|  |  |
| --- | --- |
| 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 Среднеквадратичное приближение**  **Студент Якуба Д. В.**  **Группа ИУ7-43Б**  **Оценка (баллы)**  **Преподаватель Градов В. М.** |  |

Москва.

2020 г.

Оглавление

[Лабораторная работа по теме «Среднеквадратичное приближение» 2](#_Toc36902845)

[Тема: 2](#_Toc36902846)

[Цель работы: 3](#_Toc36902847)

[Входные данные: 3](#_Toc36902848)

[Выходные данные: 3](#_Toc36902849)

[Описание алгоритма 3](#_Toc36902850)

[Итоговый алгоритм: 4](#_Toc36902851)

[Результат работы программы 5](#_Toc36902852)

[Случай 1. Веса точек равны. 5](#_Toc36902853)

[n = 1 6](#_Toc36902854)

[n = 2 7](#_Toc36902855)

[n = 4 8](#_Toc36902856)

[Случай 2. Веса точек задаются пользователем разными. 9](#_Toc36902857)

[n = 1 9](#_Toc36902858)

[n = 2 11](#_Toc36902859)

[Контрольные вопросы 12](#_Toc36902860)

[1. Что произойдёт при задании степени полинома n = N – 1 12](#_Toc36902861)

[2. Будет ли работать Ваша программа при n >= N? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке? 12](#_Toc36902862)

[3. Получить формулу для коэффициента полинома a0 при степени полинома n = 0. Какой смысл имеет величина, которую представляет данный коэффициент. 13](#_Toc36902863)

[4. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n = N = 2. 14](#_Toc36902864)

[Код программы 14](#_Toc36902865)

Лабораторная работа по теме «Среднеквадратичное приближение»

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.

Входные данные:

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

|  |  |  |
| --- | --- | --- |
| x | y | ρi |
| … | … | … |

2. Степень аппроксимирующего полинома - n.

Выходные данные:

Графики исходной функции, изображённый точками и аппроксимирующий полином. Если пользователь меняет веса в программе – будет выведен поленом с начальными весами точек, а также итоговый полином, с изменёнными весами.

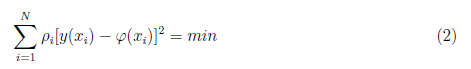
Описание алгоритма

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



где ρi - вес точки; *y(xi) –* исходная функция; *φ(xi) –* множество функций, принадлежащих линейному пространству функции. Суммирование выполняется по всем N узлам заданной функции. Такой вид аппроксимации называют среднеквадратичным приближением.

Требуется найти наилучшее приближение, то есть:



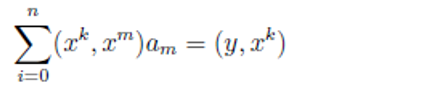
Разложим функцию *φ(x)* по системе линейно независимых функций *φk(x):*



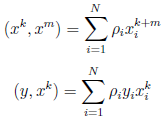
Подставляя (3) в условие (2) получим:



Дифференцируя по ak получаем:



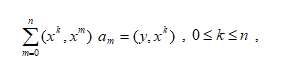
где

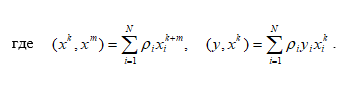


Итоговый алгоритм:

1. Выбирается степень полинома n << N. Обычно степень полинома не превышает 5-6.

2. Составляется система линейных алгебраических уравнений следующего вида:





3. В результате решения СЛАУ находятся коэффициенты полинома (в моём случае используется метод Гаусса).

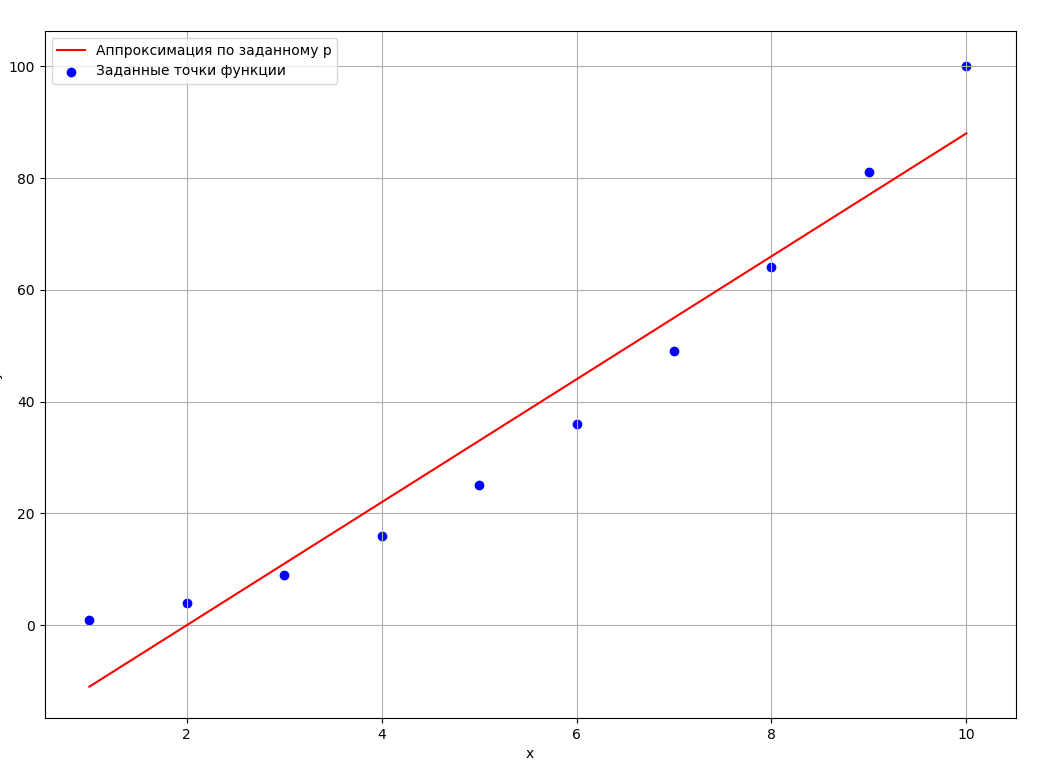
Результат работы программы

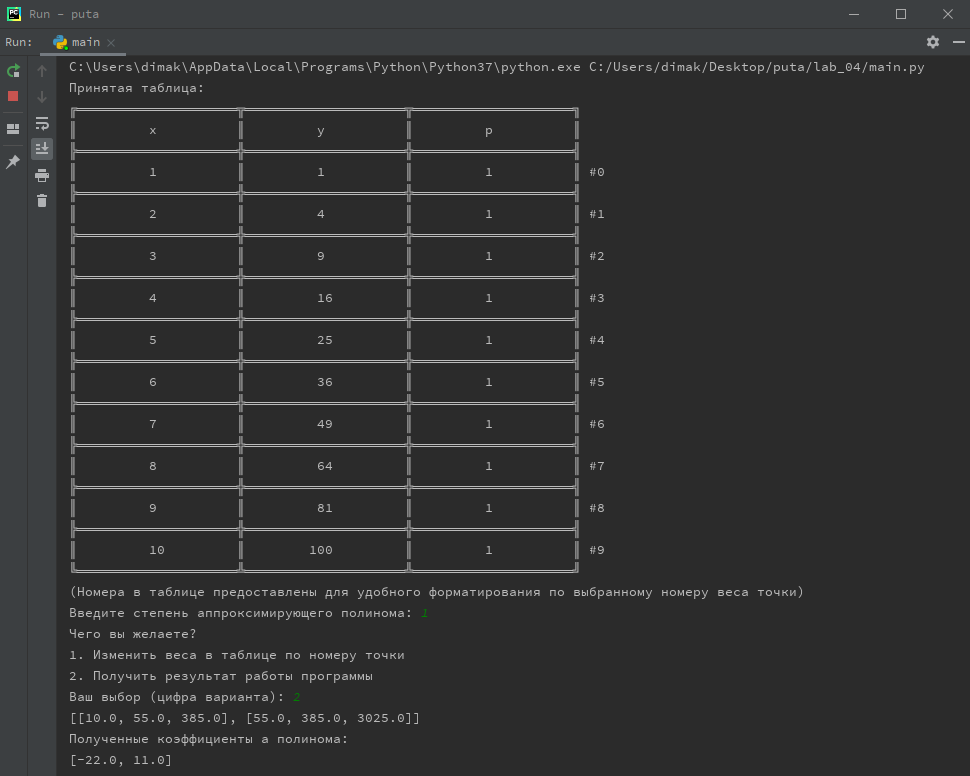
Случай 1. Веса точек равны.

Заданная пользователем таблица:

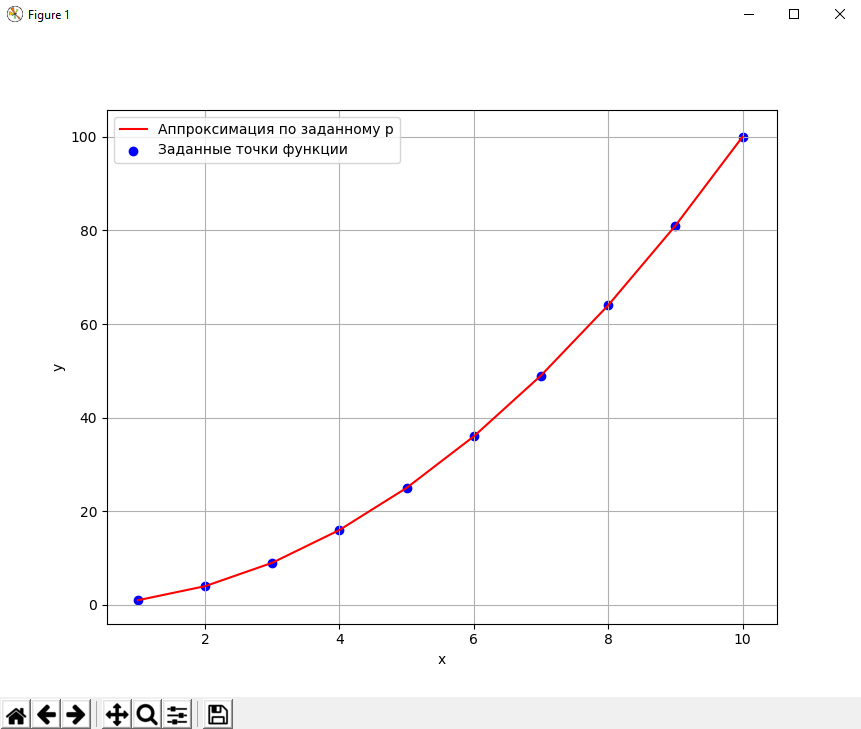
|  |  |  |
| --- | --- | --- |
| *xi* | *yi* | *ρi* |
| 1 | 1 | 1 |
| 2 | 4 | 1 |
| 3 | 9 | 1 |
| 4 | 16 | 1 |
| 5 | 25 | 1 |
| 6 | 36 | 1 |
| 7 | 49 | 1 |
| 8 | 64 | 1 |
| 9 | 81 | 1 |
| 10 | 100 | 1 |

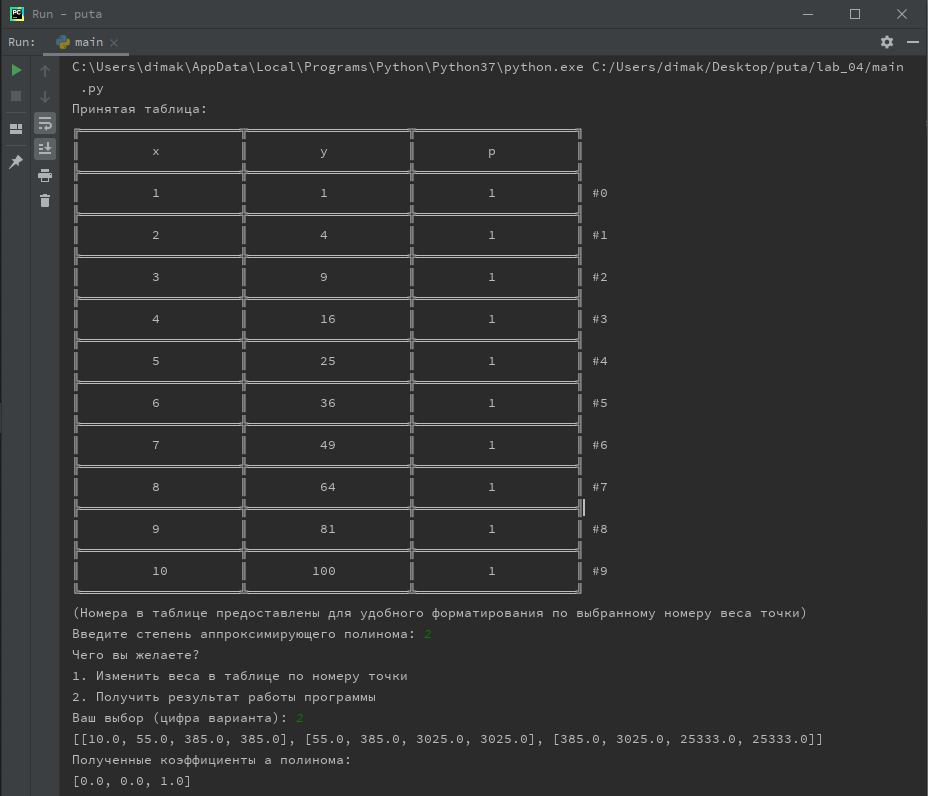
n = 1



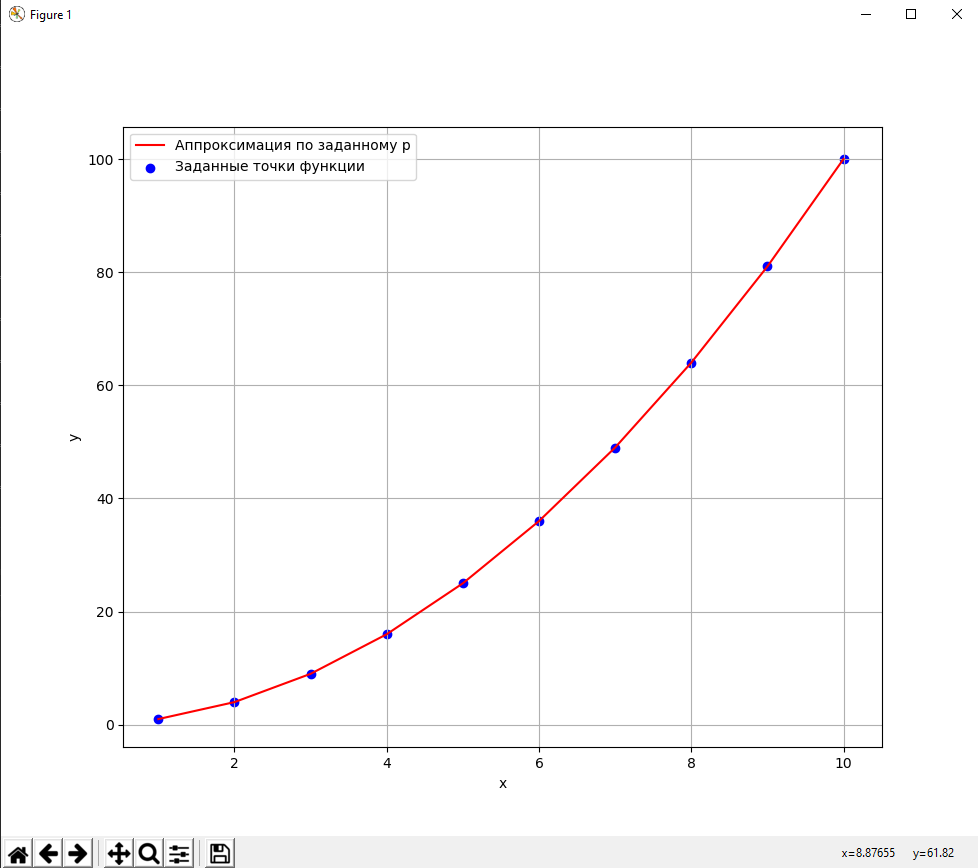


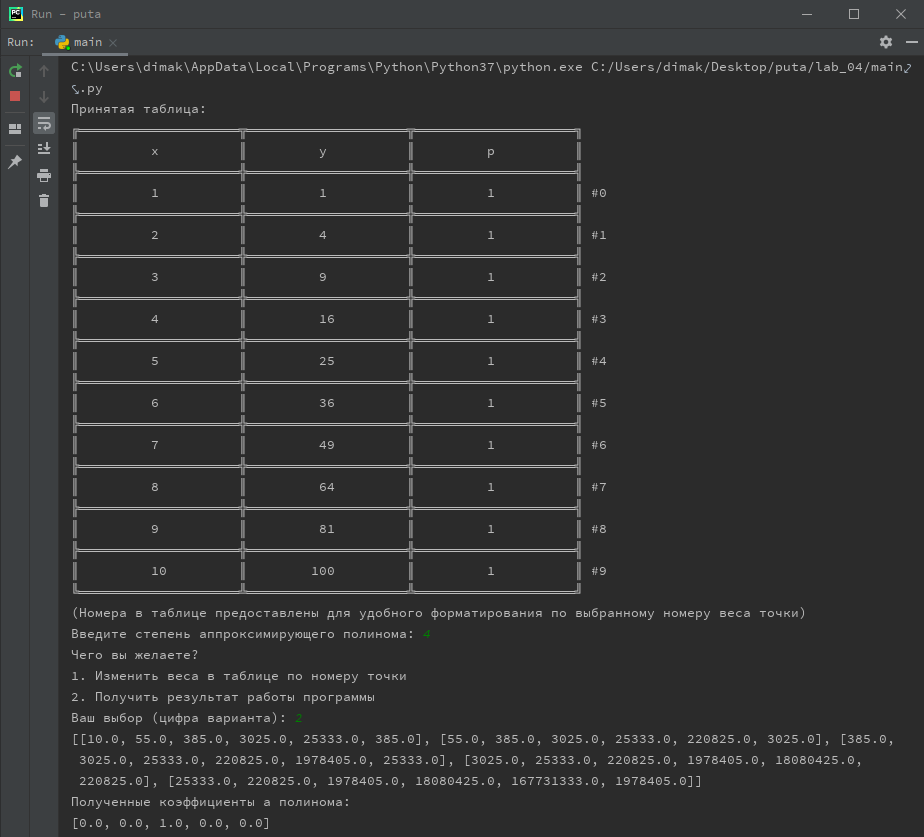
n = 2





n = 4



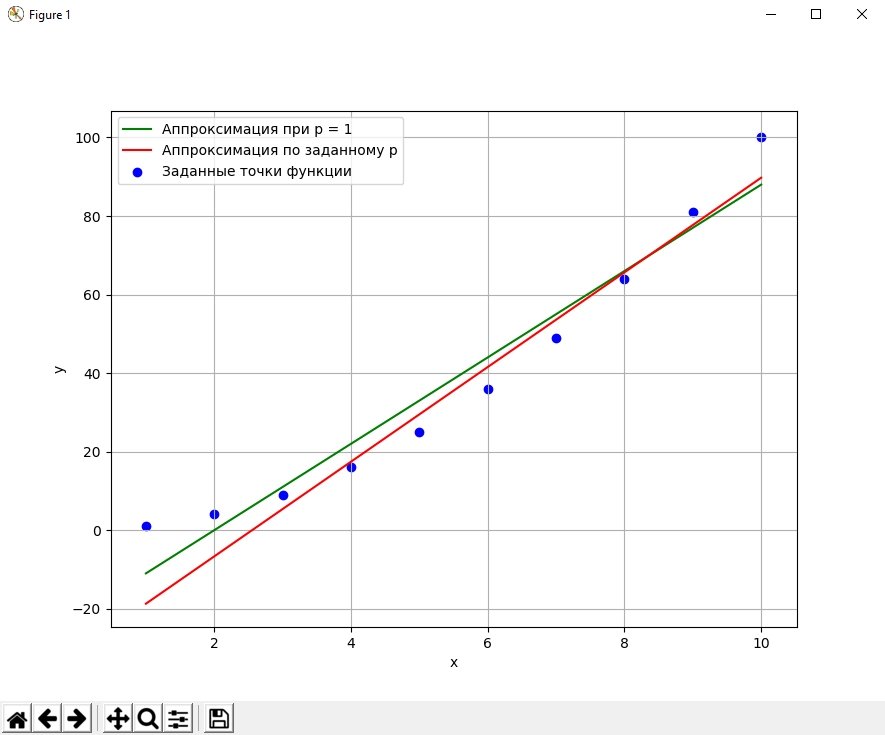


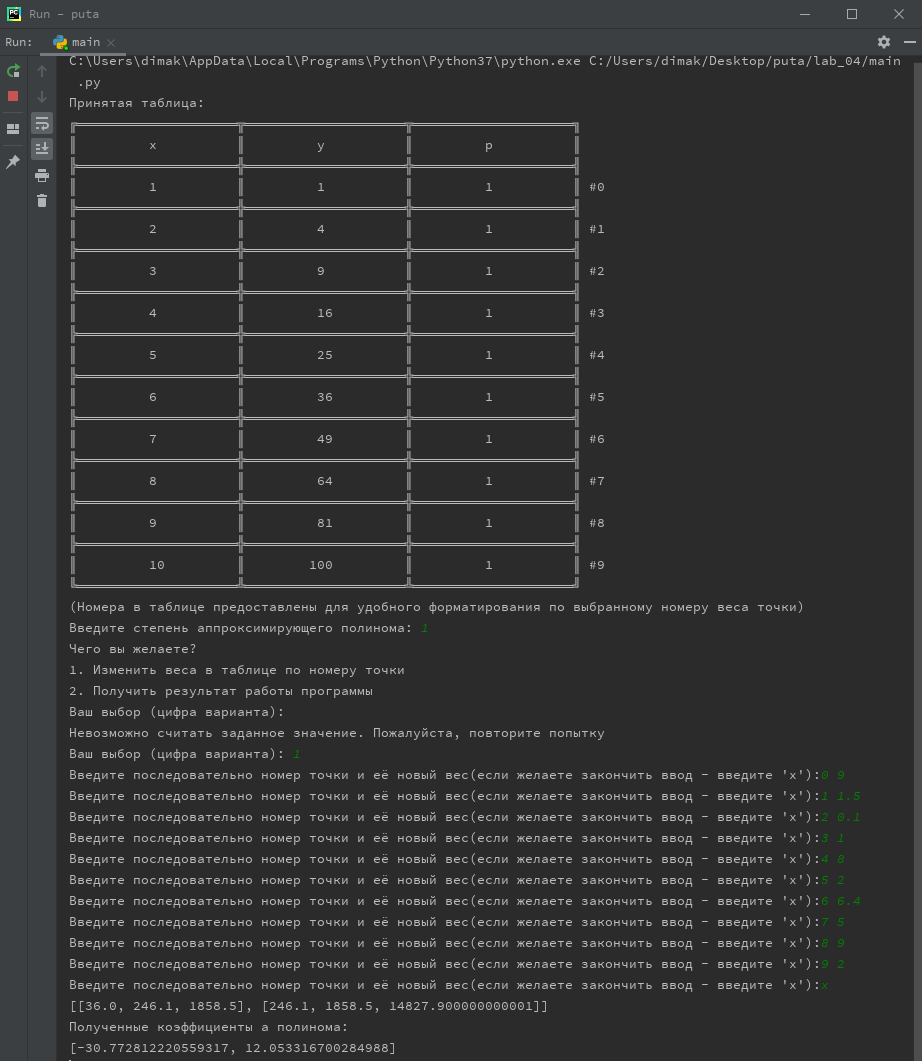
Случай 2. Веса точек задаются пользователем разными.

Заданная пользователем таблица:

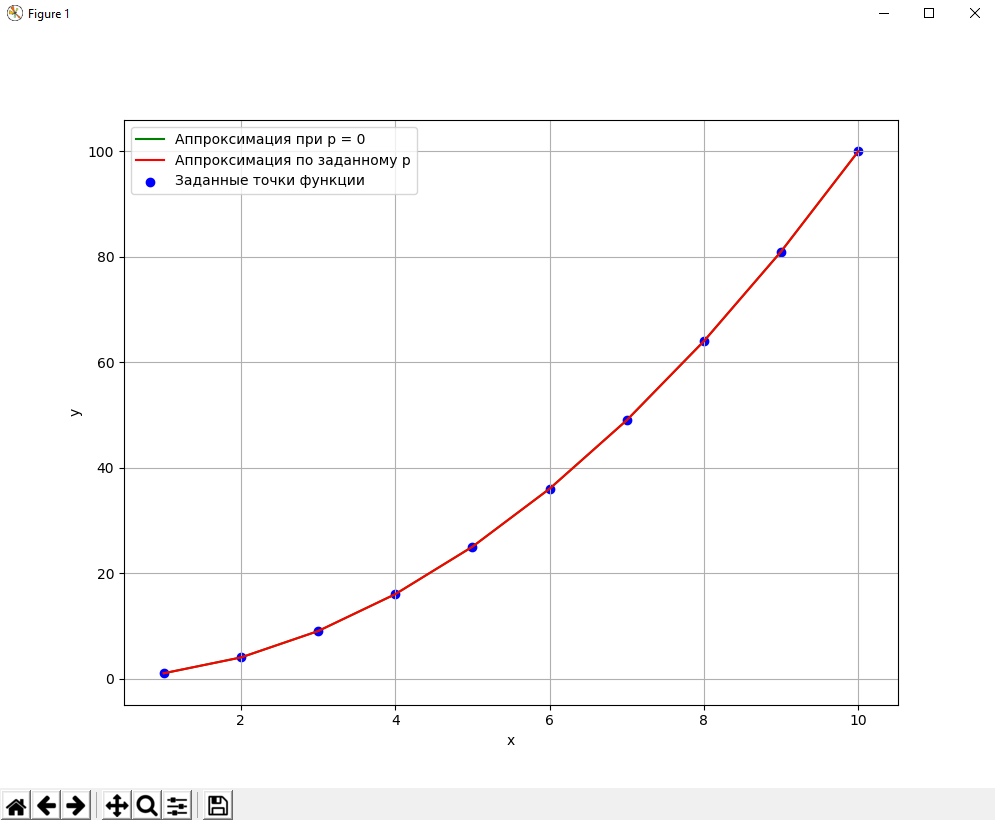
|  |  |  |
| --- | --- | --- |
| xi | yi | ρi |
| 1 | 1 | 9 |
| 2 | 4 | 1.5 |
| 3 | 9 | 0.1 |
| 4 | 16 | 1 |
| 5 | 25 | 8 |
| 6 | 36 | 2 |
| 7 | 49 | 6.4 |
| 8 | 64 | 5 |
| 9 | 81 | 9 |
| 10 | 100 | 2 |

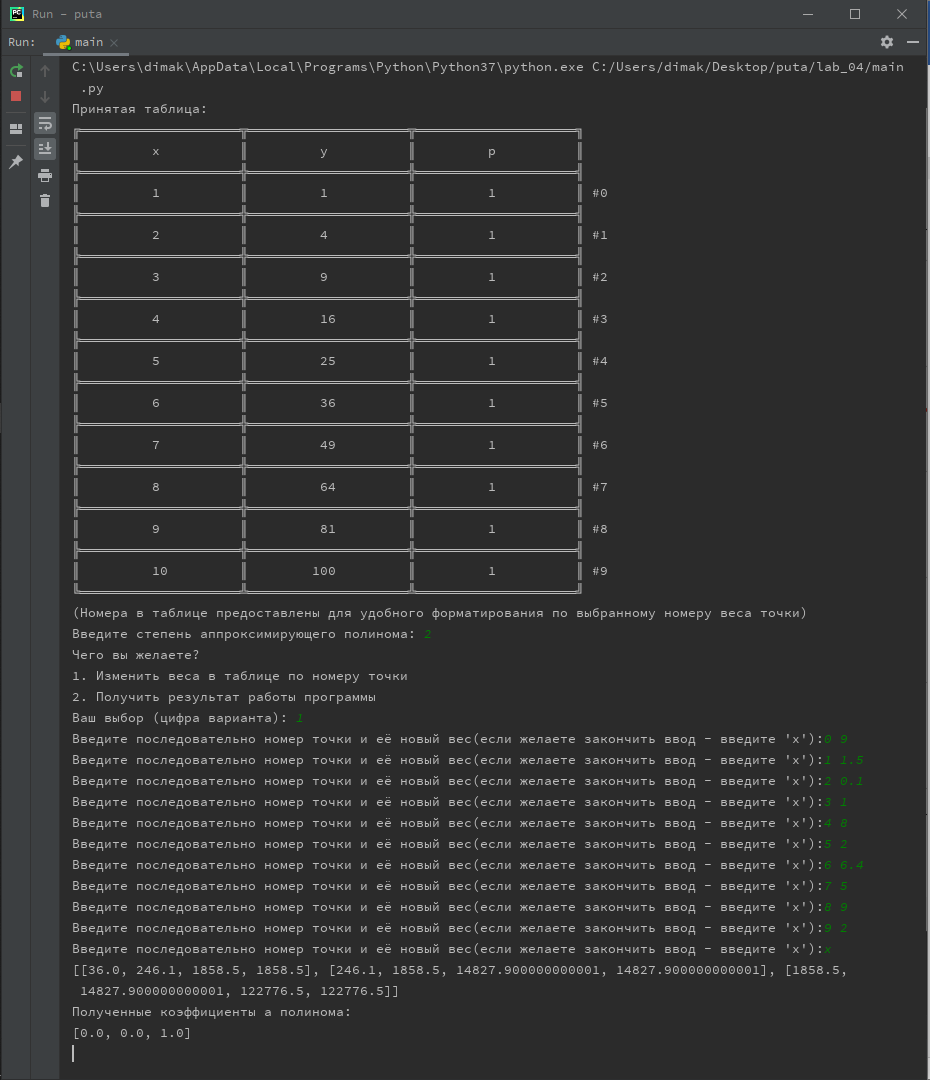
n = 1

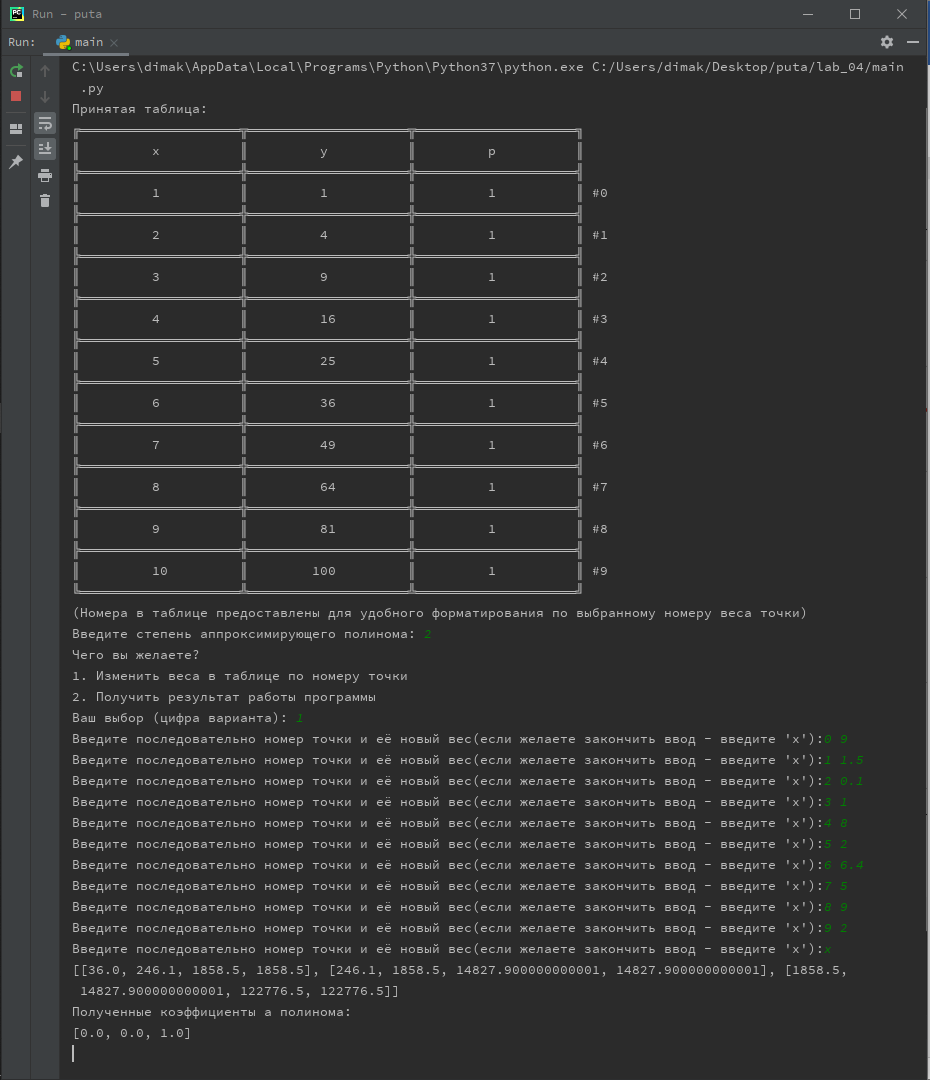




n = 2







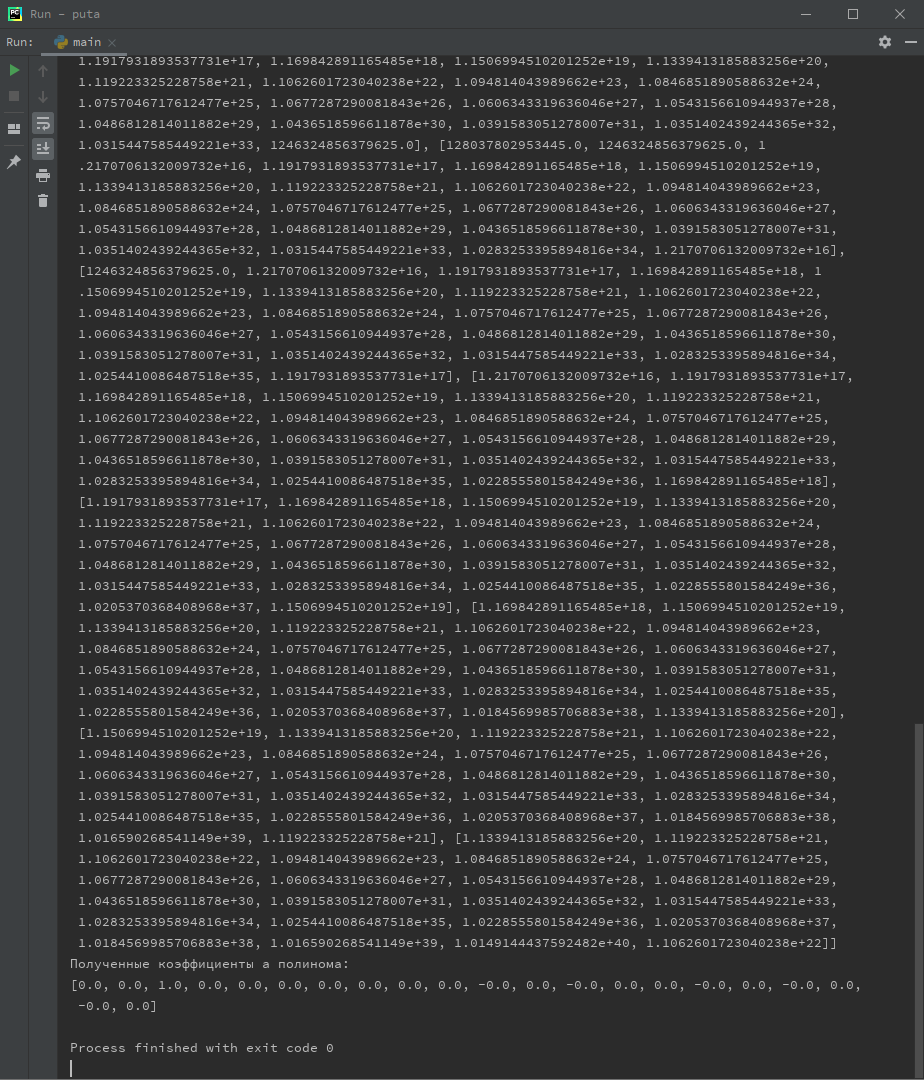
Контрольные вопросы

1. Что произойдёт при задании степени полинома n = N – 1

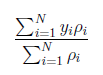
Ответ: будет построен полином по всем заданным в таблице точкам

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

Ответ: программа работать будет, если значение степени полинома будет не слишком большой, так как начнутся вычисления над погрешностями (смотреть пример на сл. рисунке). Но данную ситуацию можно назвать ошибочной, так как в данном случае определитель СЛАУ будет равен нулю. Чтобы отследить такую ошибку, при вводе степени полинома пользователем следует проверить, меньше ли она количества точек в переданной таблице.



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



Полученная формула:

Математическое значение – математическое ожидание.

**ВАЖНО: в знаменателе сумма по весам точек, а не просто количество точек!!!**

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

Ответ: пусть имеется заданная таблица таблица

|  |  |  |
| --- | --- | --- |
| x | y | ρ |
| x0 | y0 | ρ0 |
| x1 | y1 | ρ1 |

Тогда запишем следующую систему:

+

Так как определитель матрицы равняется нулю, следовательно, система является линейно-зависимой, а, следовательно, решений она не имеет.

Код программы

from matplotlib.pyplot import \*  
  
tableName = "table.txt"  
  
  
def matrLeftX(xCoordinates, k, m, weight, N):  
 back = 0  
 for i in range(N):  
 back += weight[i] \* pow(xCoordinates[i], k + m)  
 return back  
  
  
def matrRightY(xCoordinates, yCoordinates, k, weight, N):  
 back = 0  
 for i in range(N):  
 back += weight[i] \* yCoordinates[i] \* pow(xCoordinates[i], k)  
  
 return back  
  
  
def gaussMethod(matrix):  
 for col in range(len(matrix[0])):  
 for row in range(col + 1, len(matrix)):  
 r = [(rowValue \* (-(matrix[row][col] / matrix[col][col]))) for rowValue in matrix[col]]  
 matrix[row] = [sum(pair) for pair in zip(matrix[row], r)]  
 ans = []  
 matrix.reverse()  
 for sol in range(len(matrix)):  
 if sol == 0:  
 ans.append(matrix[sol][-1] / matrix[sol][-2])  
 else:  
 inner = 0  
 for x in range(sol):  
 inner += (ans[x] \* matrix[sol][-2 - x])  
 ans.append((matrix[sol][-1] - inner) / matrix[sol][-sol - 2])  
 ans.reverse()  
 return ans  
  
  
def makeMatrixOfSLAE(matrix, xCoordinates, yCoordinates, weight, degree):  
 N = len(xCoordinates)  
 for k in range(degree + 1):  
 matrix.append([])  
 for m in range(degree + 1):  
 matrix[k].append(matrLeftX(xCoordinates, k, m, weight, N))  
 matrix[k].append(matrRightY(xCoordinates, yCoordinates, k, weight, N))  
  
  
def makeApproximatedMasY(approxY, xCoordinates, decision):  
 for i in range(len(xCoordinates)):  
 currentX = 1  
 lineSum = 0  
 for j in range(len(decision)):  
 lineSum += decision[j] \* currentX  
 currentX \*= xCoordinates[i]  
 approxY.append(lineSum)  
  
  
def changeWeight(xCoordinates, yCoordinates, weight, degree):  
 matrix = list()  
 makeMatrixOfSLAE(matrix, xCoordinates, yCoordinates, weight, degree)  
 decision = gaussMethod(matrix)  
  
 approxY = []  
 makeApproximatedMasY(approxY, xCoordinates, decision)  
 plot(xCoordinates, approxY, c = "green", label = "Аппроксимация при p = 1")  
  
 inp = 0  
 while inp != 'x':  
 inp = input("Введите последовательно номер точки и её новый вес(если желаете закончить ввод - введите 'x'):")  
 if inp == 'x':  
 return  
 try:  
 inp = inp.split()  
 inp[0] = int(inp[0])  
 inp[1] = float(inp[1])  
 except Exception:  
 print("Значение невозможно считать")  
 else:  
 if inp[0] > 0 and inp[0] < len(weight):  
 try:  
 weight[inp[0]] = inp[1]  
 except Exception:  
 print("Значение невозможно считать")  
  
  
def main():  
 file = open(tableName, 'r')  
  
 xCoordinates = []  
 yCoordinates = []  
 weight = []  
 i = 0  
  
 print("Принятая таблица:\n╔════════════════════╦════════════════════╦════════════════════╗")  
 print("║{:^20}║{:^20}║{:^20}║".format('x', 'y', 'p'))  
 for line in file:  
 print("╠════════════════════╬════════════════════╬════════════════════╣")  
 splited = line.split()  
 xCoordinates.append(float(splited[0]))  
 yCoordinates.append(float(splited[1]))  
 weight.append(float(splited[2]))  
 print("║{:^20}║{:^20}║{:^20}║ #{}".format(splited[0], splited[1], splited[2], i))  
 i += 1  
 file.close()  
 print("╚════════════════════╩════════════════════╩════════════════════╝\n(Номера в таблице предоставлены для удобного форматирования по выбранному номеру веса точки)")  
  
 degree = -1  
 while degree < 0:  
 try:  
 degree = int(input("Введите степень аппроксимирующего полинома: "))  
 except Exception:  
 print("Невозможно считать заданное значение. Пожалуйста, повторите попытку")  
  
 print("Чего вы желаете?\n1. Изменить веса в таблице по номеру точки\n2. Получить результат работы программы")  
 choice = 0  
 while choice <= 0 or choice >= 3:  
 try:  
 choice = int(input("Ваш выбор (цифра варианта): "))  
 except Exception:  
 print("Невозможно считать заданное значение. Пожалуйста, повторите попытку")  
 if choice == 1:  
 changeWeight(xCoordinates, yCoordinates, weight, degree)  
  
 matrix = list()  
 makeMatrixOfSLAE(matrix, xCoordinates, yCoordinates, weight, degree)  
 print(matrix)  
 decision = gaussMethod(matrix)  
 print("Полученные коэффициенты a полинома:")  
 print(decision)  
  
 approxY = []  
 makeApproximatedMasY(approxY, xCoordinates, decision)  
  
 for i in range(len(xCoordinates) - 1):  
 scatter(xCoordinates[i], yCoordinates[i], c = "blue")  
 scatter(xCoordinates[len(xCoordinates) - 1], yCoordinates[len(xCoordinates) - 1], c = "blue", label = "Заданные точки функции")  
 plot(xCoordinates, approxY, c = "red", label = "Аппроксимация по заданному p")  
 legend()  
 grid()  
 xlabel('x')  
 ylabel('y')  
 show()  
  
  
main()