*Національний технічний університет України*

*‘Київський політехнічний інститут’*

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

***Лабораторна робота №3***

*“ Інтерполяція функцій ”*

*Виконав: Мроць Ю.Б.*

*Група: ІО-12*

*Номер ЗК: 1219*

*Київ — 2013*

***Мета***

Ознайомлення з інтерполяційними формулами Лагранжа, Ньютона, рекурентним співвідношенням Ейткена, методом оцінки похибки інтерполяції.

***Завдання***

Закріплення, поглиблення і розширення знань студентів при вирішенні практичних обчислювальних завдань. Оволодіння обчислювальними методами і практичними методами оцінки похибки обчислень. Придбання умінь і навичок при програмуванні та налагодженні обчислювальних завдань на комп'ютері.

***Варіант***

|  |  |  |
| --- | --- | --- |
| № Варіанту |  |  |
| 4 (19) |  |  |

***Лістинг***

**package** l3;

/\*\*

\* Логіка побудови графіків заданої функції і полінома Ньютона степені не вище t-1

\* **@author** Aruy

\*/

**public** **class** Logic {

**private** **final** **int** t = 11; // к-ть точок

**public** **double** a; // ліва межа відрізка, на якому будуємо функцію

**public** **double** b; // права межа відрізка, на якому будуємо функцію

**public** **double**[] xi; // абсциси вузлів

**private** **double** h; // крок між абсцисами точок

**public** **int** k; // степінь інтерполяції многочлена(1..10)

**public** **double**[][] yk; // починаючи з 1 рядка, масив кінцевих різниць; 0 рядок - ординати вузлів

/\*\*

\* **@param** a абсциса лівої межі функції

\* **@param** b абсциса правої межі функції

\* **@param** k степінь інтерполяції многочлена(1..10)

\*/

**public** Logic(**double** a, **double** b, **int** k) {

**this**.a = a;

**this**.b = b;

**this**.h = (b - a) / (t - 1);

**this**.k = k;

**this**.xi = **new** **double**[t];

**this**.yk = **new** **double**[t][t];

solve\_xi\_yi(a);

solve\_yk();

}

/\*\*

\* Розрахунок ординат і абсцис t рівновіддалених вузлів

\* **@param** a абсциса лівої межі функції

\*/

**public** **void** solve\_xi\_yi(**double** a) {

**for** (**int** i = 0; i < t; i++) {

xi[i] = a + h \* i;

yk[0][i] = func(xi[i]);

}

}

/\*\*

\* Розрахунок кінцевих різниць

\*/

**public** **void** solve\_yk() {

**for** (**int** i = 1; i < t ; i++) {

**for** (**int** j = 0; j < t - i; j++) {

yk[i][j] = (yk[i-1][j+1] - yk[i-1][j]);

}

}

}

/\*\*

\* Розрахунок ординати для заданої

\* абсциси через поліном Ньютона k-го степеня

\* **@param** x абсциса

\* **@return** ордината

\*/

**public** **double** polinom(**double** x) {

**double** p;

**double** q = (x - search\_x(x)) / h;

**double** N = func(search\_x(x));

**for** (**int** i = 0; i < k; i++) { // 1 .. k

p = 1;

**for** (**int** j = 0; j < i + 1; j++) {

p \*= (q - j);

}

N += p \* yk[i + 1][search\_i(x)] / *factorial*(i + 1);

}

**return** N;

}

/\*\*

\* Розрахунок ординати для заданої

\* абсциси через основну функцію

\* **@param** x абсциса

\* **@return** ордината

\*/

**public** **double** func(**double** x) {

**return** 1 / (0.5 + Math.*pow*(x, 2));

}

/\*\*

\* Розрахунок похибки полінома в заданій точці

\* **@param** x абсциса точки

\* **@return** похибка

\*/

**public** **double** mistake(**double** x) {

**return** Math.*abs*(func(x) - polinom(x));

}

/\*\*

\* Факторіал

\* **@param** n значення

\* **@return** результат

\*/

**public** **static** **int** factorial(**int** n) {

**if** (n == 0)

**return** 1;

**return** n \* *factorial*(n-1);

}

/\*\*

\* Пошук значення абсциси найближчого вузла зліва для абсциси точки x

\* **@param** x

\* **@return** значення абсциси знайденого вузла

\*/

**public** **double** search\_x(**double** x) {

**double** k = 0;

**for** (**int** i = xi.length - 2; i > -1; i--) {

**if** (x >= xi[i]) {

k = xi[i];

**break**;

}

}

**return** k;

}

/\*\*

\* Пошук індекса абсциси найближчого вузла зліва для абсциси точки x

\* **@param** x

\* **@return** значення абсциси знайденого вузла

\*/

**public** **int** search\_i(**double** x) {

**int** k = 0;

**for** (**int** i = xi.length - 2; i > -1; i--) {

**if** (x >= xi[i]) {

k = i;

**break**;

}

}

**return** k;

}

}

**package** l3;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**double** a = 0;

**double** b = 2;

**int** k = 10;

Logic l = **new** Logic(a, b, k);

Plot p = **new** Plot(l);

}

}

package l3;

import java.awt.BorderLayout;

import javax.swing.JFrame;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;

/\*\*

\* Візуалізація полінома і функції

\* @author Aruy

\*/

public class Plot {

public Plot(Logic l) {

XYSeries series1 = new XYSeries("y = 1 / (0.5 + x^2)");

XYSeries series2 = new XYSeries("поліном");

XYSeries series3 = new XYSeries("похибка");

for (double j = l.a; j <= l.b; j += 0.01){

series1.add(j, l.func(j));

series2.add(j, l.polinom(j));

series3.add(j, l.mistake(j));

}

XYSeriesCollection data = new XYSeriesCollection(series1);

data.addSeries(series2);

XYSeriesCollection data2 = new XYSeriesCollection(series3);

JFreeChart chart1 = ChartFactory.createXYLineChart("", "x", "y", data, PlotOrientation.VERTICAL, true, true, true);

JFreeChart chart2 = ChartFactory.createXYLineChart("", "x", "y", data2, PlotOrientation.VERTICAL, true, true, true);

ChartPanel chartPanel1 = new ChartPanel(chart1);

ChartPanel chartPanel2 = new ChartPanel(chart2);

JFrame frame = new JFrame("");

frame.setLayout(new BorderLayout());

frame.add(chartPanel1, BorderLayout.WEST);

frame.add(chartPanel2, BorderLayout.EAST);

frame.pack();

frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}