Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа №9

Дифференцирование и интегрирование

Выполнил студент 3-го курса
группа 3530901/80201
Матвеец Андрей Вадимович

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург

2021

Содержание

Список иллюстраций

Листинги

1 Часть №1: chap09.ipynb

В первой части лабораторной работы нам необходимо запустить все примеры из chap09.ipynb, прочитать все описания, после чего заменить пилообразный сигнал на входе на непериодические данные Facebook, после чего повторно запустить все примеры.

Начнем с загрузки непериодических данных Facebook и представления их в виде графика:

```
[173] df = pd.read_csv('FB_2.csv', header=0, parse_dates=[0])
      ys = df['Close']
```

```
[174] wave = Wave(ys, framerate=1)
      wave.plot()
      decorate(xlabel='Time (d)')
```

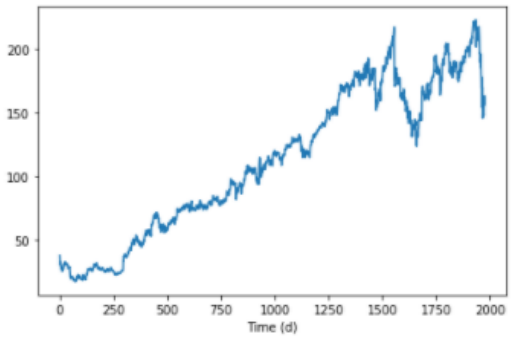

Рис. 1: Непериодические данные Facebook

После этого посмотрим на спектр полученных данных:

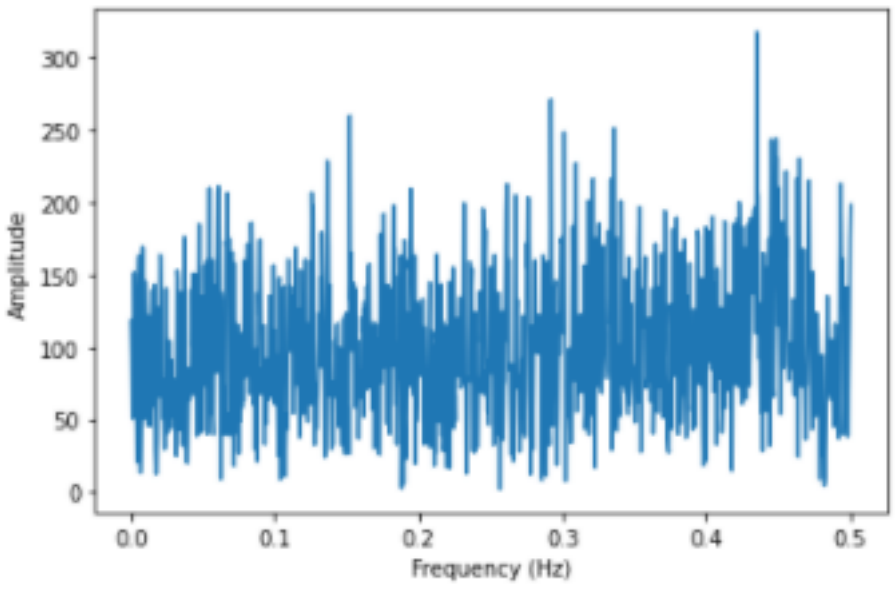

Рис. 2: Спектр данных Facebook

Теперь, из-за того, что между гармониками входные компоненты маленькие, установим эти отношения на NaN и посмотрим на результат:

```
ratio_spectrum = out_spectrum.ratio(in_spectrum, thresh=1)
ratio_spectrum.plot(marker='.', ms=4, ls='')

decorate(xlabel='Frequency (Hz)',
         ylabel='Amplitude ratio',
         yscale='log')
```

Листинг 1: `Ratio spectrum`

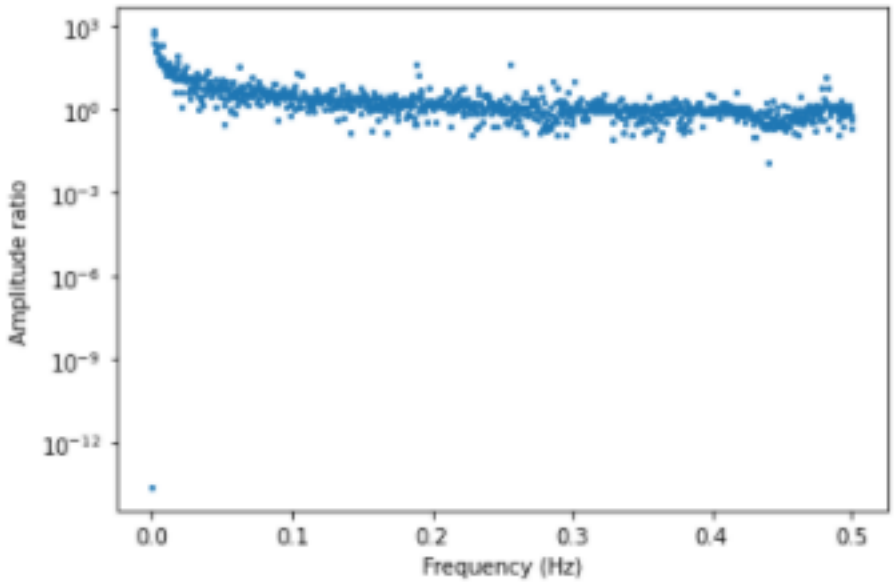

Рис. 3: Полученные результаты

После этого сравним вычисленные отношения с фильтром. Сразу можно заметить, что данные не совпадают, идет большой разброс:

```
cumsum_filter.plot(label='cumsum filter')
ratio_spectrum.plot(label='ratio', marker='.', ms=4, ls='')
decorate(xlabel='Frequency (Hz)',
         ylabel='Amplitude ratio',
         yscale='log')
```

Листинг 2: Сравнение вычисленного отношения с фильтром

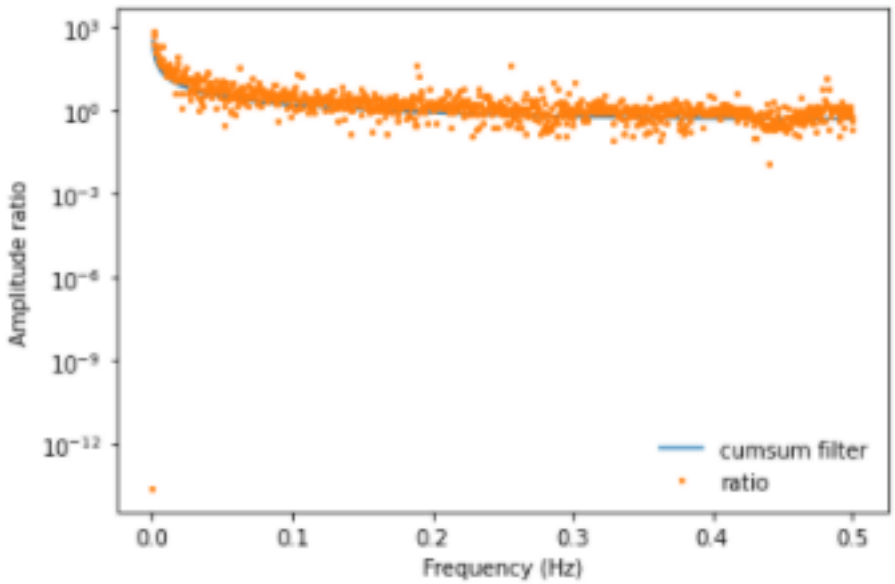

Рис. 4: Результаты сравнения вычисленного отношения с фильтром

Наконец, сравним полученные результаты с входными:

```
out_wave.plot(label='summed', alpha=0.7)

cumsum_filter.hs[0] = 0
out_wave2 = (in_spectrum * cumsum_filter).make_wave()
out_wave2.plot(label='filtered', alpha=0.7)

decorate(xlabel='Time (s)')
```

Листинг 3: Сравнение вычисленного отношения с фильтром

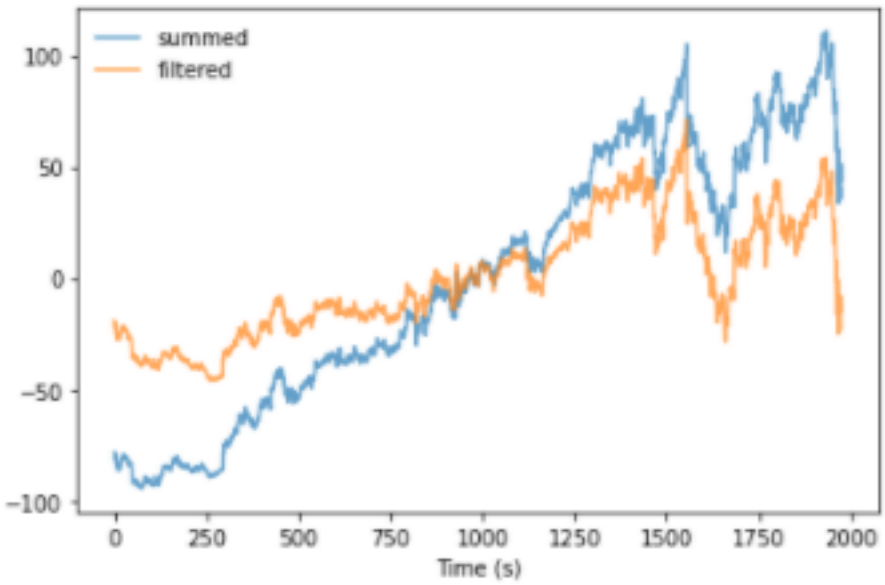

Рис. 5: Итоговое сравнение

Как видно в итоговом сравнении, полученные графики не равны.

2 Часть №2: `diff` и `differentiate`

Во втором пункте лабораторной работы нам необходимо изучить влияние `diff` и `differentiate` на сигнал. Для этого необходимо создать треугольный сигнал, распечатать его и применить к нему `diff`. Затем вычислить спектр треугольного сигнала, применить `differentiate` и потом напечатать результат.

Начнем с создания треугольного сигнала и сразу же выведем его на экран:

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

Листинг 4: Создание и вывод треугольного сигнала

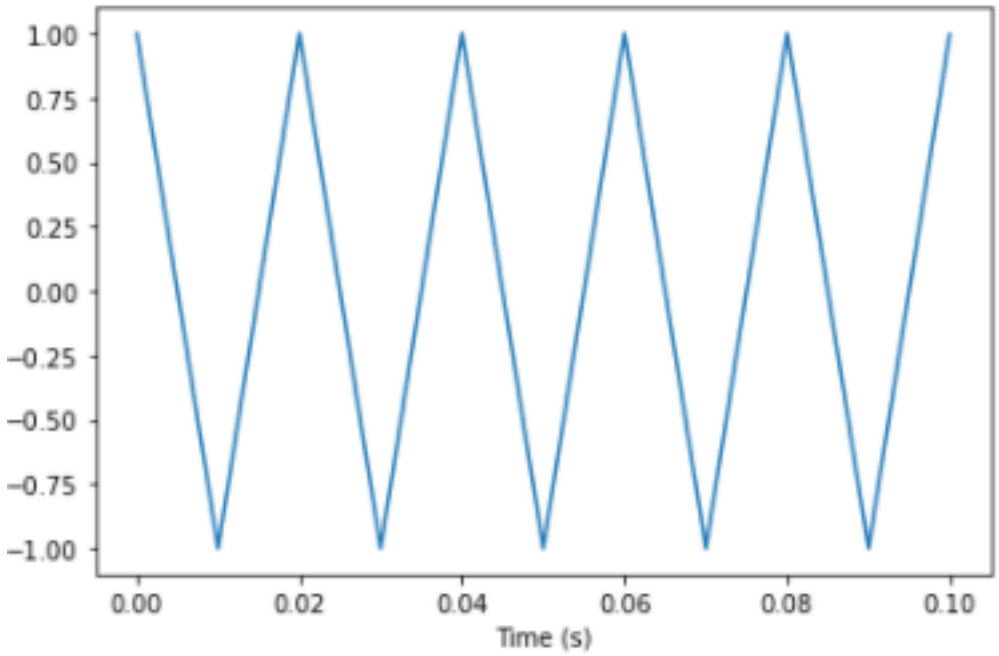

Рис. 6: Треугольный сигнал

Теперь применим `diff` и выведем полученный график:

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

Листинг 5: Применение `diff`

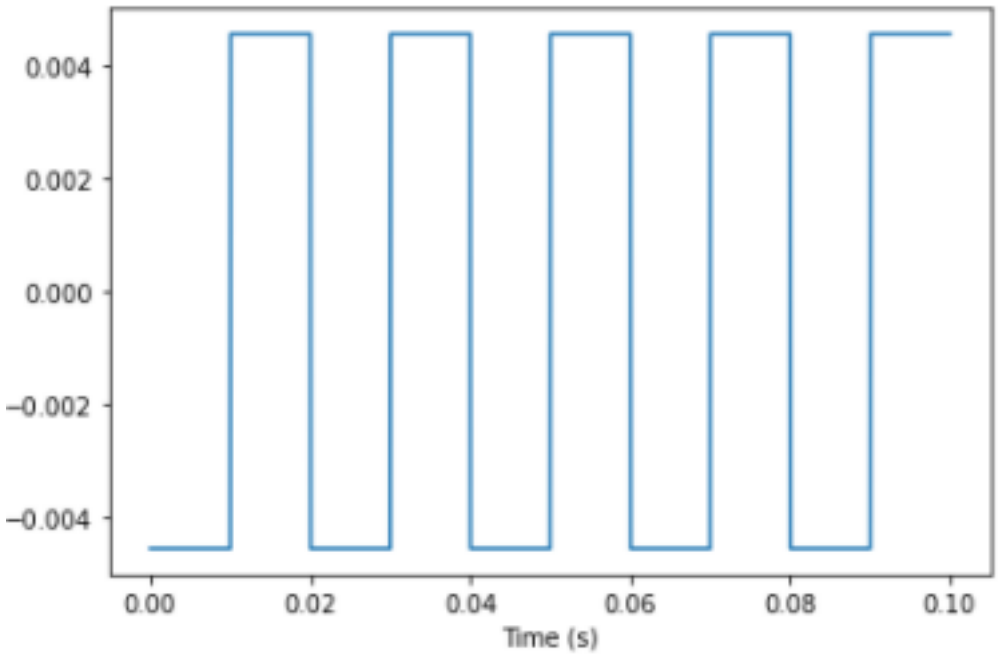

Рис. 7: Применение `diff`

Как видно - `diff` треугольной волны - прямоугольная волна.

Теперь вычислим спектр и применим к нему `differentiate`

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

Листинг 6: Вычисление спектра и применение `differentiate`

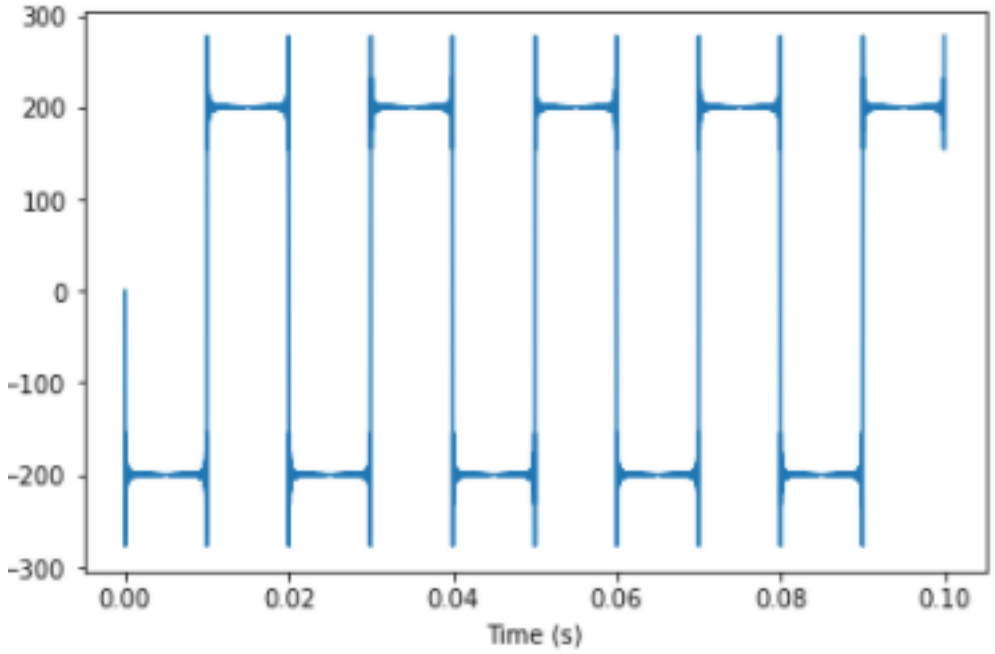

Рис. 8: Применение `differentiate`

На графике виден необычный эффект. Он вызван тем, что производная треугольной волны не определена в точках "треугольника".

3 Часть №3: cumsum и integrate

В третьем пункте лабораторной работы нам необходимо изучить влияние `cumsum` и `integrate` на сигнал. Для этого необходимо создать прямоугольный сигнал, распечатать его и применить к нему `cumsum`. Затем вычислить спектр треугольного сигнала, применить `integrate` и потом напечатать результат.

Начнем с создания прямоугольного сигнала и сразу распечатаем его:

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

Листинг 7: Создание и вывод прямоугольного сигнала

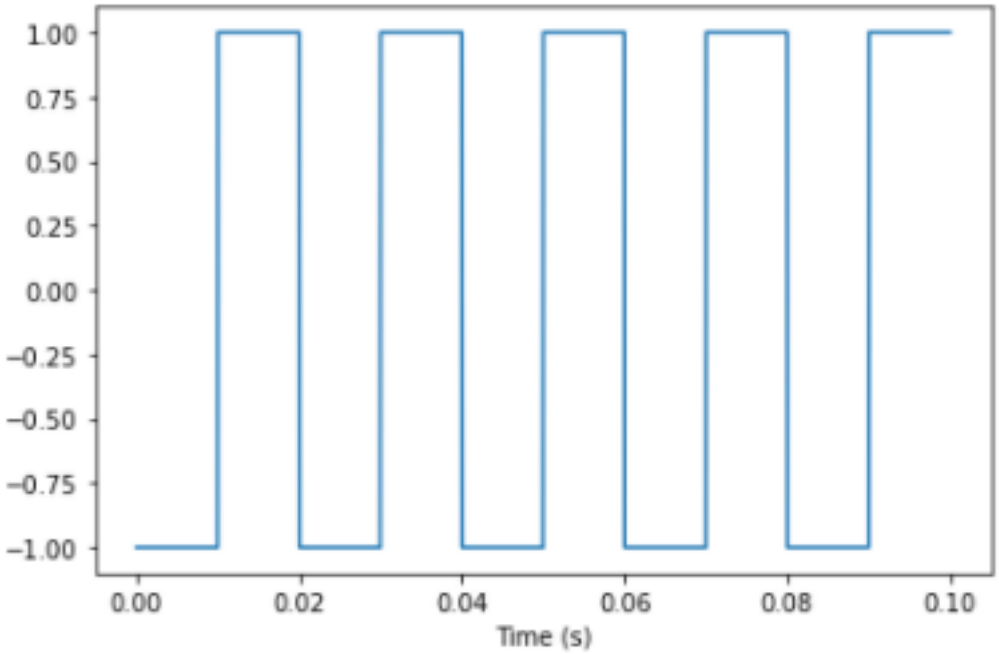

Рис. 9: Прямоугольный сигнал

Теперь применим к полученному сигналу `cumsum` и выведем полученный график:

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

Листинг 8: Применение `cumsum`

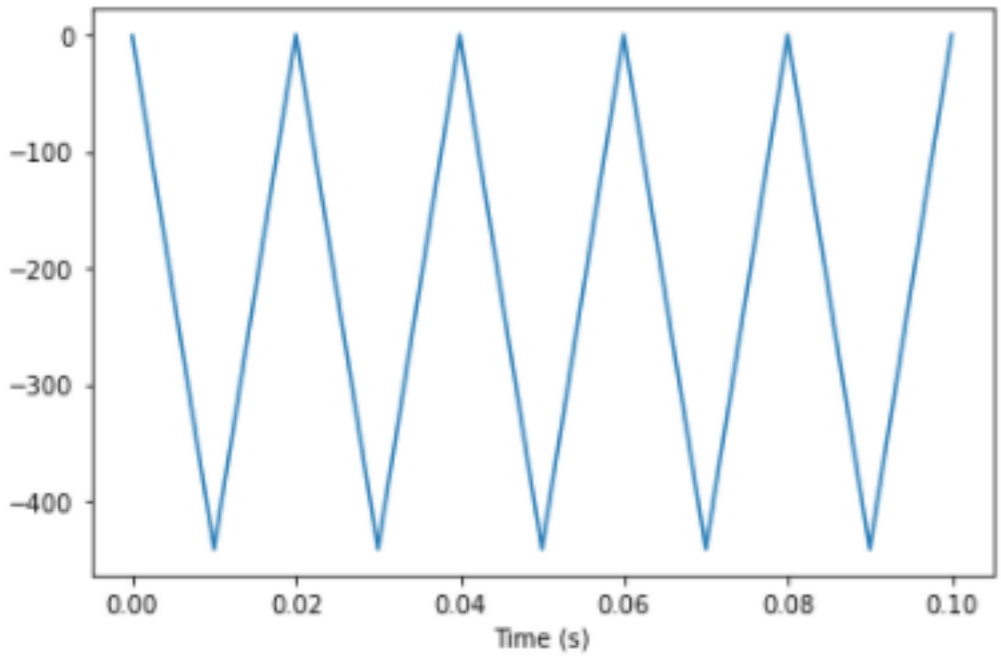

Рис. 10: Применение `cumsum`

Как видно - `cumsum` для прямоугольной волны - треугольная волна.

Кроме того спектральный интеграл так же является треугольной волной:

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

Листинг 9: Применение `integrate`

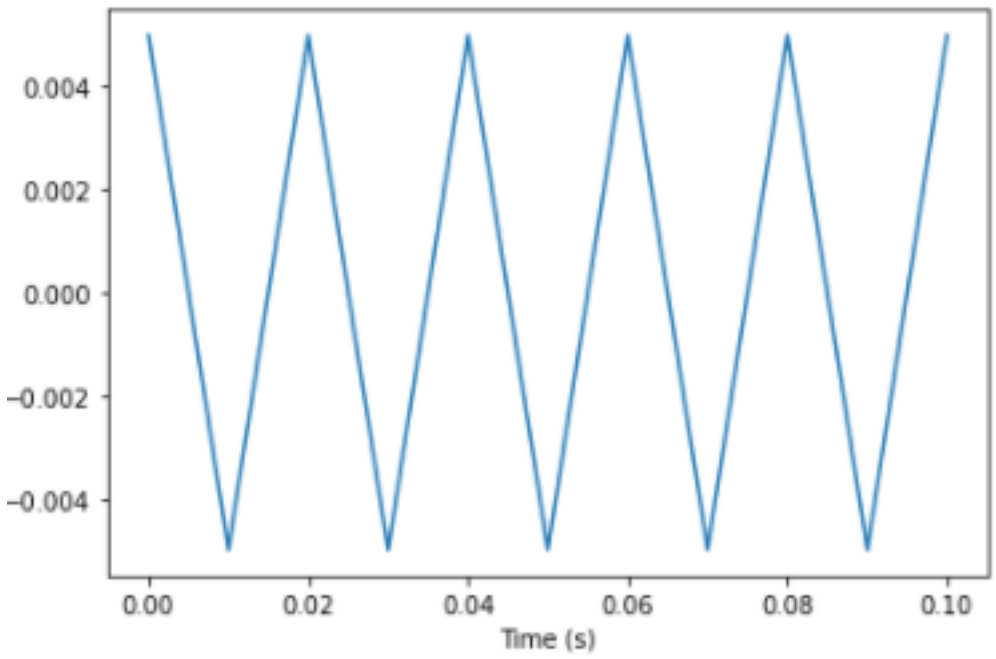

Рис. 11: Применение `integrate`

Доказать схожесть сигналов можно сместив и нормализовав обе волны:

```
out_wave.unbias()
out_wave.normalize()
out_wave2.normalize()
out_wave.plot()
out_wave2.plot()
```

Листинг 10: Смещение и нормализация сигналов

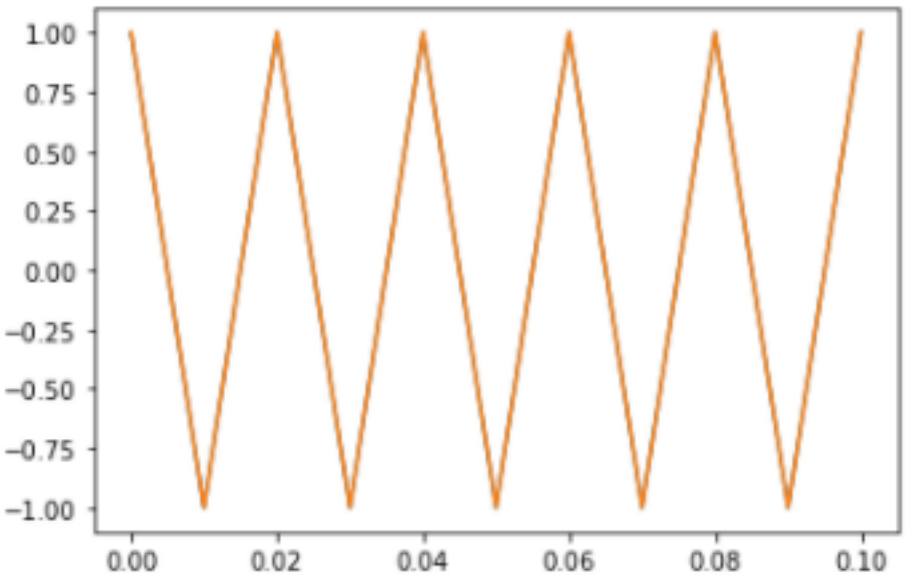

Рис. 12: Результат смещения и нормализации сигналов

Для того, чтобы точно определить насколько полученные графики схожи

напишем следующую строчку:

```
B [30]: out_wave.max_diff(out_wave2)
Out[30]: 0.0045351473922902175
```

Рис. 13: Результат сравнения

По результатам видно, что различия в сигналах почти неразличимы.

4 Часть №4: Двойное интегрирование

В четвертом пункте лабораторной работы нам необходимо изучить влияние двойного интегрирования. Для этого надо создать пилообразный сигнал, вычислить его спектр и дважды применить `integrate`, после чего напечатать результирующий сигнал, и его спектр.

Начнем с создания пилообразного сигнала и сразу распечатаем его:

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

Листинг 11: Создание и вывод пилообразнного сигнала

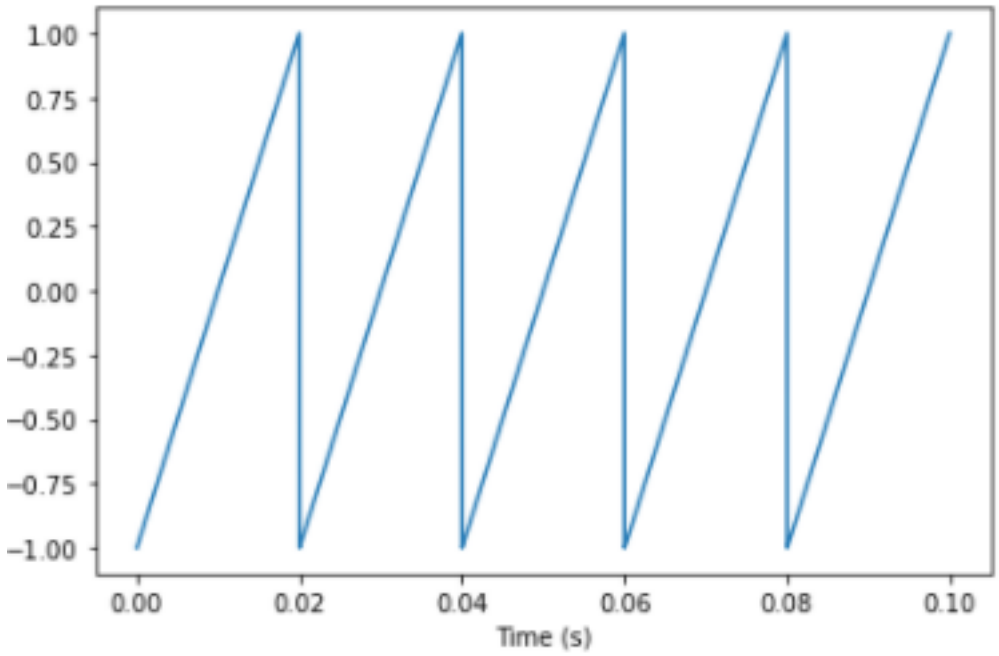

Рис. 14: Пилообразный сигнал

Теперь применим к полученному сигналу `cumsum` и выведем полученный график:

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

Листинг 12: Применение `cumsum`

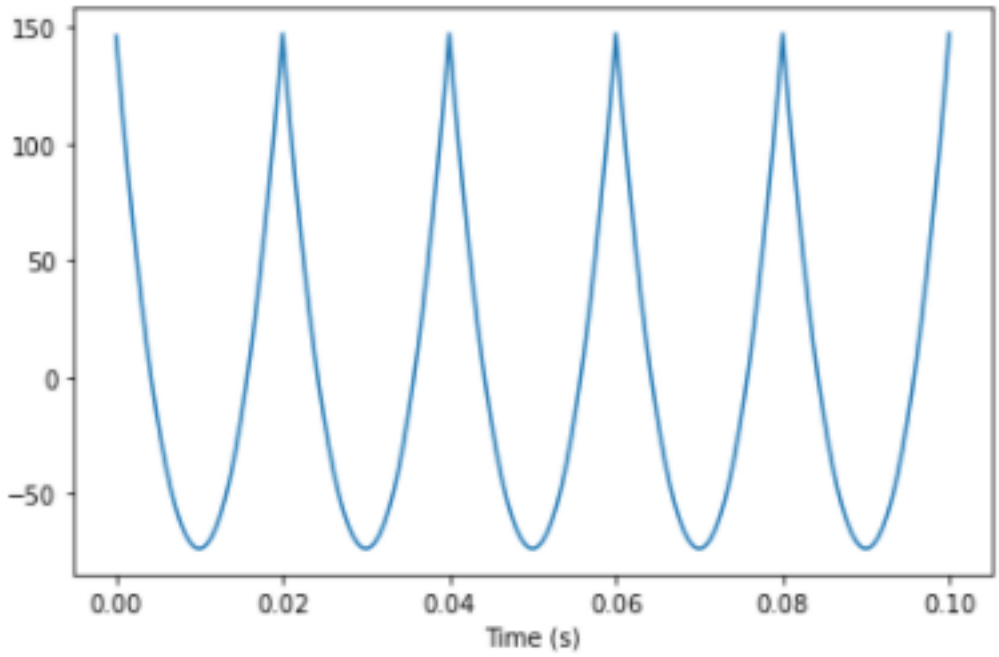

Рис. 15: Применение `cumsum`

Как видно - первая сумма пилообразного сигнала - парабола.

Опять применим функцию `cumsum`:

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

Листинг 13: Второе применение `cumsum`

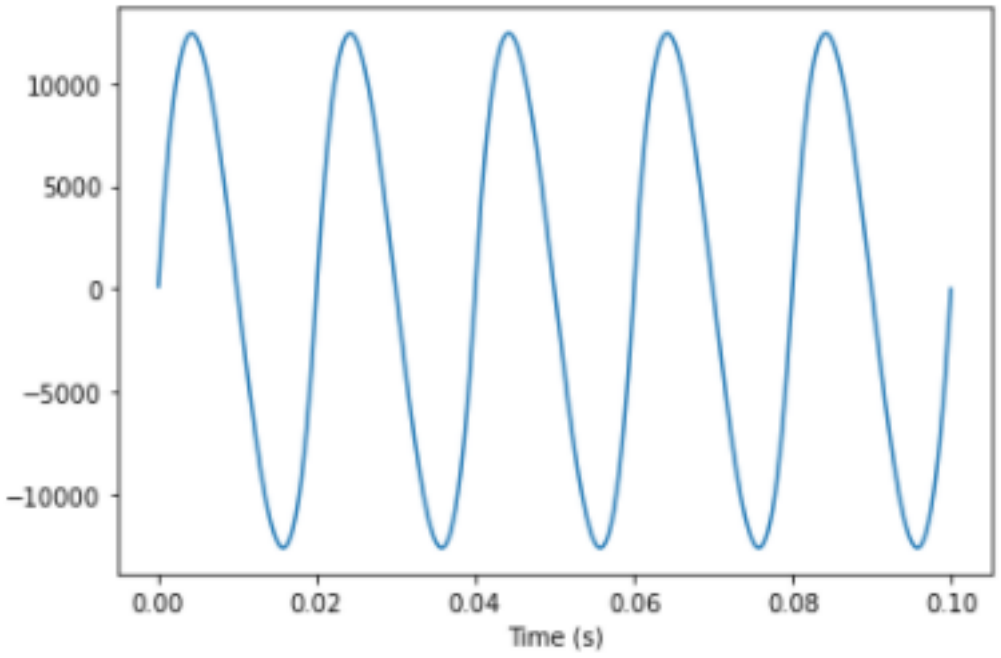

Рис. 16: Второе применение `cumsum`

Как видно - вторая сумма пилообразного сигнала - кубическая кривая.

Наконец, применим к исходному пилообразному сигналу два раза `integrate` и посмотрим на результат:

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

Листинг 14: Применение `integrate`

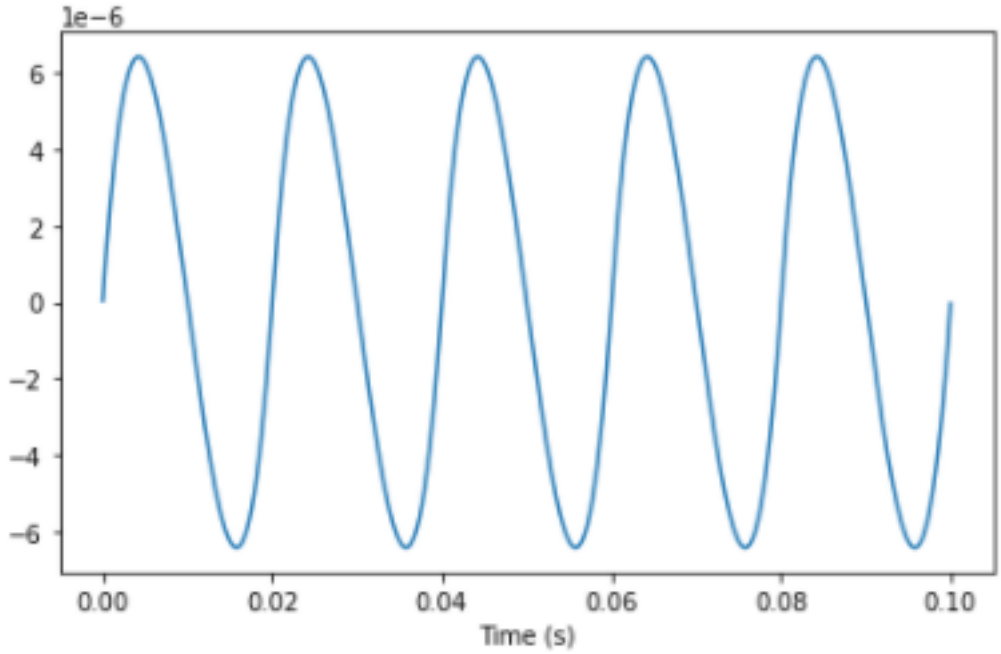

Рис. 17: Применение `integrate`

Полученный результат так же представлят собой кубическую кривую.

Данный график напоминает синусоиду, так как `integrate` действует как фильтр НЧ. Для более понятного отображения посмотрим на спектр полученного сигнала:

```
out_wave2.make_spectrum().plot(high=500)
```

Листинг 15: Применение спектра

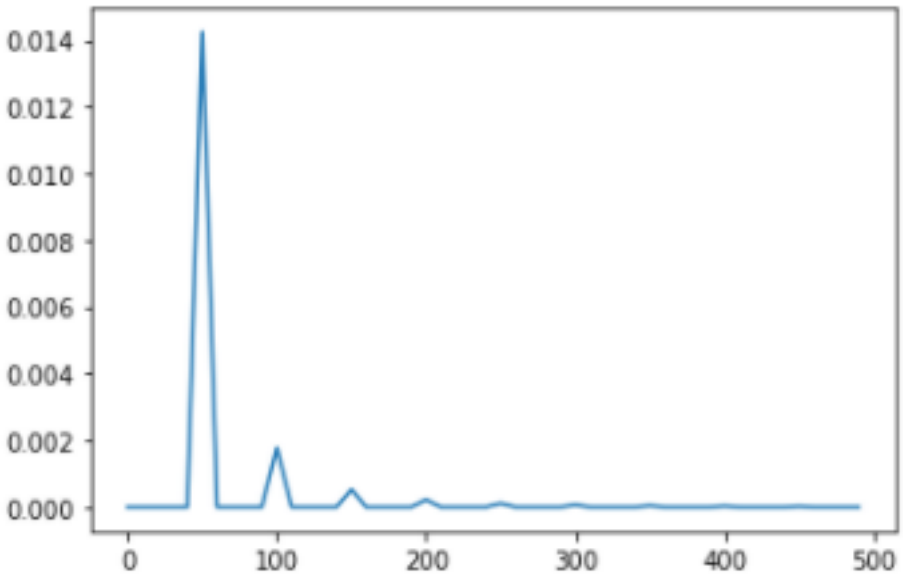

Рис. 18: Полученный спектр

В результате стало понятно, что это график не синусоиды, а именно кубической кривой.

5 Часть №5: Вторая разность и вторая производная

В пятом пункте лабораторной работы нам необходимо изучить влияние второй разности и второй производной. Для этого надо создать `CubicSignal`, вычислить двойную разность применив `diff` и проанализировать результат. После этого вычислить вторую производную, дважды применив `differentiate` к спектру.

Начнем с создания `CubicSignal` и сразу выведем его на экран:

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

Листинг 16: Создание и вывод `CubicSignal`

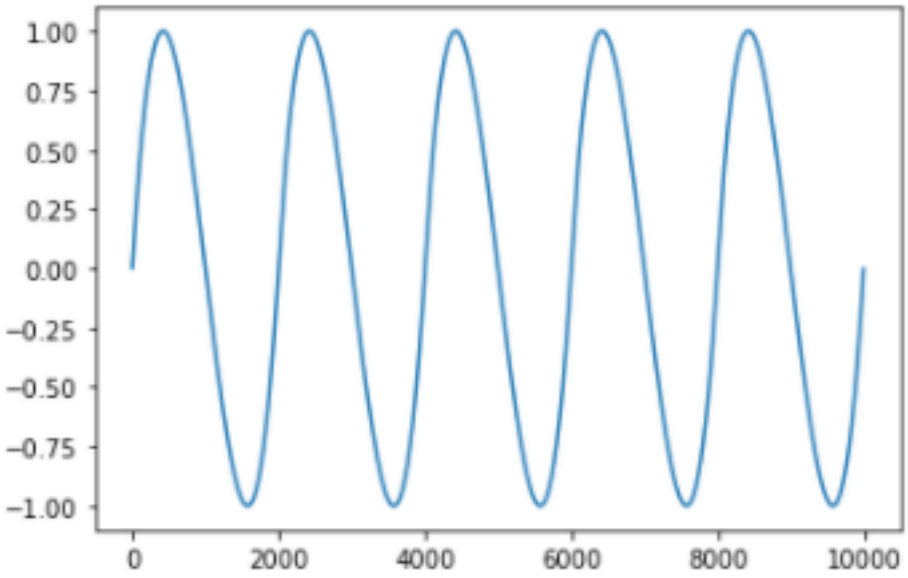

Рис. 19: `CubicSignal`

Теперь применим к полученному сигналу `diff` и выведем полученный график:

```
out_wave = in_wave.diff()
out_wave.plot()
```

Листинг 17: Применение `diff`

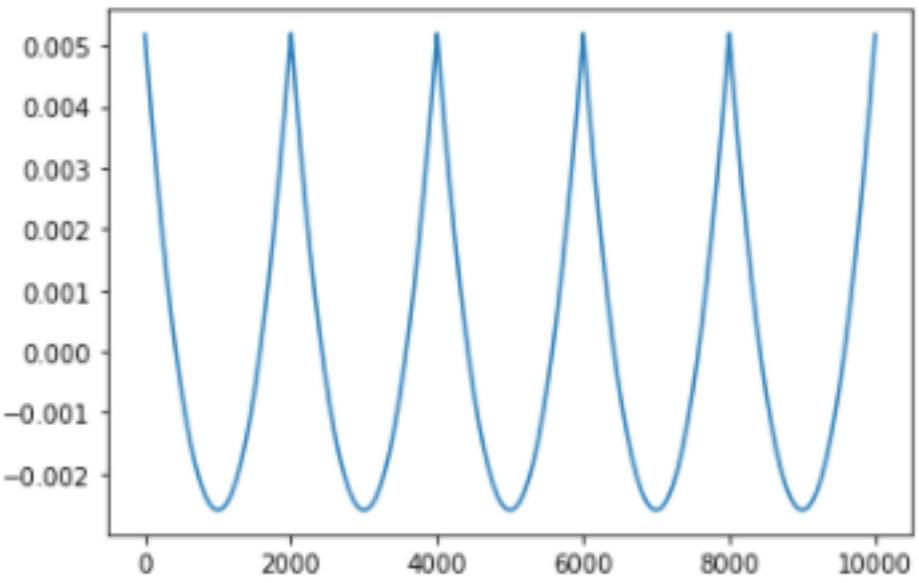

Рис. 20: Применение `diff`

В результате мы получили параболу.

Опять применим функцию `diff`:

```
out_wave = out_wave.diff()
out_wave.plot()
```

Листинг 18: Второе применение `diff`

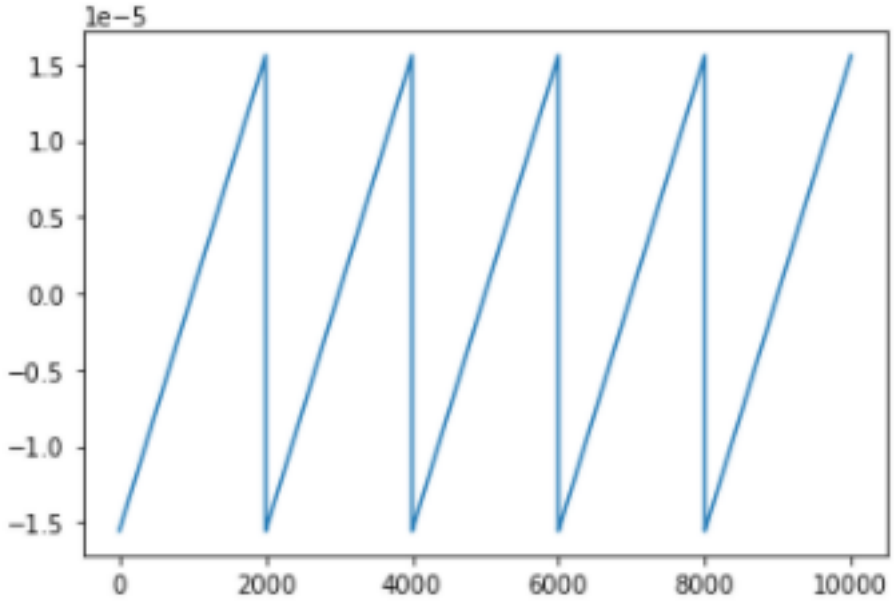

Рис. 21: Второе применение `diff`

Теперь мы получили пилообразную волну.

После этого применим к исходному пилообразному сигналу два раза `differentiat` и посмотрим на результат:

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

Листинг 19: Применение `differentiate`

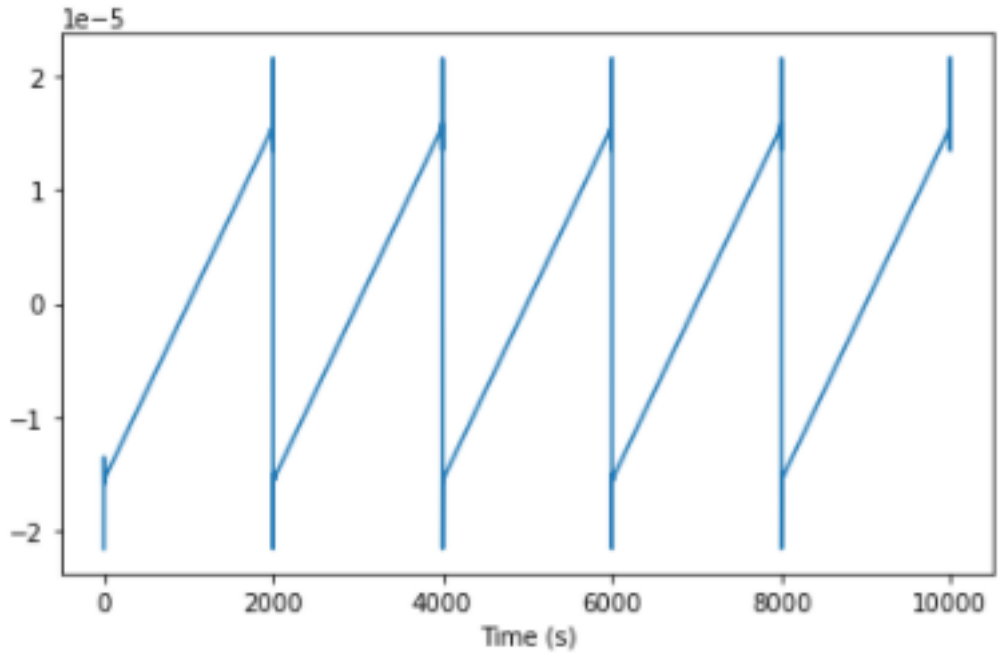

Рис. 22: Применение `differentiate`

В результате мы получили такой же пилообразный сигнал, но с неким "шумом". Это связано с тем, что производная параболического сигнала не определена в некоторых точках.

Затем найдем нужный фильтр для первой производной:

```
diff_window = np.array([-1.0, 2.0, -1.0])
padded = zero_pad(diff_window, len(in_wave))
diff_wave = Wave(padded, framerate=in_wave.framerate)
diff_filter = diff_wave.make_spectrum()
diff_filter.plot(label='2nd diff')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
```

Листинг 20: Нахождение фильтра для первой производной

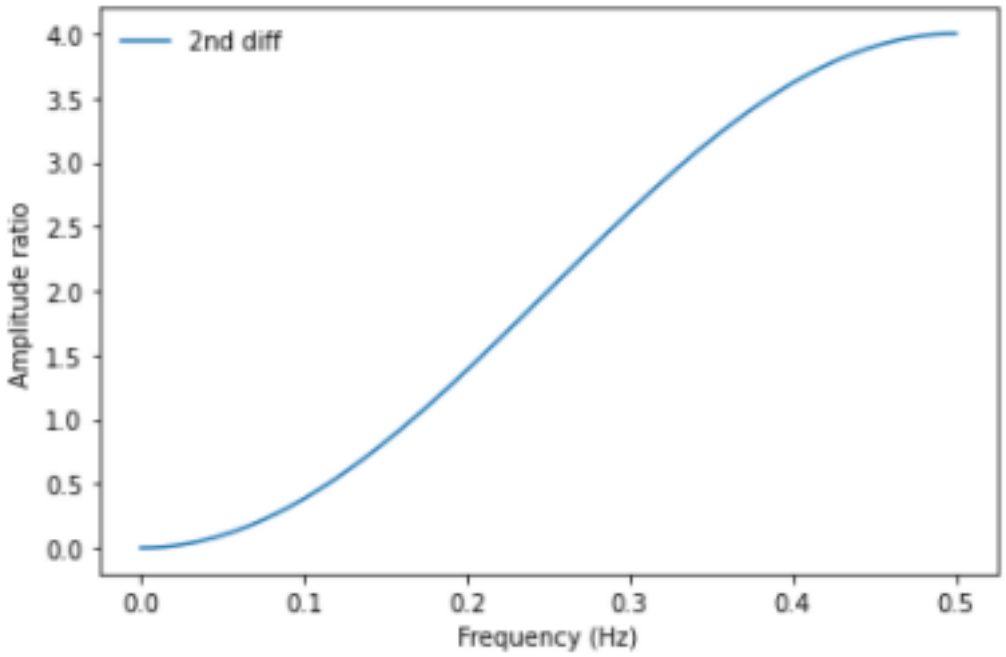

Рис. 23: Нахождение фильтра для первой производной

Чтобы найти фильтр для второй производной нужно возвести фильтр первой производной в квадрат:

```
deriv_filter = in_wave.make_spectrum()
deriv_filter.hs = (PI2 * 1j * deriv_filter.fs)**2
deriv_filter.plot(label='2nd deriv')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
```

Листинг 21: Нахождение фильтра для второй производной

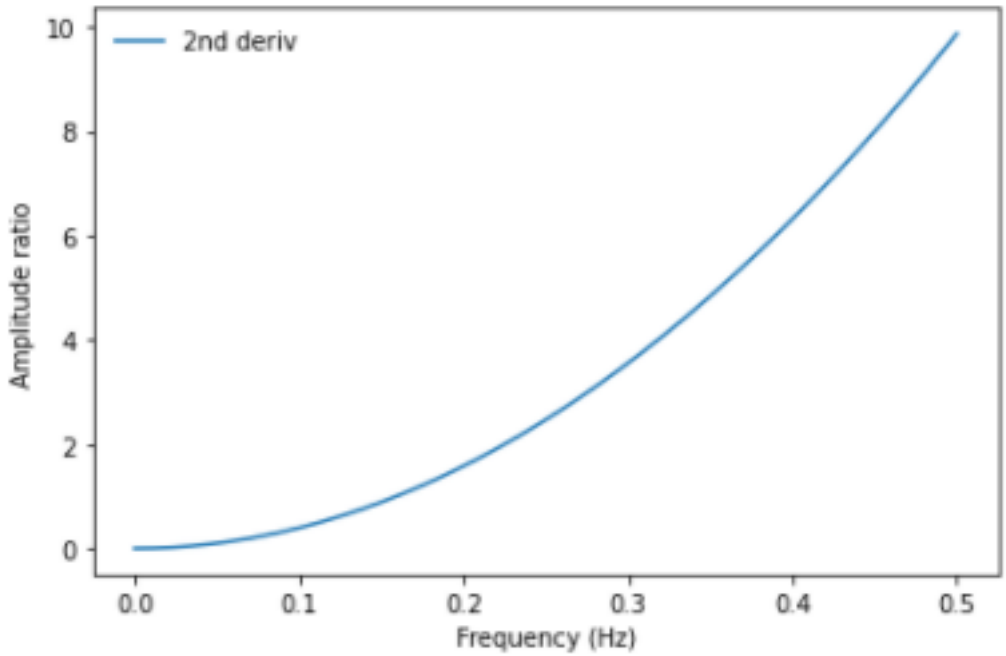

Рис. 24: Нахождение фильтра для второй производной

Полученный фильтр для второй производной является параболическим и поэтому он усиливает самые высокие частоты

Теперь представим оба полученных графика на одном:

```
diff_filter.plot(label='2nd diff')
deriv_filter.plot(label='2nd deriv')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')
```

Листинг 22: Сравнение графиков

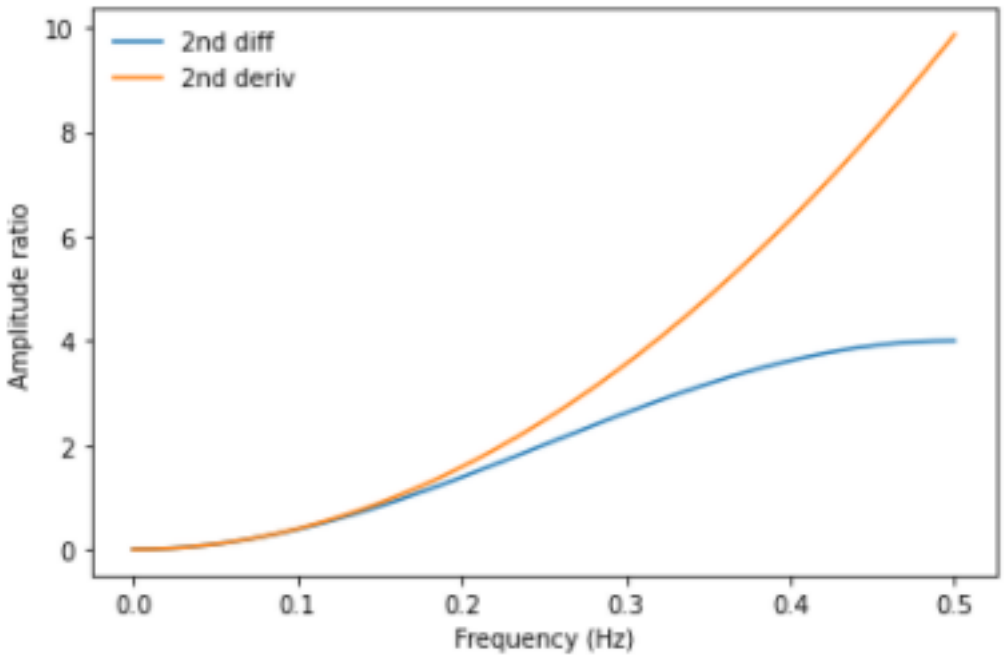

Рис. 25: Сравнение графиков

В результате можно сказать, что оба полученных фильтра являются высокочастотными, которые усиливают компоненты наивысшей частоты. Именно поэтому на низких частотах различий совершенно никаких нет, но они становятся заметными на высоких частотах.

6 Выводы

В результате выполнения лабораторной работы мы научились дифференцировать и интегрировать функции, разобрались с работой `diff`, `differentiate`, `cumsum` и `integrate`. Кроме того, изучили влияние двойного интегрирования, второй разности и второй производной на разные сигналы.