华东师范大学计算机科学技术系实验报告

实验课程： 数值计算	年级： **2019**	实验成绩：
实验名称： 函数插值方法	姓名： 林子炫	
实验编号： 1	学号： 10195102468	实验日期： 2021-09-19
指导教师： 谢董奎	组号：	实验时间： 9：00AM

1 实验目的

1. 学会常用的插值方法，求函数的近似表达式，以解决其它实际问题。
2. 明确插值多项式和分段插值多项式各自的优缺点。
3. 熟悉插值方法的程序编制。
4. 如果绘出插值函数的曲线，观察其光滑性。

2 实验环境

win10 + java

3 实验过程与分析

3.1 框架搭建

我们需要在图形面板中输入三个向量，分别是sx,sy,sw，三个字符串分别代表x向量，y向量，和所要求的插值点的向量表达式。

下面定义了三个插值类。

```
/**
 * 拉格朗日插值类
 */
class Lagrange{}
/**
 * 牛顿插值类
 */
class NewTon{}
/**
 * 分段Lagrange插值类
 */
class SegLag{}
```

在下面的框架中，`public class Interpolation`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 processInput(String sx,String sy)
    {
        //处理输入的两个x向量和y向量，将其转化成double类型的数组
    }
    public void processInput(String sw)
    {
        //处理输入的两个w，将其转化成double类型的数组
    }
    /**
     * 更新结果的数值。可以被重写，需要被重写
     */
    public void updateUI(Lagrange l)
    {
        //更新JFieldText的数值，来显示输出的结果
    }
    public void updateUI(NewTon nton)
    {
        //重写函数
    }
    public void initUI()
    {
        //初始化UI界面
    }
    public void initMenuBar()
    {
        //初始化菜单栏
    }
```

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 Interpolation{
    //三个字符串分别代表x向量，y向量，和所要求的插值点的向量表达式
    //
    static String sx = new String();
    static String sy = new String();
    static String sw = new String();
    //初始化的三个插值类
    static Lagrange lag = new Lagrange();
    static NewTon nTon = new NewTon();
    static SegLag sag = new SegLag();
    //1代表拉格朗日插值，2代表牛顿插值，3代表分段三次插值
    static String ModeType = new String("NULL");//默认的初始模式是空模
式
    static int ModeTypeInt = 0;//默认的初始模式代码是0
    //三个double类型的数组分别代表x向量，y向量，和所要求的插值点的向量表达式
    static double [] doublex;
    static double [] doubley;
    static double [] doublew;
    //
    static Graphics g;
    //定义了一个JFrame
    static JFrame frame = new JFrame();
    //定义了一个结果String，用来存输出的结果向量。
    static String result = new String("");
    //定义了一个JLabel，用来在Frame上显示输出的结果。
    static JLabel Jresult;
    //定义了一个JTextField 用来显示当前的模式。
    static JTextField jFieldMode = new JTextField(80);//模式选择
    //定义了一个JTextField 用来显示输出的结果。
    static JTextField jFieldResult = new JTextField(80);
    public static void main(String[] args) {
        System.out.println("Test Success!");
        Interpolation pola = new Interpolation();

        pola.initMenuBar();//初始化菜单栏
        pola.initUI();//初始化UI界面
    }
    /**
     * 处理输入的函数
     * @param sx
     * @param sy
     */
    public void processInput(String sx,String sy)
    {
    }
    public void processInput(String sw)
    {
    }
    /**
     * 更新结果的数值。可以被重写，需要被重写
     */
    public void updateUI(Lagrange l)
    {
    }
    public void updateUI(NewTon nton)
    {
    }
    public void initUI()
    {
    }
```

```
    public void initMenuBar()
    {
    }
}
/**
 * 拉格朗日插值类
 */
class Lagrange
{
}
/**
 * 牛顿插值类
 */
class NewTon
{
}
/**
 * 分段Lagrange插值类
 */
class SegLag
{
}
/**
 * 画图类
 */
class Draw
{
}
```

3.2 实现输入输出

`public class Interpolation`为主Public类内定义了许多静态变量，sx，sy来存储读入的数据为String类型，doublex和doubley将读入的String类型的数据分割为double数组类型，便于进行处理。

其中，这里实现输入和数据读取的方式是使用ProcessInput函数来实现，`for (int i = 0; i < strx.length;i++) doublex[i] = Double.parseDouble(strx[i]);`

```
    public void processInput(String sx,String sy)
    {
        String [] strx = sx.split(" ");
        String [] stry = sy.split(" ");
        doublex = new double[strx.length];
        doubley = new double[stry.length];
        for (int i = 0; i < strx.length;i++)
            doublex[i] = Double.parseDouble(strx[i]);
        for (int i = 0;i < stry.length;i++)
            doubley[i] = Double.parseDouble(stry[i]);
    }
    public void processInput(String sw)
    {
        String [] strw = sw.split(" ");
        doublew = new double[strw.length + 1];
        for (int i = 0;i < strw.length;i++)
            doublew[i] = Double.parseDouble(strw[i]);
```

3.3 更新 U I

插值的结果显示在JFieldText类型的jFieldResult变量中，每一次计算的时候都需要对这个变量进行更新，所以简称为更新UI。其中最主要的是应用了`jFieldResult.setText()`来实现的。

三个函数同样，只不过对其进行了重载。

```
public void updateUI(Lagrange l)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(l.calculate(doublew[i]) + "
");
        System.out.println(result);
        jFieldResult.setText(result);
    }
public void updateUI(NewTon nton)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(nton.calculate(doublew[i]) +
" ");
        System.out.println(result);
        jFieldResult.setText(result);
    }
public void updateUI(SegLag segg)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(segg.calculate(doublew[i]) +
" ");
        System.out.println(result);
        jFieldResult.setText(result);
    }
```

3.4 初始化 U I

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

JTextField 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

JPasswordField 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

JCheckBox 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

JButton 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("插值计算");
label.setBounds(20, 20, 100, 20);
label.setForeground(Color.BLUE);
p.add(label);
JLabel label2 = new JLabel("请输入X向量");
label2.setBounds(20, 100, 100, 20);
p.add(label2);
JLabel label3 = new JLabel("请输入Y向量");
label3.setBounds(20, 140, 100, 20);
p.add(label3);
JLabel label4 = new JLabel("请输入Z向量");
label4.setBounds(20, 180, 100, 20);
```

```
p.add(label4);
JLabel label5 = new JLabel("结果向量: ");
label5.setBounds(20, 220, 100, 20);
p.add(label5);
```

随后添加开始计算按钮。

```
JButton button1 = new JButton("开始计算");//
button1.setBounds(100, 270, 200, 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。

此处的`modeTypeInt`表示为当前的插值模式类型，当鼠标点击按钮时，获取sx，sy，sw字符串的值，通过`processInput(sw)和processInput(sx, sy)`函数进行处理，并调用`updateUI`函数对结果进行更新。

```
    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) {
                sx = jFieldX.getText();
                sy = jFieldY.getText();
                sw = jFieldW.getText();
                processInput(sw);
                processInput(sx, sy);

                if (ModeTypeInt == 1)
                {
                    lag.setData(doublex, doubley);
                    System.out.println("click2");
                    updateUI(lag);
                }
                if (ModeTypeInt == 2)
                {
                    nTon.setData(doublex, doubley);
                    updateUI(nTon);
                }
                if (ModeTypeInt == 3)
                {
                    System.out.println(123);
                    //updateUI(Seg);
                }
            }
        });
```

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("插值计算");
        label.setBounds(20, 20, 100, 20);
        label.setForeground(Color.BLUE);
        p.add(label);
        JLabel label2 = new JLabel("请输入X向量");
        label2.setBounds(20, 100, 100, 20);
        p.add(label2);
        JLabel label3 = new JLabel("请输入Y向量");
        label3.setBounds(20, 140, 100, 20);
        p.add(label3);
        JLabel label4 = new JLabel("请输入Z向量");
        label4.setBounds(20, 180, 100, 20);
        p.add(label4);
        JLabel label5 = new JLabel("结果向量：");
        label5.setBounds(20, 220, 100, 20);
        p.add(label5);

        //frame.add(label);
        /**
         * JTextField的设置
         * 创建文本框，指定可见列数为80列
```

```
         */
        jFieldMode.setText("当前模式：未选择");
        jFieldMode.setEditable(false);
        jFieldMode.setBounds(100, 20, 120, 30);
        jFieldMode.setForeground(Color.RED);
        p.add(jFieldMode);
        //frame.add(jFieldMode);
        final JTextField jFieldX = new JTextField(80);
        jFieldX.setBounds(100, 100, 200, 30);
        p.add(jFieldX);
        //frame.add(jFieldX);
        final JTextField jFieldY = new JTextField(80);
        jFieldY.setBounds(100, 140, 200, 30);
        p.add(jFieldY);
        //frame.add(jFieldY);
        final JTextField jFieldW = new JTextField(80);
        jFieldW.setBounds(100, 180, 200, 30);
        p.add(jFieldW);
        //frame.add(jFieldW);
        jFieldResult = new JTextField(80);
        jFieldResult.setEditable(false);
        jFieldResult.setBounds(100, 220, 200, 30);
        p.add(jFieldResult);
        //frame.add(jFieldResult);

        /*
        p.add(jFieldMode);
        p.add(jFieldX);
        p.add(jFieldY);
        p.add(jFieldW);
        p.add(jFieldResult);
        */



        /**
         * 这里是对Buttons的设置
         */
        JButton button1 = new JButton("开始计算");//
        button1.setBounds(100, 270, 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) {
                sx = jFieldX.getText();
                sy = jFieldY.getText();
                sw = jFieldW.getText();
                processInput(sw);
                processInput(sx, sy);

                if (ModeTypeInt == 1)
                {
                    lag.setData(doublex, doubley);
                    System.out.println("click2");
                    updateUI(lag);
                }
                if (ModeTypeInt == 2)
                {
                    nTon.setData(doublex, doubley);
                    updateUI(nTon);
                }
```

```
                if (ModeTypeInt == 3)
                {
                    seg.setData(doublex, doubley);
                    updateUI(seg);
                }
            }
        });

        /**
         * 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：

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，分别设置了三个JMenuItem为newt，lag，srg。

```
public void initMenuBar()
{
    JMenu Menu;
    JMenuItem lag,newt,seg;
    JMenuBar menuBar = new JMenuBar();

    lag = new JMenuItem("拉格朗日插值");
    newt = new JMenuItem("牛顿插值");
    seg = new JMenuItem("分段插值");
    Menu = new JMenu("插值类型");
    Menu.add(lag);
    Menu.add(newt);
    Menu.add(seg);
    Menu.setSelected(true);
    menuBar.add(Menu);
    frame.setJMenuBar(menuBar);
    lag.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            updateModeStr("lag");
            System.out.println("当前模式：拉格朗日");
        }
    });
    newt.addActionListener(new ActionListener(){
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                updateModeStr("newton");
                System.out.println("当前模式：牛顿");
            }
        });
        seg.addActionListener(new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e) {
                updateModeStr("seg");
                System.out.println("当前模式:分段");
            }
        });
    }
```

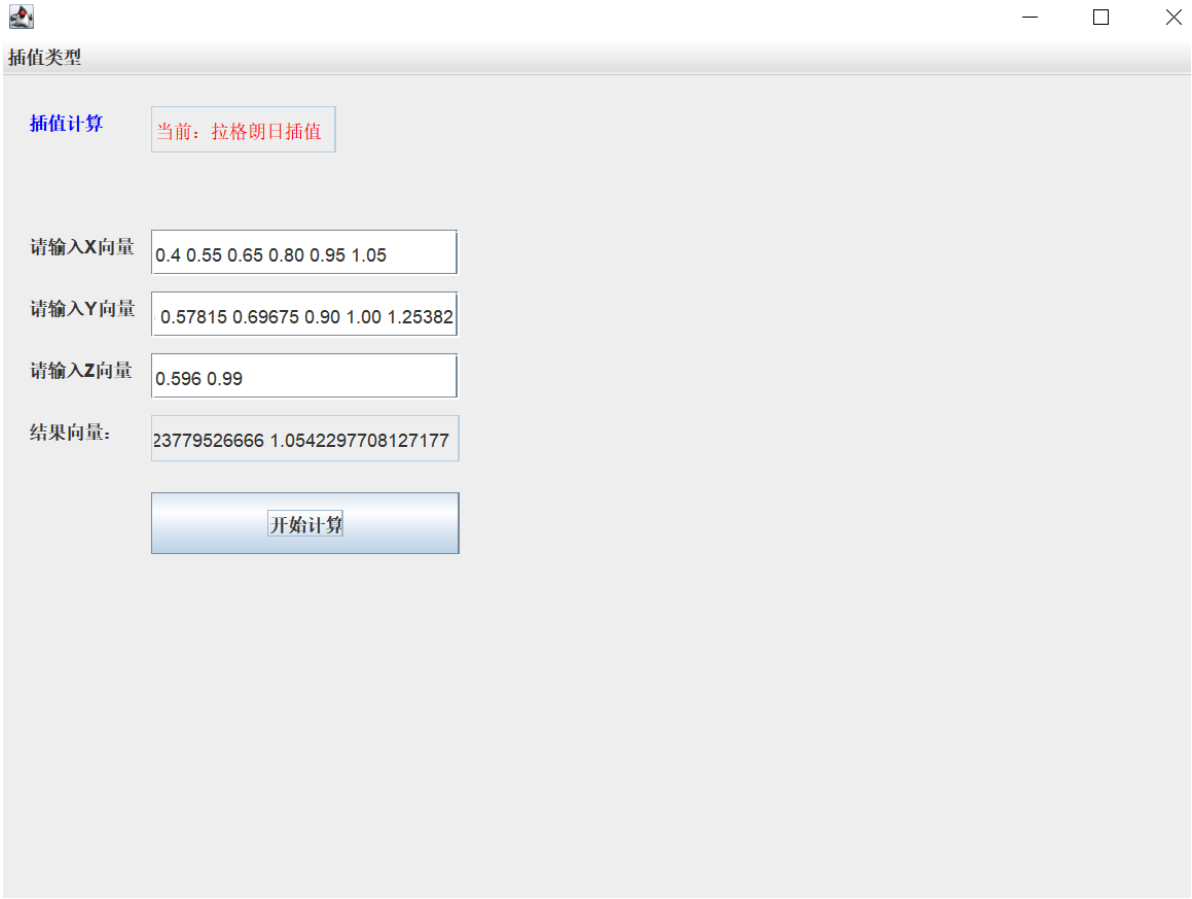
3.6 插值函数类的实现

以拉格朗日插值为例子，类内函数calculate为计算结果函数，setData为设置函数值的函数。

```
class Lagrange
{
    public int n;//几次插值
    public double x[];
    public double y[];
    //x[],y[]代表的是所有(x,y)的已知点
    public double calculate(double xx)
    {
        double result = 0;
        for (int i = 0 ;i <= n;i++)
        {
            double tmp = 1;
            for (int j = 0;j <= n;j++)
                if (j != i) tmp *= (xx - x[j]);
            for (int j = 0;j <= n;j++)
                if (j != i) tmp /= (x[i] - x[j]);
            tmp *= y[i];
            result +=  tmp;
        }
        return result;
    }
    public void setData(double xq[],double yq[])
    {
        n = xq.length - 1;
        x = new double[n + 1];
        y = new double[n + 1];
        for (int i = 0 ; i < xq.length;i++)
            x[i] = xq[i];
        for (int i = 0 ; i < yq.length;i++)
            y[i] = yq[i];
    }
}
```

4 实验结果总结

在使用时，先选择插值的类型，随后输入x，y，z向量，其中z向量表示的是所要求的插值点的函数值，输入完之后点击开始计算按钮即可得出结果。



第一次实验中遇到的困难在于java GUI的使用，其中关于JPanel和JFrame的使用有很多小坑。

比如Jframe因为时顶层容器，例子中生成了一个空的窗体，在实际编程过程中，一般很少将文本框、按钮等组件直接放在顶层容器中进行布局，大多数时候是通过布局管理器结合中间容器对组件进行布局设置。所以将组件添加到JFrame的时候用getContentPane()方法获得JFrame的内容面板，再对其加入组件，一般只使用该方式添加组件。否则直接使用add方法会产生错误。

在使用页面布局的时候，使用流式布局，或者方块布局等定义好的布局容易出错，所以这里使用的时自定义布局，使用坐标进行组件定位会更方便。

5 附录

源码

```
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 Interpolation{
    static String sx = new String();
    static String sy = new String();
    static String sw = new String();
    static Lagrange lag = new Lagrange();
    static NewTon nTon = new NewTon();
    static SegLag seg = new SegLag();
    static String ModeType = new String("NULL");//默认的初始模式是空模
式
    static int ModeTypeInt = 0;
    static double [] doublex;
    static double [] doubley;
    static double [] doublew;
    static Graphics g;
    static JFrame frame = new JFrame();
    static String result = new String("");
    static JLabel Jresult;
    static JTextField jFieldMode = new JTextField(80);//模式选择
    static JTextField jFieldResult = new JTextField(80);
    public void paint(Graphics g)
    {
        g.setColor(Color.blue);
        g.drawLine(100, 100, 300, 300);
    }
    public static void main(String[] args) {
        System.out.println("Test Success!");
        Interpolation pola = new Interpolation();

        pola.initMenuBar();//初始化菜单栏
        pola.initUI();//初始化UI界面
    }
    /**
     * 处理输入的函数
     * @param sx
     * @param sy
     */
    public void processInput(String sx,String sy)
    {
        String [] strx = sx.split(" ");
        String [] stry = sy.split(" ");
        doublex = new double[strx.length];
        doubley = new double[stry.length];
        for (int i = 0; i < strx.length;i++)
            doublex[i] = Double.parseDouble(strx[i]);
        for (int i = 0;i < stry.length;i++)
            doubley[i] = Double.parseDouble(stry[i]);
    }
    public void processInput(String sw)
    {
        String [] strw = sw.split(" ");
        doublew = new double[strw.length];
        for (int i = 0;i < strw.length;i++)
            doublew[i] = Double.parseDouble(strw[i]);
    }
    /**
```

```
     * 更新结果的数值。可以被重写，需要被重写
     */
    public void updateUI(Lagrange l)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(l.calculate(doublew[i]) + "
");
        System.out.println(result);
        jFieldResult.setText(result);
    }
    public void updateUI(NewTon nton)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(nton.calculate(doublew[i]) +
" ");
        System.out.println(result);
        jFieldResult.setText(result);
    }
    public void updateUI(SegLag segg)
    {
        result = new String("");
        for (int i = 0 ; i < doublew.length;i++)
            result += String.valueOf(segg.calculate(doublew[i]) +
" ");
        System.out.println(result);
        jFieldResult.setText(result);
    }
    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());
        //frame.setLayout(null);//设置布局
        //frame.addPanel();

        /**
         * 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.setSize(300,300);
        //p.setPreferredSize(new Dimension(300,300));
        p.setLayout(null);
        p.setOpaque(false);
        //p.setSize(200,200);
        //p.setBackground(Color.BLUE);

        /**
         * 这里是对labels的设置
         */
        JLabel label = new JLabel("插值计算");
        label.setBounds(20, 20, 100, 20);
```

```
        label.setForeground(Color.BLUE);
        p.add(label);
        JLabel label2 = new JLabel("请输入X向量");
        label2.setBounds(20, 100, 100, 20);
        p.add(label2);
        JLabel label3 = new JLabel("请输入Y向量");
        label3.setBounds(20, 140, 100, 20);
        p.add(label3);
        JLabel label4 = new JLabel("请输入Z向量");
        label4.setBounds(20, 180, 100, 20);
        p.add(label4);
        JLabel label5 = new JLabel("结果向量: ");
        label5.setBounds(20, 220, 100, 20);
        p.add(label5);

        //frame.add(label);
        /**
         * JTextField的设置
         * 创建文本框，指定可见列数为80列
         */
        jFieldMode.setText("当前模式：未选择");
        jFieldMode.setEditable(false);
        jFieldMode.setBounds(100, 20, 120, 30);
        jFieldMode.setForeground(Color.RED);
        p.add(jFieldMode);
        //frame.add(jFieldMode);
        final JTextField jFieldX = new JTextField(80);
        jFieldX.setBounds(100, 100, 200, 30);
        p.add(jFieldX);
        //frame.add(jFieldX);
        final JTextField jFieldY = new JTextField(80);
        jFieldY.setBounds(100, 140, 200, 30);
        p.add(jFieldY);
        //frame.add(jFieldY);
        final JTextField jFieldW = new JTextField(80);
        jFieldW.setBounds(100, 180, 200, 30);
        p.add(jFieldW);
        //frame.add(jFieldW);
        jFieldResult = new JTextField(80);
        jFieldResult.setEditable(false);
        jFieldResult.setBounds(100, 220, 200, 30);
        p.add(jFieldResult);
        //frame.add(jFieldResult);

        /*
        p.add(jFieldMode);
        p.add(jFieldX);
        p.add(jFieldY);
        p.add(jFieldW);
        p.add(jFieldResult);
        */



        /**
         * 这里是对Buttons的设置
         */
        JButton button1 = new JButton("开始计算");//
        button1.setBounds(100, 270, 200, 40);//设置按钮在容器中的位置

        p.add(button1);
        //button.setBounds(int x,int y,int width,int height)
(x,y)代表的是左上角顶点的位置,width和height
        //的是按钮的宽度
        //button1.setFont(new Font(null, Font.PLAIN, 20));
```

```
        //panel2.add(button);//按钮加在容器上
        //frame.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) {
                sx = jFieldX.getText();
                sy = jFieldY.getText();
                sw = jFieldW.getText();
                processInput(sw);
                processInput(sx, sy);

                if (ModeTypeInt == 1)
                {
                    lag.setData(doublex, doubley);
                    System.out.println("click2");
                    updateUI(lag);
                }
                if (ModeTypeInt == 2)
                {
                    nTon.setData(doublex, doubley);
                    updateUI(nTon);
                }
                if (ModeTypeInt == 3)
                {
                    seg.setData(doublex,doubley);
                    updateUI(seg);
                }
            }
        });

        /**
         * 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 Menu;
        JMenuItem lag,newt,seg;
        JMenuBar menuBar = new JMenuBar();

        lag = new JMenuItem("拉格朗日插值");
        newt = new JMenuItem("牛顿插值");
        seg = new JMenuItem("分段插值");
        Menu = new JMenu("插值类型");
        Menu.add(lag);
        Menu.add(newt);
        Menu.add(seg);
        Menu.setSelected(true);
        menuBar.add(Menu);
        frame.setJMenuBar(menuBar);
        lag.addActionListener(new ActionListener(){
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                updateModeStr("lag");
                System.out.println("当前模式：拉格朗日");
            }
        });
        newt.addActionListener(new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e) {
                updateModeStr("newton");
                System.out.println("当前模式：牛顿");
            }
        });
        seg.addActionListener(new ActionListener(){

            @Override
            public void actionPerformed(ActionEvent e) {
                updateModeStr("seg");
                System.out.println("当前模式:分段");
            }
        });
    }
    public void updateModeStr(String M)//mode表示模式的意思，即插值的类
型
    {
        if (M == "lag"){
            ModeType = new String("当前：拉格朗日插值");
            ModeTypeInt = 1;
            jFieldMode.setText(ModeType);
        }
        else if (M == "newton")
        {
            ModeType = new String("当前：牛顿插值");
            ModeTypeInt = 2;
            jFieldMode.setText(ModeType);
        }
        else if (M == "seg")
        {
            ModeType = new String("当前：分段插值");
            ModeTypeInt = 3;
            jFieldMode.setText(ModeType);
        }

    }
}
/**
 * 拉格朗日插值类
 */
class Lagrange
{
    public int n;//几次插值
    public double x[];
    public double y[];
    //x[],y[]代表的是所有(x,y)的已知点
    public double calculate(double xx)
    {
        double result = 0;
        for (int i = 0 ;i <= n;i++)
        {
            double tmp = 1;
            for (int j = 0;j <= n;j++)
                if (j != i) tmp *= (xx - x[j]);
            for (int j = 0;j <= n;j++)
                if (j != i) tmp /= (x[i] - x[j]);
            tmp *= y[i];
```

```
            result +=  tmp;
        }
        return result;
    }
    public void setData(double xq[],double yq[])
    {
        n = xq.length - 1;
        x = new double[n + 1];
        y = new double[n + 1];
        for (int i = 0 ; i < xq.length;i++)
            x[i] = xq[i];
        for (int i = 0 ; i < yq.length;i++)
            y[i] = yq[i];
    }

}
/**
 * 牛顿插值类
 */
class NewTon
{
    public int n;//几次插值
    public double x[];
    public double y[];
    public void setData(double xq[],double yq[])
    {
        n = xq.length - 1;
        x = new double[n + 1];
        y = new double[n + 1];
        for (int i = 0 ; i < xq.length;i++)
            x[i] = xq[i];
        for (int i = 0 ; i < yq.length;i++)
            y[i] = yq[i];
    }
    public double calculate(double xx)
    {
        double []f = new double [n + 1];//差商表f[n] 表示
f[x,x0,x1,x2...,xn];
        f[0] = y[0];//对于差商f[x0] = f(x0)
        //f[0]表示f[x,x0] , f[1] = f[x,x0,x1] = doubley
        double []F = new double [n + 1];
        double result = 0;//存储结果
        //计算累乘积
        for (int i = 0;i <= n;i++)
        {
            double tmp = 1;
            for (int j = 0;j <= n;j++)
            {
                if(j != i)
                {
                    tmp *= (x[i] - x[j]);
                }
            }
            F[i] = tmp;
        }
        //计算n阶差商
        for (int i = 1 ; i <= n;i++)
        {
            double tmp = 0;
            for (int k = 0;k <= i;k++)
            {
                tmp += ((y[i])/(F[i]));
            }
            f[i] = tmp;
        }
```

```
        //计算牛顿插值多项式
        result += f[0];
        for (int i = 0 ; i < n;i++)
        {
            double tmp = f[i+1];
            for (int j = 0;j <= n;j++)
            {
                tmp *= (xx - x[0]);
            }
            result += tmp;
        }
        return result;
    }
}
/**
 * 分段Lagrange插值类
 */
class SegLag
{
    public int n;//几次插值
    public double x[];
    public double y[];
    public double m[];
    public double x0[];
    public void setData(double xq[],double yq[])
    {
        n = xq.length;
        x = new double[n + 1];
        y = new double[n + 1];
        m = new double[n + 1];
        for (int i = 0 ; i < xq.length;i++)
            x[i] = xq[i];
        for (int i = 0 ; i < yq.length;i++)
            y[i] = yq[i];
        for (int i = 0 ; i < yq.length;i++)
            m[i] = -yq[i];

    }
    public double calculate(double xx)
    {
        int a = n;

        double add_xj_xk=0;//连加
        double mul_x0_xk=1;//连乘
        double add_h2n1x=0;//近似值
        for (int j=0;j<a;j++) {//二重循环计算H2n+1(x)
            mul_x0_xk=1;
            add_xj_xk=0;
                for(int k=0;k<a;k++) {//三重循环计算l'j(xj)和lj(x)
                if (j!=k) {
                    add_xj_xk+=1/(x[j]-x[k]);
                    mul_x0_xk*=((xx-x[k])/(x[k]-x[j]));

                }
            }
        add_h2n1x+=(y[j]*(1-(2*(xx-x[j])*add_xj_xk))*
(mul_x0_xk*mul_x0_xk))+(m[j]*(xx-x[j])*(mul_x0_xk*mul_x0_xk));
    }
    return add_h2n1x;
}
}
```