Лабораторная работа №9
Дифференцирование и интегрирование

Кобыжев Александр

11 апреля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Упражнение 9.1

В данном упражнении нас просят открыть `chap09.ipynb`, прочитать пояснения, а также запустить примеры.

Глава 2

Упражнение 9.2

Создадим волну `TriangleSignal`:

```
in_wave = thinkdsp.TriangleSignal(freq=50).make_wave(duration=0.1,
    framerate=44000)
in_wave.plot()
thinkplot.config(xlabel='Time (s)')
```

Листинг 2.1: Создание треугольного сигнала

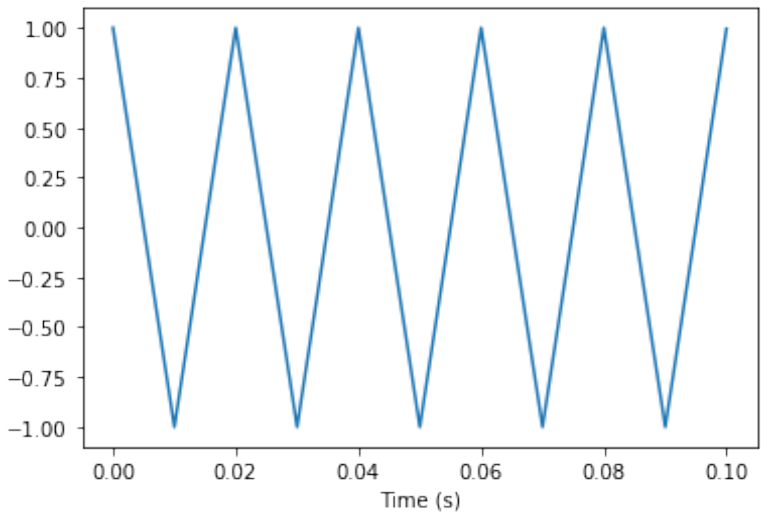

Рис. 2.1: Визуализация треугольного сигнала

`diff` треугольной волны - это прямоугольная волна, что объясняет, почему гармоники в прямоугольной волне уменьшаются как $1/f$, по сравнению с треугольной волной, которая спадает как $1/f2$.

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

Листинг 2.2: Визуализация при `diff`

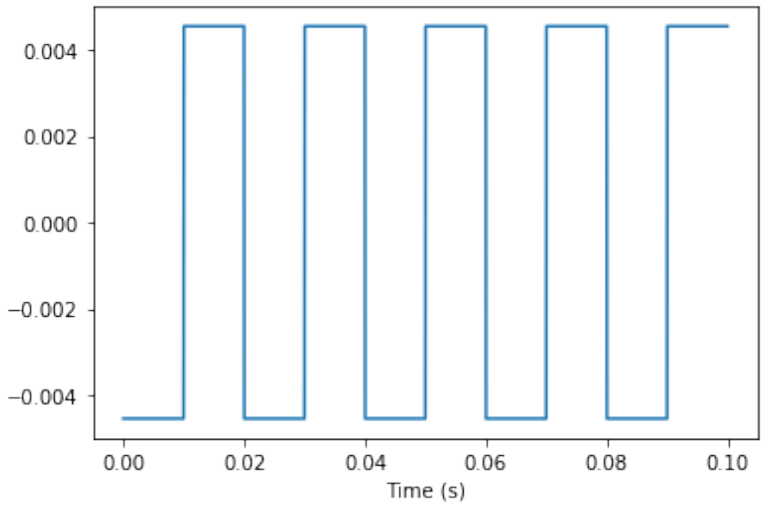

Рис. 2.2: Визуализация при `diff`

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

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

Листинг 2.3: Визуализация при `differentiate`

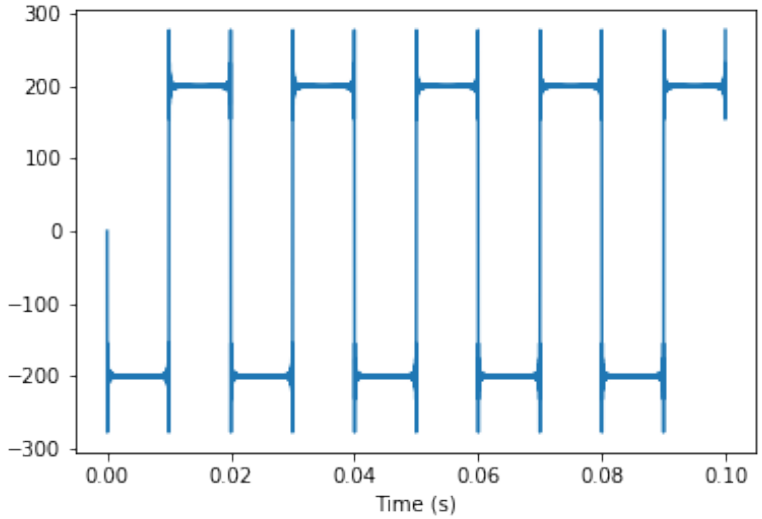

Рис. 2.3: Визуализация при `differentiate`

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

Глава 3

Упражнение 9.3

Сделаем волну `SquareSignal`:

```
in_wave = thinkdsp.SquareSignal(freq=50).make_wave(duration=0.1,
    framerate=44000)
in_wave.plot()
thinkplot.config(xlabel='Time (s)')
```

Листинг 3.1: Создание сигнала

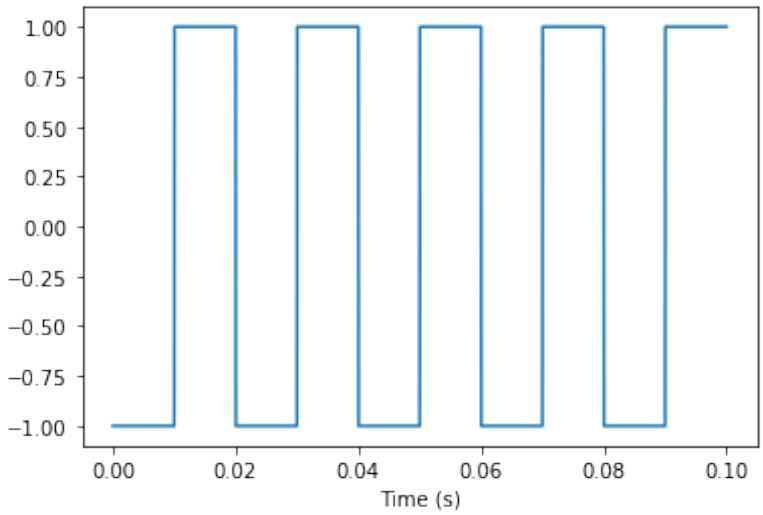

Рис. 3.1: Визуализация сигнала

Совокупная сумма прямоугольной волны - это треугольная волна. После предыдущего выполненного упражнения это не должно вызывать никакого удивления.

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

Листинг 3.2: Визуализация при `cumsum`

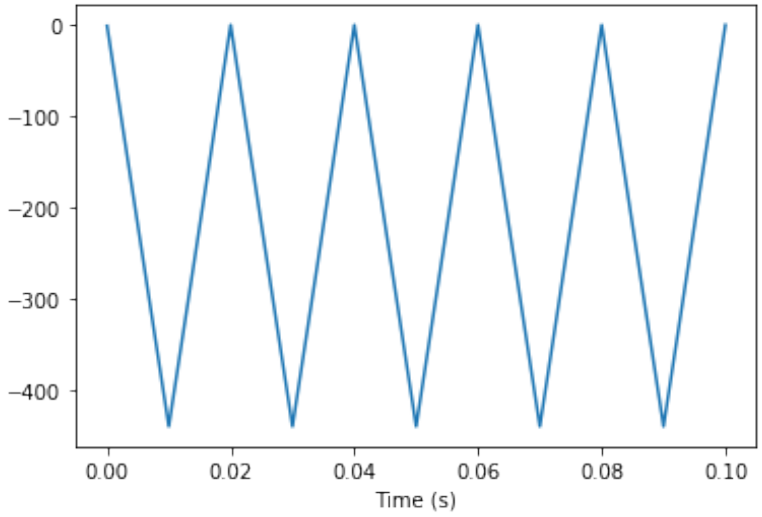

Рис. 3.2: Визуализация при `cumsum`

Спектральный интеграл также представляет собой треугольную волну, хотя амплитуда сильно отличается.

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

Листинг 3.3: Визуализация при `integrate`

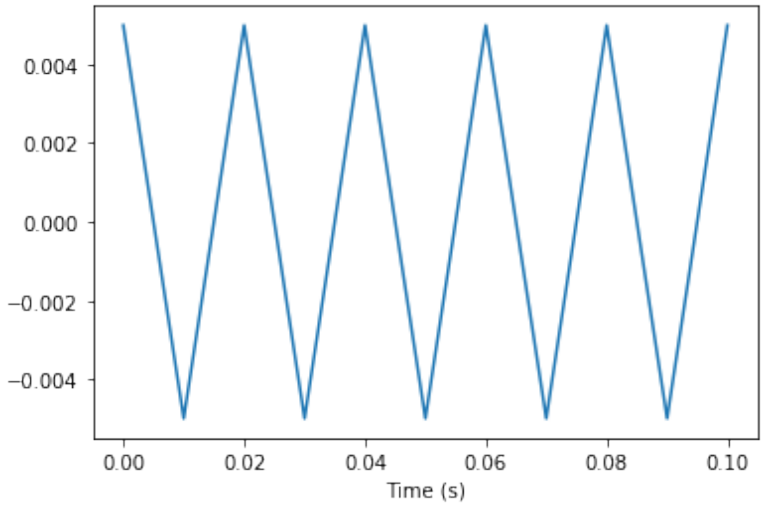

Рис. 3.3: Визуализация при `integrate`

Если уравновесить и нормализовать две волны, они будут визуально похожи.

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

Листинг 3.4: Сравнение волн

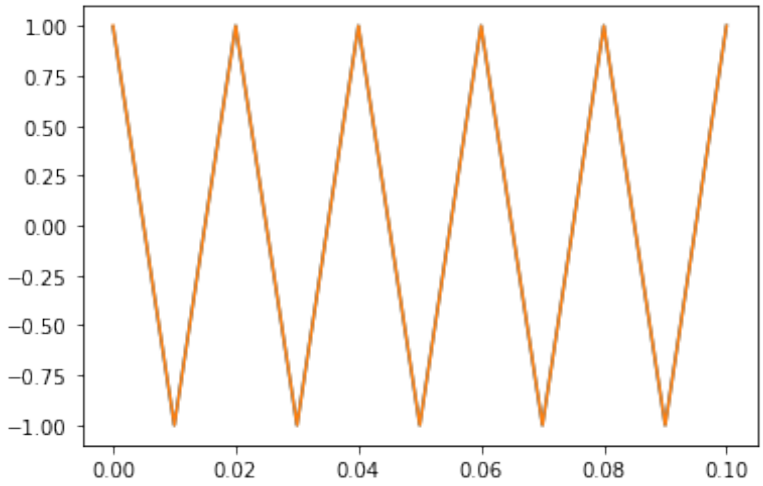

Рис. 3.4: Сравнение волн

```
max(abs(out_wave.ys - out_wave2.ys))
```

Листинг 3.5: Разница между реализациями

Разница составила `0.004545454545454519`. Они численно похожи, но с точностью около 3 цифр.

Глава 4

Упражнение 9.4

Создадим `SawtoothSignal` волну.

```
in_wave = thinkdsp.SawtoothSignal(freq=50).make_wave(duration=0.1,
    framerate=44000)
in_wave.plot()
thinkplot.config(xlabel='Time (s)')
```

Листинг 4.1: Создание сигнала

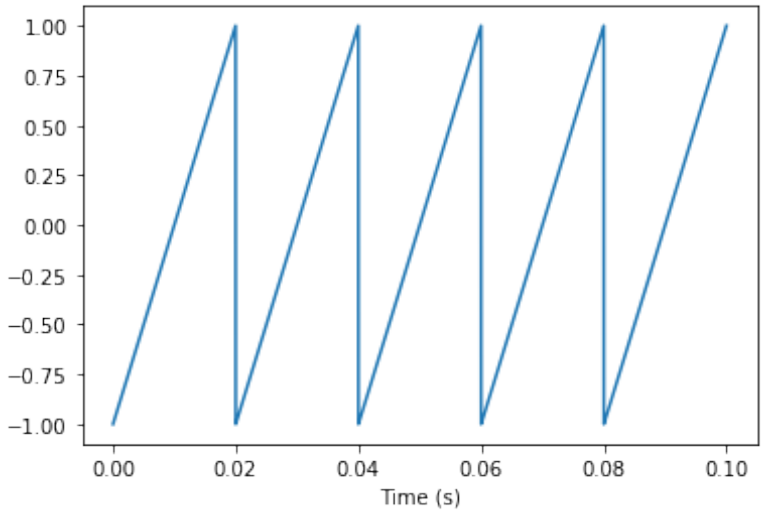

Рис. 4.1: Создание сигнала

Первая совокупная сумма зубца пилы - это парабола:

```
out_wave = in_wave.cumsum()
out_wave.unbias()
```

```
out_wave.plot()
thinkplot.config(xlabel='Time (s)')
```

Листинг 4.2: Первая совокупная сумма

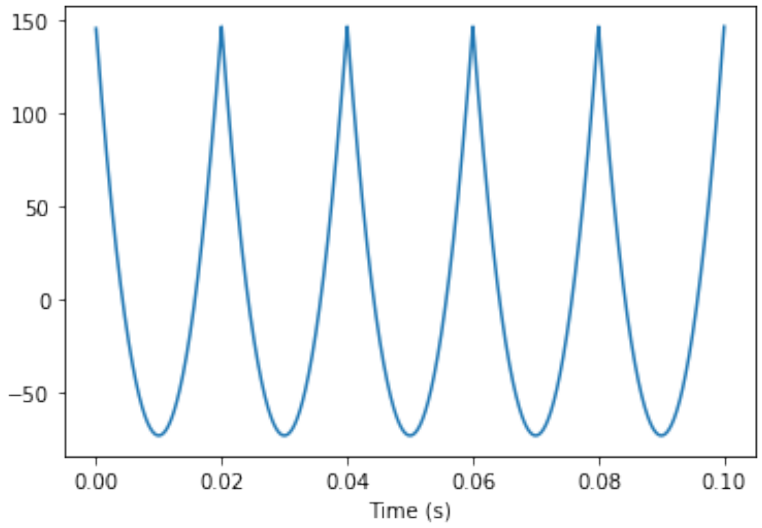

Рис. 4.2: Первая совокупная сумма

Вторая совокупная сумма - это кубическая кривая:

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

Листинг 4.3: Вторая совокупная сумма

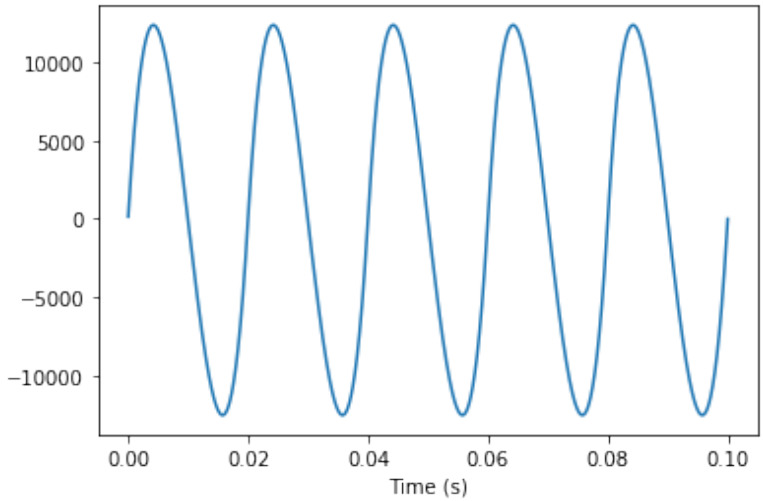

Рис. 4.3: Вторая совокупная сумма

Двойное интегрирование также дает кубическую кривую.

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

Листинг 4.4: Двойное интегрирование

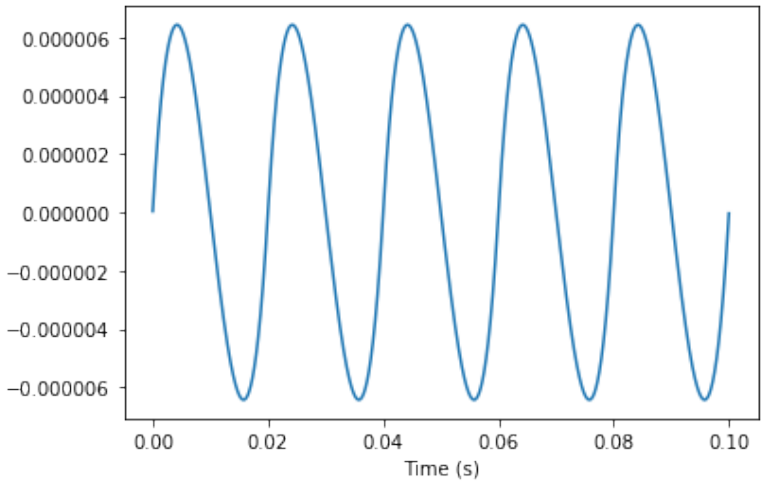

Рис. 4.4: Двойное интегрирование

На этом этапе результат всё больше и больше напоминает синусоиду. Причина в том, что интеграция действует как фильтр нижних частот. На данный момент мы отфильтровали почти все, кроме основного, как показано в спектре ниже:

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

Листинг 4.5: Спектр сигнала

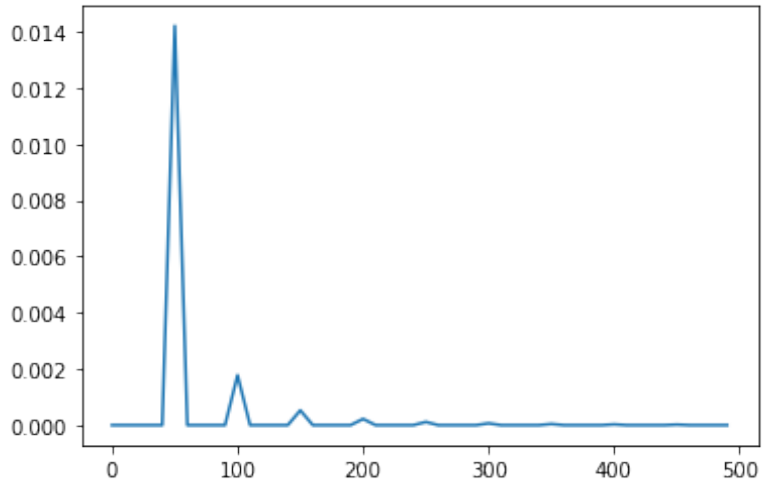

Рис. 4.5: Спектр сигнала

Глава 5

Упражнение 9.5

Создадим волну `CubicSignal`:

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

Листинг 5.1: Создание сигнала

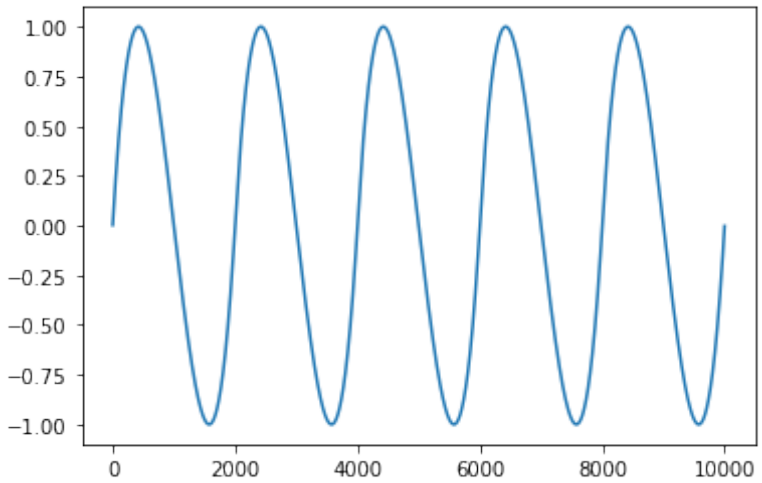

Рис. 5.1: Создание сигнала

Первый `diff` - парабола, второй - пилообразная волна:

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

```
out_wave.plot()
```

Листинг 5.2: Первый `diff`

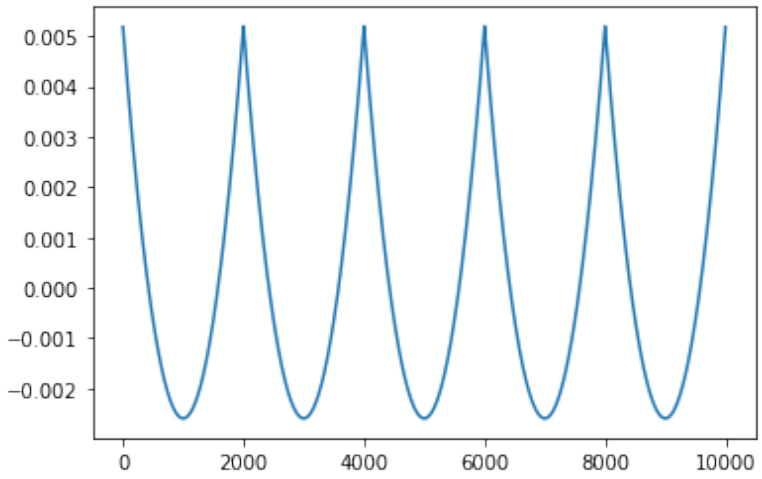

Рис. 5.2: Первый `diff`

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

Листинг 5.3: Второй `diff`

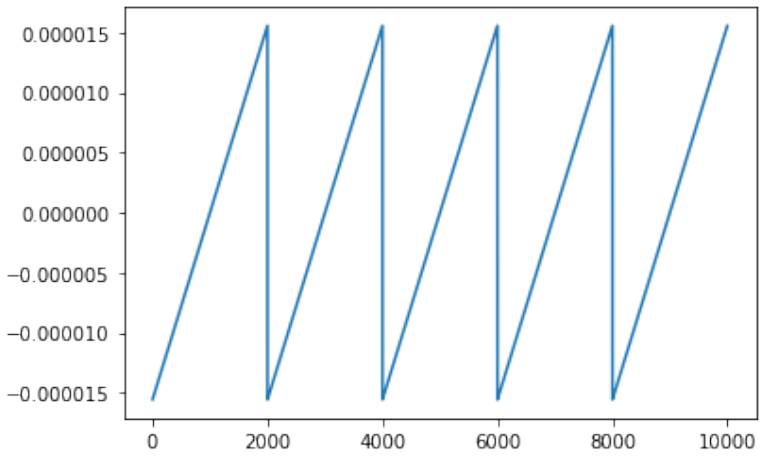

Рис. 5.3: Второй `diff`

Когда мы дифференцируем дважды, получаем пилообразную форму

с некоторым звоном. Проблема в том, что производная параболического сигнала в точках не определена.

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

Листинг 5.4: Вторая производная

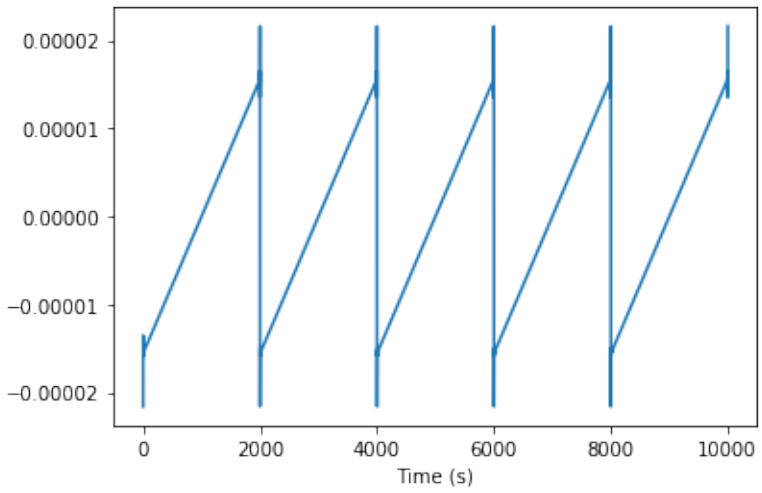

Рис. 5.4: Вторая производная

Окно второй разности -1, 2, -1. Вычисляя ДПФ окна, мы можем найти соответствующий фильтр.

```
diff_window = np.array([-1.0, 2.0, -1.0])
padded = thinkdsp.zero_pad(diff_window, len(in_wave))
diff_wave = thinkdsp.Wave(padded, framerate=in_wave.framerate)
diff_filter = diff_wave.make_spectrum()
diff_filter.plot(label='2nd diff')

thinkplot.config(xlabel='Frequency (Hz)',
                 ylabel='Amplitude ratio',
                 loc='lower right')
```

Листинг 5.5: ДПФ окна

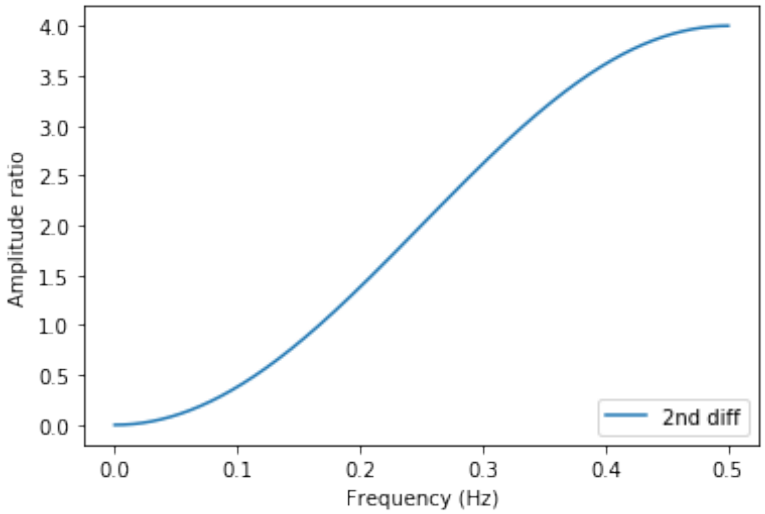

Рис. 5.5: ДПФ окна

А для второй производной мы можем найти соответствующий фильтр, вычислив фильтр первой производной и возведя его в квадрат.

```
deriv_filter = in_wave.make_spectrum()
deriv_filter.hs = (PI2 * 1j * deriv_filter.fs)**2
deriv_filter.plot(label='2nd deriv')

thinkplot.config(xlabel='Frequency (Hz)',
                 ylabel='Amplitude ratio',
                 loc='lower right')
```

Листинг 5.6: Фильтр второй производной

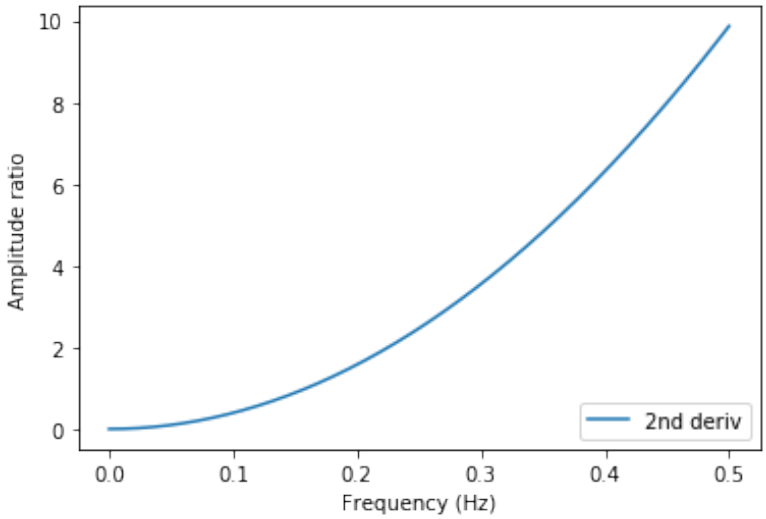

Рис. 5.6: Фильтр второй производной

Рассмотрим два фильтра в одном масштабе:

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

thinkplot.config(xlabel='Frequency (Hz)',
                 ylabel='Amplitude ratio',
                 loc='lower right')
```

Листинг 5.7: Визуализация двух фильтров

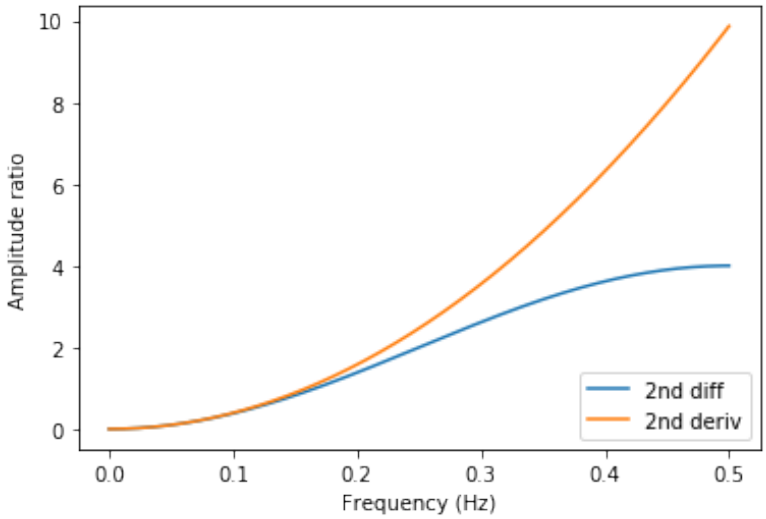

Рис. 5.7: Визуализация двух фильтров

Теперь мы можем видеть, что оба являются фильтрами верхних частот, которые усиливают компоненты самых высоких частот. Второй `deriv` параболический, поэтому он сильнее всего усиливает самые высокие частоты. Второй `diff` - хорошее приближение второй производной только на самых низких частотах, затем он существенно отклоняется.

Глава 6

Выводы

Во время выполнения лабораторной работы получены навыки работы с взаимосвязью между окнами во временной области и фильтрами в частотной области. Также изучалось влияние окна конечных разностей, которое приближает дифференцирование, и операции накопления суммы, которая приближает интегрирование.