ThinkDSP. Лабораторная 9. Дифференцирование и интегрирование.

Шерепа Никита

14 мая 2021 г.

Содержание

Список иллюстраций

Листинги

1 Упражнение 9.2

1. Задание

   Создайте треугольный сигнал и напечатайте его. Примените `diff` к сигналу и напечатайте результат. Вычислите спектр треугольного сигнала, примените `differentiate` и напечатайте результат. Преобразуйте спектр обратно в сигнал и напечатайте его. Есть ли различия в воздействии `diff` и `differentiate` на этот сигнал?

2. Ход работы

   Создадим треугольный сигнал

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

Листинг 1: Создаем треугольный сигнал

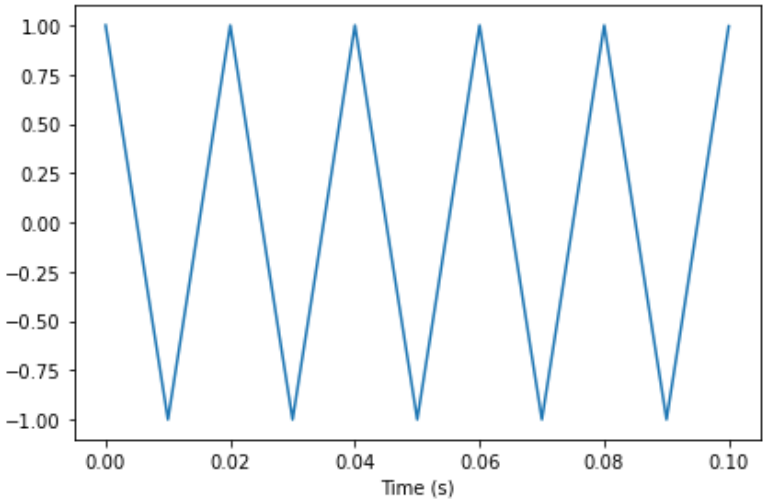

Рис. 1: Треугольный сигнал

Теперь применим в нему `diff`

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

Листинг 2: Применяем `diff`

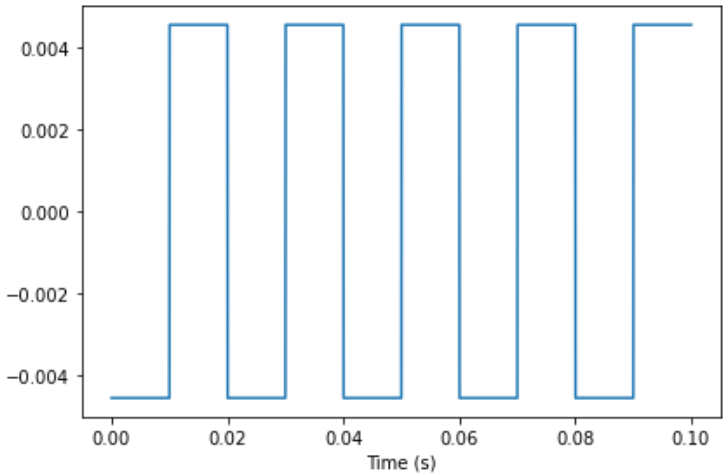

Рис. 2: Сигнал с применением `diff`

Видим, что получилась прямоугольная волна, что объясняет, почему гармоники в прямоугольной волне уменьшаются как $1/f$, по сравнению с треугольной волной, которая спадает как $1/f2$.

Теперь вычислим спектр треугольного сигнала и применим `differentiate`

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

Листинг 3: Спектр сигнала + `differentiate`

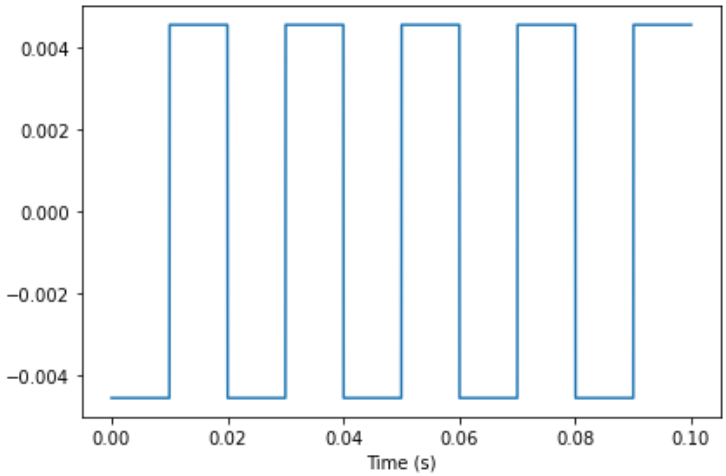

Рис. 3: Спектр сигнала + `differentiate`

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

Различия между `diff` и `differentiate` в том, что производная треугольной волны не определена в точках треугольника.

2 Упражнение 9.3

1. Задание

   Создайте прямоугольный сигнал и напечатайте его. Примените `cumsum` и напечатайте результат. Вычислие спектр прямоугольного сигнала, примените `integrate` и напечатайте результат. Преобразуйте спектр обратно в сигнал и напечатайте его. Есть ли различия в воздействии `cumsum` и `integrate` на этот сигнал?

2. Ход работы

   Создадим прямоугольный сигнал

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

Листинг 4: Создаем прямоугольный сигнал

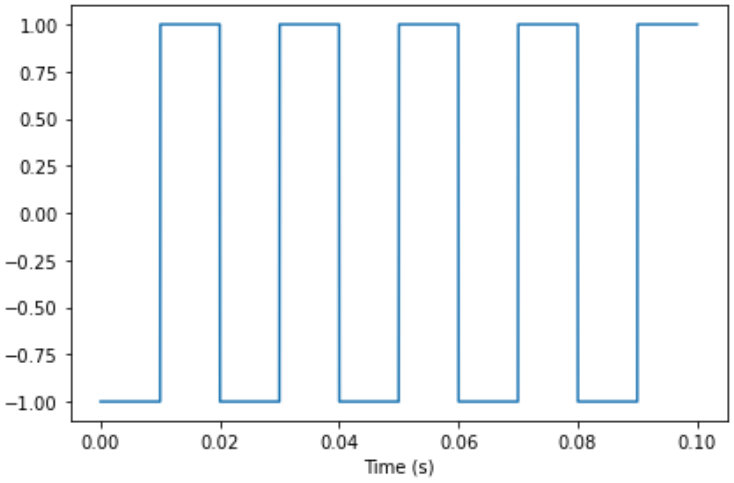

Рис. 4: Треугольный сигнал

Теперь применим в нему `cumsum`

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

Листинг 5: Применяем `cumsum`

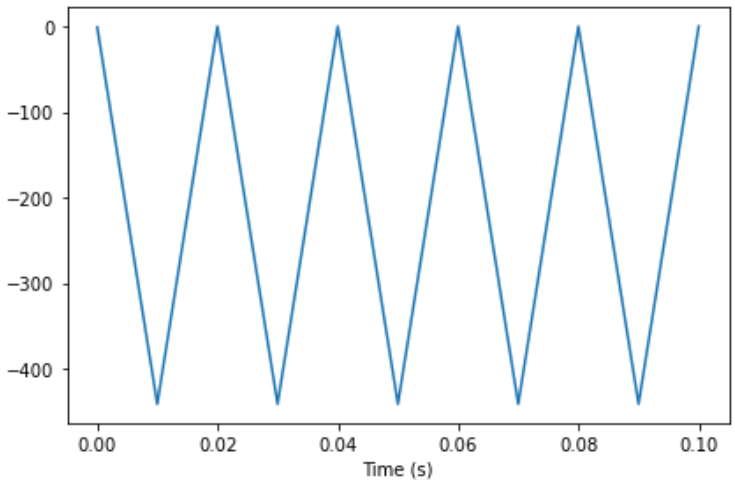

Рис. 5: Сигнал с применением `cumsum`

Видим, что получилась треугольная волна. По сути, мы сделали полностью противоположную операцию той, которая была в прошлом упражнении.

Теперь вычислим спектр треугольного сигнала и применим `integrate`

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

Листинг 6: Спектр сигнала + `integrate`

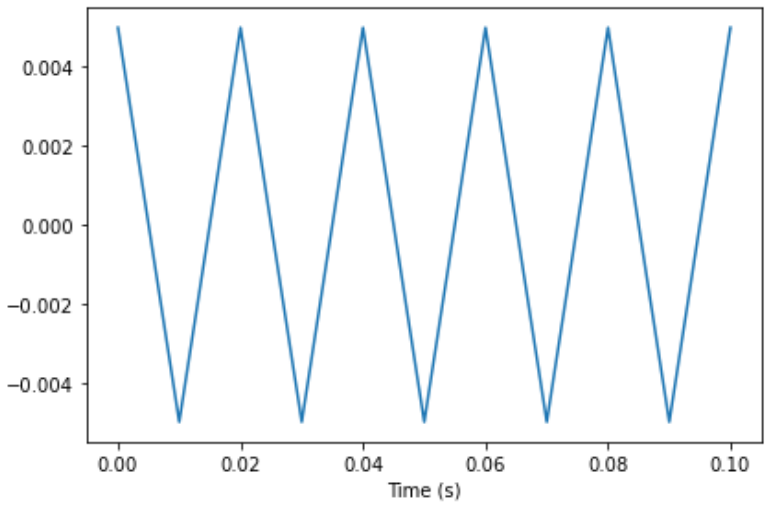

Рис. 6: Спектр сигнала + `integrate`

Теперь сравним результаты `cumsum` и `integrate`.

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

        out_wave.max_diff(out_wave2)

        Output
        0.0045351473922902175
```

Листинг 7: Сравниваем результаты `cumsum` и `integrate`

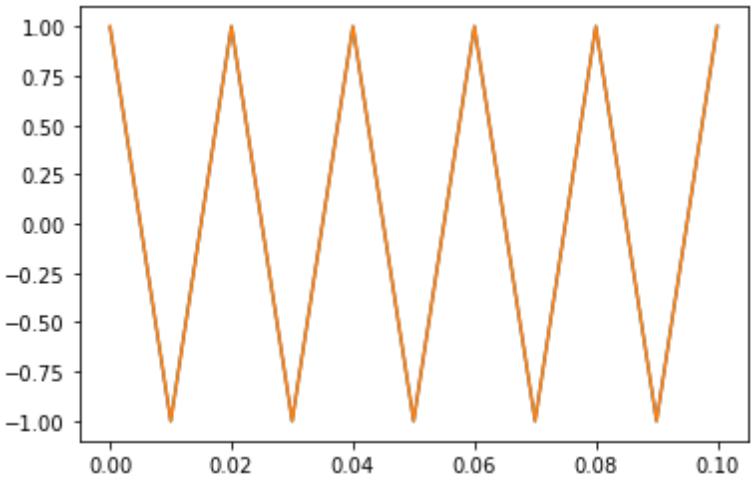

Рис. 7: Сравниваем результаты `cumsum` и `integrate`

Видим, что разница = 0.0045351473922902175, т.е отличий между `cumsum` и `integrate` почти нет.

3 Упражнение 9.4

1. Задание

   Создайте пилообразный сигнал, вычислите его спектр, а затем дважды примените `integrate`. Напечатайте результирующий сигнал и его спектр. Какова математическая форма сигнала? Почему он напоминает синусоиду?

2. Ход работы

   Создадим пилообразный сигнал

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

Листинг 8: Создаем пилообразный сигнал

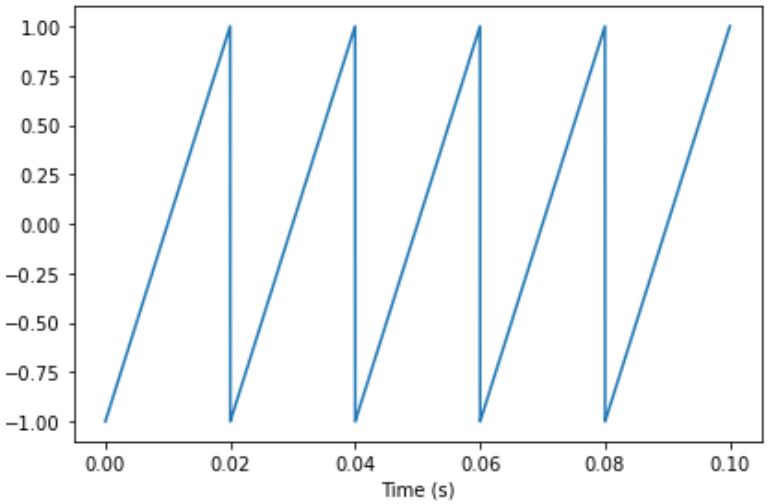

Рис. 8: Пилообразный сигнал

Для начала два раза применим `cumsum`

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

Листинг 9: 1ое применение `cumsum`

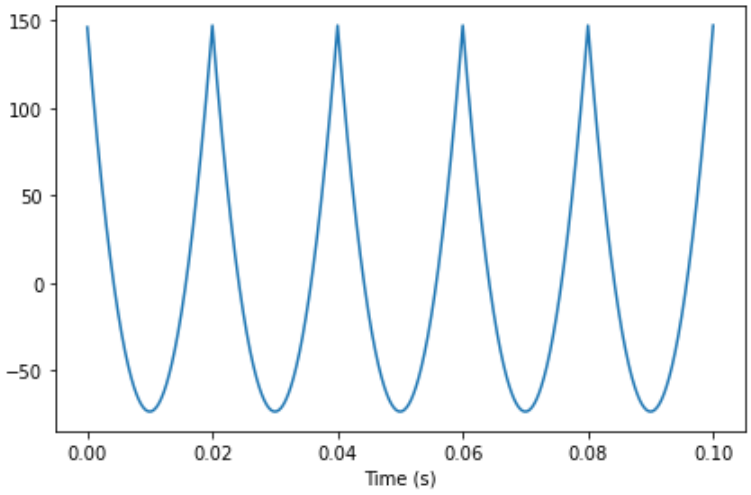

Рис. 9: 1ое применение `cumsum`

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

Листинг 10: 2ое применение `cumsum`

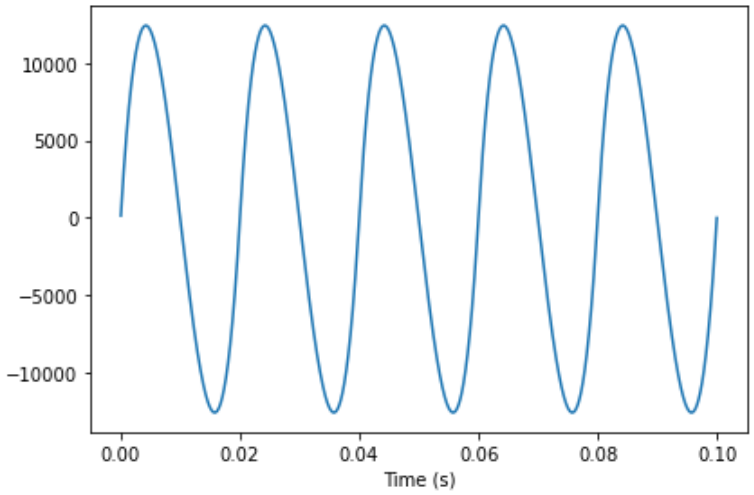

Рис. 10: 2ое применение `cumsum`

После первого применения получили параболу, а после второго - кубическую кривую

Теперь дважды применим `integrate`

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

Листинг 11: Двойное интегрирование

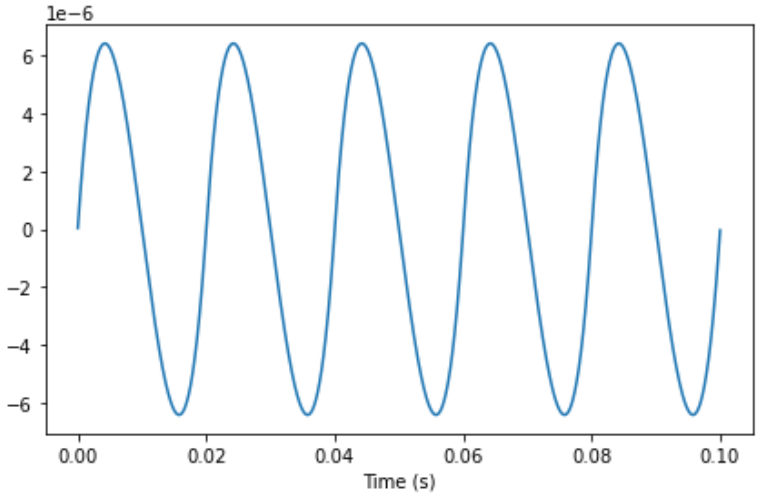

Рис. 11: Двойное интегрирование

Также получили кубическую кривую.

Теперь распечатаем итоговый спектр

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

Листинг 12: Итоговый спектр

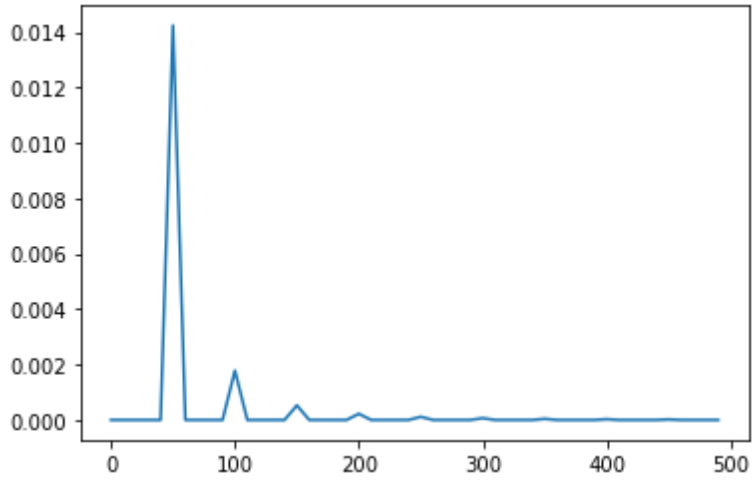

Рис. 12: Итоговый спектр

Результат напоминает синусоиду. Дело в том, что интеграция действует как фильтр нижних частот. И в результате мы отфильтровали почти все, кроме основной частоты.

4 Упражнение 9.5

1. Задание

   Создайте `CubicSignal`, определенный в `thinkdsp`. Вычислите вторую разность, дважды применив `diff`. Как выглядит результат? Вычислите вторую производную, дважды применив `differentiate` к спектру. Похожи ли результаты?

   Распечатайте фильтры, соответсвующие второй разнице и второй производной, и сравните их.

2. Ход работы

   Создадим кубический сигнал

```
from thinkdsp import CubicSignal

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

   Листинг 13: Создаем пилообразный сигнал

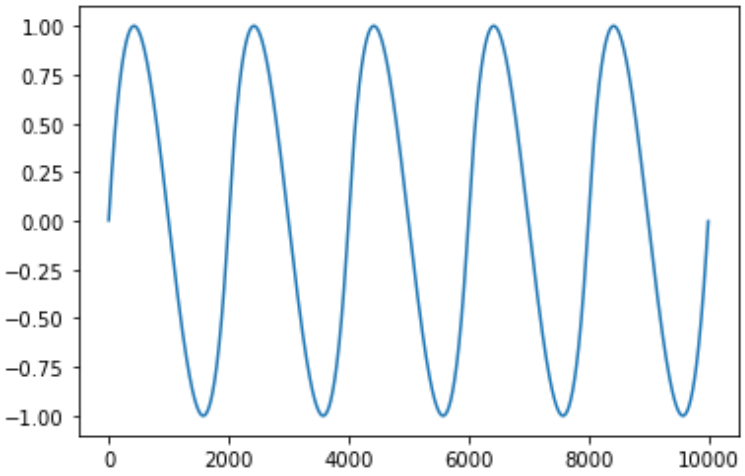

Рис. 13: Кубический сигнал

Дважды применим `diff`

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

```
2 out_wave.plot()
```

Листинг 14: 1ое применение `diff`

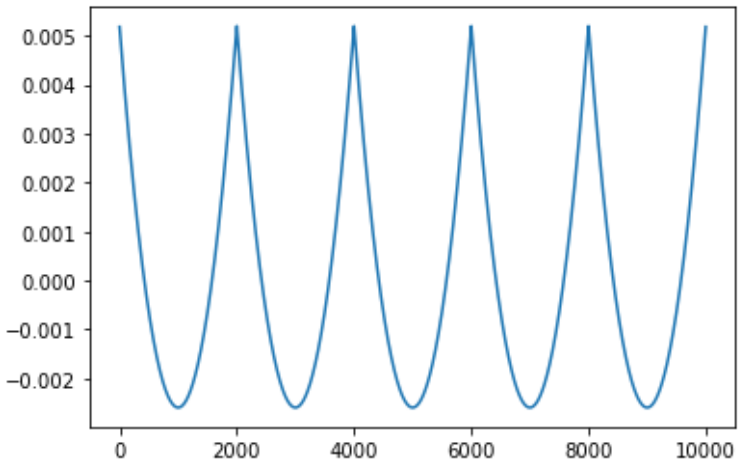

Рис. 14: 1ое применение `diff`

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

Листинг 15: 2ое применение `diff`

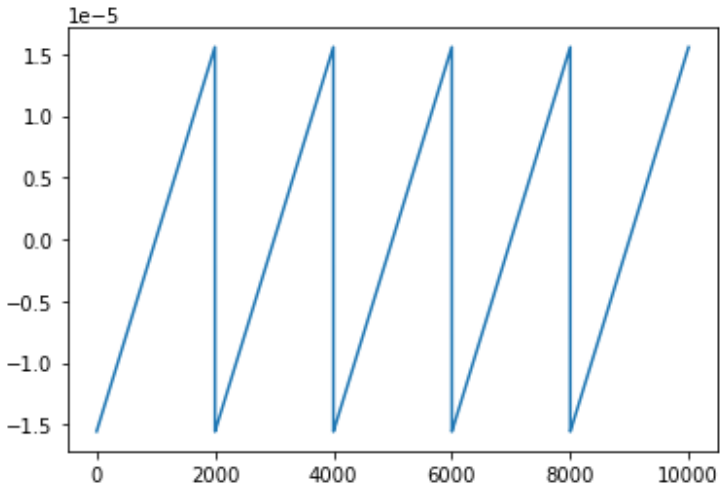

Рис. 15: 2ое применение `diff`

После первого применения получили параболу. После второго - пилообразный сигнал.

Теперь вычислим вторую производную - дважды применим `differentiate`

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

Листинг 16: Вычисляем вторую производную

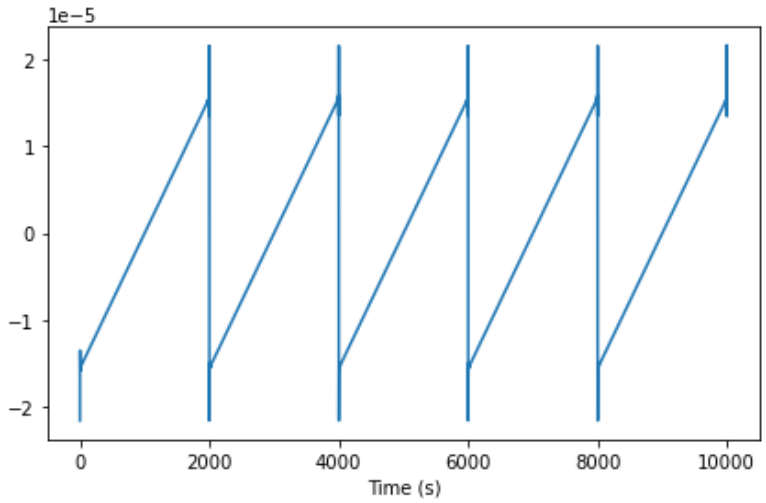

Рис. 16: Вторая производная

В итоге получили пилообразную форму с некоторым "звоном". Проблема в том, что производная параболического сигнала не определена в точках.

Теперь распечатаем фильтр второй разницы: применим ДПФ

```
from thinkdsp import zero_pad
from thinkdsp import Wave

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')
```

```
9
10          decorate(xlabel='Frequency (Hz)',
11          ylabel='Amplitude ratio')
```

Листинг 17: Фильтр второй разницы

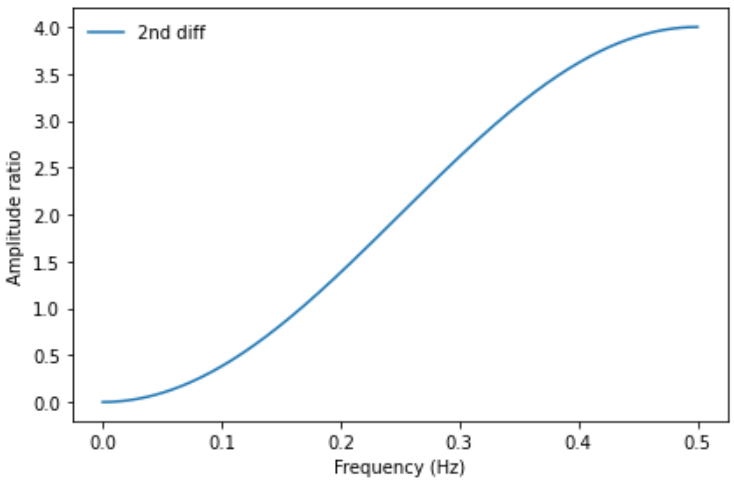

Рис. 17: Фильтр второй разницы

И фильтр второй производной: вычислим филтр первой производной и возведем его в квадрат

```
1           PI2 = np.pi * 2
2
3           deriv_filter = in_wave.make_spectrum()
4           deriv_filter.hs = (PI2 * 1j * deriv_filter.fs)**2
5           deriv_filter.plot(label='2nd deriv')
6
7           decorate(xlabel='Frequency (Hz)',
8           ylabel='Amplitude ratio')
```

Листинг 18: Фильтр второй производной

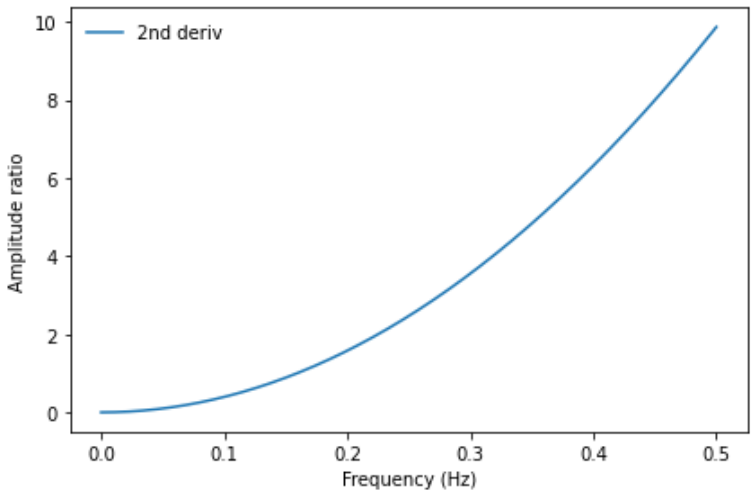

Рис. 18: Фильтр второй производной

Теперь сравним два получившихся фильтра

```
diff_filter.plot(label='2nd diff')
deriv_filter.plot(label='2nd deriv')

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

Листинг 19: Сравнение фильтров

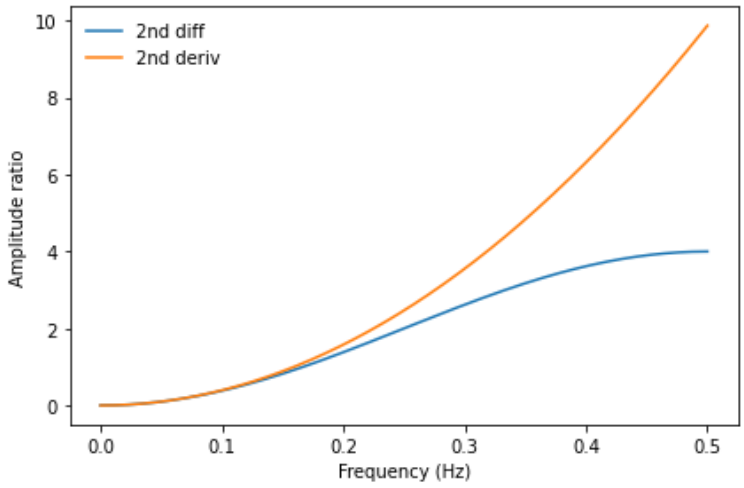

Рис. 19: Сравнение фильтров

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

5 Вывод

В результате выполнения лабораторной работы получены навыки работы с

1. `cumsum` - совокупной суммой
2. `integrate` - интегрированием
3. `diff` - разницей
4. `differentiate` - производными

Также изучены их влиялия на разные типы сигналов.