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
Сахибгареев Рамис Ринатович

Преподаватель:
Богач Наталья Владимировна

Санкт-Петербург 2021

Contents

List of Figures

Listings

1 Part 1: Research and execution of chap08

In this part we need to research and execute existing chap09.ipynb file, that contains information about derivative and integral filters.

Difference is an analogue for derivative and it acts like high-pass filter. Cumulative sum is a analogue of integral and it acts like low-pass filter.

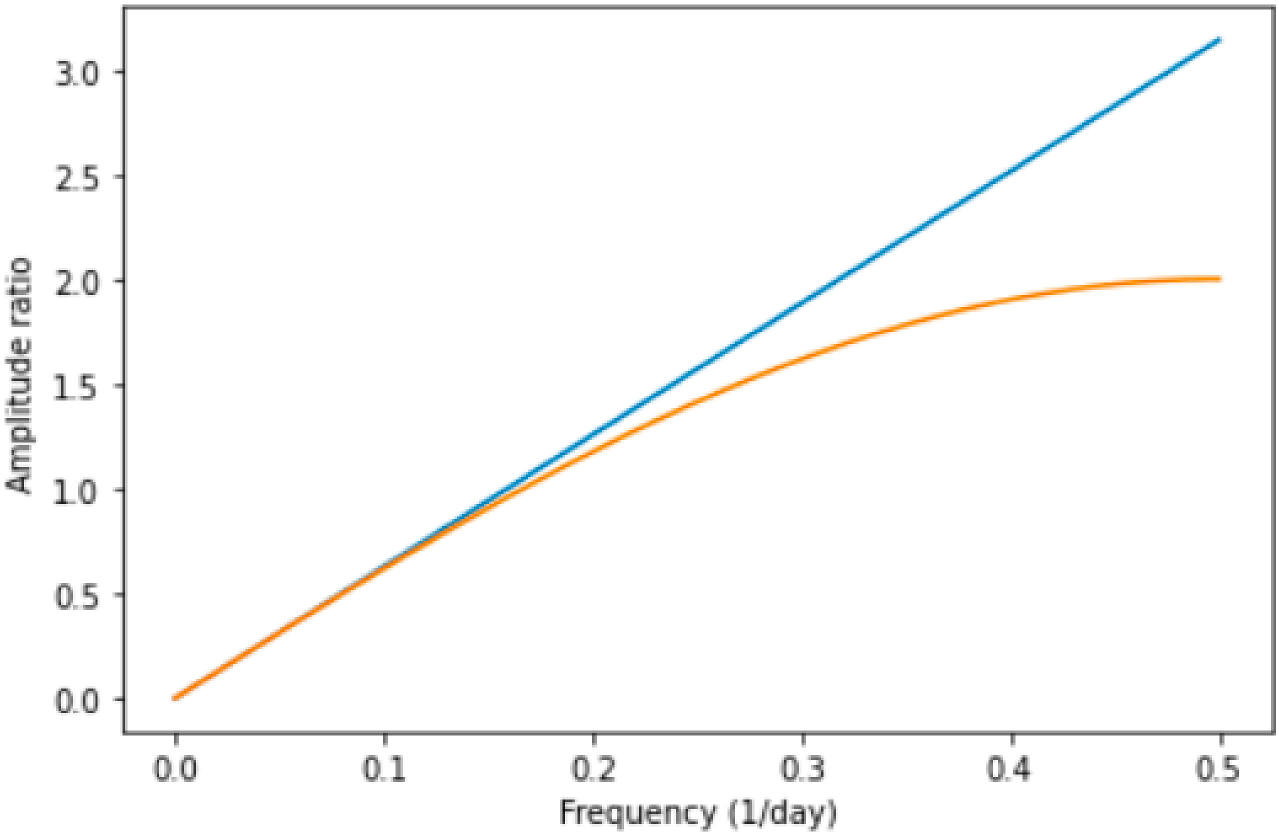


Figure 1: Diff and deriv fitlers

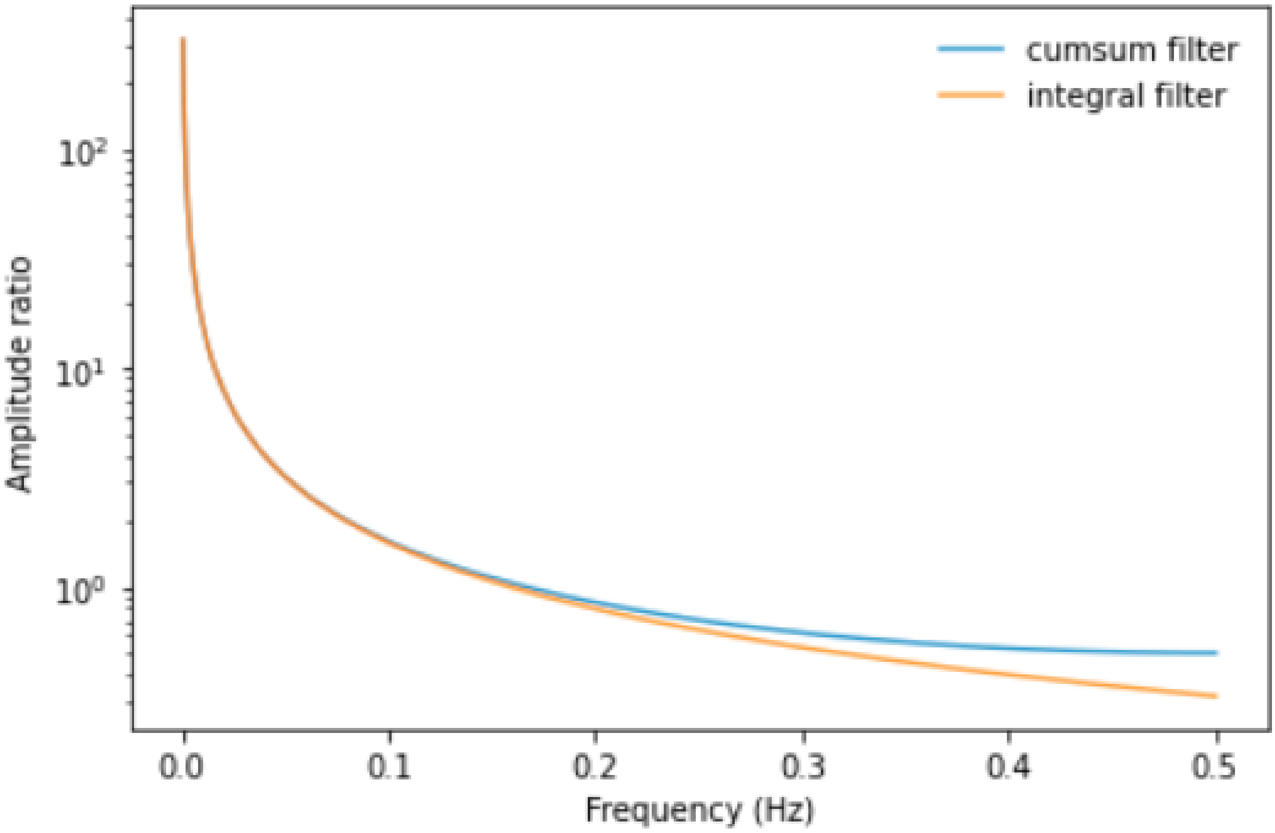
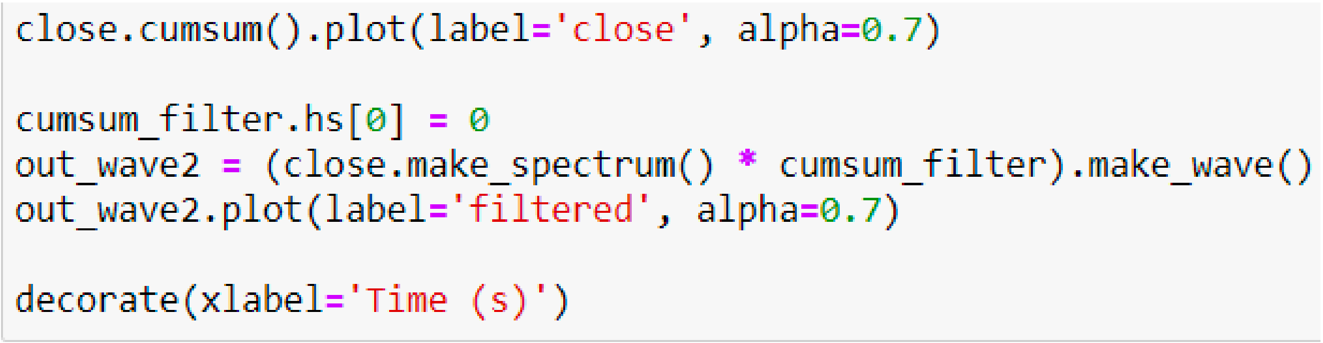


Figure 2: Cumsum and integral filters

But convolution theorem works only for periodic signals, that’s why trying to apply cumsum/integral filter on the non periodic signal results wrong result.



```
close.cumsum().plot(label='close', alpha=0.7)

cumsum_filter.hs[0] = 0
out_wave2 = (close.make_spectrum() * cumsum_filter).make_wave()
out_wave2.plot(label='filtered', alpha=0.7)

decorate(xlabel='Time (s)')
```

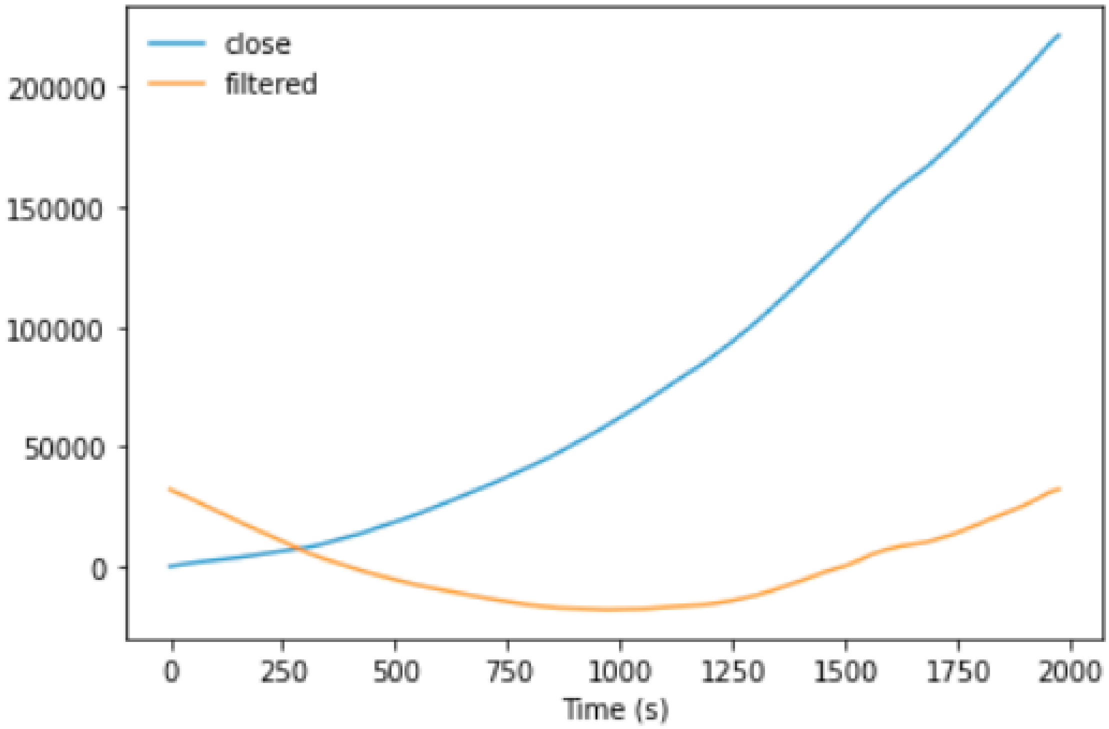


Figure 3: Trying to apply convolution theorem on non periodic signal

2 Part 2: Diff on wave and on its spectrum

In this part we need to compare diff calculation result on the wave and on its spectrum. For test triangle signal was used.

```
import numpy as np
import matplotlib as plt
from thinkdsp import *

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

wave_diff = wave.diff()
wave_diff.plot()
decorate(xlabel='Time (s)')

wave.make_spectrum().differentiate().make_wave().plot()
decorate(xlabel='Time (s)')

```

Listing 1: Diff and deriv code

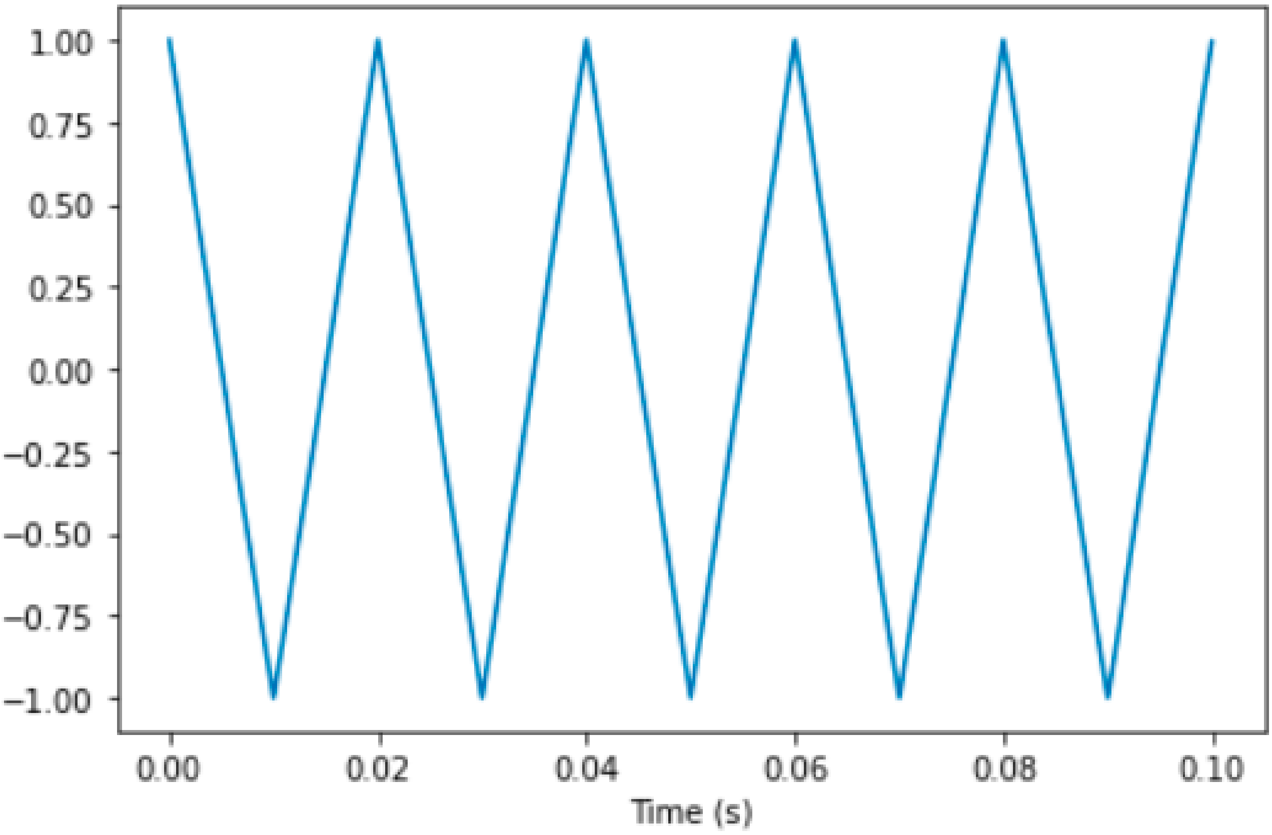


Figure 4: Triangle wave

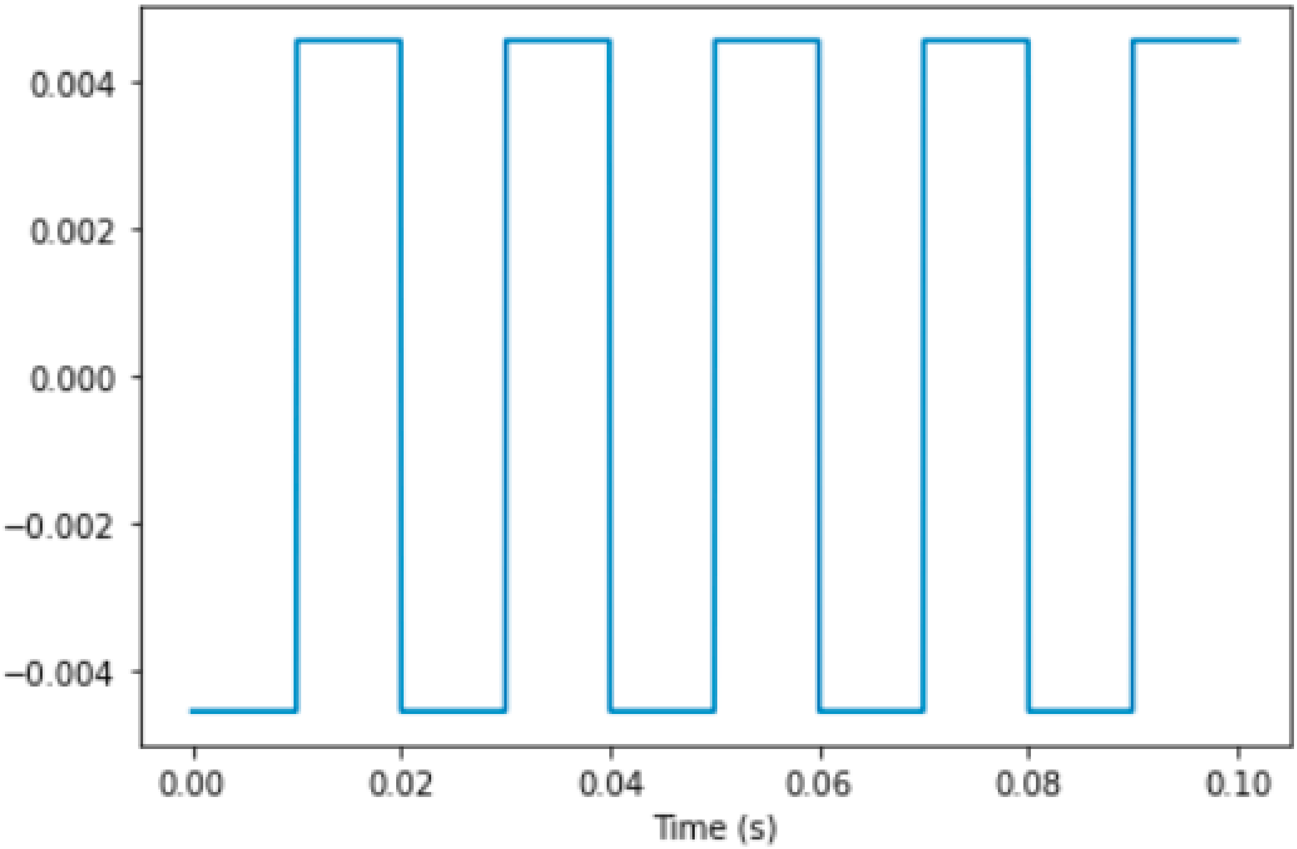


Figure 5: Triangle wave’s diff

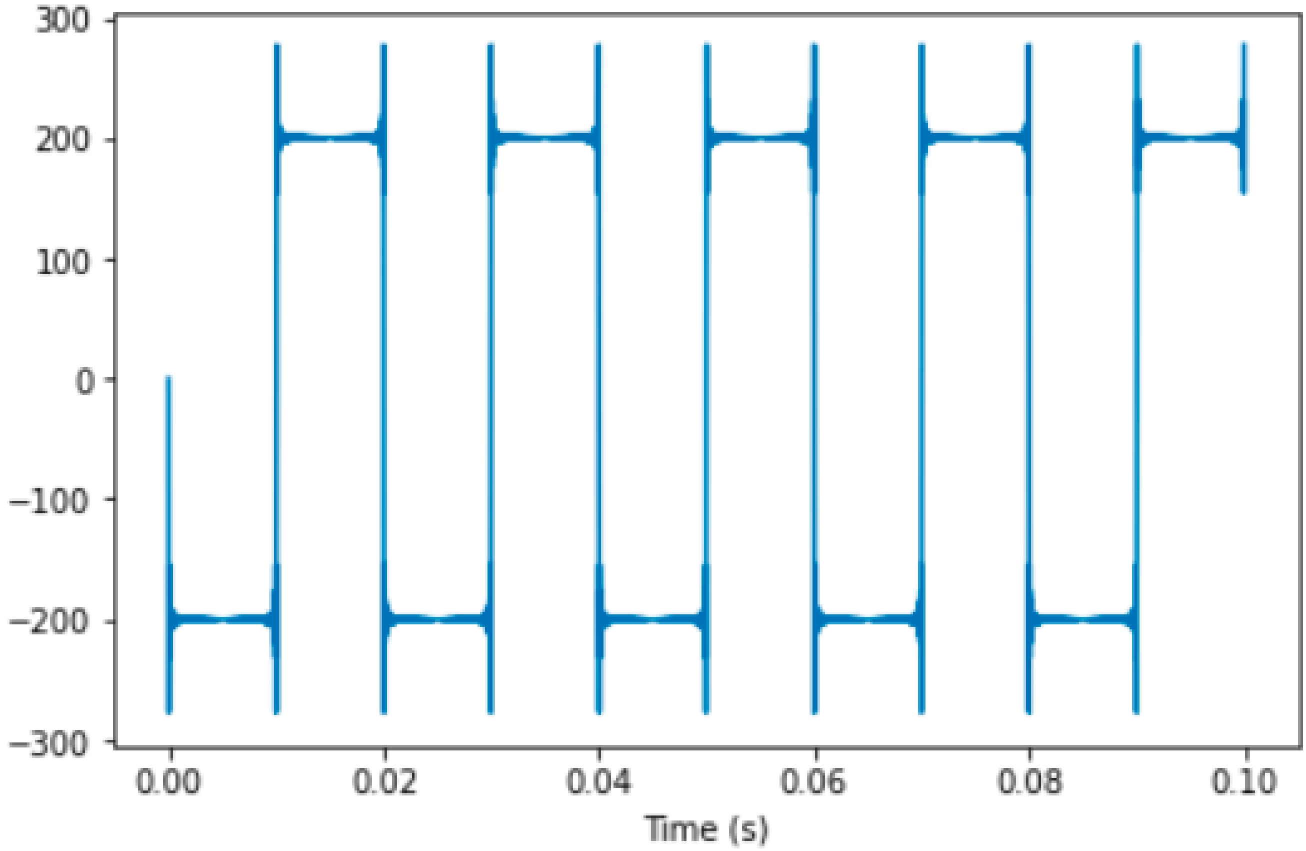


Figure 6: Spectrum's diff

We can see, that diff of a triangle signal is a square signal. However, applying diff on the spectrogram results something like square signal, but with peaks on the edges of the squares. It happens because we cannot differentiate triangle signal on the edge points.

3 Part 3: Cumsum and integrate

In this part, we need to compare cumsum and integrate. To do it we can use diff result from the previous part, trying to restore original signal.

```
wave_sum = wave_diff.cumsum()
wave_sum.unbias()
spec_int = wave_diff.make_spectrum().integrate()
spec_int.hs[0] = 0
wave_int = spec_int.make_wave()
wave_int.normalize()
wave_sum.plot()
wave_int.plot()

```

Listing 2: Integration code

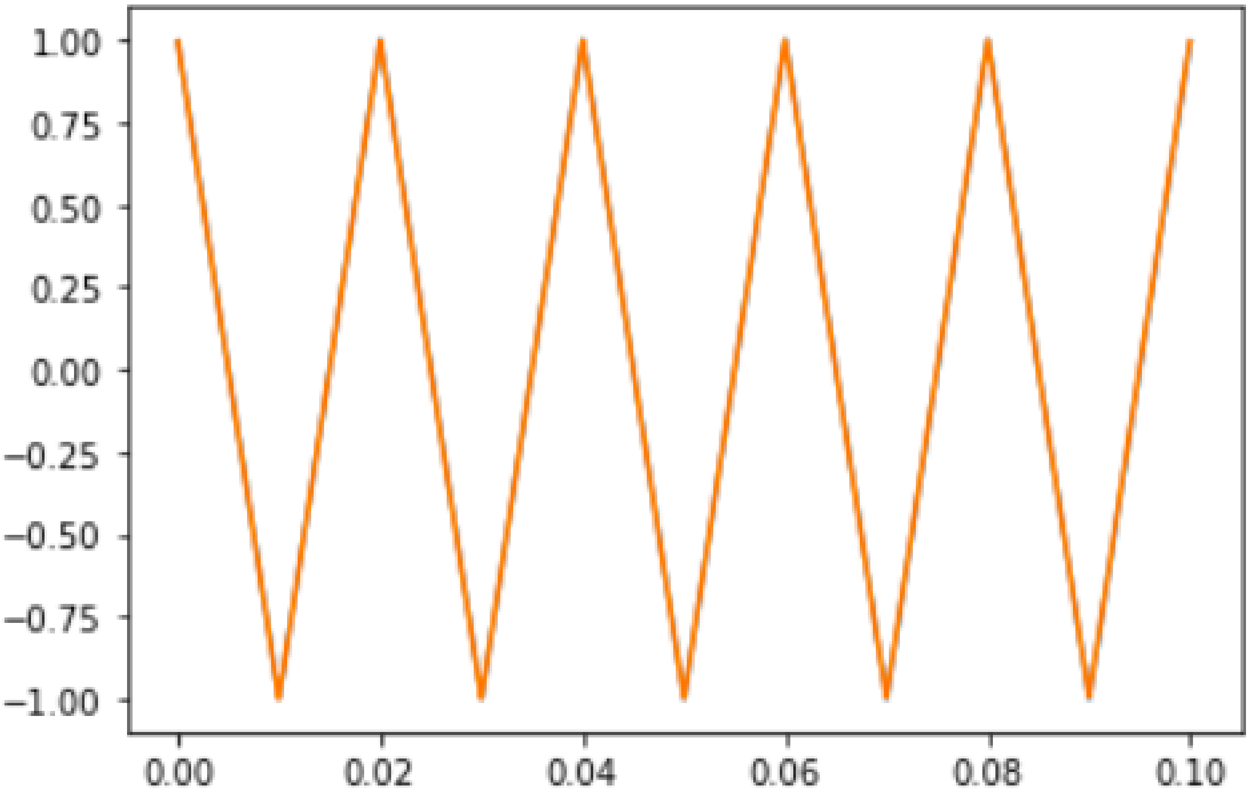


Figure 7: Restore result

We can see, that they both restored signal successful and absolutely same. However, for cumsum we need to perform unbias, and for integral we need to set hs[0] as well as normalize the result.

4 Part 4: Double integration

In this part, we need to perform double cumsum and integration using sawtooth signal.

```
wave = SawtoothSignal(freq=50).make_wave(duration=0.1, framerate=44100)
wave.plot()
decorate(xlabel='Time (s)')
wave_sum = wave.cumsum()
wave_sum.unbias()
wave_sum = wave_sum.cumsum()
wave_sum.plot()
spec_int = wave.make_spectrum().integrate().integrate()
spec_int.hs[0] = 0
wave_int = spec_int.make_wave()
wave_int.plot()
wave_int.make_spectrum().plot()
plt.xlim(left=0, right=500)

```

Listing 3: Double integration

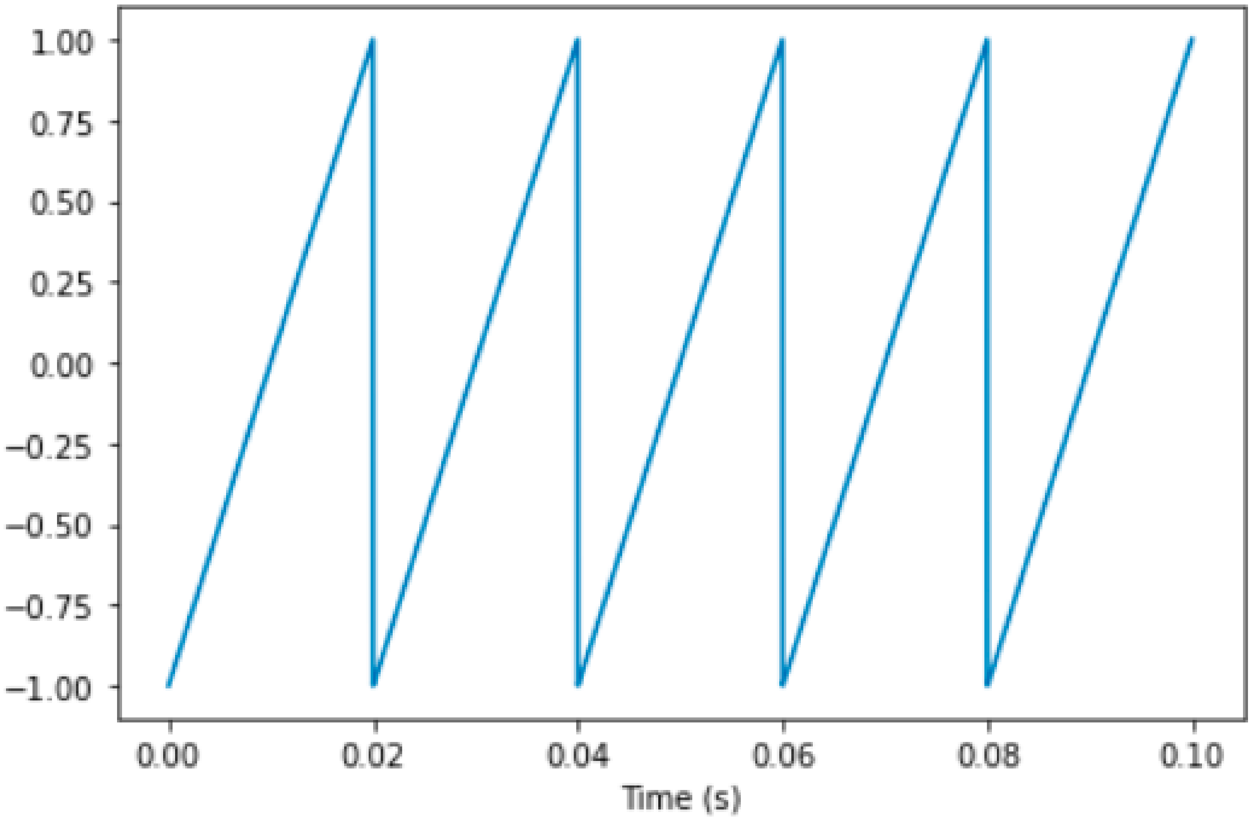


Figure 8: Sawtooth signal to be integrated

After applying cumsum and integration we got next results:

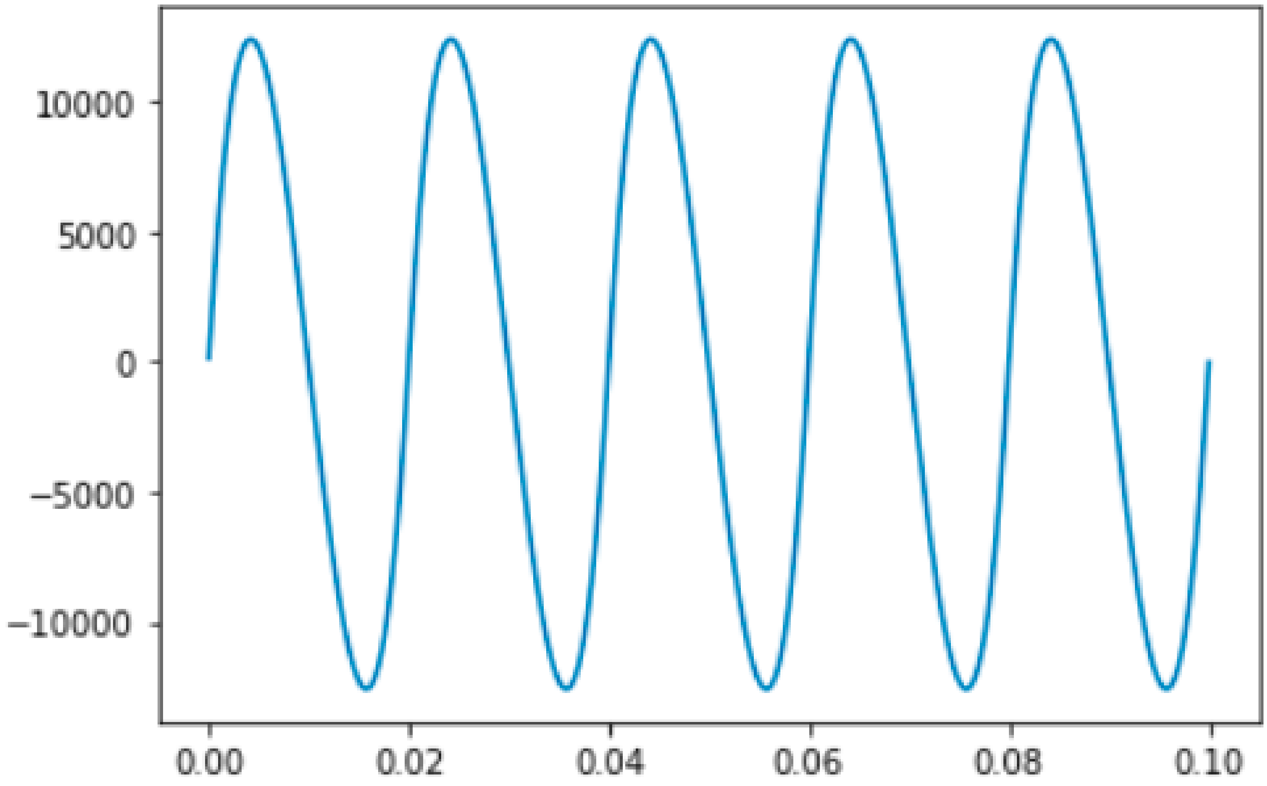


Figure 9: Cumsum result

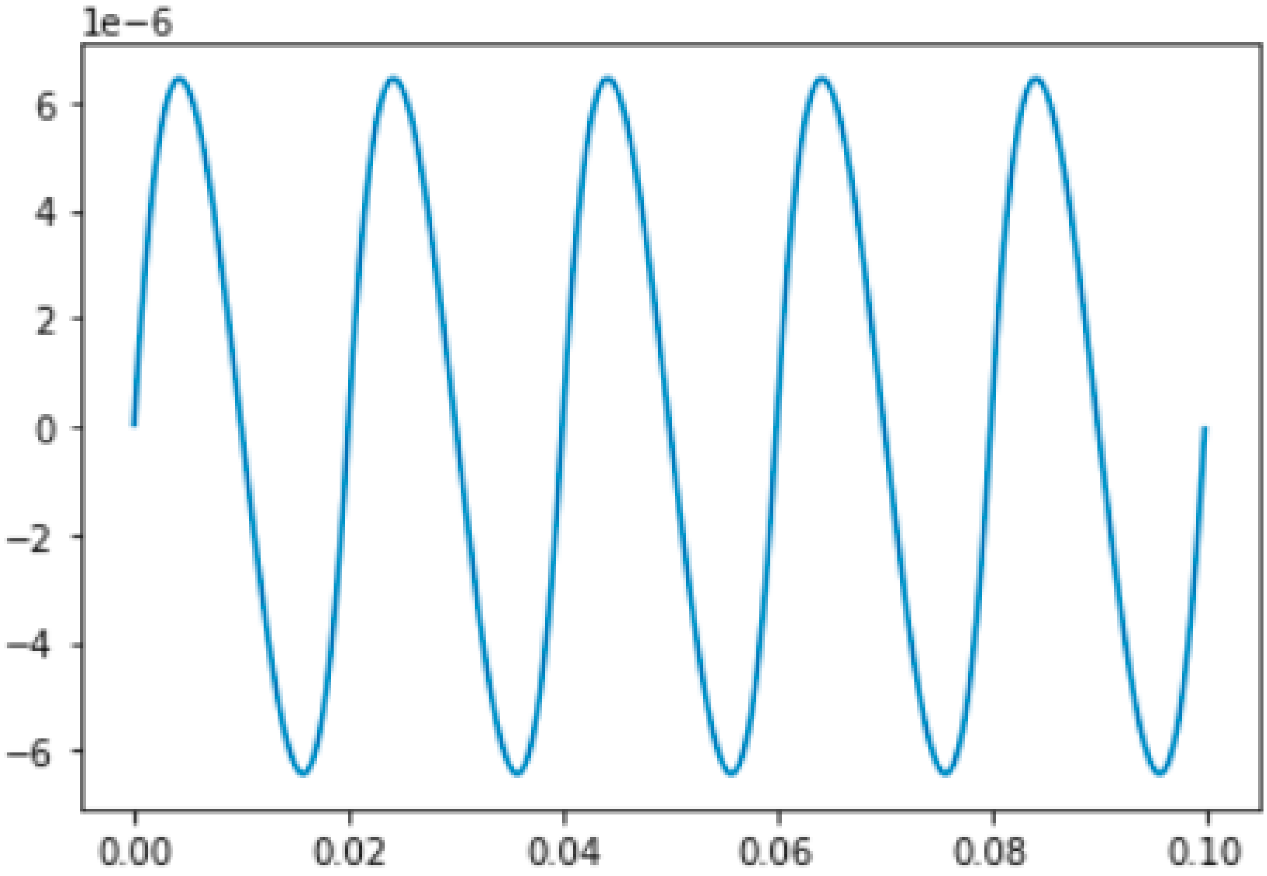


Figure 10: Integration result

We can see, that there are no differences between two methods. Signal looks like sin signal, but it is a cubic signal. The reason it happens is that integration acts like low pass filter, so only fundamental frequency has most of signals power.

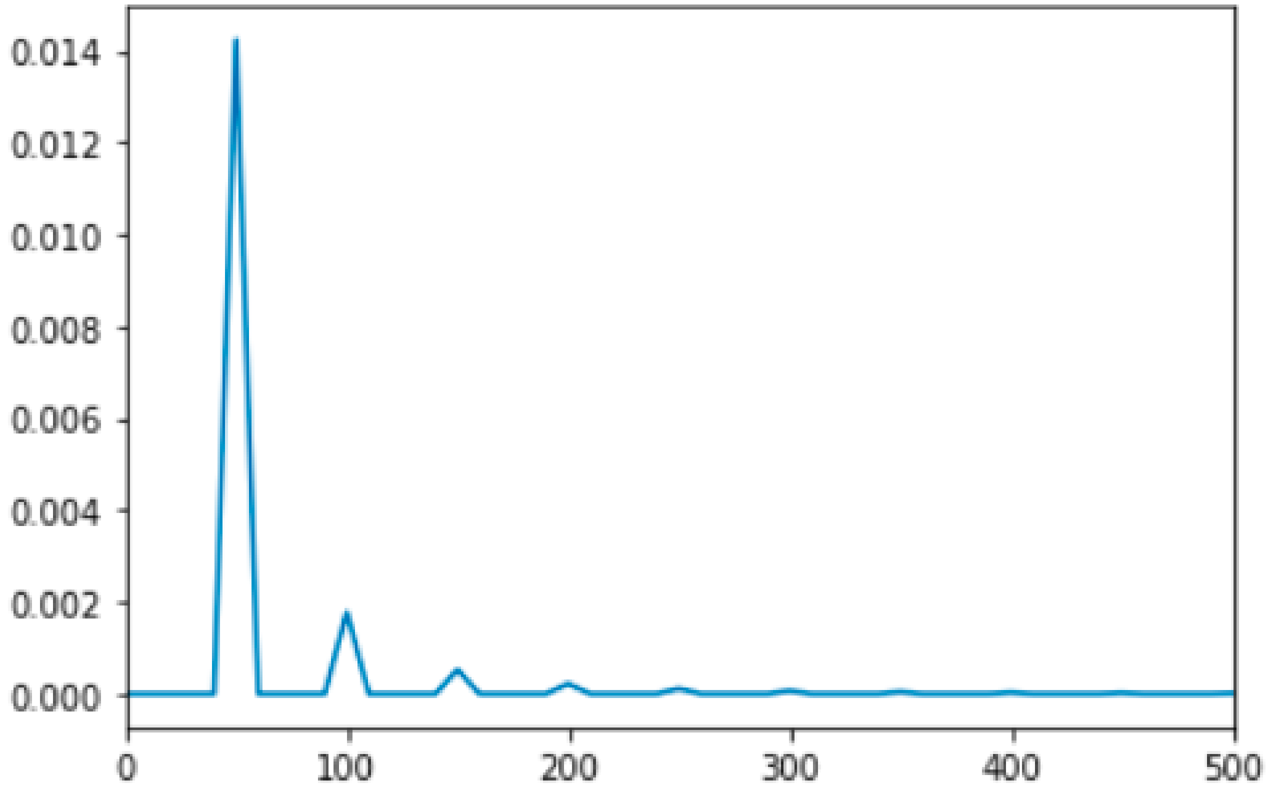


Figure 11: Integration spectrum

5 Part 5: Double differentiate

In this part, we need to perform double diff and derivative using cubic signal. It is good to have framerate = 1, because that way filters will be same scale.

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

wave_diff = wave.diff().diff()
wave_diff.plot()

wave_der = wave.make_spectrum().differentiate().differentiate().make_wave()
wave_der.plot()

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

deriv_filter = wave.make_spectrum()
deriv_filter.hs = (np.pi * 2 * 1j * deriv_filter.fs)**2
deriv_filter.plot(label='2nd deriv')
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude ratio')

```

Listing 4: Double diff

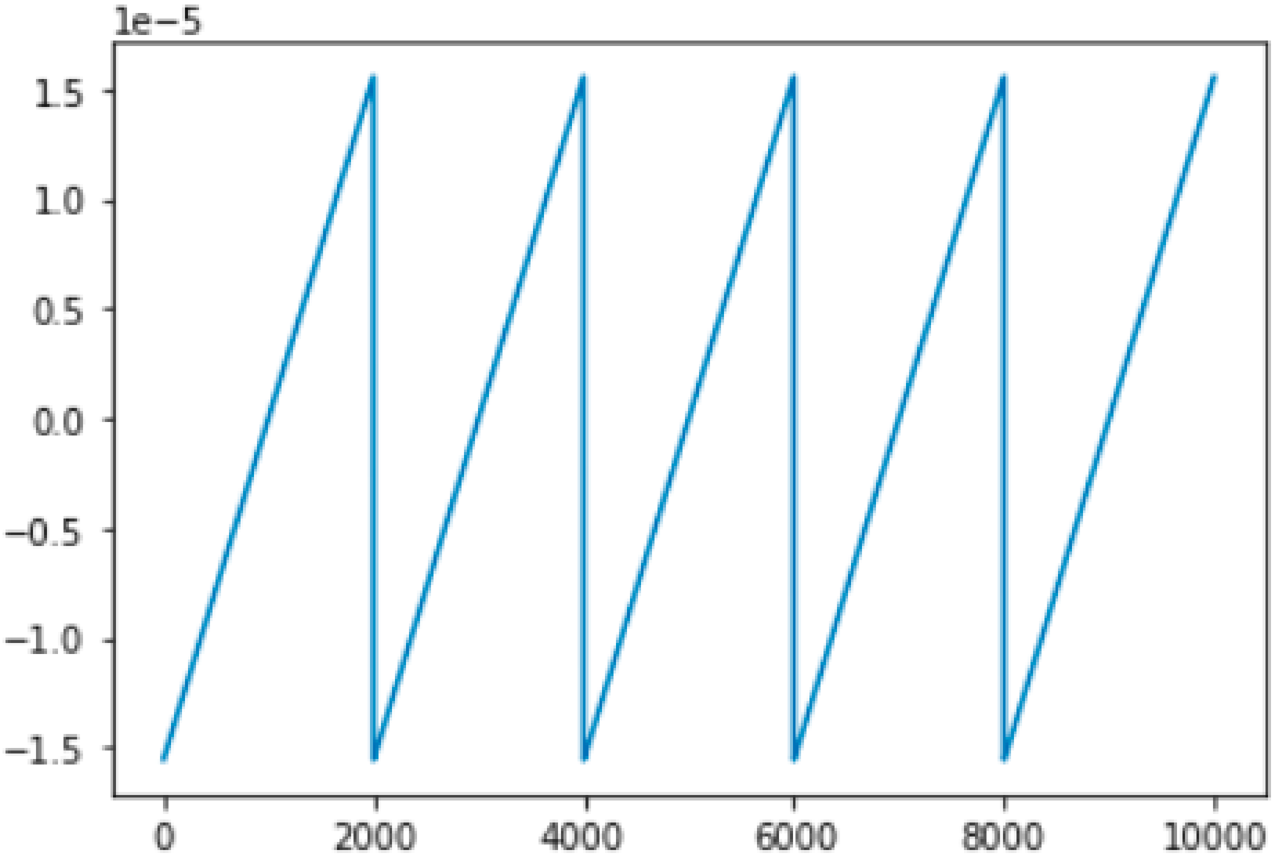


Figure 12: Diff result

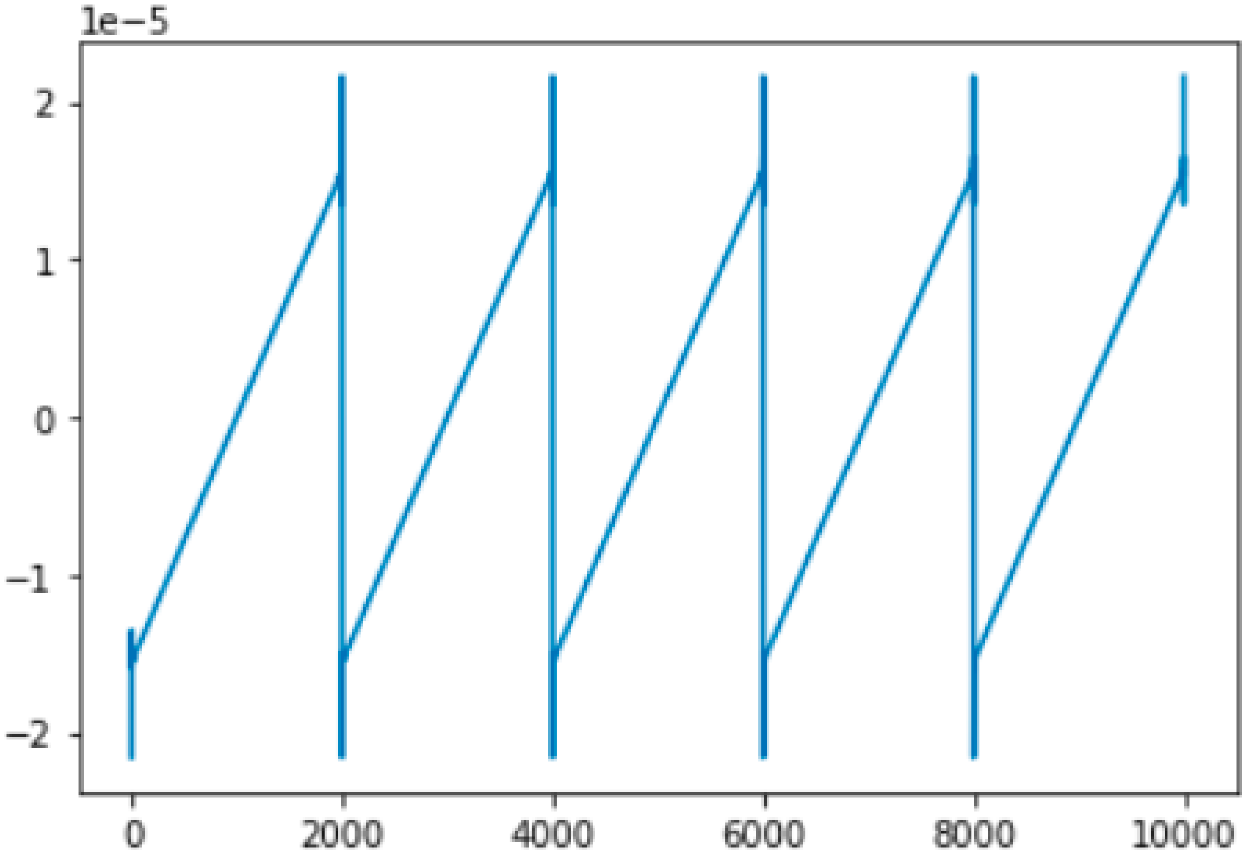


Figure 13: Deriv result

We can see, that wave differentiation results a clear sawtooth signal. Spectrum differentiation, on the other hand, also results a sawtooth signal, but it has spikes on the edges. It happens because parabolic signal, that we got on the first diff, has some points, that cannot be differentiated.

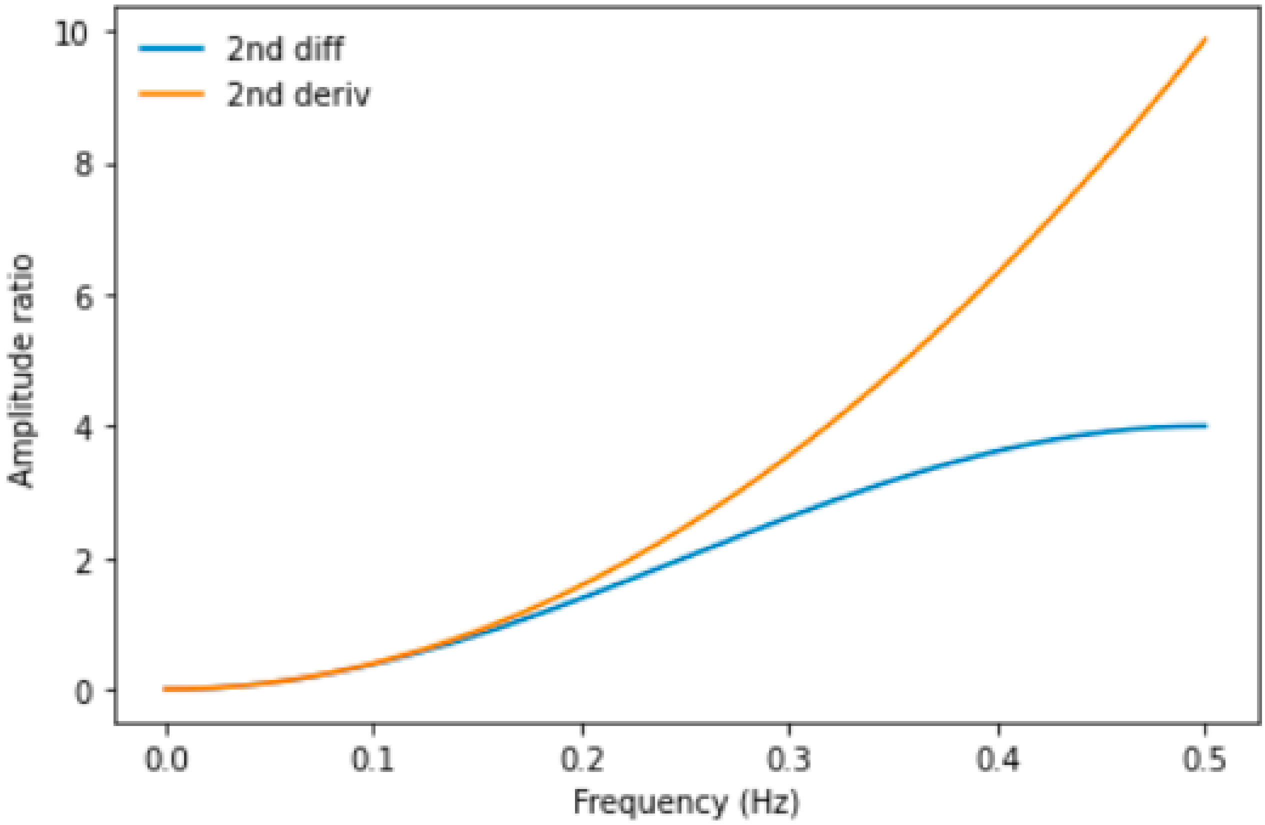


Figure 14: Comparing filters

.

For 2nd diff filter is knows - [-1 2 -1]. For deriv filter we can simply perform double applying of 1st deriv filter. We see, that trend continues - deriv filter is more exponential, while diff filter has fast rice on the low frequencies and slows down on the high frequencies. That's why they acts same on the low frequencies, but differs on the high frequencits.

6 Conclusion

We've learned, how cumulutive sum, integration, diff and deriv affects the signal. Integration and cumulative sum acts like a low pass filter, while diff and deriv acts like a high pass filter. Because they are opposites, we can convert signal into different forms using this operations, also restoring the signals. However, due to loss of an information during differentiation procedure, we cannot restore signal fully.