Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни «Алгоритми та методи обчислень»

Виконав:
студент групи ІО-33
Шуркіна Анастасія

Перевірив:
Порєв В. М.

.

Київ 2015

Тема: Інтерполяція функцій.

Мета: Ознайомлення з інтерполяційними формулами Лагранжа, Ньютона, рекурентним співвідношенням Ейткена, методами 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.

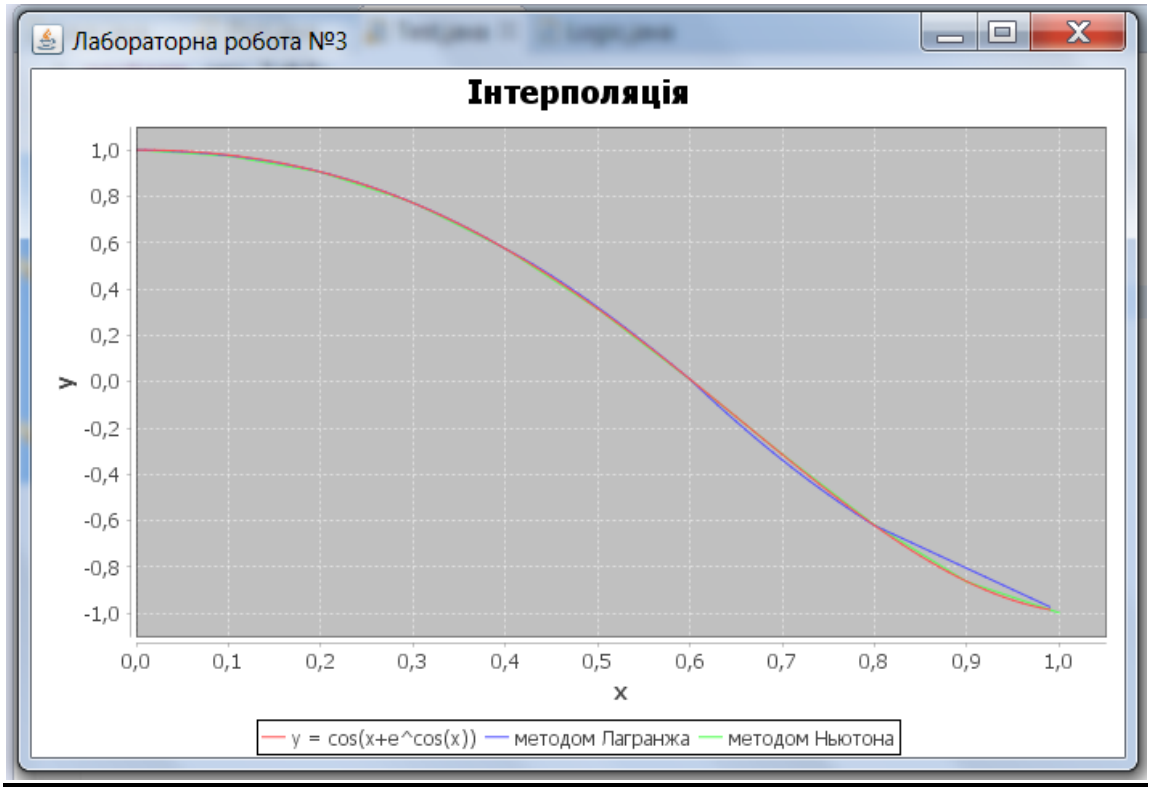
Варіант завдання: 14 $\cos\left(2x + x^2\right)$ $[0, 1]$

Код програми:

```
package amo_lab3;
public class Logic {
	private final int t = 6; // 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
	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();
	}
	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]);
		}
	}
	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]);
			}
		}
	}
	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;
	}
	public double func(double x) {
		return Math.cos(2 * x + Math.pow(x, 2));
		// return Math.sin(x);
	}
	public double mistake(double x) {
		return Math.abs(func(x) - polinom(x));
	}
	public static int factorial(int n) {
		if (n == 0)
			return 1;
		return n * factorial(n - 1);
	}
	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;
	}
	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 amo_lab3;
public class Lagr {
	private double values[][];
	private double a;
	private double b;

	public void showTable() {
```

```
                double interpol[] = new
double[5];
                for (int i = 1; i <
interpol.length; i++) {
                        countTable(a, b, i);
                        interpol[i] = lagr(a, b,
3.232323, i);
                }
                double delta;
                double deltaExact;
                double kDelta;
                for (int i = 1; i <
interpol.length - 1; i++) {
                        delta = interpol[i] -
interpol[i + 1];
                        countTable(a, b, i);
                        deltaExact = interpol[i]
- getFunctionValue(3.232323);
                        kDelta = Math.abs(1 -
delta / deltaExact);
                        System.out.print("n = "
+ i);
                        System.out.print(" delta
= " + delta);
                        System.out.print("
deltaExact = " + deltaExact);
                        System.out.print("
kDelta = " + kDelta);
                        System.out.println();
                }
        }

        public int fact(int n) {
                int result = 1;
                for (int i = 1; i < n + 1; i++) {
                        result *= i;
                }
                return result;
        }

        public void countTable(double a, double
b, int n) {
                this.a = a;
                this.b = b;
                double h = (b - a) / n;
                values = new double[n + 1][2];
                for (int i = 0; i <
values.length; i++) {
                        values[i][0] = a + i *
h;
                        values[i][1] =
getFunctionValue(a + i * h);
                }
        }

        public double lagr(double a, double b,
double x, int n) {
                double h = (b - a) / n;
                double m = (x - a) / h;

                double result = 0;
                double inres = 1;

                for (int i = 0; i <
values.length; i++) {
                        inres = 1;
                        for (int j = 0; j <
values.length; j++) {
                                if (i != j) {
                                        inres *=
(m - j);
                                }
                        }
                        result += (Math.pow(-1,
(n - i))) * inres * values[i][1]

        / (fact(i) * fact(n - i));
                }
                return result;
        }

        public double getFunctionValue(double x)
{               return Math.cos(2 * x +
Math.pow(x, 2));
        }
}
package amo_lab3;

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;
import org.jfree.ui.RefineryUtilities;
public class Plot {

        public Plot(Logic l, Lagr g) {
                double a = 0;
                double b = 1;
                int n = 5;
                g.countTable(a, b + 0.1, n);

                XYSeries series1 = new
XYSeries("y = cos(x+e^cos(x))");
                XYSeries series2 = new
XYSeries("методом Лагранжа");
                XYSeries series3 = new
XYSeries("методом Ньютона");
                // XYSeries series4 = 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));
                        // series4.add(j,
l.mistake(j));
                }

                double step = 0.1;
                for (int i = 0; i < b * 10 + 1;
i++)
                        series3.add(step * i,
g.lagr(a, b + 0.1, step * i, n));

                XYSeriesCollection data = new
XYSeriesCollection(series1);
                data.addSeries(series2);
                data.addSeries(series3);



                JFreeChart chart1 =
ChartFactory.createXYLineChart("Інтерполяція",
"x",
                                        "y", data,
PlotOrientation.VERTICAL, true, true, true);
                ChartPanel chartPanel1 = new
ChartPanel(chart1);
                JFrame frame = new
JFrame("Лабораторна робота №3");
                frame.setLayout(new
BorderLayout());
                frame.add(chartPanel1);
                frame.pack();
                frame.setVisible(true);
```

```
        frame.setDefaultCloseOperation(JFrame.EXI
T_ON_CLOSE);
        }
}
package amo_lab3;

public class Test {
        public static void main(String[] args) {
                double a = 0;
                double b = 1;
                double x = 1;
                int k = 4;

                Logic l = new Logic(a, b, k);
                Lagr g = new Lagr();
                Plot p = new Plot(l, g);
        }
}
```



Висновок: Під час виконання даної лабораторної роботи були закріплені навички інтерполяції функції різними методами. Програмно були реалізовані методи інтерполяції поліномом Лагранжа та поліномом Ньютона. Окрім того були закріплені основні навички роботи з графічною бібліотекою JFreeChart.