СПбНИУ ИТМО
Кафедра ИПМ

Лабораторная работа №4
по дисциплине
«Вычислительная математика»
«Решение систем ОДУ
методом Рунге-Кутты»

Выполнил
Широков О.И
гр.2120

Санкт-Петербург
г.2013

1. Теория

Обыкновенным ДУ первого порядка называется уравнение вида

$$y' = \frac{dy}{dx} = F(x,y)$$

Решение этого уравнения в общем виде выглядит так

$$\int_{y_i}^{y_{i+1}} dy = \int_{x_i}^{x_{i+1}} F(x,y)dx$$

Для i-го узла сетки:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} F(x,y)dx$$

Т.е. мы получили выражение для вычисления значения у в i-ом узле сетки. Сложность состоит лишь во взятии интеграла.

Чтобы получить высокую точность при вычислении интеграла используют формулу, основанную на формуле Симпсона

$$y_{i+1} = y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

где

$$k_0 = hF(x_i,\ y_i)$$

$$k_1 = hF\left(x_i + \frac{h}{2},\ y_i + \frac{k_0}{2}\right)$$

$$k_2 = hF\left(x_i + \frac{h}{2},\ y_i + \frac{k_1}{2}\right)$$

$$k_3 = hF(x_i + h,\ y_i + k_2)$$

Метод Ру́нге — Ку́тты непосредственно обобщается на случай систем обыкновенных дифференциальных уравнений путём записи системы и метода в векторной форме.

2. Исходный код

```
package RungeKutta;

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


public class Solver
{
    RightPartFunction[] Functions;
    double a;
    double b;
    double step;
    int step_n;
    double[] values;

    public Solver(double la,double lb, double lstep, RightPartFunction[] lFunctions)
    {
        this.a = la;
        this.b = lb;
        this.step = lstep;
        this.Functions = lFunctions;
        step_n = 0;
        values = new double[2];
    }


    public XYSeriesCollection solve(double[] XStart, double[] YStart)
    {
        int functions_num = Functions.length;
        double[] YPrevs = new double[functions_num];


        XYSeries[] Serieses = new XYSeries[functions_num];
        for(int i = 0; i < functions_num; i++)
        {
            YPrevs[i] = YStart[i];
            Serieses[i] = new XYSeries(Integer.toString(i));
        }
            XYSeriesCollection ReturnValue = new XYSeriesCollection();

        for(double current_x = a; current_x < b; current_x+= step )
        {
            double[] FirstFactors  = getFirstFactors(current_x, YPrevs);
            double[] SecondFactors = getSecondFactors(current_x, YPrevs, FirstFactors);
            double[] ThirdFactors  = getThirdFactors(current_x, YPrevs, SecondFactors);
            double[] FourthFactors = getFourthFactors(current_x, YPrevs, ThirdFactors);

            for(int i = 0; i < functions_num; i++)
            {
                YPrevs[i] += (FirstFactors[i] + 2*SecondFactors[i] + 2*ThirdFactors[i] +
FourthFactors[i])/6;
                Serieses[i].add(current_x, YPrevs[i]);
            }
        }

        for(int i = 0; i < functions_num; i++)
            ReturnValue.addSeries(Serieses[i]);

        values[step_n] = YPrevs[functions_num - 1];
        step_n++;
        step /= 2;
        return ReturnValue;
    }

    double getRungePrec()
    {
        return (values[1] - values[0])/15;
    }

    double[] getFirstFactors(double X, double[] prevValues)
    {
```

```
        int n = prevValues.length;
        double[] retValue = new double[n];
        for(int i = 0; i < n; i++)
        {
            retValue[i] = step*Functions[i].getValue(X, prevValues);
        }
        return retValue;
    }

    double[] getSecondFactors(double X, double[] prevValues, double[] FirstFactors)
    {
        int n = FirstFactors.length;
        double[] SecondFactors = new double[n];
        double[] Yargs         = new double[n];

        for(int i = 0; i < n; i++)
        {
            Yargs[i] = prevValues[i] + FirstFactors[i]/2;
        }

        for(int i = 0; i < n; i++)
        {
            SecondFactors[i] = step*Functions[i].getValue(X + step/2, Yargs);
        }

        return SecondFactors;
    }

    double[] getThirdFactors(double X, double[] prevValues, double[] SecondFactors)
    {
        int n = SecondFactors.length;
        double[] ThirdFactors = new double[n];
        double[] YArgs        = new double[n];

        for(int i = 0; i < n; i++)
        {
            YArgs[i] = prevValues[i] + SecondFactors[i]/2;
        }

        for(int i = 0; i < n; i++)
        {
            ThirdFactors[i] = step*Functions[i].getValue(X + step/2, YArgs);
        }

        return ThirdFactors;
    }

    double[] getFourthFactors(double X, double[] prevValues, double[] ThirdFactors)
    {
        int n = ThirdFactors.length;
        double[] FourthFactors = new double[n];
        double[] YArgs = new double[n];

        for(int i = 0; i < n; i++)
            YArgs[i] = prevValues[i] + ThirdFactors[i];

        for(int i = 0; i < n; i++)
            FourthFactors[i] = step*Functions[i].getValue(X + step, YArgs);

        return FourthFactors;
    }

}
```

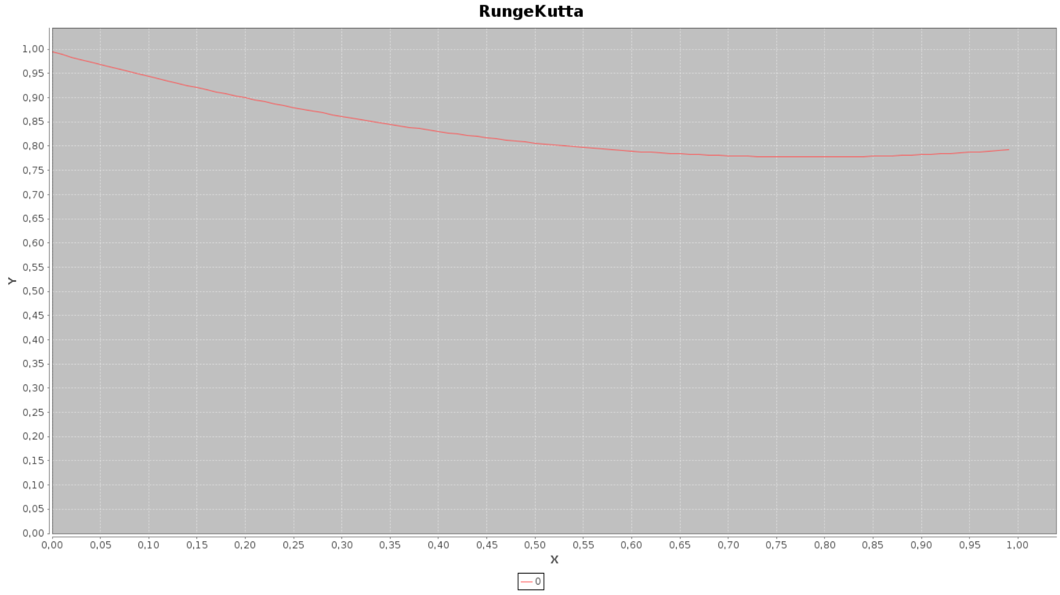
3. *Тесты*

а. Проверим решение задачи Коши для одного уравнения

начальные условия $y(0)=1$

правая часть: $\sin(x)-\cos(y)$

на промежутке от 0 до 1 с шагом 0.02



б. Решение системы из 2 ОДУ

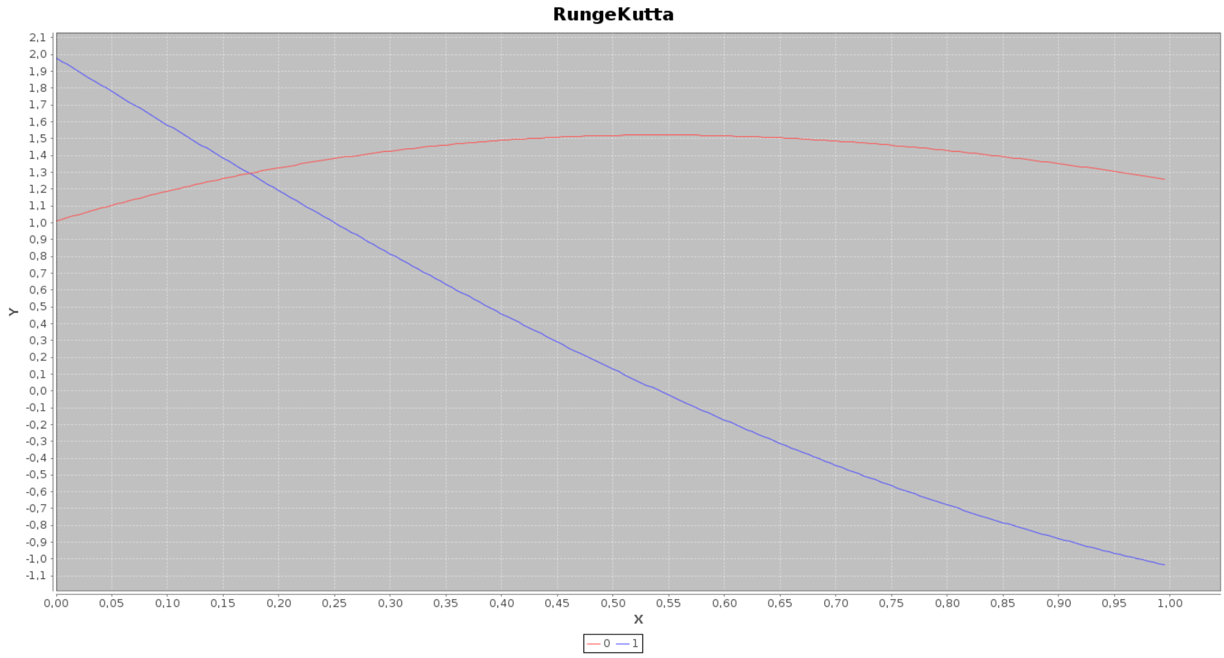
начальные условия $\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$

правые части $f(x,y,z)=z$

$g(x,y,z)=-z-2*y$

с шагом интегрирования 0.001

на диапазоне от 0 до 0.5



4. Выводы
В ходе выполнения ЛР был изучен и реализован программно метод Рунге-Кутта для решения систем ОДУ