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 Бугаенко А.П.

Группа ИУ7-45Б

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.

1 Исходные данные

1. Таблица функции с **весами** ρ_i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек.

x	y	ρ_i

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

2. Степень аппроксимирующего полинома - n.

Результат работы программы.

Графики, построенные по аналогии с рис.1 в тексте Лекции №4: *точки* - заданная табличная функция, *кривые*- найденные полиномы.

Обязательно приводить таблицы, по которым работала программа.

При каких исходных условиях надо представить результаты в отчете?

1. Веса всех точек одинаковы и равны, например, единице. Обязательно построить полиномы степеней n=1 и 2. Можно привести результаты и при других степенях полинома, однако, не загромождая сильно при этом рисунок.

2. Веса точек разные. Продемонстрировать, как за счет назначения весов точкам можно изменить положение на плоскости прямой линии (полином первой степени), аппроксимирующей **один и тот же набор точек** (**одну** таблицу y(x)). Например, назначая веса узлам в таблице **изменить знак углового коэффициента прямой**. На графике в итоге должны быть представлены точки исходной функции и две аппроксимирующие их прямые линии. Одна отвечает значениям ρ_i=1 для всех узлов, а другая- назначенным разным весам точек. Информацию о том, какие именно веса были использованы в расчете обязательно указать, чтобы можно было проконтролировать работу программы (лучше это сделать в виде таблицы).

2 Код программы

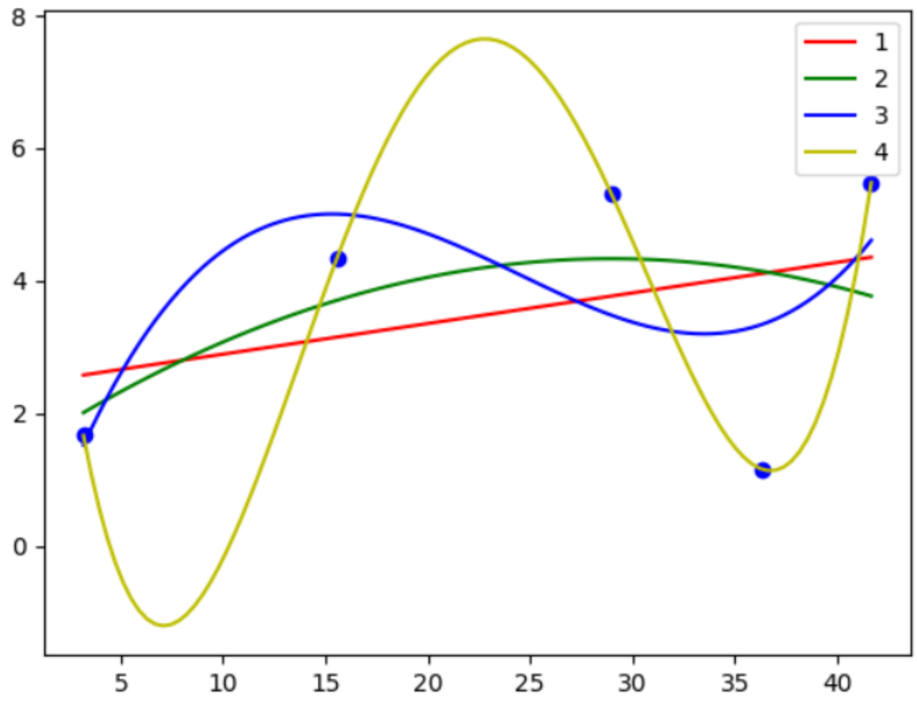
Листинг main.py

```
import math as m
import numpy as np
import random as rd
from numpy import linalg as lg

def countNPowerPolym(a, x):
    y = 0
    for i in range(0, len(a)):
        y += a[i] * x ** i
    return y

def funcScalarMult(phi, fi, k, m, P):
    scalarMult = 0
    for i in range(0, len(P)):
        scalarMult += P[i] * (phi[i] ** k) * (fi[i] ** m)
    return scalarMult

def formA(X, P, N):
    A = np.zeros([N, N])
    for i in range(0, N):
        for j in range(0, N):
            A[i][j] = funcScalarMult(X, X, i, j, P)
    return A

def formB(X, Y, P, N):
    B = np.zeros([N, 1])
    for i in range(0, N):
            B[i][0] = funcScalarMult(Y, X, 1, i, P)
    return B

def bestRmsApproximation(X, Y, P, n):
    N = n + 1
    A = formA(X, P, N)
    B = formB(X, Y, P, N)
    koefArr = np.dot(lg.inv(A), B)
    return koefArr

def generateValues(N, y_interval):
    i = 0
    P = []
    X = []
    Y = []
    while i < N:
        P.append(rd.uniform(1, 1))
        X.append(rd.uniform(i, i + 1) * 10)
        Y.append(rd.uniform(y_interval[0], y_interval[1]))
        i += 1
    return X, Y, P

from numpy import linspace
from matplotlib import pyplot as plt
```

```
def plotGraph(X, Y, P):
    a_1 = bestRmsApproximation(X, Y, P, 1)
    a_2 = bestRmsApproximation(X, Y, P, 2)
    a_3 = bestRmsApproximation(X, Y, P, 3)
    a_4 = bestRmsApproximation(X, Y, P, 4)
    X_1 = linspace(min(X), max(X), 100)
    Y_1 = [countNPowerPolym(a_1, x) for x in X_1]
    X_2 = linspace(min(X), max(X), 100)
    Y_2 = [countNPowerPolym(a_2, x) for x in X_2]
    X_3 = linspace(min(X), max(X), 100)
    Y_3 = [countNPowerPolym(a_3, x) for x in X_3]
    X_4 = linspace(min(X), max(X), 100)
    Y_4 = [countNPowerPolym(a_4, x) for x in X_4]
    plt.plot(X, Y, 'bo')
    plt.plot(X_1, Y_1, 'r', label="1")
    plt.plot(X_2, Y_2, 'g', label="2")
    plt.plot(X_3, Y_3, 'b', label="3")
    plt.plot(X_4, Y_4, 'y', label="4")
    plt.legend()
    plt.show()
import tkinter as tk
def updateGraph():
    P = []
    for i in range(1, 6):
        P.append(float(cols[i][2].get()))
    plt.close()
    X, Y, P_ = generateValues(5, [0, 10])
    plotGraph(X, Y, P)
def generate():
    X, Y, P = generateValues(5, [0, 10])
    for i in range(1, 6):
        cols[i][0].delete(0, tk.END)
        cols[i][0].insert(0, str(X[i - 1]))
        cols[i][1].delete(0, tk.END)
        cols[i][1].insert(0, str(Y[i - 1]))
        cols[i][2].delete(0, tk.END)
        cols[i][2].insert(0, str(P[i - 1]))
    plt.close()
    plotGraph(X, Y, P)
root = tk.Tk()
cols = []
for i in range(6):
    rows = []
    for j in range(3):
        e = tk.Entry(root, relief=tk.GROOVE)
        e.grid(row=i, column=j, sticky=tk.NSEW)
        rows.append(e)
    cols.append(rows)
cols[0][0].insert(0, str("X"))
cols[0][1].insert(0, str("Y"))
cols[0][2].insert(0, str("P"))
update = tk.Button(root, text="Обновить", command=updateGraph)
generate = tk.Button(root, text="Генерировать", command=generate)
update.grid()
generate.grid()
tk.mainloop()):
```

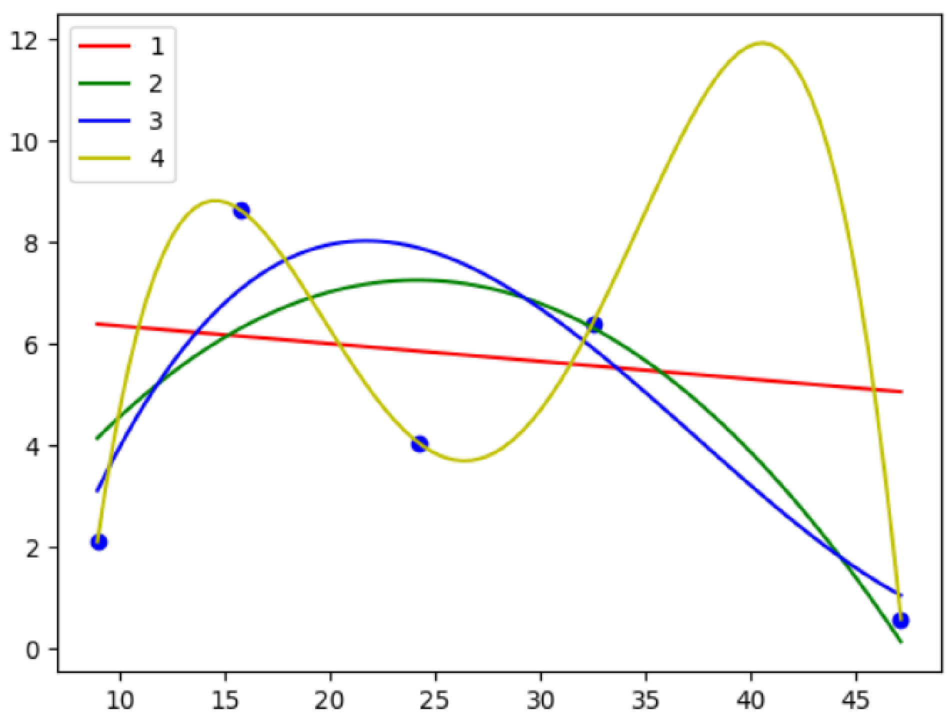
3 [illegible]

При одинаковых весах точек:



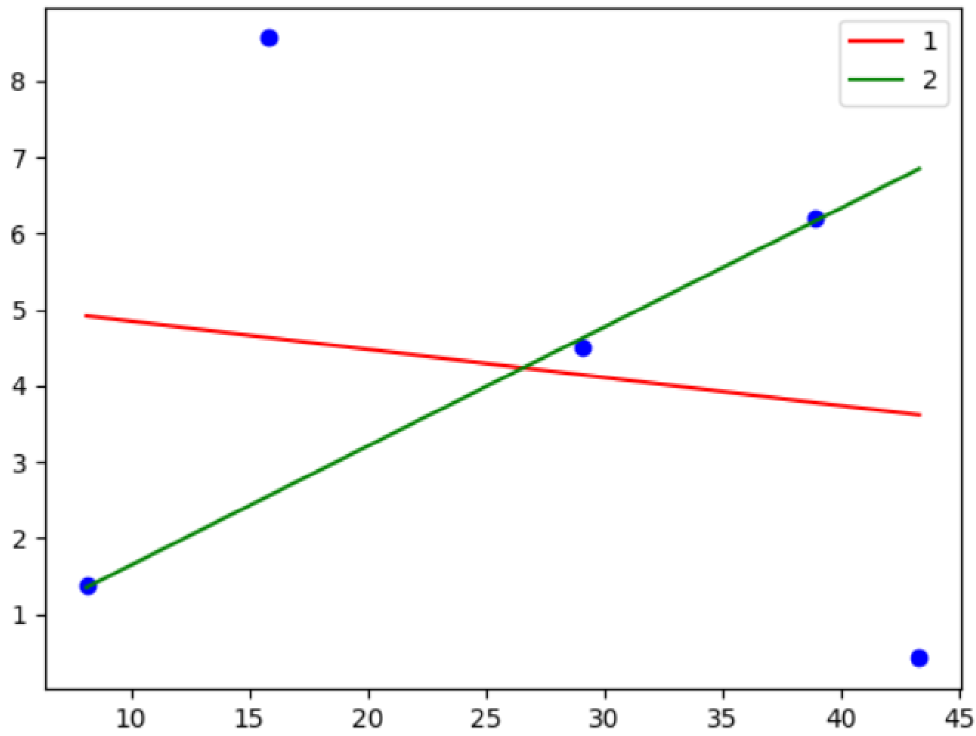
X	Y	P
3.180411906272341	1.6684226144605807	1.0
15.577278970358538	4.350635792309143	1.0
28.976230338181196	5.310130194540058	1.0
36.3427082621902	1.1584559432801023	1.0
41.66982110844724	5.482250417653878	1.0

При разных весах точек:



X	Y	P
3.180411906272341	1.6684226144605807	1.0
15.57727870358538	4.35063579230914З	2.0
28.976230338181196	5.310130194540058	1.0
36.3427082621902	1.1584559432801023	4.0
41.66982110844724	5.482250417653878	0.5

Изменение наклона прямой с помощью весов:



X	P_1	P_2
8.125090342070383	1	1
15.785745894770091	1	0
29.05407239603495	1	1
38.91393893628915	1	2
43.32510802765819	1	0

4 Контрольные вопросы

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

В таком случае полином, который будет построен, по сути будет являться полиномом Ньютона степени n. И будет проходить точно

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

Для нахождении коэффициентов полинома степени n нужно не менее n+1 условий. Если n <= N, то коэффициенты найти невозможно, поэтому при n <= N программа работать не будет.

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

$$n = 0 \Rightarrow \varphi(x) = a_0$$

Оценка суммы:

$$I = \sum_{i=0}^{N} \rho_i \left(y(x_i) - \varphi(x_i)\right)^2$$

Продифференцируем по a_0:

$$\frac{dI}{da_0} = -2\sum_{i=0}^{N} \rho_i (y_i - a_0)$$

$$\frac{dI}{da_0} = 0 \Rightarrow \sum_{i=0}^{N} \rho_i a_0 = \sum_{i=0}^{N} \rho_i y_i$$

$$a_0 = \frac{\sum_{i=0}^{N} \rho_i y_i}{\sum_{i=0}^{N} \rho_i}$$

Полученный коэффициент является взвешенным средним арифметическим значения для ординат изначального набора точек.

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

$$\varphi(x) = a_0 + a_1 x + a_2 x^2$$

$$\begin{cases} (\varphi, \varphi_0) = (y, \varphi_0) \\ (\varphi, \varphi_1) = (y, \varphi_1) \\ (\varphi, \varphi_2) = (y, \varphi_2) \end{cases}$$

$$\begin{cases} a_0 X_0 + a_1 X_1 + a_2 X_2 = Y_0 \\ a_0 X_1 + a_1 X_2 + a_2 X_3 = Y_1 \\ a_0 X_2 + a_1 X_3 + a_2 X_4 = Y_2 \end{cases}$$

$$X_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} x_i^k \qquad Y_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} y_i\, x_i^k$$

$$\begin{pmatrix} X_0 & X_1 & X_2 \\ X_1 & X_2 & X_3 \\ X_2 & X_3 & X_4 \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \end{pmatrix}$$

$$XA = Y$$

$$\det(X) = X_0X_2X_4 + 2X_1X_2X_3 - X_0X_3X_3 - X_1X_1X_4 - (X_2)^3$$

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

$$\begin{cases} (\varphi, 1) = (y, 1) \\ (\varphi, x^n) = (y, x^n) \\ (\varphi, x^m) = (y, x^m) \end{cases}$$

$$\begin{cases} a_0X_0 + a_1X_n + a_2X_m = Y_0 \\ a_0X_n + a_1X_{2n} + a_2X_{n+m} = Y_n \\ a_0X_m + a_1X_{n+m} + a_2X_{2m} = Y_m \end{cases}$$

$$X_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} x_i^k \qquad Y_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} y_i\, x_i^k$$

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

Для решения этой задачи необходимо составить цикл по всем сочетаниям m и n. На основе полученных полиномов можно посчитать наборы коэффициентов a_k После этого мы можем вычислить оценку суммы I, а затем из всех I выбрать наименьшую и использовать соответствующие наборы a_k, m, и n.