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

Тема Сплайн-интерполяция

Студент Кононенко Сергей

Группа ИУ7-43Б

Оценка (баллы) _____

Преподаватель Градов Владимир Михайлович

Москва — 2020 г.

1. Задание

Задана таблица значений функции вида x, $f(x)$. Построить алгоритм для проведения кубической сплайн-интерполяции.
Ввод: значение x_0.
Вывод: значение функции от x_0, найденное при помощи кубической сплайн-интерполяции.

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.

Интерполяционный полином на участке между каждой парой соседних точек имеет вид:

$$\phi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$$

В узлах значения многочлена и интерполируемой функции совпадают:

$$f(x_{i-1}) = y_{i-1}$$

$$f(x_i) = y_i = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$$

Формулы для определения коэффициентов:

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

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, h_i = x_i - x_{i-1}$$

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

Система уравнений для определения коэффициента c_i:

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

Матрица этой системы трёхдиагональна. Такая система решается *методом прогонки.*

Алгоритм метода прогонки:
Прямой ход: при заданных начальных значениях прогоночных коэффициентов ξ_i и η_i определяются все прогоночные коэффициенты:

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

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

Обратный ход: при известном c_N определяются все $c_i, i = \overline{1, N}$.

$$c_1 = 0$$

$$c_1 = \xi_2 c_2 + \eta_2$$

$$\begin{cases} \xi_2 = 0 \\ \eta_2 = 0 \end{cases}$$

Имея граничные условия, находим начальные коэффициенты (*прямой ход*).
Нахождение c_i (*обратный ход*):

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

3. Код программы

Основной пакет приложения.

```
package main

import (
        "fmt"
        "os"
        "sort"

        "./spline"
)

func main() {
        if len(os.Args) <= 1 {
                fmt.Printf("USAGE: lab_03 <datafile>\n")
                os.Exit(1)
        }

        f, err := os.Open(os.Args[1])
        if err != nil {
                fmt.Println("Error:", err)
                os.Exit(1)
        }

        ds := spline.ReadDots(f)
        fmt.Printf("Table loaded from file:\n\n")
        ds.PrintDots()

        fmt.Printf("\nEnter X value: ")
        d := spline.ReadX()

        sort.Sort(ds)
        spline.Spline(ds, &d)

        fmt.Printf("\nFunction value in %5.2f dot is %5.4f\n\n", d.X, d.Y)
}
```

Пакет реализации сплайн-интерполяции.

1. Обработка точек на плоскости.

```
package spline

import (
        "fmt"
        "io"
)
```

```
// Dot type used to represent plane dots.
type Dot struct {
        X float64
        Y float64
}

// DotSet type used to represent amount of plane dots.
type DotSet []Dot

// ReadX used to read function X coordinate.
func ReadX() Dot {
        var d Dot

        fmt.Scan(&d.X)

        return d
}

// ReadDots used to read Dot objects to DotSet object from file.
func ReadDots(f io.Reader) DotSet {
        var (
                ds     DotSet
                curDot Dot
        )

        for {
                _, err := fmt.Fscanln(f, &curDot.X, &curDot.Y)
                if err == io.EOF {
                        break
                }
                ds = append(ds, curDot)
        }

        return ds
}

// PrintDots used to print dots in table form to standart output.
func (ds DotSet) PrintDots() {
        for i := range ds {
                fmt.Printf("%8.2f %8.2f\n", ds[i].X, ds[i].Y)
        }
}

func (ds DotSet) Len() int {
        return len(ds)
}
```

```
func (ds DotSet) Less(i, j int) bool {
        return ds[i].X < ds[j].X
}

func (ds DotSet) Swap(i, j int) {
        ds[i], ds[j] = ds[j], ds[i]
}

func (ds DotSet) getPos(d Dot) int {
        var pos int

        if d.X < ds[0].X {
                pos = 0
        } else if d.X > ds[len(ds)-1].X {
                pos = len(ds) - 1
        } else {
                for i := 1; i < len(ds)-2; i++ {
                        if d.X > ds[i-1].X && d.X <= ds[i].X {
                                pos = i - 1
                        }
                }
        }

        return pos
}
```

2. Реализация сплайн-интерполяции.

```
package spline

import (
        "math"
)

// Spline used to find approximated value of function given in table representation.
func Spline(ds DotSet, xd *Dot) {
        aCoef := []float64{0}
        bCoef := []float64{0}
        cCoef := []float64{0}
        dCoef := []float64{0}
        hCoef := []float64{0}
        xiCoef := []float64{0, 0, 0}
        etaCoef := []float64{0, 0, 0}

        for _, d := range ds {
                aCoef = append(aCoef, d.Y)
        }
```

```
        for i := 1; i < len(ds); i++ {
                hCoef = append(hCoef, ds[i].X-ds[i-1].X)
                cCoef = append(cCoef, 0)
        }
        cCoef = append(cCoef, 0)

        for i := 3; i < len(ds)+1; i++ {
                xii := hCoef[i-1] /
                        (-2*(hCoef[i-1]+hCoef[i-2]) - hCoef[i-2]*xiCoef[i-1])
                xiCoef = append(xiCoef, xii)

                fStrong := -3 * ((ds[i-1].Y-ds[i-2].Y)/hCoef[i-1] -
                        (ds[i-2].Y-ds[i-3].Y)/hCoef[i-2])
                etai := (fStrong + hCoef[i-2]*etaCoef[i-1]) /
                        (-2*(hCoef[i-1]+hCoef[i-2]) - hCoef[i-2]*xiCoef[i-1])
                etaCoef = append(etaCoef, etai)
        }

        for i := len(ds) - 1; i > 1; i-- {
                cCoef[i] = xiCoef[i+1]*cCoef[i+1] + etaCoef[i+1]
        }

        for i := 1; i < len(ds); i++ {
                bi := (aCoef[i+1]-aCoef[i])/hCoef[i] -
                        hCoef[i]/3*(cCoef[i+1]+2*cCoef[i])
                bCoef = append(bCoef, bi)

                di := (cCoef[i+1] - cCoef[i]) / 3 / hCoef[i]
                dCoef = append(dCoef, di)
        }
        bCoef = append(bCoef, 0)
        dCoef = append(dCoef, 0)

        pos := ds.getPos(*xd)

        res := aCoef[pos+1] +
                bCoef[pos+1]*(xd.X-ds[pos].X) +
                cCoef[pos+1]*math.Pow((xd.X-ds[pos].X), 2) +
                dCoef[pos+1]*math.Pow((xd.X-ds[pos].X), 3)

        xd.Y = res
}
```