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

7. Ответы на вопросы для защиты ЛР

8. Код программы

Файл Main.hs:

```
import Parse
import Spline
import System.IO

main :: IO ()
main = do
    handle <- openFile "table.csv" ReadMode
    content <- hGetContents handle
    let table = parseTable $ lines content
    mapM_ print table
    hClose handle

    putStrLn "Enter X:"
    x0 <- fmap toDouble getLine

    putStr "Result: "
    print $ spline table x0
```

Файл Spline.hs:

```
module Spline(
    spline
) where

import Data.Tuple.Select
import Data.List
import Data.Maybe

type ValueTable = [[Double]]
type RunningCoeffs = ([Double], [Double])

type Pair2 = (Double, Double)
type Pair3 = (Double, Double, Double)
type Pair4 = (Double, Double, Double, Double)
type Pair5 = (Double, Double, Double, Double, Double)

data Polynom = Polynom { h :: [Double],
                         a_k :: [Double],
                         b_k :: [Double],
                         d_k :: [Double],
                         f_k :: [Double]
                       } deriving (Show)

data Spline = Spline { a :: [Double],
                       b :: [Double],
                       c :: [Double],
                       d :: [Double]
                     } deriving (Show)

findInterval :: [Double] -> Double -> Int
findInterval xs x_value = (fromJust $ findIndex (> x_value) xs)

calcF :: (Pair2, Pair3) -> Double
```

```
calcF x = -3 * (((sel1 (snd x) - sel2 (snd x)) / (fst $ fst x)) - ((sel2 (snd x)
    - sel3 (snd x)) / (snd $ fst x)))

calcPolynom :: [Double] -> [Double] -> Polynom
calcPolynom xs ys = Polynom h a b d f
    where h = 0 : (map (\x -> fst x - snd x) $ zip (drop 1 xs) (xs))
          a = 0 : init h
          b = 0 : 0 : (map (\x -> -2 * (fst x + snd x)) $ zip (drop 1 h) (drop 2
              h))
          d = 0 : 0 : (drop 2 h)
          h2 = zip (drop 1 h) (drop 2 h)
          y3 = zip3 (drop 2 ys) (drop 1 ys) ys
          f = 0 : 0 : (map calcF $ zip h2 y3)

calcKsi :: [Double] -> Pair3 -> [Double]
calcKsi y x = y ++ [sel1 x / (sel2 x - sel3 x * last y)]

calcEta :: [Double] -> Pair4 -> [Double]
calcEta y x = y ++ [(sel1 x * last y + sel2 x) / (sel3 x - sel1 x * sel4 x)]

calcRunningCoeffs :: Polynom -> RunningCoeffs
calcRunningCoeffs polynom = (ksi, eta)
    where a = drop 2 $ a_k polynom
          b = drop 2 $ b_k polynom
          d = drop 2 $ d_k polynom
          f = drop 2 $ f_k polynom

          ksi = foldl calcKsi [0, 0, 0] $ zip3 d b a
          eta = foldl calcEta [0, 0, 0] $ zip4 a f b $ drop 2 ksi

calcB :: Pair5 -> Double
calcB x = (sel1 x - sel2 x) / sel3 x - (sel3 x * (sel4 x + 2 * sel5 x) / 3)

calcC :: [Double] -> Pair2 -> [Double]
calcC y x = (sel1 x * head y + sel2 x) : y

calcD :: Pair3 -> Double
calcD x = (sel1 x - sel2 x) / (3 * sel3 x)

reverseCoeff :: [Double] -> [Double]
reverseCoeff = tail . reverse . drop 1

calcSpline :: Polynom -> [Double] -> [Double] -> [Double] -> [Double] -> Spline
calcSpline polynom xs ys ksi eta = Spline a b c d
    where a = 0 : (init ys)
          c = foldl calcC [0, 0] $ zip (reverseCoeff ksi) (reverseCoeff eta)
          b = (map calcB $ zip5 (reverse ys) (tail $ reverse ys) (tail $ h
              polynom) c (tail c)) ++ [0]
          d = (map calcD $ zip3 c (tail c) (reverse $ tail $ h polynom)) ++ [0]

finalValue :: Spline -> Int -> Double -> [Double] -> Double
finalValue spline ind x xs = ax + bx + cx + dx
    where x_value = x - (xs !! (ind - 1))
          ax = (a spline) !! ind
          bx = ((reverse $ b spline) !! ind) * x_value
          cx = ((reverse $ c spline) !! ind) * x_value ^ 2
          dx = ((reverse $ d spline) !! ind) * x_value ^ 3

spline :: ValueTable -> Double -> Double
spline table x = finalValue spline index x xs
    where xs = map head table
          ys = map last table

          polynom = calcPolynom xs ys
```

```
(ksi, eta) = calcRunningCoeffs polynom
spline = calcSpline polynom xs ys ksi eta
index = findInterval xs x
```