Федеральное государственное автономное

образовательное учреждение высшего образования

“Национальный исследовательский университет ИТМО”

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

по вычислительной математике

Вариант №3

Выполнил:

Голиков Андрей

Группа: P32092

Принял: Рыбаков

Степан Дмитриевич

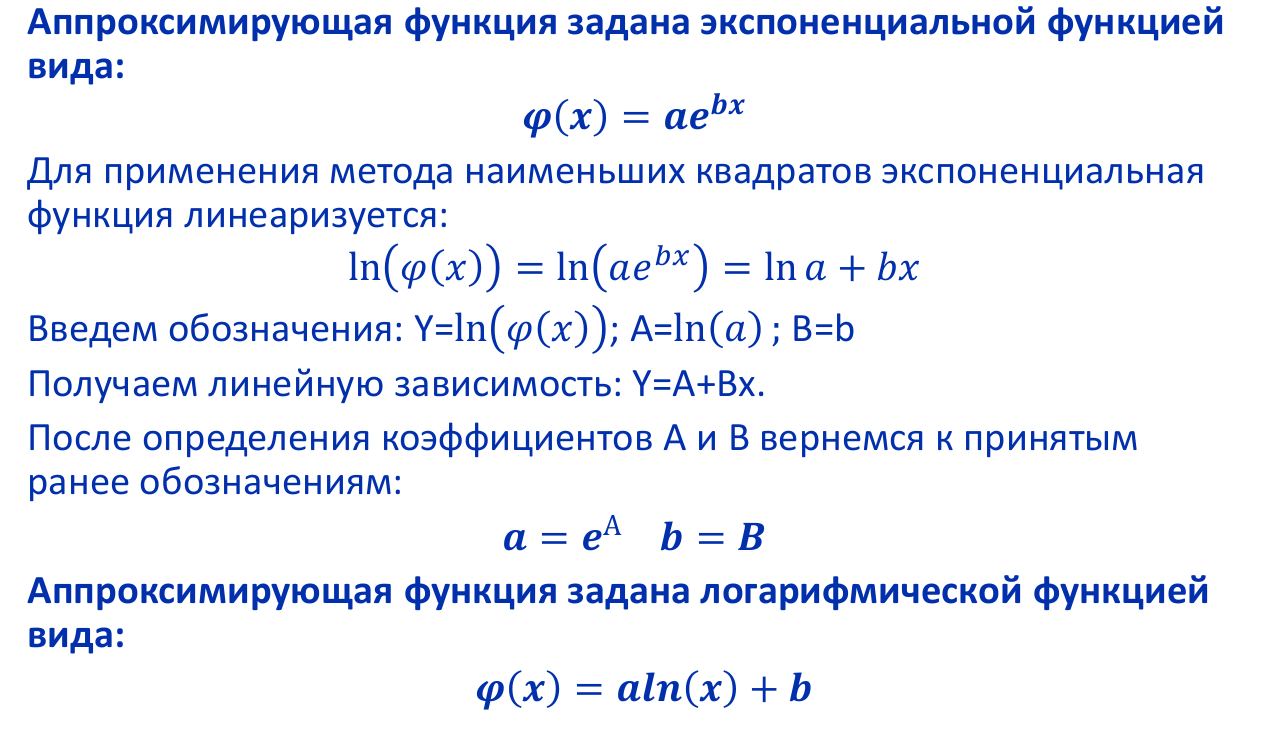
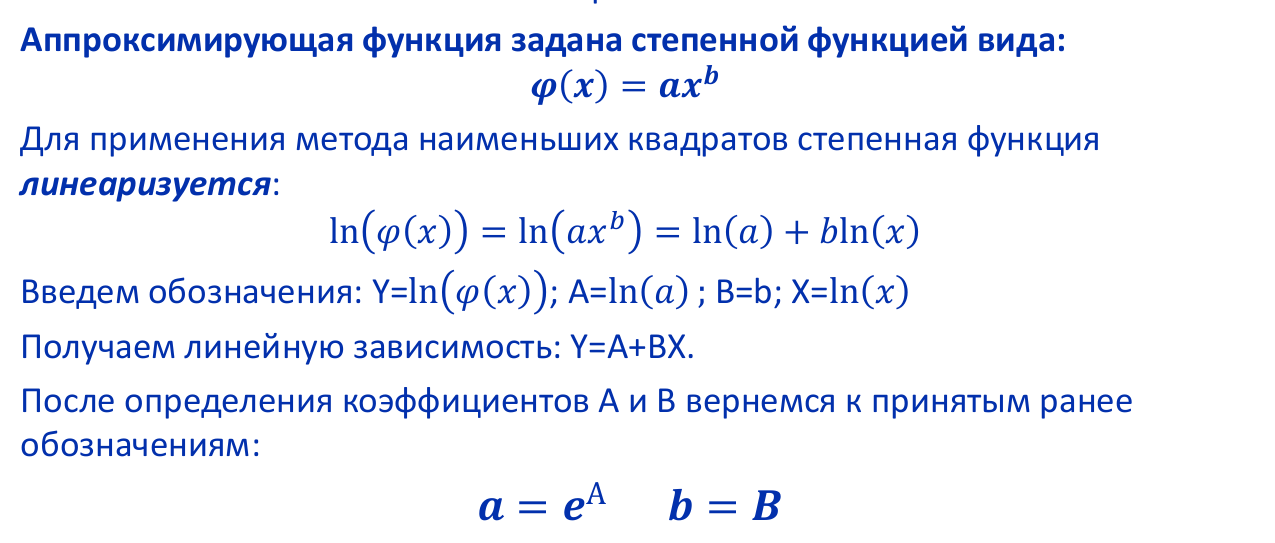
???

?

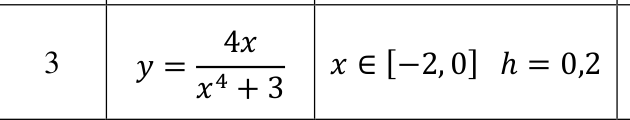


Санкт-Петербург, 2023

Цель работы: найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Рабочие формулы:

Вычислительная часть:



SX = -2 - 1.8 - … - 0 = -11

SXX = 2\*2 + 18\*1.8 + … 0\*0 = 15.4

SY = -6.8897

SXY = 7.6311

d = SXX \* 11 - SX ^ 2 = 48.4

d1 = SXX \* SY - SX \* SXY = -22.1594

d2 = SXY \* 11 - SX \* SY = 8.1554

a = -0.4578

b = 0.1685

phi(x) = 0.1685x - 0.4578

eps = 0.2788

SXXX = -24.2

SXXXX = 40.5328

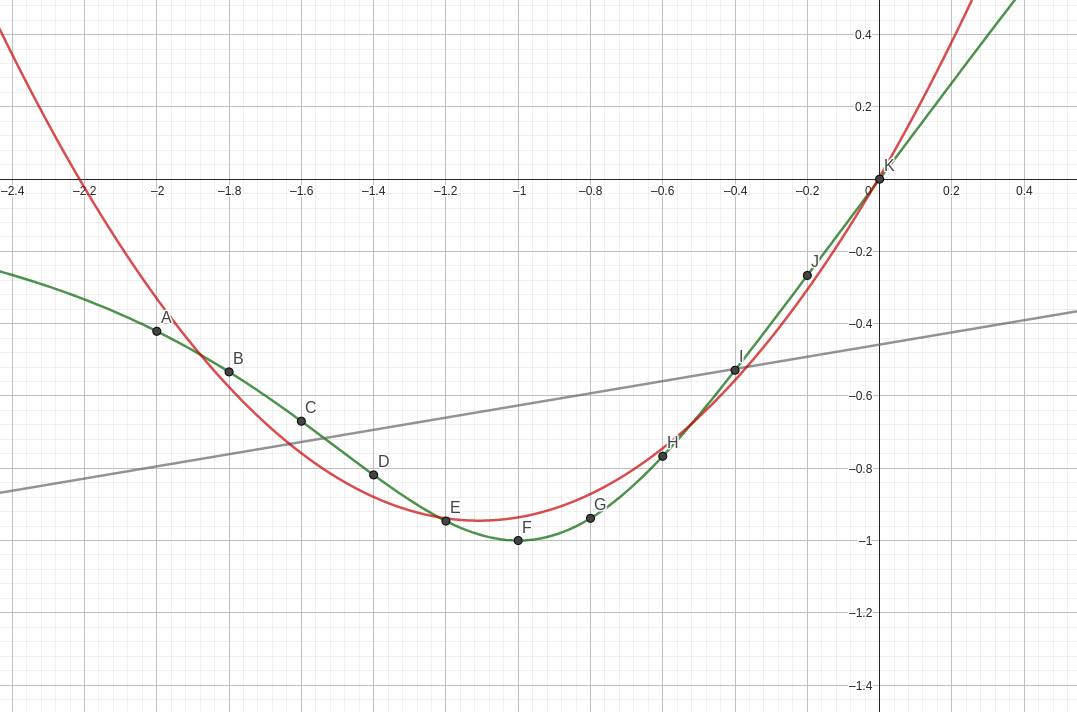
SXXY = -10.0661

a = 0.0064

b = 1.7161

c = 0.7738

phi = 0.0064 + 1.7161x + 0.7738x^2

eps = 0.0547 - лучше линейного

Листинг программы:

module Evaluator where

import Util

import Data.List (intercalate, minimumBy)

type Func = Double -> Double

type X = Double

type Y = Double

type Point = (X, Y)

type Points = [Point]

type Params = [Double]

evalS :: Points -> Func -> Double

evalS points phi = sum $ map (\(x, y) -> (phi x - y) ^ 2) points

eps :: Points -> Func -> Double

eps points phi = (\x -> sqrt (x / n)) $ evalS points phi where

n = realToFrac $ length points

getLinParams :: Points -> Params

getLinParams points = [d1/d, d2/d] where

d = xx \* n - x ^ 2

d1 = xx \* y - x \* xy

d2 = xy \* n - x \* y

n = realToFrac $ length points

x = sum $ map (\(x, \_) -> x) points

xx = sum $ map (\(x, \_) -> x^2) points

y = sum $ map (\(\_, y) -> y) points

xy = sum $ map (\(x, y) -> x \* y) points

getPol2Params :: Points -> Params

getPol2Params points = [d1/d, d2/d, d3/d] where

d = det3 (n, x1, x2,

x1, x2, x3,

x2, x3, x4)

d1 = det3 (x0y, x1, x2,

x1y, x2, x3,

x2y, x3, x4)

d2 = det3 (n, x0y, x2,

x1, x1y, x3,

x2, x2y, x4)

d3 = det3 (n, x1, x0y,

x1, x2, x1y,

x2, x3, x2y)

n = realToFrac $ length points

x1 = sum $ map (\(x, \_) -> x) points

x2 = sum $ map (\(x, \_) -> x^2) points

x3 = sum $ map (\(x, \_) -> x^3) points

x4 = sum $ map (\(x, \_) -> x^4) points

x0y = sum $ map (\(\_, y) -> y) points

x1y = sum $ map (\(x, y) -> x \* y) points

x2y = sum $ map (\(x, y) -> x \* x \* y) points

getPol3Params :: Points -> Params

getPol3Params points = [d1/d, d2/d, d3/d, d4/d] where

d = det4 (n, x1, x2, x3,

x1, x2, x3, x4,

x2, x3, x4, x5,

x3, x4, x5, x6)

d1 = det4 (x0y, x1, x2, x3,

x1y, x2, x3, x4,

x2y, x3, x4, x5,

x3y, x4, x5, x6)

d2 = det4 (n, x0y, x2, x3,

x1, x1y, x3, x4,

x2, x2y, x4, x5,

x3, x3y, x5, x6)

d3 = det4 (n, x1, x0y, x3,

x1, x2, x1y, x4,

x2, x3, x2y, x5,

x3, x4, x3y, x6)

d4 = det4 (n, x1, x2, x0y,

x1, x2, x3, x1y,

x2, x3, x4, x2y,

x3, x4, x5, x3y)

n = realToFrac $ length points

x1 = sum $ map (\(x, \_) -> x) points

x2 = sum $ map (\(x, \_) -> x^2) points

x3 = sum $ map (\(x, \_) -> x^3) points

x4 = sum $ map (\(x, \_) -> x^4) points

x5 = sum $ map (\(x, \_) -> x^5) points

x6 = sum $ map (\(x, \_) -> x^6) points

x0y = sum $ map (\(\_, y) -> y) points

x1y = sum $ map (\(x, y) -> x \* y) points

x2y = sum $ map (\(x, y) -> x \* x \* y) points

x3y = sum $ map (\(x, y) -> x \* x \* x \* y) points

getExpParams :: Points -> Params

getExpParams points = [exp a, b] where

(a:b:\_) = getLinParams $ map (\(x,y) -> (x, log y)) points

getLnParams :: Points -> Params

getLnParams points = getLinParams $ map (\(x,y) -> (log x, y)) points

getPowParams :: Points -> Params

getPowParams points = [exp a, b] where

(a:b:\_) = getLinParams $ map (\(x,y) -> (log x, log y)) points

chooseFunc :: Points -> Res

chooseFunc points = minimumBy comp ress where

ress = map getRes [getLinFunc, getPol2Func, getPol3Func, getExpFunc, getLnFunc, getPowFunc]

getRes :: (Points -> (Func, String)) -> Res

getRes f = (func, str, eps points func, pirson points) where

(func, str) = f points

comp = \(\_,\_,a,\_) (\_,\_,b,\_) -> case (isNaN a, isNaN b) of

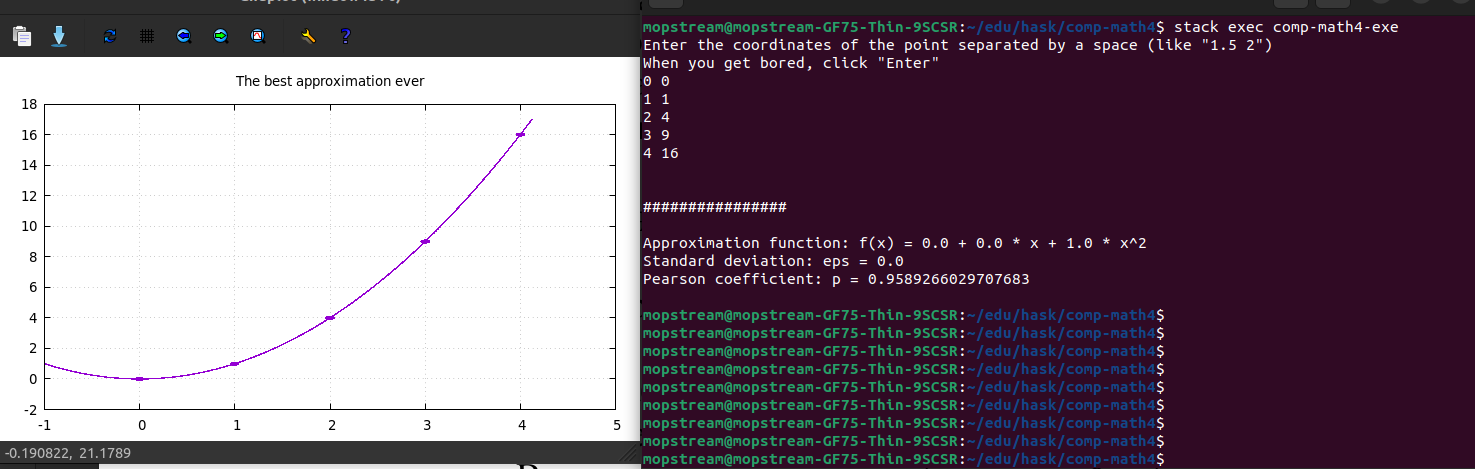
(True, True) -> EQ

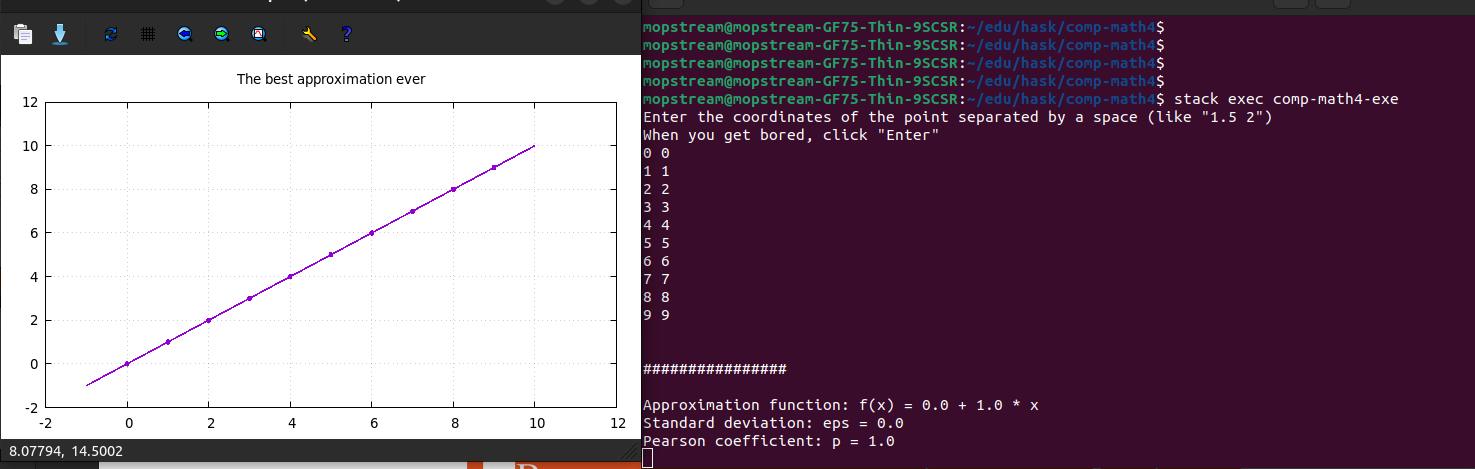
(True, \_) -> GT

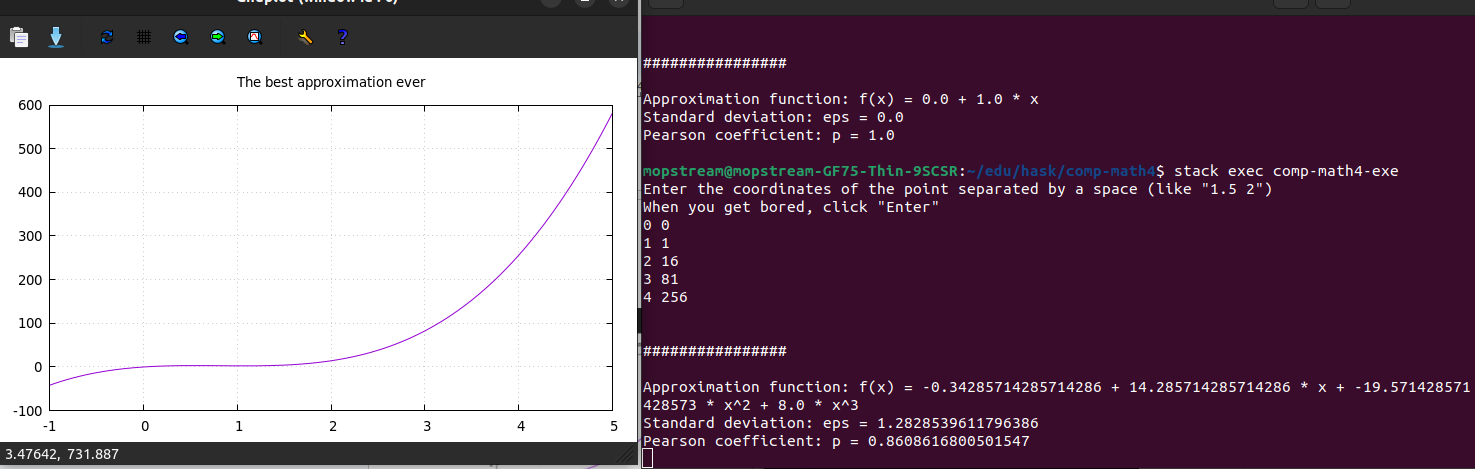
(\_, True) -> LT

(\_,\_) -> compare a b

Результаты работы программы:







Вывод: реализовал на языке Haskell поиск лучшей аппроксимирующей функции. Выбирал функции из полиномов 1, 2, 3 степеней, а так же логарифмической, экспоненциальной и показательной функций. Для оценки приближения использовал метод наименьших квадратов