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»

Лабораторная работа № 4
по дисциплине "Вычислительные алгоритмы"

Тема Алгоритм наилучшего среднеквадратичного приближения.

Студент Мицевич М. Д.

Группа ИУ7-41Б

Оценка (баллы) ________________

Преподаватель Градов В.М.

Москва — 2021 г.

1. Цель работы

Получение навыков построения алгоритмаметода наименьших квадратовс использованием полинома заданной степени при аппроксимации табличных функций с весами.

2. Исходные данные

1) Таблица функции с весами ρ_i и 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.

3. Код программы

Листинг main.py:

```
import matplotlib.pyplot as plt
from squares_method import squares_method

def plot_graphs(table, pows):
    plt.figure(1)

    plt.scatter([dot[0] for dot in table], [dot[1] for dot in table], c = 'black
        ', label = "data")

    for power in pows:
        dots = squares_method(table, power)
        plt.plot([dot[0] for dot in dots], [dot[1] for dot in dots], label = f"
            power = {power}")

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid(True)
    plt.legend(loc = 0)

    plt.show()

def print_table(table):
    print("|      |  X  |  Y  |  P  |")
    for i in range(len(table)):
        print(f"|{i : 5}|{table[i][0] : 5}|{table[i][1] : 5}|{table[i][2] : 5}|"
            )

def change_point_weight(table):
    i = int(input("Input point number: "))
    weight = int(input("Input new point weight"))

    table[i][2] = weight

def main():
    table = [[0, 1, 1],
             [1, 1.5, 1],
             [2, 2, 1],
             [3, 0, 1],
             [4, -1, 1],
             [5, -2, 1],
             [6, -0.5, 1],
             [7, 1, 1],
             [8, 4, 1],
             [9, 7, 1]]
```

```
    choise = 10
    while (choise):
        print("1 - Print table\n" +
              "2 - Change point weight\n" +
              "3 - Plot graphs\n" +
              "0 - Exit")

        choise = int(input("Your choise"))

        if choise == 1:
            print_table(table)
        elif choise == 2:
            change_point_weight(table)
        elif choise == 3:
            plot_graphs(table, [1, 2, 5, 6])



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

Листинг squares_method.py:

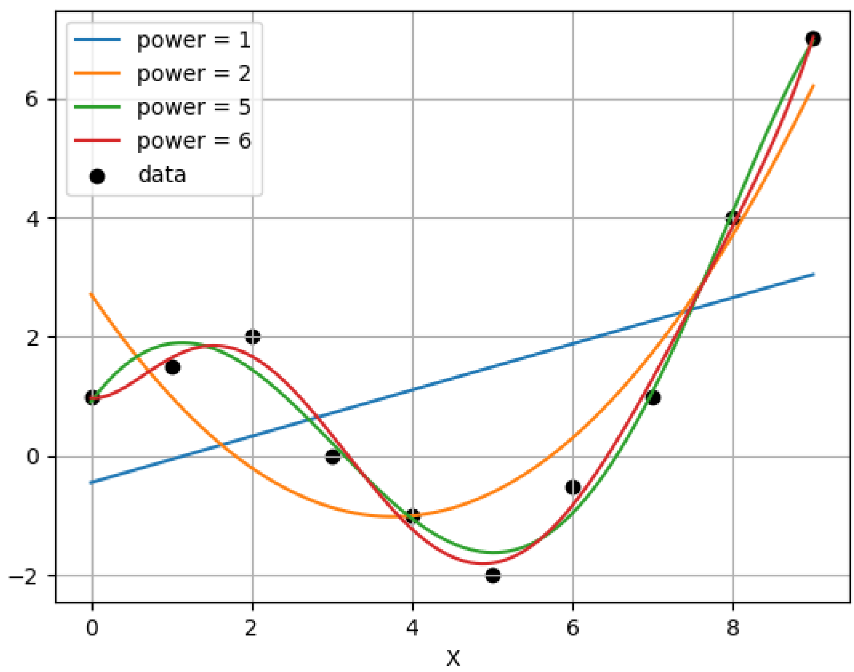
```
def xx_scalar(table, pow1, pow2):
    res = 0
    for i in range(len(table)):
        res += table[i][2] * (table[i][0] ** (pow1 + pow2))

    return res

def yx_scalar(table, y_pow, x_pow):
    res = 0
    for i in range(len(table)):
        res += table[i][2] * (table[i][1] ** y_pow) * (table[i][0] ** x_pow)

    return res

def gen_slau(table, n):
    slau = []

    for k in range(n + 1):
        row = []
        for m in range(n + 1):
            row.append(xx_scalar(table, k, m))
        row.append(yx_scalar(table, 1, k))
        slau.append(row)

    return slau

def solve_slau(slau):
    for i in range(len(slau)):
        tmp = slau[i][i]
        for j in range(len(slau[0])):
            slau[i][j] /= tmp

        for j in range(i + 1, len(slau)):
            tmp = slau[j][i]
            for k in range(len(slau[0])):
                slau[j][k] -= slau[i][k] * tmp

    coefs = []

    for i in range(len(slau) - 1, -1, -1):
        coef = slau[i][len(slau[0]) - 1]

        for j in range(len(coefs)):
```

```
            coef -= coefs[j] * slau[i][i + j + 1]

        coefs.insert(0, coef)

    return coefs

def find_dots(coefs, start, end, step):
    dots = []

    x = start

    while x <= end:
        y = 0
        for i in range(len(coefs)):
            y += coefs[i] * (x ** i)

        dots.append((x, y))

        x += step

    return dots



def squares_method(table, n):
    slau = gen_slau(table, n)

    coefs = solve_slau(slau)

    return find_dots(coefs, table[0][0], table[len(table) - 1][0], 0.01)
```

4. Результаты

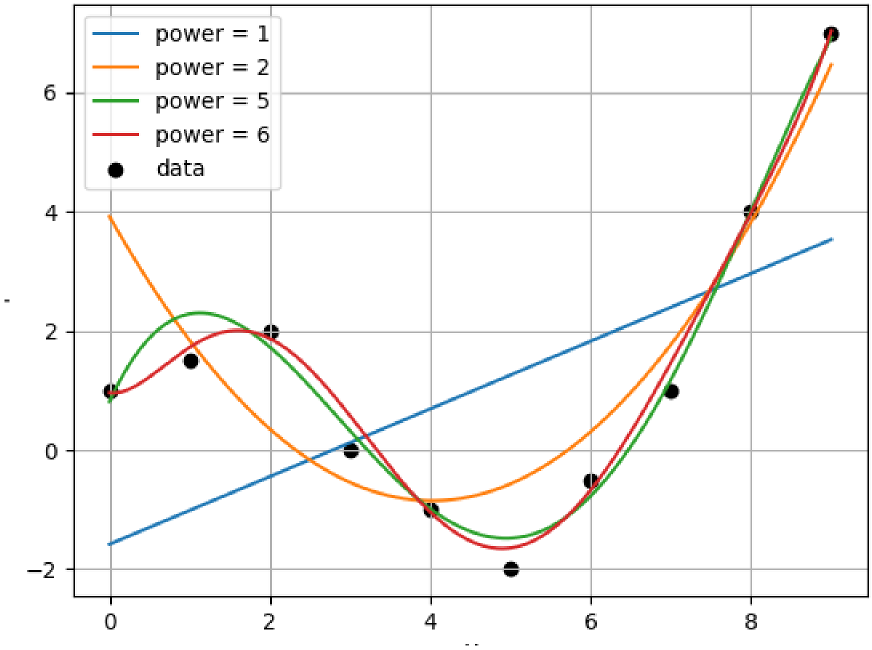
4.1. Веса всех точек одинаковы

```
1 - Print table
2 - Change point weight
3 - Plot graphs
0 - Exit
Your choise: 1
|   №   |   X   |   Y   |   P   |
|      0|      0|      1|      1|
|      1|      1|    1.5|      1|
|      2|      2|      2|      1|
|      3|      3|      0|      1|
|      4|      4|     -1|      1|
|      5|      5|     -2|      1|
|      6|      6|   -0.5|      1|
|      7|      7|      1|      1|
|      8|      8|      4|      1|
|      9|      9|      7|      1|
```



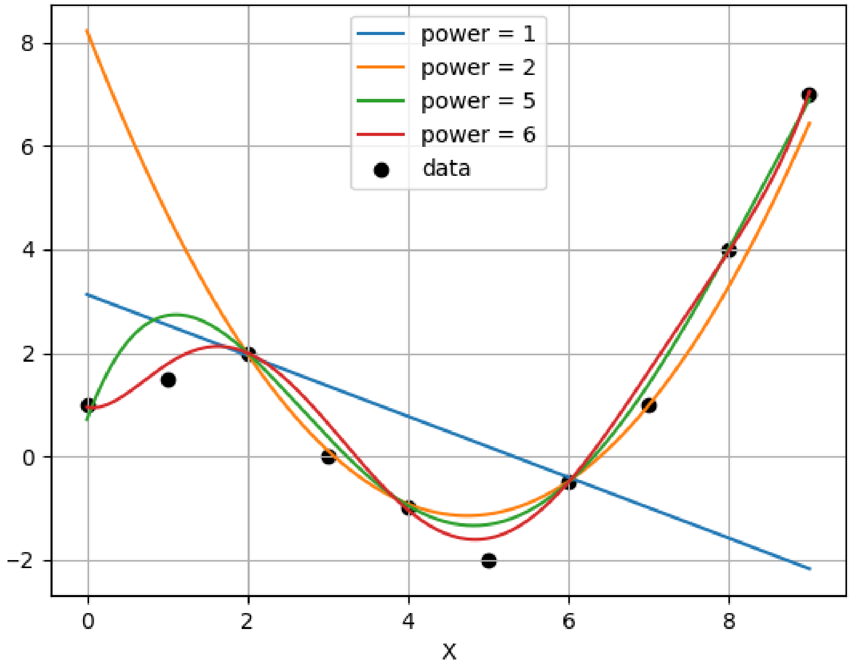
4.2. Веса точек разные

№	X	Y	P
0	0	1	1
1	1	1.5	1
2	2	2	4
3	3	0	1
4	4	-1	10
5	5	-2	1
6	6	-0.5	3
7	7	1	1
8	8	4	10
9	9	7	1



4.3. Изменение угла наклона прямой

№	X	Y	P
0	0	1	1
1	1	1.5	1
2	2	2	999
3	3	0	1
4	4	-1	10
5	5	-2	1
6	6	-0.5	999
7	7	1	1
8	8	4	10
9	9	7	1



5. Контрольные вопросы

5.1. Что произойдет при задании степени полинома n=N-1 (числу узлов таблицы минус 1)?

График полинома пройдет через все точки вне зависимости от их веса

5.2. Будет ли работать Ваша программа при $n >= N$ Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?

Результат работы программы будет неверным. Аварийная ситуация может произойти при делении на ноль в решении СЛАУ

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

$a_0 = \frac{\sum_{i=1}^{N} \rho_i * y_i}{\sum_{i=1}^{N} \rho_i}$

Данный коэффициент является взвешенным средним арифметическим ординат функции

5.4. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все ρ_i =1.

$\begin{vmatrix} 2 & x_1 + x_2 & x_1^2 + x_2^2 \\ x_1 + x_2 & x_1^2 + x_2^2 & x_1^3 + x_2^3 \\ x_1^2 + x_2^2 & x_1^3 + x_2^3 & x_1^4 + x_2^4 \end{vmatrix}$ Определитель равен 0, поэтому система не имеет решения

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

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

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

Эту систему можно решить перебором всех допустимых значений n и m. Для каждой пары значений нужно найти коэффициенты и значение ошибки, после выбираем пару с наименьшей ошибкой.