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 № 3
по дисциплине "Вычислительные алгоритмы"

Тема Интерполяция сплайнами.

Студент Романов А.В.

Группа ИУ7-43Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва — 2020 г.

1. Задание

Задана таблица значений функции вида x, $f(x)$. Провести интерполяцию сплайном на данной таблице, и найти значение $f(x)$.

Входные данные: Таблица значений функции, значение точки по координате X.
Выходные данные: значение функции $f(x)$.

2. Описание алгоритма

Кубический сплайн — это кривая, состоящая из „состыкованных“ полиномов третьей степени ($y^{IV}\ (x) = 0$). В точках стыковки значения и производные двух соседних полиномов равны.

Имеем:

$$\varphi(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$
$$\varphi(x_{i-1}) = a_i$$
$$\varphi(x_i) = a_i + b_i h_i + c_i {h_i}^2 + d_i {h_i}^3$$
$$\varphi'(x_i) = {}_i + 2_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2$$
$$\varphi''(x_i) = 2_i + 6d_i(x_i - x_{i-1})$$
$$c_i + 3d_i h_i = c_{i+1}$$

Получим СЛАУ с трехдиаганальной матрицей:

$$\begin{cases} c_1 = 0 \\ h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_i}) \\ c_{N+1} = 0 \end{cases}$$

Решается СЛАУ методом прогонки:
1. Находятся все прогочные коэффициенты по формулам:

$$\xi_{i+1} = \frac{D_i}{B_i - A_i\xi_i}$$

$$\eta_{i+1} = \frac{F_i + A_i\eta_i}{B_i - A_i\xi_i}$$

2. При известном y_N определяются все y_i – обратный ход.
Приминительно к задаче поиска коэффициентов сплайна имеем $c_{i]} \Leftrightarrow y_i$

Обратный ход:

$$c_i = \xi_{i+1}c_{i+1} + \eta_{i+1}, \text{ при } c_{N+1} = 0 \text{ и } c_N = \xi_{i+1}$$

При найденых c:

$$a_i = y_{i-1}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} d_N = -\frac{C_N}{3h_N}$$

$$b_i = \frac{y_i - y_{i-1}}{h_i} - \frac{1}{3}h_i(c_{i+1} + 2c_i)$$

3. 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
```