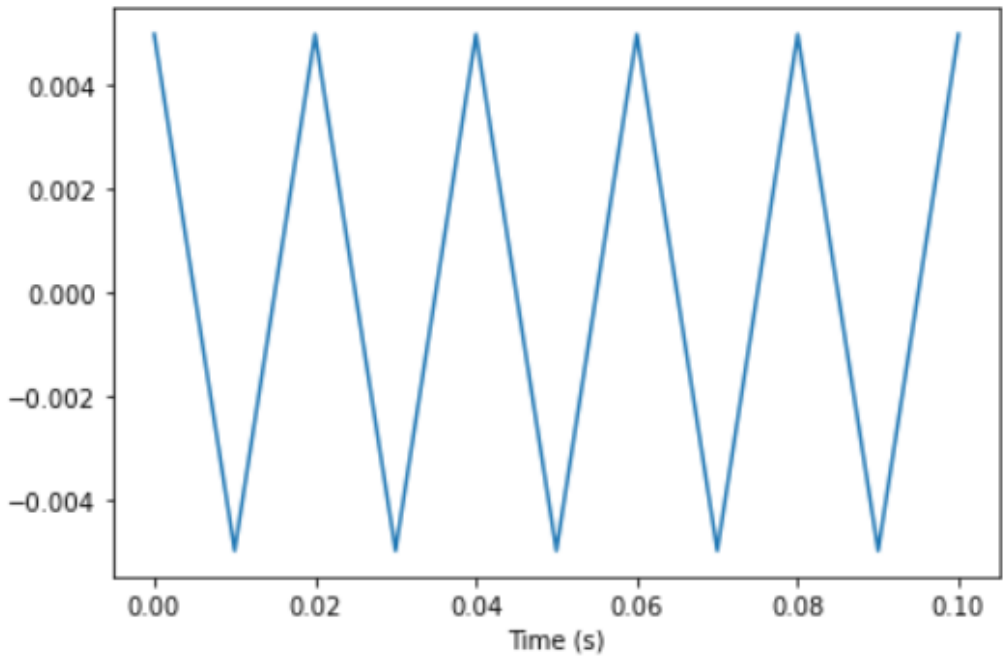

Рис. 3.4: Спектр в сигнал

Результаты `cumsum` и `integrate` с виду получились одинаковыми. Проверим, есть ли какие-то различия между этими функциями (Рис.3.5).

```
out_wave.unbias()
out_wave.normalize()
out_wave2.normalize()
out_wave.plot()
out_wave2.plot()
decorate(xlabel='Time (s)')

out_wave.max_diff(out_wave2)
```

Листинг 3.5: Сравнение cumsum и integrate

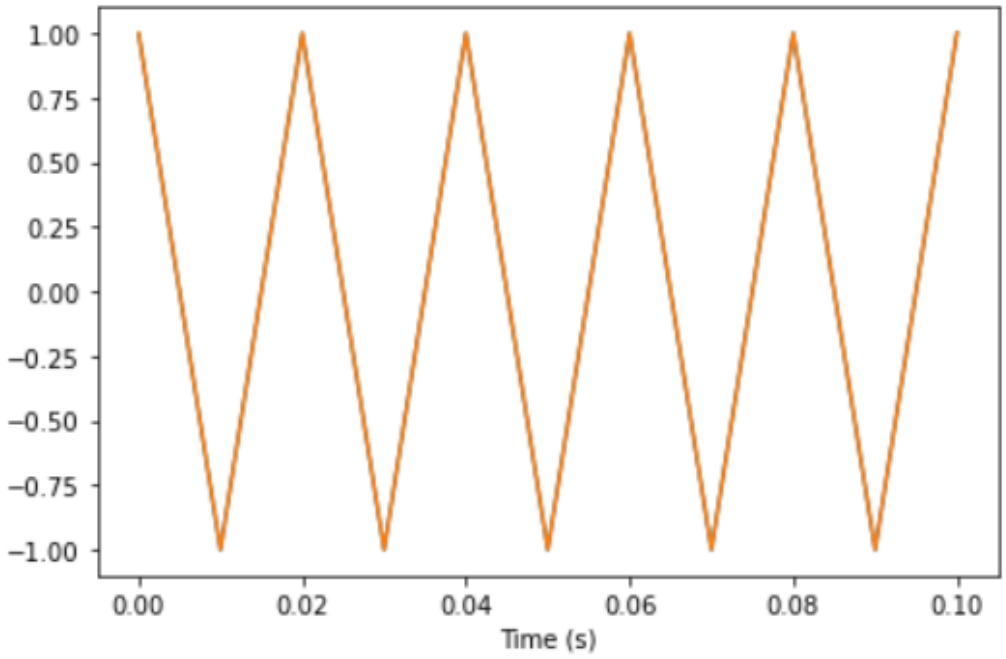

Рис. 3.5: Сравнение функций

Численно они тоже схожи, но с точностью около 3 цифр.

Глава 4

Упражнение 9.4

В этом упражнении изучается влияние двойного интегрирования.

Создадим пилообразный сигнал, вычислим его спектр и дважды применим `integrate`. Затем напечатаем результирующий сигнал и его спектр.

```
from thinkdsp import SawtoothSignal

sawtooth = SawtoothSignal(freq=50).make_wave(duration=0.1, framerate=44100)
spectrum = sawtooth.make_spectrum().integrate().integrate()
spectrum.hs[0] = 0
out_wave = spectrum.make_wave()
out_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 4.1: Создание и работа с пилообразным сигналом

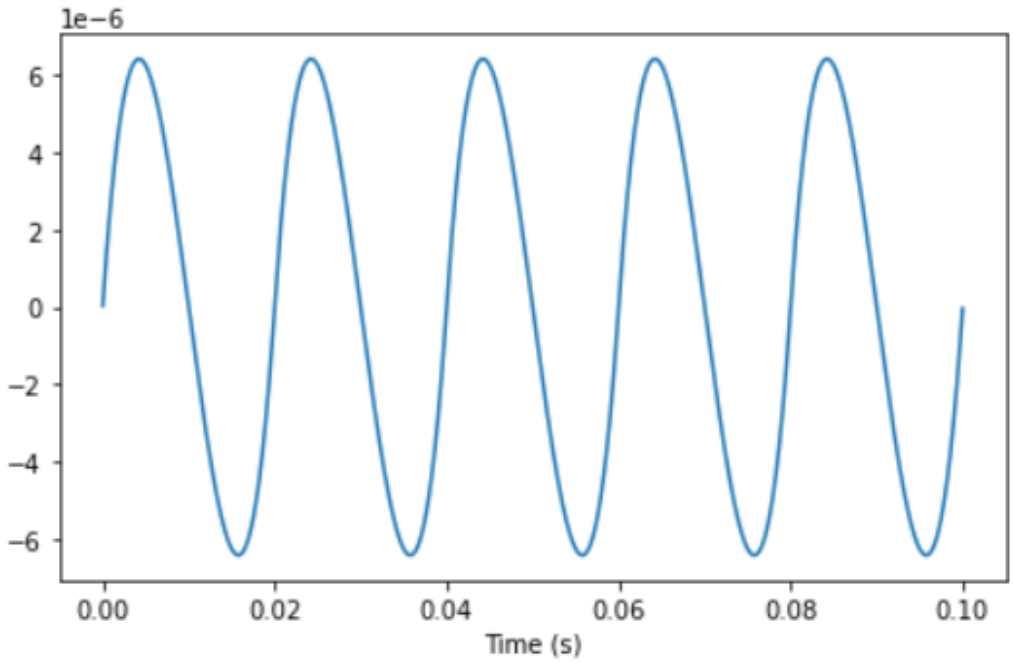

Рис. 4.1: Результирующий сигнал

```
out_wave.make_spectrum().plot(high=500)
```

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

Листинг 4.2: Создание спектра результирующего сигнала

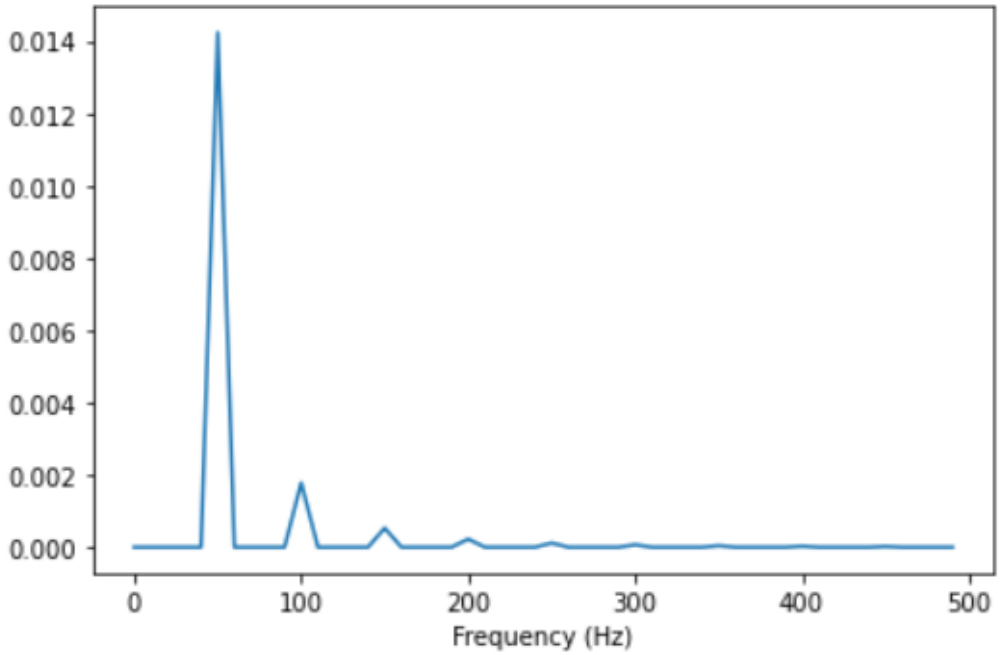

Рис. 4.2: Спектр результирующего сигнала

Из рис.4.1 мы можем видеть, что результат напоминает синусоиду. Причина в том, что интегрирование действует как фильтр нижних частот. Из спектра видно, что после двойного применения функции `integrate` мы отфильтровали почти все, кроме основных частот (Рис.4.2).

Глава 5

Упражнение 9.5

В данном упражнении изучается влияние второй разности и второй производной. Создадим `CubicSignal`, определённый в `thinkdsp`.

```
from thinkdsp import CubicSignal

cubic = CubicSignal(freq=0.0005).make_wave(duration=10000, framerate=1)
cubic.plot()
```

Листинг 5.1: Создание кубического сигнала

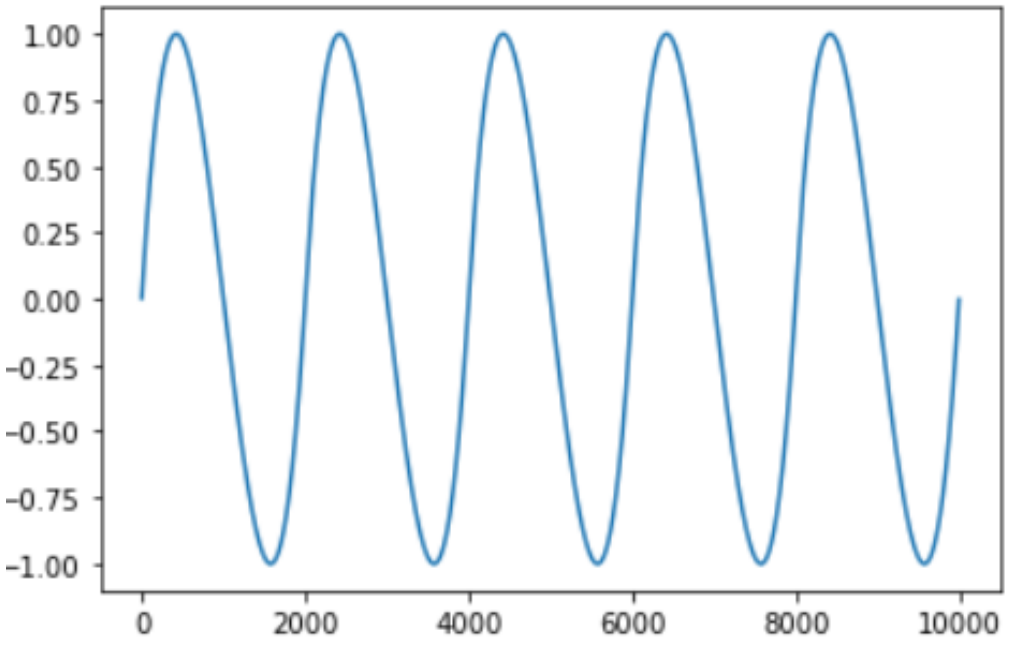

Рис. 5.1: Кубический сигнал

Вычислим его вторую разность, дважды применив `diff`. Напечатаем получившийся результат (Рис.5.2).

```
out_wave = cubic.diff()
out_wave = out_wave.diff()
out_wave.plot()
decorate(xlabel='Time (s)')
```

Листинг 5.2: Вычисление второй разности

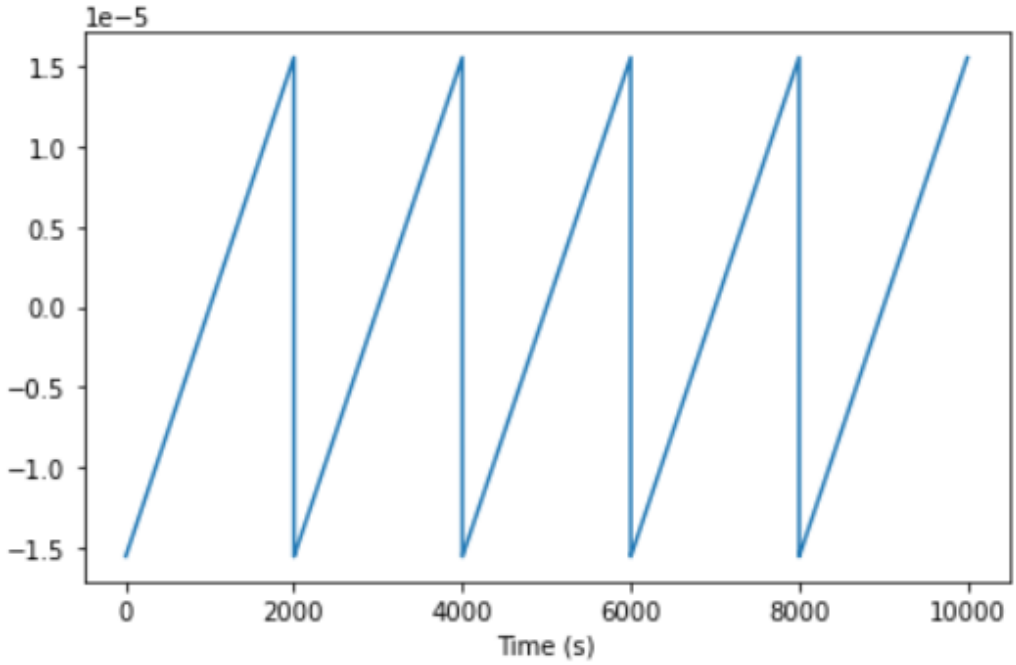

Рис. 5.2: Вторая разность кубического сигнала

Получившийся результат похож на пилообразный сигнал.

Вычислим вторую производную, дважды применив `differentiate` к спектру. Напечатаем получившийся результат (Рис.5.3).

```
spectrum = cubic.make_spectrum().differentiate().differentiate()
out_wave2 = spectrum.make_wave()
out_wave2.plot()
decorate(xlabel='Time (s)')
```

Листинг 5.3: Вычисление второй производной

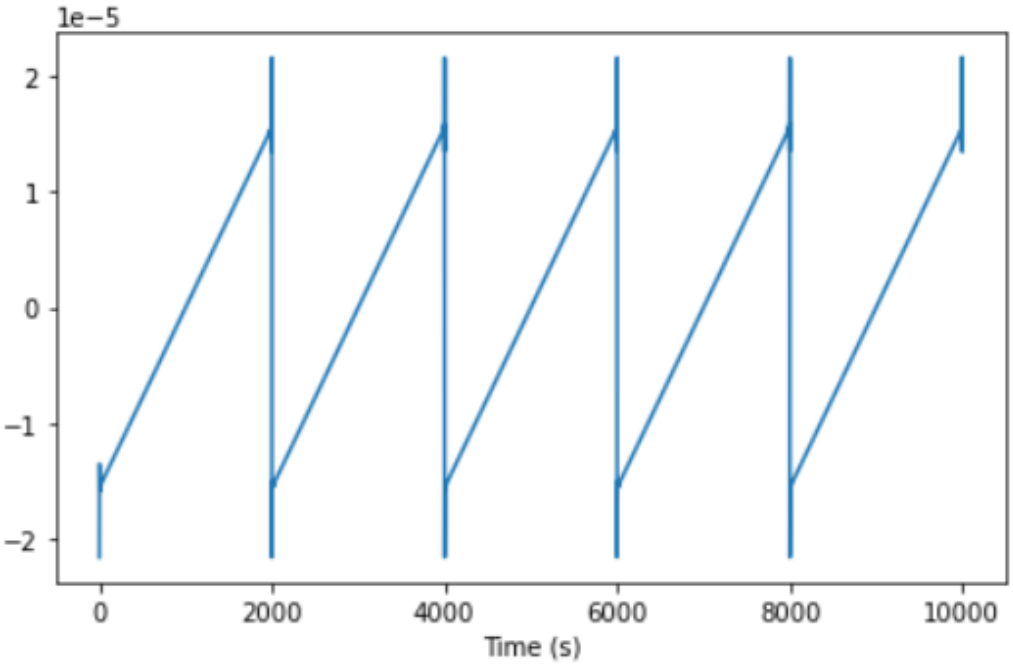

Рис. 5.3: Вторая производная кубического сигнала

Как видим, мы получили пилообразную форму с некоторым звоном.

Распечатаем фильтры, соответствующие второй разности и второй производной, и сравним их (Рис.5.4).

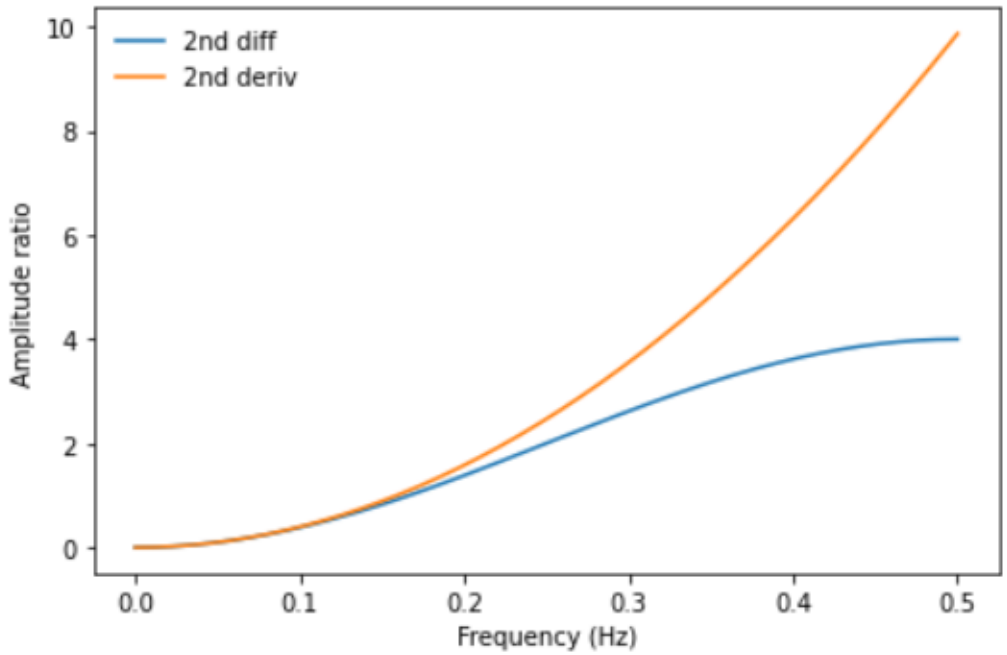

Рис. 5.4: Сравнение фильтров

Оба фильтра являются фильтрами высоких частот, которые усиливают компоненты самых высоких частот. Вторая производная параболическая, поэтому она сильнее всего усиливает самые высокие частоты. Вторая разность - хорошее приближение второй производной только на самых низких частотах, затем она существенно отклоняется.

Глава 6

Выводы

В результате выполнения данной работы мы изучили функции дифференцирования и интегрирования, а также научились применять их на различных сигналах.