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 (ИУ-7)»

Лабораторная работа №4

Тема: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

Выполнил: **Варин Д.В.**

Группа: **ИУ7-46Б**

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.

1. **Исходные данные**

   1. Таблица функции с весами P, с количеством узлов N.

   (Сформировать таблицу самостоятельно со случайным разбросом точек.)

   2. Степень аппроксимирующего полинома — n.

2. **Требования к результатам**

   - Графики: точки — заданная табличная функция, кривые- найденные полиномы.

   - Обязательно приводить таблицы, по которым работала программа.

   При каких исходных условиях надо представить результаты в отчете?

1. Веса всех точек одинаковы и равны, например, единице *(Таблица 1)*. Обязательно построить полиномы степеней n=1 и 2. Можно привести результаты и при других степенях полинома, однако, не загромождая сильно при этом рисунок.

2. Веса точек разные *(Таблица 2)*. Продемонстрировать, как за счет назначения весов точкам можно изменить положение на плоскости прямой линии (полином первой степени), аппроксимирующей один и тот же набор точек (одну таблицу y(x)). Например, назначая веса узлам в таблице изменить знак углового коэффициента прямой. На графике в итоге должны быть представлены точки исходной функции и две аппроксимирующие их прямые линии. Одна отвечает значениям P =1 для всех узлов, а другая- назначенным разным весам точек. Информацию о том, какие именно веса были использованы в расчете обязательно указать, чтобы можно было проконтролировать работу программы (лучше это сделать в виде таблицы).

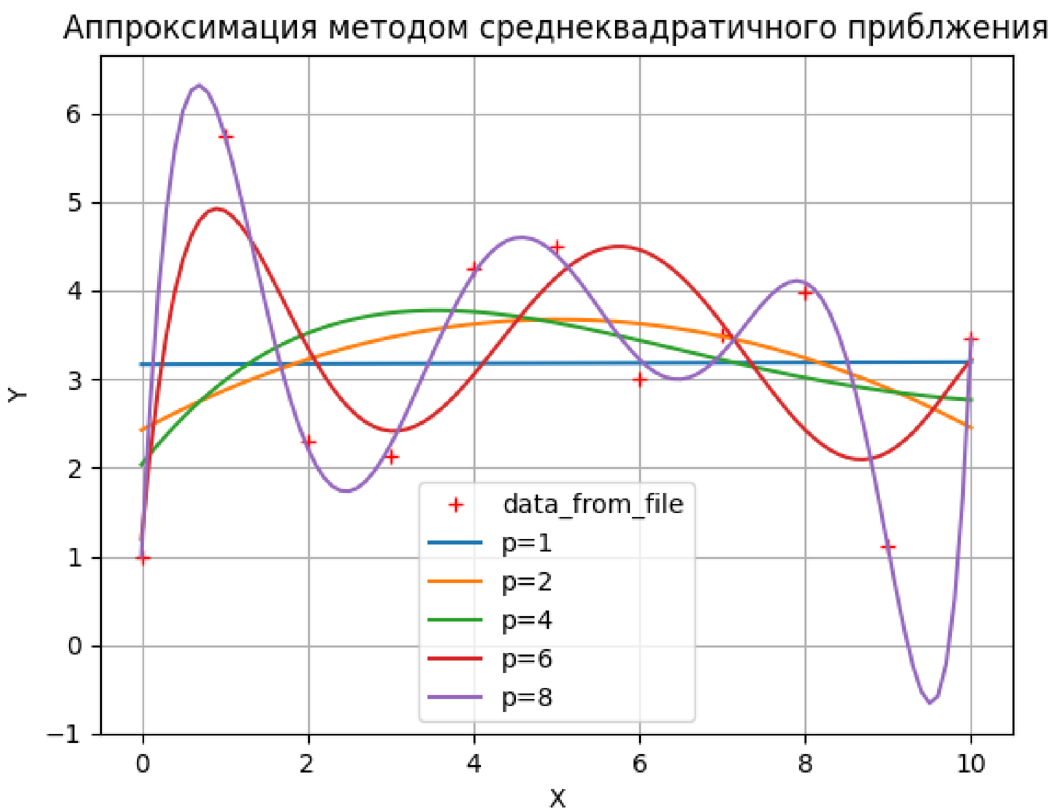
3. **Код программы и результаты**

Программа написана на языке ***Python3*** с использованием библиотеки ***matplotlib*** для отрисовки графиков.

1. Таблица 1.

Веса всех точек равны 1.
Количество точек n = 10
Таблица 1

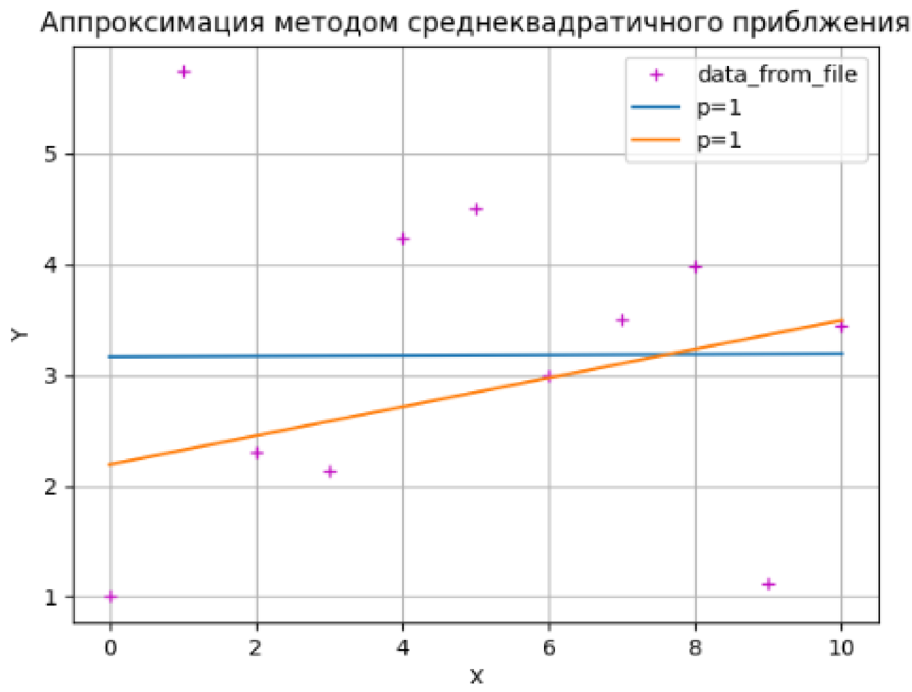
X	*Y*	*P*
0.00	1.00	1
1.00	5.74	1
2.00	2.30	1
3.00	2.14	1
4.00	4.24	1
5.00	4.50	1
6.00	3.00	1
7.00	3.50	1
8.00	3.98	1
9.00	1.12	1
10.00	3.45	1



2.

Таблица 2
Сравнение, как за счет назначения весов точкам можно изменить положение на плоскости прямой линии
Используется таблица:

X	Y	P	P_{new}
0.00	1.00	1	1
1.00	5.74	1	1
2.00	2.30	1	1
3.00	2.14	1	30
4.00	4.24	1	1
5.00	4.50	1	1
6.00	3.00	1	30
7.00	3.50	1	1
8.00	3.98	1	1
9.00	1.12	1	1



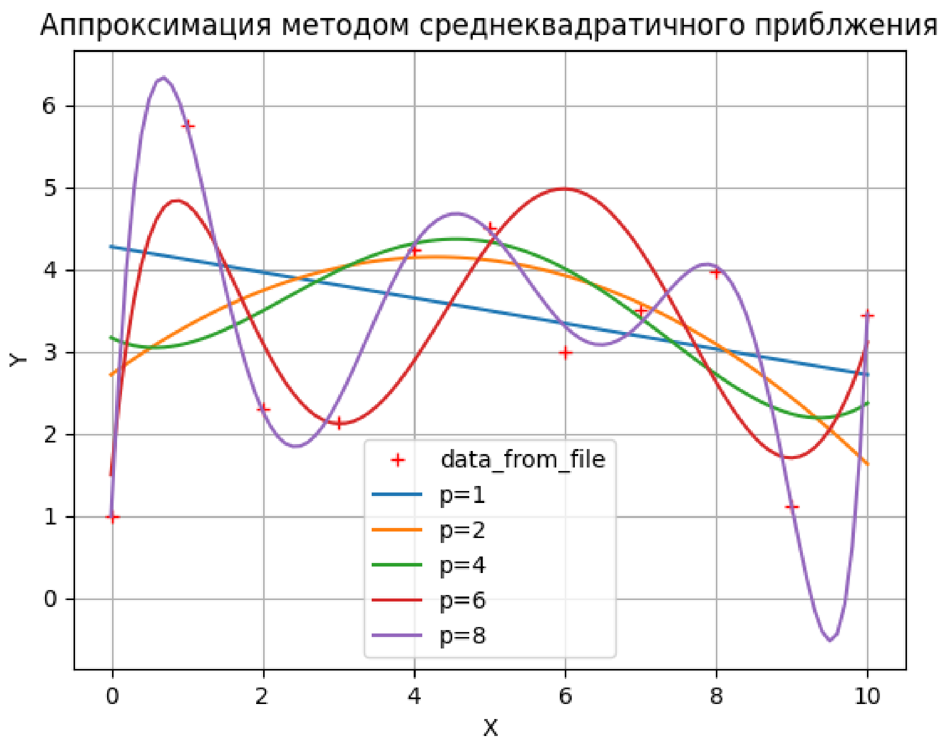
На графике синяя прямая — исходная, желтая — с измененными весами. Как можно заметить, наклон прямой изменился.

3. Таблица 3

Веса всех точек разные .
Количество точек n = 10

Таблица 2

X	Y	P
0.00	1.00	3
1.00	5.74	7
2.00	2.30	11
3.00	2.14	1
4.00	4.24	4
5.00	4.50	29
6.00	3.00	5
7.00	3.50	8
8.00	3.98	12
9.00	1.12	17
10.00	3.45	6



Изменив веса точек, наклон прямой также изменился.

Модуль ***rms_approx.py*** *— реализация алгоритма аппроксимации и решения СЛАУ.*

```python
from __future__ import annotations
from utils import Point

import numpy as np


class Approximator:
    def __init__(self):
        self.coeffs = []

    def get_coeffs(self, mat: list[list[float]]) -> Approximator:
        self.coeffs = [mat[i][len(mat)] for i in range(len(mat))]

        return self

    def build(self, pl: list[Point]) -> list[Point]:
        dots = []

        for i in np.arange(pl[0].x, pl[-1].x + 0.1, 0.1):
            d = Point(i, 0, 0)

            for j in range(len(self.coeffs)):
                d.y += d.x ** j * self.coeffs[j]

            dots += [d]

        return dots


class SLAU:
    mat: list[list[float]]
    n: int

    def build(self, pl: list[Point], _n: int) -> SLAU:
        self.n = _n
        self.mat = [[0 for i in range(self.n + 2)] for i in range(self.n + 1)]

        for i in range(self.n + 1):
            for j in range(self.n + 1):
                slae_coeffs = 0.0
                expanded_coeff = 0.0
                for k in range(len(pl)):
                    slae_coeffs += pl[k].weight * \
                                   (pl[k].x ** i) * \
                                   (pl[k].x ** j)
                    expanded_coeff += pl[k].weight * pl[k].y * (pl[k].x ** i)

                self.mat[i][j] = slae_coeffs
                self.mat[i][self.n + 1] = expanded_coeff

        return self

    def solve(self) -> list[list[float]]:
        for i in range(self.n + 1):
            for j in range(self.n + 1):
                if i == j:
                    continue

                sub_coeff = self.mat[j][i] / self.mat[i][i]
                for k in range(self.n + 2):
                    self.mat[j][k] -= sub_coeff * self.mat[i][k]

        for i in range(self.n + 1):
            divider = self.mat[i][i]
            for j in range(self.n + 2):
                self.mat[i][j] /= divider

        return self.mat
```

*Модуль **graphics.py** — создание и вывод графика*

```
from __future__ import annotations
import matplotlib.pyplot as plt
from utils import Point


def plot_show(points: list[Point], approx: list[tuple(int, list[Point])]) -> None:
    x = [p.x for p in points]
    y = [p.y for p in points]

    plt.title("Аппроксимация методом среднеквадратичного 
приблжения")
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.grid('minor')
    plt.plot(x, y, 'r+', label='data_from_file')

    for a in approx:
        plt.plot([p.x for p in a[1]], [p.y for p in a[1]], label="p=" + str(a[0]))

    plt.legend()
    plt.show()
```

*Модуль **utils.py** — утилиты для считывания файла, работы с точками*

```
from __future__ import annotations


class Point:
    x: float
    y: float
    weight: int

    def __init__(self, _x: float, _y: float, _weight: float) -> None:
        self.x = _x
        self.y = _y
        self.weight = _weight

    def __str__(self):
        return "{:^8.3f} {:^8.3f} {:^8.3f}".format(self.x, self.y, self.weight)


def read_points(fname: str) -> list[Point]:
    points = []

    with open(fname, 'r') as f:
        for line in f.readlines():
            point = Point(*list(map(float, line.split()[:3])))
            points.append(point)

    return points

def print_matrix(matrix: list[list[float]]) -> None:
    for row in matrix:
        print(('[' + ' '.join(["{:8.3f}"] * len(row)) + ' ]').format(*row))

def print_points(points: list[Point]) -> None:
    print("{:^8} {:^8} {:^8}".format('X', 'Y', "Weight"))
    for p in points:
        print(p)
```

Модуль ***main.py*** *— точка входа*

```
from __future__ import annotations
from sys import argv

from utils import *
from rms_approx import *
from graphics import *

def main():
    f = argv[1]
    points = read_points(f)

    print("Source table:")
    print_points(points)

    if len(argv) > 2 and argv[2] == 'test':
        n = [i for i in range(0, 9, 2)]
        n[0] = 1
    else:
        print("Entry degree of polynom")
        n = [*map(int, input())]

    approxs = []
    for deg in n:
        slau = SLAU().build(points, deg)

        print("\nSLAU to solve\n")
        print_matrix(slau.mat)

        slau = slau.solve()

        print("\nSolved SLAE\n")
        print_matrix(slau)
        print()

        approxs.append((deg, Approximator().get_coeffs(slau).build(points)))

    plot_show(points, approxs)

if __name__ == '__main__':
    main()
```

4. Ответы на контрольные вопросы

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

   Для однозначного определения полинома N —1 степени достаточно M точек, это означает, что полином будет построен таким образом, что его график будет проходить через все табличные точки. При такой конфигурации в выражении $\sum_{i=1}^{N} p_i \cdot [y(x_i) - \phi(x_i)]^2 = min$ *часть выражения, находящаяся в скобках, обратится в 0, что означает, что нет зависимо сти от весов (при любых заданных весах, значение полинома будет минимальным в случае прохода через табличные точки).*

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

Программа работать будет, но некорректно, по причине того, что начиная со случая n=N определитель СЛАУ, которую необходимо решить, будет равен 0 (уравнения СЛАУ не будут линейно-независимыми), однозначно коэффициенты определить не удастся. Для обработки данной ситуации можно проводить анализ при решении СЛАУ или же на начальном этапе (ввод степени полинома).

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

$$(x^0, x^0)\cdot a=(y, x^0)$$

$$\sum(p_i)\cdot a=\sum(p_i\cdot y_i)$$

$$a=\frac{\sum(p_i * y_i)}{\sum(p_i)}, 0\le i<N$$

$p_i -$ *вес точки*, $N -$ *количество точек*

Если поделить числитель и знаменатель на N *то получится математическое ожидание* .

$$M[X]=\sum_{(i=1)}^{(\infty)} x_i\cdot p_i$$

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

Пусть есть таблица точек

X_i	Y_i	P_i
x_0	y_0	1
x_1	y_1	1

По ней составим СЛАУ и решим.

$$\begin{cases} a_0+(x_0+x_1)a_1+(x_0^2+x_1^2)a_2=y_0+y_1 \\ (x_0+x_1)a_0+(x_0^2+x_1^2)a_1+(x_0^3+x_1^3)a_2=y_0x_0+y_1x_1 \\ (x_0^2+x_1^2)a_0+(x_0^3+x_1^3)a_1+(x_0^4+x_1^4)a_2=y_0x_0^2+y_1x_1^2 \end{cases}$$

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

Определтиель равен $0 \rightarrow$ *решений нет* .

5. ***Построить СЛАУ при выборочном задании степеней аргумента полинома*** $f(x)=a_0+a_1\cdot x^m+a_2\cdot x^n$ ***, причем степени n и m в этой формуле известны.***

$$\begin{cases}(x^0,x^0)\cdot a_0+(x^0,x^m)\cdot a_1+(x^0,x^n)\cdot a_2=(y,x^0)\\(x^m,x^0)\cdot a_0+(x^m,x^m)\cdot a_1+(x^m,x^n)\cdot a_2=(y,x^m)\\(x^n,x^0)\cdot a_0+(x^n,x^m)\cdot a_1+(x^n,x^n)\cdot a_2=(y,x^n)\end{cases}$$

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.***

Для каждой пары из системы m и n(учитывая, что степень полинома меньше количества точек) вычислить a_0,a_1,a_2 функции ϕ а затем выбрать пару, для которой $\sum_{i=1}^{N} p_i\cdot[y(x_i)-\phi(x_i)]^2=min$