|  |  |
| --- | --- |
| 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.**

**«Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения»**

Студент **Леонов Владислав Вячеславович**

Группа **ИУ7-46Б**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Леонов В.В.

*подпись, дата фамилия, и.о.*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Градов В.М.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

**Оглавление**

[Цель работы 3](#_Toc68624679)

[Исходные данные 3](#_Toc68624680)

[Описание алгоритма 3](#_Toc68624681)

[Код программы 5](#_Toc68624682)

[Результаты работы 7](#_Toc68624683)

[Ответы на контрольные вопросы 10](#_Toc68624684)

# 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 с количеством узлом *N*. Сформировать таблицу самостоятельно со случайным разбросом точек.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

2. Степень аппроксимирующего полинома .

# Описание алгоритма

Под близостью в среднем исходной и аппроксимирующей функций будем понимать результат оценки функции:

где – вес точки. Суммирование выполняется по всем *N* узлам.

Задача состоит в том, чтобы найти функцию , для которой будет справедливо соотношение:

Разложим функцию по системе линейно независимых функций :

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

Получаем:

Дифференцируя это выражение по и приравнивая производные к нулю, найдем:

Наиболее употребительный вариант метода наименьших квадратов соответствует случаю степенного вида функций , т.е. , причем . Обычно степень полинома не превышает 5-6.

Система уравнений при этом принимает вид:

где

# Код программы

Код программы представлен ниже.

|  |
| --- |
| Файл ***main.py*** |
| **from** data **import** Data      **def** main():  data = Data()  data.load()  data.ls\_perform()      **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

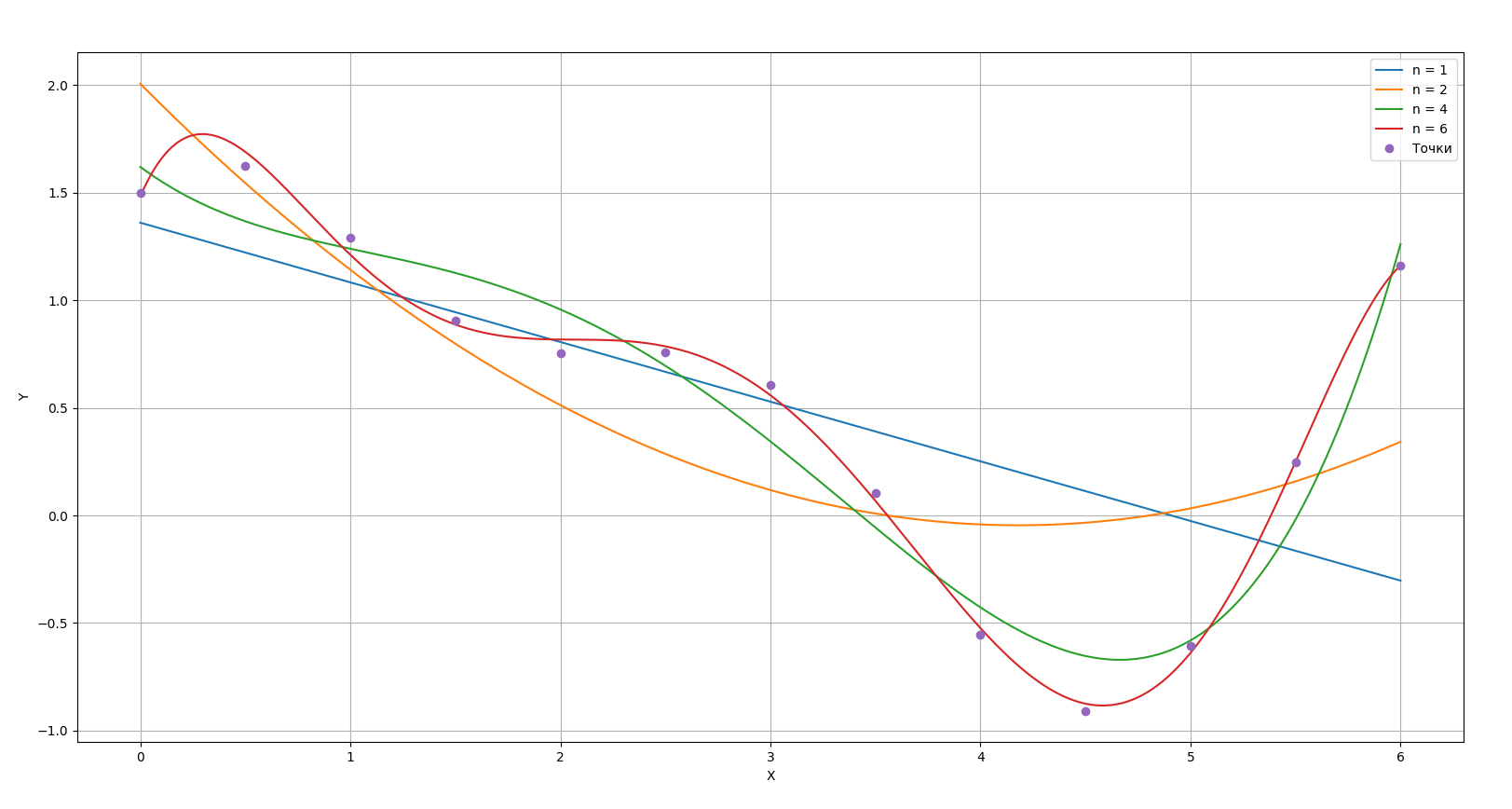
|  |
| --- |
| Файл ***data.py*** |
| **import** config  **import** numpy  **import** matplotlib.pyplot **as** plt      **class** Dot():  **def** \_\_init\_\_(self, x, y, p):  self.x = x  self.y = y  self.p = p      **class** Data():  **def** \_\_init\_\_(self):  self.data: List[Dot] = []    **def** load(self):  f = open(config.DEFAULT\_FILE\_IN)  **for** line **in** f:  dot = Dot(\*[float(num) **for** num **in** line.split(" ")])  self.data.append(dot)  f.close()    **def** ls\_perform(self):  **def** calc\_coeff(deg):  coeff = 0  **for** dot **in** self.data:  coeff += dot.p \* dot.x\*\*deg  **return** coeff    **def** sole\_sollution(mtr):  **for** i **in** range(len(mtr)):  **for** j **in** range(len(mtr)):  **if** i == j: **continue**  mult = mtr[j][i] / mtr[i][i]  **for** k **in** range(len(mtr) + 1):  mtr[j][k] -= mult \* mtr[i][k]  **for** i **in** range(len(mtr)):  mult = mtr[i][i]  **for** j **in** range(len(mtr[i])):  mtr[i][j] /= mult  **return** [mtr[i][-1] **for** i **in** range(len(mtr))]    **def** add\_sole\_mtr(mtr):  **for** i **in** range(len(mtr)):  res = 0  **for** dot **in** self.data:  res += dot.p \* dot.y \* dot.x\*\*i  mtr[i].append(res)    **def** create\_sole\_mtr(deg):  mtr = [[calc\_coeff(j + i) **for** i **in** range(deg + 1)]  **for** j **in** range(deg + 1)]  add\_sole\_mtr(mtr)  **return** mtr    **def** plot\_add\_polinom(a\_coeffs, deg):  plot\_x, plot\_y = [], []  step = (self.data[-1].x - self.data[0].x) / 1000  **for** x **in** numpy.arange(self.data[0].x, self.data[-1].x + step,  step):  plot\_x.append(x)  y = 0  **for** i **in** range(len(a\_coeffs)):  y += a\_coeffs[i] \* x\*\*i  plot\_y.append(y)  plt.plot(plot\_x, plot\_y, label="n = {0}".format(deg))    **def** plot\_add\_dots():  plot\_x = [dot.x **for** dot **in** self.data]  plot\_y = [dot.y **for** dot **in** self.data]  plt.plot(plot\_x, plot\_y, "o", label="Точки")    **def** plot\_draw():  plt.legend()  plt.xlabel("X")  plt.ylabel("Y")  plt.grid()  plt.show()    **for** deg **in** config.POL\_DEGS:  sole\_mtr = create\_sole\_mtr(deg)  a\_coeffs = sole\_sollution(sole\_mtr)  plot\_add\_polinom(a\_coeffs, deg)  plot\_add\_dots()  plot\_draw() |

# Результаты работы

***1. Веса точек одинаковые. Степени аппроксимирующего полинома . Исходная функция***

Исходная таблица:

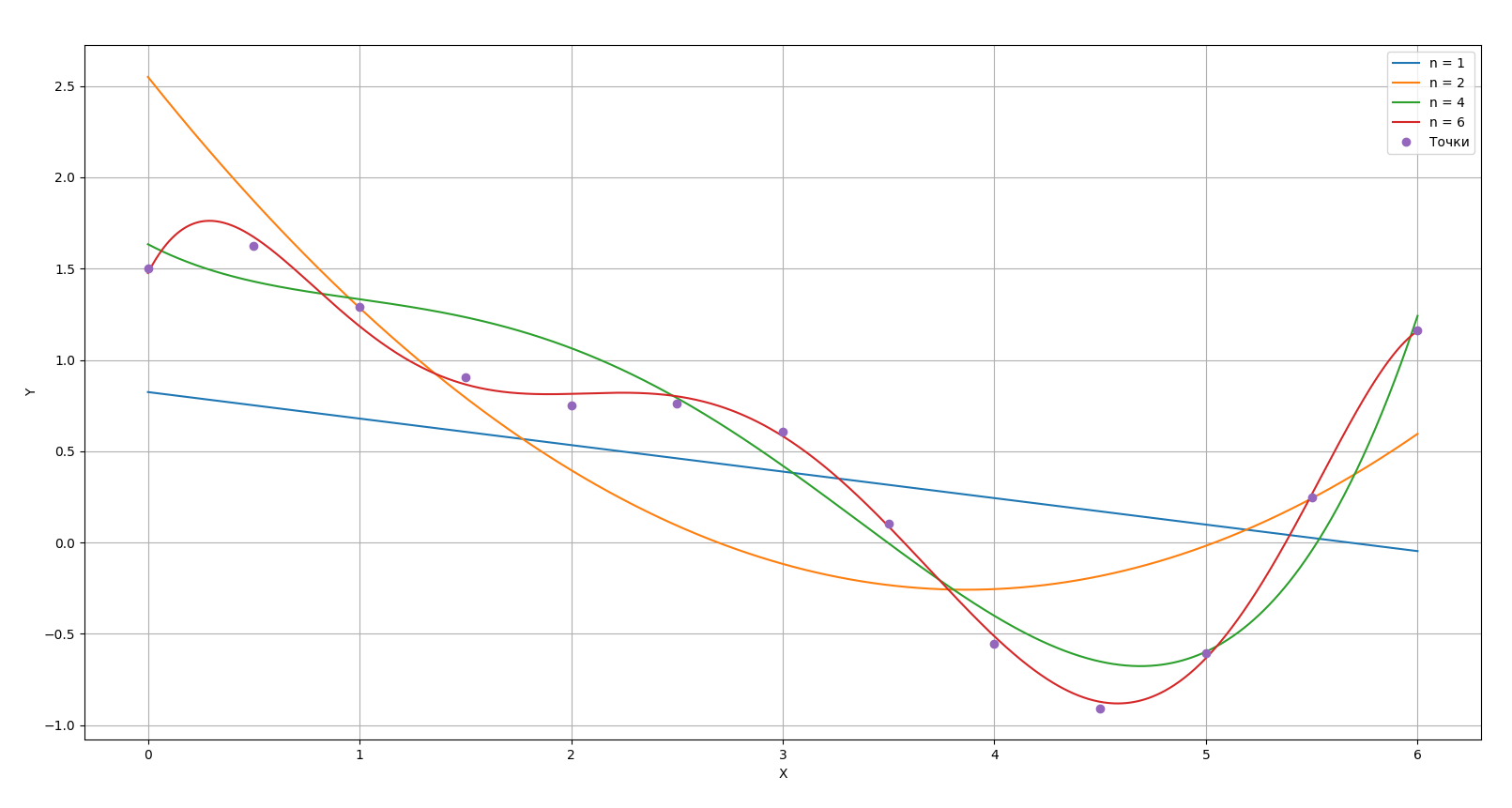
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**

***2. Веса точек разные. Степени аппроксимирующего полинома . Исходная функция***

Исходная таблица:

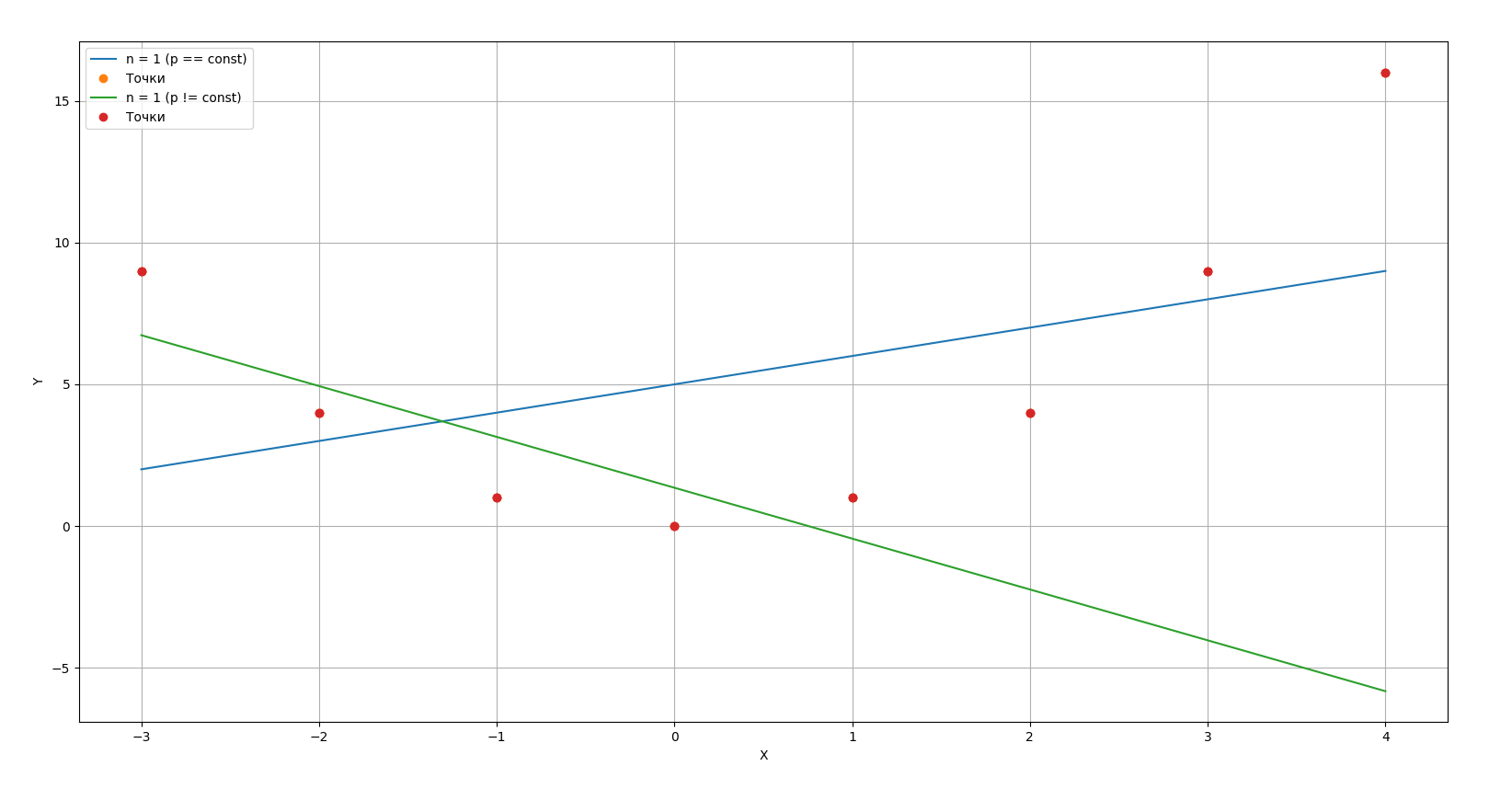
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |



***3. Сравнение влияния весов точек на положение прямой, аппроксимирую-щей один и тот же набор точек.***

Исходные таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



# Ответы на контрольные вопросы

***1. Что произойдет при задании степени полинома ?***

Программа построит полином, проходящий через все узлы, не смотря на то, какие будут веса у узлов.

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

Программа будет работать некорректно, т.к. тогда решения СЛАУ не будут линейно-независимыми (определить соответствующей матрицы будет равен 0). Для решения данной проблемы, можно анализировать степень в самом начале и, например, выдавать соответствующее сообщение или строить полином максимально возможной степени (. Более того, по точкам нельзя однозначно построить полином степени

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

где означает математическое ожидание.

***4. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда . Принять все .***

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

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

Для определения коэффициентов и можно воспользоваться перебором, , где N – количество узлов. Во время перебора нужно найти функцию, наилучшим образом аппроксимирующую заданные узлы, т.е. для которой: