*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： 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： 谢董奎	组号：	实验时间： 9：00AM

1 实验目的与要求

1、利用幂法或反幂法，求方阵$A=(a_{ij})_{n*n}$ 的按模最大或按模最小特征值及其对应的特征向量。

掌握幂法或反幂法求矩阵部分特征值的算法与程序设计；

2、会用原点平移法改进算法，加速收敛；对矩阵 B=A-PI 取不同的 P 值，试求其效果；

3、试取不同的初始向量 υ (0)，观察对结果的影响；

4、对矩阵特征值的其它分布，如 λ 1 = λ 2 且 λ 1 = λ 2 ≥ λ 3 如何计算。

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 SolvingLinearEquations` 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);// mode表示模式的意思，即插值的类型；
public void initMenuBar();
public void initUI();
public void processInput(String sa, String sb);
```

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;
```

```java
import javax.swing.*;
import java.awt.Graphics;
public class SolvingLinearEquations{

    static String strb;
    static ArrayList<String> strA = new ArrayList<String>();
    static double[][] a;
    static double[] b;
    static public int dim;
    static String test1a = new String("-1 2 1\n2 -4 1\n1 1 -6");
    static String test2a = new String("4 -2 7 3 -1 8\n-2 5 1 1 4 7\n7 1 7 2 3 5\n3 1 2
6 5 1\n-1 4 3 5 3 2\n8 7 5 1 2 4");
    static String test3a = new String("2 -1 0 0 0\n-1 2 -1 0 0\n0 -1 2 -1 0\n0 0 -1 2
-1\n0 0 0 -1 2");
    static String test4a = new String("2 1 3 4\n1 -3 1 5\n3 1 6 -2\n4 5 -2 -1");
    static String test5a = new String("-1 2 1\n2 -4 1\n1 1 -6");
    static double eps;
    static Fun fun = new Fun();
    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 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 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);//设置精度
    static JTextField jFieldResult3 = new JTextField(120);//设置初始值
    public static void main(String[] args) {
        System.out.println("Test Success!");
        SolveCharacteristicValue NI = new SolveCharacteristicValue();

        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)// 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;
    public double[] V;
    public double[] U;
    public double s,mu0,mu = -1000;
    public double eps = 0.0001;
    private int N;
    public void setData(double a[][], int dim,double u[],double e);
    public String[] cal();
}
```

3.2 实现输入输出

`public class SolveCharacteristicValue` 为主Public类内定义了许多静态变量，strb和ArrayList strA 来存取被读入的数据。

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。

处理输入A前，需要对ArrayList进行清空操作。

```
for (int i = strA.size() - 1; i >= 0; i--)    strA.remove(i);
```

然后调用 `ProcessInput` 函数。

在 `ProcessInput` 函数中，对A的输入进行了分割，分割符为空格或者回车，全部分割后再映射到 `A[][]` 矩阵，对B的分割则按照空格进行分割。

```
        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
                for (int i = strA.size() - 1; i >= 0; i--) {
                    strA.remove(i);
                }
                // 因为是ArrayList，所以每次使用前需要清空
                processInput(jarea.getText());
                // 将jArea中的字符串处理成字符串数组
                a = new double[dim][dim];
                int cnt = 0;
                for (int i = 0; i < dim; i++) {
                    for (int j = 0; j < dim; j++) {
                        a[i][j] = Double.parseDouble(strA.get(cnt++));
                    }
                }
                double u[] = new double[dim];
```

```
                for (int i = 0;i < dim;i++)
                {
                    u[i] = 1.0;
                }
                eps = Double.valueOf(jFieldResult2.getText());
                fun.setData(a, dim,u,eps);
                jFieldResult.setText(fun.cal()[0]);
                jFieldResultLam.setText(fun.cal()[1]);
            }
        });
    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]);
        }
        dim = (int)Math.sqrt(len);
    }
```

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。

```
    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("幂法");
            FunTypeInt = 1;
            jFieldMode.setText(FunType);
        } else if (num == 2) {
            FunType = new String("反幂法");
            FunTypeInt = 2;
            jFieldMode.setText(FunType);
        }
    }
```

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
> 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("输入需要求解方程组的A：");
        label.setBounds(20, 50, 200, 20);
        label.setForeground(Color.BLUE);
        p.add(label);

        JLabel label1 = new JLabel("当前选择的方程组解法：");
        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("对应的特征向量：");
        label7.setBounds(400, 365, 200, 20);
        label7.setForeground(Color.BLUE);
        p.add(label7);

        jFieldResult.setText("当前结果：未显示");
        jFieldResult.setEditable(false);
        jFieldResult.setBounds(400, 390, 300, 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(400, 425, 200, 20);
        label9.setForeground(Color.BLUE);
        p.add(label9);
```

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
                for (int i = strA.size() - 1; i >= 0; i--) {
                    strA.remove(i);
                }
                // 因为是ArrayList，所以每次使用前需要清空
                processInput(jarea.getText());
                // 将jArea中的字符串处理成字符串数组
                a = new double[dim][dim];
                int cnt = 0;
                for (int i = 0; i < dim; i++) {
                    for (int j = 0; j < dim; j++) {
                        a[i][j] = Double.parseDouble(strA.get(cnt++));
                    }
                }
                double u[] = new double[dim];
                for (int i = 0;i < dim;i++)
                {
                    u[i] = 1.0;
                }
                eps = Double.valueOf(jFieldResult2.getText());
                fun.setData(a, dim,u,eps);
                jFieldResult.setText(fun.cal()[0]);
                jFieldResultLam.setText(fun.cal()[1]);
            }
        });
```

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("输入需要求解方程组的A：");
        label.setBounds(20, 50, 200, 20);
        label.setForeground(Color.BLUE);
        p.add(label);

        JLabel label1 = new JLabel("当前选择的方程组解法：");
        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("对应的特征向量：");
        label7.setBounds(400, 365, 200, 20);
        label7.setForeground(Color.BLUE);
        p.add(label7);

        jFieldResult.setText("当前结果：未显示");
        jFieldResult.setEditable(false);
        jFieldResult.setBounds(400, 390, 300, 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(400, 425, 200, 20);
        label9.setForeground(Color.BLUE);
        p.add(label9);

        jFieldResultLam.setText("当前结果：未显示");
        jFieldResultLam.setEditable(false);
        jFieldResultLam.setBounds(400, 450, 300, 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, 200, 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("精度设置：");
        label8.setBounds(500, 20, 200, 20);
        label8.setForeground(Color.BLUE);
        p.add(label8);

        jFieldResult2.setText("0.00001");
        jFieldResult2.setEditable(true);
        jFieldResult2.setBounds(560, 18, 70, 25);
        //jFieldMode.setForeground(Color.RED);
        p.add(jFieldResult2);

        // jFieldResult.setText("");
        // jFieldResult.setEditable(false);
        // jFieldResult.setBounds(560, 18, 70, 25);
        // jFieldMode.setForeground(Color.RED);
        // p.add(jFieldResult);

        // final JTextField jFieldX = new JTextField(80);
        // jFieldX.setBounds(20, 350, 200, 30);
        // p.add(jFieldX);

        /**
         * 这里是对Buttons的设置
         */
        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);

        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
                for (int i = strA.size() - 1; i >= 0; i--) {
                    strA.remove(i);
                }
                // 因为是ArrayList，所以每次使用前需要清空
                processInput(jarea.getText());
                // 将jArea中的字符串处理成字符串数组
                a = new double[dim][dim];
                int cnt = 0;
                for (int i = 0; i < dim; i++) {
                    for (int j = 0; j < dim; j++) {
                        a[i][j] = Double.parseDouble(strA.get(cnt++));
                    }
                }
                double u[] = new double[dim];
                for (int i = 0;i < dim;i++)
                {
                    u[i] = 1.0;
```

```
            }
            eps = Double.valueOf(jFieldResult2.getText());
            fun.setData(a, dim,u,eps);
            jFieldResult.setText(fun.cal()[0]);
            jFieldResultLam.setText(fun.cal()[1]);

        }
    });

    JButton button2 = new JButton("测试样例1");//
    button2.setBounds(20, 320, 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
            jarea.setText(test1a);
        }
    });

    JButton button3 = new JButton("测试样例2");
    button3.setBounds(20, 360, 200, 40);
    p.add(button3);
    button3.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            jarea.setText(test2a);


        }
    });

    JButton button4 = new JButton("测试样例3");
    button4.setBounds(20, 400, 200, 40);
    p.add(button4);
    button4.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            jarea.setText(test3a);

        }
    });
    JButton button5 = new JButton("测试样例4");
    button5.setBounds(20, 440, 200, 40);
    p.add(button5);
    button5.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            jarea.setText(test4a);

        }
```

```
        });

        JButton button6 = new JButton("测试样例5");
        button6.setBounds(20, 480, 200, 40);
        p.add(button6);
        button6.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                jarea.setText(test5a);

            }
        });
        /**
         * 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。

实例化了JMenuItem如下：

```
        JMenu Menu1;
        JMenuItem funItem1, funItem2;
        JMenuBar menuBar = new JMenuBar();
        funItem1 = new JMenuItem("幂法");
        funItem2 = new JMenuItem("反幂法");
        Menu1 = new JMenu("求解方法选择");
        Menu1.add(funItem1);
        Menu1.add(funItem2);
        Menu1.setSelected(true);
        menuBar.add(Menu1);
        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;
        JMenuItem funItem1, funItem2;
        JMenuBar menuBar = new JMenuBar();
        funItem1 = new JMenuItem("幂法");
        funItem2 = new JMenuItem("反幂法");
        Menu1 = new JMenu("求解方法选择");
        Menu1.add(funItem1);
        Menu1.add(funItem2);
        Menu1.setSelected(true);
        menuBar.add(Menu1);
```

```
        frame.setJMenuBar(menuBar);
        funItem1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // updateModeStr("lag");
                updateModeStr(1);
                System.out.println("幂法");

            }
        });
        funItem2.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // updateModeStr("newton");
                updateModeStr(2);
                System.out.println("反幂法");
            }
        });

    }
```

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 {
    public double[][] A;
    public double[] V;
    public double[] U;
    public double s,mu0,mu = -1000;
    public double eps = 0.0001;
    private int N;
    public void setData(double a[][], int dim,double u[],double e) {
        N = a.length;
        eps = e;
        // System.out.println(dimension);
        // System.out.println(123213123);
        A = new double[N + 2][N + 2];
        U = new double[N + 2];
        V = new double[N + 2];
        int i,j;
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++) {
                A[i][j] = a[i][j];
            }
        }
        for (i = 0;i < N;i++)
        {
            U[i] = u[i];
        }
    }
    /**
     * 幂法
```

```
     * @return
     */
    public String[] cal()
    {
        int i,j;
        do{
            mu0 = mu;
            for (i = 0;i < N;i++)
            {
                s = 0;
                for (j = 0;j < N;j++) s += A[i][j] * U[j];
                V[i] = s;
            }
            mu = Math.abs(V[0]);
            for (i = 0 ;i < N;i++)
            {
                if (mu < Math.abs(V[i])) mu = V[i];
            }
            for (i = 0 ; i < N;i++)
            {
                U[i] = V[i] / mu;
            }
        }while (Math.abs(mu - mu0) >= eps);
        System.out.println("///////");
        for(i = 0 ; i < N;i++)
        {
            System.out.println(U[i]);
        }
        NumberFormat nf = NumberFormat.getNumberInstance();
        nf.setMinimumFractionDigits(4);
        System.out.println(mu);
        String res[] = new String[2];
        res[0] = "";
        res[1] = "";
        for (i = 0; i < N; i++) {
            res[0] += String.valueOf(nf.format(U[i]) + " ");
        }
        res[1] = String.valueOf(nf.format(mu0));
        return res;
    }
    /**
     * 反幂法
     */
    public String[] cal2()
    {
        int n ;
        int i,j,k;
        double xmax = -9999,oxmax = 0;
        double [][]L = new double[N + 2][N + 2];
        double [][]U = new double[N + 2][N + 2];
        double []x = new double[N + 2];
        double []nx = new double[N + 2];
        for (i = 0 ; i < N;i++)
        {
            x[i] = 1;
        }
        oxmax = 0;
```

```
        for (i = 0 ; i < N;i++)
        {
            U[i][i] = 1;
        }
        for (k = 0;k < N;k++)
        {
            for (i = k;i < N ;i++)
            {
                L[i][k] = A[i][k];
                for (j = 0 ; j <= k - 1;j++)
                    L[i][k] -= (L[i][j] * U[j][k]);
            }
            for (j = k + 1;j < N;j++)
            {
                U[k][j] = A[k][j];
                for (i = 0;i <= k - 1;i++)
                {
                    U[k][j] -= (L[k][i] * U[i][j]);
                }
                U[k][j] /= L[k][k];
            }
        }
        for (i = 0;i < N;i++)
        {
            x[i] = 1;
        }
        for (i = 0;i < 100;i++)
        {
            for (j = 0;j < N;j++)
            {
                nx[j] = x[j];
                for (k = 0;k <= j - 1;k++)
                {
                    nx[j] -= L[j][k] * nx[k];
                }
                nx[j] /= L[j][j];
            }
            for (j = N - 1;j >= 0;j--)
            {
                x[j] = nx[j];
                for (k = j + 1;k < N;k++)
                {
                    x[j] -= U[j][k] * x[k];
                }
            }
            xmax = 0;
            for (j = 0;j < N;j++)
            {
                if (Math.abs(x[j]) > xmax)
                {
                    xmax = Math.abs(x[j]);
                }
            }
            for (j = 0;j < N;j++)
            {
                x[j] /= xmax;
            }
            if (Math.abs(xmax - oxmax) < eps)
```

```
            {
                break;
            }
            else
            {
                oxmax = xmax;
            }
        }
        NumberFormat nf = NumberFormat.getNumberInstance();
        nf.setMinimumFractionDigits(4);
        System.out.println(mu);
        String res[] = new String[2];
        res[0] = "";
        res[1] = "";
        for (i = 0; i < N; i++) {
            res[0] += String.valueOf(nf.format(x[i]) + " ");
        }
        res[1] = String.valueOf(nf.format(1/ xmax));
        return res;
    }

}
```

4 实验结果总结

经过试验，对于不同的初始vector(x)$，可以看到，在精度不变的情况下，对实验结果并无太大的影响。

本实验对幂法和反幂法的模拟结果较好。

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 java.util.concurrent.Flow.Subscriber;

import javax.swing.*;

import java.awt.Graphics;

public class SolveCharacteristicValue {

    static String strb;
    static ArrayList<String> strA = new ArrayList<String>();
    static double[][] a;
    static double[] b;
    static public int dim;
```

```
    static String test1a = new String("-1 2 1\n2 -4 1\n1 1 -6");
    static String test2a = new String("4 -2 7 3 -1 8\n-2 5 1 1 4 7\n7 1 7 2 3 5\n3 1 2
6 5 1\n-1 4 3 5 3 2\n8 7 5 1 2 4");
    static String test3a = new String("2 -1 0 0 0\n-1 2 -1 0 0\n0 -1 2 -1 0\n0 0 -1 2
-1\n0 0 0 -1 2");
    static String test4a = new String("2 1 3 4\n1 -3 1 5\n3 1 6 -2\n4 5 -2 -1");
    static String test5a = new String("-1 2 1\n2 -4 1\n1 1 -6");
    static double eps;
    static Fun fun = new Fun();
    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 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);//设置精度
    static JTextField jFieldResult3 = new JTextField(120);//设置初始值
    public static void main(String[] args) {
        System.out.println("Test Success!");
        SolveCharacteristicValue NI = new SolveCharacteristicValue();

        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);
        dim = (int)Math.sqrt(len);
        //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("输入需要求解方程组的A：");
        label.setBounds(20, 50, 200, 20);
        label.setForeground(Color.BLUE);
        p.add(label);

        JLabel label1 = new JLabel("当前选择的方程组解法：");
        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("对应的特征向量：");
        label7.setBounds(400, 365, 200, 20);
        label7.setForeground(Color.BLUE);
        p.add(label7);

        jFieldResult.setText("当前结果：未显示");
        jFieldResult.setEditable(false);
        jFieldResult.setBounds(400, 390, 300, 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(400, 425, 200, 20);
        label9.setForeground(Color.BLUE);
        p.add(label9);

        jFieldResultLam.setText("当前结果：未显示");
        jFieldResultLam.setEditable(false);
        jFieldResultLam.setBounds(400, 450, 300, 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("精度设置：");
        label8.setBounds(500, 20, 200, 20);
        label8.setForeground(Color.BLUE);
        p.add(label8);

        jFieldResult2.setText("0.00001");
        jFieldResult2.setEditable(true);
        jFieldResult2.setBounds(580, 18, 120, 25);
        //jFieldMode.setForeground(Color.RED);
        p.add(jFieldResult2);

        JLabel label11 = new JLabel("初始值设置：");
        label11.setBounds(500, 60, 200, 20);
        label11.setForeground(Color.BLUE);
        p.add(label11);

        jFieldResult3.setText("NULL");
        jFieldResult3.setEditable(true);
        jFieldResult3.setBounds(580, 58, 120, 25);
        //jFieldMode.setForeground(Color.RED);
        p.add(jFieldResult3);



        /**
         * 这里是对Buttons的设置
         */
        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);

        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
                for (int i = strA.size() - 1; i >= 0; i--) {
                    strA.remove(i);
                }
                // 因为是ArrayList，所以每次使用前需要清空
                processInput(jarea.getText());
                // 将jArea中的字符串处理成字符串数组
                a = new double[dim][dim];
                int cnt = 0;
                for (int i = 0; i < dim; i++) {
                    for (int j = 0; j < dim; j++) {
                        a[i][j] = Double.parseDouble(strA.get(cnt++));
                    }
                }
                double u[] = new double[dim];
```

```
                if (jFieldResult3.getText().equals("NULL"))
                {
                    String tm = new String("");
                    for (int i = 0;i < dim;i++)
                    {
                        u[i] = 1.0;
                        tm += "1.0 ";
                    }
                    jFieldResult3.setText(tm);

                }
                else
                {
                    String tm[] = jFieldResult3.getText().split(" ");
                    if (tm.length != dim)
                    {
                        String tm2 = new String("");
                        for (int i = 0;i < dim;i++)
                        {
                            u[i] = 1.0;
                            tm2 += "1.0 ";
                        }
                        jFieldResult3.setText(tm2);
                    }
                    else
                    {
                        for (int i = 0 ; i < dim;i++)
                        {
                            u[i] = Double.parseDouble(tm[i]);
                        }
                    }
                }
                eps = Double.valueOf(jFieldResult2.getText());
                fun.setData(a, dim,u,eps);
                if (FunTypeInt == 1)
                {
                    jFieldResult.setText(fun.cal()[0]);
                    jFieldResultLam.setText(fun.cal()[1]);
                }
                else if (FunTypeInt == 2)
                {
                    if (flag == 0)
                        {
                            jFieldResult.setText(fun.cal2()[0]);
                            jFieldResultLam.setText(fun.cal2()[1]);
                        }
                    else if (flag == 1)
                    {
                        jFieldResult.setText(fun.cal()[0]);
                        jFieldResultLam.setText(fun.cal()[1]);
                    }
                }

            }
        });

        JButton button2 = new JButton("测试样例1");//
        button2.setBounds(20, 320, 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
                jarea.setText(test1a);
                flag = 1;
            }
        });

        JButton button3 = new JButton("测试样例2");
        button3.setBounds(20, 360, 200, 40);
        p.add(button3);
        button3.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                jarea.setText(test2a);
                flag = 0;

            }
        });

        JButton button4 = new JButton("测试样例3");
        button4.setBounds(20, 400, 200, 40);
        p.add(button4);
        button4.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                jarea.setText(test3a);
                flag = 0;
            }
        });
        JButton button5 = new JButton("测试样例4");
        button5.setBounds(20, 440, 200, 40);
        p.add(button5);
        button5.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                jarea.setText(test4a);
                flag = 0;
            }
        });

        JButton button6 = new JButton("测试样例5");
        button6.setBounds(20, 480, 200, 40);
        p.add(button6);
        button6.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
```

```
                jarea.setText(test5a);
                flag = 1;
            }
        });
        /**
         * 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;
        JMenuItem funItem1, funItem2;
        JMenuBar menuBar = new JMenuBar();
        funItem1 = new JMenuItem("幂法");
        funItem2 = new JMenuItem("反幂法");
        Menu1 = new JMenu("求解方法选择");
        Menu1.add(funItem1);
        Menu1.add(funItem2);
        Menu1.setSelected(true);
        menuBar.add(Menu1);
        frame.setJMenuBar(menuBar);
        funItem1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // updateModeStr("lag");
                updateModeStr(1);
                System.out.println("幂法");

            }
        });
        funItem2.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // updateModeStr("newton");
                updateModeStr(2);
                System.out.println("反幂法");
            }
        });

    }

    /***
     * 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) {
            FunType = new String("幂法");
```

```
            FunTypeInt = 1;
            jFieldMode.setText(FunType);
        } else if (num == 2) {
            FunType = new String("反幂法");
            FunTypeInt = 2;
            jFieldMode.setText(FunType);
        }
    }
}

class Fun {
    public double[][] A;
    public double[] V;
    public double[] U;
    public double s,mu0,mu = -1000;
    public double eps = 0.0001;
    private int N;
    public void setData(double a[][], int dim,double u[],double e) {
        N = a.length;
        eps = e;
        // System.out.println(dimension);
        // System.out.println(123213123);
        A = new double[N + 2][N + 2];
        U = new double[N + 2];
        V = new double[N + 2];
        int i,j;
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++) {
                A[i][j] = a[i][j];
            }
        }
        for (i = 0;i < N;i++)
        {
            U[i] = u[i];
        }
    }
    /**
     * 幂法
     * @return
     */
    public String[] cal()
    {
        int i,j;
        do{
            mu0 = mu;
            for (i = 0;i < N;i++)
            {
                s = 0;
                for (j = 0;j < N;j++) s += A[i][j] * U[j];
                V[i] = s;
            }
            mu = Math.abs(V[0]);
            for (i = 0 ;i < N;i++)
            {
                if (mu < Math.abs(V[i])) mu = V[i];
            }
            for (i = 0 ; i < N;i++)
            {
```

```
                U[i] = V[i] / mu;
            }
        }while (Math.abs(mu - mu0) >= eps);
        System.out.println("///////");
        for(i = 0 ; i < N;i++)
        {
            System.out.println(U[i]);
        }
        NumberFormat nf = NumberFormat.getNumberInstance();
        nf.setMinimumFractionDigits(4);
        System.out.println(mu);
        String res[] = new String[2];
        res[0] = "";
        res[1] = "";
        for (i = 0; i < N; i++) {
            res[0] += String.valueOf(nf.format(U[i]) + " ");
        }
        res[1] = String.valueOf(nf.format(mu0));
        return res;
    }
    /**
     * 反幂法
     */
    public String[] cal2()
    {
        int n ;
        int i,j,k;
        double xmax = -9999,oxmax = 0;
        double [][]L = new double[N + 2][N + 2];
        double [][]U = new double[N + 2][N + 2];
        double []x = new double[N + 2];
        double []nx = new double[N + 2];
        for (i = 0 ; i < N;i++)
        {
            x[i] = 1;
        }
        oxmax = 0;

        for (i = 0 ; i < N;i++)
        {
            U[i][i] = 1;
        }
        for (k = 0;k < N;k++)
        {
            for (i = k;i < N ;i++)
            {
                L[i][k] = A[i][k];
                for (j = 0 ; j <= k - 1;j++)
                    L[i][k] -= (L[i][j] * U[j][k]);
            }
            for (j = k + 1;j < N;j++)
            {
                U[k][j] = A[k][j];
                for (i = 0;i <= k - 1;i++)
                {
                    U[k][j] -= (L[k][i] * U[i][j]);
                }
                U[k][j] /= L[k][k];
```

```
            }
        }
        for (i = 0;i < N;i++)
        {
            x[i] = 1;
        }
        for (i = 0;i < 100;i++)
        {
            for (j = 0;j < N;j++)
            {
                nx[j] = x[j];
                for (k = 0;k <= j - 1;k++)
                {
                    nx[j] -= L[j][k] * nx[k];
                }
                nx[j] /= L[j][j];
            }
            for (j = N - 1;j >= 0;j--)
            {
                x[j] = nx[j];
                for (k = j + 1;k < N;k++)
                {
                    x[j] -= U[j][k] * x[k];
                }
            }
            xmax = 0;
            for (j = 0;j < N;j++)
            {
                if (Math.abs(x[j]) > xmax)
                {
                    xmax = Math.abs(x[j]);
                }
            }
            for (j = 0;j < N;j++)
            {
                x[j] /= xmax;
            }
            if (Math.abs(xmax - oxmax) < eps)
            {
                break;
            }
            else
            {
                oxmax = xmax;
            }
        }
        NumberFormat nf = NumberFormat.getNumberInstance();
        nf.setMinimumFractionDigits(4);
        System.out.println(mu);
        String res[] = new String[2];
        res[0] = "";
        res[1] = "";
        for (i = 0; i < N; i++) {
            res[0] += String.valueOf(nf.format(x[i]) + " ");
        }
        res[1] = String.valueOf(nf.format(1/ xmax));
        return res;
    }
```

```
}
```