Санкт-Петербургский государственный политехнический
университет Петра Великого.
**Высшая школа интеллектуальных сисем и
суперкомпьютерных технологий**

Лабораторная работа

Дифференцирование и интегрирование.

Работу выполнила студентка:
_____ А. И.Луцкевич
«____» _____ 2021 г.

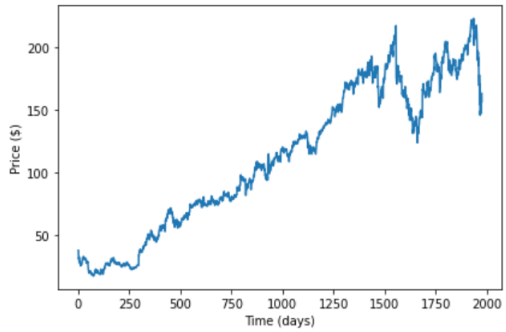
Преподаватель лабораторных
работ:
_____ Н. В.Богач
«___» _____ 2021 г.

Санкт-Петербург, 2021 г.

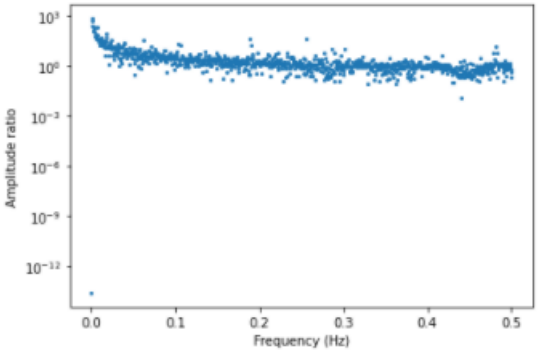
Суть работы 9.1:

Необходимо запустить все примеры из chap09.ipynb, прочитать все описания, после чего заменить пилообразный сигнал на входе на непериодические данные Facebook, а далее повторно запустить все примеры.

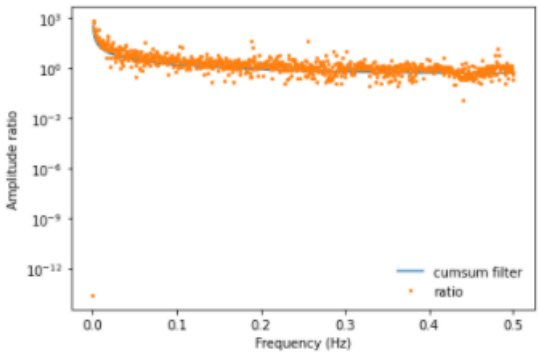
Загрузим непериодические данные Facebook и представим их в виде графика.



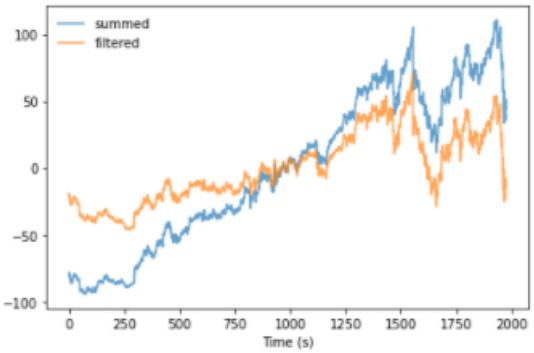
Между гармониками входные компоненты маленькие, поэтому установим эти отношения на NaN и напечатаем Ratio spectrum.



Необходимо дальше сравнить с фильтром - данные не совпадают из-за большого разброса.



И дальше сравним полученные нами результаты с исходными. Графики не сопадают и не равны.

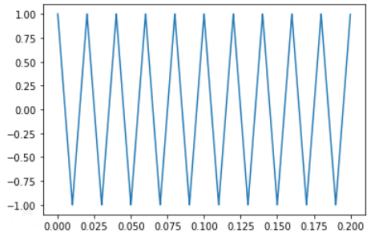


Суть работы 9.2:

Необходимо изучить влияние diff и differentiate на сигнал. Для этого необходимо создать треугольный сигнал, распечатать его и применить к нему diff. Затем вычислить спектр треугольного сигнала, применить differentiate и потом напечатать результат.

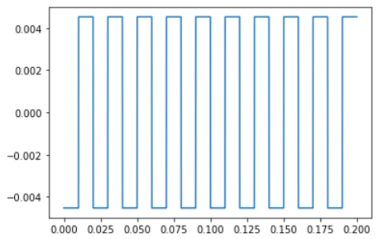
Создадим треугольный сигнал длиной 0.2 с и выведем его.

```
In [3]: in_wave = TriangleSignal(freq=50).make_wave(duration=0.2, framerate=44100)
        in_wave.plot()
```



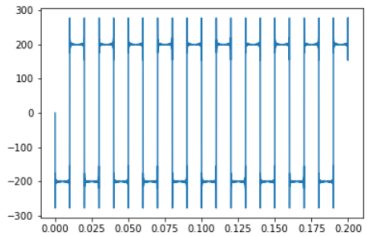
После изучим влияние diff - применим к сигналу diff и выведем график.

```
In [4]: out_wave = in_wave.diff()
        out_wave.plot()
```



Получилась прямоугольная волна (т.е. diff треугольной волны - это промоугольная).
Теперь к спектру сигнала применим differentiate и изучим его влияние.

```
In [5]: out_wave2 = in_wave.make_spectrum().differentiate().make_wave()
        out_wave2.plot()
```



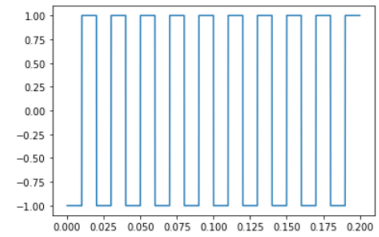
Некие "скопления"происходят из-за того, что производная треугольной волны не определена в самих точках "треугольника". Из-за этого и получается такой необычный эффект.

Суть работы 9.3:

Необходимо изучить влияние cumsum и integrate на сигнал. Для этого необходимо создать прямоугольный сигнал, распечатать его и применить к нему cumsum. Затем вычислить спектр треугольного сигнала, применить integrate и потом напечатать результат.

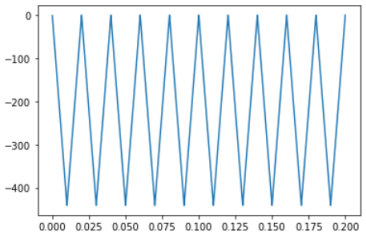
Создадим прямоугольный сигнал длиной 0.2 с и выведем его.

```
In [8]: in_wave = SquareSignal(freq=50).make_wave(duration=0.2, framerate=44100)
        in_wave.plot()
```



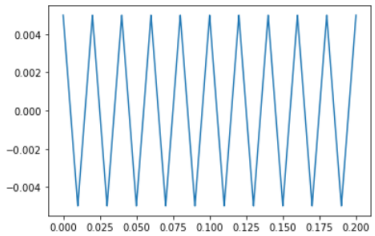
Изучим влияние cumsum на полученный сигнал - применим к сигналу и выведем график.

```
In [9]: out_wave = in_wave.cumsum()
        out_wave.plot()
```



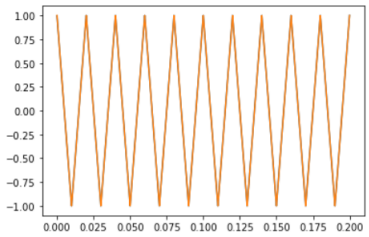
Получилась треугольная волна (т.е. cumsum прямоугольной волны - треугольная волна).
Посмотрим спектральный интеграл, который также является треугольной волной.

```
In [10]: spectrum = in_wave.make_spectrum().integrate()
         spectrum.hs[0] = 0
         out_wave2 = spectrum.make_wave()
         out_wave2.plot()
```



Эти сигналы схожи. Для докахательства сместим две волны и нормальзуем. Также определим схожесть данных графиков.

```
In [11]: out_wave.unbias()
         out_wave.normalize()
         out_wave2.normalize()
         out_wave.plot()
         out_wave2.plot()
```




```
In [12]: out_wave.max_diff(out_wave2)
Out[12]: 0.0045351473922902175
```

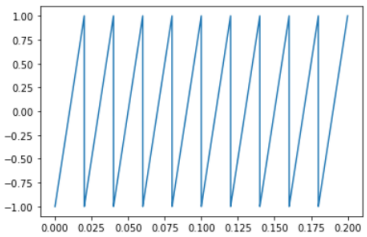
В результате видно, что различий в сигнал практически нет.

Суть работы 9.4:

Необходимо изучить влияние двойного интегрирования. Для этого надо создать пилообразный сигнал, вычислить его спектр и дважды применить integrate, после чего напечатать результирующий сигнал, и его спектр.

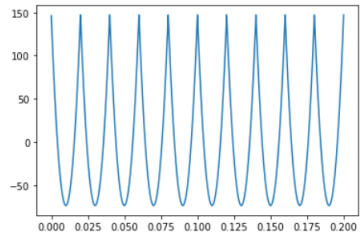
Создадим пилообразный сигнал длиной 0.2 и выведем его.

```
In [15]: in_wave = SawtoothSignal(freq=50).make_wave(duration=0.2, framerate=44100)
         in_wave.plot()
```



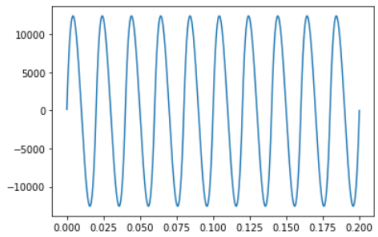
Применим к сигналу cumsum и выведем график.

```
In [16]: out_wave = in_wave.cumsum()
         out_wave.unbias()
         out_wave.plot()
```



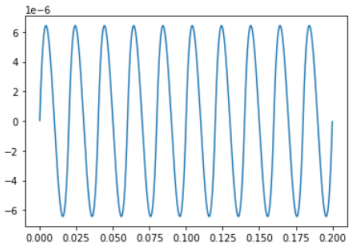
Получилась парабола.
Применим второй раз функцию cumsum.

```
In [17]: out_wave = out_wave.cumsum()
         out_wave.plot()
```



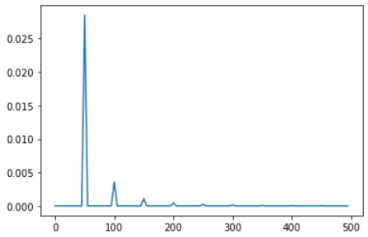
После второго раза получилась кубическая кривая.
Для исходного пилообразного сигнала также применим функцию integrate.
В результате получается такая же кубическая кривая.

```
In [18]: spectrum = in_wave.make_spectrum().integrate().integrate()
         spectrum.hs[0] = 0
         out_wave2 = spectrum.make_wave()
         out_wave2.plot()
```



Для доказательства, что график не является синусоидой, посмотрим на спектр полученного сигнала.

```
In [19]: out_wave2.make_spectrum().plot(high=500)
```

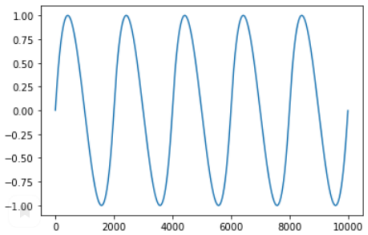


Суть работы 9.5:

Необходимо изучить влияние второй разности и второй производной. Для этого надо создать CubicSignal, вычислить двойную разность применив diff и проанализировать результат. После этого вычислить вторую производную, дважды применив differentiate к спектру.

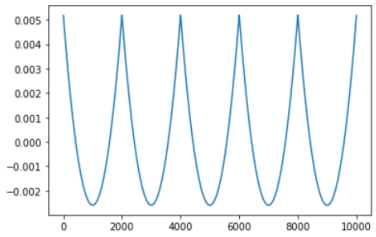
Создадим CubicSignal и выведем его на экран.

```
In [20]: in_wave = CubicSignal(freq=0.0005).make_wave(duration=10000, framerate=1)
         in_wave.plot()
```



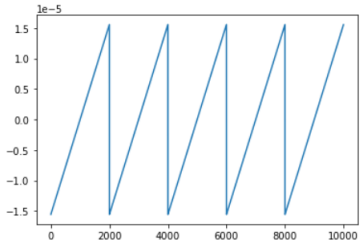
Применим к сигналу diff и выведем опять же график.

```
In [21]: out_wave = in_wave.diff()
         out_wave.plot()
```



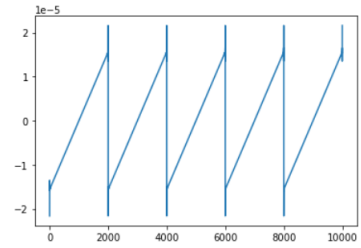
Получили в итое параболу. Применим второй раз функцию diff.

```
In [22]: out_wave = out_wave.diff()
         out_wave.plot()
```



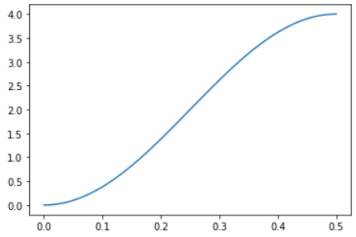
В итоге получилась пилообразная волна. Применим к исходному сигналу два раза differentiate.

```
In [23]: spectrum = in_wave.make_spectrum().differentiate().differentiate()
         out_wave2 = spectrum.make_wave()
         out_wave2.plot()
```



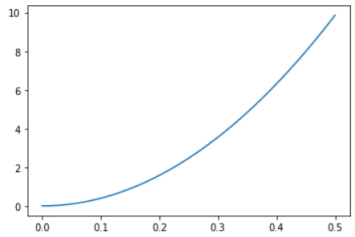
Получился похожий пилообразный сигнал, но с каким-то "шумом"(т.к. производная параболического сигнала не определена в некоторых точках). Теперь найдем необходимый фильтр для первой производной.

In [27]:
```
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()
```



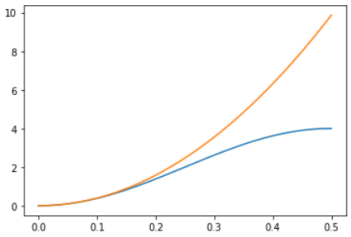
Возведем фильтр первой производной в квадрат и найдем фильтр для второй производной.

In [28]:
```
deriv_filter = in_wave.make_spectrum()
deriv_filter.hs = (PI2 * 1j * deriv_filter.fs)**2
deriv_filter.plot()
```



Полученный фильтр является параболическим. Из-за этого он усиливает самые высокие частоты.
Сделаем два графика на одном для более удобного представления.

In [29]:
```
diff_filter.plot(label='2nd diff')
deriv_filter.plot(label='2nd deriv')
```



Оба фильтра высокочастотные и усиливают компоненты наивысшей частоты. Из-за этого на низких частотах разницы между ними нет, она появляется уже на высоких частотах.

Заключение:

В данной лабороторной работы было изучено диференциирование и интегрирование функций. Были разобраны такие функции как diff, differentiate,

cumsum и integrate, а также изучено двойное интегрирование, вторая разность и действие второй производной на разные сигналы.