ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Кафедра «ВТ»

ЛАБОРАТОРНАЯ РАБОТА № _3_

ПО ДИСЦИПЛИНЕ «Вычислительная математика»

Выполнил(а): Чан Чунг Дык

Группа: Р3202

Вариант : 18

Санкт-Петербург

2017/2018

1. Цель лабораторной работы: Решить задачу интерполяции: найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Для исследования использовать:

* линейную и квадратичную интерполяцию;

* многочлен Лагранжа;

* многочлен Ньютона

Таблица 2

X	y
0.593	0.5320
0.598	0.5356
0.605	0.5406
0.613	0.5462
0.619	0.5504
0.627	0.5559
0.632	0.5594

№ вариант	X1	X4
18	0.610	0.628

Таблица 6

X	y
0.50	1.5320
0.55	2.5356
0.60	3.5406
0.65	4.5462
0.70	5.5504
0.75	6.5559
0.80	7.5594

№ вариант	X2	X3
18	0.545	0.765

Результат

X	Линейная	квадратичная	Лагранжа	1-Ньютона	2-Ньютона	Ньютон для неравноотстоящих узлов
0,610	0.544	0.544	0.544	----	----	0.557
0.545	---	---	---	2.435	---	---
0.765	---	---	---	---	6.858	---

```
/*
   Name : Tran Trung Duc    Group : P3202    ITMO    Variant : 18
 */
import java.io.IOException;
import java.lang.*;

public class Lab3 {
   public double arr[][] = new double[2][10] ;
   private int n;
   public void input() throws IOException{
      String fileName = "D:\\Russia\\thirdsemester\\math\\lab3\\input1.txt";
      readFile inp = new readFile(fileName);
      String tmp;
      int i =0;
      while ((tmp = inp.readLine())!= null ){
         String[] _tmp = tmp.split("\\s+");
         this.n = _tmp.length;
         for(int j=0;j<n;j++) arr[i][j] = Double.parseDouble(_tmp[j]);
         i++;
      }
   }
   public void process(){
      double x1 =0.610;
      //2.1
      linear tmp = new linear(arr,x1,n);
      System.out.printf("%.3f\n",tmp.solve());

      Quadratic _tmp = new Quadratic(arr,x1,n);
      System.out.printf("%.3f\n",_tmp.solve());

      //2.2
      Lagrange tmp_ = new Lagrange(arr,x1,n);
      System.out.printf("%.3f\n",tmp_.solve());

      //2.3
      Newton duc = new Newton(arr,n);
      System.out.printf("%.3\nf",duc.solve_equal(0.545));
      System.out.printf("%.3\nf",duc.solve_equal(0.765));

      //2.4
      Newton duc_ = new Newton(arr,n);
      System.out.printf("%.3\nf",duc_.solve_not_equal(0.628));
   }
   public static void main(String[] args) throws IOException{
      Lab3 duc = new Lab3();
      duc.input();
      duc.process();
   }
}
```

```
class Lagrange {
   private double arr[][];
   private double x1;
   private int n;
   public Lagrange(double arr[][], double x1,int n){
      this.arr = arr;
      this.x1 = x1;
      this.n = n;
   }

   public double solve(){
      double res = 0;
      for(int i=0;i<n;i++){
         double res1 = 1;
         for(int j=0;j<n;j++) if (i!=j) res1 *= (x1-arr[0][j]) / (arr[0][i] - arr[0][j]);
         res+= res1 * arr[1][i];
      }
      return res;
   }
}
```

```
public class linear {
   private double arr[][];
   private double x;
   private int n;
   public linear(double arr[][], double x1,int n){
      this.arr = arr;
      this.x = x1;
      this.n = n;
   }
   public double solve(){
      int i=0;
      while ((arr[0][i]< this.x) && (i<n)) i++;

      double a = (arr[1][i] - arr[1][i-1])/(arr[0][i]-arr[0][i-1]);
      double b = (arr[1][i-1] - a* arr[0][i-1]);
      double res = a * this.x + b;
      return res;
   }
}
```

```
public class Quadratic {
   private double arr[][];
   private double x;
   private int n;
   public Quadratic(double arr[][], double x1,int n){
      this.arr = arr;
      this.x = x1;
      this.n = n;
   }

   private double findDet(double x1,double x2,double x3, double y1,double y2,double y3,double z1, double
z2,double z3){
```

```
        double res = x1*y2*z3 + x2*y3*z1 + x3*y1*z2 - x1*z2*y3 - y1*x2*z3 - z1*y2*x3;
        return res;
    }
    private double process(int x,int y, int z){
        double res = 0;
        double a,b,c;
        double det = findDet(Math.pow(arr[0][x],2), arr[0][x], 1, Math.pow(arr[0][y],2), arr[0][y], 1,
Math.pow(arr[0][z],2), arr[0][z], 1 );
        double detA = findDet(arr[1][x], arr[0][x], 1, arr[1][y], arr[0][y], 1, arr[1][z], arr[0][z], 1 );
        double detB = findDet(Math.pow(arr[0][x],2), arr[1][x], 1, Math.pow(arr[0][y],2), arr[1][y], 1,
Math.pow(arr[0][z],2), arr[1][z], 1 );
        double detC = findDet(Math.pow(arr[0][x],2), arr[0][x], arr[1][x], Math.pow(arr[0][y],2), arr[0][y], arr[1][y],
Math.pow(arr[0][z],2), arr[0][z], arr[1][z]);
        a = detA/det;
        b = detB/det;
        c = detC/det;
        res = a * Math.pow(this.x,2) + b * this.x + c;
        return res;
    }
    public double solve(){
        int i = 0;
        while ((arr[0][i]< this.x) && (i<this.n)) i++;
        if (i>1){ // neu ben trai co 2 phan tu >> thi chon 1 phai 2 trai
            return process(i-2,i-1,i);
        }
        else { // chon 2 phai 1 trai
            return process(i,i+1,i+2);
        }
    }
}
```

```
public class Newton {
    private double arr[][];
    private int n ;
    public Newton(double arr[][],int n){
        this.arr = arr;
        this.n = n;
    }
    //find n!
    private double fact(int n){
        double res = 1;
        for(int i=2;i<=n;i++) res*=i;
        return res;
    }
    // find delta (^delta) i
    private double findDelta(int i, int delta){
        if (delta == 0) return 1;
        if (delta == 1){
            double res = arr[1][i+1] - arr[1][i];
            return res;
        }
        else return findDelta(i+1,delta-1) - findDelta(i,delta-1);
    }
    /**
```

```
 * this function use for equidistant nodes
 * 3 - 4 - 5 - 6 (Xi - X(i-1) = const)
 * @return double
 * dang sai khi chon x0
 */
public double solve_equal(double x){
   if (x < (arr[0][n-1]+arr[0][0])/2) {
      double tmp_ =1;
      //find the max be hon x
      int k = 0;
      while ((arr[0][k]<x) && (k<n)) k++;
      k--;

      double tmp = (x - arr[0][k])/(arr[0][k+1]-arr[0][k]);
      double res = arr[1][k];

      for(int i=1;i<n-k;i++){
         tmp_ = 1;
         for(int j=0;j<i;j++) tmp_ *= (tmp-j); // t(t-1)(t-2)...(t-n+1)
         double tg = findDelta(k,i);
         res += (tmp_/fact(i)) * tg;
      }
      return res;
   }else{
      double tmp_;

      double tmp = (x - arr[0][n-1])/(arr[0][1]-arr[0][0]);
      double res = arr[1][n-1];

      for(int i=1;i<n;i++){
         tmp_ = 1;
         for(int j=0;j<i;j++) tmp_ *= (tmp+j);
         res += (tmp_/fact(i))*findDelta(n-1-i,i);
      }
      return  res;
   }
}
private double findFk(int k1, int k2){
   double tmp = arr[1][k2] -arr[1][k1];
   tmp /= (arr[0][k2]-arr[0][k1]);
   return tmp;
}
private double findFk(int k1,int k2,int k3){
   double tmp = findFk(k2,k3) - findFk(k1,k2);
   tmp /= (arr[0][k3]-arr[0][k1]);
   return tmp;
}
/**
 * this function use for un-euidistant nodes (Xi - X(i-1) != const)
 * @param x
 * @return double
 */
public double solve_not_equal(double x){
   int k =0;
   while ((arr[0][k] < x) && (k<n)) k++;
```

```
        if (k>=2) k-=2;
        else k = 0;

        double res = 0;

        res += arr[1][k] + findFk(k,k+1)* (x-arr[0][k]) + findFk(k,k+1,k+2) * (x-arr[0][k])* (x-arr[0][k+1]);

        k++;
        res+= arr[1][k] + findFk(k,k+1)* (x-arr[0][k]) + findFk(k,k+1,k+2) * (x-arr[0][k])* (x-arr[0][k+1]);
        res /=2;

        return res;
    }
}
```