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: Чаушев Александър

Группа: ИУ7-46Б

Оценка (баллы) _______________

Преподаватель : Градов В. М.

Москва.
2020 г.

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

Входные данные:

1. Таблица функции с весами с количеством узлов N.
Интерфейс с возможности изменения пользователем весов в таблице. Таблица функции с весами считывается из файла.

2. Степень аппроксимирующего полинома - n.

Результат работы программы:
Графики, где точки - заданная табличная функция, кривые - найденные полиномы.

Краткие теоретические сведения и алгоритм.

Неточные значения функции в узлах приближать функции не по точкам а в среднем. Пусть множество функции, принадлежащих линейному пространству функции. Мы будем понимать что близость средная исходная и аппроксимирующая функция это результат оценки суммы отклонений. Следовательно где сумма меньше та же функция и точнее.
Найдем наилучшее приближение, т.е. такую функцию ф(x), чтобы было справедливым соотношение

$$\sum_{i=1}^{N} \rho_i [\, y(x_i) - \varphi(x_i)\,]^2 = min$$

Далее займемся отысканием наилучшего приближения, которое применительно к таблично заданным функциям называется методом наименьших квадратов.

Алгоритм:
Метод наименьших квадратов
1. Выбирается степень полинома . Обычно степень полинома не превышает 5-6.
2. Потом составляется система линейных алгебраических уравнений:

$$\sum_{m=0}^{n} (x^k, x^m)\, a_m = (y, x^k)\ ,\ 0 \le k \le n\ ,$$ 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. Находим коэффициенты полинома через СЛАУ

Ответы на контрольные вопросы:

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

Пройдет кривая через все точки. Вес точек не будет влиять на результат

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?*

Мы не можем построить кривую n-й степени по n точкам, потому что определитель равен нулю, а чтобы получить единственное решение определитель должен отличатся от нуля. Из-за погрешности программа все считает.

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

Проведем линию, параллельную ОХ. СЛАУ будет иметь одно уравнение:

$(x^0, x^0)\, a_0 = (y, x^0),$ (1) $(x^k, x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}$ тогда $(x^0, x^0) = \sum_{i=1}^{N} \rho_i$ (2)

$(y, x^0) = \sum_{i=1}^{N} \rho_i y_i$ (3)

Выражаем a0 подставляя (2) и (3) в (1).

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

Y = AX^2 + BX + C
Все веса = 1

F(a ; b ; c) = $\sum^{n} (y_i - (ax_i^2 + bx_i + c))^2$

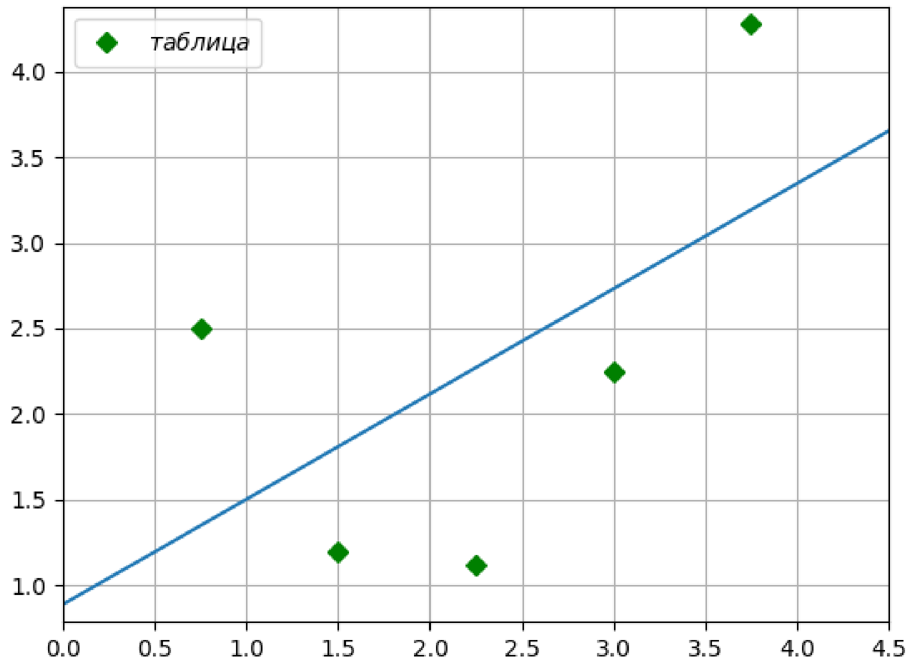
Получим СЛАУ

$$\begin{cases} \dfrac{\partial F}{\partial a} = 0 \\ \dfrac{\partial F}{\partial b} = 0 \\ \dfrac{\partial F}{\partial c} = 0 \end{cases} \Rightarrow \begin{cases} a\sum x_i^4 + b\sum x_i^3 + c\sum x_i^2 = \sum x_i^2 y_i \\ a\sum x_i^3 + b\sum x_i^2 + c\sum x_i = \sum x_i y_i \\ a\sum x_i^2 + b\sum x_i + cn = \sum y_i \end{cases}$$

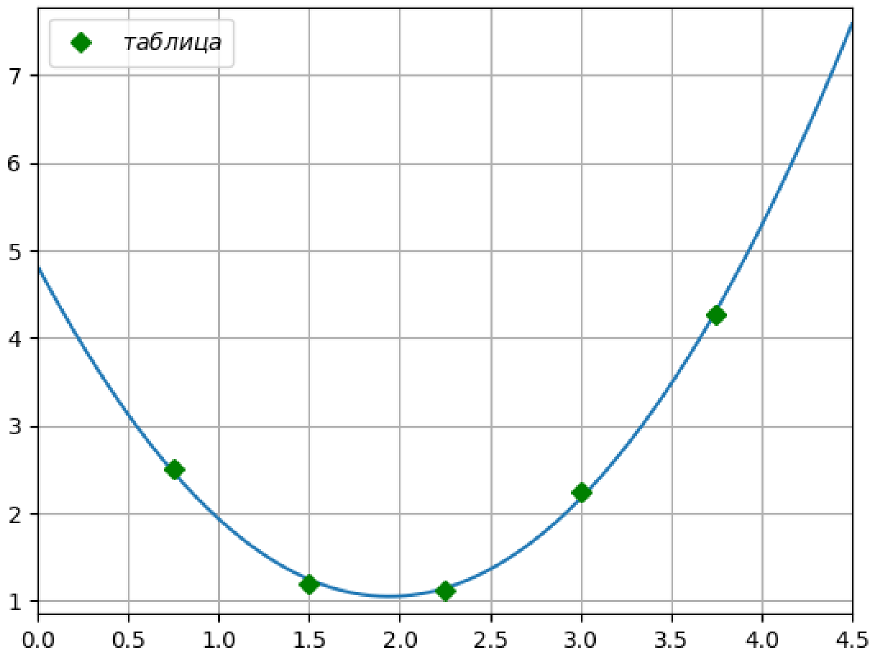
Код программы и результаты ее работы

Пример 1: веса всех точек одинаковы.

```
Таблица:
  X    Y
[0.75, 2.5]
[1.5, 1.2]
[2.25, 1.12]
[3.0, 2.25]
[3.75, 4.28]
Введите степень аппроксимирующей функции = 1
```




```
Таблица:
  X    Y
[0.75, 2.5]
[1.5, 1.2]
[2.25, 1.12]
[3.0, 2.25]
[3.75, 4.28]
Введите степень аппроксимирующей функции = 2
```



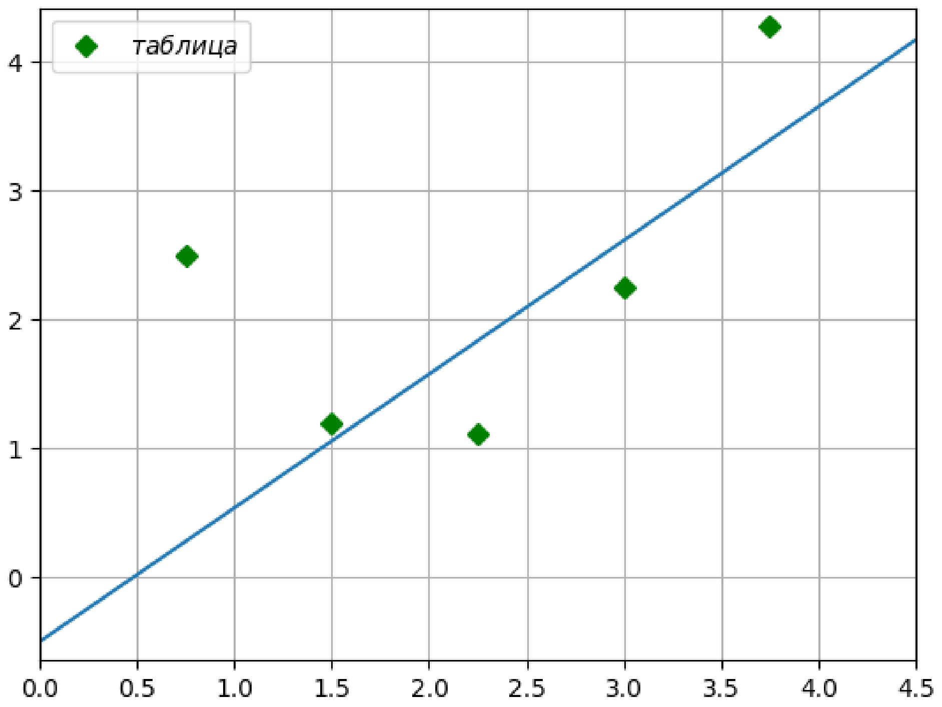
Пример 2: веса точек разные

Чем больше вес точки, тем ближе к точке проходит аппроксимирующая кривая.

Таблица данных

```
0.75 2.50 1
1.50 1.20 1
2.25 1.12 6
3.00 2.25 2
3.75 4.28 3
```

n= 1




```
"""Наилучшее среднеквадратичное приближние"""

import numpy as np
import matplotlib.pyplot as plt
from operator import mul, sub


def fi(x, k):
    return x ** k


def print_result(table, A, n):
    dx = 10
    if len(table) > 1:
        dx = (table[1][0] - table[0][0])

    x = np.linspace(table[0][0] - dx, table[-1][0] + dx, 1000)
    y = []
    for i in x:
        tmp = 0;
        for j in range(0, n + 1):
            tmp += fi(i, j) * A[j]
        y.append(tmp)

    plt.plot(x, y)

    x1 = [a[0] for a in table]
```

```
    y1 = [a[1] for a in table]

    plt.plot(x1, y1, 'kD', color='green', label='$таблица$')
    plt.grid(True)
    plt.legend(loc='best')
    miny = min(min(y), min(y1))
    maxy = max(max(y), max(y1))
    dy = (maxy - miny) * 0.03
    plt.axis([table[0][0] - dx, table[-1][0] + dx, miny - dy, maxy + dy])

    plt.show()
    return


def read_from_file(file_name):
    data = []
    f = open(file_name, "r")
    for line in f:
        if line:
            a, b, c = map(float, line.split())
            data.append([a, b, c])
    f.close()
    return data


def get_slau_matrix(table, n):
    N = len(table)
    matrix = [[0 for i in range(0, n + 1)] for j in range(0, n + 1)]
    col = [0 for i in range(0, n + 1)]

    for m in range(0, n + 1):
        for i in range(0, N):
            tmp = table[i][2] * fi(table[i][0], m)
            for k in range(0, n + 1):
                matrix[m][k] += tmp * fi(table[i][0], k)
            col[m] += tmp * table[i][1]
    return matrix, col


def Gauss(matr):
    n = len(matr)

    for k in range(n):
        for i in range(k + 1, n):
            coeff = -(matr[i][k] / matr[k][k])
            for j in range(k, n + 1):
```

```
                matr[i][j] += coeff * matr[k][j]

    a = [0 for i in range(n)]
    for i in range(n - 1, -1, -1):
        for j in range(n - 1, i, -1):
            matr[i][n] -= a[j] * matr[i][j]
        a[i] = matr[i][n] / matr[i][i]
    return a


def det(b):
    res = 1
    a = b
    n = len(a)
    for i in range(n):
        j = max(range(i, n), key=lambda k: abs(a[k][i]))
        if i != j:
            a[i], a[j] = a[j], a[i]
            res *= -1
        if a[i][i] == 0:
            return 0
        res *= a[i][i]
        for j in range(i + 1, n):
            b = a[j][i] / a[i][i]
            a[j] = [a[j][k] - b * a[i][k] for k in range(n)]
    return res


def get_approx_coef(table, n):
    m, z = get_slau_matrix(table, n)

    for i in range(len(z)):
        m[i].append(z[i])
    ma = list(m)
    det_m = det(ma)
    if abs(det_m) > 0.0000001:
        a_array = Gauss(m)
        return a_array
    else:
        print("Определитель матрицы равен нулю")
        return None


table = read_from_file("table.txt")
print("Таблица:")
print("  X    Y")
```

```
for i in range(len(table)):
    print(table[i][:2])

n = int(input("Введите степень аппроксимирующей функции = "))
A = get_approx_coef(table, n)

if A != None:
    print_result(table, A, n)
```