Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №6
Вычислительная математика

Вариант: №1

Группа	P3208
Студент	Абдуллин И.Э.
Преподаватель	Машина Е.А.

Цель работы:

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Код программы

```
package Abdullin_367039.lab6;


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.ApplicationFrame;


import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.function.BiFunction;


public class DifferentialMethods extends ApplicationFrame {


 public DifferentialMethods(String title, XYSeriesCollection dataset) {
   super(title);
   JFreeChart chart = ChartFactory.createScatterPlot(
           title,
           "X", "Y",
           dataset,
           PlotOrientation.VERTICAL,
           true, true, false);
   ChartPanel chartPanel = new ChartPanel(chart);
   chartPanel.setPreferredSize(new java.awt.Dimension(800, 600));
   setContentPane(chartPanel);
 }


 public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);


   System.out.println("1. y' = x");
   System.out.println("2. y' = e^x");
   System.out.println("3. y' = x^2");
   System.out.print("Выберите функцию: ");
   int mode = scanner.nextInt();
```

```
    System.out.print("Введите n: ");
    int n = scanner.nextInt();

    System.out.print("Введите x0: ");
    double x0 = scanner.nextDouble();

    System.out.print("Введите xn: ");
    double xn = scanner.nextDouble();

    double h = (xn - x0) / n;
    List<Double> xs = new ArrayList<>();
    for (int i = 0; i <= n; i++) {
      xs.add(x0 + h * i);
    }

    BiFunction<Double, Double, Double> f = null;
    BiFunction<Double, Double, Double> exactY = null;

    switch (mode) {
      case 1:
        f = (x, y) -> x;
        exactY = (x, y) -> x * x / 2 + 1;
        break;
      case 2:
        f = (x, y) -> Math.exp(x);
        exactY = (x, y) -> Math.exp(x) - 1;
        break;
      case 3:
        f = (x, y) -> x * x;
        exactY = (x, y) -> x * x * x / 3 + 5;
        break;
      default:
        System.out.println("Неверный ввод");
        System.exit(1);
    }

    double y0 = exactY.apply(x0, null);

    runAndPlotMethod("Euler Method", f, xs, y0, exactY, DifferentialMethods::eulerMethod);
    runAndPlotMethod("Runge-Kutta Method", f, xs, y0, exactY,
DifferentialMethods::rungeKuttaMethod);
```

```
    runAndPlotMethod("Adams Method", f, xs, y0, exactY, DifferentialMethods::adamsMethod);
  }

  private static List<Double> eulerMethod(BiFunction<Double, Double, Double> f, List<Double>
xs, double y0) {
    List<Double> ys = new ArrayList<>();
    ys.add(y0);
    double h = xs.get(1) - xs.get(0);

    for (int i = 1; i < xs.size(); i++) {
      double xi = xs.get(i - 1);
      double yi = ys.get(i - 1);
      ys.add(yi + h * f.apply(xi, yi));
    }
    return ys;
  }

  private static List<Double> rungeKuttaMethod(BiFunction<Double, Double, Double> f,
List<Double> xs, double y0) {
    List<Double> ys = new ArrayList<>();
    ys.add(y0);
    double h = xs.get(1) - xs.get(0);

    for (int i = 1; i < xs.size(); i++) {
      double xi = xs.get(i - 1);
      double yi = ys.get(i - 1);
      double k1 = h * f.apply(xi, yi);
      double k2 = h * f.apply(xi + h / 2, yi + k1 / 2);
      double k3 = h * f.apply(xi + h / 2, yi + k2 / 2);
      double k4 = h * f.apply(xi + h, yi + k3);
      ys.add(yi + (k1 + 2 * k2 + 2 * k3 + k4) / 6);
    }
    return ys;
  }

  private static List<Double> adamsMethod(BiFunction<Double, Double, Double> f, List<Double>
xs, double y0) {
    List<Double> ys = rungeKuttaMethod(f, xs.subList(0, 4), y0);
    double h = xs.get(1) - xs.get(0);

    for (int i = 4; i < xs.size(); i++) {
      double xi1 = xs.get(i - 1);
```

```
    double xi2 = xs.get(i - 2);
    double xi3 = xs.get(i - 3);
    double xi4 = xs.get(i - 4);
    double yi1 = ys.get(i - 1);
    double yi2 = ys.get(i - 2);
    double yi3 = ys.get(i - 3);
    double yi4 = ys.get(i - 4);

    double f1 = f.apply(xi1, yi1);
    double f2 = f.apply(xi2, yi2);
    double f3 = f.apply(xi3, yi3);
    double f4 = f.apply(xi4, yi4);

    double df = f1 - f2;
    double d2f = f1 - 2 * f2 + f3;
    double d3f = f1 - 3 * f2 + 3 * f3 - f4;

    double y = yi1 + h * f1 + (h * h / 2) * df + (5 * Math.pow(h, 3) / 12) * d2f + (3 *
Math.pow(h, 4) / 8) * d3f;
    ys.add(y);
  }
  return ys;
 }

 private static void runAndPlotMethod(String methodName, BiFunction<Double, Double, Double> f,
List<Double> xs, double y0, BiFunction<Double, Double, Double> exactY, MethodRunner method) {
  List<Double> ys = method.run(f, xs, y0);

  double maxError = 0.0;
  StringBuilder table = new StringBuilder();
  table.append("+--------------------+\n");
  table.append("|    x    |    y    |\n");
  table.append("+--------------------+\n");

  for (int i = 0; i < xs.size(); i++) {
    double x = xs.get(i);
    double y = ys.get(i);
    double exact = exactY.apply(x, null);
    double error = Math.abs(y - exact);
    if (error > maxError) {
      maxError = error;
    }
```

```
      table.append(String.format("| %.5f | %.5f |\n", x, y));
    }
    table.append("+-------------------+\n");

    System.out.printf("%s - max error: %.5f\n", methodName, maxError);
    System.out.println(table);

    XYSeries series = new XYSeries(methodName);
    for (int i = 0; i < xs.size(); i++) {
      series.add(xs.get(i), ys.get(i));
    }

    XYSeries exactSeries = new XYSeries("Exact");
    for(double i = xs.get(0); i < xs.get(xs.size()-1); i+=0.001) {
      exactSeries.add(i, exactY.apply(i, null));
    }
    for (double x : xs) {
      exactSeries.add(x, exactY.apply(x, null));
    }

    XYSeriesCollection dataset = new XYSeriesCollection();
    dataset.addSeries(series);
    dataset.addSeries(exactSeries);

    DifferentialMethods chart = new DifferentialMethods(methodName, dataset);
    chart.pack();
    chart.setVisible(true);
  }

  @FunctionalInterface
  interface MethodRunner {
    List<Double> run(BiFunction<Double, Double, Double> f, List<Double> xs, double y0);
  }
}
```

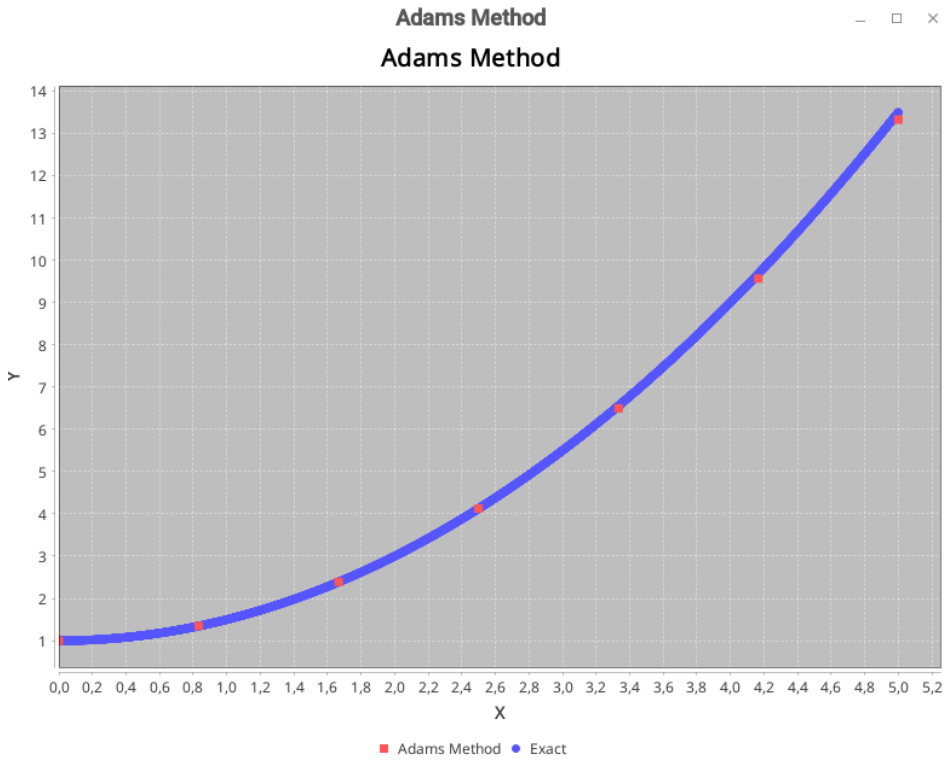
Вывод программы:

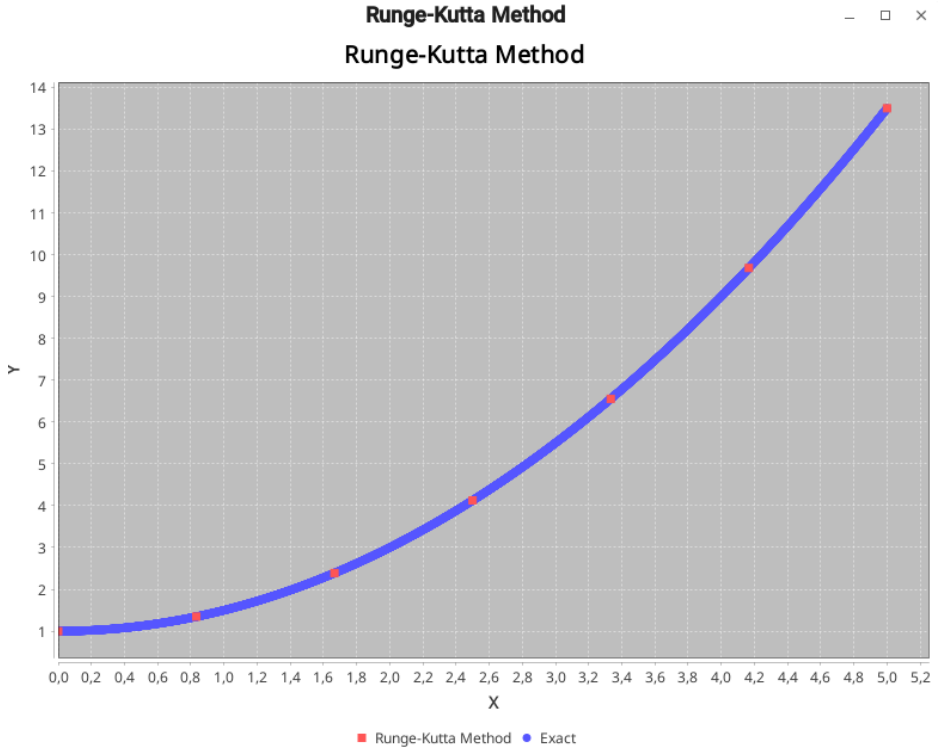
```
1. y' = x
2. y' = e^x
3. y' = x^2
Выберите функцию: 1
Введите n: 6
```

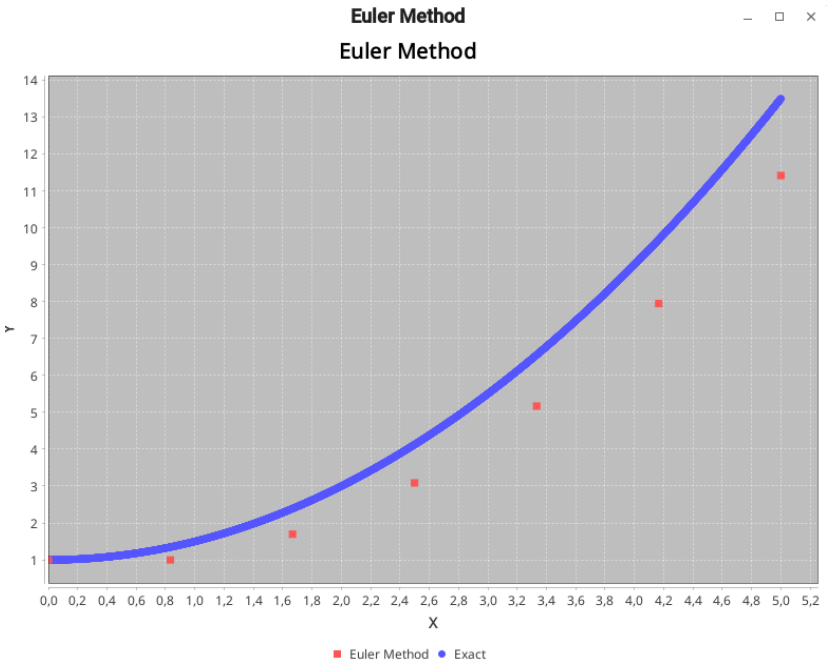
```
Введите x0: 0
Введите xn: 5
Euler Method - max error: 2,08333
+--------------------+
|      x      |      y      |
+--------------------+
| 0,00000 | 1,00000 |
| 0,83333 | 1,00000 |
| 1,66667 | 1,69444 |
| 2,50000 | 3,08333 |
| 3,33333 | 5,16667 |
| 4,16667 | 7,94444 |
| 5,00000 | 11,41667 |
+--------------------+

Runge-Kutta Method - max error: 0,00000
+--------------------+
|      x      |      y      |
+--------------------+
| 0,00000 | 1,00000 |
| 0,83333 | 1,34722 |
| 1,66667 | 2,38889 |
| 2,50000 | 4,12500 |
| 3,33333 | 6,55556 |
| 4,16667 | 9,68056 |
| 5,00000 | 13,50000 |
+--------------------+

Adams Method - max error: 0,17361
+--------------------+
|      x      |      y      |
+--------------------+
| 0,00000 | 1,00000 |
| 0,83333 | 1,34722 |
| 1,66667 | 2,38889 |
| 2,50000 | 4,12500 |
| 3,33333 | 6,49769 |
| 4,16667 | 9,56481 |
| 5,00000 | 13,32639 |
+--------------------+
```


Adams Method
Adams Method
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Y
0,0 0,2 0,4 0,6 0,8 1,0 1,2 1,4 1,6 1,8 2,0 2,2 2,4 2,6 2,8 3,0 3,2 3,4 3,6 3,8 4,0 4,2 4,4 4,6 4,8 5,0 5,2
X
Adams Method
Exact


Runge-Kutta Method
Runge-Kutta Method
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Y
0,0 0,2 0,4 0,6 0,8 1,0 1,2 1,4 1,6 1,8 2,0 2,2 2,4 2,6 2,8 3,0 3,2 3,4 3,6 3,8 4,0 4,2 4,4 4,6 4,8 5,0 5,2
X
Runge-Kutta Method
Exact



Вывод

В ходе лабораторной работы я научился интерполировать функции с помощью метода Лагранжа, метода Ньютона с разделенными и конечными разностями, метода Гаусса.