

实验课程：数值计算	年级：2019	实验成绩：
实验名称：矩阵特征值问题计算	姓名：林子炫	
实验编号：5	学号：10195102468	实验日期：2021-12-01
指导教师：谢堇奎	组号：	实验时间：9：00AM

1 实验目的与要求

- 1、根据初值问题数值算法，分别选择二个初值问题编程计算；
- 2、试分别取不同步长，考察某节点 x_j 处数值解的误差变化情况；
- 3、试用不同算法求解某初值问题，结果有何异常；
- 4、分析各个算法的优缺点。

目的和意义

- 1、熟悉各种初值问题的算法，编出算法程序；
- 2、明确各种算法的精度与所选步长有密切关系；
- 3、通过计算更加了解各种算法的优越性。

2 实验环境

win10 + java

3 实验过程与分析

3.1 框架搭建

我们需要在图形界面中输入两个参数，A和B。

```
class Fun{  
    ...  
}
```

在下面的框架中，`public class solveODE` 为主Public类，类中定义了许多Static类型的静态变量，在下面注释中有所解释。类内的函数有：

```
public void updateModeStr(int num); // 更新模式  
public void initMenuBar(); // 初始化顶部菜单  
public void initUI(); // 初始化菜单  
public void processInput(String sa); // 处理输入函数
```

所以总体的框架如下：

```
import java.applet.Applet;  
import java.awt.*;
```

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import java.awt.Graphics;
public class SolveODE {

    static String strb;
    static ArrayList<String> strA = new ArrayList<String>();

    static JLabel labelImg = new JLabel();//载入函数图片的jlabel
    static double step;//方程步长
    static double low,high;//方程上下限
    static double yStart;//初始值
    static Fun fun = new Fun();
    static String SolveType = new String("NULL");
    // 默认的初始模式是空模式
    static String ModeType = new String("NULL");
    //
    static int SolveTypeInt = 0;// 1 2 3 4
    // 表示选择的函数 有1, 2, 3, 4四个函数
    static Graphics g;
    static int flag = 0;
    // 暂时不会用到画图
    static JFrame frame = new JFrame();
    // 定义了一个Frame
    static String result = new String("");
    /*
     * 下面对JTextField进行static的初始化定义，方便下面直接对其修改
     */
    static JTextField jFieldMode = new JTextField(120);// 模式选择
    static JTextField jFieldResult = new JTextField(120);
    static JTextField jFieldResultLam = new JTextField(120);//设置特征值
    static JTextField jFieldResult2 = new JTextField(120);//设置步长
    static JTextField jFieldResult3 = new JTextField(120);//设置上下限
    static JTextField jFieldResult4 = new JTextField(120);//设置初始值
    public static void main(String[] args) {
        System.out.println("Test Success!");
        SolveODE NI = new SolveODE();

        NI.initMenuBar();// 初始化菜单栏
        NI.initUI();// 初始化UI界面
    }

    /**
     * 处理文本框输入的函数
     * @param ABNE
     */
    public void processInput(String sa)
    {
        ...
    }

    public void updateModeStr(int num)//
    {
        //更新模式
    }
    public void initUI()
    {

```

```

        ...
    }
    public void initMenuBar()
    {
        ...
    }
}

class Fun {
    private double a,b;//上下限`
    private double h,y0;//
    private double funNumber;//选择某个函数
    /**
     *
     * @param aa 方程下限
     * @param bb 方程上限
     * @param step 步长
     * @param ys 初始值
     * @param num 选择的方程的编号
     */
    public void setData(double aa, double bb,double step,double ys,int num);
    public double fun(double x,double y);//返回函数值
    public String calEuler();//三种求解方法
    public String calEulerImproved();//
    public String calRK();//
}

```

3.2 实现输入输出

输入参数本实验设置为三个，步长设置，上下限限制和初始值，分别在JTextField文本框中实现。

```

static JTextField jFieldResultLam = new JTextField(120);//设置特征值
static JTextField jFieldResult2 = new JTextField(120);//设置步长
static JTextField jFieldResult3 = new JTextField(120);//设置上下限
static JTextField jFieldResult4 = new JTextField(120);//设置初始值

```

按了计算按钮后，读取文本框的值并进行相关处理。

```

button1.addActionListener(new ActionListener()// 对按钮增加监听
{
    // 此处需要使用的是匿名类，需要重写actionPerformed函数，否则会出错
    @Override
    public void actionPerformed(ActionEvent e) {

```

```

//处理输入步长和上下限
String []ab = jTextFieldResult3.getText().split(" ");
low = Double.parseDouble(ab[0]);
high = Double.parseDouble(ab[1]);
step = Double.parseDouble(jTextFieldResult2.getText());
yStart = Double.parseDouble(jTextFieldResult4.getText());
fun.setData(low, high, step, yStart, flag);
//System.out.println(fun.calEuler());
if (SolveTypeInt == 1)
    jTextFieldResult.setText(fun.calEuler());
if (SolveTypeInt == 2)
    jTextFieldResult.setText(fun.calEulerImproved());
if (SolveTypeInt == 3)
    jTextFieldResult.setText(fun.calRK());
    }
});

```

3.3 更新 UI

对求解模式选定的结果做出更新，在 `updateModeStr` 中得以实现。

```

if (num == 1) {
    SolveType = new String("Euler法");
    SolveTypeInt = 1;
    jTextFieldMode.setText(SolveType);
} else if (num == 2) {
    SolveType = new String("改进Euler法");
    SolveTypeInt = 2;
    jTextFieldMode.setText(SolveType);
} else if (num == 3) {
    SolveType = new String("R-K法");
    SolveTypeInt = 3;
    jTextFieldMode.setText(SolveType);
}

```

3.4 初始化 UI

3.4.1 java常用的组件类型

1、容器组件类

所谓容器，就是类似于收纳盒、包、锅碗瓢盆等可以容纳东西的物体。类似地，容器组件就是指可以容纳其他组件的组件，最典型的就是我们经常看到的窗口（窗体）组件。

JFrame是SWING包下的顶级容器组件类。所谓顶级容器，就是说它只能装别的组件，而不能被其他组件所包含。JFrame的作用就是实现一个基本的窗口以及其开关、调整大小等作用。

JPanel是SWING包下的一个容器组件，我们称之为“面板”，可以加在窗体上以实现我们想要的各种布局。

2、元素组件类

元素组件就是想按钮、标签、复选框等的一类实现某种具体功能的组件。我们经常使用的有以下几种：

JLabel 标签元素组件类 显示文字或者图片

TextField 文本输入框元素组件类 接收输入信息，将输入信息显示出来

PasswordField 密码输入框元素组件类 接收输入信息，将输入的信息以某个符号代替显示

CheckBox 复选框(多选框)元素组件类 首先又一个选择框，在选择框后还能显示文字或者图片信息

Button 按钮元素组件类 显示文字或图片，提供一个点击效果

3.4.1 布局设置

首先对frame的size进行了设置，然后对frame的布局设置成自定义布局，方便下面进行排布。

```
frame.setSize(800,600);//设置容器尺寸
frame.setLayout(new BorderLayout());
```

然后设置了Jpanel放置在Jframe上，

```
JPanel p = new JPanel();
p.setLayout(null);
p.setOpaque(false);
```

随后定义了5个label来显示指示信息，并将其add到panel上。

这里需要注意的是，我们对每一个label对定义了bounds，即它的长宽和位于panel的x和y的位置。即
void java.awt.Component.setBounds(int x, int y, int width, int height)

```
JLabel label = new JLabel("当前选择的常微分方程为: ");
label.setBounds(20, 50, 200, 20);
label.setForeground(Color.BLUE);
p.add(label);

JLabel label1 = new JLabel("当前选择的常微分方程解法: ");
label1.setBounds(20, 20, 200, 20);
label1.setForeground(Color.BLUE);
p.add(label1);

JLabel label7 = new JLabel("对应的数值解: ");
label7.setBounds(20, 385, 200, 20);
label7.setForeground(Color.BLUE);
p.add(label7);

JLabel label8 = new JLabel("步长设置:");
label8.setBounds(500, 20, 200, 20);
label8.setForeground(Color.BLUE);
p.add(label8);

JLabel label11 = new JLabel("上下限设置: ");
label11.setBounds(500, 60, 200, 20);
label11.setForeground(Color.BLUE);
p.add(label11);

JLabel label12 = new JLabel("初始值: ");
label12.setBounds(500, 100, 200, 20);
label12.setForeground(Color.BLUE);
p.add(label12);
```

随后添加开始计算按钮。

```
JButton button1 = new JButton("开始计算");//
button1.setBounds(400, 320, 300, 40);// 设置按钮在容器中的位置
p.add(button1);
```

并对按钮添加点击事件，可以看到实际上这个接口里仅仅有一个方法——“actionPerformed”这个方法就是可以实现动作监听的方法。我们在应用中可以继承这个接口，重写方法并且定义一个“ActionEvent”类型的对象作为参数传到方法里面，然后用“e.getActionCommand();”这个方法获取组件上的字符串，以进行相应的操作。

```

button1.addActionListener(new ActionListener()// 对按钮增加监听
{
    // 此处需要使用的是匿名类，需要重写actionPerformed函数，否则会出错
    @Override
    public void actionPerformed(ActionEvent e) {
        //处理输入步长和上下限
        String []ab = jTextField3.getText().split(" ");
        low = Double.parseDouble(ab[0]);
        high = Double.parseDouble(ab[1]);
        step = Double.parseDouble(jTextField2.getText());
        yStart = Double.parseDouble(jTextField4.getText());
        //设置fun的参数!!!!!!
        fun.setData(low, high, step, yStart, flag);
        //System.out.println(fun.calEuler());
        if (SolveTypeInt == 1)
            jTextField.setText(fun.calEuler());
        if (SolveTypeInt == 2)
            jTextField.setText(fun.calEulerImproved());
        if (SolveTypeInt == 3)
            jTextField.setText(fun.calRK());
    }
});

```

下面函数结尾的必要设置

```

/**
 * 这里是函数结尾的必要设置
 */

frame.getContentPane().add(p2);
frame.getContentPane().add(p);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //界面结束后关闭程序
frame.setLocationRelativeTo(null); //在屏幕上居中显示框架
frame.setVisible(true); //界面可视化，需要放在最后面，对所有的组件进行渲染。

```

initUI代码如下:

```

public void initUI() {

    /**
     * 这里是对frame的设置
     */
    frame.setSize(800, 600); // 设置容器尺寸
    frame.setLayout(new BorderLayout());
    /**
     * 中间容器
     */
    JPanel p2 = new JPanel() {

        public void paint(Graphics g) {
            super.paint(g);
            g.drawLine(350, 100, 500, 400);
        }
    };
}

```

```

JPanel p = new JPanel();
p.setLayout(null);
p.setOpaque(false);

/**
 * 这里是对labels的设置
 */
JLabel label = new JLabel("当前选择的常微分方程为: ");
label.setBounds(20, 50, 200, 20);
label.setForeground(Color.BLUE);
p.add(label);

JLabel label1 = new JLabel("当前选择的常微分方程解法: ");
label1.setBounds(20, 20, 200, 20);
label1.setForeground(Color.BLUE);
p.add(label1);

// JLabel label6 = new JLabel("请输入需要求解的方程组的B: ");
// label6.setBounds(20, 310, 200, 20);
// label6.setForeground(Color.BLUE);
// p.add(label6);

JLabel label7 = new JLabel("对应的数值解: ");
label7.setBounds(20, 385, 200, 20);
label7.setForeground(Color.BLUE);
p.add(label7);

jFieldResult.setText("当前结果: 未显示");
jFieldResult.setEditable(false);
jFieldResult.setBounds(20, 410, 600, 30);
jFieldMode.setForeground(Color.RED);
p.add(jFieldResult);

/*
JLabel label9 = new JLabel("按模最大或最小的特征值: ");
label9.setBounds(20, 445, 200, 20);
label9.setForeground(Color.BLUE);
p.add(label9);
*/

/*
jFieldResultLam.setText("当前结果: 未显示");
jFieldResultLam.setEditable(false);
jFieldResultLam.setBounds(20, 470, 600, 30);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResultLam);
*/

jFieldMode.setText("当前求解方法: 未选择");
jFieldMode.setEditable(false);
jFieldMode.setBounds(250, 20, 150, 30);
jFieldMode.setForeground(Color.RED);
p.add(jFieldMode);

/*
final JTextArea jarea = new JTextArea("请输入方程组的A", 200, 200);
jarea.setBounds(20, 90, 200, 200);
p.add(jarea);

```

```

*/

JLabel label8 = new JLabel("步长设置:");
label8.setBounds(500, 20, 200, 20);
label8.setForeground(Color.BLUE);
p.add(label8);

jFieldResult2.setText("0.1");
jFieldResult2.setEditable(true);
jFieldResult2.setBounds(580, 18, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult2);

JLabel label11 = new JLabel("上下限设置: ");
label11.setBounds(500, 60, 200, 20);
label11.setForeground(Color.BLUE);
p.add(label11);

jFieldResult3.setText("0 1");
jFieldResult3.setEditable(true);
jFieldResult3.setBounds(580, 58, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult3);

JLabel label12 = new JLabel("初始值: ");
label12.setBounds(500, 100, 200, 20);
label12.setForeground(Color.BLUE);
p.add(label12);

jFieldResult4.setText("0");
jFieldResult4.setEditable(true);
jFieldResult4.setBounds(580, 98, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult4);

labelImg = new JLabel();
labelImg.setBounds(20, 60, 300, 180);
labelImg.setIcon(new ImageIcon("img/fun1.png"));
p.add(labelImg);

/**
 * 这里是对Buttons的设置
 */
JButton button1 = new JButton("开始计算");//
button1.setBounds(20, 320, 200, 40);// 设置按钮在容器中的位置
p.add(button1);

button1.addActionListener(new ActionListener()// 对按钮增加监听
{
    // 此处需要使用的是匿名类，需要重写actionPerformed函数，否则会出错
    @Override
    public void actionPerformed(ActionEvent e) {
        //处理输入步长和上下限
        String []ab = jFieldResult3.getText().split(" ");
        low = Double.parseDouble(ab[0]);
        high = Double.parseDouble(ab[1]);
        step = Double.parseDouble(jFieldResult2.getText());
    }
}

```



```

        yStart = Double.parseDouble(jFieldResult4.getText());
        fun.setData(low, high, step, yStart, flag);
        //System.out.println(fun.calEuler());
        if (SolveTypeInt == 1)
            jFieldResult.setText(fun.calEuler());
        if (SolveTypeInt == 2)
            jFieldResult.setText(fun.calEulerImproved());
        if (SolveTypeInt == 3)
            jFieldResult.setText(fun.calRK());

    }
});

/**
 * 这里是函数结尾的必要设置
 */

frame.getContentPane().add(p2);
frame.getContentPane().add(p);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 界面结束后关闭程序
frame.setLocationRelativeTo(null); // 在屏幕上居中显示框架
frame.setVisible(true); // 界面可视化，需要放在最后面，对所有的组件进行渲染。
}

```

显示效果如下：



0

$$\begin{cases} y' = x^2 - y^2 \\ y(-1) = 0 \end{cases}$$

当前结果: 未显示



1

$$\begin{cases} y' = \frac{4x}{y} - xy \\ y(0) = 0 \end{cases}$$

1.00 0.90 0.82 0.757 0.708 0.674 0.654 0.647 0.654 0.675 0.711

3.5 初始化菜单栏

一、菜单条 (JMenuBar)

JMenuBar 的构造方法是 JMenuBar(), 相当简单。在构造之后, 还要将它设置成窗口的菜单条, 这里要用 setJMenuBar 方法:

```
JMenuBar TestJMenuBar=new JMenuBar();
TestFrame.setJMenuBar(TestJMenuBar);
```

需要说明的是, JMenuBar 类根据 JMenu 添加的顺序从左到右显示, 并建立整数索引。

二、菜单 (JMenu)

在添加完菜单条后, 并不会显示任何菜单, 所以还需要在菜单条中添加菜单。菜单 JMenu 类的构造方法有4种:

JMenu() 构造一个空菜单。JMenu(Action a) 构造一个菜单, 菜单属性由相应的动作来提供。

JMenu(String s) 用给定的标志构造一个菜单。JMenu(String s, Boolean b) 用给定的标志构造一个菜单。如果布尔值为false, 那么当释放鼠标按钮后, 菜单项会消失; 如果布尔值为true, 那么当释放鼠标按钮后, 菜单项仍将显示。这时的菜单称为 tearOff 菜单。

在构造完后, 使用 JMenuBar 类的 add 方法添加到菜单条中。

三、菜单项 (JMenuItem)

接下来的工作是往菜单中添加内容。在菜单中可以添加不同的内容, 可以是菜单项

(JMenuItem), 可以是一个子菜单, 也可以是分隔符。

在构造完后, 使用 JMenu 类的 add 方法添加到菜单中。

子菜单的添加是直接将一个子菜单添加到母菜单中, 而分隔符的添加只需要将分隔符作为菜单项添加到菜单中。

JMenuBar要set,JMenu要add, JMenu在new的时候直接指定名字。

这里初始化了JMenu, JMenuItem, JMenuBar。

实例化了JMenuItem如下:

```
public void initMenuBar() {
    JMenu Menu1,Menu2;
    JMenuItem funItem1, funItem2,funItem3;
    JMenuItem funItem4, funItem5,funItem6;
    JMenuBar menuBar = new JMenuBar();
    funItem1 = new JMenuItem("Euler法");
    funItem2 = new JMenuItem("改进Euler法");
    funItem3 = new JMenuItem("R-K法");
    funItem4 = new JMenuItem("样例1");
    funItem5 = new JMenuItem("样例2");
    //funItem6 = new JMenuItem("样例3");
    Menu1 = new JMenu("求解方法选择");
    Menu2 = new JMenu("求解方程组选择");
    Menu1.add(funItem1);
    Menu1.add(funItem2);
    Menu1.add(funItem3);
    Menu2.add(funItem4);
    Menu2.add(funItem5);
    //Menu2.add(funItem6);
    Menu1.setSelected(true);
    Menu2.setSelected(true);
    menuBar.add(Menu1);
    menuBar.add(Menu2);
    frame.setJMenuBar(menuBar);
}
```

最后需要对每一个JMenuItem增加一个监听，实现选中后内部的逻辑变化。

以下为initMenuBar()函数源码：

```
public void initMenuBar() {
    JMenu Menu1,Menu2;
    JMenuItem funItem1, funItem2,funItem3;
    JMenuItem funItem4, funItem5,funItem6;
    JMenuBar menuBar = new JMenuBar();
    funItem1 = new JMenuItem("Euler法");
    funItem2 = new JMenuItem("改进Euler法");
    funItem3 = new JMenuItem("R-K法");
    funItem4 = new JMenuItem("样例1");
    funItem5 = new JMenuItem("样例2");
    //funItem6 = new JMenuItem("样例3");
    Menu1 = new JMenu("求解方法选择");
    Menu2 = new JMenu("求解方程组选择");
    Menu1.add(funItem1);
    Menu1.add(funItem2);
    Menu1.add(funItem3);
    Menu2.add(funItem4);
    Menu2.add(funItem5);
    //Menu2.add(funItem6);
    Menu1.setSelected(true);
    Menu2.setSelected(true);
    menuBar.add(Menu1);
    menuBar.add(Menu2);
    frame.setJMenuBar(menuBar);
    funItem1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("lag");
            updateModeStr(1);
            System.out.println("Euler法");
        }
    });
    funItem2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("newton");
            updateModeStr(2);
            System.out.println("改进Euler法");
        }
    });
    funItem3.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("newton");
            updateModeStr(3);
            System.out.println("R-K法");
        }
    });
    funItem4.addActionListener(new ActionListener() {
        @Override
```

```

        public void actionPerformed(ActionEvent e) {
            // updateModeStr("newton");
            //updateModeStr(3);
            labelImg.setIcon(new ImageIcon("img/fun1.png"));
            //System.out.println("R-K法");
            flag = 1;
        }
    });
    funItem5.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("newton");
            //updateModeStr(3);
            labelImg.setIcon(new ImageIcon("img/fun2.png"));
            flag = 2;
            //System.out.println("R-K法");
        }
    });
}

```

3.6 Fun类实现

类内函数主要有：

- `public void setData(double a[][], int dim,double u[],double e)`
- `public String[] cal()//幂法`

```

class Fun {
    private double a,b;//上下限`
    private double h,y0;//
    private double funNumber;//选择某个函数
    /**
     *
     * @param aa 方程下限
     * @param bb 方程上限
     * @param step 步长
     * @param ys 初始值
     * @param num 选择的方程的编号
     */
    public void setData(double aa, double bb,double step,double ys,int num)
    {
        a = aa;
        b = bb;
        h = step;
        y0 = ys;
        funNumber = num;
    }
    public double fun(double x,double y)
    {
        if (funNumber == 1)
            return 4 * x / y - x * y;
        if (funNumber == 2)
            return x * x - y * y;
        if (funNumber == 3)
            return y * y * Math.cos(x);
        return 0;
    }
}

```

```

}
public String calEuler()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double tmp = y; //y -> yn
        y += h * fun(x, tmp); //y -> yn+1
        x += h;
    }
    res += String.valueOf(nf.format(y));
    return res;
}

public String calEulerImproved()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double tmp = y; //y = tmp -> yn
        double tmpx = x + h; //tmpx -> xn+1
        y += h / 2 * (fun(x, tmp) + fun(x + h, tmp + h * fun(x, tmp))); //y-
>yn+1
        x += h;
    }
    res += String.valueOf(nf.format(y));
    return res;
}

public String calRK()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double k1, k2, k3, k4;
        k1 = fun(x, y);
        k2 = fun(x + 0.5 * h, y + 0.5 * h * k1);
        k3 = fun(x + 0.5 * h, y + 0.5 * h * k2);
        k4 = fun(x + h, y + h * k3);
    }
}

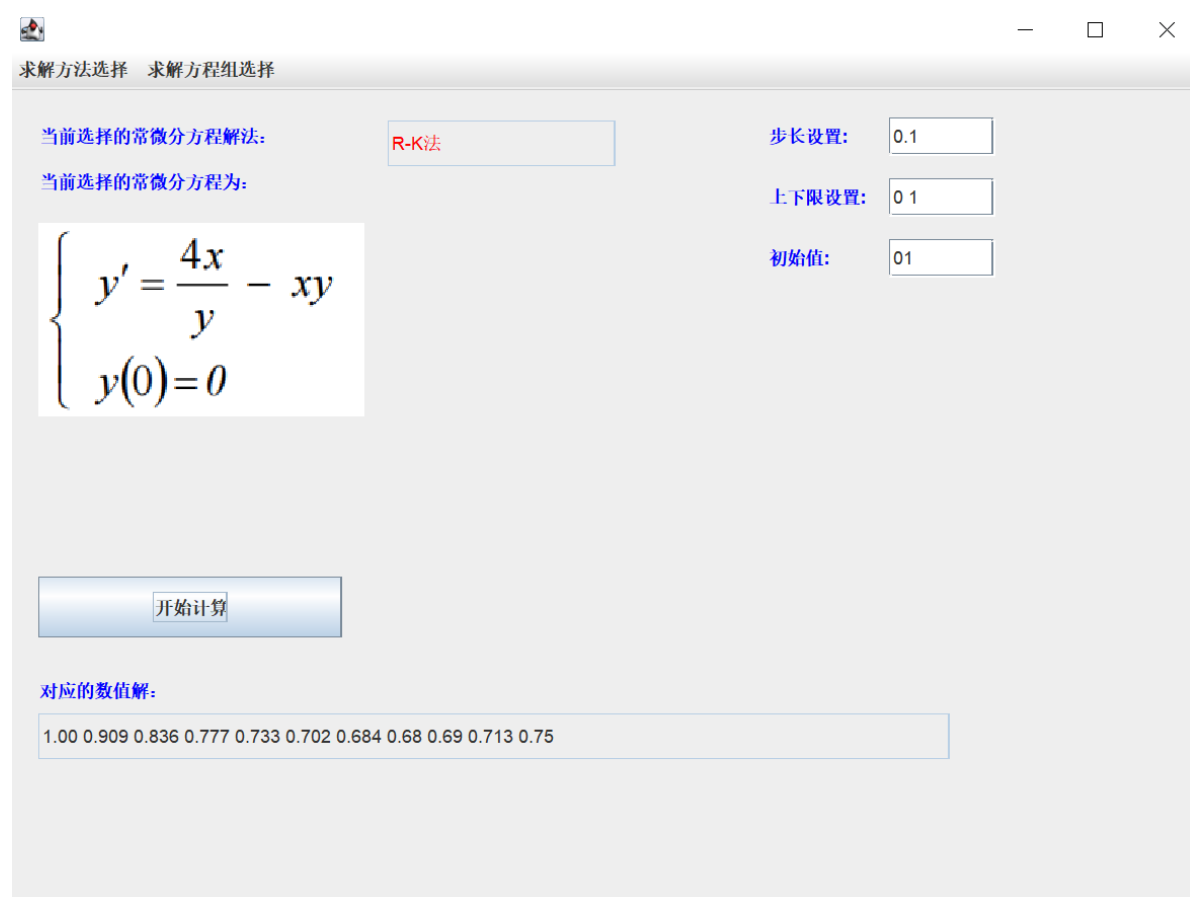
```

```

        y += h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);
        x += h;
    }
    res += String.valueOf(nf.format(y));
    return res;
}
}

```

4 实验结果总结



求解方法选择 求解方程组选择

当前选择的常微分方程解法: R-K法

当前选择的常微分方程为:

$$\begin{cases} y' = \frac{4x}{y} - xy \\ y(0) = 0 \end{cases}$$

步长设置: 0.1

上下限设置: 0 1

初始值: 01

开始计算

对应的数值解:

1.00 0.909 0.836 0.777 0.733 0.702 0.684 0.68 0.69 0.713 0.75

本次实验仅实现了常微分方程的数值解的求解。

通过改变步长等方式可以发现，欧拉方法简单的取切线的端点作为下一步的起点进行计算，当步数增多的时候，误差会因为积累而越来越大。因此欧拉公式一般不用于实际的计算。

而改进的欧拉公式，则在欧拉公式上继续改进，采取区间两端的斜率的平均值作为直线方程的斜率，提高了精度。

RK方法则是欧拉法的一种推广。

使用不同的方法计算某初值问题，其结果可能有微小偏差。

5 附录

源码

```

import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

```

```

import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.*;

import java.awt.Graphics;

public class SolveODE {

    static String strb;
    static ArrayList<String> strA = new ArrayList<String>();

    static JLabel labelImg = new JLabel();//载入函数图片的jlabel
    static double step;//方程步长
    static double low,high;//方程上下限
    static double yStart;//初始值
    static Fun fun = new Fun();
    static String SolveType = new String("NULL");
    // 默认的初始模式是空模式
    static String ModeType = new String("NULL");
    //
    static int SolveTypeInt = 0;// 1 2 3 4
    // 表示选择的函数 有1, 2, 3, 4四个函数
    static Graphics g;
    static int flag = 0;//函数的选择
    // 暂时不会用到画图
    static JFrame frame = new JFrame();
    // 定义了一个Frame
    static String result = new String("");
    /*
     * 下面对JTextField进行static的初始化定义，方便下面直接对其修改
     */
    static JTextField jFieldMode = new JTextField(120);// 模式选择
    static JTextField jFieldResult = new JTextField(120);
    static JTextField jFieldResultLam = new JTextField(120);//设置特征值
    static JTextField jFieldResult2 = new JTextField(120);//设置步长
    static JTextField jFieldResult3 = new JTextField(120);//设置上下限
    static JTextField jFieldResult4 = new JTextField(120);//设置初始值
    public static void main(String[] args) {
        System.out.println("Test Success!");
        SolveODE NI = new SolveODE();

        NI.initMenuBar();// 初始化菜单栏
        NI.initUI();// 初始化UI界面
    }
    /**
     * 处理A的输入
     * @param sa
     */
    public void processInput(String sa) {
        String[] tmpa = sa.split("\n|\\s+");
        int len = tmpa.length;
        for (int i = 0; i < tmpa.length; i++) {
            strA.add(tmpa[i]);
        }
    }
}

```



```

        //System.out.println(strA);

        //System.out.println(dim);
    }

    public void initUI() {

        /**
         * 这里是对frame的设置
         */
        frame.setSize(800, 600); // 设置容器尺寸
        frame.setLayout(new BorderLayout());
        /**
         * 中间容器
         */
        JPanel p2 = new JPanel() {

            public void paint(Graphics g) {
                super.paint(g);
                g.drawLine(350, 100, 500, 400);
            }
        };
        JPanel p = new JPanel();
        p.setLayout(null);
        p.setOpaque(false);

        /**
         * 这里是对labels的设置
         */
        JLabel label = new JLabel("当前选择的常微分方程为: ");
        label.setBounds(20, 50, 200, 20);
        label.setForeground(Color.BLUE);
        p.add(label);

        JLabel label1 = new JLabel("当前选择的常微分方程解法: ");
        label1.setBounds(20, 20, 200, 20);
        label1.setForeground(Color.BLUE);
        p.add(label1);

        // JLabel label6 = new JLabel("请输入需要求解的方程组的B: ");
        // label6.setBounds(20, 310, 200, 20);
        // label6.setForeground(Color.BLUE);
        // p.add(label6);

        JLabel label7 = new JLabel("对应的数值解: ");
        label7.setBounds(20, 385, 200, 20);
        label7.setForeground(Color.BLUE);
        p.add(label7);

        jTextFieldResult.setText("当前结果: 未显示");
        jTextFieldResult.setEditable(false);
        jTextFieldResult.setBounds(20, 410, 600, 30);
        jTextFieldMode.setForeground(Color.RED);
        p.add(jTextFieldResult);

        /**
         *
         */
        JLabel label9 = new JLabel("按模最大或最小的特征值: ");
        label9.setBounds(20, 445, 200, 20);
    }

```

```

label9.setForeground(Color.BLUE);
p.add(label9);
*/

/*
jFieldResultLam.setText("当前结果: 未显示");
jFieldResultLam.setEditable(false);
jFieldResultLam.setBounds(20, 470, 600, 30);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResultLam);
*/

jFieldMode.setText("当前求解方法: 未选择");
jFieldMode.setEditable(false);
jFieldMode.setBounds(250, 20, 150, 30);
jFieldMode.setForeground(Color.RED);
p.add(jFieldMode);

/*
final JTextArea jarea = new JTextArea("请输入方程组的A", 200, 200);
jarea.setBounds(20, 90, 200, 200);
p.add(jarea);
*/

JLabel label8 = new JLabel("步长设置:");
label8.setBounds(500, 20, 200, 20);
label8.setForeground(Color.BLUE);
p.add(label8);

jFieldResult2.setText("0.1");
jFieldResult2.setEditable(true);
jFieldResult2.setBounds(580, 18, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult2);

JLabel label11 = new JLabel("上下限设置: ");
label11.setBounds(500, 60, 200, 20);
label11.setForeground(Color.BLUE);
p.add(label11);

jFieldResult3.setText("0 1");
jFieldResult3.setEditable(true);
jFieldResult3.setBounds(580, 58, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult3);

JLabel label12 = new JLabel("初始值: ");
label12.setBounds(500, 100, 200, 20);
label12.setForeground(Color.BLUE);
p.add(label12);

jFieldResult4.setText("1");
jFieldResult4.setEditable(true);
jFieldResult4.setBounds(580, 98, 70, 25);
//jFieldMode.setForeground(Color.RED);
p.add(jFieldResult4);

labelImg = new JLabel();

```

```

labelImg.setBounds(20, 60, 300, 180);
labelImg.setIcon(new ImageIcon("img/fun1.png"));
p.add(labelImg);

/**
 * 这里是对Buttons的设置
 */
JButton button1 = new JButton("开始计算");//
button1.setBounds(20, 320, 200, 40);// 设置按钮在容器中的位置
p.add(button1);

button1.addActionListener(new ActionListener()// 对按钮增加监听
{
    // 此处需要使用的是匿名类，需要重写actionPerformed函数，否则会出错
    @Override
    public void actionPerformed(ActionEvent e) {
        //处理输入步长和上下限
        String []ab = jTextFieldResult3.getText().split(" ");
        low = Double.parseDouble(ab[0]);
        high = Double.parseDouble(ab[1]);
        step = Double.parseDouble(jTextFieldResult2.getText());
        yStart = Double.parseDouble(jTextFieldResult4.getText());
        fun.setData(low, high, step, yStart, flag);
        //System.out.println(fun.calEuler());
        if (SolveTypeInt == 1)
            jTextFieldResult.setText(fun.calEuler());
        if (SolveTypeInt == 2)
            jTextFieldResult.setText(fun.calEulerImproved());
        if (SolveTypeInt == 3)
            jTextFieldResult.setText(fun.calRK());

    }
});

/**
 * 这里是函数结尾的必要设置
 */

frame.getContentPane().add(p2);
frame.getContentPane().add(p);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);// 界面结束后关闭程序
frame.setLocationRelativeTo(null);// 在屏幕上居中显示框架
frame.setVisible(true);// 界面可视化，需要放在最后面，对所有的组件进行渲染。
}

public void initMenuBar() {
    JMenu Menu1, Menu2;
    JMenuItem funItem1, funItem2, funItem3;
    JMenuItem funItem4, funItem5, funItem6;
    JMenuBar menuBar = new JMenuBar();
    funItem1 = new JMenuItem("Euler法");
    funItem2 = new JMenuItem("改进Euler法");
    funItem3 = new JMenuItem("R-K法");
    funItem4 = new JMenuItem("样例1");
    funItem5 = new JMenuItem("样例2");

```

```

//funItem6 = new JMenuItem("样例3");
Menu1 = new JMenu("求解方法选择");
Menu2 = new JMenu("求解方程组选择");
Menu1.add(funItem1);
Menu1.add(funItem2);
Menu1.add(funItem3);
Menu2.add(funItem4);
Menu2.add(funItem5);
//Menu2.add(funItem6);
Menu1.setSelected(true);
Menu2.setSelected(true);
menuBar.add(Menu1);
menuBar.add(Menu2);
frame.setJMenuBar(menuBar);
funItem1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // updateModeStr("lag");
        updateModeStr(1);
        System.out.println("Euler法");
    }
});
funItem2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // updateModeStr("newton");
        updateModeStr(2);
        System.out.println("改进Euler法");
    }
});
funItem3.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // updateModeStr("newton");
        updateModeStr(3);
        System.out.println("R-K法");
    }
});
funItem4.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // updateModeStr("newton");
        //updateModeStr(3);
        labelImg.setIcon(new ImageIcon("img/fun1.png"));
        //System.out.println("R-K法");
        flag = 1;
    }
});
funItem5.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // updateModeStr("newton");
        //updateModeStr(3);
        labelImg.setIcon(new ImageIcon("img/fun2.png"));
    }
});

```

```

        flag = 2;
        //System.out.println("R-K法");
    }
});
}

/**
 * 更新函数的select选中和积分方法select选中的UI
 *
 * @param num
 */
public void updateModeStr(int num)// mode表示模式的意思，即插值的类型
{
    if (num == 1) {
        solveType = new String("Euler法");
        solveTypeInt = 1;
        jFieldMode.setText(solveType);
    } else if (num == 2) {
        solveType = new String("改进Euler法");
        solveTypeInt = 2;
        jFieldMode.setText(solveType);
    } else if (num == 3) {
        solveType = new String("R-K法");
        solveTypeInt = 3;
        jFieldMode.setText(solveType);
    }
}

}

class Fun {
    private double a,b;//上下限`
    private double h,y0;//
    private double funNumber;//选择某个函数
    /**
     *
     * @param aa 方程下限
     * @param bb 方程上限
     * @param step 步长
     * @param ys 初始值
     * @param num 选择的方程的编号
     */
    public void setData(double aa, double bb,double step,double ys,int num)
    {
        a = aa;
        b = bb;
        h = step;
        y0 = ys;
        funNumber = num;
    }
    public double fun(double x,double y)
    {
        if (funNumber == 1)
            return 4 * x / y - x * y;
        if (funNumber == 2)
            return x * x - y * y;
        if (funNumber == 3)
            return y * y * Math.cos(x);
        return 0;
    }
}

```

```

}
public String calEuler()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double tmp = y; //y -> yn
        y += h * fun(x, tmp); //y -> yn+1
        x += h;
    }
    res += String.valueOf(nf.format(y));
    return res;
}

public String calEulerImproved()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double tmp = y; //y = tmp -> yn
        double tmpx = x + h; //tmpx -> xn+1
        y += h / 2 * (fun(x, tmp) + fun(x + h, tmp + h * fun(x, tmp))); //y-
>yn+1
        x += h;
    }
    res += String.valueOf(nf.format(y));
    return res;
}

public String calRK()
{
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMinimumFractionDigits(2);
    double x = a;
    double y = y0;
    String res = new String("");
    while (x + h < b)
    {
        res += String.valueOf(nf.format(y));
        //System.out.println(y);
        res += " ";
        double k1, k2, k3, k4;
        k1 = fun(x, y);
        k2 = fun(x + 0.5 * h, y + 0.5 * h * k1);
        k3 = fun(x + 0.5 * h, y + 0.5 * h * k2);
        k4 = fun(x + h, y + h * k3);
    }
}

```

```
        y += h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);  
        x += h;  
    }  
    res += String.valueOf(nf.format(y));  
    return res;  
}  
}
```