# Лабораторная работа №9
# Дифференцирование и интегрирование

Крынский Павел

27 мая 2021 г.

# Оглавление

# Список иллюстраций

# Листинги

# Глава 1

# Упражнение 9.1

В данном упражнении нам нужно открыть `chap09.ipynb`, прочитать пояснения и запустить примеры. Поэтому я просто изучил все примеры с комментариями.

# Глава 2

# Упражнение 9.2

Создаю треугольный сигнал `TriangleSignal`:

```
iwave = thinkdsp.TriangleSignal(freq=40).make_wave(duration=0.1,
    framerate=44100)
iwave.plot()
```

Листинг 2.1: Создание треугольного сигнала

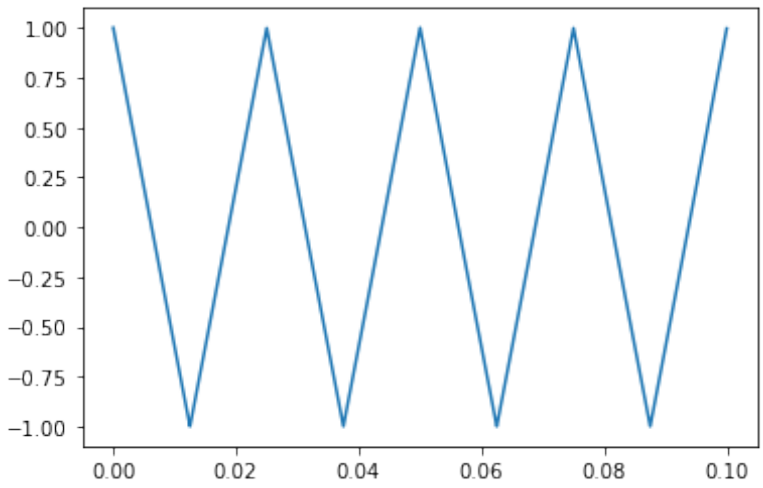


Рис. 2.1: Треугольный сигнал

Применяю `diff`. `diff` от треугольной функции - прямоугольная функция. Можно сказать, что гармнрнтки прямоугльной и треугольной гармоники совпадают по $1/f$ и $1/f2$.

```
owave = iwave.diff()
```

```
owave.plot()
```

Листинг 2.2: Визуализация при `diff`

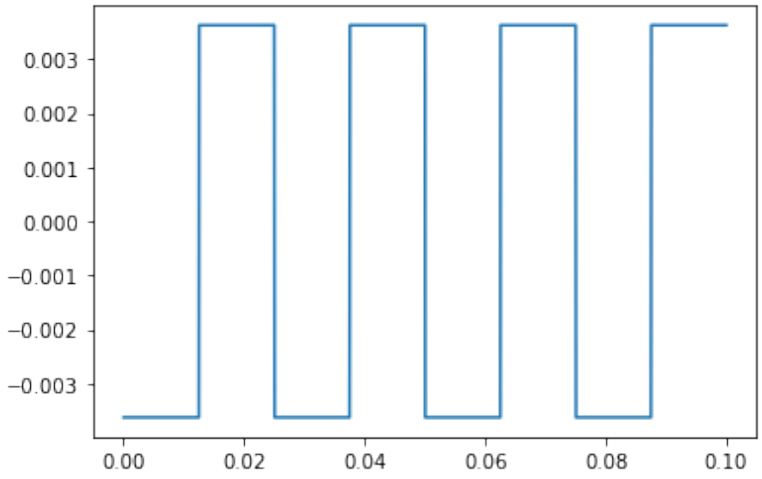


Рис. 2.2: Визуализация при использовании`diff`

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

```
owave2 = iwave.make_spectrum().differentiate().make_wave()
owave2.plot()
```

Листинг 2.3: Визуализация при `differentiate`

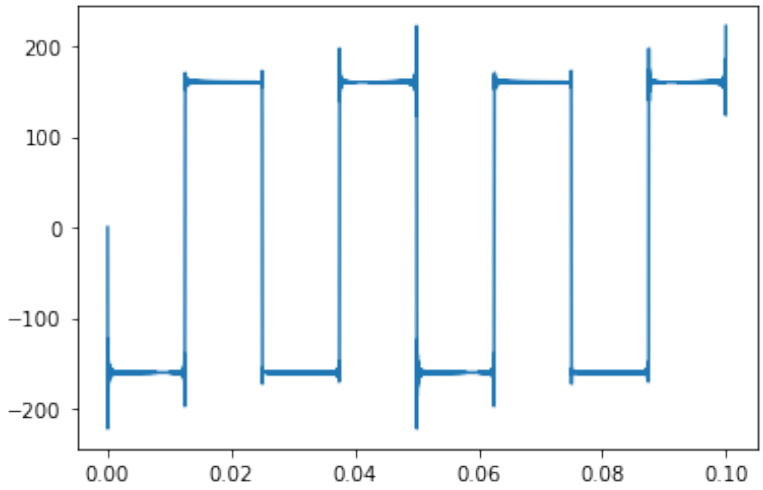


Рис. 2.3: Визуализация при использованииdifferentiate

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

# Глава 3

# Упражнение 9.3

Для начала я создал прямоугольный сигнал.

```
iwave = thinkdsp.SquareSignal(freq=40).make_wave(duration=0.1,
    framerate=44100)
iwave.plot()
```

Листинг 3.1: Создание сигнала

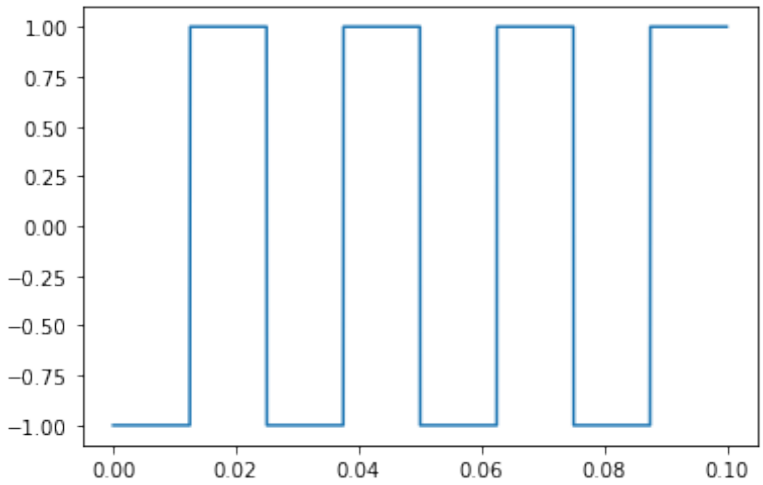


Рис. 3.1: Прямоугольный сигнал

Мы получим треугольный сигнал из прямоугольного, что довольно логично после выполнения предыдущего упражнения.

```
owave = iwave.cumsum()
```

```
owave.plot()
```

Листинг 3.2: Визуализация при `cumsum`

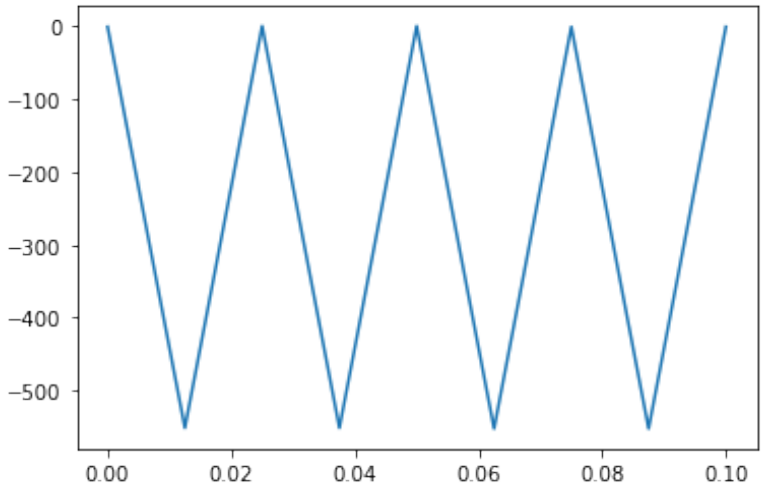


Рис. 3.2: Визуализация при использовании`cumsum`

Создаем спектр сигнала и используем фйнкцию `integrate` и получаем новый сигнал на его основе.

```
spectr = iwave.make_spectrum().integrate()
spectr.hs[0] = 0
owave2 = spectr.make_wave()
owave2.plot()
```

Листинг 3.3: Визуализация при `integrate`

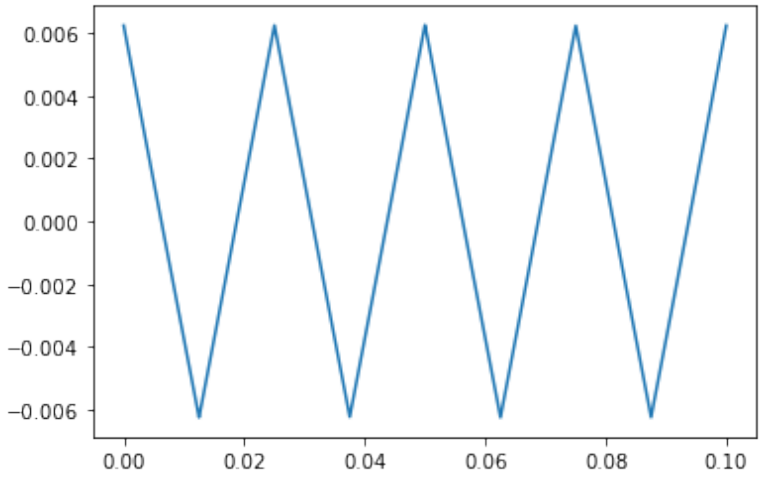


Рис. 3.3: Визуализация при использовании`integrate`

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

```
owave.unbias()
owave.normalize()
owave2.normalize()
owave.plot()
owave2.plot()
```

Листинг 3.4: Сравнение волн

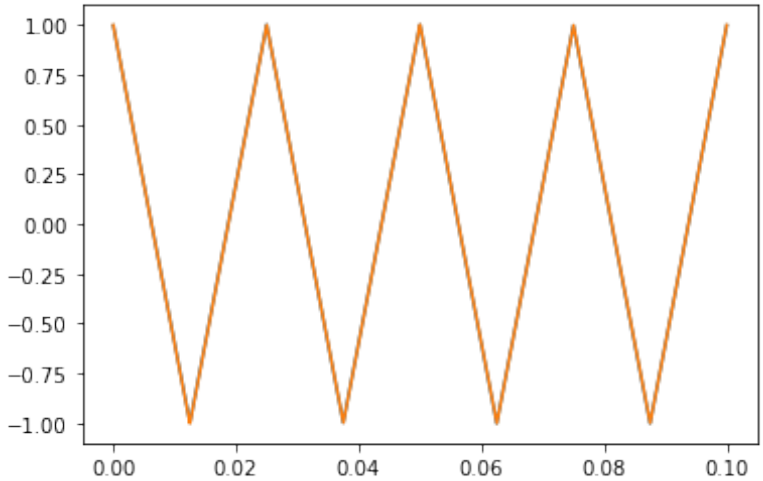


Рис. 3.4: Сравнение волн

```
max(abs(owave.ys - owave2.ys))
```

Листинг 3.5: Разница между реализациями

Считаем разницу между реализациями и получаем `0.00464956318805787`.

# Глава 4

# Упражнение 9.4

Создаю пилообразный сигнал.

```
iwave = thinkdsp.SawtoothSignal(freq=40).make_wave(duration=0.1,
    framerate=44100)
iwave.plot()
```

Листинг 4.1: Создание сигнала

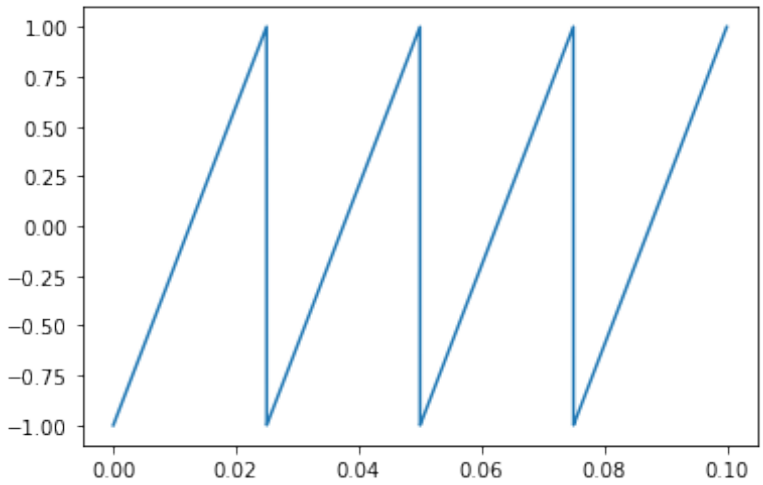


Рис. 4.1: Создание сигнала

Вычисляем спектр и применяем к нему функцию `integrate` два раза.

```
owave = iwave.cumsum()
owave.unbias()
owave.plot()
owave = owave.cumsum()
```

```
5 owave.plot()
```

Листинг 4.2: двойное применение `integrate`

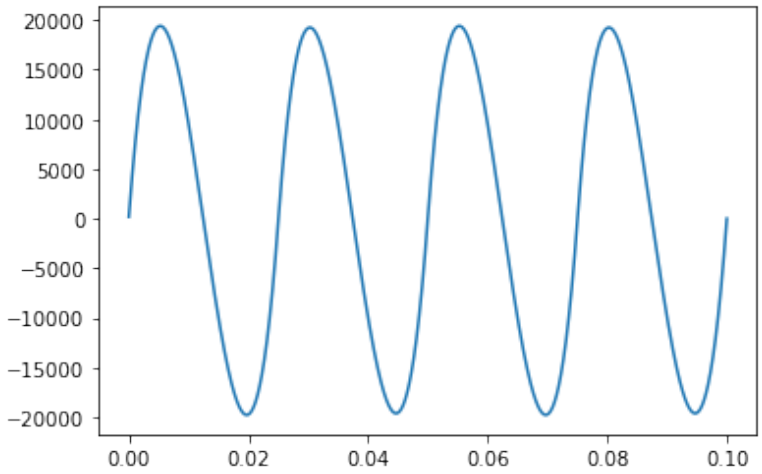


Рис. 4.2: Двойное применение `integrate`

Двойное интегрирование дает кубическую кривую. На этом этапе результат всё больше и больше напоминает синусоиду. Причина в том, что интеграция действует как фильтр нижних частот.

# Глава 5

# Упражнение 9.5

Создадим кубически сигнал.

```
iwave = thinkdsp.CubicSignal(freq=0.0004).make_wave(duration=10000,
    framerate=1)
iwave.plot()
```

Листинг 5.1: Создание сигнала

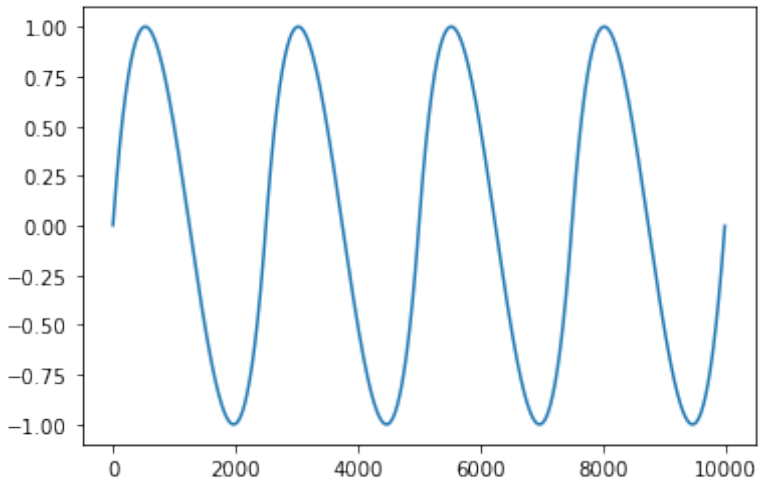


Рис. 5.1: Создание сигнала

Дважды применим функцию `diff` и получи пилообразный сигнал, что довольно логично после выполнения предыдущего упражнения.

```
owave = iwave.diff().diff()
```

```
owave.plot()
```

Листинг 5.2: Применение `diff`

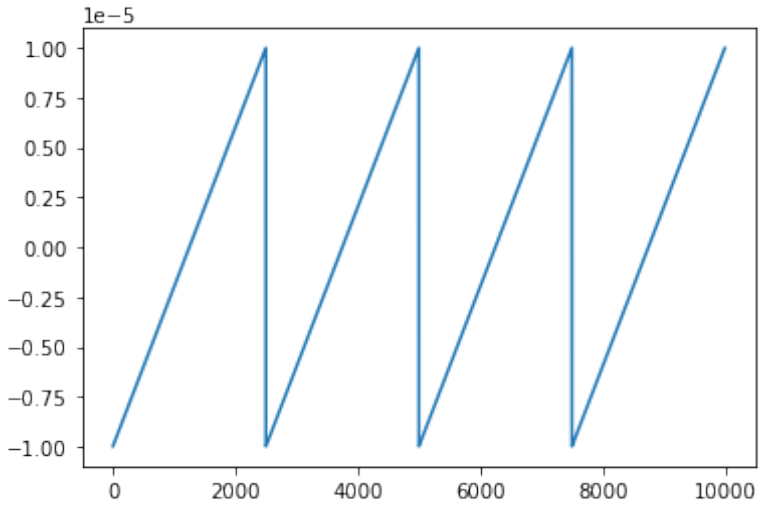


Рис. 5.2: Применение `diff`

Когда мы дифференцируем дважды, получаем пилообразную форму с некоторым звоном. Проблема в том, что производная параболического сигнала в точках не определена.

```
spectr = iwave.make_spectrum().differentiate().differentiate()
owave2 = spectr.make_wave()
owave2.plot()
```

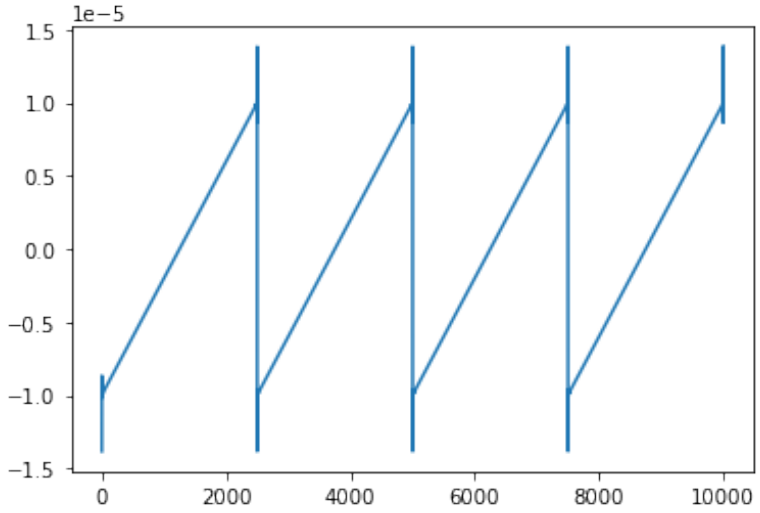


Рис. 5.3:

В конце работы я вывел фильтры для второй разницы и второй производной фильтры и сравнил их:

Рассмотрим два фильтра в одном масштабе:

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

Листинг 5.3: Визуализация двух фильтров

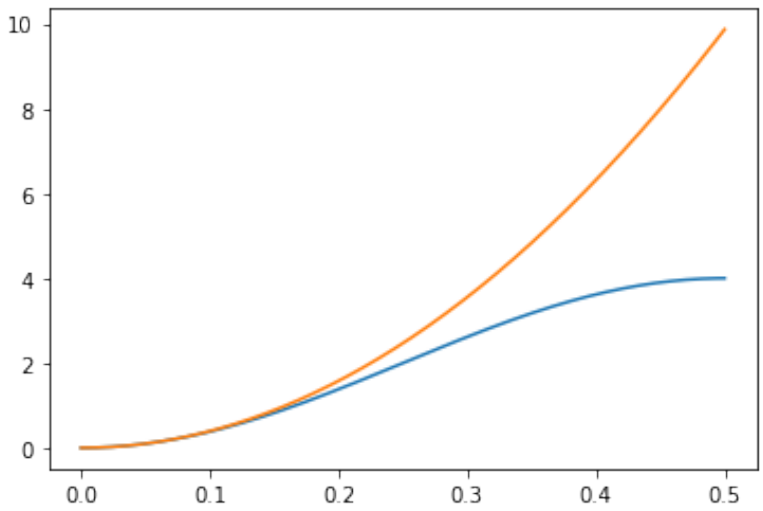


Рис. 5.4: Визуализация двух фильтров(синий - diff, оранжевый - deriv)

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.