|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 4** |

Название:

**Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.**

Дисциплина: Вычислительные алгоритмы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-46Б |  |  | Нгуен Ань Тхы |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Градов В.М. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

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

**I. Исходные данные.**

1. Таблица функции с **весами**  с количеством узлов N.

|  |  |  |
| --- | --- | --- |
| x | y |  |
|  |  |  |
|  |  |  |

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

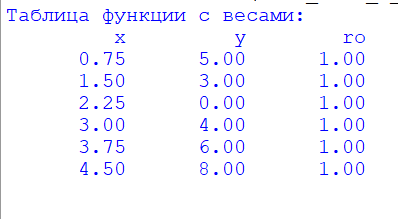
2. Степень аппроксимирующего полинома - n.

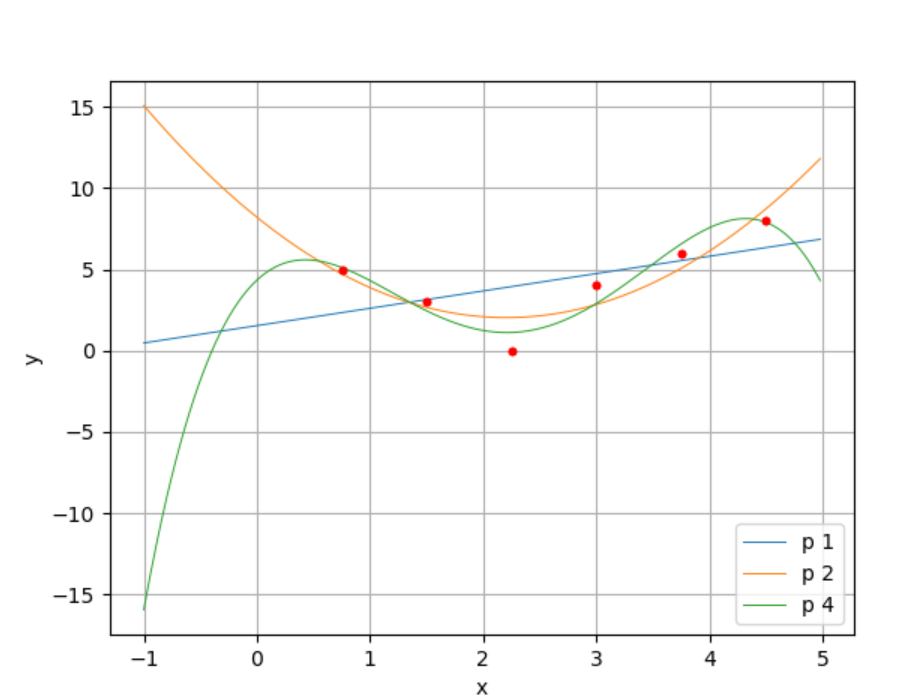
**II. Результат работы программы.**

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

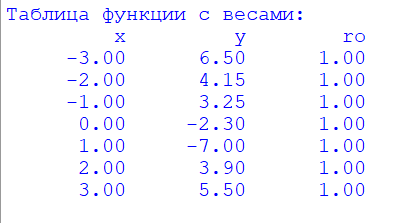
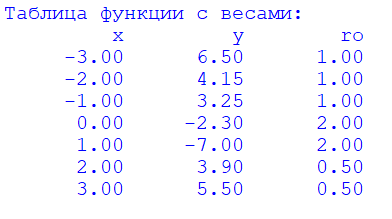
**III. Пример выполнения прграммы:**

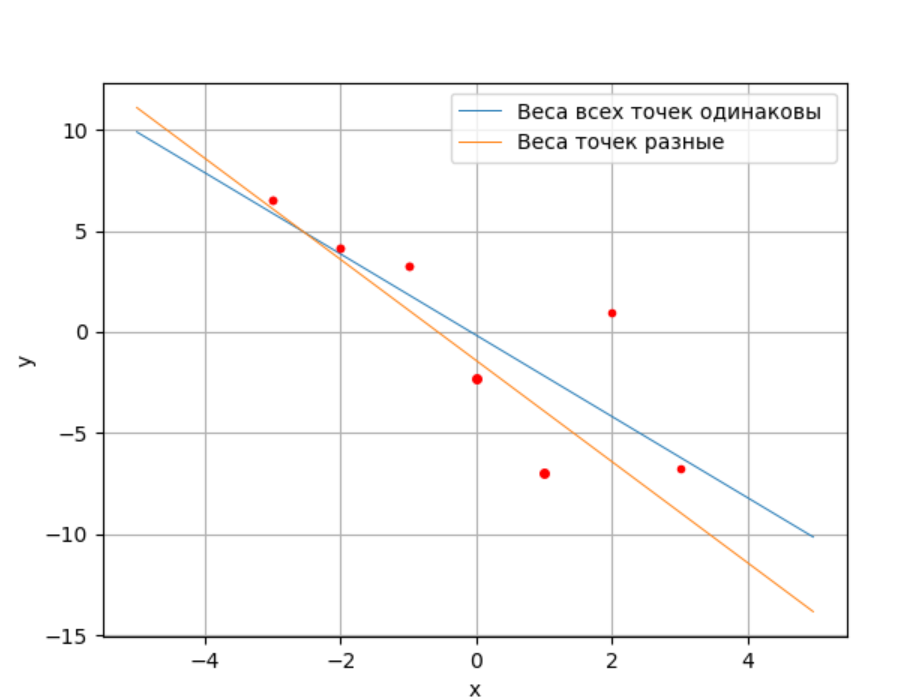
1. Веса всех точек одинаковы и равны:



****

Веса точек разные:

**** ****

****

**IV. Описание алгоритма:**

Процедура интерполяции функции подразумевает построение некоторой новой функции, совпадающей с заданной в фиксированных узлах. В ряде случаев целесообразно приближать функции не по точкам, а в среднем, например, когда значения функции в узлах определены неточно.

Пусть имеется множество функций , принадлежащих линейному пространству функций. Под близостью в среднем исходной и аппроксимирующей  функций будем понимать результат оценки суммы

 (4.1)

где  - вес точки. Суммирование выполняется по всем N узлам заданной функции.

Такой вид аппроксимации называют среднеквадратичным приближением. Можно рассмотреть две задачи:

1 - подобрать функцию  так, чтобы выполнялось неравенство

 ;

2 - найти наилучшее приближение, т.е. такую функцию , чтобы было справедливым соотношение

 (4.2)

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

Разложим функцию  по системе линейно независимых функций :

 . (4.3)

В дальнейшем для сокращения записи будем пользоваться определением скалярного произведения в пространстве дискретно заданных функций

.

Несложно установить, что имеют место следующие равенства, справедливые для обычного скалярного произведения элементов линейного пространства

1. 

2.  (4.4)

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

 .

Дифференцируя это выражение по  и приравнивая производные нулю, найдем

 . (4.5)

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

В качестве  чаще всего используют полиномы Лежандра, Чебышева, Лагерра, Эрмита, ортогональные с заданным весом.

Наиболее употребительный вариант метода наименьших квадратов соответствует случаю степенного вида функций , т.е. , причем . Обычно в сумме (4.3) берут не более пяти-шести членов.

Система уравнений (4.5) при этом принимает вид

 ,  , (4.6)

где .

**Пример 2.1.** Методом наименьших квадратов аппроксимировать функцию линейной зависимостью вида  *.*

В данном случае . В итоге система уравнений (4.6) имеет вид





Скалярные произведения в полученной системе записываются следующим образом:

 ,  , 

 ,  .

Окончательно

 ,

 .

Система функций  не ортогональна, поэтому при больших  задача (4.5) плохо обусловлена, в связи с чем на практике ограничиваются значениями  .

**V. Ответ на вопросы защита:**

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

При этом случае, кривая пройдет по всем точкам независимо от вессов.

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

По идее невозможно построить кривую n степени по n точкам. Определитель равен нулю. Программа все считает из-за погрешности расчетов.

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

- математическое ожидание

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

Таблица значение с весами:

|  |  |  |
| --- | --- | --- |
| X | Y | P |
|  |  | 1 |
|  |  | 1 |

Δ = 2 + (

+  *-*

*-* - = 0

*Т.к* Δ = 0, системы решений нет

5. Построить СЛАУ при выборочном задании степеней аргумента полинома

, причем степени n и m в этой формуле известны.

6. Решить задачу из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами , т.е. количество неизвестных равно 5.

*+ + =*

*Где k:* степень полинома; *0 ik; a = i < n; b = i < m*

**VI. Код программы:**

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

#read datas from file

def read\_from\_file(filename):

f = open(filename, "r")

N = list(map(int, f.readline().split()))

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 N, x, y, ro

def print\_table(x, y, ro):

print("%10s%10s%10s" % ("x", "y", "ro"))

for i in range(len(x)):

print("%10.2f%10.2f%10.2f" % (x[i], y[i], ro[i]))

print()

def print\_matr(matr):

for line in matr:

for value in line:

print("%8.2f" % (value), end = '')

print()

#Calculate

def root\_mean\_square(x, y, ro, n):

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])

return Gauss(matr)

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]

print("\nTriangled:")

print\_matr(matr)

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

#Draw graphics

def show(a, x, y, ro, n):

t = np.arange(-5, 5, 0.04)

#plt.figure(1)

plt.ylabel("y")

plt.xlabel("x")

plt.plot(t, f(t, a), label = LINENAME[n - 1], lw = 0.7)

for i in range(len(x)):

plt.plot(x[i], y[i], "ro", markersize = ro[i] + 2)

def process(filename):

N, x, y, ro = read\_from\_file(filename)

print("Таблица функции с весами: ")

print\_table(x, y, ro)

for n in N:

a = root\_mean\_square(x, y, ro, n + 1)

print("\na: ", end = '')

for value in a:

print("%8.2f" % (value), end = '')

print()

show(a, x, y , ro, n)

plt.legend()

plt.grid(True)

#Task 1:

LINENAME = ["p1", "p2","p3", "p4"]

process("data.txt")

plt.show()

#Task 2:

LINENAME = ["Веса всех точек одинаковы "]

process("data1.txt")

LINENAME = ["Веса точек разные"]

process("data2.txt")

plt.show()