**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

**Л А Б О Р А Т О Р Н А Р О Б О Т А № 4**

***Розв’язання нелінійних рівнянь на ЕОМ***

Виконав:

Студент ФІОТ

Групи ІО – 31

Долинний Олександр

**Київ**

**2015**

**Лабораторна робота №4**

**1. Тема:** «Розв’язання нелінійних рівнянь на ЕОМ»

**Мета:** Метою даного заняття є ознайомлення з методиками та вивчення різних алгоритмів розв’язання нелінійних рівнянь на ЕОМ.

**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:**

1. Ознайомитись з методикою розв’язання нелінійних рівнянь на ЕОМ.

2. Конкретний метод розв’язання рівняння визначає викладач.

3. Скласти схему алгоритму розв’язання нелінійного рівняння, вибраного з таблиці 1 за вказівкою викладача.

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

5. Відокремити корені в рівнянні, що досліджується. (Відокремлення коренів виконати на ЕОМ, розробивши для цього програму).

6. Вибрати значення точності обчислень ε.

7. На кожному проміжку уточнити корені, користуючись розробленою програмою. Початкові дані (проміжок, точність обчислень ε) і результати роздрукувати.

8. Використовуючи блок-схему алгоритму, написати програму розв’язування нелінійного рівняння на мові програмування.

9. Оформити звіт, що відповідає вимогам.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Варіанту** | **Метод розв’язання** | **f(x)** | **Корені** |
| **11** | **Метод хорд** |  | **0.0; 0.787** |
|  |  |  |  |

**3. Лістинг програми:**

**Function**

public class Function {

public double getValue (double x) {

return x\*x - Math.sin(Math.PI\*x);}

public double getFirstDerivative (double x) {

return 2\*x - Math.PI\*Math.cos(Math.PI\*x);}

public double getSecondDerivative (double x) {

return 2 + Math.PI\*Math.PI\*Math.sin(Math.PI \* x);

}

}

**Spliter**

public class Spliter {

public LinkedList<Double> getInterval(double a, double b, double e) {

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

Function function = new Function();

int count = (int) ((b - a) / e);

result.add(a);

for (int i = 0; i < count; i++) {

if (function.getValue(a) \* function.getValue(a + e) <= 0)

result.add(a + e);

a = a + e;

}

result.add(b);

return result;

}

}

**Solver**

import java.util.LinkedList;

public class Solver {

private double a;

private double b;

private double e;

private Function function = new Function();

private LinkedList<Double> intervals;

private int gr;

Solver(double a, double b, double e) {

this.a = a;

this.b = b;

this.e = e;

this.gr = (int) (1.0 / e);

Spliter s = new Spliter();

intervals = s.getInterval(a, b, e);

}

private double getNextA(double xx, double c) {

return xx - function.getValue(xx) \* (c - xx) / (function.getValue(c) - function.getValue(xx));

}

private double getNextB(double xx, double c) {

return xx - function.getValue(xx) \* (xx - c) / (function.getValue(xx) - function.getValue(c));

}

public LinkedList<Double> getIntervals() {

Spliter s = new Spliter();

return s.getInterval(a, b, e);

}

public LinkedList<Double> getSolution() {

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

if ((((function.getValue(a) \* function.getValue(b)) > 0) && (intervals.size() != 2))

|| ((function.getValue(a) \* function.getValue(b)) < 0)) {

for (int i = 0; i < intervals.size() - 1; i++) {

double left = intervals.get(i);

double right = intervals.get(i + 1);

if ((function.getValue(left) \* function.getSecondDerivative(left)) > 0) {

double xTemp = right;

double xPrev = xTemp - 10 \* e;

while (Math.abs(xTemp - xPrev) > e) {

xPrev = xTemp;

xTemp = getNextA(xPrev, left);

}

if ((result.size() == 0) || ((result.size() > 0) && (Math.abs(result.get(result.size() - 1) - xTemp) > e)))

result.add(xTemp);

} else {

double xTemp = left;

double xPrev = xTemp - 10 \* e;

while (Math.abs(xTemp - xPrev) > e) {

xPrev = xTemp;

xTemp = getNextB(xPrev, right);

}

if ((result.size() == 0) || ((result.size() > 0) && (Math.abs(result.get(result.size() - 1) - xTemp) > e)))

result.add(xTemp);

}

}

}

return result;

}

}

**Viewer**

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;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Formatter;

import java.util.LinkedList;

public class Viewer {

public static int FRAME\_WIDTH = 400;

public static int FRAME\_HEIGHT = 600;

JPanel panel1 = new JPanel();

GridLayout g1 = new GridLayout(12, 2);

JPanel panel2 = new JPanel();

GridLayout g2 = new GridLayout(2, 1);

JPanel panel3 = new JPanel();

GridLayout g3 = new GridLayout(1, 1);

JLabel label1 = new JLabel("Рівняння");

JLabel label2 = new JLabel("Оберіть функцію");

JLabel label3 = new JLabel("Границя a");

JTextField field3 = new JTextField(10);

JLabel label4 = new JLabel("Границя b");

JTextField field4 = new JTextField(10);

JLabel label5 = new JLabel("Точність");

JTextField field5 = new JTextField(10);

JButton task1 = new JButton("Обчислити");

JLabel label6 = new JLabel("");

JLabel label7 = new JLabel("");

JLabel label7A = new JLabel("");

JLabel label8 = new JLabel("Результати");

JTextField field8 = new JTextField(10);

XYSeriesCollection data = new XYSeriesCollection();

XYSeries series1 = new XYSeries("Графік рівняння");

public Viewer() {

//створюємо вкладку

JFrame frame = new JFrame();

frame.setTitle("Четверта лабораторна робота");

frame.setSize(FRAME\_WIDTH, FRAME\_HEIGHT);

panel1.setBackground(Color.white);

panel1.setLayout(g1);

panel2.setBackground(Color.white);

panel2.setLayout(g2);

panel1.add(label1);

panel1.add(label2);

panel1.add(label3);

panel1.add(field3);

panel1.add(label4);

panel1.add(field4);

panel1.add(label5);

panel1.add(field5);

panel1.add(task1);

panel1.add(label6);

panel1.add(label7);

panel1.add(label7A);

panel1.add(label8);

panel1.add(field8);

task1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

series1.clear();

String result = new String();

double a = Double.parseDouble(field3.getText());

double b = Double.parseDouble(field4.getText());

double e = Double.parseDouble(field5.getText());

Solver solv = new Solver(a, b, e);

LinkedList<Double> out = solv.getSolution();

if (out.size() > 0) {

Formatter fmt = new Formatter();

for (int i = 0; i < out.size(); i++) {

fmt.format("%10.5f", out.get(i));

result = result + fmt + "; ";

// result = result + out.get(i) + "; ";

}

} else {

result = "На даному проміжку рішень немає ";

}

double a1 = a - (b - a) \* 0.2;

double b1 = b + (b - a) \* 0.2;

int count = 1000;

double step = (b1 - a1) / count;

series1.clear();

Function function = new Function();

for (int i = 0; i <= (count); i++) {

series1.add(a1, function.getValue(a1));

a1 += step;

}

field8.setText(result);

}

});

data.addSeries(series1);

JFreeChart chart = ChartFactory.createXYLineChart(

"Графік рівняння",

"X",

"Y",

data,

PlotOrientation.VERTICAL,

true,

true,

false);

final ChartPanel chartPanel = new ChartPanel(chart);

chartPanel.setPreferredSize(new java.awt.Dimension(300, 400));

panel2.add(panel1);

panel2.add(chartPanel);

frame.add(panel2);

frame.setVisible(true);

}

}

**Main**

public class Main {

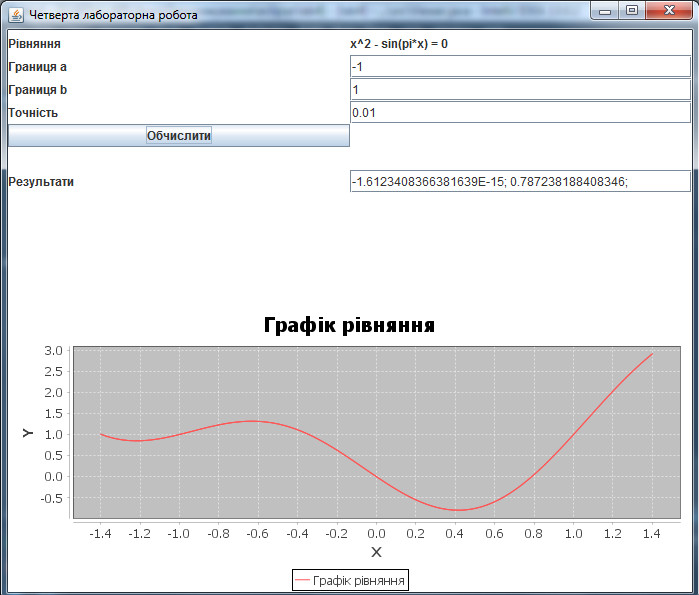
public static void main(String[] args) {

Viewer dr = new Viewer();

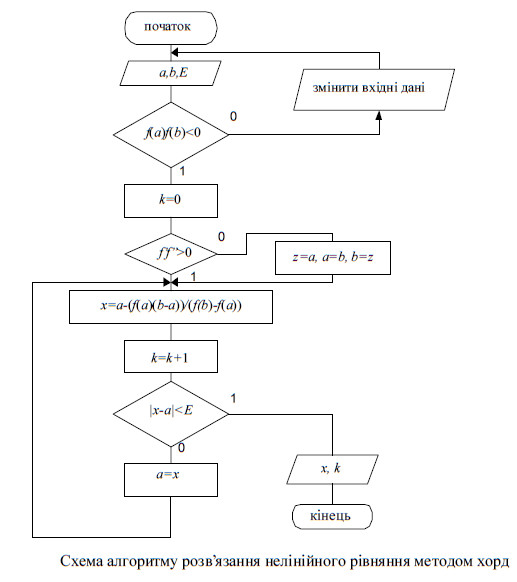
}

}

**4. Приклад**

****

**4. Блок-схема алгоритму**

****