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 Прохорова Л. А.

Группа ИУ7-43Б

Оценка (баллы) _______________

Преподаватель Градов В. М.

Москва.
2020 г.

Тема: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

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** ρ_i с количеством узлов N.

x	y	ρ_i

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

2. Степень аппроксимирующего полинома - n.

Выходные данные

График, в котором изображён аппроксимирующий полином и точки из исходной таблицы значений.

Описание алгоритма.

Пусть имеется множество функций $\phi(x)$, принадлежащих линейному пространству функций. Под близостью в среднем исходной yи аппроксимирующей ϕ функций будем понимать результат оценки суммы $I = \sum_{i=1}^{N} \rho_i$ (4.1) где ρ_i - вес точки. Суммирование выполняется по всем N узлам заданной функции. Такой вид аппроксимации называют среднеквадратичным приближением.

Будем искать наилучшее приближение, т.е. такую функцию $\bar{\phi}(x)$, чтобы было справедливым соотношение

$\sum_{i=1}^{N} \rho_i$ (4.2)

Разложим функцию $\phi(x)$ по системе линейно независимых функций $\phi_k(x)$:

$\phi(x) = \sum_{k=0}^{n} a_k \phi_k(x)$ (4.3) $\quad (f, \phi) = \sum_{i=1}^{N} \rho_i f(x_i)\phi(x_i), \rho_i > 0.$

1. $(f, \phi) = (\phi, f)$

2. $(f + \phi, \gamma) = (f, \gamma) + (\phi, \gamma)$ (4.4)

Подставляя (4.3) в условие (4.2), получим с учетом (4.4.)

$$\big((y - \phi), (y - \phi)\big) = (y, y) - 2\sum_{k=0}^{n} a_k (y, \phi_k) + \sum_{k=0}^{n}\sum_{m=0}^{n} a_k a_m (\phi_k, \phi_m) = \min$$

Дифференцируя это выражение по a_k и приравнивая производные нулю, найдем

$$\sum_{m=0}^{n}(\phi_k, \phi_m)a_m = (y, \phi_k), 0 \le k \le n \,. \tag{4.5}$$

Определитель этой системы в силу линейной независимости функций $\phi_k(x)$ не равен нулю. Следовательно, из системы (4.5) можно найти коэффициенты a_k, определяющие функцию $\phi(x)$ согласно (4.3) и минимизирующие (4.1). Таким образом, наилучшее среднеквадратичное приближение существует и оно единственно.

Если $\phi_k(x) = x^k$, причем $0 \le k \le n$, то система уравнений (4.5) принимает вид

$$\sum_{m=0}^{n}\left(x^k, x^m\right)a_m = \left(y, x^k\right), 0 \le k \le n \,, \tag{4.6}$$

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

Код программы

```
# Аппроксимация ф-и
# Наилучшее среднеквадратичное значение
import matplotlib.pyplot as plt
import numpy as np

def f(x_arr, coeff):
    res = np.zeros(len(x_arr))
    for i in range(len(coeff)):
        res += coeff[i] * (x_arr ** i)
    return res

### Считать данные с файла
def read_from_file(filename):
    f = open(filename, "r")
    x, y, ro = [], [], []
    for line in f:
        line = line.split(" ")
        x.append(float(line[0]))
        y.append(float(line[1]))
        ro.append(float(line[2]))
    return x, y, ro
```

```python
def print_table(x, y, ro):
    length = len(x)
    print("x      y      ro")
    for i in range(length):
        print("%.4f %.4f %.4f" % (x[i], y[i], ro[i]))
    print()

def print_matr(matr):
    for i in matr:
        print(i)

### Вычислить значение
def root_mean_square(x, y, ro, n):  # n - кол-во искомых коэффициентов
    length = len(x)
    sum_x_n = [sum([x[i] ** j * ro[i] for i in range(length)]) for j in range(n * 2 - 1)]
    sum_y_x_n = [sum([x[i] ** j * ro[i] * y[i] for i in range(length)]) for j in range(n)]
    matr = [sum_x_n[i:i + n] for i in range(n)]
    for i in range(n):
        matr[i].append(sum_y_x_n[i])
    print_matr(matr)
    return Gauss(matr)

def Gauss(matr):
    n = len(matr)
    # приводим к треугольному виду
    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]
    print("\ntriangled:")
    print_matr(matr)
    # находим неизвестные
    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
```

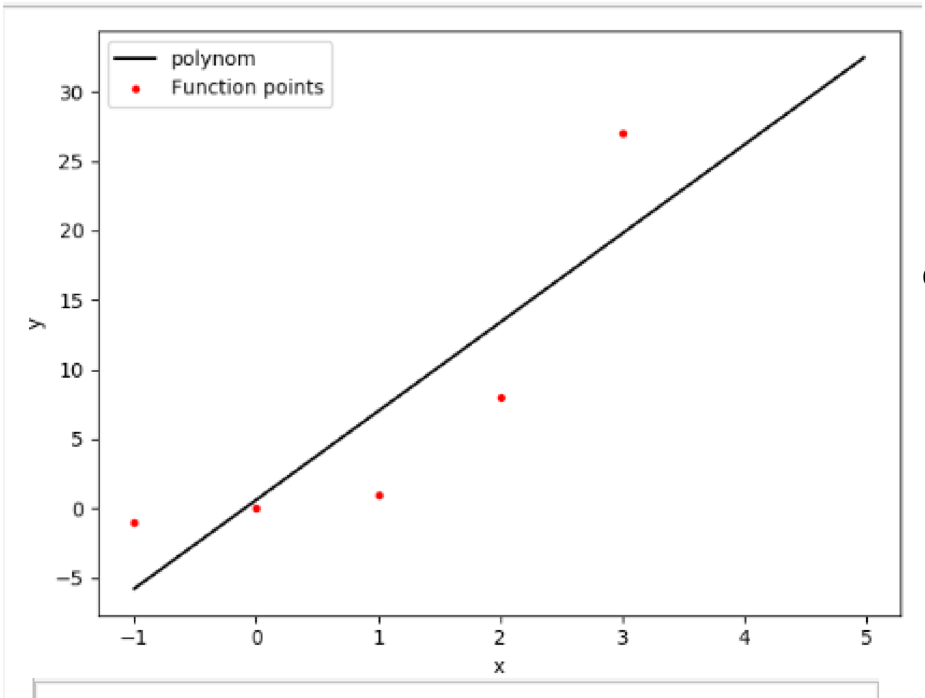
```python
### Отобразить результат
def show(a, x, y, ro, filename, n):
    t = np.arange(-1.0, 5.0, 0.02)
    plt.figure(1)
    plt.ylabel("y")
    plt.xlabel("x")
    plt.plot(t, f(t, a), 'k', label='polynom')
    if (filename == "data2.txt"):
        ro1 = []
        for i in range(len(x)):
            ro1.append(1)
        a1 = root_mean_square(x, y, ro1, n + 1)
        plt.plot(t, f(t, a1), 'b', label='polynom1')

    plt.plot(x[0], y[0], 'ro', markersize=ro[0] + 2, label="Function points")
    for i in range(1, len(x)):
        plt.plot(x[i], y[i], 'ro', markersize=ro[i] + 2)
    plt.legend()
    plt.show()

def main():
    filename = input("Input filename: ")
    x, y, ro = read_from_file(filename)
    n = int(input("Input n: "))  # Степень многочлена
    print_table(x, y, ro)
    a = root_mean_square(x, y, ro, n + 1)
    print("\na:", a)
    show(a, x, y, ro, filename, n)

main()
```

Результат работы программы.

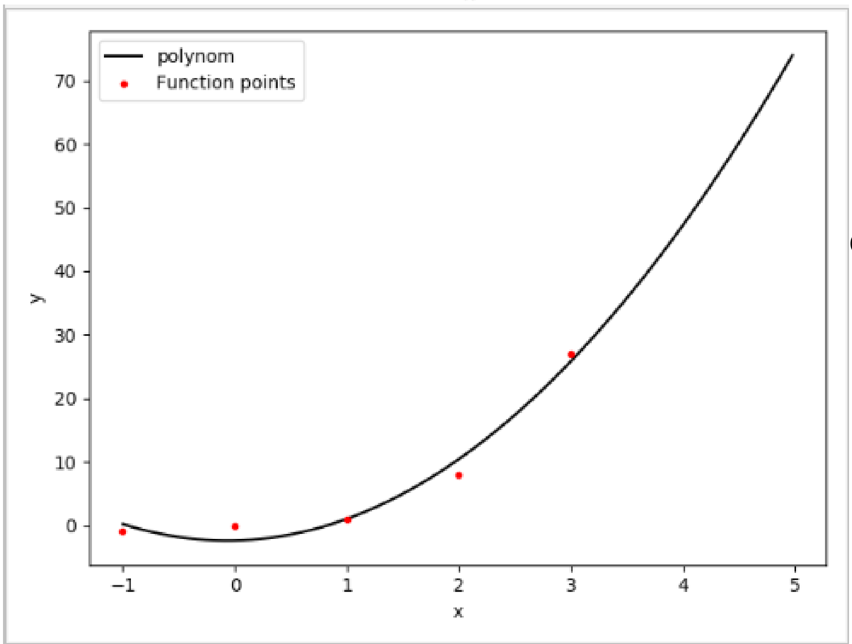
1. Веса всех точек равны 1.

Исходная таблица

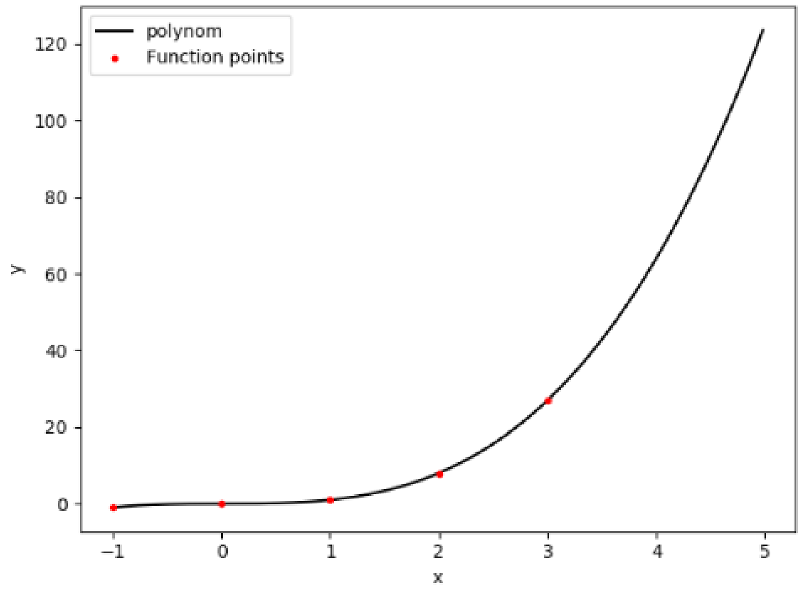
x_i	y_i	ρ_i
-1	-1	1
0	0	1
1	1	1
2	8	1
3	27	1



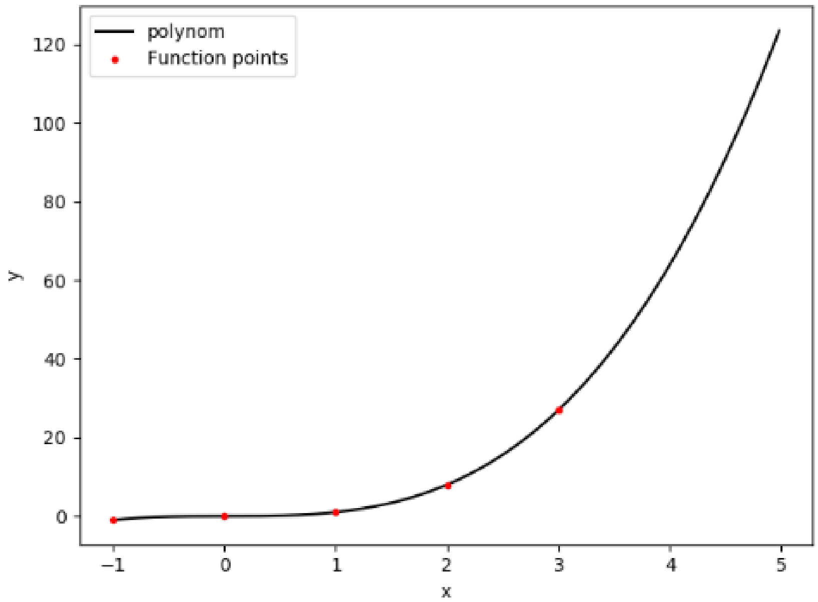
Степень полинома = 1.



Степень полинома = 2.



Степень полинома = 3.



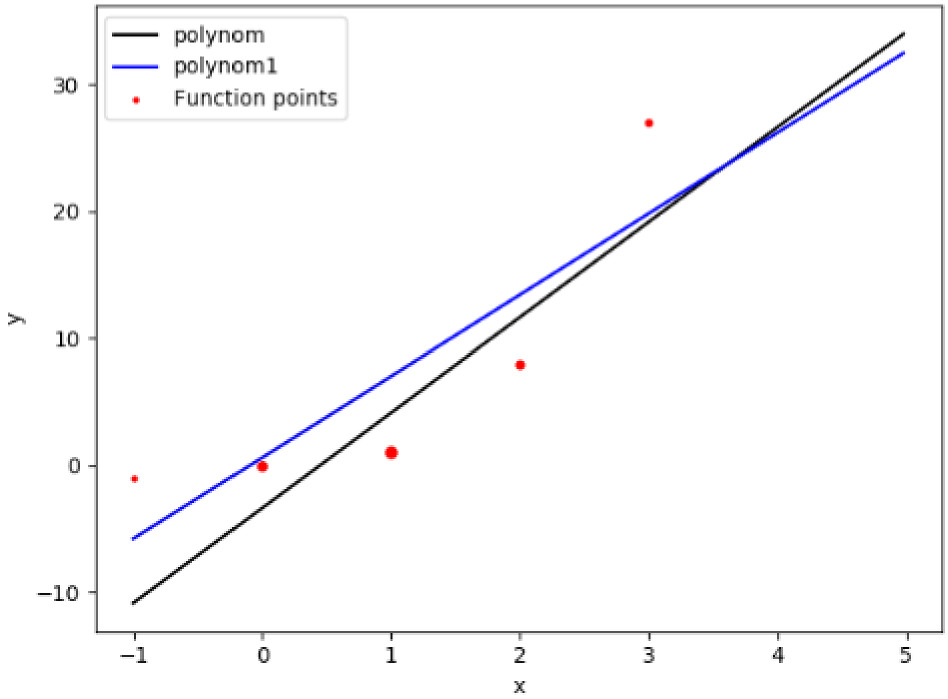
Степень полинома = 7.

2. **Веса точек разные.**

Таблица значений

x_i	y_i	ρ_i
-1	-1	0.1
0	0	2
1	1	3

2	8	1.5
3	27	0.9



Степень полинома = 1.

На графике диаметр точки прямо пропорционален её весу. Синим цветом обозначен полином при единичных весах. Чёрным цветом полином с текущими весами.

Вопросы при защите лабораторной работы.

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

Для построения полинома будут использоваться все имеющиеся точки, независимо от весов которые они имеют.

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?

Программа будет работать(доказательство есть в одном из примеров, когда n = 7, а N = 5), однако полином n-ой в данном случае нельзя построить по N точкам, так как определитель будет равен нулю. Программа будет работать из-за погрешностей.

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

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

Формула для расчёта a_0. Эта величина выражает математическое ожидание.

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

Пусть есть таблица:

x_i	y_i	ρ_i
x_0	y_0	1
x_1	y_1	1

Тогда имеем СЛАУ:

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

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