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

Тема: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

Студент: Артемьев И.О.

Группа: ИУ7-43Б

Оценка (баллы): _______________

Преподаватель: Градов В.М.

Москва
2021 г

Цель работы

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

Исходные данные

1) Степень аппроксимирующего полинома - n
2) Таблица функции с количеством узлов N с весами ρ_i, которая формируется случайным образом при каждом запуске программы

Выходные данные

Графики, на которых построены точки (по таблице) и кривые (найденные полиномы).

1) Веса одинаковые: Полиномы степеней 1, 2 и введенной
2) Веса разные: Полиномы степеней 1, 2 и введенной (такой же набор с одинаковыми весами для сравнения)

Анализ алгоритма

1) Выбирается степень полинома n << N (размера таблицы)
2) Составляется СЛАУ следующим образом:

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

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

3) Получившаяся система решается методом Гаусса, в результате чего получаются коэффициенты полинома

Код программы

```
"""
    main.py
"""

from square_approx import RootMeanSquareApproximation


def main():
    app = RootMeanSquareApproximation()

    while True:
        app.print_menu()

        try:
            option = int(input("\n\nInput the menu command: "))
        except:
            print("\n\nInvalid command\n\n")
            continue

        if option == 0:
            break
        elif option == 1:
            app.print_table()
        elif option == 2:
            app.change_weight()
        elif option == 3:
            app.draw()


if __name__ == "__main__":
    main()

"""
    square_approx.py
"""

from random import randint
```

```python
import matplotlib.pyplot as plt
from copy import deepcopy


class RootMeanSquareApproximation:
    def __init__(self):
        self.table = self.__gen_table()
        self.__flag_changed = False

    @staticmethod
    def print_menu():
        print(
            "\n\nMenu\n\
        \n1. Print the table\
        \n2. Change the point weight\
        \n3. Print the results\
        \n0. Exit"
        )

    @staticmethod
    def __gen_table(size=7, default_weight=1):

        table = [
            [randint(1.0, 100.0), randint(1.0, 100.0), default_weight]
            for _ in range(size)
        ]

        table.sort()

        return table

    @staticmethod
    def get_slau_matrix(table, power):
        size = len(table)

        matrix = [[0 for i in range(power + 2)] for i in range(power + 1)]

        for i in range(power + 1):
            for j in range(power + 1):
                a_coeff = 0.0
```

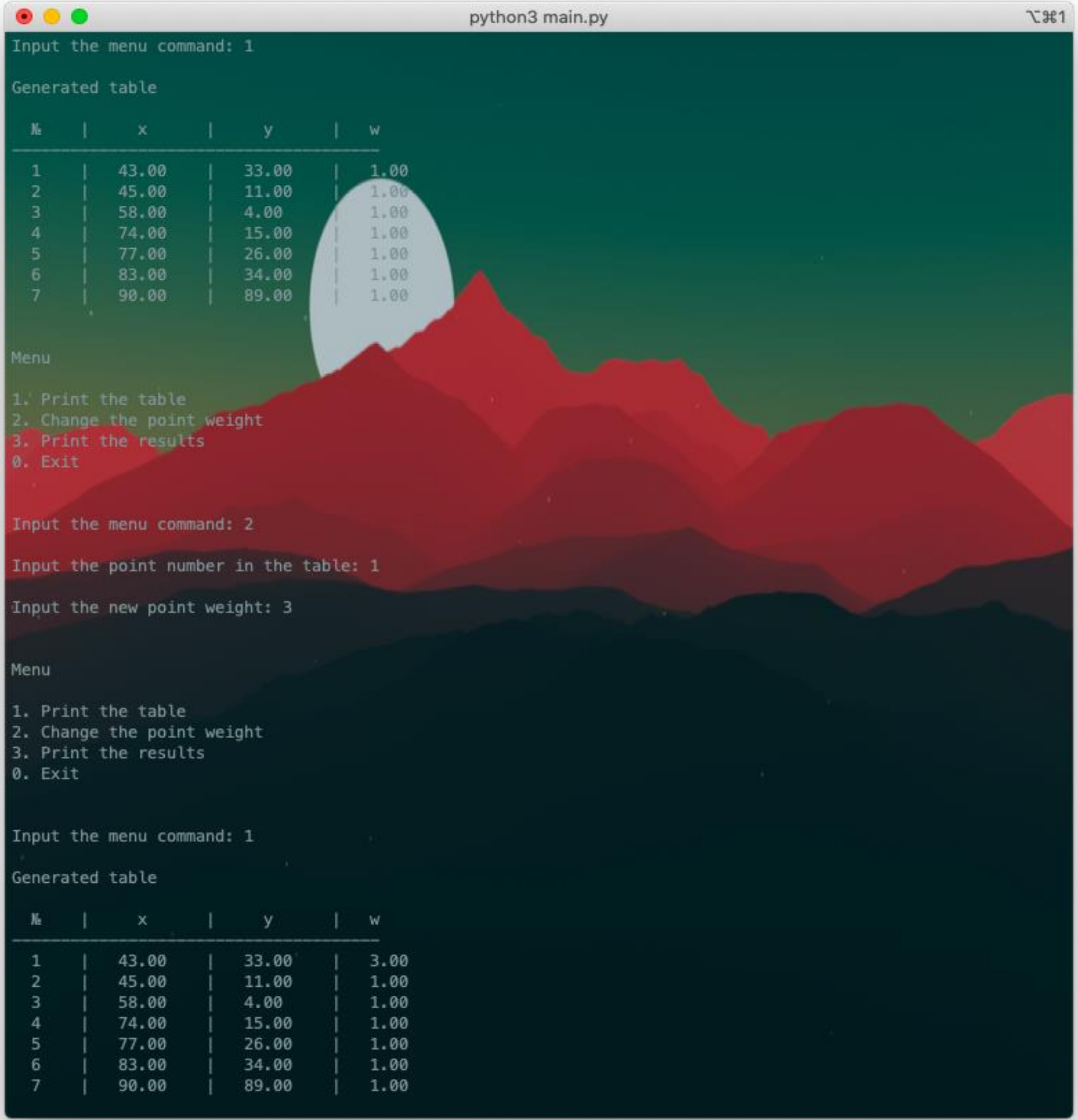
```
            rs_coeff = 0.0

            for k in range(size):
                weight = table[k][2]
                x = table[k][0]
                y = table[k][1]

                a_coeff += weight * pow(x, i + j)
                rs_coeff += weight * y * pow(x, i)

            matrix[i][j] = a_coeff
            matrix[i][power + 1] = rs_coeff

    return matrix

@staticmethod
def gauss(matrix):
    size = len(matrix)

    for i in range(size):
        for j in range(i + 1, size):
            if i == j:
                continue

            k = matrix[j][i] / matrix[i][i]

            for q in range(i, size + 1):
                matrix[j][q] -= k * matrix[i][q]

    result = [0 for i in range(size)]

    for i in range(size - 1, -1, -1):
        for j in range(size - 1, i, -1):
            matrix[i][size] -= result[j] * matrix[i][j]

        result[i] = matrix[i][size] / matrix[i][i]

    return result

@staticmethod
```

```python
    def get_coords(table):
        x_arr = []
        y_arr = []

        for i in range(len(table)):
            x_arr.append(table[i][0])
            y_arr.append(table[i][1])

        return x_arr, y_arr

    @staticmethod
    def set_default_weights(table):
        for i in range(len(table)):
            table[i][2] = 1

    def print_table(self):
        print("\nGenerated table\n")

        print("  №    |    x     |     y     |   w   ")
        print("--------------------------------------")

        for i in range(len(self.table)):
            print(
                "  %-3d  |   %-6.2f   |   %-6.2f   |   %-6.2f   "
                % (i + 1, self.table[i][0], self.table[i][1], self.table[i][2])
            )

    def change_weight(self):
        self.__flag_changed = True

        try:
            position = int(input("\nInput the point number in the table: "))
            new_weight = float(input("\nInput the new point weight: "))
        except:
            print("\n\nInvalid data\n\n")
            return

        if position < 1 or position > len(self.table):
            print("\n\nInvalid data\n\n")
            return
```

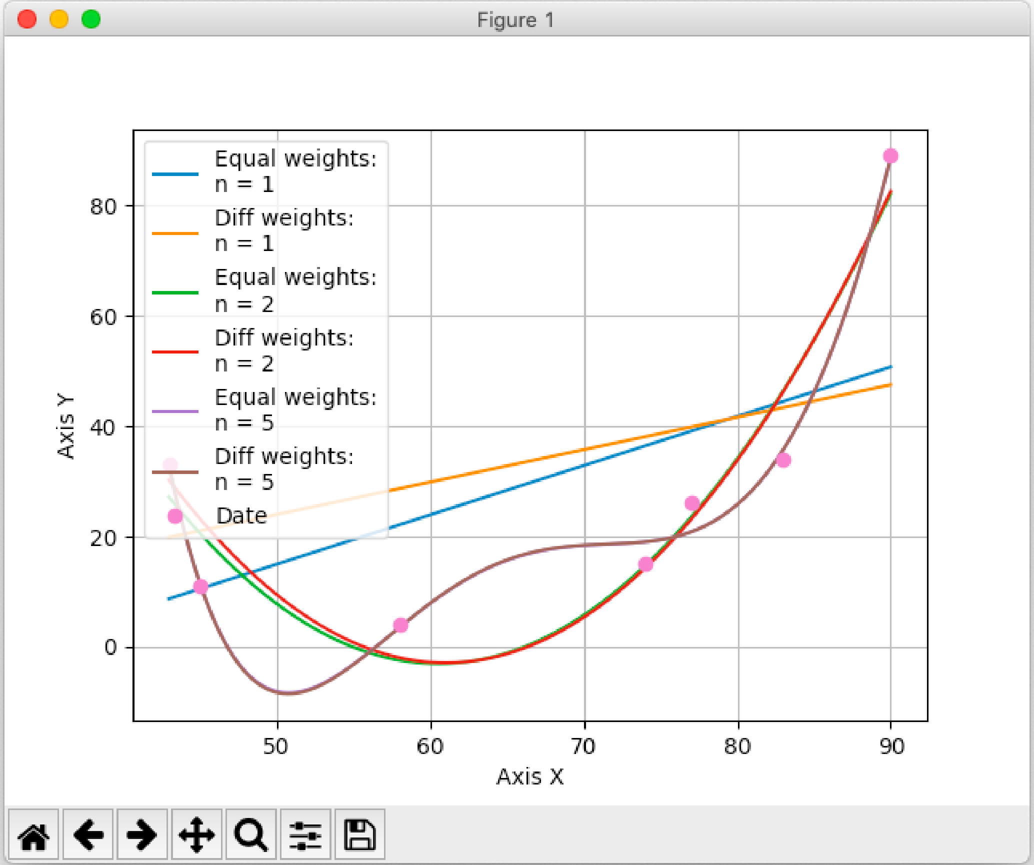
```
        self.table[position - 1][2] = new_weight

    def get_dots(self, table, cur_power, eps=0.01):
        matrix = self.get_slau_matrix(table, cur_power)
        result = self.gauss(matrix)

        x, y = [], []
        k = table[0][0] - eps

        while k <= table[len(table) - 1][0] + eps:
            y_cur = 0
            for j in range(0, cur_power + 1):
                y_cur += result[j] * pow(k, j)

            x.append(k)
            y.append(y_cur)

            k += eps

        return x, y

    def draw(self):
        try:
            power = int(input("\nInput the degree of the approximating polynomial: "))
        except:
            print("\n\nInvalid data\n\n")
            return

        if self.__flag_changed:
            changed_table = deepcopy(self.table)
            self.set_default_weights(self.table)

        for cur_power in range(1, power + 1):
            if cur_power > 2 and cur_power < power:
                continue

            x, y = self.get_dots(self.table, cur_power)

            plt.plot(x, y, label="Equal weights:\nn = %d" % (cur_power))
```

```
    if self.__flag_changed:
        x, y = self.get_dots(changed_table, cur_power)

        plt.plot(x, y, label="Diff weights:\nn = %d" % (cur_power))

x_arr, y_arr = self.get_coords(self.table)

plt.plot(x_arr, y_arr, "o", label="Date")
plt.legend()
plt.grid()
plt.xlabel("Axis X")
plt.ylabel("Axis Y")
plt.show()

if self.__flag_changed:
    self.table = changed_table
```

Пример работы программы



```
python3 main.py

Input the menu command: 1

Generated table

  №    |     x      |     y      |    w
--------------------------------------------
  1    |   43.00    |   33.00    |   1.00
  2    |   45.00    |   11.00    |   1.00
  3    |   58.00    |   4.00     |   1.00
  4    |   74.00    |   15.00    |   1.00
  5    |   77.00    |   26.00    |   1.00
  6    |   83.00    |   34.00    |   1.00
  7    |   90.00    |   89.00    |   1.00


Menu

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


Input the menu command: 2

Input the point number in the table: 1

Input the new point weight: 3


Menu

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


Input the menu command: 1

Generated table

  №    |     x      |     y      |    w
--------------------------------------------
  1    |   43.00    |   33.00    |   3.00
  2    |   45.00    |   11.00    |   1.00
  3    |   58.00    |   4.00     |   1.00
  4    |   74.00    |   15.00    |   1.00
  5    |   77.00    |   26.00    |   1.00
  6    |   83.00    |   34.00    |   1.00
  7    |   90.00    |   89.00    |   1.00
```



Результаты

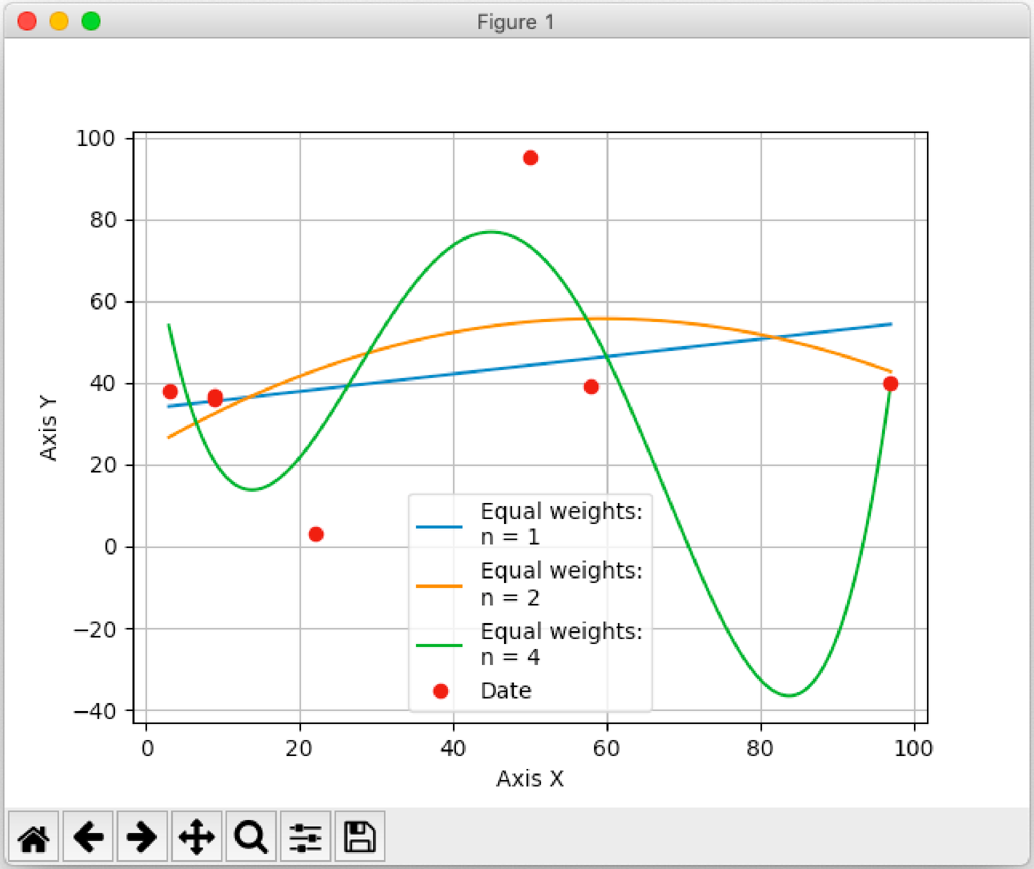
1) **Одинаковые веса**

Входная таблица:

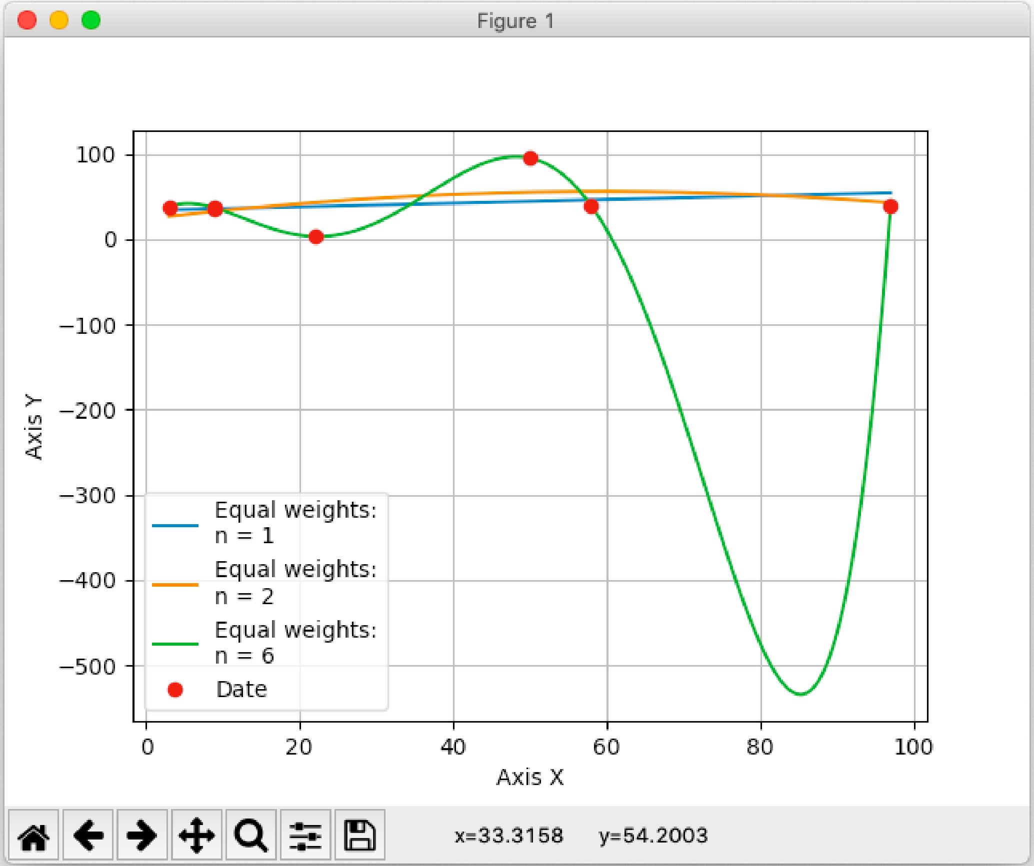
Generated table

№	x	y	w
1	3.00	38.00	1.00
2	9.00	36.00	1.00
3	9.00	37.00	1.00
4	22.00	3.00	1.00
5	50.00	95.00	1.00
6	58.00	39.00	1.00
7	97.00	40.00	1.00

Полиномы степеней 1, 2, 4:



Полиномы степеней 1, 2, 6:



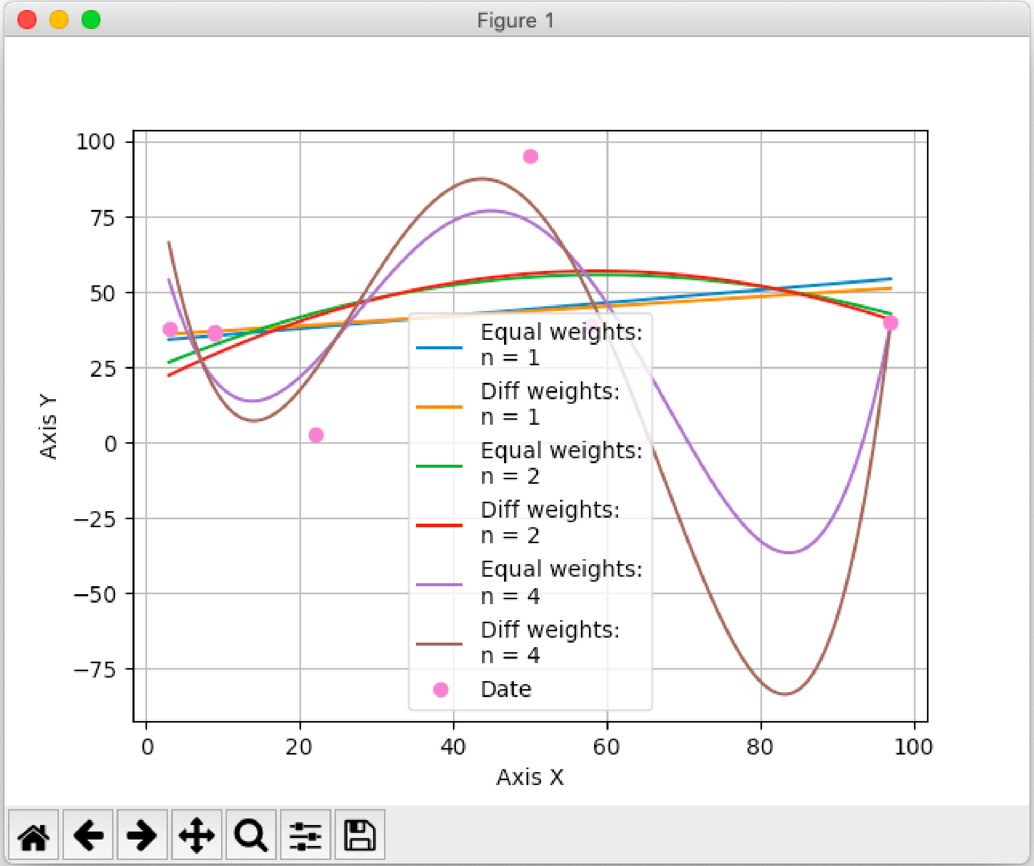
2) Веса разные

Входная таблица:

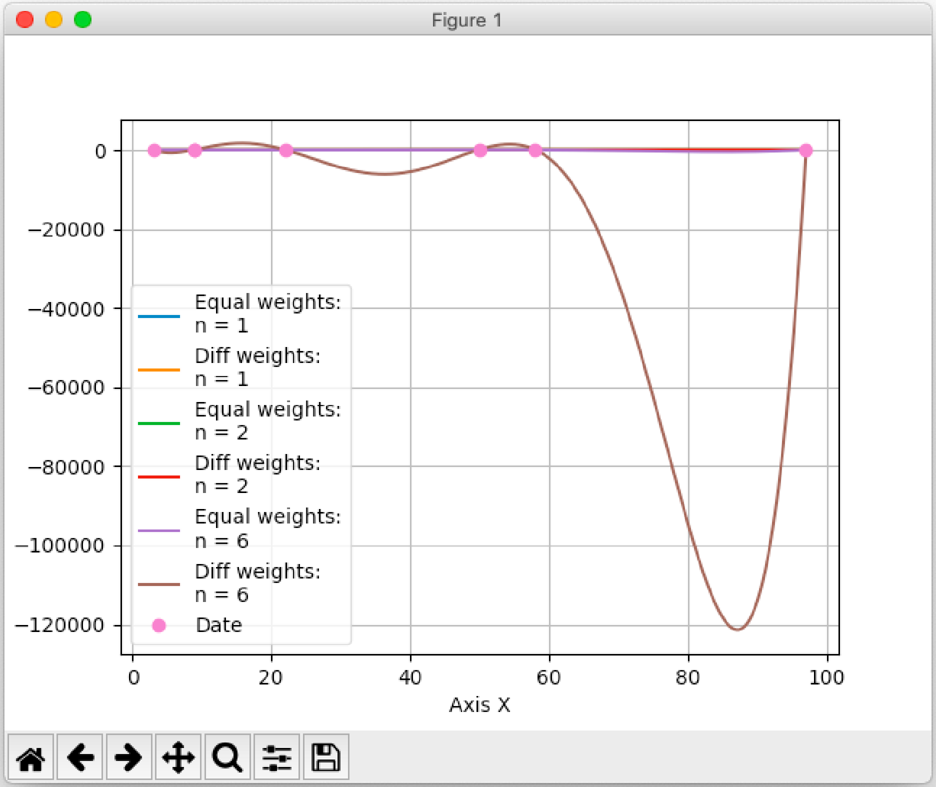
Generated table

№	x	y	w
1	3.00	38.00	2.00
2	9.00	36.00	3.00
3	9.00	37.00	3.00
4	22.00	3.00	4.00
5	50.00	95.00	5.00
6	58.00	39.00	6.00
7	97.00	40.00	7.00

Полиномы степеней 1, 2, 4:



Полиномы степеней 1, 2, 6:



Вопросы при защите лабораторной работы

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

N точками можно определить однозначно полином N - 1 степени. Таким образом, мы построим полином, который пройдет через все табличные точки, причем в выражении выражение в скобках будет тождественно равно нулю, что позволяет сделать вывод о том, что в данном случае у нас еще и нет зависимости от весов (то есть при любых весах полином будет иметь минимально возможное значение в случае прохода через заданные в таблице точки — то есть иметь одни и те же коэффициенты)

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

При заданном условии (n >= N) невозможно построить полином n степени (из-за недостатка точек), так как при этом определитель матрицы будет равен нулю. Программа работать будет, то в некоторые моменты может возникнуть непредвиденная ситуация

3. Получить формулу для коэффициента полинома a0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?

$a_0 = (\sum p_i y_i) / \sum p_i$;

где p_i — вес i-ой точки.
Если разделить числитель и знаменатель на сумму весов, то в знаменателе будет единица, а в числителе — значения точек умноженные на их вес в приведенном состоянии (все веса в пределах от 0 до 1, соотношения остаются). Данная величина — математическое ожидание.

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

$$(\rho_0+\rho_1)a_0 + (\rho_0 x_0 + \rho_1 x_1)a_1 + (\rho_0 x_0^2 + \rho_1 x_1^2)a_2 = \rho_0 y_0 + \rho_1 y_1$$

$$(\rho_0 x_0+\rho_1 x_1)a_0 + (\rho_0 x_0^2 + \rho_1 x_1^2)a_1 + (\rho_0 x_0^3 + \rho_1 x_1^3)a_2 = \rho_0 x_0 y_0 + \rho_1 x_1 y_1$$

$$(\rho_0 x_0^2+\rho_1 x_1^2)a_0 + (\rho_0 x_0^3 + \rho_1 x_1^3)a_1 + (\rho_0 x_0^4 + \rho_1 x_1^4)a_2 = \rho_0 x_0^2 y_0 + \rho_1 x_1^2 y_1$$

$$\Delta = \begin{vmatrix} 2 & x_0+x_1 & x_0^2+x_1^2 \\ x_0+x_1 & x_0^2+x_1^2 & x_0^3+x_1^3 \\ x_0^2+x_1^2 & x_0^3+x_1^3 & x_0^4+x_1^4 \end{vmatrix}$$

$$\Delta = 2\cdot\left[(x_0^2+x_1^2)\cdot(x_0^4+x_1^4) - (x_0^3+x_1^3)(x_0^3+x_1^3)\right] -$$

$$- (x_0+x_1)\left[(x_0+x_1)(x_0^4+x_1^4) - (x_0^2+x_1^2)\cdot(x_0^3+x_1^3)\right] +$$

$$+ (x_0^2+x_1^2)\cdot\left[(x_0+x_1)(x_0^3+x_1^3) - (x_0^2+x_1^2)(x_0^2+x_1^2)\right] =$$

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

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

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.**

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

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

Устроить полный перебор по степеням n и m для вычисления коэффициенты ***a*** и вычислить значение матрицы, а потом выбрать самое близкое значение