Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

ОТЧЕТ

по лабораторной работе № 4

Название: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения

Дисциплина: Вычислительные алгоритмы

Студент ИУ7И - 46Б (Группа) __________ (Подпись, дата) Андрич К. (И.О. Фамилия)

Преподаватель __________ (Подпись, дата) В.М. Градов (И.О. Фамилия)

Москва, 2021

Цель работы

Получение навыков построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами

Исходные данные

1. Таблица функции с весами i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек. Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице
2. Степень аппроксимирующего полинома - n

Описание алгоритма

1. Возьмем степень полинома $n < N$
2. Составим СЛАУ вида

$$\sum_{m=0}^{n} (x^k, x^m)\, a_m = (y, x^k)\ ,\ 0 \le k \le n$$

$$(x^k, x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}, \quad (y, x^k) = \sum_{i=1}^{N} \rho_i y_i x_i^k$$

3. В решении СЛАУ находится коэффициенты полинома

Код программы

В программе есть 3 файлов: main.py, functions.py и result.py

main.py

```
from functions import *

def main():
    tbl = make_table()
    option = 2
    while (option != 0):
        choice(option, tbl)
        menu()
        try:
            option = int(input("\nInput choice: "))
        except:
            print("Invalid input")
            continue

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

functions.py

```
from random import random
from result import *


def make_table():
    tbl = []
    for i in range(SIZE):
        tbl.append([i + 1, random() * 5, 1])
    return tbl

def table_print(tbl):
    print("Table:")
    for i in range(SIZE):
        print("%-5.2f    |    %-5.2f     |    %-5.2f    " %(tbl[i][0], tbl[i][1],
tbl[i][2]))
    print("\n")

def change_weight(tbl):
    try:
        indx = int(input("\nInput row number: "))
    except:
        print("Invalid input")
        return tbl
    if (indx < 1 or indx > SIZE):
        print("The row you chose doesn't exist")
        return tbl
    try:
        weight = int(input("\nInput weight: "))
    except:
        print("Invalid input")
        return tbl
    tbl[indx - 1][2] = weight
    return tbl

def choice(argument, tbl):
    if argument == 1:
        result(tbl)
    elif argument == 2:
        table_print(tbl)
    if argument == 3:
        change_weight(tbl)

def menu():
    print("1. See the results")
    print("2. Print table")
    print("3. Change weight")
    print("0. Exit")
```

result.py

```
import matplotlib.pyplot as plt

EPS = 0.001
SIZE = 10

def check_for_changes(tbl):
    for i in range(SIZE):
        if not (tbl[i][2] == 1):
            return True
    return False

def mtx_model(i):
    mtx = []
    for j in range(i + 1):
        row = []
        for k in range(i + 2):
            row.append(0)
        mtx.append(row)
    return mtx

def slae_mtx(tbl, i):
    mtx = mtx_model(i)
    for j in range(i + 1):
        for k in range(i + 1):
            mtx[j][k] = 0
            mtx[j][i + 1] = 0
            for l in range(SIZE):
                weight = tbl[l][2]
                x, y = tbl[l][0], tbl[l][1]
                mtx[j][k] += weight * pow(x, (j + k))
                mtx[j][i + 1] += weight * y * pow(x, j)
    return mtx

def gauss(mtx):
    res = []
    mxlen = len(mtx)
    for i in range(SIZE):
        for j in range(i + 1, mxlen):
            if (i == j):
                continue
            k = mtx[j][i] / mtx[i][i]
            for l in range(i, mxlen + 1):
                mtx[j][l] -= k * mtx[i][l]
    for i in range(mxlen):
        res.append(0)
    k = -1
    for i in range(mxlen - 1, k, -1):
        for j in range(mxlen - 1, i, -1):
            mtx[i][mxlen] -= res[j] * mtx[i][j]
        res[i] = mtx[i][mxlen] / mtx[i][i]
```

```
    return res

def get_points(tbl, i):
    mtx = slae_mtx(tbl, i)
    res = gauss(mtx)
    X, Y = [], []
    x = tbl[0][0] - EPS
    while (x <= tbl[SIZE - 1][0] + EPS):
        y = 0
        for j in range(0, i + 1):
            y += res[j] * pow(x, j)
        X.append(x)
        Y.append(y)
        x += EPS
    return X, Y

def plot(tbl, n, points):
    for i in range(1, n + 1):
        if (i > 2 and i < n):
            continue
        X, Y = get_points(tbl, i)
        plt.plot(X, Y, "-", label = "%s\nn = %d" %(points, i))

def get_original(tbl):
    new = []
    for i in range(SIZE):
        new.append([tbl[i][0], tbl[i][1], 1])
    return new

def draw_points(tbl):
    X, Y = [], []
    for i in range(SIZE):
        X.append(tbl[i][0])
        Y.append(tbl[i][1])
    return X, Y

def result(tbl):
    try:
        n = int(input("\nEnter the degree of the approximating polynomial: "))
    except:
        print("Invalid input")
        return tbl
    if n <= 0 or n >= SIZE:
        print("Invalid degree input \n")
        return tbl
    change_indx = check_for_changes(tbl)
    if change_indx:
        points = "Different weights"
    else:
        points = "Same weights"
    plot(tbl, n, points)
```

```
    if change_indx:
        points = "Same weights"
        same = get_original(tbl)
        plot(same, n, points)
    PX, PY = draw_points(tbl)
    plt.plot(PX, PY, 'o', label = "points")
    plt.legend()
    plt.grid()
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

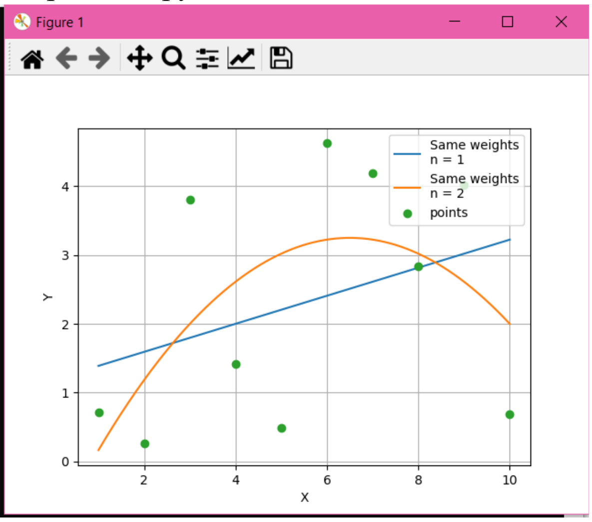
Результаты работы

Интерфейс:

```
Table:
1.00    |   0.72     |   1.00
2.00    |   0.26     |   1.00
3.00    |   3.81     |   1.00
4.00    |   1.42     |   1.00
5.00    |   0.49     |   1.00
6.00    |   4.62     |   1.00
7.00    |   4.19     |   1.00
8.00    |   2.83     |   1.00
9.00    |   4.02     |   1.00
10.00   |   0.69     |   1.00


1. See the results
2. Print table
3. Change weight
0. Exit

Input choice: _
```

Если не изменаем вес (степень аппроксимирующего полинома 2):

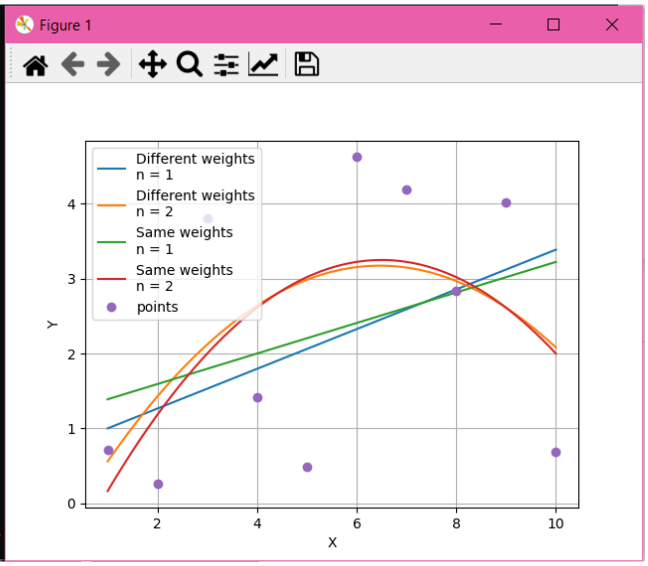
```
Table:
1.00    |   0.72    |   1.00
2.00    |   0.26    |   1.00
3.00    |   3.81    |   1.00
4.00    |   1.42    |   1.00
5.00    |   0.49    |   1.00
6.00    |   4.62    |   1.00
7.00    |   4.19    |   1.00
8.00    |   2.83    |   1.00
9.00    |   4.02    |   1.00
10.00   |   0.69    |   1.00


1. See the results
2. Print table
3. Change weight
0. Exit

Input choice: 1

Enter the degree of the approximating polynomial: 2
```



Если изменаем вес (степень аппроксимирующего полинома 2):

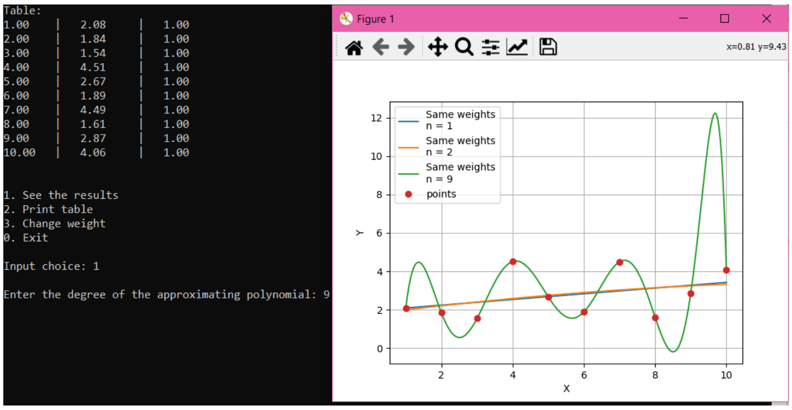
```
1. See the results
2. Print table
3. Change weight
0. Exit

Input choice: 3

Input row number: 1

Input weight: 5
```

```
1. See the results
2. Print table
3. Change weight
0. Exit

Input choice: 2
Table:
1.00    |   0.72    |   5.00
2.00    |   0.26    |   1.00
3.00    |   3.81    |   1.00
4.00    |   1.42    |   1.00
5.00    |   0.49    |   1.00
6.00    |   4.62    |   1.00
7.00    |   4.19    |   1.00
8.00    |   2.83    |   1.00
9.00    |   4.02    |   1.00
10.00   |   0.69    |   1.00

1. See the results
2. Print table
3. Change weight
0. Exit

Input choice: 1

Enter the degree of the approximating polynomial: 2
```

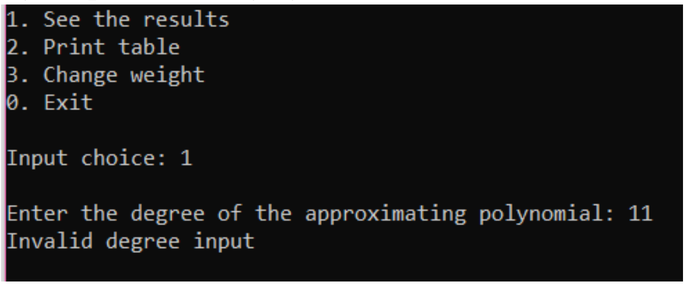


Вопросы при защите лабораторной работы

1. *Что произойдет при задании степени полинома n=N-1 (числу узлов таблицы минус 1)?*
   Функция пройдет по всем заданным точкам



2. *Будет ли работать Ваша программа при $n \geq N$? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?*
   В этом случайе, программа выведет сообщение об ошибке. Так будет, из-за того что при $n \geq N$ определитель будет равен 0 (слау будет линейно-зависима).



3. *Получить формулу для коэффициента полинома a_0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?*
   математическое ожидание

$$\frac{\sum_{i=1}^{N} y_i \rho_i}{\sum_{i=1}^{N} \rho_i}$$

4. *Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все* $\rho_i = 1$.

$$\begin{cases} a_0 + (x_0 + x_1)a_1 + (x_0^2 + x_1^2)a_2 = y_0 + y_1 \\ (x_0 + x_1)a_0 + (x_0^2 + x_1^2)a_1 + (x_0^3 + x_1^3)a_2 = y_0 x_0 + y_1 x_1 \\ (x_0^2 + x_1^2)a_0 + (x_0^3 + x_1^3)a_1 + (x_0^4 + x_1^4)a_2 = y_0 x_0^2 + y_1 x_0^2 \end{cases}$$

$$\Delta = (x_0^2 + x_1^2)(x_0^4 + x_1^4) + (x_0 + x_1)(x_0^3 + x_1^3)(x_0^2 + x_1^2) + (x_0^2 + x_1^2)(x_0 + x_1)(x_0^3 + x_1^3) - (x_0^2 + x_1^2)(x_0^2 + x_1^2)(x_0^2 + x_1^2) - (x_0^3 + x_1^3)(x_0^3 + x_1^3) - (x_0 + x_1)(x_0 + x_1)(x_0^4 + x_1^4) = 0$$

Из-за того что $\Delta = 0$, ситема не имеет решение

5. *Построить СЛАУ при выборочном задании степеней аргумента полинома* $\varphi(x) = a_0 + a_1 x^m + a_2 x^n$, *причем степени n и m в этой формуле известны.*

$$\begin{cases} (x^0, x^0)a_0 + (x^0, x^m)a_1 + (x^0, x^n)a_2 = (y, x^0) \\ (x^m, x^0)a_0 + (x^m, x^m)a_1 + (x^m, x^n)a_2 = (y, x^m) \\ (x^n, x^0)a_0 + (x^n, x^m)a_1 + (x^n, x^n)a_2 = (y, x^n) \end{cases}$$

6. *Предложить схему алгоритма решения задачи из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами ak , т.е. количество неизвестных равно 5*

   Сначала сделаем перебор всех степеней n и m которые возможны. Затем, для каждый пар степени найдем коэффициент a. Наконец, выберем пар, у которого значение найболе блиско эталонному.