Федеральное государственное автономное образовательное учреждение высшего образования «Национальный Исследовательский Университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №4
ПРЕДМЕТ «ЧАСТОТНЫЕ МЕТОДЫ»
ТЕМА «ЛИНЕЙНАЯ ФИЛЬТРАЦИЯ»

Лектор: Перегудин А. А.
Практик: Пашенко А. В.
Студент: Румянцев А. А.
Поток: ЧАСТ.МЕТ. 1.3

Факультет: СУиР
Группа: R3241

Санкт-Петербург
2024

Содержание

1 Задание 1. Спектральное дифференцирование.

Зададим в python список t от -100 до 100 включительно с шагом dt и рассмотрим зашумленный сигнал вида

$$y = \sin(t) + a \cdot (\text{rand}(\text{len}(t)) - 0.5).$$

Построим соответствующий график при переменных $a = 0.2$, $dt = 0.25$. На всех графиках в названии указываются значения используемых параметров для удобства рассматривания различных результатов и последующего сравнения.

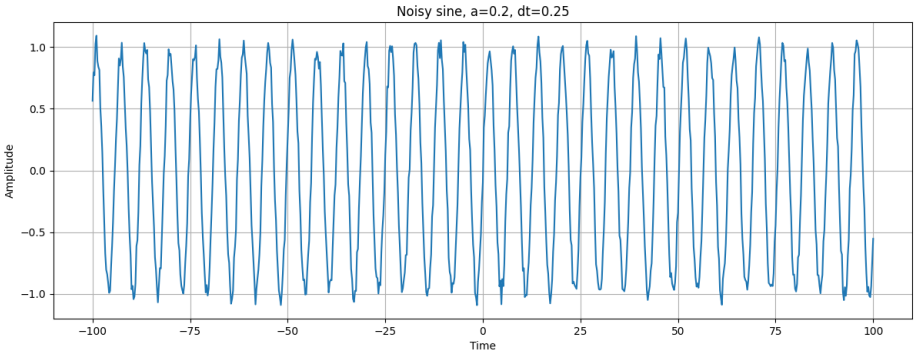

Рис. 1: График зашумленного сигнала.

Найдем численную производную от данного сигнала, используя формулу поэлементного дифференцирования

$$\frac{y(k+1) - y(k)}{dt},$$

после чего построим график.

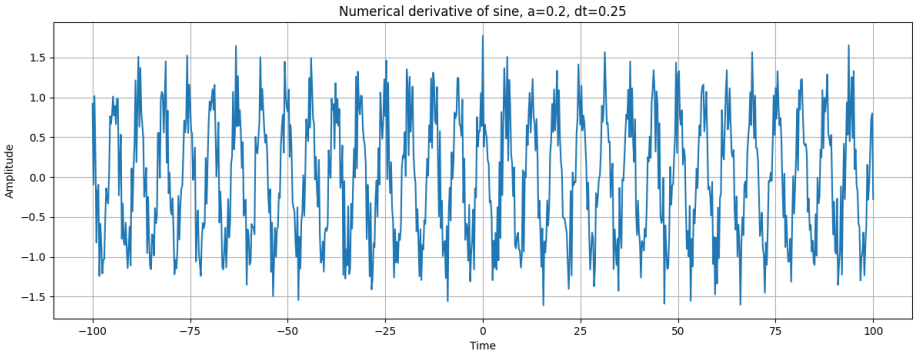

Рис. 2: Численная производная зашумленного сигнала.

Найдем спектральную производную от зашумленного сигнала. Для прямого и обратного преобразования Фурье будем использовать численное интегрирование (trapz). Чтобы превратить Фурье-образ сигнала в Фурье-образ производной, необходимо домножить результат преобразования Фурье на $2\pi i\nu$, где ν – частота (Гц), таким образом получим формулу

$$\mathcal{F}\left\{\frac{d}{dt}f\right\} = 2\pi i\nu \mathcal{F}\{f\}.$$

Теперь остается только выполнить обратное преобразование Фурье, чтобы получить спектральную производную сигнала. Далее приведены графики вещественной и мнимой компонент Фурье-образа сигнала и его спектральной производной.

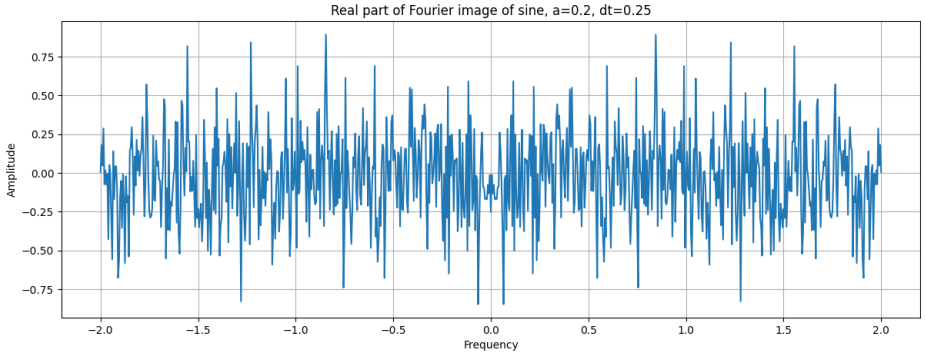

Рис. 3: Вещественная компонента Фурье-образа зашумленного сигнала.

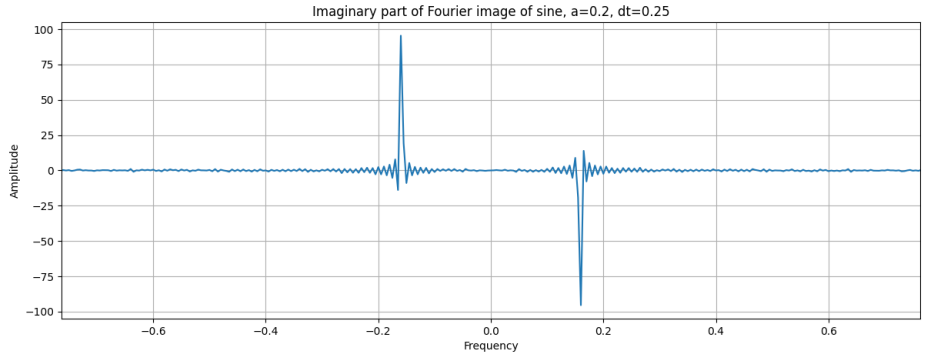

Рис. 4: Мнимая компонента Фурье-образа зашумленного сигнала.

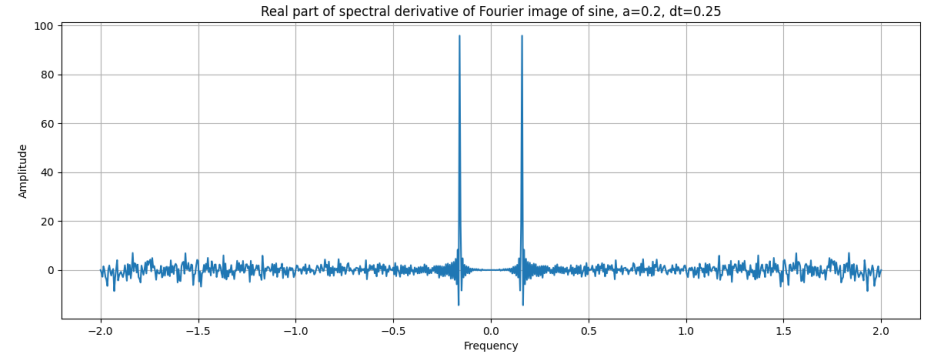

Рис. 5: Вещественная компонента спектральной производной Фурье-образа зашумленного сигнала.

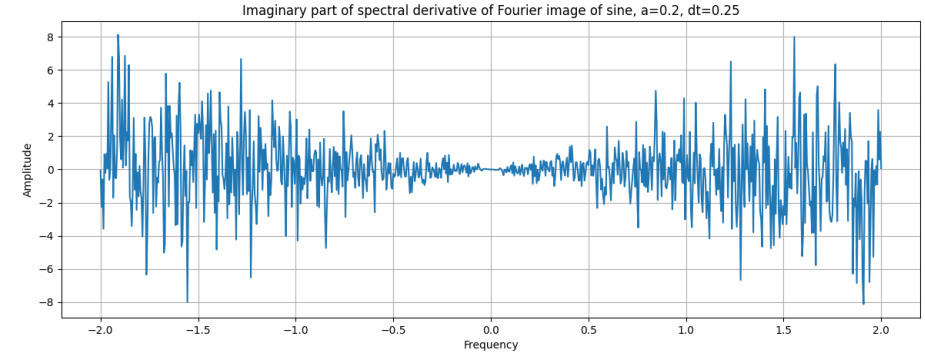

Рис. 6: Мнимая компонента спектральной производной Фурье-образа зашумленного сигнала.

Видим, что вещественная компонента Фурье-образа зашумленного сигнала и его спектральной производной симметричны относительно оси OY, а их мнимые компоненты относительно OX. Подобная симметричность сохраняется в не зависимости от четности исходной функции.

На следующем рисунке приведен график вещественной части спектральной производной зашумленного сигнала, найденной с помощью численного интегрирования. Результат похож на численную производную, но с резкими возрастаниями амплитуд по краям. Данное поведение не зависит от наличия шума в сигнале или выбора шага дискретизации dt.

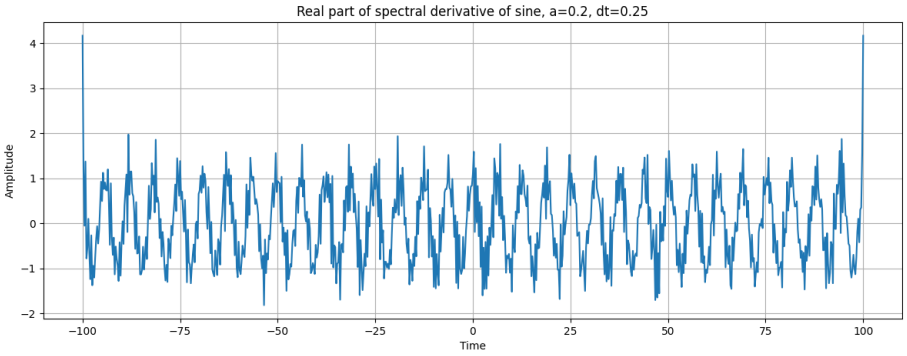

Рис. 7: Вещественная компонента спектральной производной зашумленного сигнала.

Теперь сравним график истинной производной $\cos(t)$ с графиками численной и спектральной производных зашумленного синуса. Оранжевым цветом обозначена спектральная производная, синим численная. Красным цветом выделена производная косинуса.

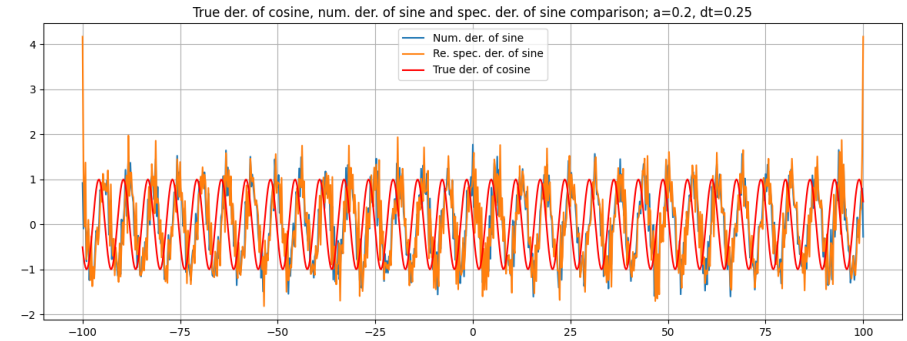

Рис. 8: Сравнительный график производной $\cos(t)$ с численной и спектральной производными зашумленного $\sin(t)$.

Графики численной и спектральной производных похожи друг на друга и на истинную производную косинуса (с разницей в смещении по фазе на $-\pi \div 2$). Спектральная производная имеет немного больше выбросов по сравнению с численной. Достаточно посмотреть на края графика спектральной производной и на ее амплитуды в точках максимума и минимума каждой волны.

В ходе работы было выяснено, что при маленьком шаге dt или при большом значении a спектральная и численная производные становятся не похожи на истинную производную косинуса. По большей части на графиках видно белый шум. Если исходный сигнал имеет шумы, то его производная будет зашумлена сильнее. Наличие мнимой части у Фурье-образа не зависит от четности исходной функции. Более того, вещественные компоненты

после преобразования Фурье симметричны относительно оси ординат, а мнимые относительно оси абсцисс.

1.1 Используемые программы

Все графики перового задания строились с помощью языка программирования Python с подключенной библиотекой matplotlib. Далее по ходу работы графики будут строиться той же программой.

```
import matplotlib.pyplot as plt

def build_f(x, y, fz1=16,
            fz2=5, clr=None, ttl=None,
            grid=True, legend=False, xlab=None,
            ylab=None, xl1=None, xl2=None,
            yl1=None, yl2=None, lbl=None,
            ls='-', ticks=None, rot=None):
    plt.plot(x, y, color=clr, label=lbl, linestyle=ls)
    plt.xlabel(xlab)
    plt.ylabel(ylab)
    plt.xlim(xl1, xl2)
    plt.ylim(yl1, yl2)
    plt.xticks(ticks, rotation=rot)
    plt.title(ttl)
    plt.gcf().set_size_inches(fz1, fz2)
    plt.grid(grid)
    if legend:
        plt.legend()
    plt.show()

def build_fs(x, y: list, colors: list=None,
             labels: list=None, fz1=16, fz2=5,
             ttl=None, grid=True, legend=False,
             xlab=None, ylab=None, xl1=None,
             xl2=None, yl1=None, yl2=None,
             ls:list=None, ticks=None, rot=None):
    if (y is None or len(y) <= 0):
        print('y is None or its len <= 0')
        return
    if (colors is None): colors = [None] * len(y)
    if (labels is None): labels = [None] * len(y)
    if (ls is None): ls = ['-'] * len(y)

    for k in range(len(y)):
        plt.plot(x, y[k], color=colors[k],
                 label=labels[k], linestyle=ls[k])
    plt.xlabel(xlab)
    plt.ylabel(ylab)
    plt.xlim(xl1, xl2)
    plt.ylim(yl1, yl2)
    plt.xticks(ticks, rotation=rot)
    plt.title(ttl)
    plt.gcf().set_size_inches(fz1, fz2)
    plt.grid(grid)
    if legend: plt.legend()
    plt.show()
```

Листинг 1: Файл с программой для построения графиков.

Для нахождения Фурье-образа и производных использовалась библиотека numpy.

```
import numpy as np

def trapz(y, t, v):
    Y = []
    for k in v:
        Y_k = np.trapz(y * np.exp(-1j * 2 * np.pi * k * t), t)
        Y.append(Y_k)
    return Y

def undo_trapz(Y, t, v):
    y = []
    for k in t:
        y_k = np.trapz(Y * np.exp(1j * 2 * np.pi * k * v), v)
        y.append(y_k)
    return y

def numerical_diff(y, dt):
    ndiff = []
    for k in range(len(y) - 1):
        ndiff_k = (y[k + 1] - y[k]) / dt
        ndiff.append(ndiff_k)
    return ndiff

def spectral_diff(y, t, v):
    Y = trapz(y, t, v)
    dY = 2 * np.pi * 1j * v * Y
    spdiff = undo_trapz(dY, t, v)
    return spdiff, Y, dY
```

Листинг 2: Программа для вычисления Фурье-образа численным интегрированием и производных.

Программа, где используются написанные функции и задаются необходимые параметры, расположена ниже.

```
import numpy as np

import build_func as bf
import fourier_math as fm

T = 200
dt = 0.25
t = np.arange(-T / 2, T / 2 + dt, dt)
y = np.sin(t)

a = 0.2
y += a * (np.random.rand(len(t)) - 0.5)

ndsin = fm.numerical_diff(y, dt)
ndsin.append(y[-1] / 2)

V = 1 / dt
dv = 1 / T
v = np.arange(-V / 2, V / 2 + dv, dv)
spdsin, Y, dY = fm.spectral_diff(y, t, v)
tdcos = -np.sin(t)

bf.build_f(t, y, ttl=f'Noisy sine, a={a}, dt={dt}',
           xlab='Time', ylab='Amplitude')
bf.build_f(v, np.array(Y).real,
```

```
26                ttl=f'Real part of Fourier image of sine, a={a}, dt={dt}',
27                xlab='Frequency', ylab='Amplitude')
28    bf.build_f(v, np.array(Y).imag,
29              ttl=f'Imaginary part of Fourier image of sine, a={a}, dt={dt}',
30              xlab='Frequency', ylab='Amplitude', xl1=-0.763, xl2=0.763)
31    bf.build_f(v, dY.real,
32                ttl=f'Real part of spectral derivative of Fourier image of
                      sine, a={a}, dt={dt}',
33                xlab='Frequency', ylab='Amplitude')
34    bf.build_f(v, dY.imag,
35                ttl=f'Imaginary part of spectral derivative of Fourier image
                      of sine, a={a}, dt={dt}',
36                xlab='Frequency', ylab='Amplitude')
37
38    bf.build_f(t, ndsin,
39                ttl=f'Numerical derivative of sine, a={a}, dt={dt}',
40                xlab='Time', ylab='Amplitude')
41    bf.build_f(t, np.array(spdsin).real,
42             ttl=f'Real part of spectral derivative of sine, a={a}, dt={dt}',
43             xlab='Time', ylab='Amplitude')
44    bf.build_fs(t, y=[ndsin, np.array(spdsin).real, tdcos],
45                colors=[None, None, 'r'], legend=True,
46                labels=['Num. der. of sine', 'Re. spec. der. of sine', 'True
                        der. of cosine'],
47                ttl=f'True der. of cosine, num. der. of sine and spec. der.
                      of sine comparison; a={a}, dt={dt}')
```