Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Отчёт по лабораторной работе №9
Дисциплина: Телекоммуникационные технологии
Тема: Дифференцирование и интегрирование

Выполнил студент гр. 3530901/80201 В.А. Пучкина

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

Листинги

1 Упражнение 9.1

В этом упражнении необходимо изучить примеры в файле `chap09.ipynb`. После этого заменим периодический пилообразный сигнал на непериодические данные Facebook, чтобы проверить, какие именно примеры перестанут работать корректно.

Итак, все примеры были изучены. Перейдём к эксперименту. Считаем файл с данными Facebook и получим `wave`.

```
df = pandas.read_csv('resources/task1_FB_2.csv', header=0, parse_dates=[0])
ys = df['Close']
in_wave = Wave(ys, framerate=1)
in_wave.plot()
decorate(xlabel='Time (days)', ylabel='Price')
```

Листинг 1: Чтение файла и получение `wave`.

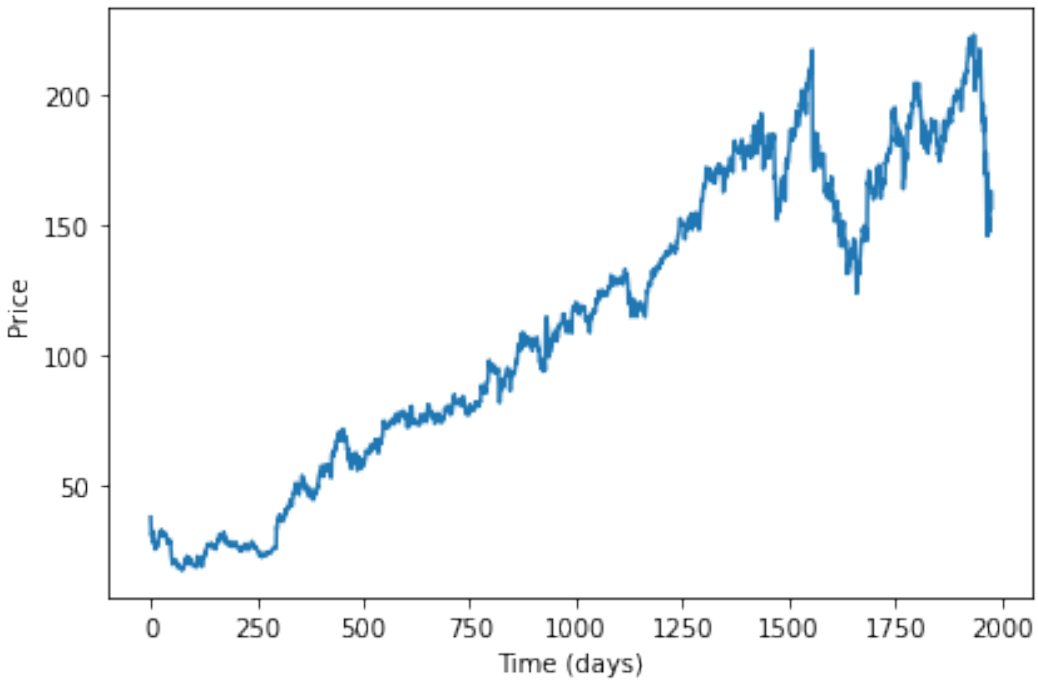

Рис. 1: Сигнал Facebook.

Теперь получим спектр нашего сигнала.

```
in_spectrum = in_wave.make_spectrum()
in_spectrum.plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 2: Получение спектра сигнала Facebook.

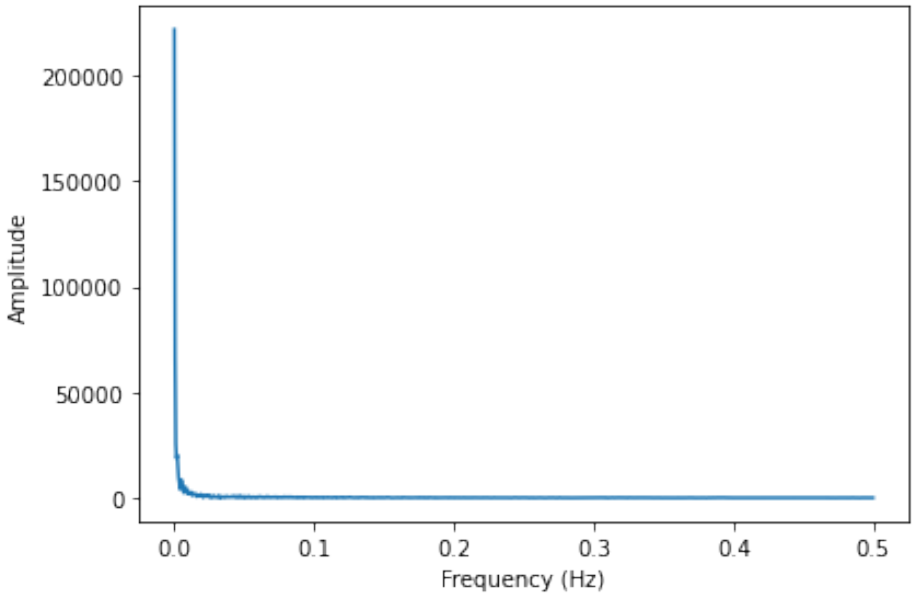

Рис. 2: Спектр сигнала Facebook.

Теперь рассчитаем выходной сигнал как совокупную сумму входных сигналов.

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

Листинг 3: Рассчёт выходного сигнала.

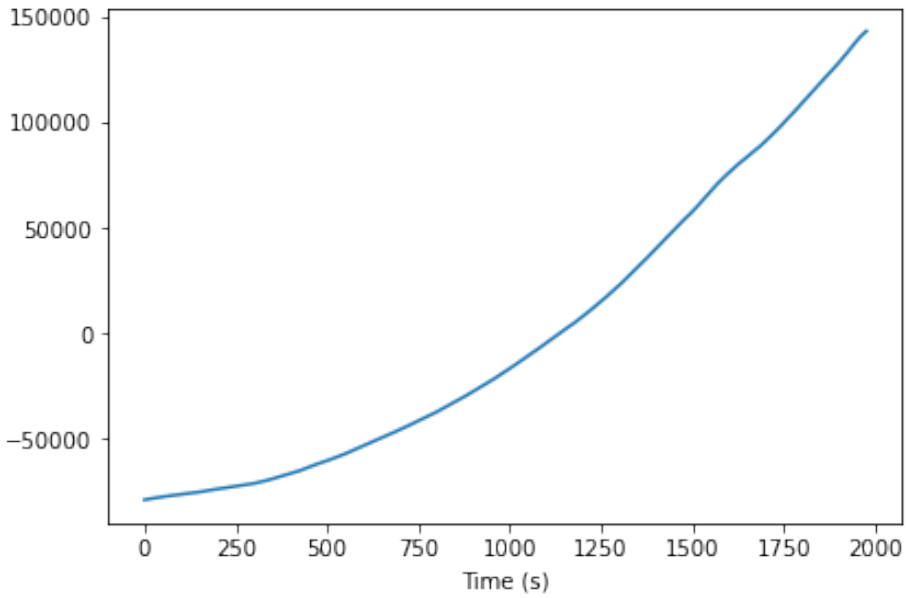

Рис. 3: Выходной сигнал.

И получим спектр выходного `wave`.

```
out_spectrum = out_wave.make_spectrum()
out_spectrum.plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 4: Получение спектра выходного сигнала.

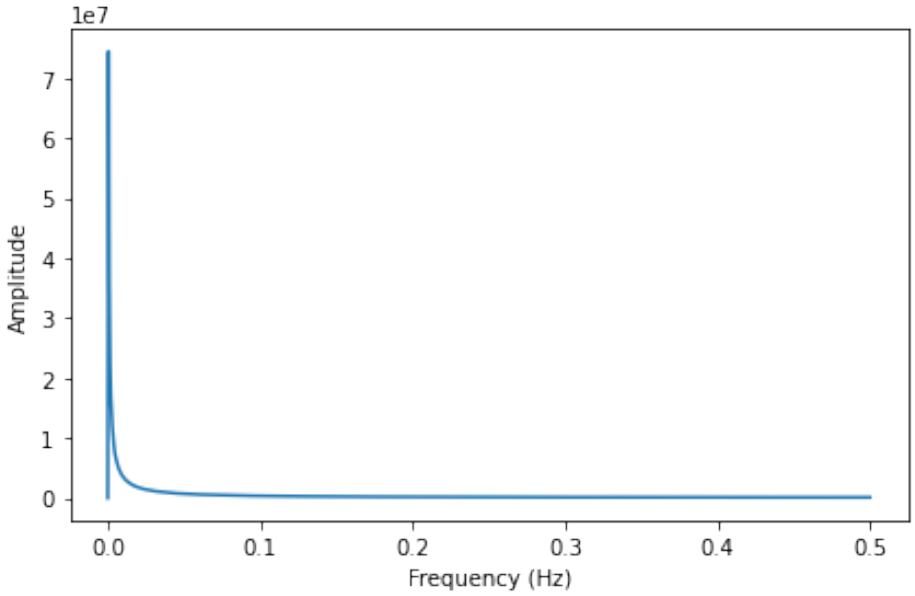

Рис. 4: Спектр выходного сигнала.

Получим отношение между входными и выходными данными.

```
sum(in_spectrum.amps < 1), len(in_spectrum)
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 5: Получение отношения между входными и выходными данными.

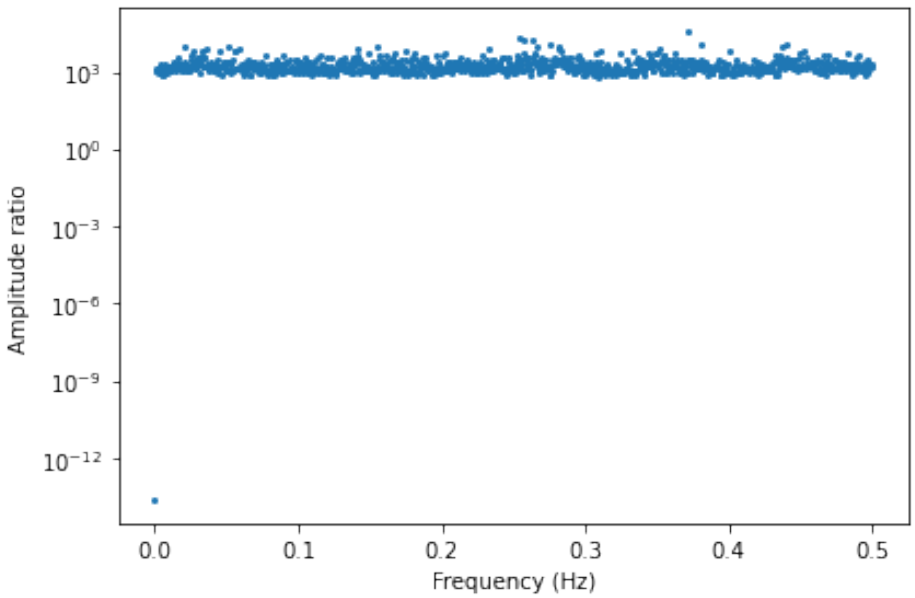

Рис. 5: Отношение между входными и выходными данными.

Теперь получим фильтр cumsum и интеграционный фильтр и сравним их.

```
diff_window = numpy.array([1.0, -1.0])
padded = zero_pad(diff_window, len(in_wave))
diff_wave = Wave(padded, framerate=in_wave.framerate)
diff_filter = diff_wave.make_spectrum()

cumsum_filter = diff_filter.copy()
cumsum_filter.hs[1:] = 1 / cumsum_filter.hs[1:]
cumsum_filter.hs[0] = numpy.inf

PI2 = numpy.pi * 2
integ_filter = cumsum_filter.copy()
integ_filter.hs[1:] = integ_filter.framerate / (PI2 * 1j * integ_filter.fs[1:])
integ_filter.hs[0] = numpy.inf

cumsum_filter.plot(label='cumsum filter', alpha=0.7)
integ_filter.plot(label='integral filter', alpha=0.7)
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio', yscale='log')
```

Листинг 6: Вычисление фильтра cumsum и интеграционного фильтра.

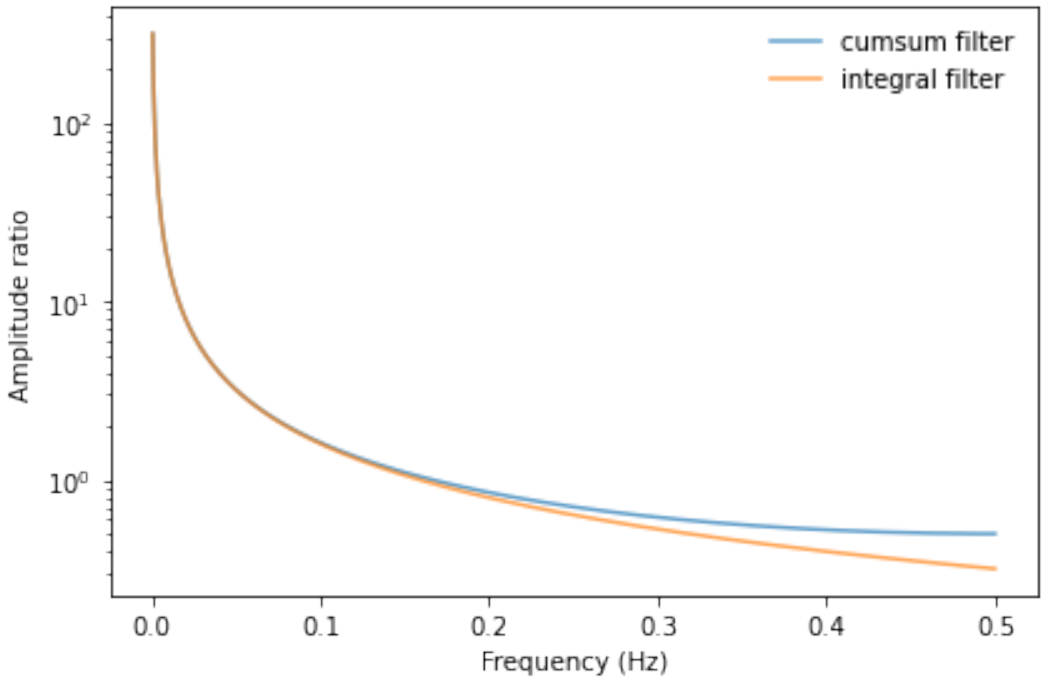

Рис. 6: Сравнение фильтров.

Теперь сравним вычисленные отношения с фильтром.

```
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 7: Сравнение отношений с фильтром.

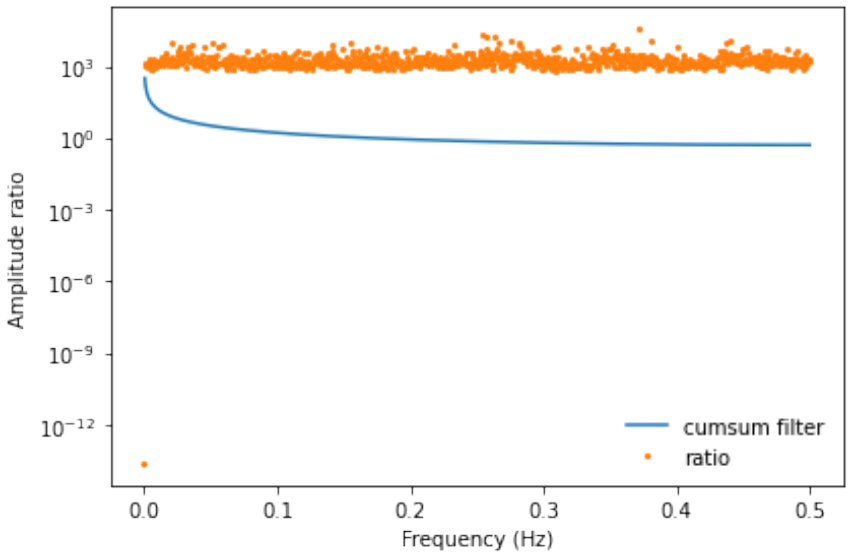

Рис. 7: Сравнение отношений с фильтром `cumsum`.

Тут уже можно заметить различия: графики не совпадают, хотя должны. Их совпадение подвердило бы, что фильтр `cumsum` является обратным фильтру `diff`.

Теперь вычислим выходной сигнал, используя теорему свертки и сравним результаты.

```
len(in_spectrum), len(cumsum_filter)
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 8: Вычисление выходного `wave`.

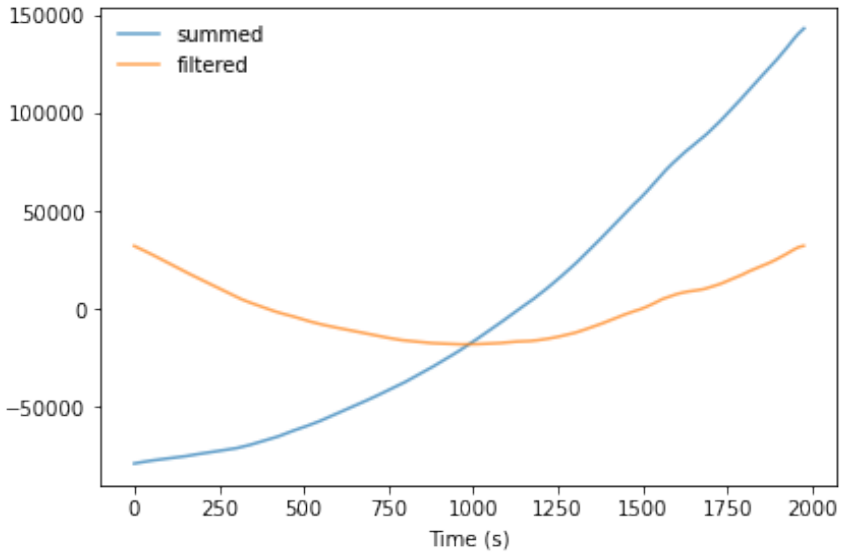

Рис. 8: Сравнение выходных сигналов.

Можно заметить, что выходные сигналы, полученные разными способами, не совпадают. Такое поведение говорит о некорректной работе примеров с апериодическими сигналами.

В данном упражнении были изучены примеры в файле `chap09.ipynb`. После этого был проведён следующий эксперимент: периодический пилообразный сигнал был заменён на непериодические данные Facebook. В ходе эксперимента выяснилось, что некоторые примеры действительно работают некорректно с апериодическими сигналами.

2 Упражнение 9.2

В этом упражнении изучается влияние `diff` и `differentiate` на сигнал.

Итак, начнём с создания треугольного сигнала.

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

Листинг 9: Создание треугольного сигнала.

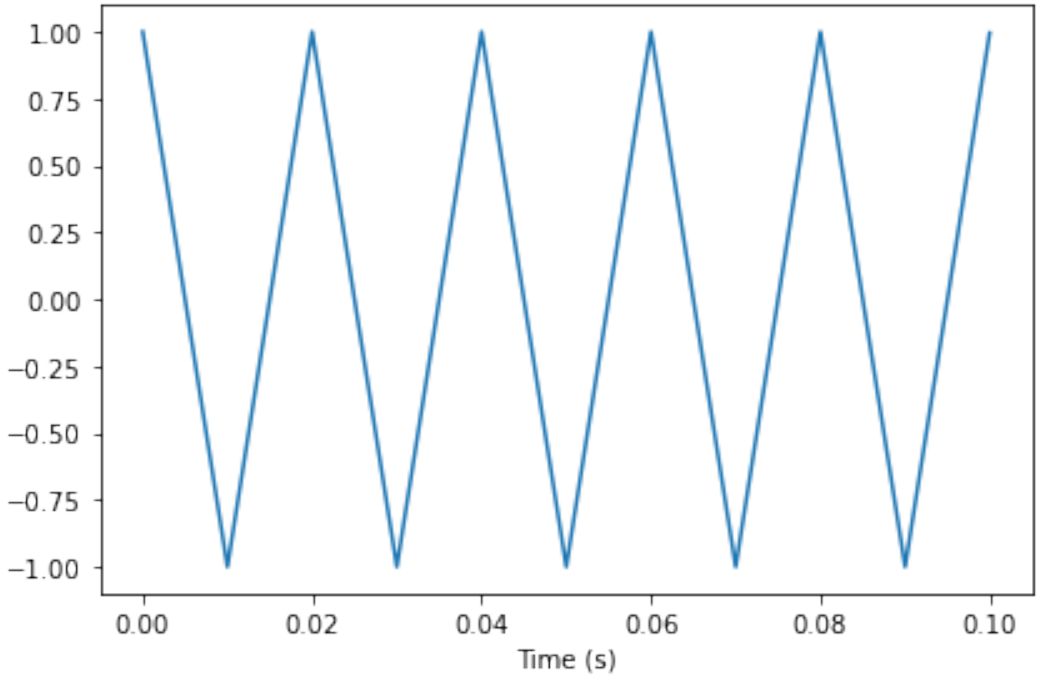

Рис. 9: Треугольный сигнал.

Теперь применим `diff` к нашему сигналу.

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

Листинг 10: Применение `diff`.

Теперь посмотрим, как на сигнал повлияет `differentiate`. Для этого получим спектр треугольного сигнала и применим к нему `differentiate`. После чего преобразуем полученный результат обратно в `wave`.

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

Листинг 11: Применение `differentiate`.

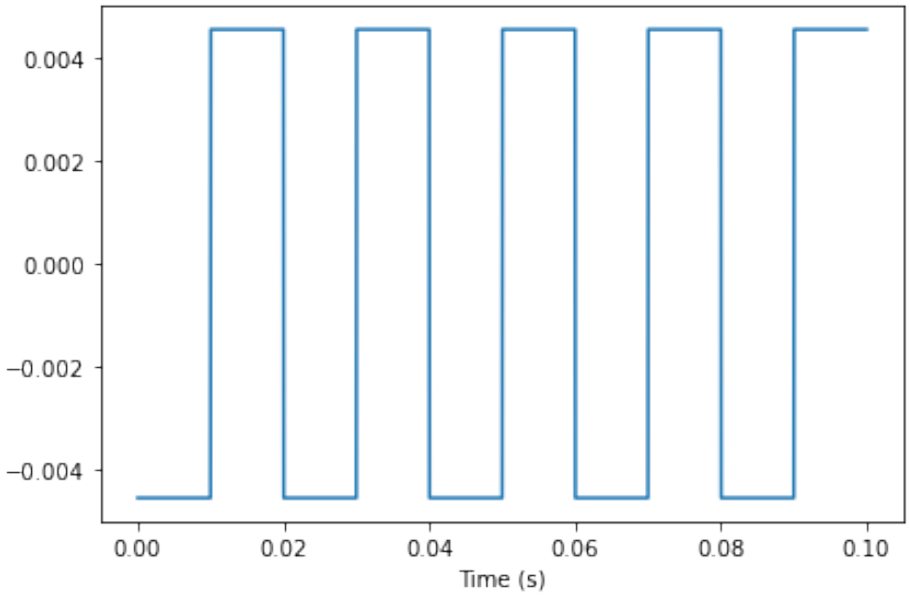

Рис. 10: Результат после применения `diff`.

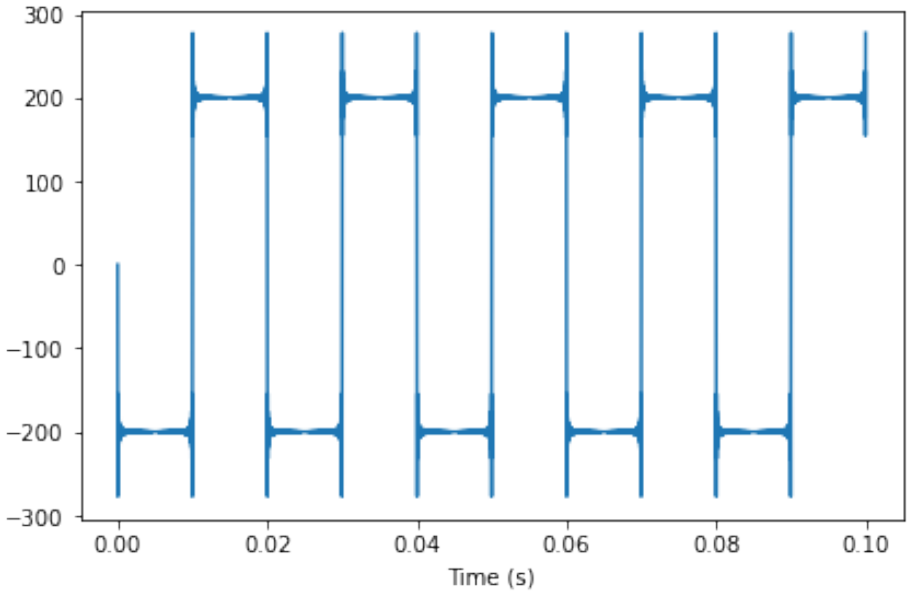

Рис. 11: Результат после применения `differentiate`.

Когда мы берем спектральную производную, мы получаем ”звон”вокруг разрывов. С математической точки зрения проблема в том, что производная треугольной волны не определена в точках треугольника.

В данном упражнении было изучено влияние `diff` и `differentiate` на сигнал. Был сделан вывод, что при взятии спектральной производной треугольного сигнала получается ”звон”вокруг разрывов. Это проблема возникает из-за того, что производная треугольной волны не определена в точках треугольника.

3 Упражнение 9.3

В этом упражнении изучается влияние `cumsum` и `integrate` на сигнал.

Для начала создадим прямоугольный сигнал.

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

Листинг 12: Создание прямоугольного сигнала.

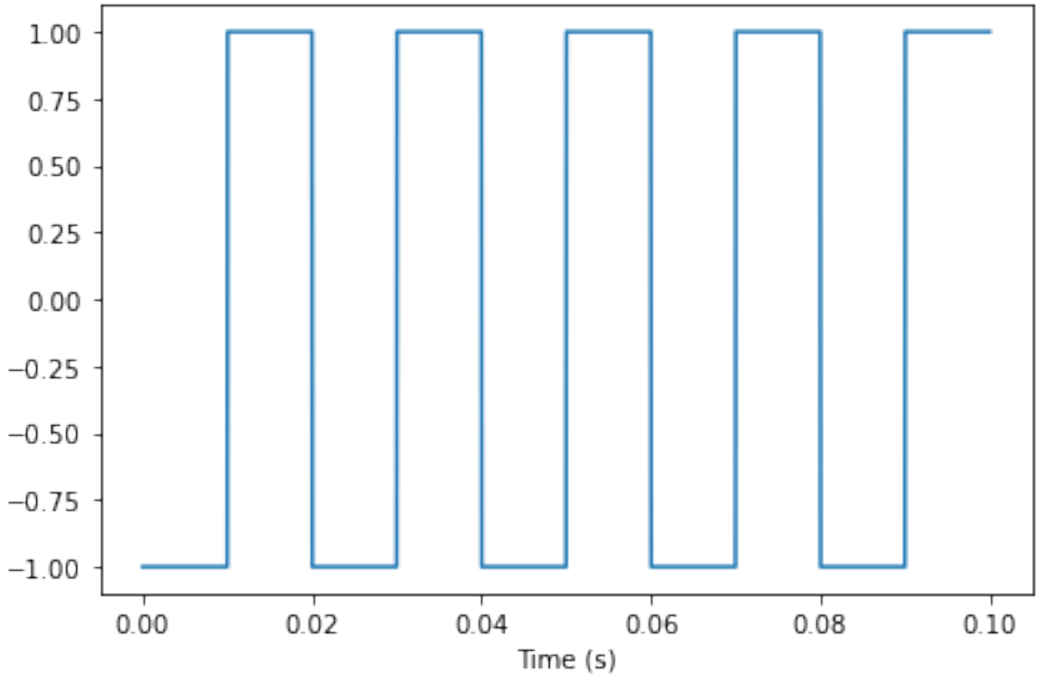

Рис. 12: Прямоугольный сигнал.

Теперь применим к нашему сигналу `cumsum` и посмотрим на результат.

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

Листинг 13: Применение `cumsum`.

Как мы видим из Рис.13, совокупная сумма прямоугольного сигнала - это треугольный сигнал.

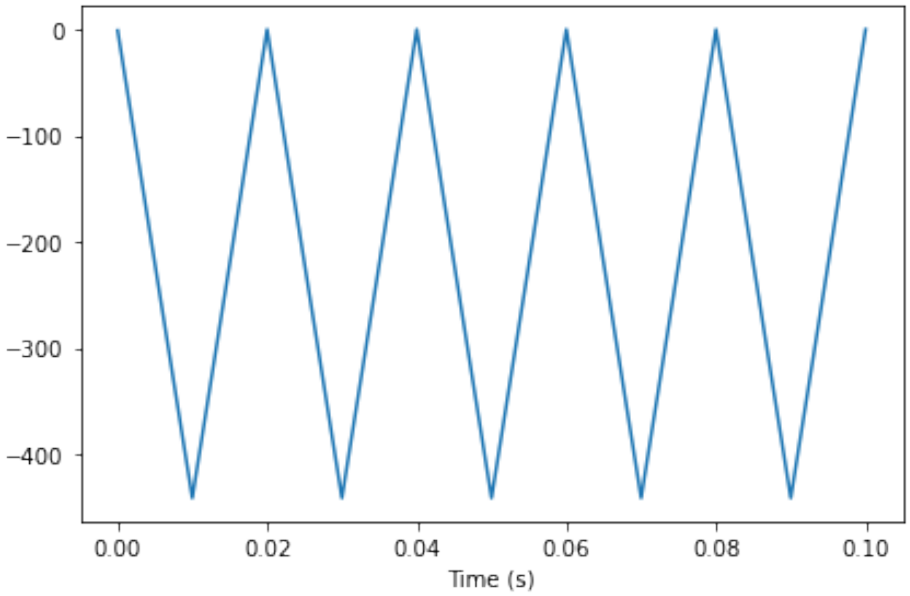

Рис. 13: Результат после применения `cumsum`.

Теперь вычислим спектр нашего прямоугольного сигнала и применим к нему `integrate`.

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

Листинг 14: Применение `integrate`.

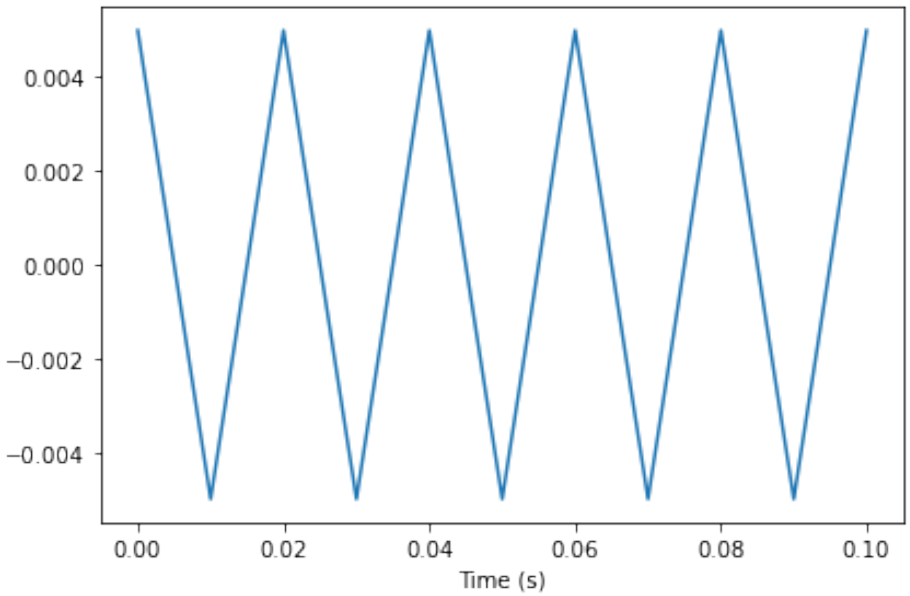

Рис. 14: Результат после применения `integrate`.

Из Рис.14 видно, что спектральный интеграл также является треугольным сигналом. У полученных сигналов, однако, сильно отличается амплитуда.

Попробуем уравновесить и нормализовать оба сигнала.

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

Листинг 15: Работа с сигналами.

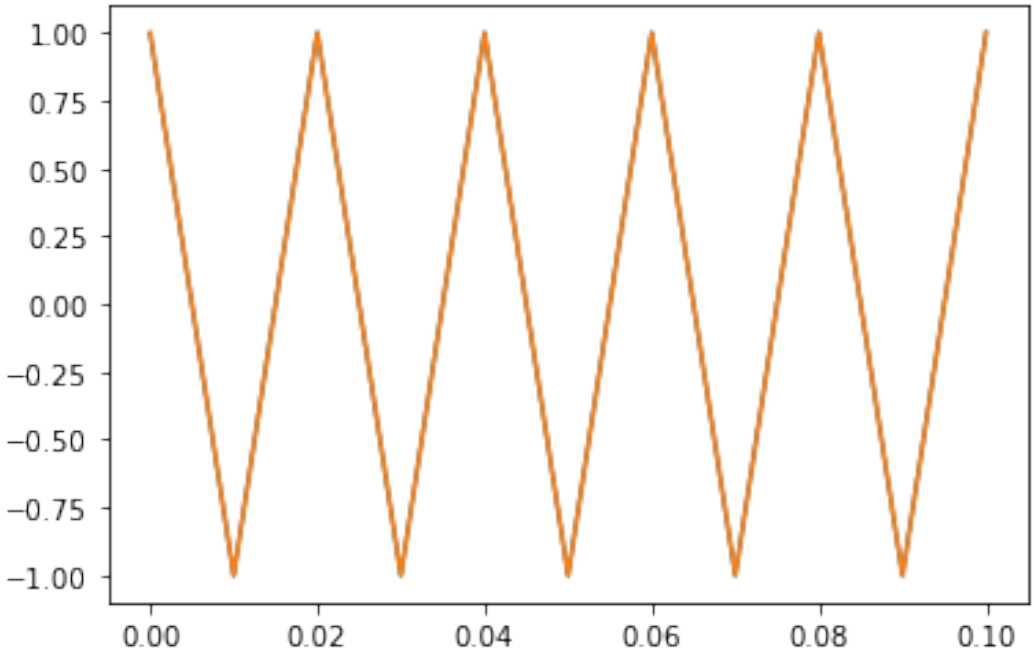

Рис. 15: Сравнение полученных сигналов.

Теперь они визуально похожи. Их же максимальная разница (`0.00453514739229021`) говорит о том, что их численное значение похоже с точностью до 2 знаков после запятой.

В данном упражнении было изучено влияние `cumsum` и `integrate` на сигнал. Для этого был создан прямоугольный сигнал, к которому по очереди были применены эти функции. По полученным результатам был сделан вывод, что совокупная сумма и спектральный интеграл прямоугольного сигнала очень похожи. При их нормализации можно увидеть, что их значения похожи с точностью до 2 знаков после запятой.

4 Упражнение 9.4

В данном упражнении изучается влияние двойного интегрирования.

Начнём с создания пилообразного сигнала.

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

Листинг 16: Создание пилообразного сигнала.

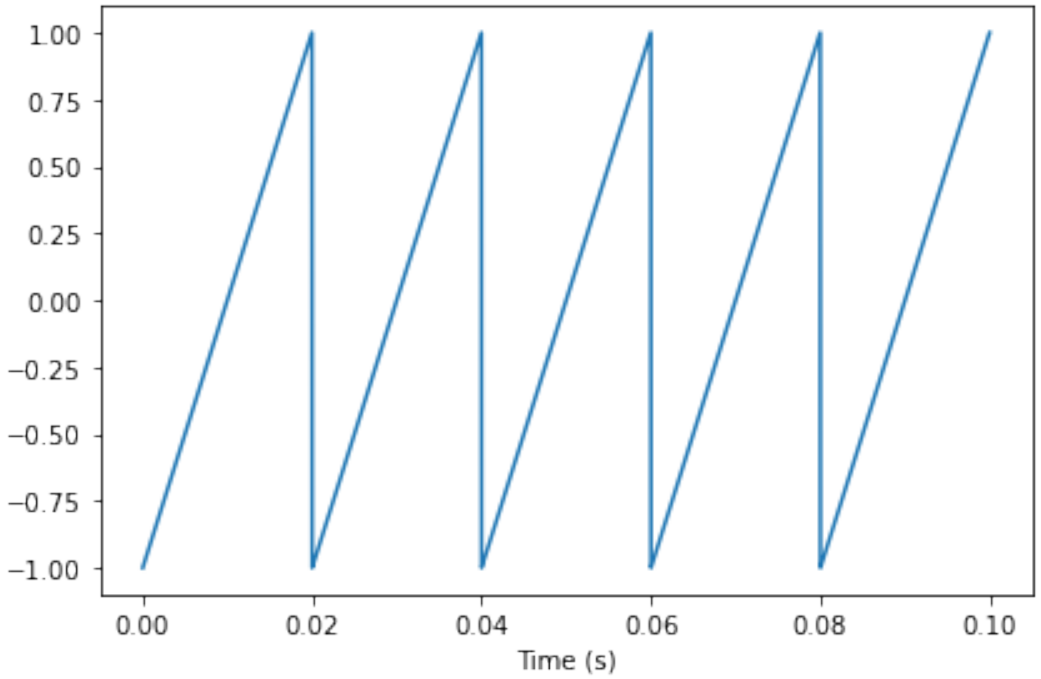

Рис. 16: Пилообразный сигнал.

Теперь применим `integrate` к нашему сигналу.

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

Листинг 17: Применение `integrate`.

Как мы видим из Рис.17, первый спектральный интеграл - это парабола.

Повторно применим `integrate` и посмотрим на результат.

```
spectrum2 = spectrum.integrate()
spectrum2.hs[0] = 0
out_wave2 = spectrum2.make_wave()
out_wave2.plot()
decorate(xlabel='Time (s)')
```

Листинг 18: Второе применение `integrate`.

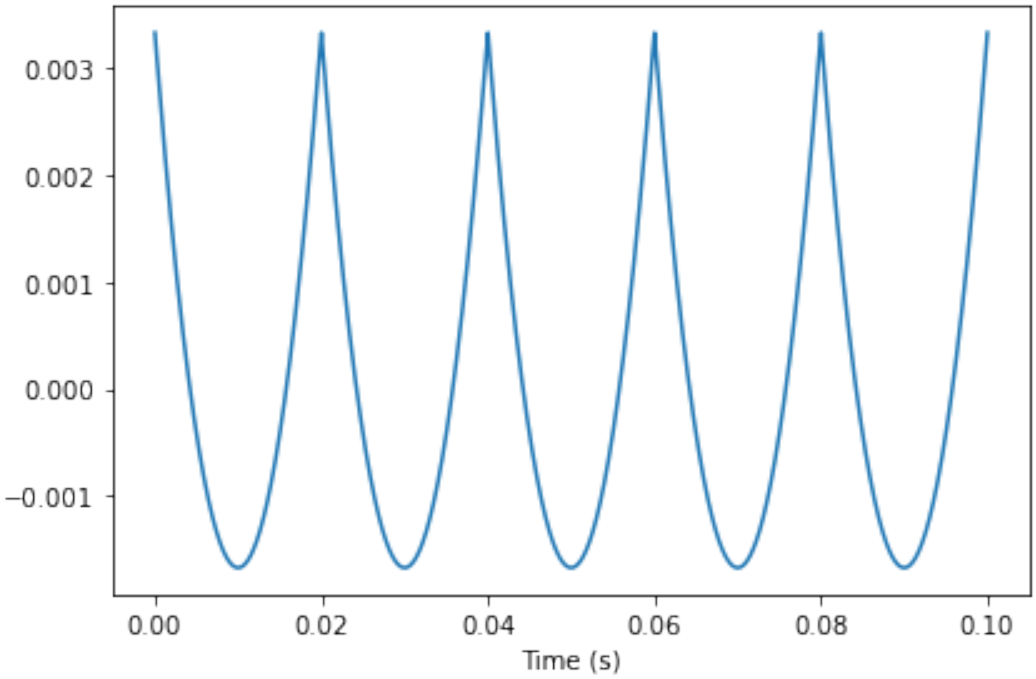

Рис. 17: Сигнал после первого применения `integrate`.

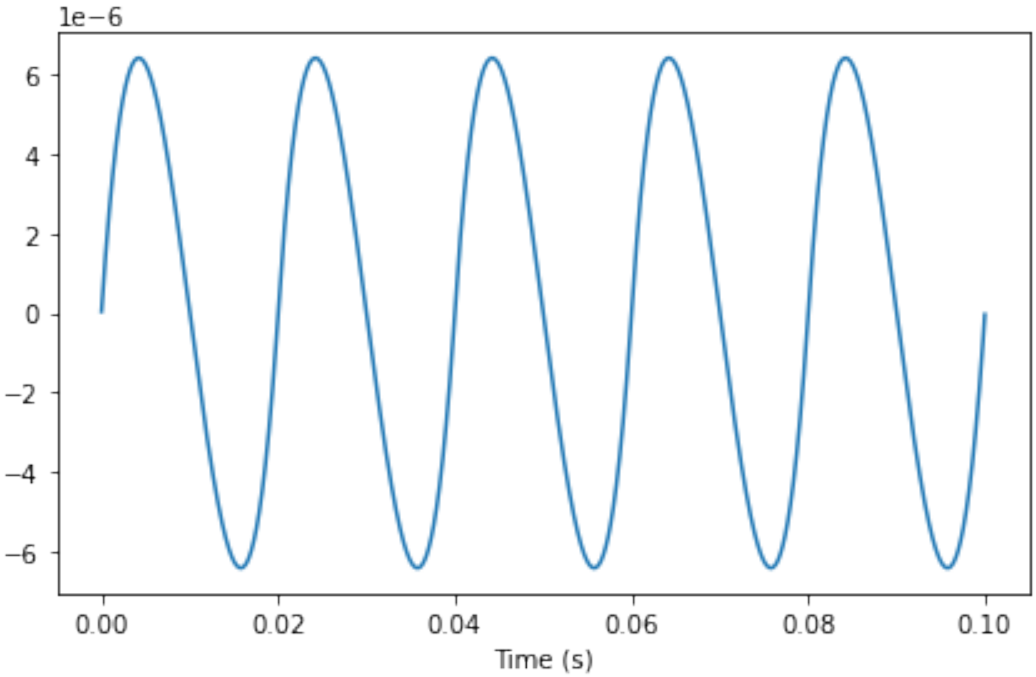

Рис. 18: Сигнал после второго применения `integrate`.

Как мы видим, двойное интегрирование даёт кубическую кривую.

Результат напоминает синусоиду. Это связано с тем, что интегрирование действует как фильтр нижних частот. Посмотрим, какие частоты у нас остались.

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

Листинг 19: Получение спектра.

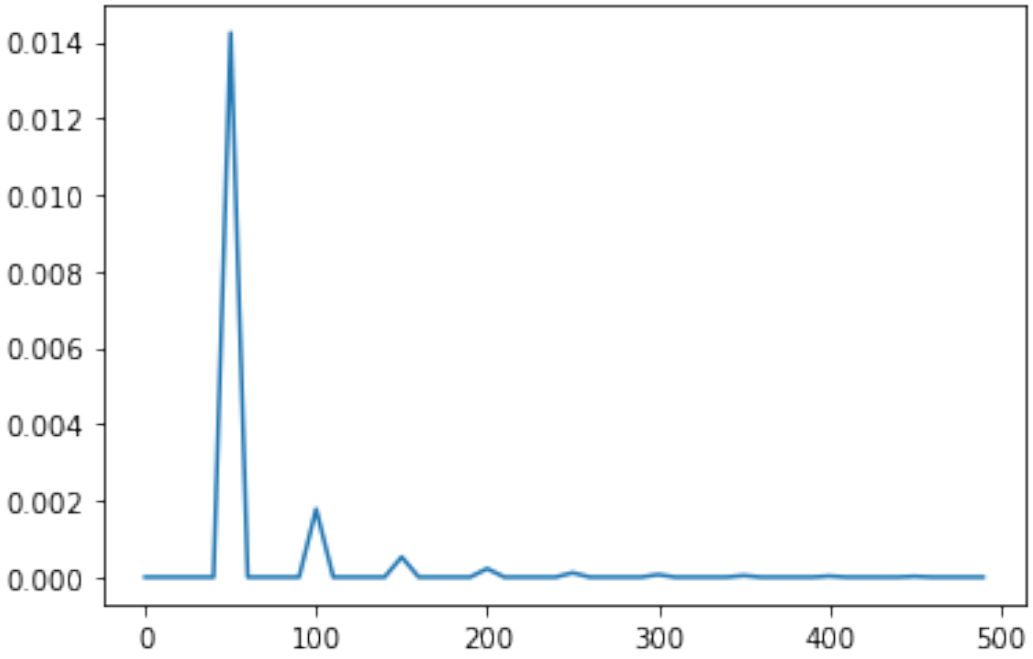

Рис. 19: Спектр полученного сигнала.

Из рисунка выше (Рис.19) видно, что мы отфильтровали почти всё, кроме основного.

В данном упражнении было изучено влияние двойного интегрирования на сигнал. Для этого был создан пилообразный сигнал, к которому два раза был применён `integrate`. По полученным результатам был сделан вывод, что интегрирование действует как фильтр нижних частот.

5 Упражнение 9.5

В этом упражнении изучается влияние второй разности и второй производной.

Начнём с создания кубического сигнала.

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

Листинг 20: Кубический сигнал.

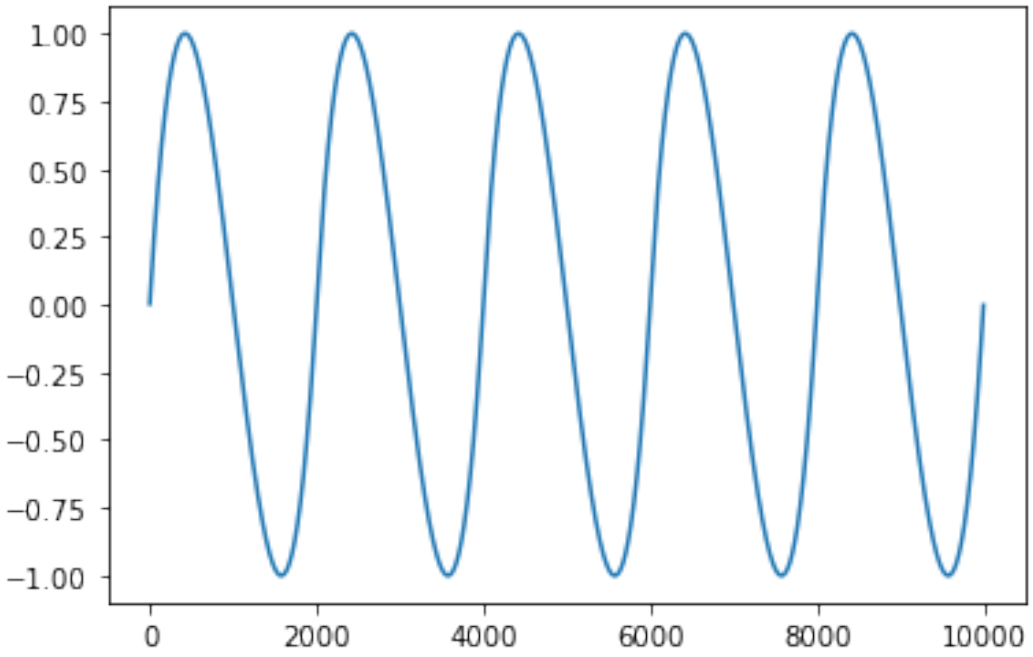

Рис. 20: Кубический сигнал.

Теперь вычислим вторую разность, дважды применив `diff`.

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

Листинг 21: Двойное примнение `diff`.

Результат второго применения - пилообразная волна.

Теперь вычислим вторую производную, дважды применив `differentiate`

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

Листинг 22: Двойное примнение `differentiate`.

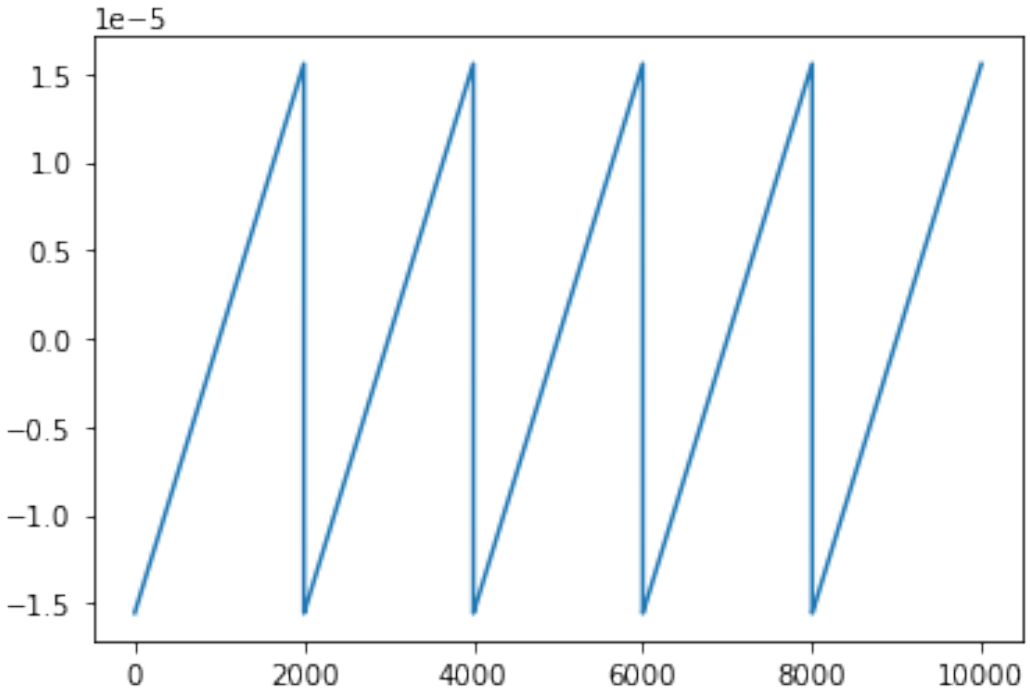

Рис. 21: Сигнал после двойного применения `diff`.

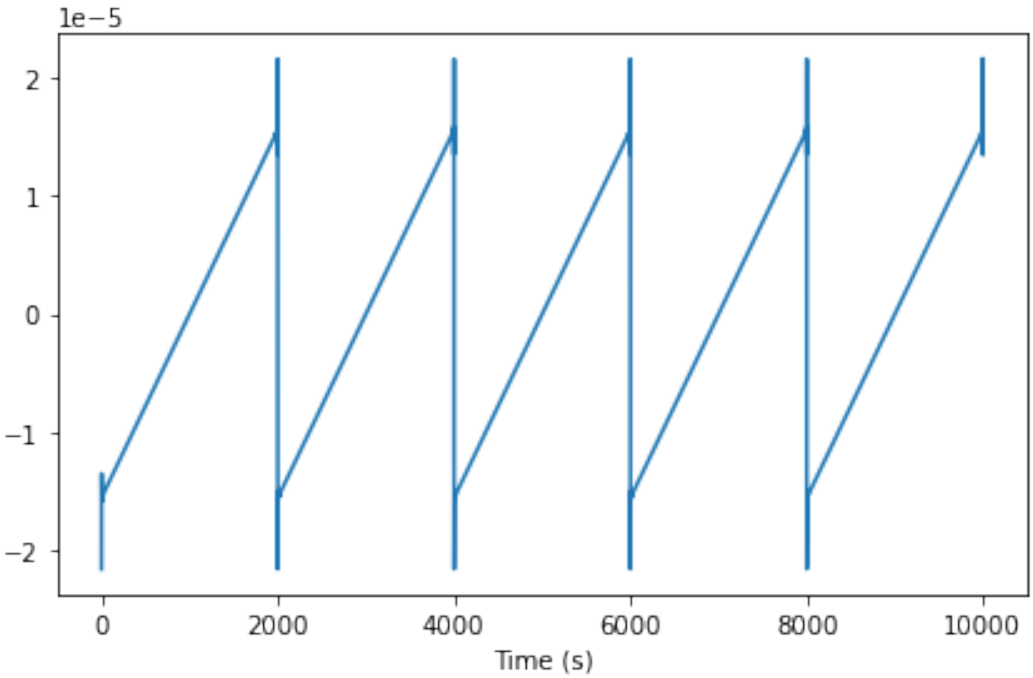

Рис. 22: Сигнал после двойного применения `differentiate`.

Видно, что в результате двойного дифференцирования был получен пилообразный сигнал со ”звоном”. Как мы уже выяснили, это связано с тем, что производная параболического сигнала не определена в точках.

Теперь рассчитаем фильтры, соответствующие второй разнице и второй про-

изводной, и сравним их.

```
diff_window = numpy.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()
```

Листинг 23: Вычисление фильтра для второй разности.

```
deriv_filter = in_wave.make_spectrum()
deriv_filter.hs = (PI2 * 1j * deriv_filter.fs)**2

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

Листинг 24: Вычисление фильтра для второй производной и сравнение фильтров.

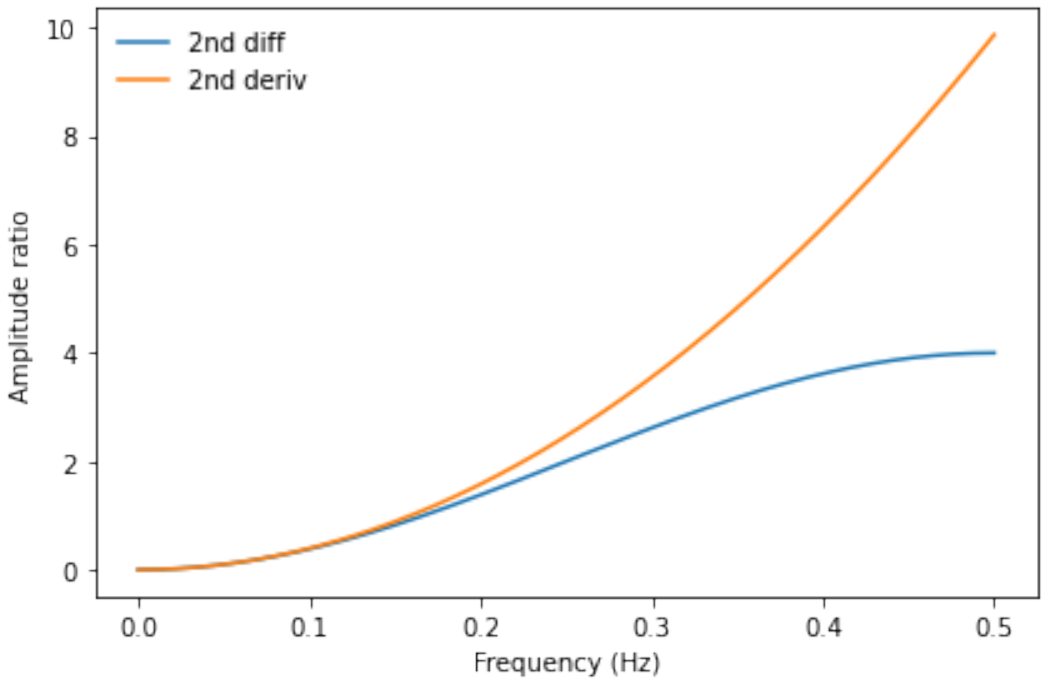

Рис. 23: Сравнение фильтров.

Как мы видим из Рис.23, оба фильтра - фильтры высоких частот, усиливающие компоненты самых высоких частот. Вторая производная - параболическая, поэтому она сильнее всего усиливает самые высокие частоты. Вторая разница очень близка к второй производной на низких частотах, однако потом достаточно сильно отклоняется от неё.

В данном упражнении было изучено влияние второй разности и второй производной на сигнал. Для этого был создан кубический сигнал, для которого были получены вторая разность и вторая производная. Их результат - пилообразный сигнал. Однако для второй производной сигнал получился со ”звоном”. Затем для них были вычислены фильтры. При сравнении полученных фильтров был сделан вывод, что вторая производная усиливает высокие частоты сильнее, чем вторая разница.

6 Выводы

В ходе выполнения данной лабораторной работы были изучены дифференцирование и интегрирование и их влияние на сигнал. Был сделан вывод, что интегрирование действует как фильтр нижних частот, а дифференцирование усиливает высокие частоты. Также был сделан вывод, что результат совокупной суммы и спектрального интеграла очень похожи.

Кроме того, важно помнить, что при взятии производных сигнал часто получается со ”звоном”.