Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3
З алгоритмів та методів обчислень

Виконав:
Студент групи ІО-22
Бас А. В.

м. Київ
2014 р.

1. Тема 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.

2. Завдання:

1) За вказівкою викладача вибрати метод інтерполяції (многочлени Лагранжа, Ньютона або рекурентне співвідношення Ейткена).

2) Скласти програму, що обчислює значення заданої функції у вузлах інтерполяції на відрізку [a, b].

3) Передбачити в програмі оцінку похибки на основі порівняння значень, отриманих за допомогою інтерполяційних многочленів різного степеня.

4) Оцінити розмитість оцінки похибки.

5) Налагодити програму шляхом інтерполяції функції sin(x*ч) (див. «Чисельний експеримент»).

6) Результат оцінки похибки представити у вигляді графіка

3. Лістинг програми:

```
public interface Function {
    public double value(double x);
}

public class Interpolation {

    enum InterpolationType {
        LINEAR,
        SQUARE
    }

    private double[] x;
    private double[] y;
    private Function function;
    private double a;
    private double b;
    private int numOfPoints;
    private InterpolationType type;
    private double deltaH;
    private double[] error;
    private double meanError = -47;

    private int n;


    private int typeToInt(InterpolationType type) {
        switch (type) {
            case LINEAR:
                return 1;
            case SQUARE:
                return 2;
            default:
                return 0;
        }
    }

    public Interpolation(Function f, int numOfPoints, InterpolationType type, double a, double
b) {

        this.a = a;
        this.b = b;
        this.numOfPoints = numOfPoints;

        this.type = type;
```

```
        n = typeToInt(type);

        deltaH = (b - a) / (numOfPoints - 1);

        x = new double[numOfPoints];
        y = new double[numOfPoints];

        function = f;
        x[0] = a;
        y[0] = function.value(x[0]);
        for (int i = 1; i < x.length; i++) {
            x[i] = x[i - 1] + deltaH;
            y[i] = function.value(x[i]);
        }
    }

    public double interpolate(double value) {
        double result = 0;
        double mul;
        for (int k = 0; k < numOfPoints - 1; k++) {
            // point not in this interval
            if (!(value >= x[k] && value < x[k + 1])) {
                continue;
            }

            if (k <= numOfPoints / 2) {
                for (int j = k; j <= k + n; j++) {
                    mul = y[j];
                    for (int i = k; i <= k + n; i++) {
                        if (i != j) {
                            mul = mul * (value - x[i]) / (x[j] - x[i]);
                        }
                    }
                    result = result + mul;
                }
            } else {
                for (int j = k + 1; j >= k - n + 1; j--) {
                    mul = y[j];
                    for (int i = k + 1; i >= k - n + 1; i--) {
                        if (i != j) {
                            mul = mul * (value - x[i]) / (x[j] - x[i]);
                        }
                    }
                    result = result + mul;
                }
            }
        }

        return result;
    }

    public double[] getError() {
        if (error != null) {
            return error;
        }

        error = new double[numOfPoints];
        for (int i = 0; i < error.length; i++) {
            double value = deltaH / 2.0 + i * deltaH;
            error[i] = Math.abs(interpolate(value)
                    - function.value(value));
        }
```

```
        return error;
    }

    public double getMeanError() {
        if (meanError > -1.0) {
            return meanError;
        }
        getError();
        double result = 0;
        for (int i = 0; i < error.length; i++) {
            result += error[i];
        }
        result /= error.length;
        return result;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < x.length; i++) {
            builder.append("(" + x[i] + ", " + y[i] + ") \n");
        }
        return builder.toString();
    }

    public Function getFunction() {
        return function;
    }

}

public class InterpolationFrame extends JFrame {
    private JPanel rootPanel;
    private JTabbedPane mTabbedPane1;
    private JTextField mTextFieldA;
    private JTextField mTextFieldB;
    private JTextField mTextFieldNum;
    private JComboBox mComboBox1;
    private JButton mButtonGo;
    private JPanel panelGraph;
    private JPanel panelError;
    private JComboBox mComboBoxFunc;

    public InterpolationFrame() {
        super("Інтерполяція");

        setContentPane(rootPanel);

        setPreferredSize(new Dimension(750, 550));
        pack();

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        setVisible(true);

        mButtonGo.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                perform();
            }
        });
    }
```

```
private void perform() {

    int numOfPoints = Integer.parseInt(mTextFieldNum.getText());

    Interpolation.InterpolationType type = null;
    if (mComboBox1.getSelectedIndex() == 0) {
        type = Interpolation.InterpolationType.LINEAR;
    } else {
        type = Interpolation.InterpolationType.SQUARE;
    }

    int a = Integer.parseInt(mTextFieldA.getText());
    int b = Integer.parseInt(mTextFieldB.getText());

    Function function = null;
    switch (mComboBoxFunc.getSelectedIndex()) {
        case 0:
            function = new Function() {
                @Override
                public double value(double x) {
                    return Math.sin(x);
                }
            };
            break;
        case 1:
            function = new Function() {
                @Override
                public double value(double x) {
                    return Math.cos(x);
                }
            };
            break;
        case 2:
            function = new Function() {
                @Override
                public double value(double x) {
                    return Math.sin(x * x);
                }
            };
            break;
        case 3:
            function = new Function() {
                @Override
                public double value(double x) {
                    return Math.cos(x * x);
                }
            };
            break;
    }

    Interpolation interpolation = new Interpolation(function, numOfPoints, type, a, b);

    XYSeries series0 = new XYSeries("Функція");
    XYSeries seriesI = new XYSeries("Інтерполяція");
    XYSeries seriesDiff = new XYSeries("Похибка");
    double tempI;
    double tempO;
    for (int i = 0; i < 50 * b; i++) {
        double value = i / 50.0f;
        tempI = interpolation.interpolate(value);
        tempO = interpolation.getFunction().value(value);
        series0.add(value, tempO);
        seriesI.add(value, tempI);
        seriesDiff.add(value, Math.abs(tempI - tempO));
```

```
        }

        XYSeriesCollection dataGraph = new XYSeriesCollection();
        XYSeriesCollection dataError = new XYSeriesCollection();

        dataGraph.addSeries(series0);
        dataGraph.addSeries(seriesI);

        dataError.addSeries(seriesDiff);

        final JFreeChart chartGraph = ChartFactory.createXYLineChart(
                "Декартова система", "X", "Y", dataGraph, PlotOrientation.VERTICAL,
                true, true, false);

        final JFreeChart chartError = ChartFactory.createXYLineChart(
                "Декартова система", "X", "Y", dataError, PlotOrientation.VERTICAL,
                true, true, false);

        final ChartPanel chartPanelGraph = new ChartPanel(chartGraph);
        final ChartPanel chartPanelError = new ChartPanel(chartError);

        mTabbedPane1.removeAll();
        mTabbedPane1.addTab("Графік", chartPanelGraph);
        mTabbedPane1.addTab("Похибка", chartPanelError);

        System.out.println("\nПохибка на кожному проміжку: ");
        double[] error = interpolation.getError();
        for (int i = 0; i < error.length; i++) {
            System.out.format("%2.8f\n", error[i]);
        }

        System.out.println("\nСередня похибка: ");
        System.out.format("%.8f%n", interpolation.getMeanError());
    }

}
```