

# 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：2	学号：10195102468	实验日期：2021-10-27
指导教师：谢堇奎	组号：	实验时间：9:00AM

## 1 实验目的

1. 编制数值积分算法的程序。
2. 分别用两种算法计算同一个积分，并比较其结果。
3. 分别取不同步长 $(/ ab h) \rightarrow n$ ，试比较计算结果（如 $n = 10, 20$ 等）。
4. 给定精度要求 $\epsilon$ ，试用变步长算法，确定最佳步长。

## 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两个向量，分别是ABE和N，A代表积分下限，B代表了积分上限，E代表了 `epsilon`，

下面定义了一个插值积分类。

---

```
class fun{  
    ...  
}
```

---

在下面的框架中，`public class NumericalIntegrating` 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 strABEN)  
{  
    //处理输入文本框输入的字符串  
}  
/**  
 * 更新结果的数值。可以被重写，需要被重写  
 */  
public void updateUI(double res)  
{  
    //更新JFieldText的数值，来显示输出的结果  
}  
public void updateUI2(double res)  
{  
    //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{  
  
    static String strValueN = new String();  
    //代表文本框输入的步长  
    static String strValueABE = new String();  
    //代表文本框输入的参数，a,b,e  
    static int valueN;  
    //代表int值的n步长
```

```

static double valueA,valueB,valueE;
//代表double值的a, b, e
static int flag = 0;
//0表示没有e, 1表示有e
static String FunType = new String("NULL");
// 默认的初始模式是空模式
static String ModeType = new String("NULL");
//
static int FunTypeInt = 0;//1 2 3 4
//表示选择的函数 有1, 2, 3, 4四个函数
static int ModeTypeInt = 0;//5 6 7
//表示使用的积分的类型, 有三种不同的积分类型
static Graphics g;
//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 JLabel Jresult;
static JTextField jFieldMode = new JTextField(120);// 模式选择
static JTextField jFieldFun = new JTextField(120);
static JTextField jFieldResult = new JTextField(120);
static JTextField jFieldResult2 = new JTextField(120);
static JLabel labelImg = new JLabel();//载入函数图片的jlabel
static fun f = new fun();//初始化一个fun
public static void main(String[] args) {
    System.out.println("Test Success!");
    NumericalIntegrating NI = new NumericalIntegrating();

    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 ABNE)
{
    ...
}
/**
 * 更新结果的数值。可以被重写, 需要被重写
 */
public void updateUI(double res)
{
    //更新插值结果UI的函数
}
public void updateModeStr(int num)// mode表示模式的意思, 即插值的类
型
{
    //更新插值模式UI的函数
}
public void initUI()
{
    ...
}
public void initMenuBar()
{
    ...
}
}

class fun{

```

```

public double a, b, h, epsilon;
public int n;
public int selectedFun;
public void setData(double aa, double bb, int nn, int f){
    //设置a, b, n, 以及插值函数的类型
    ...
}
public void setEpsilon(double e){
    //设置e的值
    ...
}
private double calculate(double x, int num) {
    //计算函数值, x代表x点, num代表函数的类型
    //1表示第一个函数, 2表示第二个函数, 以此类推
    ...
}
/**
 * modetype == 1 复合梯形
 * modetype == 2 simpson
 * modetype == 3 Romberg
 * @return
 */
public double calculateFun(int modetype) {
    //计算积分
    ...
}
}

```

---

## 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, valueA, valueB, valueN, valueE来存储读入的数据。

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。

其中, 因为一个函数要实现同时可以对两个文本框进行读取, 所以需要判断参数的数量。

```
if (str.length == 1)//一个参数, 只有N
```

```
else if (str.length == 2)//两个参数, 没有e, 只有AB
```

```
else if (str.length == 3)//三个参数ABE
```

---

```

public void processInput(String strABEN)
{
    String [] str = strABEN.split(" ");
    if (str.length == 1)//一个参数, 只有N
    {
        valueN = Integer.valueOf(str[0]);
    }
    else if (str.length == 2)//两个参数, 没有e, 只有AB
    {
        valueA = Double.parseDouble(str[0]);
    }
}

```

```

        valueB = Double.parseDouble(str[1]);
        flag = 0;
    }
    else if (str.length == 3)//三个参数ABE
    {
        valueA = Double.parseDouble(str[0]);
        valueB = Double.parseDouble(str[1]);
        valueE = Double.parseDouble(str[2]);
        flag = 1;
    }
}

```

### 3.3 更新 UI

积分的结果显示在JTextField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。

同时模式选定的结果也会做出更新，在 `updateModeStr` 中得以实现。

当前函数：2

当前积分公式：复化梯形积分

**NOTE**：相比于第一次实验，这次UI更新增加了 **错误信息** 的提示，错误信息主要用于处理 **简单的错误输入**。

请输入积分上下限(请...  
输入格式: a b e(若有)

请输入积分步长中的n(偶数):

积分结果:

err: 请选择函数和积分方法！

请输入积分上下限(请... 输入格式: a b e(若有)	<input type="text"/>
请输入积分步长中的n(偶数):	<input type="text"/>
积分结果:	err: 请输入参数!

请输入积分上下限(请... 输入格式: a b e(若有)	123
请输入积分步长中的n(偶数):	3
积分结果:	err: 请输入正确数量的参数!

```
/**
 * 更新结果的UI
 * @param res
 */
public void updateUI(double res)
{
    String restr = String.valueOf(res);
    jTextFieldResult.setText(restr);
}
```

```
/**
 * 更新错误信息的UI
 * @param error
 */
public void updateUI(String error)
{
    String err = new String(error);
    jTextFieldResult.setText(err);
}
```

```
/**
 * 更新结果2的UI
 * @param res
 */
public void updateUI2(double res)
{
    String restr = String.valueOf(res);
```

```

        jTextFieldResult2.setText(restr);
    }

    /**
     * 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)
    {
        if (num == 1) {
            FunType = new String("当前函数: 1");
            FunTypeInt = 1;
            jTextFieldMode.setText(FunType);
        } else if (num == 2) {
            FunType = new String("当前函数: 2");
            FunTypeInt = 2;
            jTextFieldMode.setText(FunType);
        } else if (num == 3) {
            FunType = new String("当前函数: 3");
            FunTypeInt = 3;
            jTextFieldMode.setText(FunType);
        } else if (num == 4) {
            FunType = new String("当前函数: 4");
            FunTypeInt = 4;
            jTextFieldMode.setText(FunType);
        }
        else if (num == 5)
        {
            ModeType = new String("当前积分公式: 复化梯形积分");
            ModeTypeInt = 1;
            jTextFieldFun.setText(ModeType);
        }
        else if (num == 6)
        {
            ModeType = new String("当前积分模式: 复化Simpson积分");
            ModeTypeInt = 2;
            jTextFieldFun.setText(ModeType);
        }
        else if (num == 7)
        {
            ModeType = new String("当前积分模式: Romberg积分");
            ModeTypeInt = 3;
            jTextFieldFun.setText(ModeType);
        }
    }
}

```

---

## 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("数值积分函数: ");
label.setBounds(20, 20, 130, 20);
label.setForeground(Color.BLUE);
p.add(label);
JLabel labelfun = new JLabel("数值积分公式: ");
labelfun.setBounds(20, 70, 130, 20);
labelfun.setForeground(Color.BLUE);
p.add(labelfun);
JLabel label2 = new JLabel("请输入积分上下限(请输入非分数): ");
label2.setBounds(20, 120, 130, 20);
p.add(label2);
JLabel labelPrompt = new JLabel("输入格式: a b e(若有)");
labelPrompt.setBounds(20, 135, 130, 20);
p.add(labelPrompt);
JLabel label3 = new JLabel("请输入积分步长中的n(偶数): ");
label3.setBounds(20, 180, 180, 20);
p.add(label3);
JLabel labelImg = new JLabel();
labelImg.setBounds(470, 20, 300, 180);
labelImg.setIcon(new ImageIcon("img/fun1.png"));
p.add(labelImg);
JLabel label5 = new JLabel("积分结果: ");
label5.setBounds(20, 240, 100, 20);
p.add(label5);

JLabel label6 = new JLabel("最佳步长: ");
label6.setBounds(20, 310, 100, 20);
p.add(label6);

```

随后添加开始计算按钮。

```

JButton button1 = new JButton("开始计算");//
button1.setBounds(250, 350, 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` 表示为当前的积分模式类型，当鼠标点击按钮时，获取两个文本框字符串的值，并存入字符串 `tmpStrABE`、`tmpStrN`，通过 `public void processInput(String strABEN)` 函数进行处理，并调用 `updateUI` 函数对结果进行更新。

```

监听      button1.addActionListener(new ActionListener()// 对按钮增加
        {
            // 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) {
            //定义了两个临时的字符串存储数据
            String tmpStrABE = new String("");
            String tmpStrN = new String("");
            tmpStrABE = jTextFieldX.getText();
            tmpStrN = jTextFieldN.getText();
            //以下是对输入异常的处理，出现异常则调用updateUI来提示用户
            if (ModeTypeInt == 0 || FunTypeInt == 0)
            {
                updateUI("err: 请选择函数和积分方法 !");
            }
            else if (tmpStrABE.compareTo("") == 0 ||
tmpStrN.compareTo("") == 0)
            {
                updateUI("err: 请输入参数!");
            }
            else if (tmpStrABE.split(" ").length == 1)
            {
                updateUI("err: 请输入正确数量的参数 !");
            }
            //没有出现输入异常，就调用processInput函数处理输入
            else
            {
                processInput(tmpStrABE);
                processInput(tmpStrN);

                if (flag == 0)
                {
                    f.setData(valueA, valueB, valueN,
FunTypeInt);
                }
                else if (flag == 1)
                {
                    f.setData(valueA, valueB, valueN,
FunTypeInt);
                    f.setEpsilon(valueE);
                }
                //调用f.calculate来实现结果的更新
                updateUI(f.calculateFun(ModeTypeInt));
            }
        }
    }
});

```

---

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());
    // 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("数值积分函数: ");
    label.setBounds(20, 20, 130, 20);
    label.setForeground(Color.BLUE);
    p.add(label);
    JLabel labelfun = new JLabel("数值积分公式: ");
    labelfun.setBounds(20, 70, 130, 20);
    labelfun.setForeground(Color.BLUE);
    p.add(labelfun);
    JLabel label2 = new JLabel("请输入积分上下限(请输入非分数: ");
    label2.setBounds(20, 120, 130, 20);
    p.add(label2);
    JLabel labelPrompt = new JLabel("输入格式: a b e(若有)");
    labelPrompt.setBounds(20, 135, 130, 20);
    p.add(labelPrompt);
    JLabel label3 = new JLabel("请输入积分步长中的n(偶数: ");
    label3.setBounds(20, 180, 180, 20);
    p.add(label3);
    JLabel labelImg = new JLabel();
    labelImg.setBounds(470, 20, 300, 180);
    labelImg.setIcon(new ImageIcon("img/fun1.png"));
    p.add(labelImg);
    JLabel label5 = new JLabel("积分结果: ");
    label5.setBounds(20, 240, 100, 20);
    p.add(label5);
    JLabel label6 = new JLabel("最佳步长: ");
    label6.setBounds(20, 310, 100, 20);
    p.add(label6);

    // frame.add(label);
    /**
     * JTextField的设置 创建文本框, 指定可见列数为80列
     */
    jTextFieldMode.setText("当前函数: 未选择");
    jTextFieldMode.setEditable(false);
}

```

```

jFieldMode.setBounds(250, 20, 200, 30);
jFieldMode.setForeground(Color.RED);
p.add(jFieldMode);

JFieldFun.setText("当前积分公式: 未选择");
JFieldFun.setEditable(false);
JFieldFun.setBounds(250, 70, 200, 30);
JFieldFun.setForeground(Color.RED);
p.add(JFieldFun);

final JTextField jFieldX = new JTextField(80);
jFieldX.setBounds(250, 120, 200, 30);
p.add(jFieldX);
final JTextField jFieldN = new JTextField(80);
jFieldN.setBounds(250, 180, 200, 30);
p.add(jFieldN);
JFieldResult = new JTextField(80);
JFieldResult.setEditable(false);
JFieldResult.setBounds(250, 240, 200, 30);
p.add(JFieldResult);

JFieldResult2 = new JTextField(80);
JFieldResult2.setEditable(false);
JFieldResult2.setBounds(250, 300, 200, 30);
p.add(JFieldResult2);
/**
 * 这里是对Buttons的设置
 */
JButton button1 = new JButton("开始计算");//
button1.setBounds(250, 350, 200, 40);// 设置按钮在容器中的位置

p.add(button1);

```

监听

```

button1.addActionListener(new ActionListener()// 对按钮增加
{
    // 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) {
    //定义了两个临时的字符串存储数据
    String tmpStrABE = new String("");
    String tmpStrN = new String("");
    tmpStrABE = jFieldX.getText();
    tmpStrN = jFieldN.getText();
    //以下是对输入异常的处理，出现异常则调用updateUI来提示用户
    if (ModeTypeInt == 0 || FunTypeInt == 0)
    {
        updateUI("err: 请选择函数和积分方法 !");
    }
    else if (tmpStrABE.compareTo("") == 0 ||
tmpStrN.compareTo("") == 0)
    {
        updateUI("err: 请输入参数!");
    }
    else if (tmpStrABE.split(" ").length == 1)
    {
        updateUI("err: 请输入正确数量的参数 !");
    }
    //没有出现输入异常，就调用processInput函数处理输入
    else
    {
        processInput(tmpStrABE);
        processInput(tmpStrN);
    }
}

```

```

        if (flag == 0)
        {
            f.setData(valueA, valueB, valueN,
FunTypeInt);
        }
        else if (flag == 1)
        {
            f.setData(valueA, valueB, valueN,
FunTypeInt);
            f.setEpsilon(valueE);
        }
        //调用f.calculate来实现结果的更新
        updateUI(f.calculateFun(ModeTypeInt));
    }
}
});
JButton button2 = new JButton("确定最佳步长");//
button2.setBounds(20, 350, 200, 40);// 设置按钮在容器中的位置

p.add(button2);
button2.addActionListener(new ActionListener(){

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String tmpStrABE = new String("");
        String tmpStrN = new String("");
        tmpStrABE = jTextFieldX.getText();
        tmpStrN = jTextFieldN.getText();

        processInput(tmpStrABE);
        processInput(tmpStrN);
        if (flag == 0)
        {
            f.setData(valueA, valueB, valueN, FunTypeInt);
        }
        else if (flag == 1)
        {
            f.setData(valueA, valueB, valueN, FunTypeInt);
            f.setEpsilon(valueE);
        }

        double [] resarr = new double[120];
        for (int i = 2 ;i <= 100;i++)
        {
            f.setData(valueA, valueB, i,FunTypeInt);
            resarr[i - 2] = f.calculateFun(ModeTypeInt);
        }
    }
});
/**
 * 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。

本次实验实例化了两个JMenu为Menu1和Menu2，分别代表了 **积分函数选择** 和 **积分公式选择**。

实例化了JMenuItem如下：

---

```
funItem1 = new JMenuItem("函数1");
funItem2 = new JMenuItem("函数2");
funItem3 = new JMenuItem("函数3");
funItem4 = new JMenuItem("函数4");
trape = new JMenuItem("复合梯形公式");
simpson = new JMenuItem("复合Simpson公式");
romberg = new JMenuItem("Romberg算法");
```

---

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, funItem4;
    JMenuItem trape, simpson, romberg;
    JMenuBar menuBar = new JMenuBar();

    funItem1 = new JMenuItem("函数1");
    funItem2 = new JMenuItem("函数2");
    funItem3 = new JMenuItem("函数3");
    funItem4 = new JMenuItem("函数4");
    trape = new JMenuItem("复合梯形公式");
    simpson = new JMenuItem("复合Simpson公式");
    romberg = new JMenuItem("Romberg算法");
    Menu1 = new JMenu("积分函数选择");
    Menu2 = new JMenu("积分公式选择");

    Menu1.add(funItem1);
    Menu1.add(funItem2);
    Menu1.add(funItem3);
    Menu1.add(funItem4);
    Menu1.setSelected(true);
    Menu2.add(trape);
    Menu2.add(simpson);
    Menu2.add(romberg);
    Menu2.setSelected(true);
}
```

```

menuBar.add(Menu1);
menuBar.add(Menu2);
frame.setJMenuBar(menuBar);
funItem1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(1);
        System.out.println("当前函数: 1");
        labelImg.setIcon(new ImageIcon("img/fun1.png"));
    }
});
funItem2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(2);
        System.out.println("当前函数: 2");
        labelImg.setIcon(new ImageIcon("img/fun2.png"));
    }
});
funItem3.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(3);
        labelImg.setIcon(new ImageIcon("img/fun3.png"));
        System.out.println("当前函数: 3");
    }
});
funItem4.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(4);
        labelImg.setIcon(new ImageIcon("img/fun4.png"));
        System.out.println("当前函数: 4");
    }
});
trape.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(5);
        System.out.println("当前积分公式: 复化梯形积分");
    }
});
simpson.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(6);
        System.out.println("当前积分公式: 复化Simpson积分");
    }
});
romberg.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateModeStr(7);
        System.out.println("当前积分公式: Romberg积分");
    }
});
}

```



---

## 3.6 插值积分的实现

类内函数主要有：

- `public void setData(double aa, double bb, int nn, int f)`
- `public void setEpsilon(double e)`
- `private double calculate(double x, int num)`
- `public double calculateFun(int modetype)`

---

```
class fun{
    //定义了积分的上下限，步长，n和e
    public double a, b, h, epsilon;
    public int n;
    //selectedFun表示被选定的函数的类型
    public int selectedFun;
    //设置类内参数的a, b, n, 和用来表示函数类型的int值
    public void setData(double aa, double bb, int nn, int f) {
        a = aa;
        b = bb;
        n = nn;
        h = (b - a) / n;
        selectedFun = f;
    }
    //定义了类内参数的e
    public void setEpsilon(double e)
    {
        epsilon = e;
    }
    /**
     * @param x
     * @param num 1表示第一个函数，2表示第二个函数，以此类推
     * @return 计算出的类型为num的函数值
     */
    private double calculate(double x, int num) {
        //计算函数值，x代表x点，num代表函数的类型
        if (num == 1)
            return Math.sqrt(4 - (Math.sin(x)) * (Math.sin(x)));
        if (num == 2)
        {
            if (x == 0)
                return (double)(Math.cos(x)); //洛必达法则，0/0型
            else
                return (double) (Math.sin(x) / x);
        }
        if (num == 3)
            return (double) ((Math.pow(Math.E, x)) / (4 + x * x));
        if (num == 4)
            return (double) (Math.log(1 + x) / (1 + x * x));
        return -1;
    }

    /**
     * modetype == 1 复合梯形
     * modetype == 2 simpson
     * modetype == 3 Romberg
     * @return
```

```

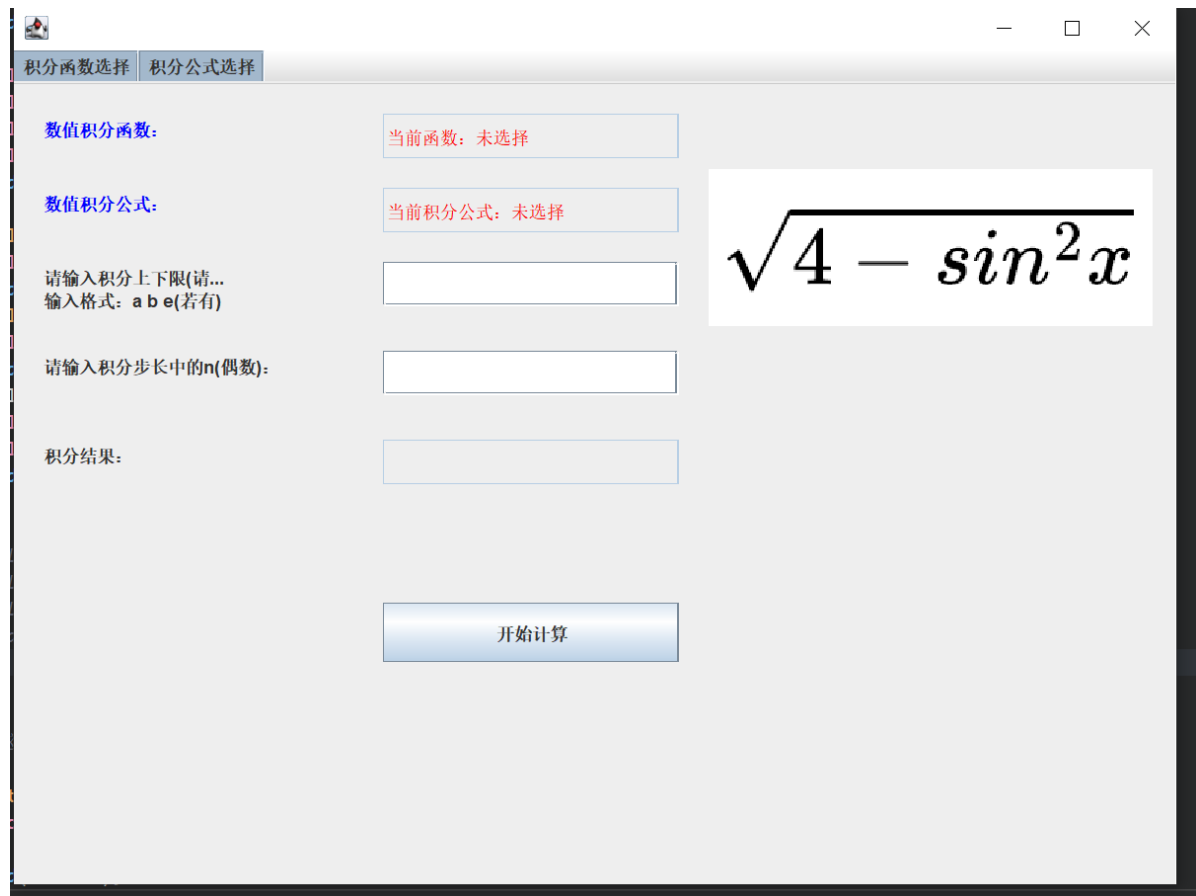
    */
    public double calculateFun(int modetype) {
        if (modetype == 1)
        {
            double res = 0;
            double cur = a;
            int times = n;
            while (times > 0) {
                res += h / 2 * (calculate(cur, selectedFun) +
calculate(cur + h, selectedFun));
                times--;
                cur += h;
            }
            return res;
        }
        if (modetype == 2)//如果是
        {
            double res = 0;
            double cur = a;
            int times = n / 2;
            while (times > 0)
            {
                res += h / 3 * (calculate(cur, selectedFun) + 4 *
calculate(cur + h, selectedFun) + calculate(cur + 2 * h,
selectedFun));
                times--;
                cur += 2 * h;
            }
            return res;
        }
        if (modetype == 3)
        {
            int m = 1, k = 1;
            double hh = (b - a) / 2.0;
            double T0 = hh * (calculate(a,selectedFun) +
calculate(b,selectedFun)), T = 3;
            double F = 0;
            while(Math.abs(T - T0) >= 3 * epsilon)
            {
                if(m != 1)
                    T0 = T;
                F = 0;
                k = (int) Math.pow(2., m - 1);
                for(int i = 1; i <= k ; i++)
                {
                    F += calculate(a + (2 * i - 1) * hh ,
selectedFun);
                }
                T = T0 / 2.0 + hh * F;
                m += 1;
                hh /= 2.0;
            }
            return T;
        }
        return -1;
    }
}

```

---

## 4 实验结果总结

在使用的時候，先在Menu菜单中选择函数的类型和插值方法，然后输入积分上下限以及e的值，和n的值，点击开始计算后即可获得结果



同时结尾指出实验指导书的一处错误

$$I = \int_0^{\frac{1}{4}} \sqrt{4 - \sin^2 x} dx \quad (I \approx 1.5343916)$$

通过三种不同类型的插值积分计算以及Wolfram计算得出正确结果应该如下：

```
问题 2 输出 终端 调试控制台
t.java\jdt_ws\code2_18a0c212\bin' 'NumericalIntegrating'
Test Success!
0.4987095448522963
0.4987111176296314
0.49855371057938846
```

### Definite integral

$$\int_0^{0.25} \sqrt{4 - \sin^2(x)} \, dx = 0.498711$$

对第二个函数测试的时候发现了

```
Test Success!  
NaN  
NaN  
3.0  
█
```

在debug的过程中，发现代码并没有错误，后来意识到 $\sin x / x$  是零比零型，代码并没有对此进行处理。

原本直接

---

```
return (double) (Math.sin(x) / x);
```

---

需要式用洛必达法则

---

```
if (x == 0)  
    return (double) (Math.cos(x)); //洛必达法则, 0/0型  
else  
    return (double) (Math.sin(x) / x);
```

---

## 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 NumericalIntegrating {

    static String strValueN = new String();
    //代表文本框输入的步长
    static String strValueABE = new String();
    //代表文本框输入的参数, a,b,e
    static int valueN;
    //代表int值的n步长
    static double valueA,valueB,valueE;
    //代表double值的a, b, e
    static int flag = 0;
    //0表示没有e, 1表示有e
    static String FunType = new String("NULL");
    // 默认的初始模式是空模式
    static String ModeType = new String("NULL");
    //
    static int FunTypeInt = 0;//1 2 3 4
    //表示选择的函数 有1, 2, 3, 4四个函数
    static int ModeTypeInt = 0;//5 6 7
    //表示使用的积分的类型, 有三种不同的积分类型
    static Graphics g;
    //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 JLabel Jresult;
    static JTextField jFieldMode = new JTextField(120);// 模式选择
    static JTextField jFieldFun = new JTextField(120);
    static JTextField jFieldResult = new JTextField(120);
    static JTextField jFieldResult2 = new JTextField(120);
    static JLabel labelImg = new JLabel();//载入函数图片的jlabel
    static fun f = new fun();
    public static void main(String[] args) {
        System.out.println("Test Success!");
        NumericalIntegrating NI = new NumericalIntegrating();

        NI.initMenuBar();// 初始化菜单栏
        NI.initUI();// 初始化UI界面
        /*
        fun f = new fun();
        f.setData(0, 1, 20, 4);
        f.setEpsilon(0.001);
        System.out.println(f.calculateFun(1));
        System.out.println(f.calculateFun(2));
        System.out.println(f.calculateFun(3));
        */
    }
    public void processInput(String strABEN)
    {
        String [] str = strABEN.split(" ");
        if (str.length == 1)//一个参数, 只有N
        {
            valueN = Integer.valueOf(str[0]);
        }
        else if (str.length == 2)//两个参数, 没有e, 只有AB
        {
            valueA = Double.parseDouble(str[0]);
            valueB = Double.parseDouble(str[1]);
            flag = 0;
        }
    }
}

```

```

        else if (str.length == 3)//三个参数ABE
        {
            valueA = Double.parseDouble(str[0]);
            valueB = Double.parseDouble(str[1]);
            valueE = Double.parseDouble(str[2]);
            flag = 1;
        }
    }
    /**
     * 更新结果的UI
     * @param res
     */
    public void updateUI(double res)
    {
        String restr = String.valueOf(res);
        jTextFieldResult.setText(restr);
    }
    /**
     * 更新错误信息的UI
     * @param error
     */
    public void updateUI(String error)
    {
        String err = new String(error);
        jTextFieldResult.setText(err);
    }
    /**
     * 更新结果2的UI
     * @param res
     */
    public void updateUI2(double res)
    {
        String restr = String.valueOf(res);
        jTextFieldResult2.setText(restr);
    }
}

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("数值积分函数: ");
    label.setBounds(20, 20, 130, 20);
    label.setForeground(Color.BLUE);
    p.add(label);
    JLabel labelfun = new JLabel("数值积分公式: ");
    labelfun.setBounds(20, 70, 130, 20);
    labelfun.setForeground(Color.BLUE);
    p.add(labelfun);
    JLabel label2 = new JLabel("请输入积分上下限(请输入非分数: ");
    label2.setBounds(20, 120, 130, 20);
    p.add(label2);
    JLabel labelPrompt = new JLabel("输入格式: a b e(若有)");
    labelPrompt.setBounds(20, 135, 130, 20);
    p.add(labelPrompt);
    JLabel label3 = new JLabel("请输入积分步长中的n(偶数: ");
    label3.setBounds(20, 180, 180, 20);
    p.add(label3);
    JLabel labelImg = new JLabel();
    labelImg.setBounds(470, 20, 300, 180);
    labelImg.setIcon(new ImageIcon("img/fun1.png"));
    p.add(labelImg);
    JLabel label5 = new JLabel("积分结果: ");
    label5.setBounds(20, 240, 100, 20);
    p.add(label5);
    JLabel label6 = new JLabel("最佳步长: ");
    label6.setBounds(20, 310, 100, 20);
    p.add(label6);

    // frame.add(label);
    /**
     * JTextField的设置 创建文本框, 指定可见列数为80列
     */
    JFieldMode.setText("当前函数: 未选择");
    JFieldMode.setEditable(false);
    JFieldMode.setBounds(250, 20, 200, 30);
    JFieldMode.setForeground(Color.RED);
    p.add(JFieldMode);

    JFieldFun.setText("当前积分公式: 未选择");
    JFieldFun.setEditable(false);
    JFieldFun.setBounds(250, 70, 200, 30);
    JFieldFun.setForeground(Color.RED);
    p.add(JFieldFun);

    final JTextField jFieldx = new JTextField(80);
    jFieldx.setBounds(250, 120, 200, 30);
    p.add(jFieldx);
    final JTextField jFieldn = new JTextField(80);
    jFieldn.setBounds(250, 180, 200, 30);
    p.add(jFieldn);
    JFieldResult = new JTextField(80);
    JFieldResult.setEditable(false);
    JFieldResult.setBounds(250, 240, 200, 30);
    p.add(JFieldResult);

    JFieldResult2 = new JTextField(80);
    JFieldResult2.setEditable(false);
    JFieldResult2.setBounds(250, 300, 200, 30);
    p.add(JFieldResult2);
    /**
     * 这里是对Buttons的设置
     */
    JButton button1 = new JButton("开始计算");//

```

```

button1.setBounds(250, 350, 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) {
        //定义了两个临时的字符串存储数据
        String tmpStrABE = new String("");
        String tmpStrN = new String("");
        tmpStrABE = jTextFieldX.getText();
        tmpStrN = jTextFieldN.getText();
        //以下是对输入异常的处理，出现异常则调用updateUI来提示用户
        if (ModeTypeInt == 0 || FunTypeInt == 0)
        {
            updateUI("err: 请选择函数和积分方法 !");
        }
        else if (tmpStrABE.compareTo("") == 0 ||
tmpStrN.compareTo("") == 0)
        {
            updateUI("err: 请输入参数!");
        }
        else if (tmpStrABE.split(" ").length == 1)
        {
            updateUI("err: 请输入正确数量的参数 !");
        }
        //没有出现输入异常，就调用processInput函数处理输入
        else
        {
            processInput(tmpStrABE);
            processInput(tmpStrN);

            if (flag == 0)
            {
                f.setData(valueA, valueB, valueN,
FunTypeInt);
            }
            else if (flag == 1)
            {
                f.setData(valueA, valueB, valueN,
FunTypeInt);
                f.setEpsilon(valueE);
            }
            //调用f.calculate来实现结果的更新
            updateUI(f.calculateFun(ModeTypeInt));
        }
    }
});
JButton button2 = new JButton("确定最佳步长");//
button2.setBounds(20, 350, 200, 40);// 设置按钮在容器中的位置

p.add(button2);
button2.addActionListener(new ActionListener(){

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String tmpStrABE = new String("");
        String tmpStrN = new String("");
        tmpStrABE = jTextFieldX.getText();
        tmpStrN = jTextFieldN.getText();
    }
});

```



```

        processInput(tmpStrABE);
        processInput(tmpStrN);
        if (flag == 0)
        {
            f.setData(valueA, valueB, valueN, FunTypeInt);
        }
        else if (flag == 1)
        {
            f.setData(valueA, valueB, valueN, FunTypeInt);
            f.setEpsilon(valueE);
        }

        double [] resarr = new double[120];
        for (int i = 2 ; i <= 100; i++)
        {
            f.setData(valueA, valueB, i, FunTypeInt);
            resarr[i - 2] = f.calculateFun(ModeTypeInt);
        }
    }
});
/**
 * 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, funItem4;
    JMenuItem trape, simpson, romberg;
    JMenuBar menuBar = new JMenuBar();

    funItem1 = new JMenuItem("函数1");
    funItem2 = new JMenuItem("函数2");
    funItem3 = new JMenuItem("函数3");
    funItem4 = new JMenuItem("函数4");
    trape = new JMenuItem("复合梯形公式");
    simpson = new JMenuItem("复合Simpson公式");
    romberg = new JMenuItem("Romberg算法");
    Menu1 = new JMenu("积分函数选择");
    Menu2 = new JMenu("积分公式选择");

    Menu1.add(funItem1);
    Menu1.add(funItem2);
    Menu1.add(funItem3);
    Menu1.add(funItem4);
    Menu1.setSelected(true);
    Menu2.add(trape);
    Menu2.add(simpson);
    Menu2.add(romberg);
    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("当前函数: 1");
            labelImg.setIcon(new ImageIcon("img/fun1.png"));
        }
    });
    funItem2.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("newton");
            updateModeStr(2);
            System.out.println("当前函数: 2");
            labelImg.setIcon(new ImageIcon("img/fun2.png"));
        }
    });
    funItem3.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("seg");
            updateModeStr(3);
            labelImg.setIcon(new ImageIcon("img/fun3.png"));
            System.out.println("当前函数: 3");
        }
    });
    funItem4.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("seg");
            updateModeStr(4);
            labelImg.setIcon(new ImageIcon("img/fun4.png"));
            System.out.println("当前函数: 4");
        }
    });
    trape.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("seg");
            // updateModeStr(4);
            updateModeStr(5);
            System.out.println("当前积分公式: 复化梯形积分");
        }
    });
    simpson.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("seg");
            // updateModeStr(4);
            updateModeStr(6);
            System.out.println("当前积分公式: 复化Simpson积分");
        }
    });
    romberg.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // updateModeStr("seg");
            // updateModeStr(4);

```

```

        updateModeStr(7);
        System.out.println("当前积分公式: Romberg积分");
    }
});
}
/**
 * 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) {
        FunType = new String("当前函数: 1");
        FunTypeInt = 1;
        jFieldMode.setText(FunType);
    } else if (num == 2) {
        FunType = new String("当前函数: 2");
        FunTypeInt = 2;
        jFieldMode.setText(FunType);
    } else if (num == 3) {
        FunType = new String("当前函数: 3");
        FunTypeInt = 3;
        jFieldMode.setText(FunType);
    } else if (num == 4) {
        FunType = new String("当前函数: 4");
        FunTypeInt = 4;
        jFieldMode.setText(FunType);
    }
    else if (num == 5)
    {
        ModeType = new String("当前积分公式: 复化梯形积分");
        ModeTypeInt = 1;
        JFieldFun.setText(ModeType);
    }
    else if (num == 6)
    {
        ModeType = new String("当前积分模式: 复化Simpson积分");
        ModeTypeInt = 2;
        JFieldFun.setText(ModeType);
    }
    else if (num == 7)
    {
        ModeType = new String("当前积分模式: Romberg积分");
        ModeTypeInt = 3;
        JFieldFun.setText(ModeType);
    }
}

}

class fun{
    //定义了积分的上下限，步长，n和e
    public double a, b, h, epsilon;
    public int n;
    //selectedFun表示被选定的函数的类型
    public int selectedFun;
    //设置类内参数的a, b, n, 和用来表示函数类型的int值
    public void setData(double aa, double bb, int nn, int f) {
        a = aa;
        b = bb;
        n = nn;
        h = (b - a) / n;
        selectedFun = f;
    }
    //定义了类内参数的e

```

```

public void setEpsilon(double e)
{
    epsilon = e;
}
/**
 * @param x
 * @param num 1表示第一个函数, 2表示第二个函数, 以此类推
 * @return 计算出的类型为num的函数值
 */
private double calculate(double x, int num) {
    //计算函数值, x代表x点, num代表函数的类型
    if (num == 1)
        return Math.sqrt(4 - (Math.sin(x)) * (Math.sin(x)));
    if (num == 2)
    {
        if (x == 0)
            return (double)(Math.cos(x)); //洛必达法则, 0/0型
        else
            return (double) (Math.sin(x) / x);
    }
    if (num == 3)
        return (double) ((Math.pow(Math.E, x)) / (4 + x * x));
    if (num == 4)
        return (double) (Math.log(1 + x) / (1 + x * x));
    return -1;
}

/**
 * modetype == 1 复合梯形
 * modetype == 2 simpson
 * modetype == 3 Romberg
 * @return
 */
public double calculateFun(int modetype) {
    if (modetype == 1)
    {
        double res = 0;
        double cur = a;
        int times = n;
        while (times > 0) {
            res += h / 2 * (calculate(cur, selectedFun) +
calculate(cur + h, selectedFun));
            times--;
            cur += h;
        }
        return res;
    }
    if (modetype == 2) //如果是
    {
        double res = 0;
        double cur = a;
        int times = n / 2;
        while (times > 0)
        {
            res += h / 3 * (calculate(cur, selectedFun) + 4 *
calculate(cur + h, selectedFun) + calculate(cur + 2 * h,
selectedFun));
            times--;
            cur += 2 * h;
        }
        return res;
    }
    if (modetype == 3)
    {
        int m = 1, k = 1;
    }
}

```

```

        double hh = (b - a) / 2.0;
        double T0 = hh * (calculate(a,selectedFun) +
calculate(b,selectedFun)), T = 3;
        double F = 0;
        while(Math.abs(T - T0) >= 3 * epsilon)
        {
            if(m != 1)
                T0 = T;
            F = 0;
            k = (int) Math.pow(2., m - 1);
            for(int i = 1; i <= k ; i++)
            {
                F += calculate(a + (2 * i - 1) * hh ,
selectedFun);
            }
            T = T0 / 2.0 + hh * F;
            m += 1;
            hh /= 2.0;
        }
        return T;
    }
    return -1;
}
}

```

---