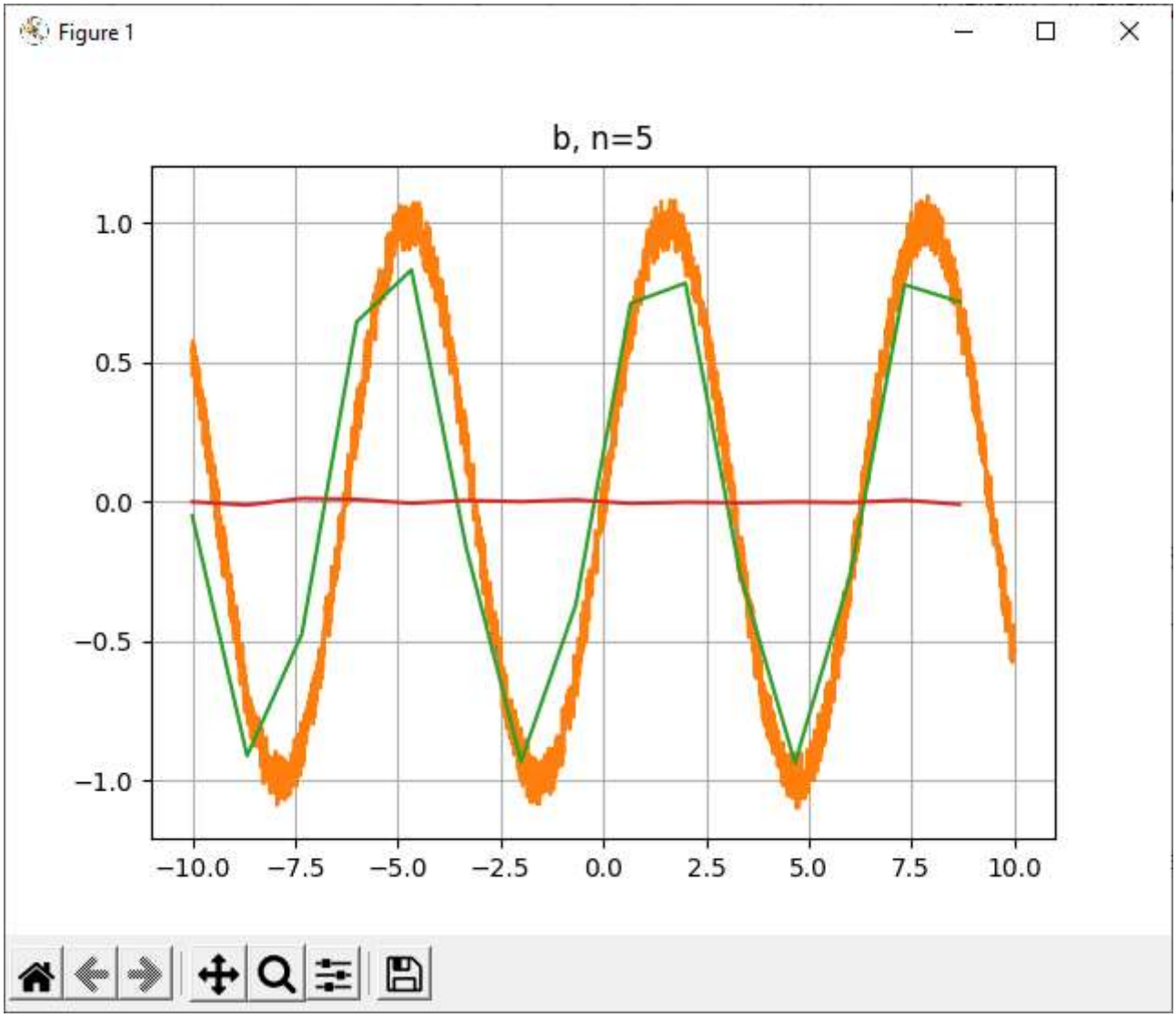
Задание

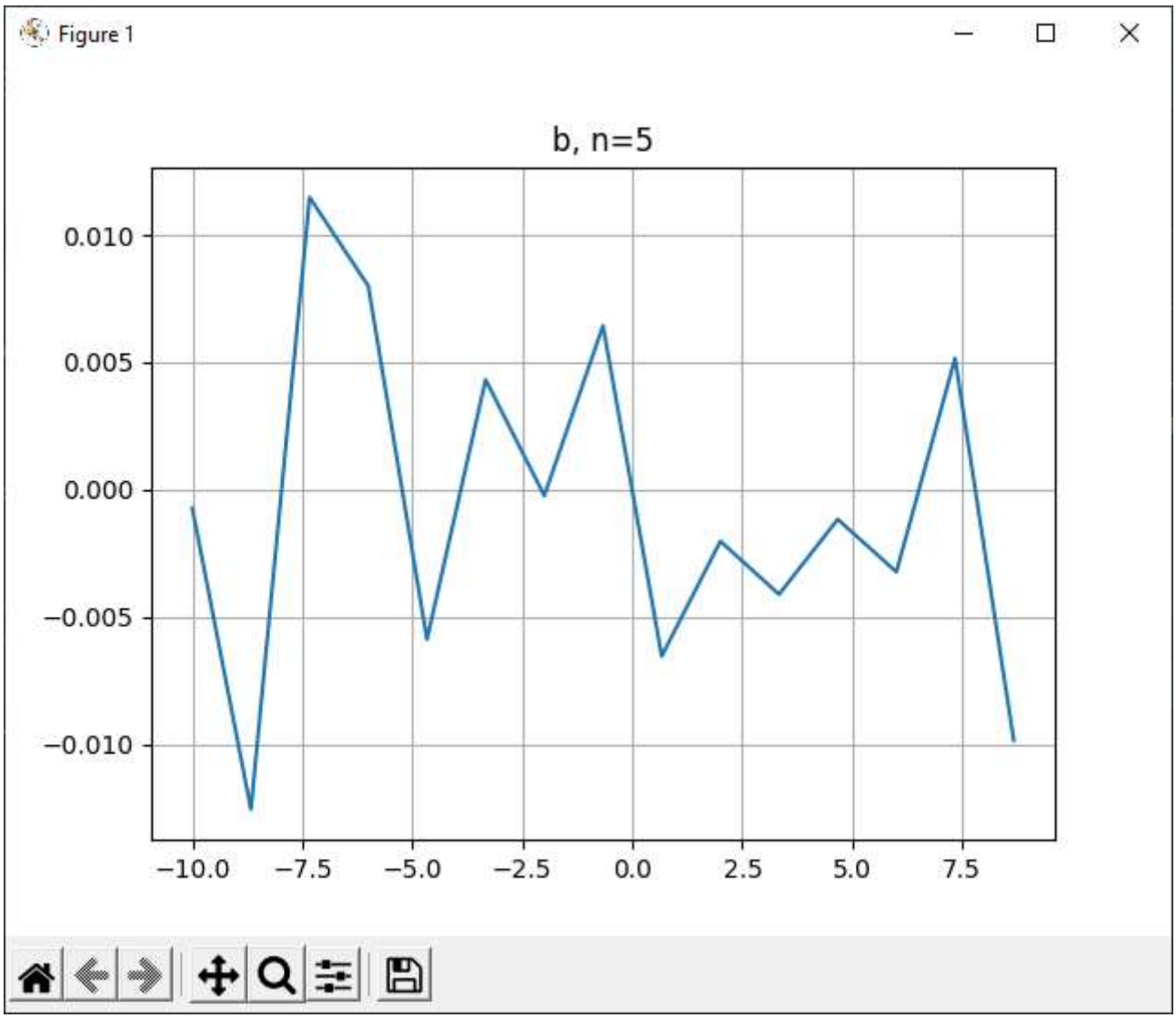
Отфильтровать вейвлетами Хаара функцию sin(x), применив пороговую фильтрацию. Степень порога выбрать произвольную.

Рассчитать

1. Квадратичное отклонение отфильтрованного сигнала от исходного и зашумленного.
2. Степень зашумленности полученных сигналов.

Ход работы





Отклонение от зашумленного: 0.802649852623777

Отклонение от исходного: 0.7926993640964444

Зашумленность: 0.006723773609410393

Вывод

В ходе выполнения работы вейвлеты Хаара были применены для сжатия и фильтрации функции с зашумлением.

Листинг

```
from copy import deepcopy
import random

import matplotlib.pyplot as plt
import numpy as np
import math

def haar(y_plot, n=1):
    a = []
    b = []

    l = int(len(y_plot)/2)
    for i in range(l):
        a.append((y_plot[2*i] + y_plot[2*i + 1])/2)
        b.append((y_plot[2*i] - y_plot[2*i + 1]) / 2)

    a_old = deepcopy(a)
    a_new = []
    b_old = deepcopy(b)
    b_new = []
    l = int(len(a_old) / 2)
    for i in range(l):
        a_new.append((a_old[2 * i] + a_old[2 * i + 1]) / 2)
        b_new.append((b_old[2 * i] - b_old[2 * i + 1]) / 2)
    for _ in range(n - 1):
        a_old = deepcopy(a_new)
        a_new = []
        b_old = deepcopy(b_new)
        b_new = []
        l = int(len(a_old) / 2)
        for i in range(l):
            a_new.append((a_old[2 * i] + a_old[2 * i + 1]) / 2)
            b_new.append((b_old[2 * i] - b_old[2 * i + 1]) / 2)
    a_new = np.array(a_new)
    b_new = np.array(b_new)
    return a_new, b_new

def simple_plot(x, y, title):
    plt.plot(x, y)
    plt.title(title)
    plt.grid(True)
    plt.show()

def function(x: int or float) -> int or float:
    return math.sin(x)

def deviation(y_fil, y_test):
    res = 0.
    for i in range(len(y_fil)):
        res += (y_fil[i] - y_test[2 * i]) ** 2 + \
               (y_fil[i] - y_test[2 * i + 1]) ** 2
    res /= 2 * len(y_fil)
    return np.sqrt(res)

def main():
    x_plot = np.arange(-10, 10, 0.01) # значения по x
    y_plot = np.array([function(x) for x in x_plot]) # значения по y
    simple_plot(x_plot, y_plot, "Sin (x)")

    random.seed(43)
```

```
    noise_plot = np.array([random.uniform(-0.1, 0.1) for _ in
range(len(y_plot))])
    y_nois = y_plot + noise_plot
    simple_plot(x_plot, y_nois, "Функция с шумом")

    a, b = haar(y_nois, n=6)

    simple_plot(np.arange(-10, 10, 20 / len(a)), a, "a, n=5")

    simple_plot(np.arange(-10, 10, 20 / len(b)), b, "b, n=5")

    print("Отклонение от зашумленного:", deviation(a, y_nois))
    print("Отклонение от исходного:", deviation(a, y_plot))

    s1 = 0.
    s2 = 0.
    for i in range(len(y_plot)):
        s1 += y_plot[i] ** 2
        s2 += (y_nois[i] - y_plot[i]) ** 2
    print("Зашумленность:", s2 / s1)

if __name__ == '__main__':
    main()
```