Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Инфокоммуникационные системы и технологии

Лабораторная работа №5

Выполнил:

Таякин Д.Р.

Проверил:

Мусаев А.А.

Санкт-Петербург, 2022

Задание

В первом пункте лабораторной работы нужно было реализовать сглаживание данных, используя динамическое скользящее окно. На рисунке 1 изображены модули, котрые используются в основном файле main.py. В util реализованы полезные функции такие как МНК, динамическое скользящее окно, СКО. Рисунок 2 показывает содержание модуля util.

```
from math import sin, cos, ceil
from util import *
import matplotlib.pyplot as plt
import numpy as np
import openpyxl
```

Рисунок 1 – Используемые модули

```
import numpy as np

# deviation (sigma)
def std_deviation(orig_data: list, data: list):
    n = len(data)
    summ = 0
    for i in range(n):
        summ += (data[i] - orig_data[i])**2
    return ((1/n) * summ) ** 0.5

def MAW(Y, k=2):
    n = len(Y)
    temp_Y = Y[0:k]
    for t in range(k, n):
        slice_Y = Y[t-k:t]
        temp_Y.append(sum(slice_Y) / len(slice_Y))
    return temp_Y

def mnk(x: list, y: list):
    n = len(x)

    sum_x = sum(x); sum_y=sum(y)
    sum_xy = sum(map(lambda x, y: x * y, x, y))
    sum_x2 = sum(map(lambda x: x ** 2, x))

    det_m = np.linalg.det(np.matrix([[sum_x2, sum_x], [sum_x, n]]))
    det_a = np.linalg.det(np.matrix([[sum_xy, sum_x], [sum_y, n]]))
    det_b = np.linalg.det(np.matrix([[sum_x2, sum_xy], [sum_x, sum_y]]))

    a = det_a / det_m
    b = det_b / det_m

    return a, b
```

Рисунок 2 – Содержание модуля util

```python
def create_excel_file(x: list, y: list):
    wb = openpyxl.Workbook()
    sheet = wb.active

    sheet.cell(1, 1, "x")
    sheet.cell(1, 2, "y")

    for i, (coord_x, coord_y) in enumerate(zip(x, y)):
        sheet.cell(2+i, 1, coord_x)
        sheet.cell(2+i, 2, coord_y)

    wb.save("data.xlsx")

def default_f(x):
    # return cos(x) + 0.1*cos(x**5)
    # return cos(x**2) + x**2
    return sin(x) + 0.1*sin(x**5)

# forecasts new value with linear approximation
# returns prognosis x and y lists
def prognosis_data(step, x_data: list, y_data: list):
    x_data_copy = x_data.copy(); y_data_copy = y_data.copy()

    a, b = mnk([x_data[-2], x_data[-1]], [y_data[-2], y_data[-1]])

    new_x = x_data[-1] + step
    new_y = a*new_x + b

    x_data_copy.append(new_x)
    y_data_copy.append(new_y)

    return x_data_copy, y_data_copy
```

Рисунок 3 – Первая часть кода из main.py

На рисунке 3 изображены несколько функций. Функия create_excel_file, используя модуль openpyxl, предназначена для создания таблицы. default_f – задает значения функции для значений x. Две дополнительные функции закомментированы. В prognosis_data реализован алгоритм прогнозирования на один шаг вперед, используя линейную аппроксимацию. В качестве возвращаемых значений – списки значений x и y.

На рисунке 4 изображена часть кода, использающая все вышеперечисленные функции и алгоритмы. Сначала, пользователь задает все необходимые входные данные: начальное, конечное значения и шаг. Затем,

```python
if __name__ == "__main__":
    print("Enter input data")
    start = int(input("start: "))
    stop  = int(input("stop: "))
    step  = float(input("step: "))

    x_data = list(np.arange(start, stop+step, step))
    y_data = [default_f(x) for x in x_data]

    create_excel_file(x_data, y_data)

    smooth_y_data = MAW(y_data)

    sig = std_deviation(y_data, smooth_y_data)
    smooth_y_data = MAW(y_data, k=max(2, ceil(sig)))

    # a, b = mnk(x_data, smooth_y_data)
    # print(a, b)

    new_x_data, new_y_data = prognosis_data(step, x_data, y_data)
    new_smth_x_data, new_smth_y_data = prognosis_data(step, x_data, smooth_y_data)

    print(new_y_data)
    print(new_smth_y_data)

    # show what we've done
    fig, axs = plt.subplots(1)
    axs.set_title("Data")
    axs.plot(x_data, y_data, color="#FE7A60", label="Initial")
    # prognosis point
    axs.scatter(new_x_data[-1], new_y_data[-1], color="#FE7A60")
    axs.plot(x_data, smooth_y_data, color="#77D5FF", label="Antialised")
    # prognosis point
    axs.scatter(new_x_data[-1], new_smth_y_data[-1], color="#77D5FF")
    axs.legend()

    plt.text(0.8, 1, f"σ = {sig}")

    fig.tight_layout()
    plt.show()
```

Рисунок 4 – Вторая часть кода из main.py

формируются списки из данных и создается excel файл. Далее, применяется алгоритм сглаживания данных при помощи динамического скользящего окна. После этого, создаются прогнозируемые данные для сглаженных и исходных данных. Потом строются графики.

На рисунках 5, 6 и 7 приведены примеры получившихся данных, где start = 1, stop = 10, step = 0.1. На первом графике исходная функция $f(x) = sin(x) + 0.1 * sin(x^5)$. На втором: $f(x) = cos(x^2) + x^2$. На третьем: $f(x) = cos(x) + 0.1 * cos(x^5)$. Точки на графиках обозначают прогнозируемые значения.

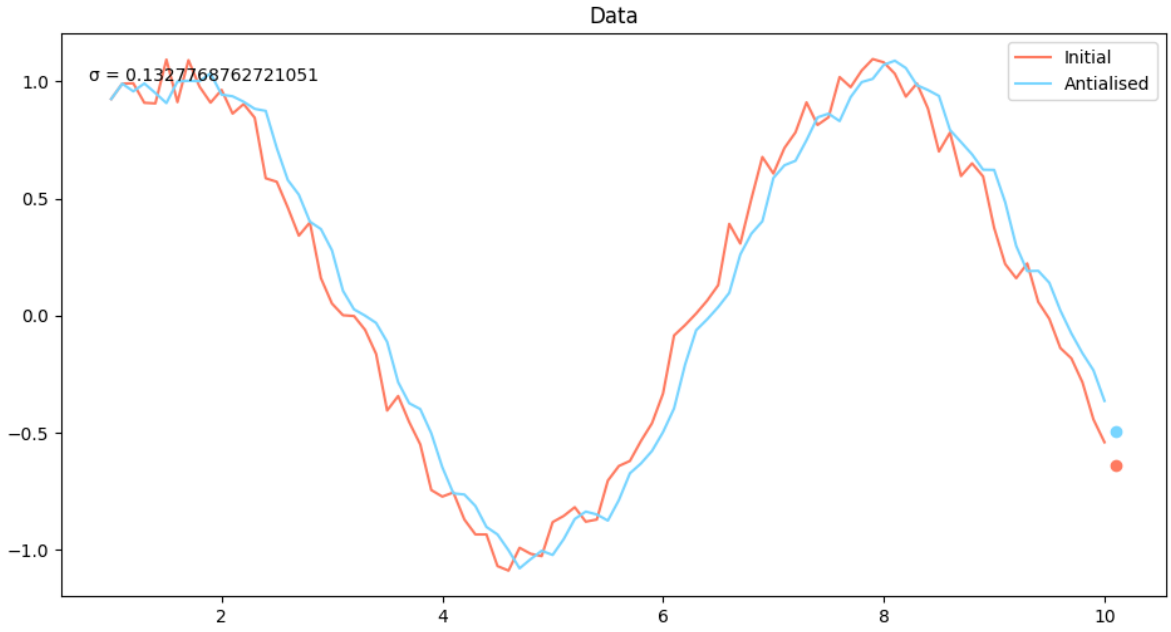

Рисунок 5 – Получившийся график при первых исходных данных

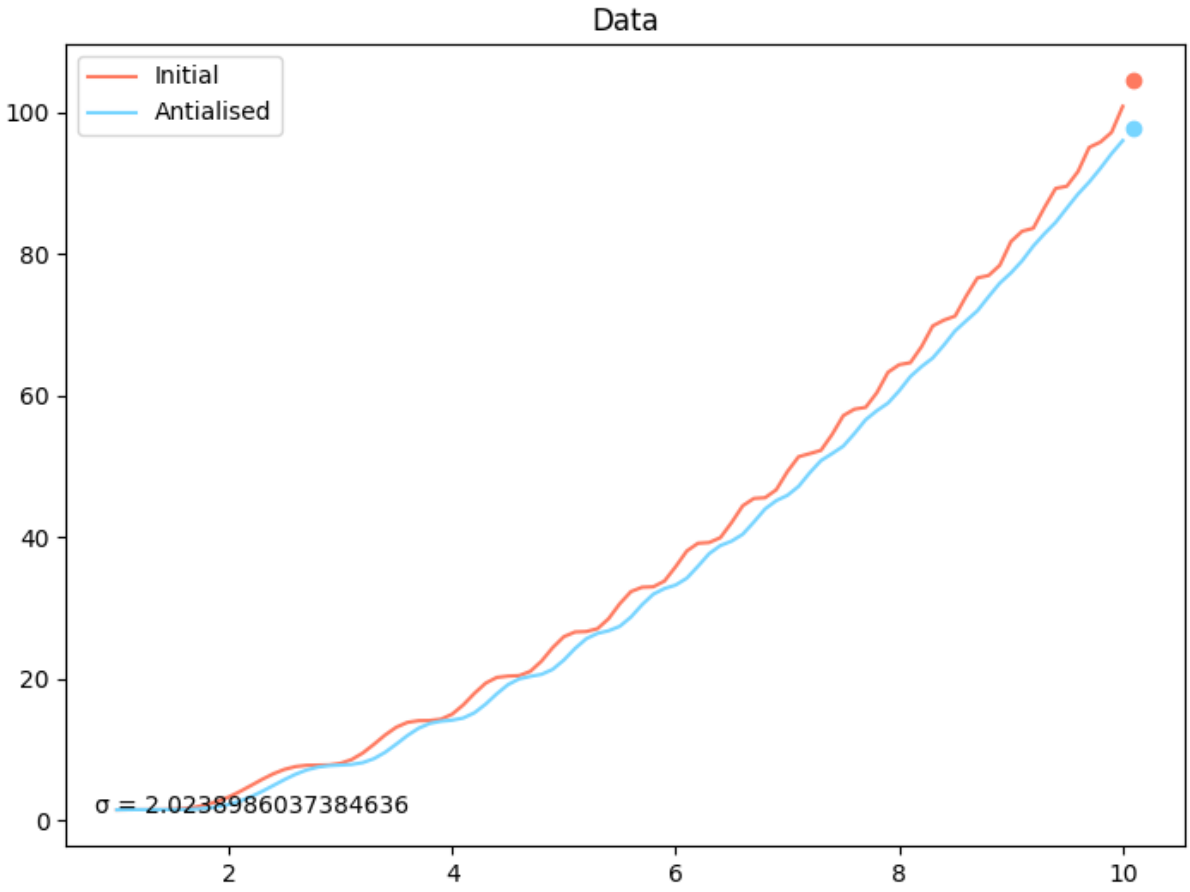

Рисунок 6 – Получившийся график при вторых исходных данных

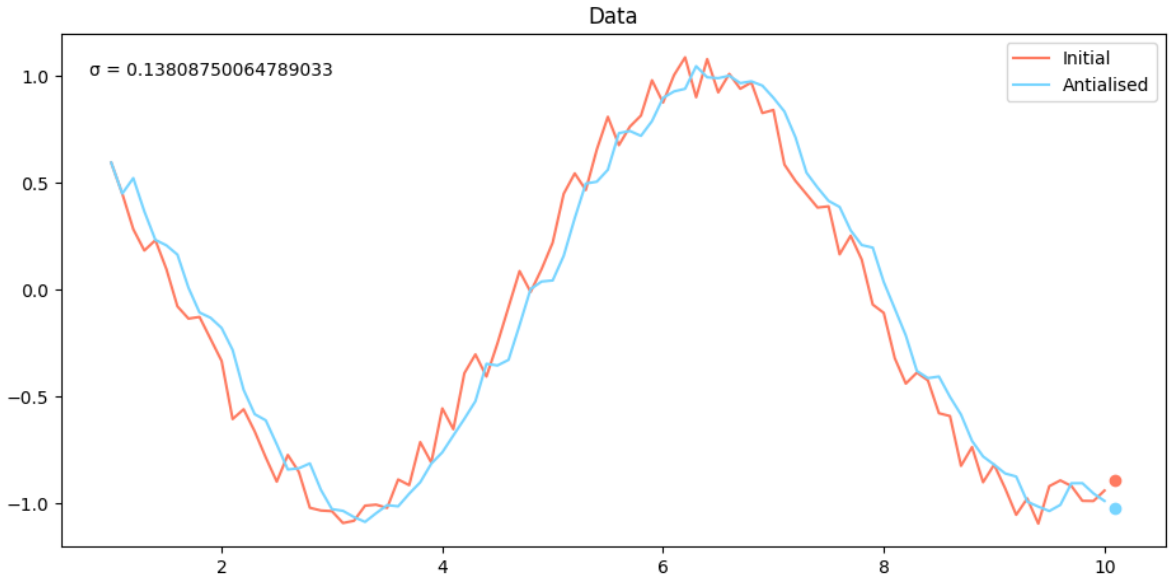

Рисунок 7 – Получившийся график при третьих исходных данных

В лабораторной работе №5 были применены алгоритмы сглаживания данных, используя динамическое скользящее окно. Также было проведено прогнозирование и сравнение между исходными данными и сглаженными посредством нахождения среднего квадратичного отклонения (СКО). В полученных данных СКО получилось маленьким.