

Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

Дифференцирование и интегрирование

Работу выполнил студент
3-го курса, группа 3530901/80201
Солянкин Илья Андреевич

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург 2021

Содержание

1	Часть №1: chap09.ipynb	6
2	Часть №2: diff и differentiate	9
3	Часть №3: cumsum и integrate	12
4	Часть №4: Двойное интегрирование	16
5	Часть №5: Вторая разность и вторая производная	20
6	Выводы	25

Список иллюстраций

1	Непериодические данные Facebook	6
2	Спектр данных Facebook	6
3	Полученные результаты	7
4	Результаты сравнения вычисленного фотношения с фильтром	8
5	Итоговое сравнение	8
6	Треугольный сигнал	9
7	Применение diff	10
8	Применение differentiate	10
9	Прямоугольный сигнал	12
10	Применение cumsum	13
11	Применение integrate	14
12	Результат смещения и нормализации сигналов	14
13	Результат сравнения	15
14	Пилообразный сигнал	16
15	Применение cumsum	17
16	Второе применение cumsum	17
17	Применение integrate	18
18	Полученный спектр	19
19	CubicSignal	20
20	Применение diff	21
21	Второе применение diff	21
22	Применение differentiate	22

23	Нахождение фильтра для первой производной	23
24	Нахождение фильтра для второй производной	23
25	Сравнение графиков	24

Листинги

1	Ratio spectrum	7
2	Сравнение вычисленного фотношения с фильтром	7
3	Сравнение вычисленного фотношения с фильтром	8
4	Создание и вывод треугольного сигнала	9
5	Применение diff	9
6	Вычисление спектра и применение differentiate	10
7	Создание и вывод прямоугольного сигнала	12
8	Применение cumsum	12
9	Применение integrate	13
10	Смещение и нормализация сигналов	14
11	Создание и вывод пилообразного сигнала	16
12	Применение cumsum	16
13	Второе применение cumsum	17
14	Применение integrate	18
15	Применение спектра	18
16	Создание и вывод CubicSignal	20
17	Применение diff	20
18	Второе применение diff	21
19	Применение differentiate	22
20	Нахождение фильтра для первой производной	22
21	Нахождение фильтра для второй производной	23
22	Сравнение графиков	24

1 Часть №1: chap09.ipynb

В первой части девятой 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)')
```

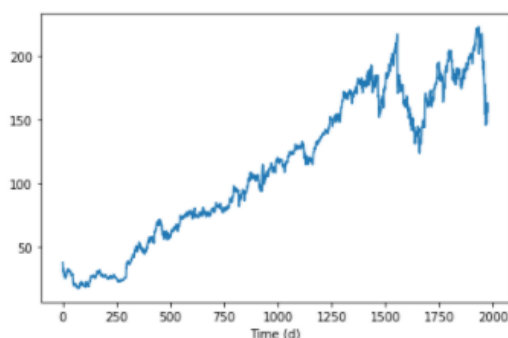


Рис. 1: Непериодические данные Facebook

После этого посмотрим на спектр полученных данных:

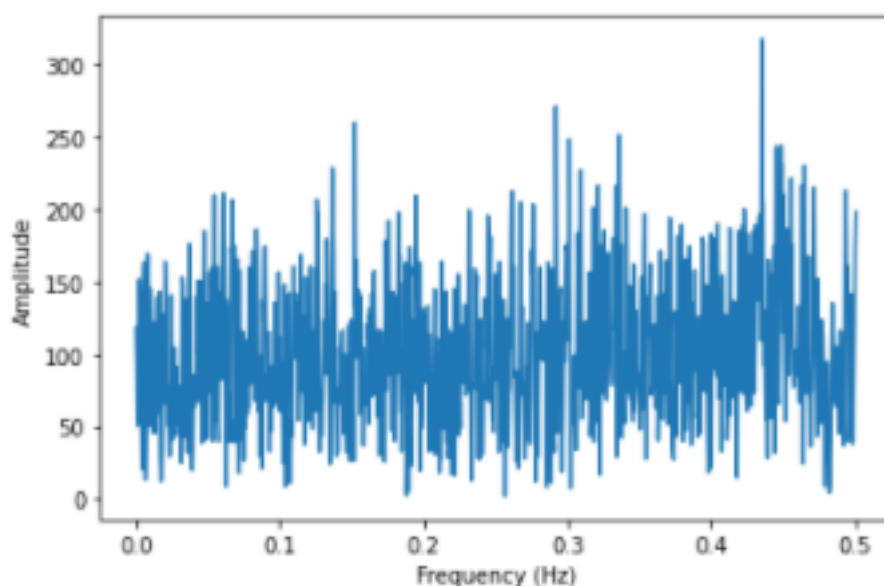


Рис. 2: Спектр данных Facebook

Теперь, т.к. между гармониками входные компоненты маленькие, поэтому установим эти отношения на NaN и посмотрим на результат:

```
1 ratio_spectrum = out_spectrum.ratio(in_spectrum, thresh=1)
2 ratio_spectrum.plot(marker='.', ms=4, ls='')
3
4 decorate(xlabel='Frequency (Hz)',
5          ylabel='Amplitude ratio',
6          yscale='log')
```

Листинг 1: Ratio spectrum

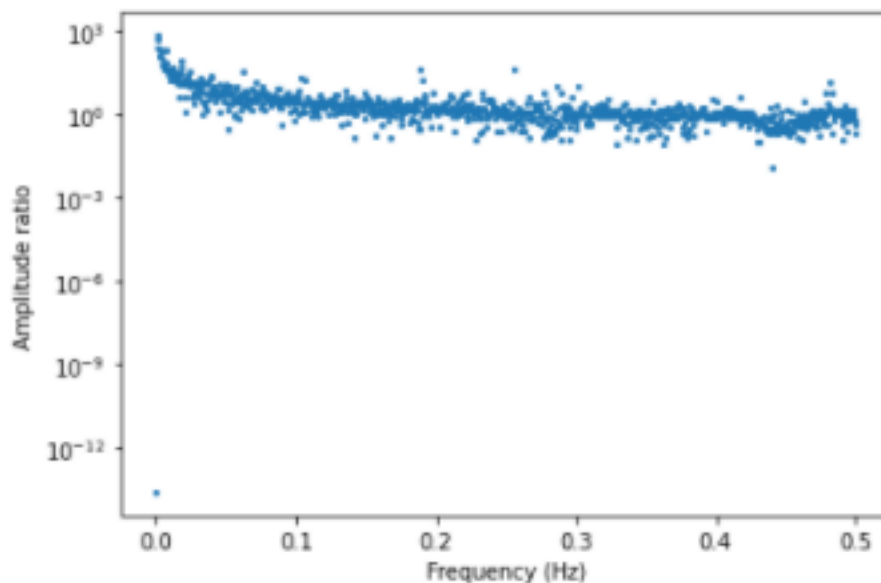


Рис. 3: Полученные результаты

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

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

Листинг 2: Сравнение вычисленного фотношения с фильтром

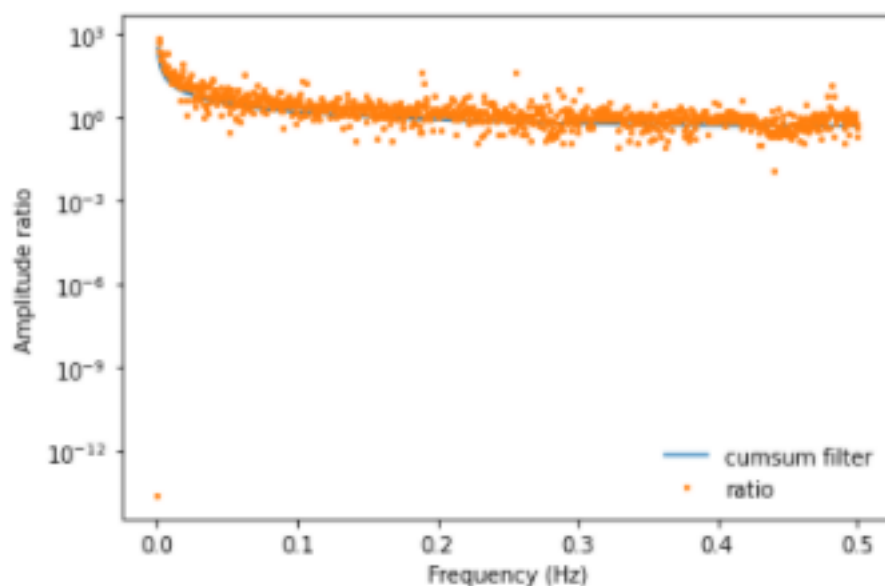


Рис. 4: Результаты сравнения вычисленного фотношения с фильтром

Наконец, сравним полученные результаты с входными:

```

1 out_wave.plot(label='summed', alpha=0.7)
2
3 cumsum_filter.hs[0] = 0
4 out_wave2 = (in_spectrum * cumsum_filter).make_wave()
5 out_wave2.plot(label='filtered', alpha=0.7)
6
7 decorate(xlabel='Time (s)')
```

Листинг 3: Сравнение вычисленного фотношения с фильтром

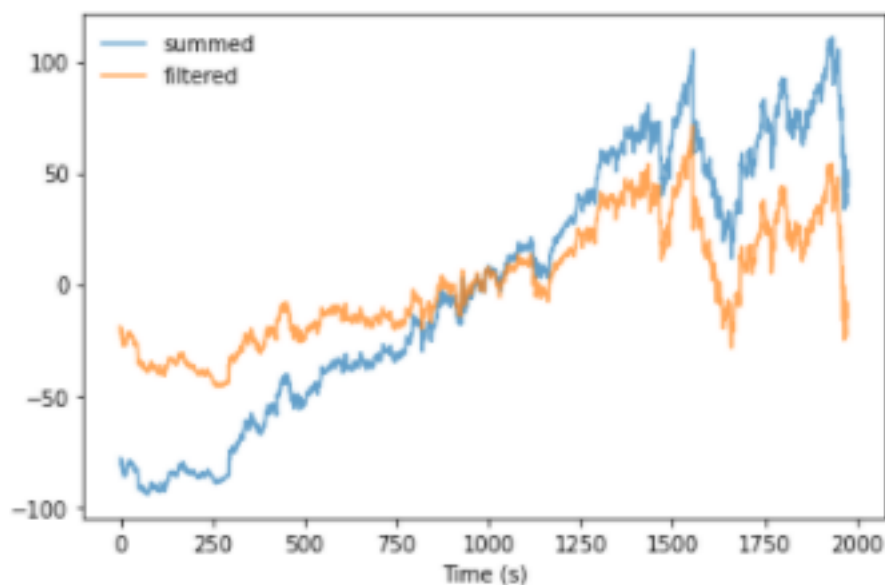


Рис. 5: Итоговое сравнение

Как видно в итоговом сравнении, полученные графики не равны.

2 Часть №2: diff и differentiate

Во втором пункте девятой 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:

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

Листинг 4: Создание и вывод треугольного сигнала

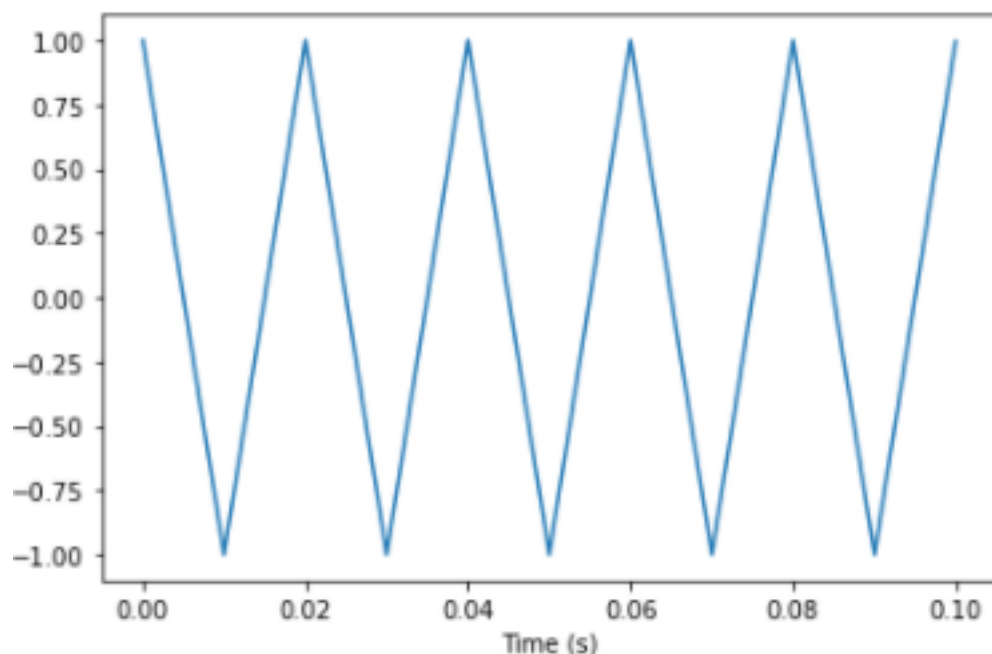


Рис. 6: Треугольный сигнал

Теперь применим `diff` и выведем полученный график:

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

Листинг 5: Применение `diff`

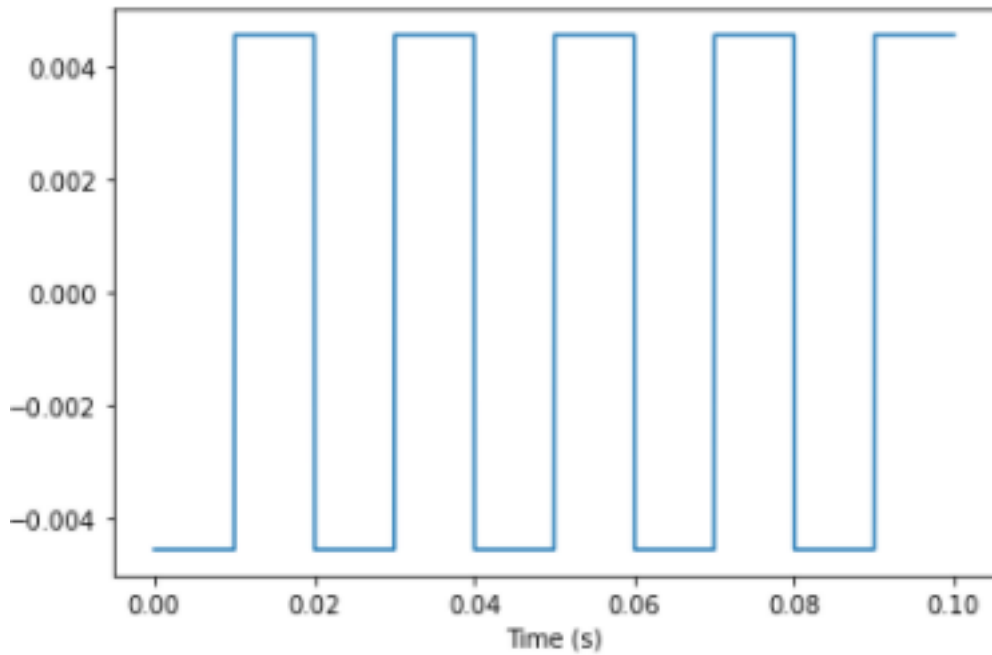


Рис. 7: Применение diff

Как видно - diff треугольной волны - прямоугольная волна.

Теперь вычислим спектр и применим к нему differentiate

```

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

Листинг 6: Вычисление спектра и применение differentiate

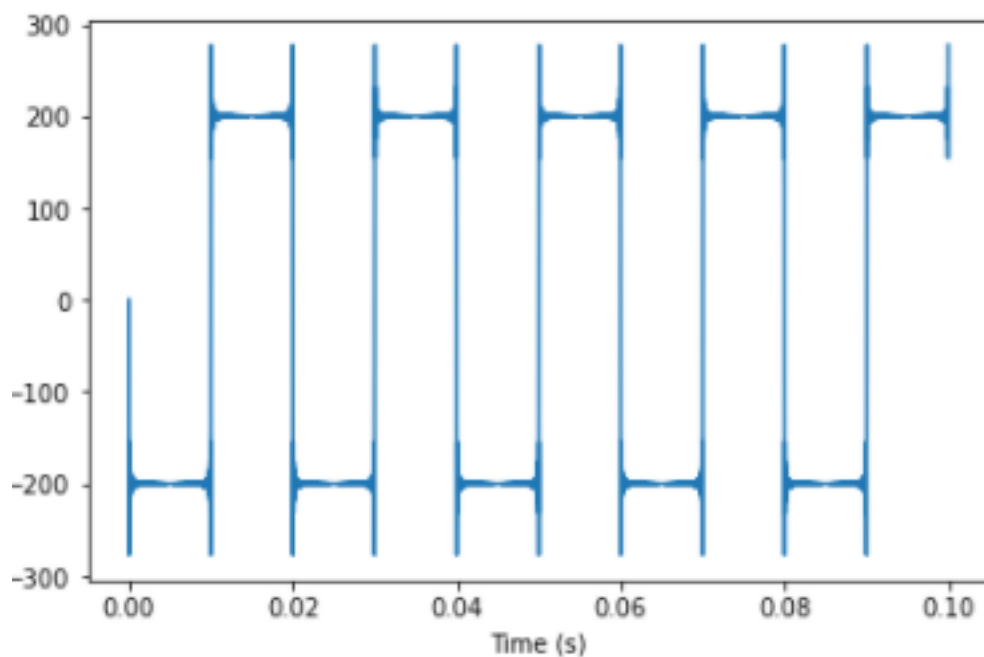


Рис. 8: Применение differentiate

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 девятой 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:

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

Листинг 7: Создание и вывод прямоугольного сигнала

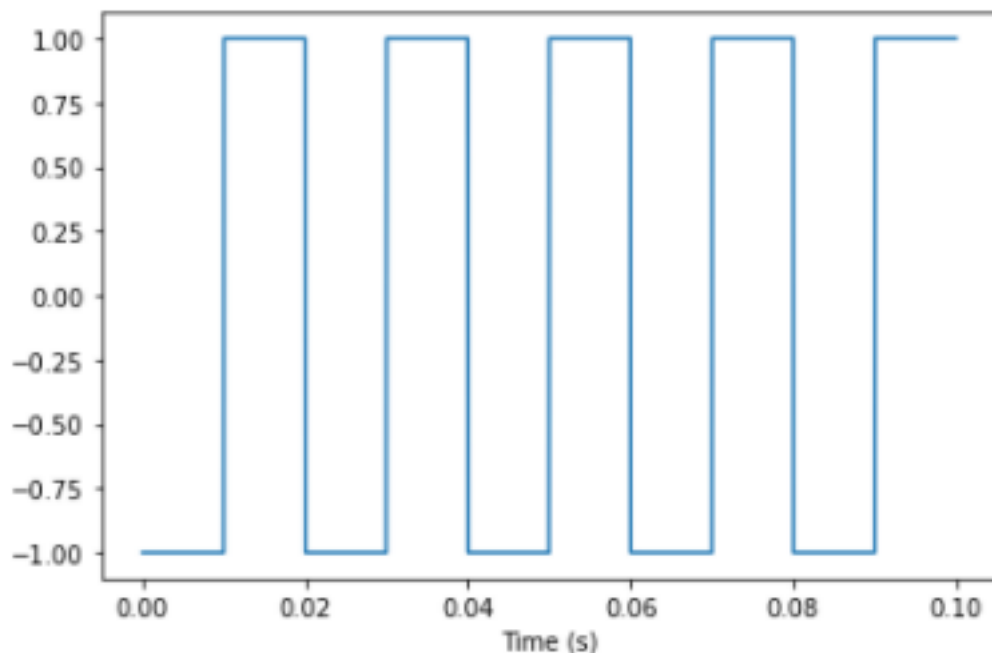


Рис. 9: Прямоугольный сигнал

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

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

Листинг 8: Применение `cumsum`

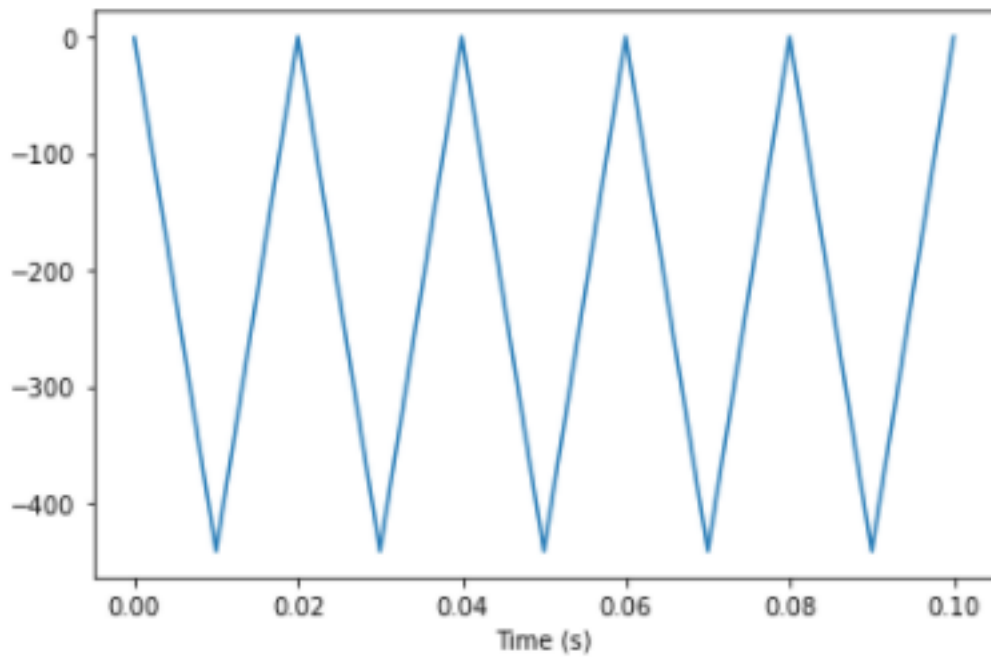


Рис. 10: Применение cumsum

Как видно - cumsum для прямоугольной волны - треугольная волна.

Кроме того спектральный интеграл так же является треугольной волной:

```

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

Листинг 9: Применение integrate

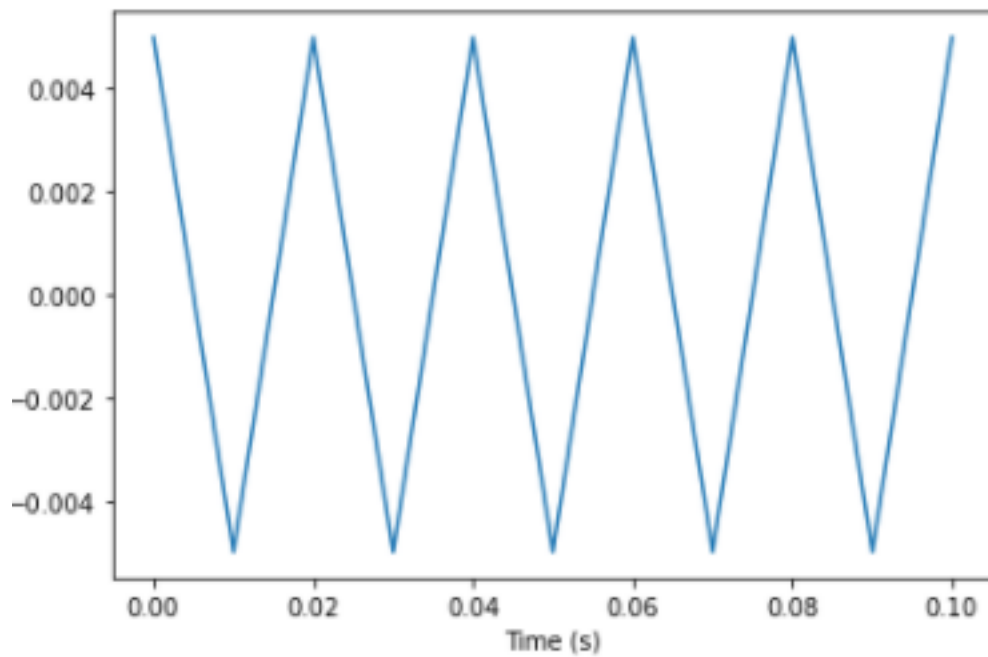


Рис. 11: Применение integrate

Доказать схожесть сигналов можно сместив и нормализовав обе волны:

```

1 out_wave.unbias()
2 out_wave.normalize()
3 out_wave2.normalize()
4 out_wave.plot()
5 out_wave2.plot()

```

Листинг 10: Смещение и нормализация сигналов

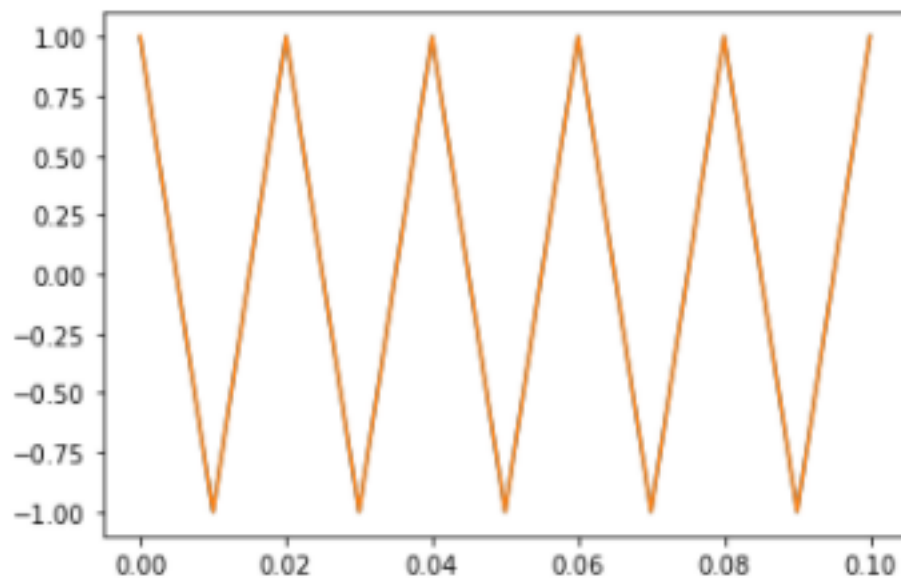


Рис. 12: Результат смещения и нормализации сигналов

Для того, чтобы точно определить насколько полученные графики схожи на-

пишем следующую строчку:

```
In [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 девятой 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:

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

Листинг 11: Создание и вывод пилообразного сигнала

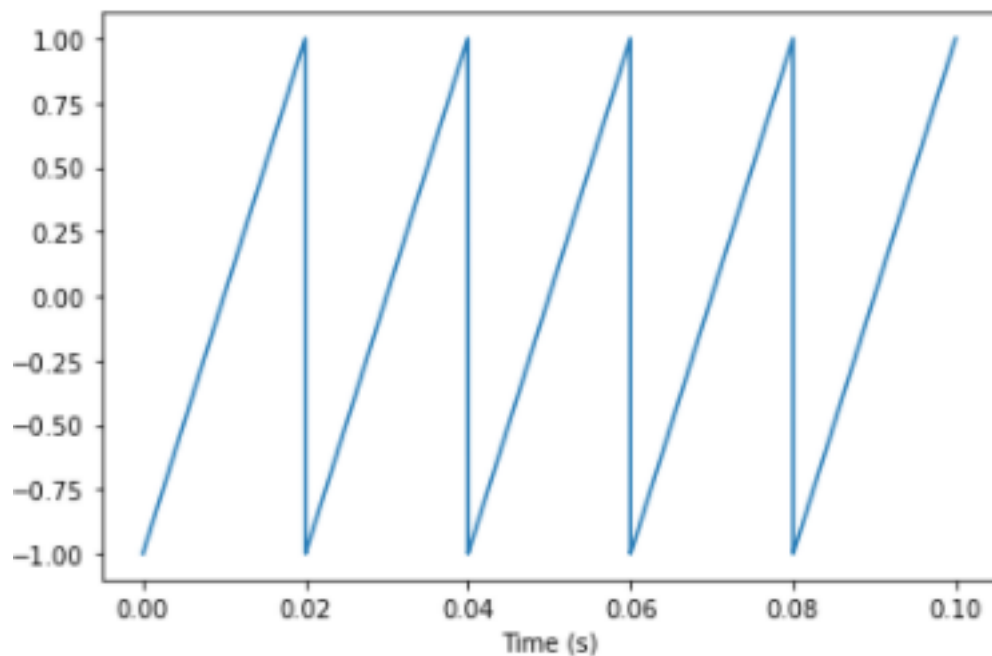


Рис. 14: Пилообразный сигнал

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

```
1 out_wave = in_wave.cumsum()
2 out_wave.unbias()
3 out_wave.plot()
4 decorate(xlabel='Time (s)')
```

Листинг 12: Применение `cumsum`

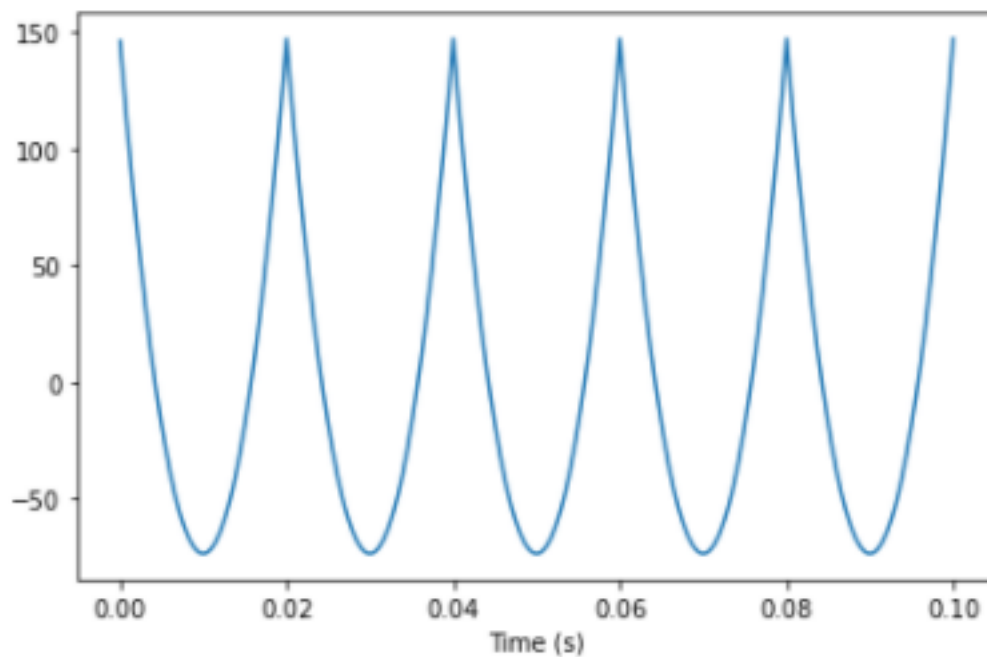


Рис. 15: Применение cumsum

Как видно - первая сумма пилообразного сигнала - парабола.

Опять применим функцию cumsum:

```

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

Листинг 13: Второе применение cumsum

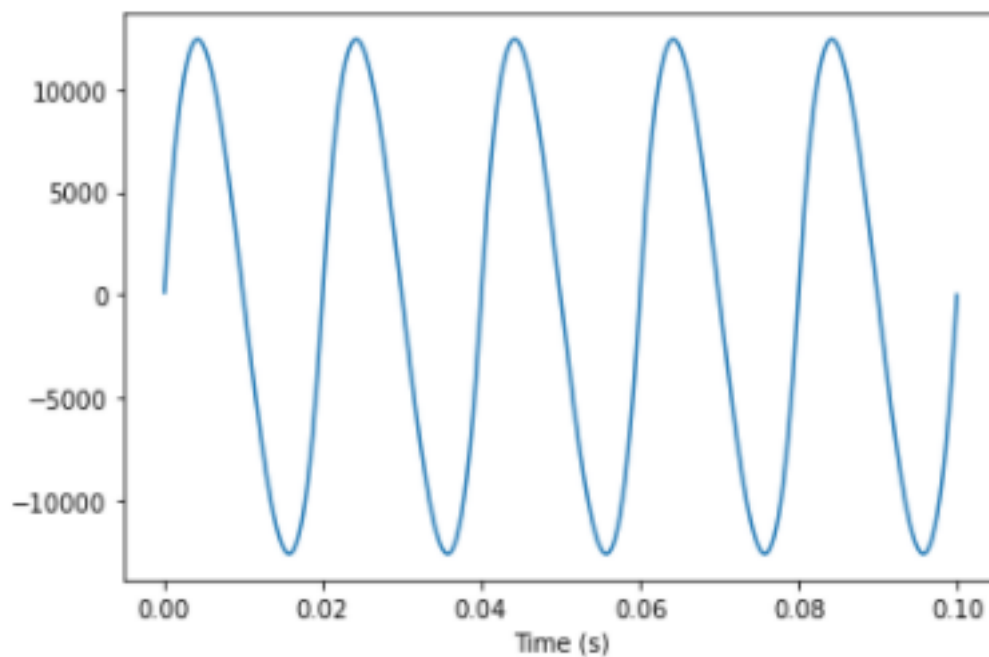


Рис. 16: Второе применение cumsum

Как видно - вторая сумма пилообразного сигнала - кубическая кривая.

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

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

Листинг 14: Применение `integrate`

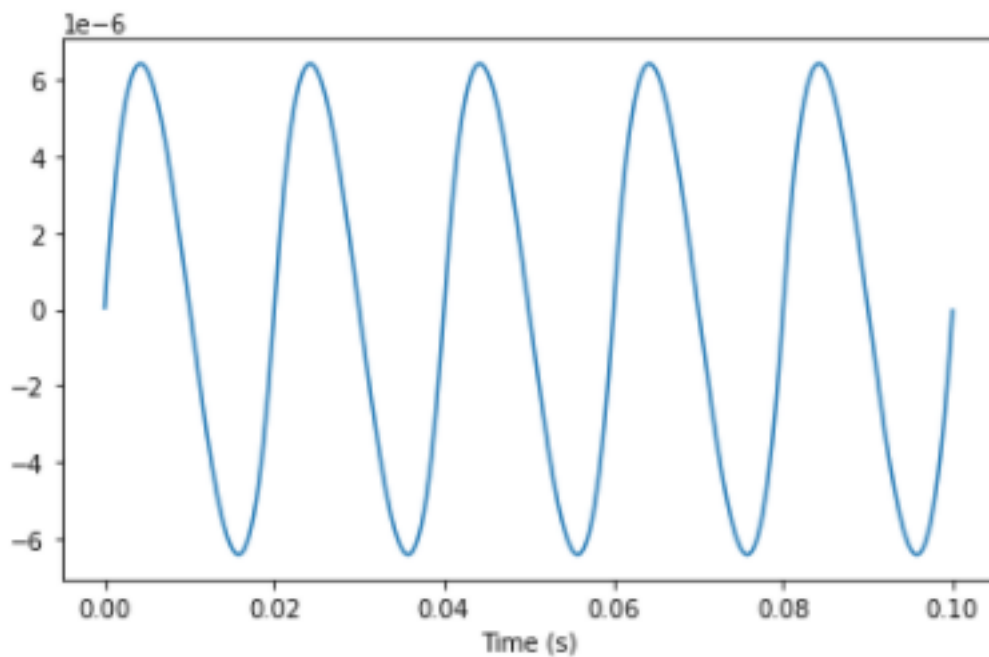


Рис. 17: Применение `integrate`

Полученный результат так же представляет собой кубическую кривую.

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

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

Листинг 15: Применение спектра

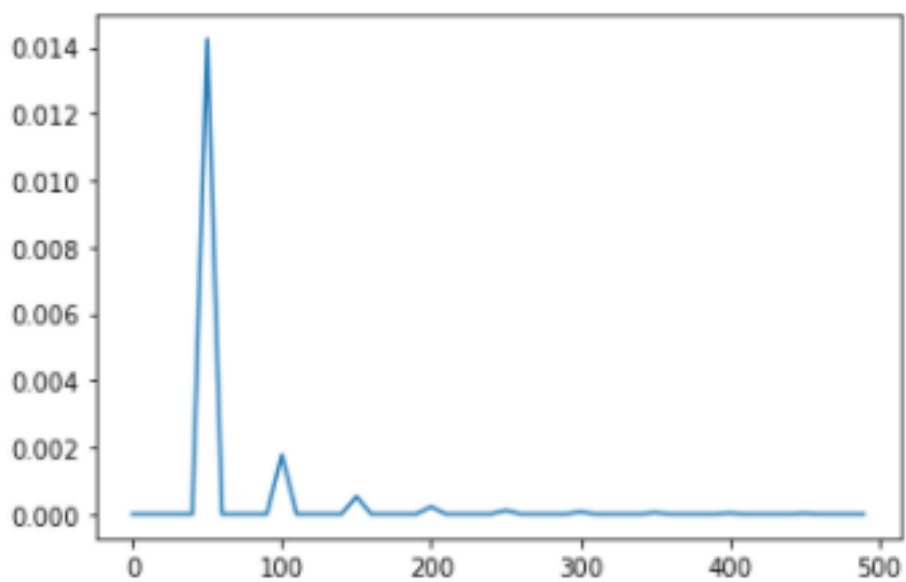


Рис. 18: Полученный спектр

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

5 Часть №5: Вторая разность и вторая производная

В пятом пункте девятой 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:

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

Листинг 16: Создание и вывод `CubicSignal`

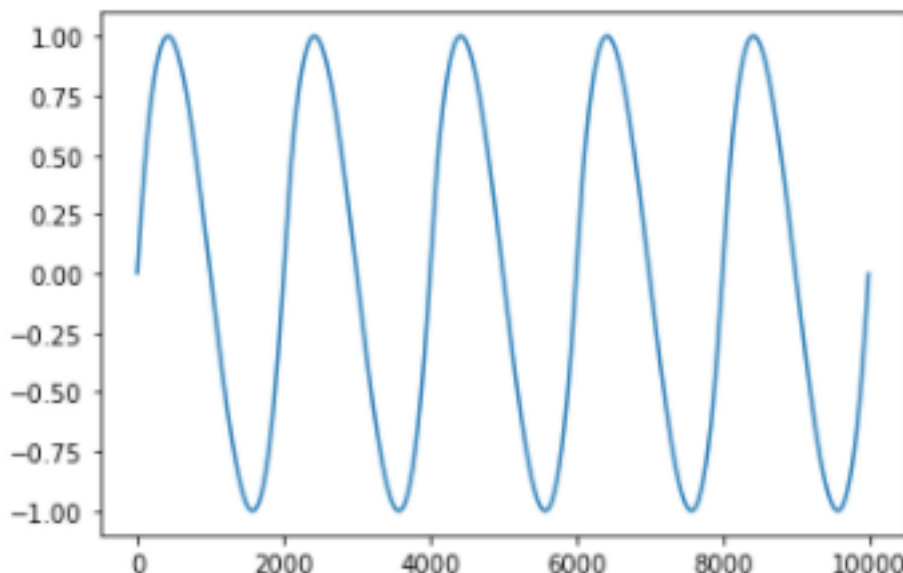


Рис. 19: `CubicSignal`

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

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

Листинг 17: Применение `diff`

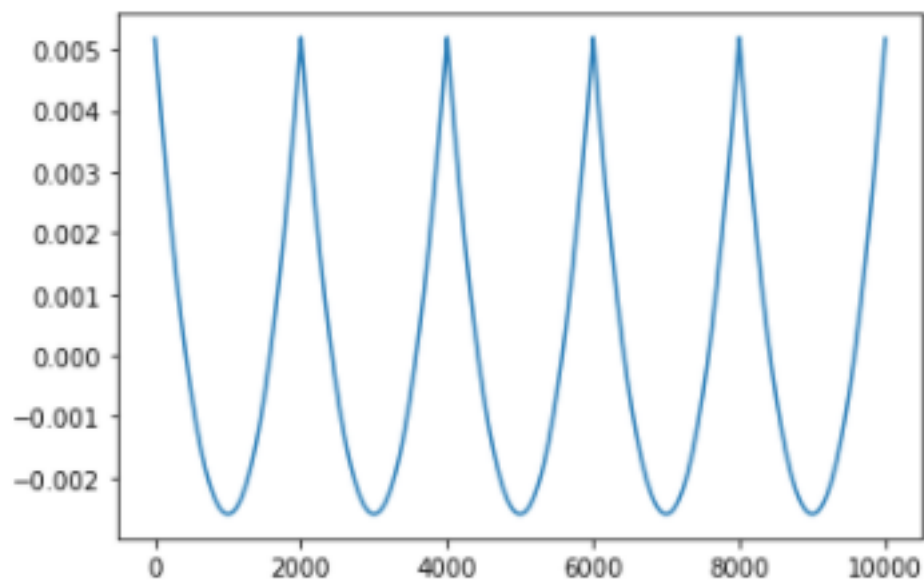


Рис. 20: Применение diff

В результате мы получили параболу.

Опять применим функцию diff:

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

Листинг 18: Второе применение diff

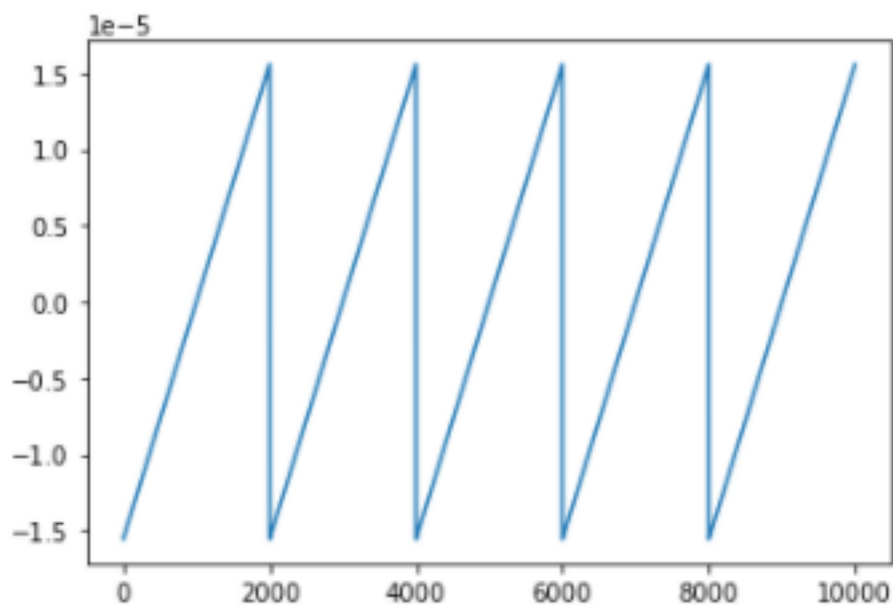


Рис. 21: Второе применение diff

Теперь мы получили пилообразную волну.

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

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

Листинг 19: Применение `differentiate`

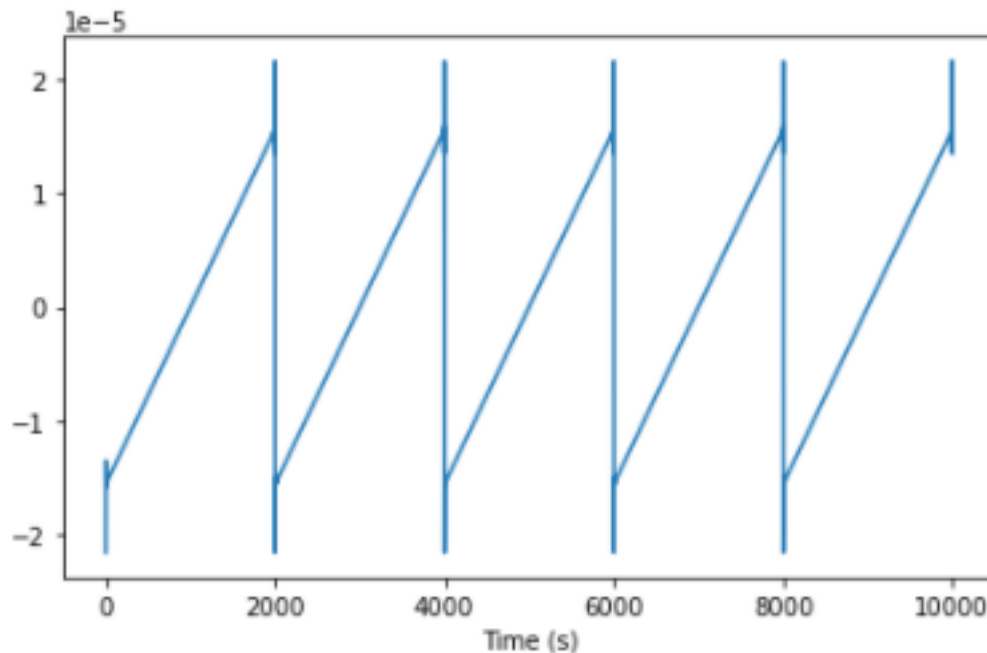


Рис. 22: Применение `differentiate`

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:

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

Листинг 20: Нахождение фильтра для первой производной

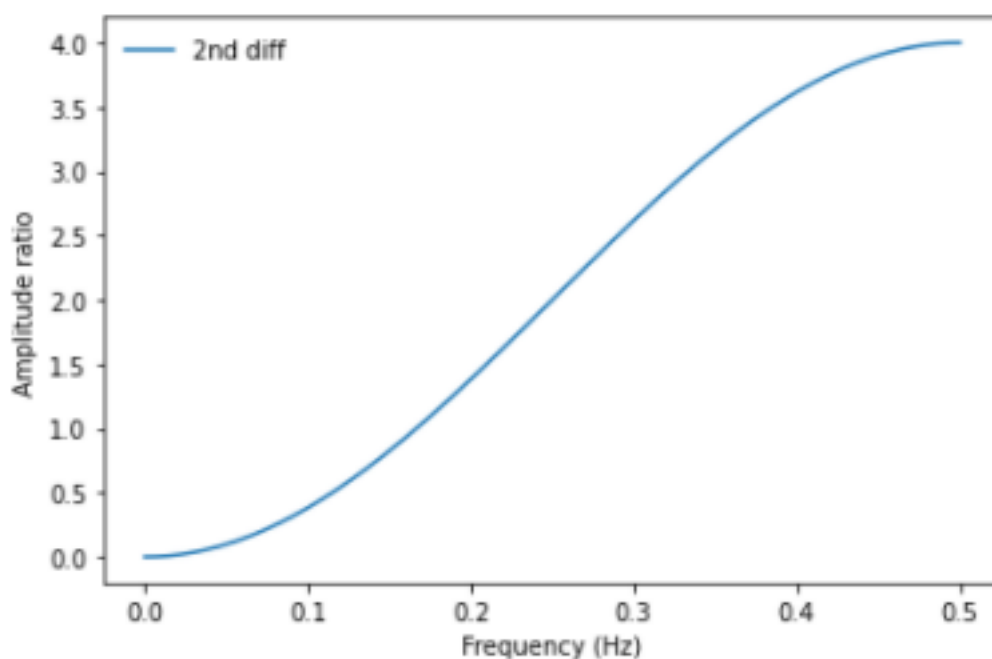


Рис. 23: Нахождение фильтра для первой производной

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

```

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

```

Листинг 21: Нахождение фильтра для второй производной

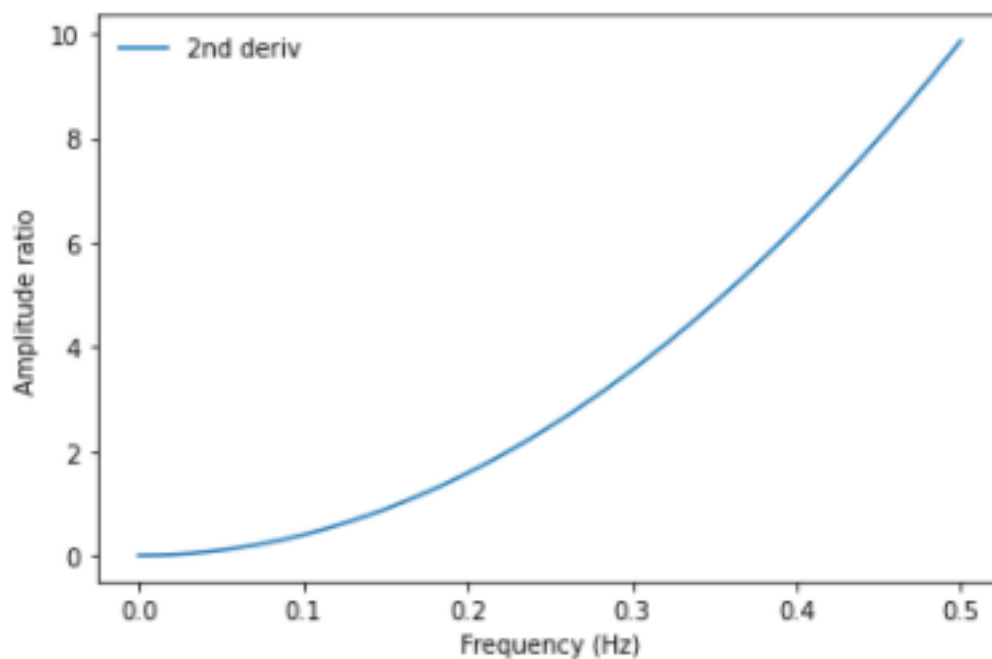


Рис. 24: Нахождение фильтра для второй производной

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

Теперь представим оба полученных графика на одном:

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

Листинг 22: Сравнение графиков

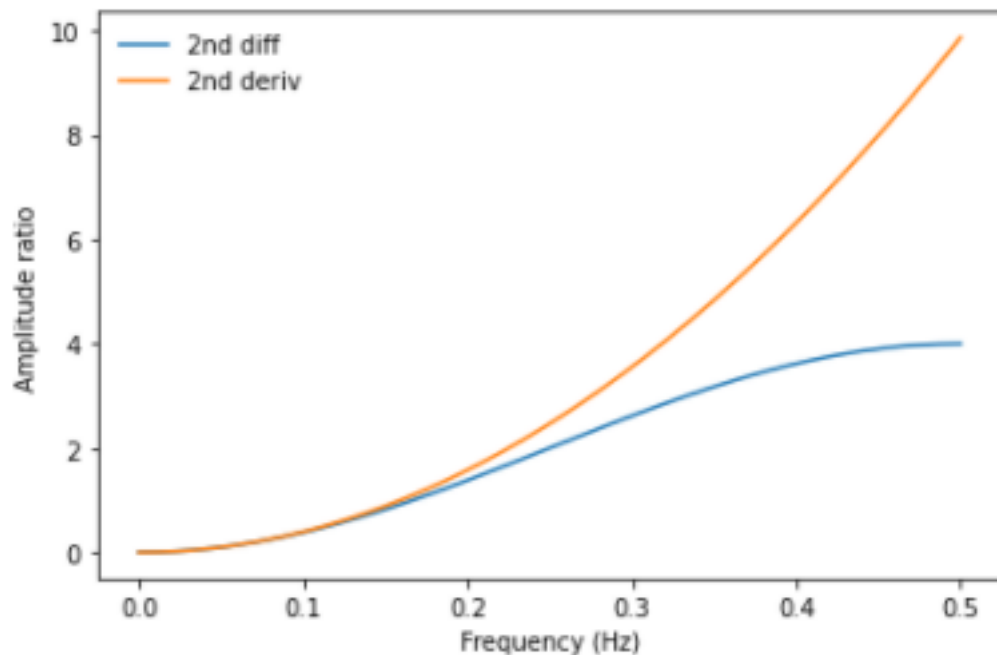


Рис. 25: Сравнение графиков

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