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
по дисциплине "Вычислительные алгоритмы"

Тема Среднеквдратичное приближение.

Студент Романов А.В.

Группа ИУ7-43Б

Оценка (баллы) ________________

Преподаватель Градов В.М.

Москва — 2020 г.

1. Тема работы

Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

2. Цель работы

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

3. Входные данные

1. Таблица функции с веми p_i с количеством узлов N.

x	y	ρ
x_i	y_i	ρ_i

2. Степень аппроксимирующего полинома – n.

4. Выходные данные

График, на котором изоброжённ аппроксимирующий полином, и точки из исходной таблицы значений.

5. Описание алгоритма

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

$$I = \sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 \tag{1}$$

$y(x)$ - исходная функция
$\varphi(x)$ - множество функций , принадлежащих линейному пространству функций
ρ_i - вес точки

Нужно найти наилучшее приближение, т.е

$$\sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 = min \tag{2}$$

Разложим функцию $\varphi(x)$ по системе линейно независимых функций $\varphi_k(x)$:

$$\varphi(x) = \sum_{k=0}^{N} a_k \varphi_k(x) \tag{3}$$

Подставляя (3) в условие (2) получим:

$$((y - \varphi), (y - \varphi)) = (y, y) - 2 \sum_{k=0}^{n} a_k (y, \varphi_k) + \sum_{k=0}^{n} \sum_{m=0}^{n} a_k a_m (\varphi_k, \varphi_m) = min \tag{4}$$

Дифференцируя по a_k получаем:

$$\sum_{i=0}^{n} (x^k, x^m) a_m = (y, x^k) \tag{5}$$

где

$$(x^k, x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}$$

$$(y, x^k) = \sum_{i=1}^{N} \rho_i y_i x_i^k$$

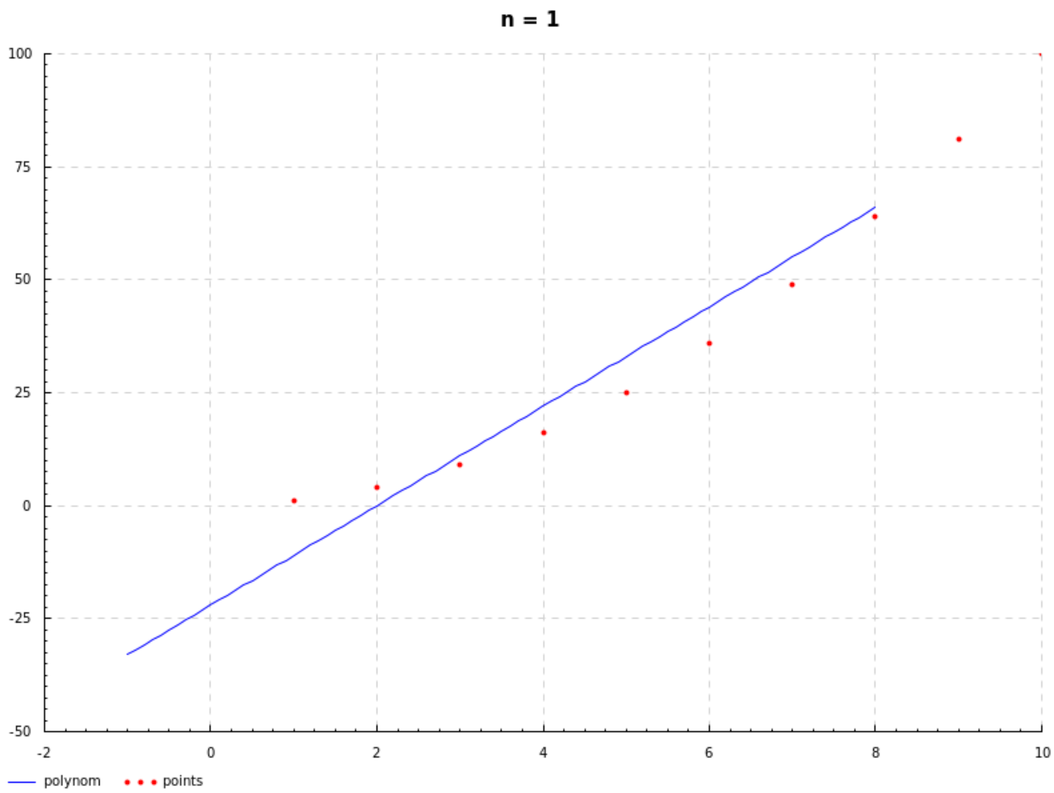
Итоговый алгоритм:
1. Выбирается степень полинома $n < N$.
2. Составляется система линейных алгебраических уравнений типа.
3. В результате решения СЛАУ находятся коэффицинты полинома.

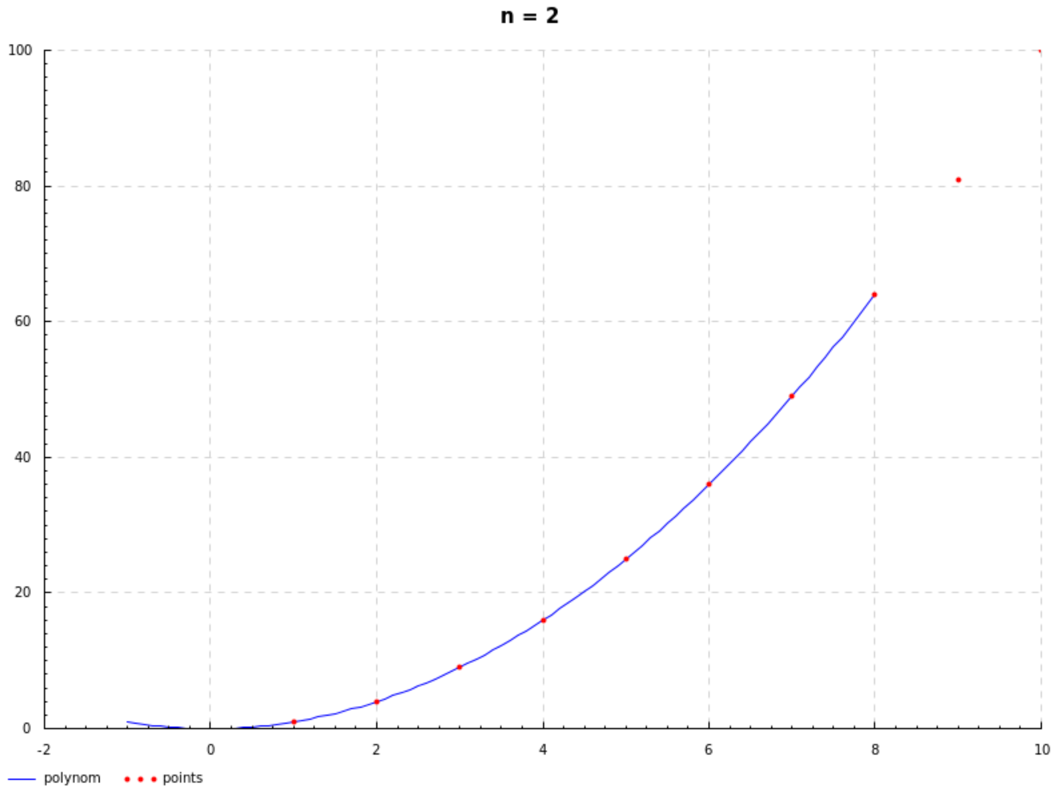
6. Результаты работы программы

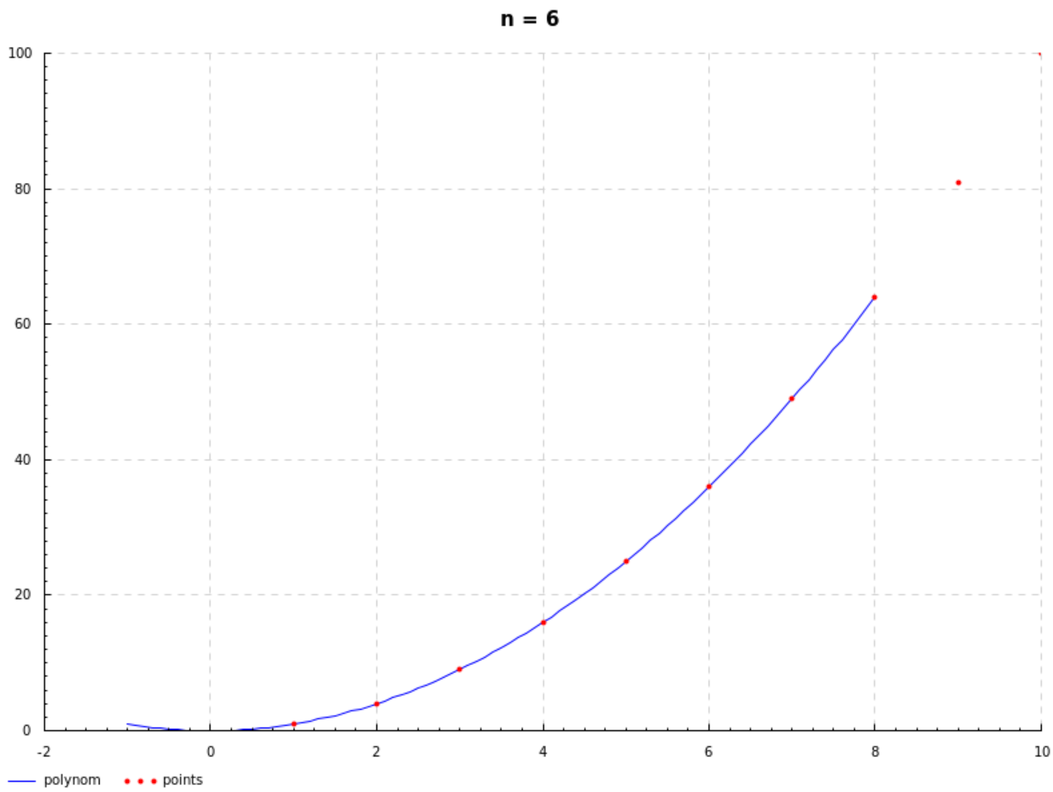
1. Веса точек одинаковы и равны.

Исходная таблица:

x_i	y_i	ρ_i
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



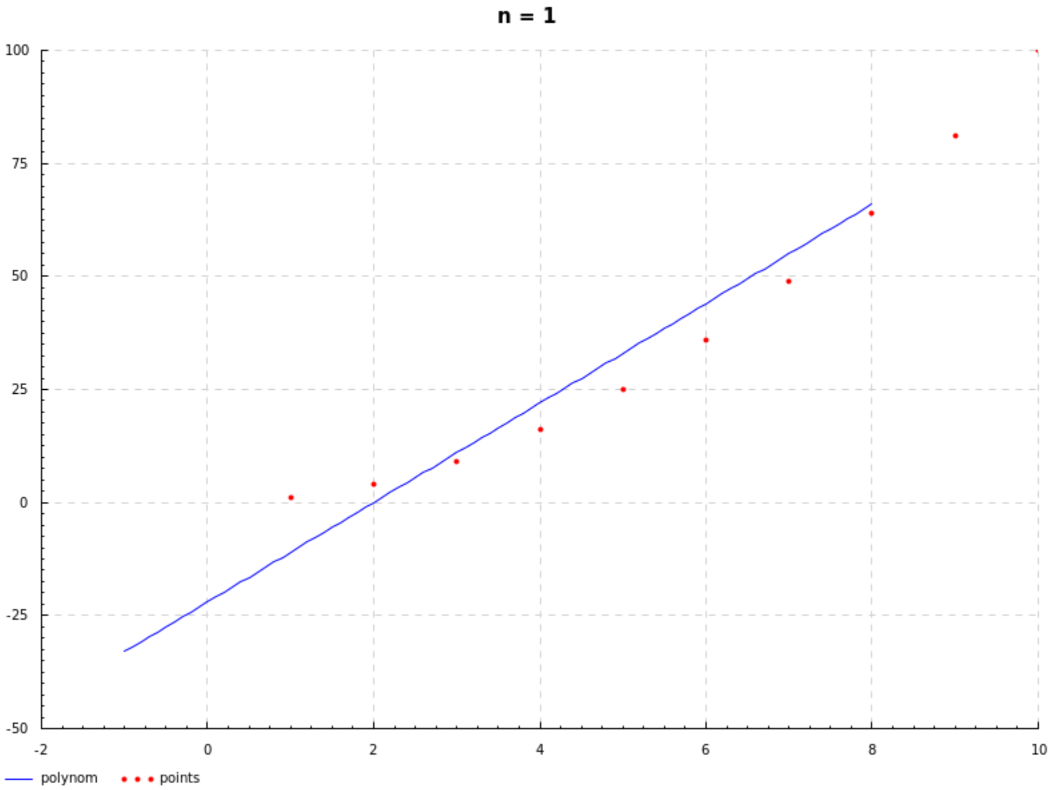

n = 2
100
80
60
40
20
0
-2
0
2
4
6
8
10
polynom
points

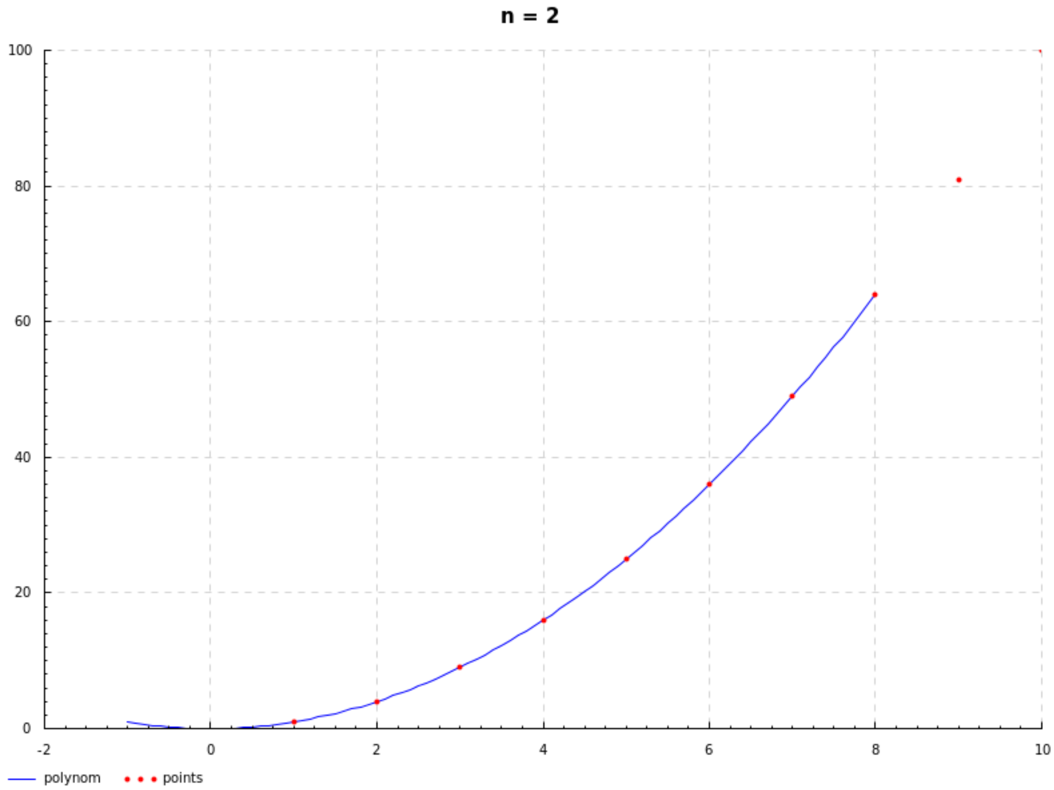

n = 6
100
80
60
40
20
0
-2
0
2
4
6
8
10
polynom
points

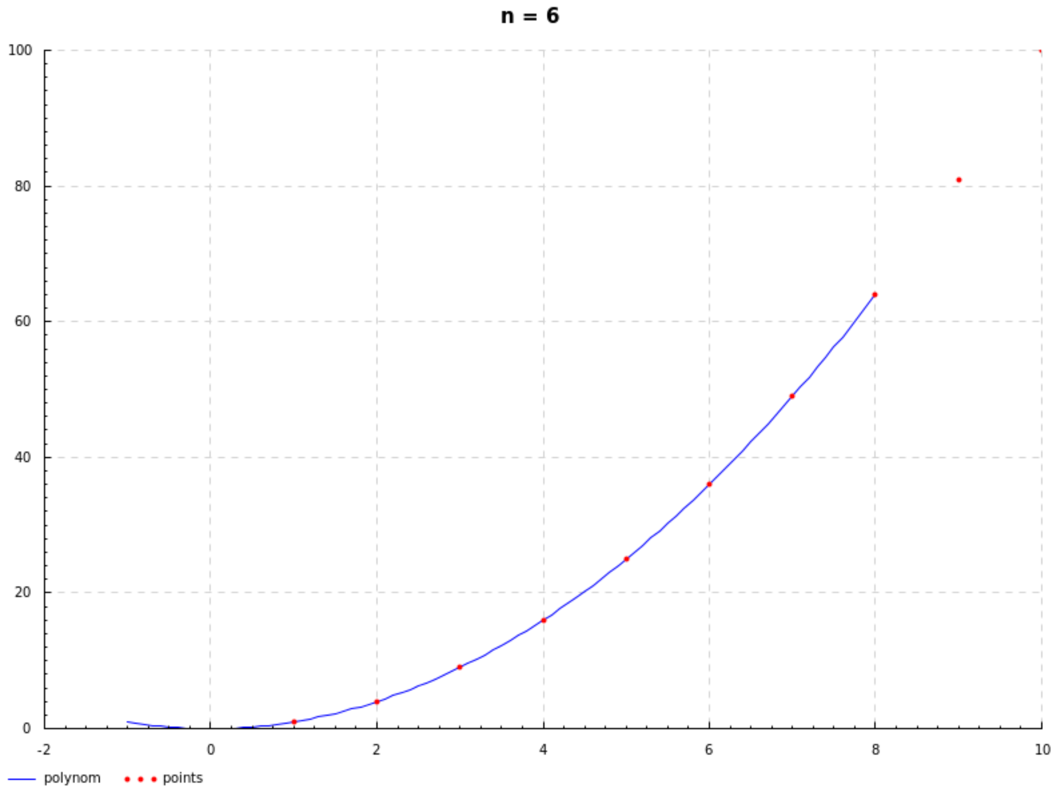
2. Веса точек разные.

Исходная таблица:

x_i	y_i	ρ_i
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 = 2
100
80
60
40
20
0
-2
0
2
4
6
8
10
polynom
points


n = 6
100
80
60
40
20
0
-2
0
2
4
6
8
10
polynom
points

7. Ответы на вопросы для защиты ЛР

1. Что произойдет при задании степени полинома $n = N - 1$

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

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

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

5. Построить СЛАУ при выборочном задании степеней аргумента полинома $\varphi(x) = a_0 + a_1 x^m + a_2 x^n$, 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 a_k, т.е. количество неизвестных равно 5.

8. Код программы

Файл Main.hs:

```
import Parse
import Gauss
import Plot
import Approximation
import System.IO

printRow :: ((Double, Double), Double) -> IO ()
printRow row = putStrLn $ show (fst $ fst row) ++ "␣" ++ show (snd $ fst row) ++
    "␣" ++ show (snd row)

main :: IO ()
main = do
    table <- openFile "table.csv" ReadMode >>= hGetContents >>= return .
        parseTable . lines
    putStrLn "X␣␣␣Y␣␣␣P" >> mapM_ printRow (zip (xy table) $ weight table) >>
        putStrLn "Enter␣n:"
    coeffs <- fmap toInt getLine >>= return . (+ 1) >>= return .
        quadraticApproximation table
    print coeffs
    plotApproximation (f coeffs) $ xy table
```

Файл Gauss.hs:

```
module Gauss(
    gauss
) where

type Matrix = [[Double]]
type Coeffs = [Double]

subtractRow :: [Double] -> [Double] -> [Double]
subtractRow subRow row = map (\x -> fst x - snd x * (head row / head subRow)) $
    zip row subRow

triangulation :: Matrix -> Matrix
triangulation matrix
    | length matrix == 0 = matrix
    | otherwise = head matrix :
        triangulation (map tail (map (subtractRow $ head matrix) $ tail matrix))

gauss :: Matrix -> Coeffs
gauss = coeffs . reverse . triangulation
    where coeffs = foldl (\x y -> (last y - (sum $ zipWith (*) (init $ tail y) x
        )) / (head y) : x) []
```

Файл Approximation.hs:

```
module Approximation(
    quadraticApproximation,
    f
) where

import Gauss
import Parse

type Coeffs = [Double]
type Weights = [Double]

f :: Coeffs -> Double -> Double
f coeffs x = sum $ zipWith (*) coeffs (map (\y -> x ^ y) [0..length coeffs - 1])
```

```haskell
mult3 :: Double -> Double -> Double -> Double
mult3 x y z = x * y * z

quadraticApproximation :: Table -> Int -> Coeffs
quadraticApproximation table n = gauss matrix
    where
        xs = map fst $ xy table
        ys = map snd $ xy table
        x_coeffs =
            map (\k -> sum $ zipWith (*) (map (^ k) xs) $ weight table) [0..n *
                2 - 2]
        y_coeffs =
            map (\k -> sum $ zipWith3 mult3 (map (^ k) xs) ys $ weight table)
                [0..n - 1]
        matrix = zipWith (\x y -> x ++ [y]) (map (\x -> take n $ drop x x_coeffs
            ) [0..n - 1]) y_coeffs
```

Файл Plot.hs:

```haskell
module Plot (
    plotApproximation
) where

import Graphics.Rendering.Chart.Easy
import Graphics.Rendering.Chart.Backend.Cairo

signal :: (Double -> Double) -> [Double] -> [(Double,Double)]
signal f xs = [ (x, f x) | x <- xs ]

plotApproximation f pts = toFile def "n6w1.png" $ do
    layout_title .= "n = 6"
    setColors [opaque blue, opaque red]
    plot (line "polynom" [signal f [-1,(-0.9)..8]])
    plot (points "points" pts)
```