Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

З алгоритмів та методів обчислень

*Виконав:*

Студент групи ІО-12

Нестерук Ю.О.

*Перевірив:*

Подрубайло О.О.

м. Київ

2013 р.

**1. Тема завдання:**

Закріплення знань студентів при вирішенні практичних завдань з розв’язування нелінійних рівнянь. Оволодіння методами і практичними навичками розв’язування нелінійних рівнянь на ЕОМ. Набуття умінь і навичок при програмуванні та налагодженні програм для розв’язування нелінійних рівнянь на комп'ютері.

**2. Завдання:**

Скласти програму розв’язання нелінійного рівняння, користуючись схемою алгоритму.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Номер варіанту | Рівняння | Примітка |
| Комбінований метод | 20 |  | -0.465 |

**3. Лістинг програми:**

**package** lab4;

**import** java.util.LinkedList;

**public** **class** Main {

**static** **int** *n* = 100;

**static** **double** *e* = 0.5;

**static** **double** *a* = -4.0;

**static** **double** *b* = 4.0;

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

Function p = **new** Polynome();

CombinedSolver cs = **new** CombinedSolver();

Viewer v = **new** Viewer(p.getXySeries(*a*, *b*, *n*, "x^3 - 2x^2 + x + 1"));

LinkedList<Double> intervals = Spliter.*getIntervals*(p, *a*, *b*, *e*);

LinkedList<Double> roots = **new** LinkedList<Double>();

LinkedList<Integer> ks = **new** LinkedList<Integer>();

**for**(Double inter: intervals) {

roots.add(cs.findRoots(p, inter, inter + *e*, *e*/*n*));

ks.add(cs.getK());

}

System.*out*.println(roots);

System.*out*.println(ks);

}

}

**package** lab4;

**import** java.security.InvalidParameterException;

**import** java.util.LinkedList;

**public** **class** Spliter {

**public** **static** LinkedList<Double> getIntervals(Function f, **double** a, **double** b, **double** e) {

**if**(a == b)

**throw** (**new** InvalidParameterException("Права і ліва границя побудови графіка рівні."));

**if**(a > b) {

**double** c;

c = a;

a = b;

b = c;

}

LinkedList<Double> result = **new** LinkedList<Double>();

**int** n = (**int**) ((b-a)/e);

**double** fa, fae;

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

fa = f.evaluate(a);

fae = f.evaluate(a + e);

**if**(fa \* fae < 0)

result.add(a);

a = a + e;

}

**return** result;

}

}

**package** lab4;

**import** java.security.InvalidParameterException;

**import** org.jfree.data.xy.XYSeries;

**public** **class** CombinedSolver {

**private** **int** k = 0;

**private** Viewer v;

**public** **double** findRoots(Function f, **double** a, **double** b, **double** e) {

**if**(f.evaluate(a)\*f.evaluate(b) > 0)

**throw** **new** InvalidParameterException("Значення функції на кінцях проміжку [a,b] повинні бути різних знаків.");

v = **new** Viewer();

String s = "Пошук кореня на відкізку [" + a + "; " + b + "]";

v.addSeries(f.getXySeries(a, b, 200, "Графік функції на відкізку [" + a + "; " + b + "]"));

k = 0;

**while**(Math.*abs*(b - a) >= e) {

**double** temp = (b-a)/2.0;

**double** fa = f.evaluate(a);

**double** fb = f.evaluate(b);

XYSeries series1 = **new** XYSeries("Хорда[" + k + "]");

XYSeries series2 = **new** XYSeries("Дотична[" + k + "]");

series1.add(a, fa);

series1.add(b, fb);

v.addSeries(series1);

**if**(f.derivative(temp) \* f.secondDerivative(temp) < 0) {

b = b - fb\*(b - a)/(fb - fa);

series2.add(a, fa);

a = a - fa/f.derivative(a);

series2.add(a, 0.0);

v.addSeries(series2);

}

**else** {

a = a - fa\*(b - a)/(fb - fa);

series2.add(b, fb);

b = b - fb/f.derivative(b);

series2.add(b, 0.0);

v.addSeries(series2);

}

k++;

}

v.display(s);

**return** (0.5\*(b+a));

}

**public** **int** getK() {

**return** k;

}

}

**package** lab4;

**public** **class** Polynome **extends** Function {

@Override

**public** **double** evaluate(**double** x) {

// x^3 - 2x^2 + x + 1 = 0;

**return** (x\*x\*x - 2\*x\*x + x + 1);

}

@Override

**public** **double** derivative(**double** x) {

// 3x^2 - 4x + 1

**return** (3\*x\*x - 4\*x + 1);

}

@Override

**public** **double** secondDerivative(**double** x) {

// 6x - 4

**return** (6\*x - 4);

}

}

**package** lab4;

**import** java.security.InvalidParameterException;

**import** org.jfree.data.xy.XYSeries;

**public** **abstract** **class** Function {

**public** **abstract** **double** evaluate(**double** x);

**public** **abstract** **double** derivative(**double** x);

**public** **abstract** **double** secondDerivative(**double** x);

**public** XYSeries getXySeries(**double** a, **double** b, **int** n, String name) **throws** InvalidParameterException{

XYSeries result = **new** XYSeries(name);

**if**(a == b)

**throw** (**new** InvalidParameterException("Права і ліва границя побудови графіка рівні."));

**double** c;

**if**(a > b) {

c = a;

a = b;

b = c;

}

**double** h = (b - a)/n ;

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

c = a + h\*i;

result.add(c, evaluate(c));

}

**return** result;

}

}

package lab4;

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;

public class Viewer {

JFrame frame;

private XYSeriesCollection dat;

public Viewer() {

frame = new JFrame();

frame.setTitle("Графік функції");

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

dat = new XYSeriesCollection();

}

public Viewer(XYSeries series) {

JFrame frame = new JFrame();

frame.setTitle("Графік функції");

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

XYSeriesCollection data = new XYSeriesCollection();

data.addSeries(series);

JFreeChart chart = ChartFactory.createXYLineChart(

"Графік функції",

"X",

"Y",

data,

PlotOrientation.VERTICAL,

true,

true,

false);

final ChartPanel chartPanel = new ChartPanel(chart);

chartPanel.setPreferredSize(new java.awt.Dimension(600, 600));

frame.setContentPane(chartPanel);

frame.pack();

frame.setVisible(true);

}

public void display(String s) {

JFreeChart chart = ChartFactory.createXYLineChart(

s,

"X",

"Y",

dat,

PlotOrientation.VERTICAL,

true,

true,

false);

final ChartPanel chartPanel = new ChartPanel(chart);

chartPanel.setPreferredSize(new java.awt.Dimension(600, 600));

frame.setContentPane(chartPanel);

frame.pack();

frame.setVisible(true);

}

public void addSeries(XYSeries series) {

dat.addSeries(series);

}

}

**4. Аналіз результатів:**

Створена мною програма знаходить корені нелінійних рівнянь з вказаною точністю на вказаному проміжку. Також у вигляді графіка покроково відображається процес знаходження кореня на відрізку.