|  |  |
| --- | --- |
| 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 «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 , которая формируется случайным образом при каждом запуске программы

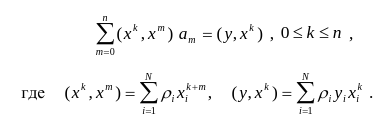
**Выходные данные**

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:



1. 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

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

*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()

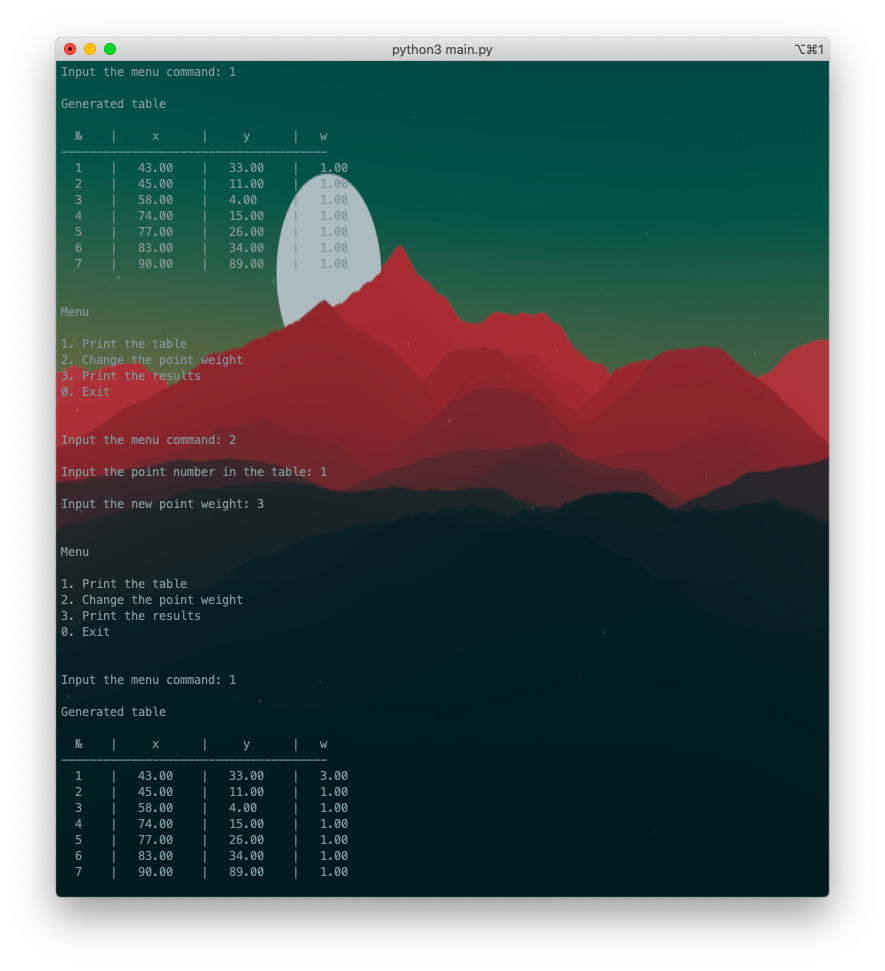
plt.xlabel("Axis X")

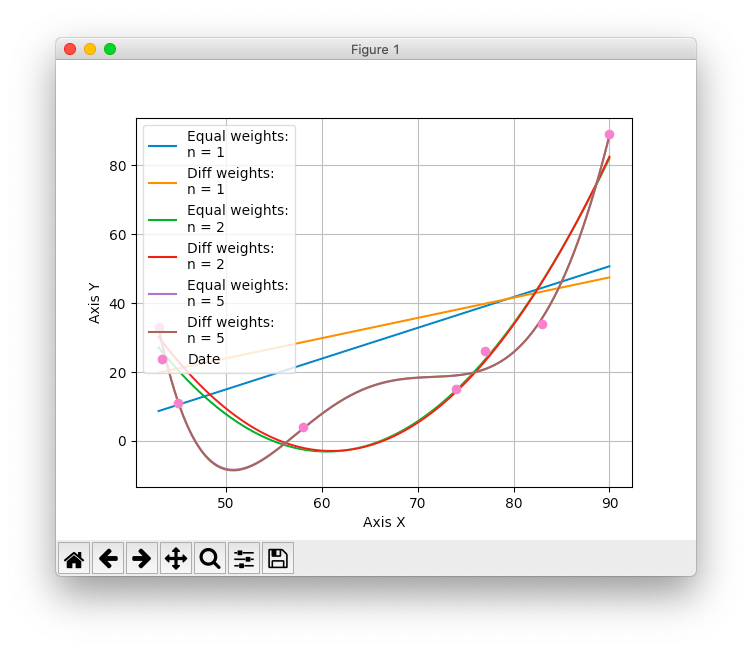
plt.ylabel("Axis Y")

plt.show()

if *self*.\_\_flag\_changed:

*self*.table = changed\_table

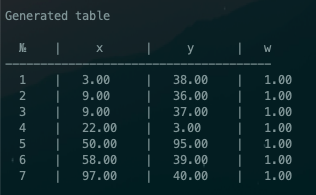
**Пример работы программы**  




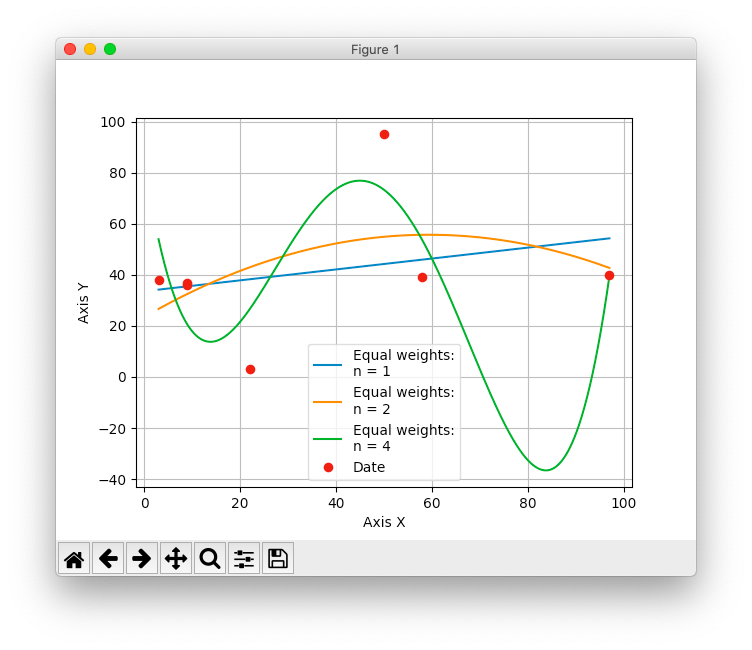
**Результаты**

1. **Одинаковые веса**

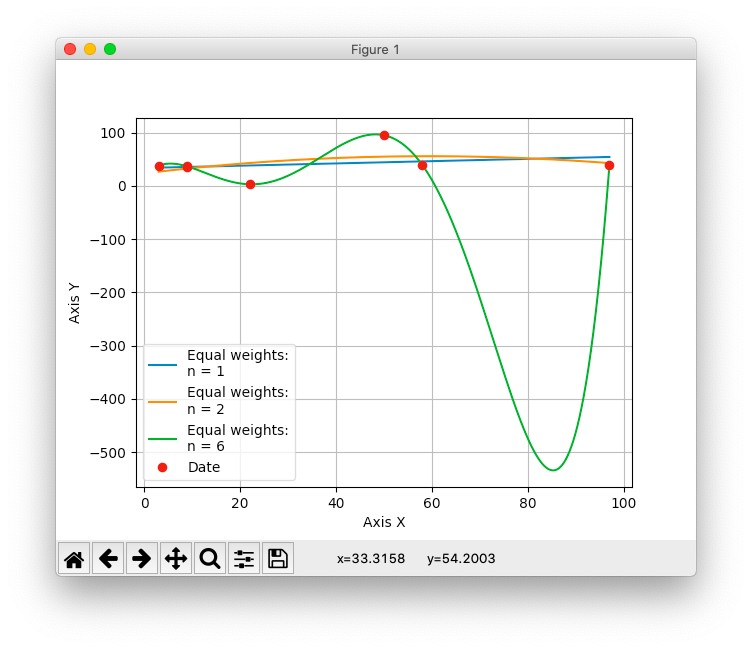
Входная таблица:



Полиномы степеней 1, 2, 4:

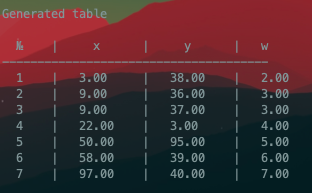


Полиномы степеней 1, 2, 6:

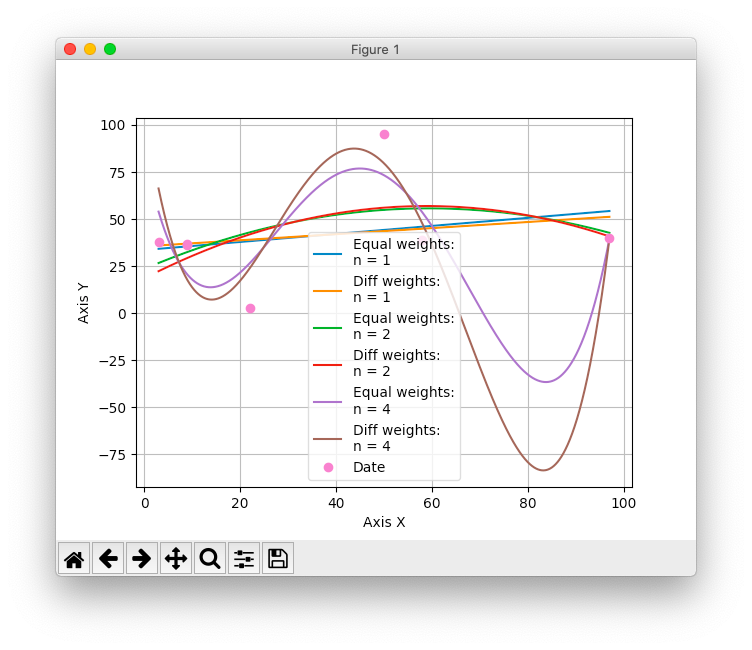


1. **Веса разные**

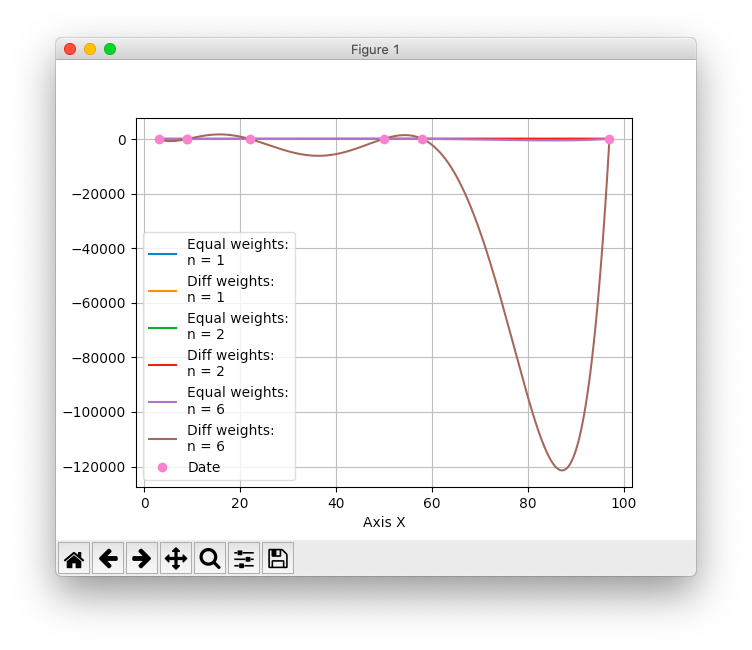
Входная таблица:



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?**

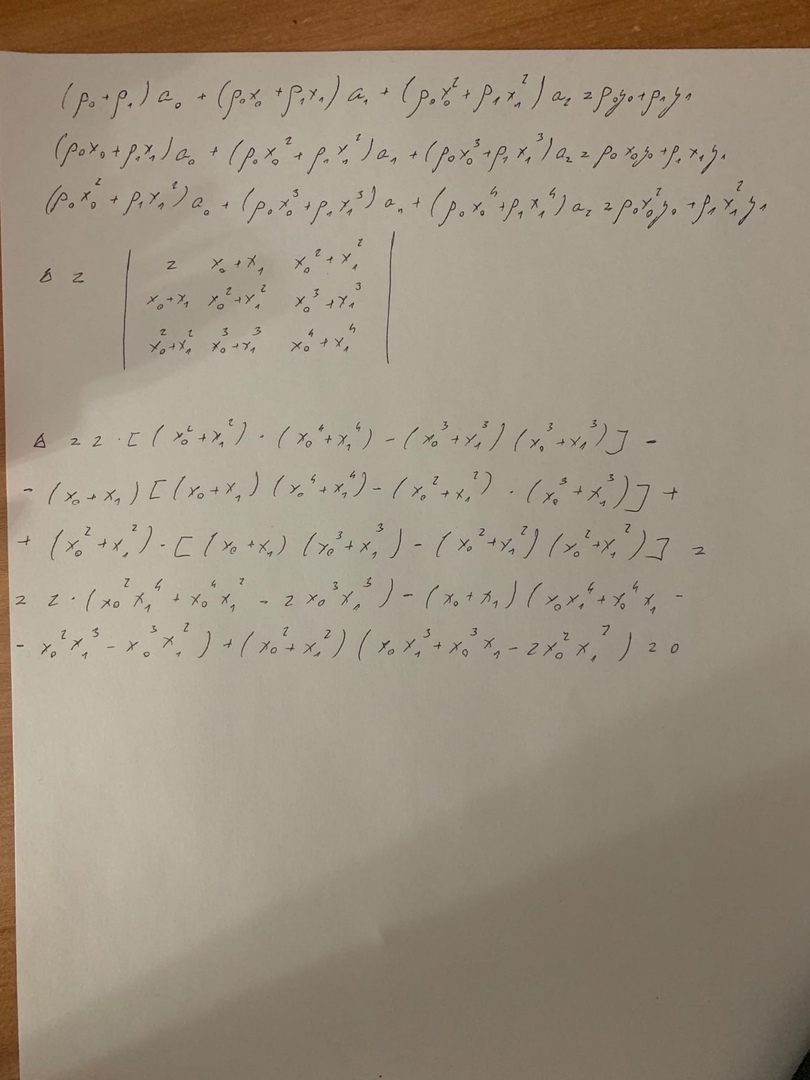
При заданном условии (n >= N) невозможно построить полином n степени (из-за недостатка точек), так как при этом определитель матрицы будет равен нулю. Программа работать будет, то в некоторые моменты может возникнуть непредвиденная ситуация

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

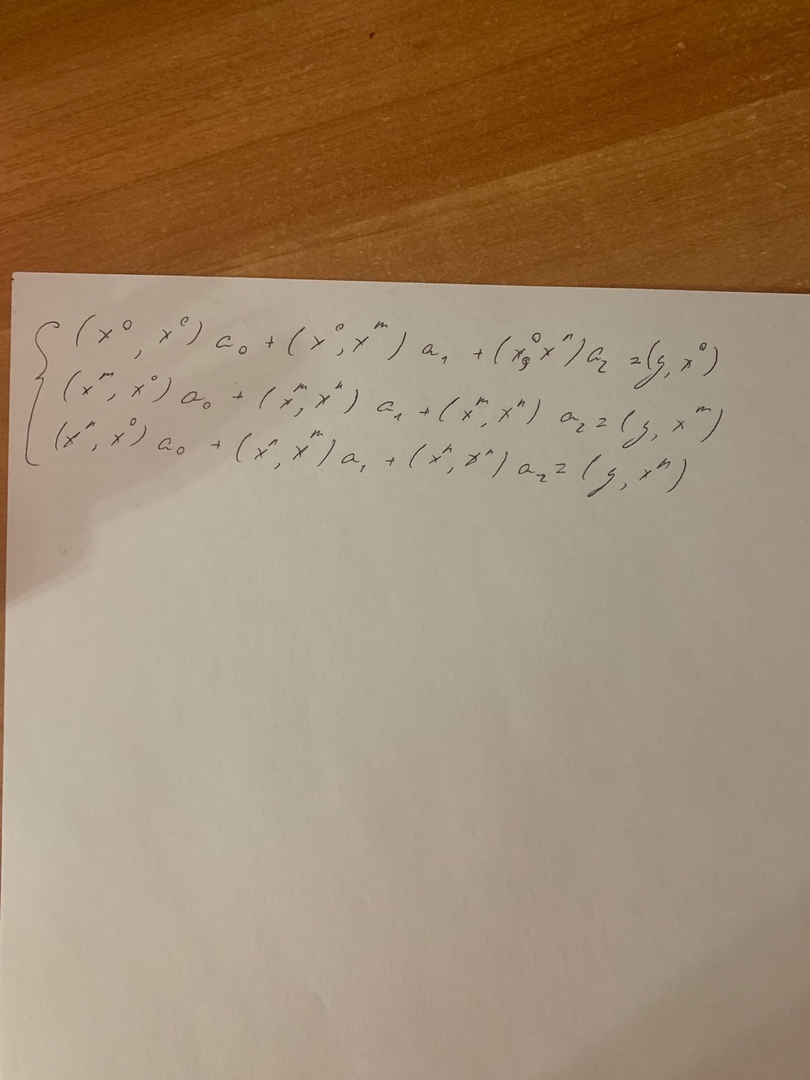
a0 = (∑piyi) / ∑pi ;

где pi — 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.**



**5. Построить СЛАУ при выборочном задании степеней аргумента полинома  , причем степени n и m в этой формуле известны.**



**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