

Help in Java swings(internal Frame)

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-05/msg00717.html>

- *From:* "sweta" <05.sweta@xxxxxxxxx>
 - *Date:* 7 May 2006 23:50:20 -0700
-

I Have one Internal Frame in which one list is attached with 1 to 19 options.

my problem is that changes occurs in Internal's Frame graph with respect to only one option which we have already selected in list3.java file and in peakcontainer.java in which instance of list3 is passed but I want changes with respect to every option(1 to 19) in Internal's frame graph

How can I overcome from this problem.I Have attached all my codes NMRUI.java is main file

actually I am working in versions of Nuclear magnetic resonance(project).

```
//list3.java
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
public class list3 extends JPanel
{
public int Frame_index;
public list3(int Frame_index)
{
this.Frame_index = Frame_index;
initComponents();
}

```

```
private void initComponents() {
scrollPane1 = new JScrollPane();
list = new JList();

```

```
//===== this =====
```

```
setLayout(new FlowLayout());
```

```
//===== scrollPane1 =====
{

```

```
//---- list ----
```

```
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
list.setModel(new AbstractListModel() {
String[] values = {
```

Help in Java swings(internal Frame)

```
"1","2","3","4",
"5",
"6",
"7",
"8",
"9",
"10",
"11",
"12",
"13",
"14",
"15",
"16","17","18","19"
};
public int getSize() { return values.length; }
public Object getElementAt(int i) { return values[i]; }
});
list.setVisibleRowCount(19);
list.setSelectedIndex(Frame_index);
list.addListSelectionListener(new ListSelectionListener() {
public void valueChanged(ListSelectionEvent e) {
listValueChanged(e);
}
});
scrollPane1.setViewportView(list);
}
add(scrollPane1);
// JFormDesigner – End of component initialization
//GEN-END: initComponents
}

public void listValueChanged(ListSelectionEvent e) {
// TODO add your code here
JList jl = (JList)e.getSource();
this.Frame_index = jl.getSelectedIndex();
}

private JScrollPane scrollPane1;
private JList list;
}

//peakContainer.java/*
* Created on Aug 26, 2004
*
*/
//package nmrui.mri;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

Help in Java swings(internal Frame)

```
import javax.swing.ButtonGroup;
import javax.swing.JInternalFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPopupMenu;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JRootPane;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import java.io.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.JOptionPane;//new code for pope-up
import javax.swing.*;
import javax.swing.event.*;
/*
 * @ Author ANADI MISHRA
 *
 * @ version 1.00
 */
public class PeakContainer extends JInternalFrame
{
    JMenuBar menuBar;
    JMenu fileMenu, dispMenu, psMenu;
    JMenuItem[] fmi;
    JRadioButtonMenuItem[] mmi, tmi, psi;
    JRootPane rootPane;
    public final int inset = 50;
    ButtonGroup mbg, tbg, psbg;
    JRadioButtonMenuItem sb; //new code
    static final Integer DOCLAYER = new Integer(5);
    GraphGE gr;
    GraphDataGE1 gd;
    PeakSupres ps;
    //XAxismod ax; // for pope-up

    JPopupMenu m_menu = new JPopupMenu(); //new codes
    double max, min; // for pope-up
    double xvalue;
    boolean linePresent = false;
    boolean popupPresent = false;
    String msg=" ";
    public int frameindex = 6;
    public PeakContainer()
    {
        super();
    }
}
/*
 * @ s the name of the GE File
 * @ data the double array containing file's values
```

Help in Java swings(internal Frame)

```
*/
public PeakContainer(String s, double[][] data)
{
    super(s);
    final int point = data.length/19;
    rootPane = this.getRootPane();
    //JPopupMenu.setDefaultLightWeightPopupEnabled(false);

    menuBar = new JMenuBar();
    menuBar.setOpaque(true);

    /******* codes for pope-up
    *****/
    JMenuItem m_pv = new JMenuItem("Previous view");
    JMenuItem m_iv = new JMenuItem("Initial view");
    JMenuItem m_srz = new JMenuItem("Set zero reference");
    JMenuItem m_sr = new JMenuItem("Set reference");

    m_menu.add (m_pv);
    m_menu.add (m_iv);

    // Separator
    m_menu.add (new JMenuItem("-"));

    m_menu.add (m_srz);
    m_menu.add (m_sr);
    add(m_menu);

    addMouseListener ( new MouseAdapter ()
    {
        public void mouseReleased(MouseEvent e)
        {
            // Check for a right click pop-up
            if (e.isPopupTrigger() )
            {
                int x = e.getX();
                int y = e.getY();
                popupPresent = true;

                genppm_frq pmfr = new genppm_frq(point);
                double[] ppm = pmfr.returnppm();
                max_min mm = new max_min();
                max = mm.getMax(ppm);
                min = mm.getMin(ppm);
                xvalue = -(min+((720-(x-80))*(max-min)/720));
                // Show the popup meun
                m_menu.show (rootPane, x, y );
            }
        }
    });
    m_sr.addActionListener(new PopupMenuListener(this));
```

Help in Java swings(internal Frame)

```
/** *********************************************************************
```

```
fileMenu = new JMenu("File");
fileMenu.setMnemonic('F');

fmi = new JMenuItem[3];

fmi[0] =
new JMenuItem(
"Open",
(Icon) new ImageIcon("./resources/images/open.gif"));
fmi[1] =
new JMenuItem(
"Save",
(Icon) new ImageIcon("./resources/images/save.gif"));
fmi[2] = new JMenuItem("Exit");

for (int i = 0; i < 3; i++)
fileMenu.add(fmi[i]);

for (int i = 0; i < 3; i++)
fmi[i].addActionListener(new GraphContainerListener(this));

dispMenu = new JMenu("Display");
dispMenu.setMnemonic('D');

mbg = new ButtonGroup();
tbg = new ButtonGroup();

mmi = new JRadioButtonMenuItem[2];
tmi = new JRadioButtonMenuItem[7];

mmi[0] = new JRadioButtonMenuItem("FID");
mmi[0].setActionCommand("fid");
mmi[1] = new JRadioButtonMenuItem("Spectrum",true);
mmi[1].setActionCommand("sp");
tmi[0] = new JRadioButtonMenuItem("Real", true);
tmi[0].setActionCommand("r");
tmi[1] = new JRadioButtonMenuItem("Imaginary");
tmi[1].setActionCommand("i");
tmi[2] = new JRadioButtonMenuItem("Absolute");
tmi[2].setActionCommand("a");
tmi[3] = new JRadioButtonMenuItem("Real^2");
tmi[3].setActionCommand("r2");
tmi[4] = new JRadioButtonMenuItem("Imaginary^2");
tmi[4].setActionCommand("i2");
tmi[5] = new JRadioButtonMenuItem("Absolute^2");
tmi[5].setActionCommand("a2");
tmi[6] = new JRadioButtonMenuItem("Mag[Mag-real]");
tmi[6].setActionCommand("msr");
```

Help in Java swings(internal Frame)

```
//MyListSelectionListener l2=new MyListSelectionListener();
//JListSimpleExample1 l=new JListSimpleExample1();

for (int i = 0; i < 2; i++)
{
dispMenu.add(mmi[i]);
mbg.add(mmi[i]);
}

for (int i = 0; i < 2; i++)
mmi[i].addActionListener(new GraphContainerListener(this));

dispMenu.addSeparator();

for (int i = 0; i < 7; i++)
{
dispMenu.add(tmi[i]);
tbg.add(tmi[i]);
}

for (int i = 0; i < 7; i++)
{
tmi[i].addActionListener(new GraphContainerListener(this));

//new code
//tmi[i].addMouseListener(new handelmouse(this));
//tmi[i].addMouseMotionListener(new handelmouse(this));
}

////###////////////////////////////////////
psMenu = new JMenu("PeakSup");
psMenu.setMnemonic('P');

psbg = new ButtonGroup();
psi = new JRadioButtonMenuItem[2];

psi[0] = new JRadioButtonMenuItem("Water Peak");
psi[0].setActionCommand("wps");
psi[1] = new JRadioButtonMenuItem("Viewps");
psi[1].setActionCommand("vps");

for (int i = 0; i < 2; i++)
{
psMenu.add(psi[i]);
psbg.add(psi[i]);
}

for (int i = 0; i < 2; i++)
psi[i].addActionListener(new GraphContainerListener(this));
```

Help in Java swings(internal Frame)

```
////###////////////////////////////////////  
  
//new code  
sb = new JRadioButtonMenuItem("SaveData", (Icon) new  
ImageIcon("./resources/images/save.gif"));  
sb.setActionCommand("savedata");  
sb.addActionListener(new GraphContainerListener(this));  
  
menuBar.add(fileMenu);  
menuBar.add(dispatchMenu);  
menuBar.add(psMenu);  
menuBar.add(sb); //new code  
  
rootPane.setJMenuBar(menuBar);  
setClosable(true);  
setIconifiable(true);  
setMaximizable(false);  
setResizable(false);  
setVisible(true);  
this.setBounds(10,10,600,400);  
  
/*  
 * Adding the default Graph  
 */  
list3 l3 = new list3(frameindex);  
rootPane.getContentPane().add(l3, BorderLayout.EAST);  
  
gd = new GraphDataGE1(data.length);  
  
if (data != null)  
{  
 //choicetest t=new choicetest();  
 //msg+=t.cbg.getSelectedCheckbox().getLabel();  
  
 //System.out.println("index @nkur:"+l1.getIndex());  
 //System.out.println(l3.index);  
 gd.addSignals(data,frameindex);  
 gd.doPhaseFID();  
 gd.doFFT();  
 gd.doPhase();  
 }  
 else  
 {  
 System.out.println("Cannot Proceed\n System Shall Exit");  
 System.exit(0);  
 }  
  
gr = new GraphGE(gd.getPoint(), s);  
gr.clearValues();
```

Help in Java swings(internal Frame)

```
gr.setValues(gd.getReal());
gr.reDraw();

rootPane.getContentPane().add(gr, BorderLayout.CENTER);
//ax = new XAxismod(gd.getPoint())

}

public GraphGE getGraphGE()
{
return this.gr;
}

public GraphDataGE1 getGraphDataGE()
{
return this.gd;
}

/*public XAxismod gettrans()
{
return this.ax; //pope-up
}*/

/*public PeakSupres getPeakSupres()
{
return this.ps;
}*/

public void message()
{
System.out.println("Hello!");
}

public PeakContainer getPeakContainer()
{
return this;
}

public int[] getSelectedMenu()
{
int [] idx = new int[4];

if(mmi[0].isSelected())
idx[0] = 0;
else
idx[0] = 1;

for(int i=0;i<7;i++)
{
if(tmi[i].isSelected())
{
```

```

idx[1] = i;
}
}

if(sb.isSelected())
idx[2] = 0;

for(int i=0;i<2;i++)
{
if(psi[i].isSelected())
{
idx[3] = i;
}
}
return idx;
}

}

#####new code for
popup
class PopupMenuListener implements ActionListener
{
PeakContainer cp;
int [] idx = new int[2];
public PopupMenuListener(PeakContainer cp)
{
this.cp = cp;
}

public void actionPerformed(ActionEvent e)
{
GraphGE grp = cp.getGraphGE();
GraphDataGE1 gdp = cp.getGraphDataGE();
int pt = gdp.getPoint();

if(cp.popupPresent)
{
String Input = JOptionPane.showInputDialog("Enter first integer");
double inputnumber = Double.parseDouble(Input);
double d = cp.xvalue;
System.out.println(inputnumber);
System.out.println(d);
double numsub = inputnumber - d;
double numsubfrq = (inputnumber * 63.86) - d;

ppm_freq_save pfs = new ppm_freq_save(pt, numsub,
numsubfrq);
pfs.writefrq();

idx = cp.getSelectedMenu();

```

Help in Java swings(internal Frame)

```
//System.out.println("Pope-up present"+idx[0]);

if(idx[0] == 0)
{
int ch = idx[1];
switch (ch)
{
case 0: // Real
grp.clearValues();
grp.setValues(gdp.getRealFID());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 1: //Imag
grp.clearValues();
grp.setValues(gdp.getImagFID());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 2: //Absolute
grp.clearValues();
grp.setValues(gdp.getMagnitudeFID());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 3: //Real Square
grp.clearValues();
grp.setValues(gdp.getRealFIDSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 4 : // Imag Square
grp.clearValues();
grp.setValues(gdp.getImagFIDSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 5:
grp.clearValues();
grp.setValues(gdp.getMagnitudeFIDSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
}
}
```

```

else if(idx[0] == 1)
{
int ch = idx[1];

switch (ch)
{
case 0: // Real
grp.clearValues();
grp.setValues(gdp.getReal());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 1: //Imag
grp.clearValues();
grp.setValues(gdp.getImag());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 2: //Absolute
grp.clearValues();
grp.setValues(gdp.getMagnitude());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 3: //Real Square
grp.clearValues();
grp.setValues(gdp.getRealSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 4 : // Imag Square
grp.clearValues();
grp.setValues(gdp.getImagSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
case 5:
grp.clearValues();
grp.setValues(gdp.getMagnitudeSq());
grp.settrans(numsub);
cp.repaint();
grp.reDraw();
break;
}
}
}

```

```
}  
  
}  
#####  
  
class GraphContainerListener implements ActionListener  
{  
    PeakContainer cg;  
    String str = new String("");  
    public GraphContainerListener(PeakContainer cg)  
    {  
        this.cg = cg;  
    }  
  
    public String getActionString()  
    {  
        return str;  
    }  
  
    public void actionPerformed(ActionEvent evt)  
    {  
        str = evt.getActionCommand();  
        GraphGE gr = cg.getGraphGE();  
        GraphDataGE1 d = cg.getGraphDataGE();  
        //PeakSupres s = cg.getPeakSupres();  
  
        fileHandling fw = new fileHandling();  
        File fp = new File(".");  
        String savepath = fp.getAbsolutePath();  
        String foldername = "\\Result\\";  
  
        if (str.equals("fid"))  
        {  
            return;  
        }  
        else if (str.equals("sp"))  
        {  
            return;  
        }  
        if (str.equals("r"))  
        {  
            gr.clearValues();  
            if(cg.mmi[1].isSelected())  
            gr.setValues(d.getReal());  
            else  
            gr.setValues(d.getRealFID());  
            cg.repaint();  
            gr.reDraw();  
        }  
    }  
}
```

```

}
else if (str.equals("i"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getImag());
else
gr.setValues(d.getImagFID());
cg.repaint();
gr.reDraw();
}
else if (str.equals("a"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getMagnitude());
else
gr.setValues(d.getMagnitudeFID());
cg.repaint();
gr.reDraw();
}
else if (str.equals("r2"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getRealSq());
else
gr.setValues(d.getRealFIDSq());
cg.repaint();
gr.reDraw();
}
else if (str.equals("i2"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getImagSq());
else
gr.setValues(d.getImagFIDSq());
cg.repaint();
gr.reDraw();
}
else if (str.equals("a2"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getMagnitudeSq());
else
gr.setValues(d.getMagnitudeFIDSq());
cg.repaint();
gr.reDraw();
}

```

```

else if (str.equals("msr"))
{
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(d.getMagofMagSubRe());
else
gr.setValues(d.getMagofMagSubReFID());
cg.repaint();
gr.reDraw();
}

////###////////////////////////////////////
else if (str.equals("wps"))
{

PeakSupres s = new PeakSupres(d.getReal(),d.getImag());
double[] pcreal = s.getpsreal();

//System.out.println(pcreal[0]);
gr.clearValues();
if(cg.mmi[1].isSelected())
gr.setValues(pcreal);
//System.out.println("Error1");
else
System.out.println("Error2");
cg.repaint();
gr.reDraw();
}

else if (str.equals("vps"))
{
Frame_ParalleDisplay pf = new Frame_ParalleDisplay(d.getReal());
//Frame f = new Frame_ParalleDisplay();
System.out.println("ViewWP");
}

else if (str.equals("savedata"))
{
/*try
{
//File fp = new File(".");
//String savepath = fp.getAbsolutePath();
String foldername1 = "\\BMP_Graph\\";
File file = new File(savepath+foldername1);
file.mkdir();
File filef = new File(savepath+foldername1+"Mag.bmp");
Filesave sf = new Filesave();
sf.saveToFile(cg, filef);
System.out.println("Done saving File!");
}
}

```

```

}
catch(Exception e)
{ }*/

if(cg.sb.isSelected())
{
double[] pcreal = d.getReal();
double[] pcimag = d.getImag();
double[] pcrealtd = d.getRealFID();
double[] pcimagtd = d.getImagFID();
fw.filewrite(savepath, pcreal,"pcreal",foldername);
fw.filewrite(savepath, pcimag,"pcimag",foldername);
fw.filewrite(savepath, pcrealtd,"pcrealtd",foldername);
fw.filewrite(savepath, pcimagtd,"pcimagtd",foldername);
System.out.println("Done saving File!");
}
}
}
}

```

//graphDataGe1.java

```

/*
 * Created on Aug 26, 2004
 *
 */
//package nmrui.mri;

import java.util.Vector;
//import java.io.*;
import fft.*;
/*
 * @author ANADI MISHRA
 *
 * @version 1.00
 */

```

```

public class GraphDataGE1
{
double[] real;
double[] imag;
double[] real_fd;
double[] imag_fd;
double[] realTD;
double[] imagTD;
double [][] signal;
double phi;
int originalLength = 0;
Vector setOfFIDs,setOFSpectra;
////////###////////////////////////////////////
double a;

```

```

double dw;
double d;
///////#####
public GraphDataGE1(int size)
{
originalLength = size/19;
//originalLength = size/11;
System.out.println("originalLength"+originalLength);
//originalLength = size;
real = new double[originalLength];
imag = new double[originalLength];
real_fd = new double[originalLength];
imag_fd = new double[originalLength];
realTD = new double[originalLength];
imagTD = new double[originalLength];
signal = new double[originalLength][2];
phi=0.0D;
///////#####
a=0.0D;
dw = 0.4;
d = 0;
///////#####
setOfFIDs = new Vector();
setOfSpectra = new Vector();
}

/**
 * @param s
 */
/*public void addSignals(double[][] s)
{
for(int i=0;i<originalLength;i++)
{
signal[i][0] = s[i][0];
signal[i][1] = s[i][1];
}
}*/

public void addSignals(double[][] s,int frameindex)
{
//int ctr = 0;
//int ctr = originalLength;
int ctr = (frameindex-1)*originalLength;
//int ctr = 4*originalLength;
for(int k=0;k<1;k++)
//for(int k=0;k<1;k++)
//for(int k=1;k<3;k++)
for (int l = 0; l < originalLength; l++)
{
signal[l][0] += s[ctr][0];
signal[l][1] += s[ctr][1];
}
}

```

```

ctr++;
}

}

public void doPhaseFID()
{
for(int i=0; i<originalLength; i++)
{
realTD[i]= (Math.cos(Math.toRadians(phi))) * signal[i][0] +
(Math.sin(Math.toRadians(phi))) * signal[i][1];
imagTD[i]= -(Math.sin(Math.toRadians(phi))) * signal[i][0] +
(Math.cos(Math.toRadians(phi))) * signal[i][1];
}
}

#####
/*public void doApodizeFID()
{
for(int i=0; i<originalLength; i++)
{

realTD[i]= (Math.exp(-a*i*dw)) * realTD[i];
imagTD[i]= (Math.exp(-a*i*dw)) * imagTD[i];
}
}*/

#####

public void doFFT()
{
Fft fft = new Fft();
double [][] input_fft = new double[2][originalLength];
for(int i=0; i<input_fft[0].length; i++)
{
input_fft[0][i] = signal[i][0];
input_fft[1][i] = signal[i][1];
}

double [][] estimate = fft.direct_1D_Estim_fft(input_fft);
//double [][] output_fft = fft.calcFFTD(estimate,true);
for (int i = 0; i < originalLength; i++)
{
real_fd[i] = estimate[0][i];
imag_fd[i] = estimate[1][i];
}
}

public int getPoint()
{
return originalLength;
}

```

```

}
#####
public double geta()
{
return a;
}

public void seta(double a)
{
this.a = a;
}

public void doApodize()
{
Fft fft = new Fft();
double [][] input_fft_ap = new double[2][originalLength];
for(int i=0;i<input_fft_ap[0].length;i++)
{
d = 1/(a * dw);
//double bell = (d*d)/(((Math.PI)*i*i) + d*d); // For Lorentzian
//double bell = Math.exp((-i*i)/(2*d*d)); // For Gaussian
double bell = Math.exp((i*i)/(2*d*d));
//input_fft_ap[0][i] =(Math.exp(-a*i*dw))*realTD[i];
//input_fft_ap[1][i] = (Math.exp(-a*i*dw))*imagTD[i];
input_fft_ap[0][i] = bell*realTD[i];
input_fft_ap[1][i] = bell*imagTD[i];
}
double [][] estimate_ap =
fft.direct_1D_Estim_fft(input_fft_ap);
for (int i = 0; i < originalLength; i++)
{
real[i] = estimate_ap[0][i];
imag[i] = estimate_ap[1][i];
}
}

#####
public double getPhi()
{
return phi;
}

public void setPhi(double phi)
{
this.phi = phi;
}

public void doPhase()
{
for(int i=0; i<originalLength; i++)

```

```

{
real[i]= (Math.cos(Math.toRadians(phi))) * real_fd[i] +
(Math.sin(Math.toRadians(phi))) * imag_fd[i];
imag[i]= -(Math.sin(Math.toRadians(phi))) * real_fd[i] +
(Math.cos(Math.toRadians(phi))) * imag_fd[i];
}
}

```

```

public double[][] getSignal()
{
return signal;
}

```

```

public double[] getReal()
{
return real;
}

```

```

public double[] getRealFID()
{
return realTD;
}

```

```

public double[] getImag()
{
return imag;
}

```

```

public double[] getImagFID()
{
return imagTD;
}

```

```

public int getoriginalLength()
{
return originalLength;
}

```

```

public double[] getRealSq()
{
double[] sq = new double[real.length];

for (int i = 0; i < real.length; i++)
sq[i] = Math.pow(real[i], 2.0);

return sq;
}

```

```

public double[] getRealFIDSq()
{
double[] sq = new double[realTD.length];

```

```
for (int i = 0; i < realTD.length; i++)
sq[i] = Math.pow(realTD[i], 2.0);
```

```
return sq;
}
```

```
public double[] getImagSq()
{
double[] sq = new double[imag.length];
```

```
for (int i = 0; i < imag.length; i++)
sq[i] = Math.pow(imag[i], 2.0);
```

```
return sq;
}
```

```
public double[] getImagFIDSq()
{
double[] sq = new double[imagTD.length];
```

```
for (int i = 0; i < imagTD.length; i++)
sq[i] = Math.pow(imagTD[i], 2.0);
```

```
return sq;
}
```

```
public double[] getMagnitude()
{
double[] mag = new double[originalLength];
for (int i = 0; i < originalLength; i++)
mag[i] = Math.sqrt((Math.pow(real[i], 2.0) + Math.pow(imag[i],
2.0)));
```

```
return mag;
}
```

```
public double[] getMagnitudeFID()
{
double[] mag = new double[originalLength];
for (int i = 0; i < originalLength; i++)
mag[i] = Math.sqrt((Math.pow(realTD[i], 2.0) + Math.pow(imagTD[i],
2.0)));
```

```
return mag;
}
```

```
public double[] getMagnitudeSq()
{
double[] mag = getMagnitude();
double[] magsq = new double[originalLength];
```

Help in Java swings(internal Frame)

```
for (int i = 0; i < originalLength; i++)
{
magsq[i] = Math.pow(mag[i], 2.0);
}

return magsq;
}

public double[] getMagnitudeFIDSq()
{
double[] mag = getMagnitudeFID();
double[] magsq = new double[originalLength];
for (int i = 0; i < originalLength; i++) {
magsq[i] = Math.pow(mag[i], 2.0);
}

return magsq;
}

public double[] getMagofMagSubRe()
{
double[] magmsr = new double[originalLength];
double[] mag = new double[originalLength];
for (int i = 0; i < originalLength; i++)
{
mag[i] = Math.sqrt((Math.pow(real[i], 2.0) + Math.pow(imag[i],
2.0)));
magmsr[i] = Math.sqrt(Math.pow((mag[i] - real[i]), 2.0));
}

return magmsr;
}

public double[] getMagofMagSubReFID()
{
double[] magmsr = new double[originalLength];
double[] mag = new double[originalLength];
for (int i = 0; i < originalLength; i++)
{
mag[i] = Math.sqrt((Math.pow(realTD[i], 2.0) + Math.pow(imagTD[i],
2.0)));
magmsr[i] = Math.sqrt(Math.pow((mag[i] - real[i]), 2.0));
}

return magmsr;
}

}/*
* Created on Aug 26, 2004
*
```

Help in Java swings(internal Frame)

```
*/  
//package nmrui.mri;  
/*  
* @ Author Manoj Sarma  
*  
* @version 1.00  
*/  
  
import javax.swing.JPanel;  
import java.awt.Color;  
import java.awt.Graphics;  
import java.awt.event.*;  
import java.io.*;  
  
public class GraphGE extends JPanel  
{  
// handelmouse hm;  
//int mouseX, mouseY;  
//String msg;  
int point = 0;  
double numsub = 0D;  
double xmax, ymax, xmin, ymin;  
  
double [] yVal;  
double[] yVal1;  
int[] yVal2;  
int[] yVal3;  
  
double[] xVal;  
int[] xVal1;  
int[] xVal2;  
  
String name;  
//XAxis xaxis;  
XAxismod xaxis;  
//YAxis yaxis;  
YAxismod yaxis;  
  
////###////////////////////////////////////  
  
boolean tracking = false;  
boolean mouseresele = false;  
int startX = 0;  
int startY = 0;  
int currentX = 0;  
int currentY = 0;  
////###////////////////////////////////////  
  
/*  
* @param pt gives the length of the array  
* @param s gives the name of the GE File
```

Help in Java swings(internal Frame)

```
*/
GraphGE(int pt, String s)
{
point = pt;
System.out.println("Point"+point);
name = s;
yVal = new double[point];
yVal1 = new double[point];
yVal2 = new int[point];
yVal3 = new int[point];
xVal = new double[point];
xVal1 = new int[point];
xVal2 = new int[point];

xmax = ymax = xmin = ymin = 0.0;

setBackground(new Color(255, 255, 255));
setForeground(new Color(0, 0, 0));
setOpaque(true);

/*###//////////////////////////////////////
this.addMouseListener (new mousePressHandler());
this.addMouseListener (new mouseReleaseHandler());
this.addMouseMotionListener (new mouseMotionHandler());

//###////////////////////////////////////*/
}

/*###//////////////////////////////////////
void mouseMotion(int x, int y)
{
if(tracking)
{
currentX = x;
currentY = y;
}
requestFocus();
repaint();
}

void startMotion(int x, int y)
{
tracking = true;
startX = x;
startY = y;
currentX = x+4;
currentY = y+4; // nonzero size, may choose to ignore later
requestFocus();
repaint();
}
```

```

void stopMotion(int x, int y)
{
tracking = false; // no more rubber_rect
// save final figure data for 'display' to draw
currentX = x;
currentY = y;
requestFocus();
repaint();
}

class mousePressHandler extends MouseAdapter
{
public void mousePressed (MouseEvent e)
{
int b, x, y;
b = e.getButton();
x = e.getX();
y = e.getY();
System.out.println("press x="+x+" y="+y+" b="+b); // debug
print
if(b==1) startMotion(x, y);
}
}

class mouseReleaseHandler extends MouseAdapter
{
public void mouseReleased (MouseEvent e)
{
int b, x, y;
b = e.getButton();
x = e.getX();
y = e.getY();
System.out.println("release x="+x+" y="+y+" b="+b); //
debug print
System.out.println("Mouse Starting:"+startX-80));
System.out.println("Mouse Ending:"+x-80));
int sp = (int)((2047/900)*(startX-80));
int ep = (int)((2047/900)*(x-80)); // Check these instead of
2047 put point
//scx = new double[(ep-sp)+1];
//scy = new double[(ep-sp)+1];
//System.out.println(scx.length);
for(int i= sp; i<=ep; i++)
{
//scx[i-sp] = ppm[i];
//scy[i-sp] = real[i];
//System.out.println(scx[i-sp]);
}

if(b==1) stopMotion(x, y);
}
}

```

Help in Java swings(internal Frame)

```
mouserelease = true;
}
}
```

```
class mouseMotionHandler extends MouseMotionAdapter
{
public void mouseDragged (MouseEvent e)
{
int b, x, y;
b = e.getButton();
x = e.getX();
y = e.getY();
System.out.println("motion x="+x+" y="+y+" b="+b); // debug
print
mouseMotion(x, y);
}
}
```

```
void rubberRect(Graphics g, int x0, int y0, int x1 , int y1)
{
// can apply to all figures
// draw a rubber rectangle, mouse down, tracks mouse
int x,y,x2,y2,x3,y3; // local coordinates
x2=x0;
x3=x1;
if(x1<x0) {x2=x1; x3=x0;};
y2=y0;
y3=y1;
if(y1<y0) {y2=y1; y3=y0;};
g.setColor(Color.black);
for(x=x2; x<x3-3; x=x+8)
{
g.drawLine(x, y0, x+4, y0);
g.drawLine(x, y1, x+4, y1);
}
for(y=y2; y<y3-3; y=y+8)
{
g.drawLine(x0, y, x0, y+4);
g.drawLine(x1, y, x1, y+4);
}
}
}
#####*/
```

```
public void paint(Graphics g)
{
g.setColor(Color.black);
g.drawLine(80, 440, 800, 440);
g.drawLine(80, 35, 80, 440);
//new code
```

Help in Java swings(internal Frame)

```
//yaxis = new YAxis(this);
//yaxis.displayY(g);
xaxis = new XAxismod(point, numsub);
yaxis = new YAxismod(ymax, ymin);

//x-axis scale plotting
xaxis.displayppm(g);

//y-axis scale plotting
yaxis.displayY(g);

g.setColor(Color.blue);
//if(tracking) rubberRect(g, startX, startY, currentX,
currentY);
for (int p = 0; p <(point-1); p++)
{
g.drawLine((xVal2[p]),yVal3[p],(xVal2[p + 1]),yVal3[p + 1]);
}
}

public void update(Graphics g)
{
paintImmediately(0,0,800,500);
paint(g);
}
/*
* @param g The current graphics object context
*/
private void drawMargins(Graphics g)
{
}
/*
* @param g the current graphics object context
*/
private void drawValues(Graphics g)
{
}

public void reDraw()
{
MinMax();
scale();
repaint();
}

public void clearValues()
{

for(int x = 0; x < point ; x++)
{
yVal1[x] = 0;
}
```

```

yVal2[x] = 0;
yVal3[x] = 0;
xVal[x] = 0.0;
xVal1[x] = 0;
xVal2[x] = 0;
}
ymax = ymin = 0;
xmax = xmin = 0.0;
}
/*
 * @param arr the double array whose value is to be stored for y –
axis
 */
public void setValues(double[] arr)
{
for(int i=0;i<point;i++)
yVal[i] = arr[i];

for (int x = 0; x < point; x++)
{
xVal[x] = (double)x;
}

for (int l = 0; l < point; l++)
{
yVal1[l] = arr[l];
}

}

public double gettrans()
{
return numsub;
}
public void settrans(double numsub)
{
this.numsub = numsub;
//numb = numsub;
}

/*
 * @ Provides with the minimum and maximum values
 */
private void MinMax()
{
ymin = ymax = yVal1[0];
for (int i = 0; i < point; i++)
{
if (yVal1[i] > ymax)
ymax = yVal1[i];
if (yVal1[i] < ymin)

```

Help in Java swings(internal Frame)

```
ymin = yVal1[i];

xmin = 0.0;
xmax = 2047.0;
}

}
/*
 * @ scales the array to screen size
 */
private void scale() {
for (int k = 0; k < point; k++) {
yVal2[k] = (int) (((yVal1[k] - ymin) / (ymax - ymin)) * 400);
xVal1[k] = (int) (((xVal[k] - xmin) / (xmax - xmin)) * 720);
}

for (int l = 0; l < point; l++)
{
yVal3[l] = 440 - yVal2[l];
xVal2[l] = 80 + xVal1[l];
}

}

public double[] getMarginY()
{
return yVal;
}
//new code
/* public void start()
{
}
class handelmouse extends MouseAdapter implements MouseMotionListener
{
public void mouseMoved(MouseEvent me)
{
mouseX = me.getX();
mouseY = me.getY();
repaint();
reDraw();
}
public void mouseDragged(MouseEvent me)
{
}
public void mousePressed(MouseEvent me)
{

}
}*/import java.awt.image.*;
import java.awt.*;
import java.util.*;
```

```

import java.io.*;

public class PeakSupres
{
    GraphDataGE gd;
    int totalpoint = 2048;
    int point = 2048;
    //int point = 300, p =750;
    int pointarpk = point;
    //int pointarpk = 3;
    int iter = 15;
    int p = 0;
    int l= 0;
    int countloop = 0;
    double[] iterreal;
    double[] iterimag;
    double[] ppm;
    double ampl, dc, fr, ph;

    public PeakSupres(double[] realpart, double[] imagpart)
    {
        File fp = new File(".");
        String savepath = fp.getAbsolutePath();
        String foldername = "\\Result\\";
        try
        {
            PrintStream ps1 = new PrintStream(new
            FileOutputStream(savepath+foldername+"\\Out.log", true) ,true);
            System.setOut(ps1);
            PrintStream ps2 = new PrintStream(new
            FileOutputStream(savepath+foldername+"\\Error.txt", true) ,true);
            //Out.Err
            System.setErr(ps2);
        }
        catch(IOException e){};

        iterreal = new double[totalpoint];
        iterimag = new double[totalpoint];
        double x1=0, x2=0, x3=0, x4=0;
        double max, min, max1, min1, max2, min2;
        double[] real = new double[point];
        double[] imag = new double[point];
        double[] imag1 = new double[pointarpk];
        double[] R = new double[pointarpk];
        double[] Rdesh = new double[pointarpk];
        double[] real1 = new double[pointarpk];
        double[] qt = new double[pointarpk];
        double[] ee = new double[pointarpk];
        double[] f = new double[pointarpk];
        double[] g = new double[pointarpk];
        double[] h= new double[pointarpk];
    }
}

```

```
double[] realdiffsq = new double[pointarpk];
double[] realdiffsqsum = new double[iter];
double[] realsq = new double[pointarpk];

double[] itRp = new double[totalpoint];
double[] itIp = new double[totalpoint];
double[] itratpeaks = new double[totalpoint];
double[] itiatpeaks = new double[totalpoint];

double[] I = new double[pointarpk];
double[] Idesh = new double[pointarpk];
double[] imag2 = new double[point];
double[] imag3 = new double[point];
double[] imagdiffsq = new double[pointarpk];
double[] realimagdiffsqadd = new double[pointarpk];
double[] imagdiff = new double[pointarpk];
double[] imagsq = new double[pointarpk];
double[] imagdiffsqsum = new double[iter];

double[] edevwamp = new double[pointarpk];
double[] edevwdc = new double[pointarpk];
double[] edevwfreq = new double[pointarpk];
double[] edevwphase = new double[pointarpk];
double[] fdevwamp = new double[pointarpk];
double[] fdevwdc = new double[pointarpk];
double[] fdevwfreq = new double[pointarpk];
double[] fdevwphase = new double[pointarpk];
double[] gdevwamp = new double[pointarpk];
double[] gdevwdc = new double[pointarpk];
double[] gdevwfreq = new double[pointarpk];
double[] gdevwphase = new double[pointarpk];
double[] hdevwamp = new double[pointarpk];
double[] hdevwdc = new double[pointarpk];
double[] hdevwfreq = new double[pointarpk];
double[] hdevwphase = new double[pointarpk];

double[] amplp = new double[iter+1];
double[] dcp = new double[iter+1];
double[] frdp = new double[iter+1];
double[] phasp = new double[iter+1];

double[] Fdesh = new double[iter];
double[] FuncE = new double[iter];
double[] FuncF = new double[iter];
double[] FuncG = new double[iter];
double[] FuncH = new double[iter];
double[] deltaamp = new double[iter];
double[] deltadcp = new double[iter];
double[] deltafrd = new double[iter];
double[] deltaphas = new double[iter];
```

Help in Java swings(internal Frame)

```
double[] DevFuncE = new double[4];
double[] DevFuncF = new double[4];
double[] DevFuncG = new double[4];
double[] DevFuncH = new double[4];
double[][] J = new double[4][4];
double[][] Func = new double[4][1];
double[][] b = new double[4][1];

double[] Fdeshexp = new double[iter+1];
double[] ppm = new double[totalpoint];
double[] freq = new double[totalpoint];

double[] output = new double[4];
double[] realdata = new double[realpart.length];
double[] imagdata = new double[realpart.length];

/*if (data != null)
{
gd.addSignals(data);
gd.doPhaseFID();
gd.doFFT();
gd.doPhase();
}
else
{
System.out.println("Cannot Proceed\n System Shall Exit");
System.exit(0);
}*/

/*public GraphDataGE getGraphDataGE()
{
return this.gd;
}*/

//double[] realdata = gd.getReal();
//double[] imagdata = gd.getImag();
for(int i = 0 ; i < realpart.length; i++)
{
realdata[i] = realpart[i];
imagdata[i] = imagpart[i];
}

File fp1 = new File(".");
String path = fp1.getAbsolutePath();

mainfileread mffr = new mainfileread("C:\\Documents and
Settings\\manoj\\Desktop\\ApodizeMRS\\Result\\nfreq.txt");
freq = mffr.returnarray();
```

```
mainfileread mfp = new mainfileread("C:\\Documents and
Settings\\manoj\\Desktop\\ApodizeMRS\\Result\\nppm.txt");
ppm = mfp.returnarray();
```

```
for(int i = 0 ; i <point; i++)
{
real[i] = realdata[i+p];
imag[i] = imagdata[i+p];
}
for(int i = 0 ; i <point; i++)
{
real[i] = realdata[i+p];
imag[i] = imagdata[i+p];
}
```

```
max_min mm = new max_min();
max = mm.getMax(real);
min = mm.getMin(real);
```

```
int count=0;
int w=0, w1=0, w2;
double frd=0.0, length;
for(int i = 0 ; i < point; i++)
{
if(real[i] == max)
{
w= count+p;
System.out.println(" array no" + " " +w);
System.out.println("imaginary" + " " + imag[count]);
frd = freq[count+p];
System.out.println("Height" + " " + max);
System.out.println("Frequency" + " " + frd);
}
count++;
}
```

```
ArrayList X = new ArrayList();
ArrayList Y = new ArrayList();
System.out.println("");
double frahh=0.0, frahh1=0.0, frahh2=0.0;
```

```
for(int i = 0 ; i <point-1; i++)
{
if(((real[i] - (max/2))>0) && ((real[i+1] - (max/2))<0))
{
X.add(new Integer(Math.abs(w-(i+p))));
}
```

```
if(((real[i] - (max/2))<0) && ((real[i+1] - (max/2))>0))
{
```

```

Y.add(new Integer(Math.abs(w-(i+p))));
}
}
//*****
Object[] X1 = X.toArray();
int min3 =((Integer) X1[0]).intValue();
for(int i = 0 ; i <X1.length;i++)
{
if( ((Integer) X1[i]).intValue()<min3)
min3 = ((Integer) X1[i]).intValue();
}

//*****
int r = X1.length;
double frahh5 = 0.0;
for(int i = 0 ; i <point-1; i++)
{
if(((real[i] - (max/2D))>0) && ((real[i+1] - (max/2D))<0) &&
(Math.abs(w-(i+p))) == min3)
{
w1 = i+p;
frahh5 = (freq[i+p] + freq[i+p+1])/2D;
System.out.println(" array no1" + " " + w1 + " " + " array
no2" + " " + (w1+1));
}
}

//*****
Object[] Y1 = Y.toArray();
int min4 =((Integer) Y1[0]).intValue();
for(int i = 0 ; i <Y1.length;i++)
{
if( ((Integer) Y1[i]).intValue()<min4)
min4 = ((Integer) Y1[i]).intValue();
}
//*****
int r1 = Y1.length;
double frahh6 = 0.0;
for(int i = 0 ; i <point-1; i++)
{
if(((real[i] - (max/2))<0) && ((real[i+1] - (max/2))>0) &&
(Math.abs(w-(i+p))) == min4)
{
w2 = i+p;
frahh6 = (freq[i+p] + freq[i+p+1])/2D;
System.out.println(" array no1" + " " + w2 + " " + " array
no2" + " " + (w2 +1));
}
}

//*****

```

Help in Java swings(internal Frame)

```
double wabh = Math.abs(frahh5 - frahh6);
System.out.println("FWHM" + " " + wabh);
length = max;
System.out.println("length" + " " + length);
```

```
double dc = wabh * (Math.PI) ;
double ampl = length * dc;
System.out.println("DecayConst" + " " + dc);
System.out.println("Amplitude" + " " + ampl);
System.out.println("Frequency" + " " + frd);
```

```
amplp[0] = ampl;
dcp[0] = dc;
//frdp[0] = -frd;
frdp[0] = frd;
//phasp[0] = Math.toRadians(90.0);
phasp[0] = 0.0;
```

```
for(int i=0; i<iter ; i++ )
{
Fdesh[i]= 0.0;
FuncE[i] = 0.0;
for(int m=0; m<4; m++)
{
DevFuncE[m] = 0.0;
}
}
```

```
FuncF[i] = 0.0;
for(int m=0; m<4; m++)
{
DevFuncF[m] = 0.0;
}
}
```

```
FuncG[i] = 0.0;
for(int m=0; m<4; m++)
{
DevFuncG[m] = 0.0;
}
}
```

```
FuncH[i] = 0.0;
for(int m=0; m<4; m++)
{
DevFuncH[m] = 0.0;
}
}
```

```
for(int k=0; k< pointarpk; k++)
{
//int j = k + w-p-24;
int j = k ;
//int j = k + w-p-((pointarpk-1)/2);
}
```

```

real1[k] = real[j+1];
realsq[k] = real[j+1]*real[j+1];
imag1[k] = imag[j+1];
imagsq[k] = imag[j+1] * imag[j+1];
double
devRwrtamp=0.0,devRwrtdc=0.0,devRwrtfreq=0.0,devIwrtamp=0.0,devIwrtdc=0.0,devIwrtfreq=0.0;
double term1=0.0, term2=0.0, term3=0.0, term22=0.0;
//term1 = 2.0*(Math.PI)*(frdp[i] + freq[j+p]);
term1 = 2.0*(Math.PI)*(frdp[i] - freq[j+p]);
term2 = term1 * term1;
term3 = ((dcp[i] * dcp[i]) + term2) * ((dcp[i] * dcp[i]) +
term2);
term22 = ((dcp[i]* dcp[i]) + term2) * ((dcp[i] * dcp[i]) +
term2) * ((dcp[i] * dcp[i]) + term2);
R[k] = (amplp[i]*dcp[i])/((dcp[i] * dcp[i]) + term2);
devRwrtamp= dcp[i]/((dcp[i] * dcp[i]) + term2);
double term4=0.0,term5=0.0;
term4 = amplp[i]/((dcp[i] * dcp[i]) + term2);
term5 = (2.0*amplp[i]*dcp[i] * dcp[i])/term3;
devRwrtdc = term4 - term5;
devRwrtfreq =
-(4.0*(Math.PI)*amplp[i]*dcp[i]*term1)/term3;

I[k] = (amplp[i]*term1)/((dcp[i] * dcp[i]) + term2);
devIwrtamp = term1/((dcp[i] * dcp[i]) + term2);
devIwrtdc = -(2.0*amplp[i]*dcp[i]*term1)/term3;
double term6=0.0,term7=0.0;
term6 = (2.0*(Math.PI)*amplp[i])/((dcp[i] * dcp[i]) + term2);
term7 = (4.0*(Math.PI)*amplp[i]*term2)/term3;
devIwrtfreq = term6 - term7;

Rdesh[k]= ((Math.cos(phasp[i])) * R[k]) - ((Math.sin(phasp[i]))
* I[k]);
realdiffsq[k] = (real1[k] - Rdesh[k]) * (real1[k] - Rdesh[k]);
realdiffsqsum[i] = realdiffsqsum[i] + realdiffsq[k];

double
devrealwrtamp=0.0,devrealwrtdc=0.0,devrealwrtfreq=0.0,
devrealwrtphase=0.0;
devrealwrtamp = ((Math.cos(phasp[i])) * devRwrtamp) -
((Math.sin(phasp[i])) * devIwrtamp);
devrealwrtdc = ((Math.cos(phasp[i])) * devRwrtdc) -
((Math.sin(phasp[i])) * devIwrtdc);
devrealwrtfreq = ((Math.cos(phasp[i])) * devRwrtfreq) -
((Math.sin(phasp[i])) * devIwrtfreq);
devrealwrtphase = -((Math.sin(phasp[i])) * R[k]) -
((Math.cos(phasp[i])) * I[k]);

Idesh[k]= ((Math.sin(phasp[i])) * R[k]) +
((Math.cos(phasp[i])) * I[k]);
imagdiff[k]= (imag1[k] - Idesh[k]);

```

```
imagdiffsq[k] = (imag1[k] - Idesh[k]) * (imag1[k] - Idesh[k]);
imagdiffsqsum[i] = imagdiffsqsum[i] + imagdiffsq[k];
```

```
double devimagwrtamp=0.0,devimagwrtdc=0.0,devimagwrtfreq=0.0,
devimagwrtphase=0.0;
devimagwrtamp = ((Math.sin(phasp[i])) * devRwrtamp) +
((Math.cos(phasp[i])) * devIwrtamp);
devimagwrtdc = ((Math.sin(phasp[i])) * devRwrtdc) +
((Math.cos(phasp[i])) * devIwrtdc);
devimagwrtfreq= ((Math.sin(phasp[i])) * devRwrtfreq) +
((Math.cos(phasp[i])) * devIwrtfreq);
devimagwrtphase = ((Math.cos(phasp[i])) * R[k]) -
((Math.sin(phasp[i])) * I[k]);
```

```
realimagdiffsqadd[k] = realdiffsq[k] + imagdiffsq[k];
Fdesh[i] = Fdesh[i] + realimagdiffsqadd[k];
```

```
ee[k] = ((real1[k] - Rdesh[k])*(-devrealwrtamp)) + ((imag1[k] -
Idesh[k])*(-devimagwrtamp));
FuncE[i] = FuncE[i] + ee[k];
```

```
f[k] = ((real1[k] - Rdesh[k])*(-devrealwrtdc)) + ((imag1[k] -
Idesh[k])*(-devimagwrtdc));
FuncF[i] = FuncF[i] + f[k];
```

```
g[k] = ((real1[k] - Rdesh[k])*(-devrealwrtfreq)) + ((imag1[k] -
Idesh[k])*(-devimagwrtfreq));
FuncG[i] = FuncG[i] + g[k];
```

```
h[k] = ((real1[k] - Rdesh[k])*(-devrealwrtphase)) + ((imag1[k] -
Idesh[k])*(-devimagwrtphase));
FuncH[i] = FuncH[i] + h[k];
```

Differentiating Functions E with respect to ampl., dc, freq,
and phase

```
edevwamp[k] = ((devrealwrtamp * devrealwrtamp) +
(devimagwrtamp * devimagwrtamp));
DevFuncE[0] = DevFuncE[0] + edevwamp[k];
```

```
double
term8=0.0,term9=0.0,Rdeshddcamp=0.0,Ideshddcamp=0.0,term10=0.0,
term11=0.0;
term8 = (1.0/((dcp[i] * dcp[i]) + term2))-((2.0* dcp[i]
* dcp[i])/term3);
term9 = -(2.0*dcp[i]*term1)/term3;
Rdeshddcamp = ((Math.cos(phasp[i])) * term8) -
((Math.sin(phasp[i])) * term9);
Ideshddcamp = ((Math.sin(phasp[i])) * term8) +
```

```
((Math.cos(phasp[i])) * term9);
term10 = ((devrealwrtdc*devrealwrtamp)-((real1[k] -
Rdesh[k]) * Rdeshddcamp));
term11 = ((devimagwrtdc * devimagwrtamp)-((imag1[k] -
Idesh[k]) * Ideshddcamp));
edevwdc[k] = term10 + term11;
DevFuncE[1]= DevFuncE[1] + edevwdc[k];
```

```
double
term12=0.0,term13=0.0,Rdeshdfrqamp=0.0,Ideshdfrqamp=0.0,
term14=0.0,term15=0.0;
term12 = -(4.0*(Math.PI)*dcp[i]*term1)/term3;
term13 = ((2.0*(Math.PI))/((dcp[i] * dcp[i] + term2))
- ((4.0*(Math.PI)*term2)/term3);
Rdeshdfrqamp = ((Math.cos(phasp[i])) * term12) -
((Math.sin(phasp[i])) * term13);
Ideshdfrqamp = ((Math.sin(phasp[i])) * term12) +
((Math.cos(phasp[i])) * term13);
term14 = ((devrealwrtfreq*devrealwrtamp)-((real1[k] -
Rdesh[k]) * Rdeshdfrqamp));
term15 = ((devimagwrtfreq*devimagwrtamp)-((imag1[k] -
Idesh[k]) * Ideshdfrqamp));
edevwfreq[k] = term14 + term15;
DevFuncE[2] = DevFuncE[2] + edevwfreq[k];
```

```
double
Rdeshdphasamp=0.0,Ideshdwphasamp=0.0,term16=0.0,term17=0.0;
Rdeshdphasamp = -((Math.sin(phasp[i])) * devRwrtamp) -
((Math.cos(phasp[i])) * devIwrtamp);
Ideshdwphasamp = ((Math.cos(phasp[i])) * devRwrtamp) -
((Math.sin(phasp[i])) * devIwrtamp);
term16 = (devrealwrtphase*devrealwrtamp)-((real1[k] -
Rdesh[k]) * Rdeshdphasamp);
term17 = (devimagwrtphase*devimagwrtamp)-((imag1[k] -
Idesh[k]) * Ideshdwphasamp);
edevwphase[k] = term16 + term17;
DevFuncE[3] = DevFuncE[3] + edevwphase[k];
```

```
/******
Differentiating Functions F with respect to ampl., dc, freq, and
phase
```

```
*****
double
term18=0.0,term19=0.0,Rdeshdampdc=0.0,Ideshdampdc=0.0,term20=0.0,term21=0.0;
term18 = (1.0/((dcp[i] * dcp[i] + term2)) -
((2.0*dcp[i]*dcp[i])/term3);
term19 = -(2.0*dcp[i]*term1)/term3;
Rdeshdampdc = ((Math.cos(phasp[i])) * term18) -
((Math.sin(phasp[i])) * term19);
Ideshdampdc = ((Math.sin(phasp[i])) * term18 ) +
```

```

(Math.cos(phasp[i])) * term19);
term20 = (devrealwrtamp*devrealwrtdc)-((real1[k] -
Rdesh[k]) * Rdeshdampdc);
term21 = (devimagwrtamp*devimagwrtdc)-((imag1[k] -
Idesh[k]) * Ideshdampdc);
fdevwamp[k] = term20 + term21;
DevFuncF[0] = DevFuncF[0] + fdevwamp[k];

double
term23=0.0,term24=0.0,Rdeshddcdc=0.0,Ideshddcdc=0.0,term25=0.0,term26=0.0;
term23 = -((6.0*amplp[i]*dcp[i])/term3) +
((8.0*amplp[i]*dcp[i]*dcp[i]*dcp[i])/term22);
term24 = -((2.0*amplp[i]*term1)/term3) +
((8.0*amplp[i]*term1*dcp[i]*dcp[i])/term22);
Rdeshddcdc = ((Math.cos(phasp[i])) * term23) -
((Math.sin(phasp[i])) * term24);
Ideshddcdc = ((Math.sin(phasp[i])) * term23) +
((Math.cos(phasp[i])) * term24);
term25 = (devrealwrtdc*devrealwrtdc)-((real1[k] - Rdesh[k]) *
Rdeshddcdc );
term26 = (devimagwrtdc*devimagwrtdc)-((imag1[k] - Idesh[k]) *
Ideshddcdc);
fdevwdc[k] = term25 + term26;
DevFuncF[1] = DevFuncF[1] + fdevwdc[k];

double
term27=0.0,term28=0.0,Rdeshdfrqdc=0.0,Ideshdfrqdc=0.0,term29=0.0,term30=0.0;
term27 = -((4.0*(Math.PI)*amplp[i]*term1)/term3) +
((16.0*(Math.PI)*amplp[i]*dcp[i]*dcp[i]*term1)/term22);
term28 = -((4.0*(Math.PI)*amplp[i]*dcp[i])/term3) +
((16.0*(Math.PI)*amplp[i]*dcp[i]*term2)/term22);
Rdeshdfrqdc = ((Math.cos(phasp[i])) * term27) -
((Math.sin(phasp[i])) * term28);
Ideshdfrqdc = ((Math.sin(phasp[i])) * term27) +
((Math.cos(phasp[i])) * term28);
term29 = (devrealwrtfreq*devrealwrtdc)-((real1[k] - Rdesh[k]) *
Rdeshdfrqdc);
term30 = (devimagwrtfreq*devimagwrtdc)-((imag1[k] - Idesh[k]) *
Ideshdfrqdc);
fdevwfreq[k] = term29 + term30;
DevFuncF[2] = DevFuncF[2] + fdevwfreq[k];

double
Rdeshdphasdc=0.0,Ideshdphasdc=0.0,term31=0.0,term32=0.0;
Rdeshdphasdc = -((Math.sin(phasp[i])) * devRwrtdc) -
((Math.cos(phasp[i])) * devIwrtdc);
Ideshdphasdc = ((Math.cos(phasp[i])) * devRwrtdc) -
((Math.sin(phasp[i])) * devIwrtdc);
term31 = (devrealwrtphase*devrealwrtdc)-((real1[k] - Rdesh[k]) *
Rdeshdphasdc);
term32 = (devimagwrtphase*devimagwrtdc)-((imag1[k] - Idesh[k]) *

```

```

Ideshdphasdc);
fdevwphase[k] = term31 + term32;
DevFuncF[3] = DevFuncF[3] + fdevwphase[k];

```

```

/*****

```

Differentiating Functions G with respect to ampl, dc, freq,
and phase

```

*****

```

```

double
term33=0.0,term34=0.0,Rdeshdampfrq=0.0,Ideshdampfrq=0.0,term35=0.0,term36=0.0;
term33 = -(4.0*(Math.PI)*dcp[i]*term1)/term3;
term34 = ((2.0*(Math.PI))/((dcp[i] * dcp[i]) + term2))
- ((4.0*(Math.PI)*term2)/term3);
Rdeshdampfrq = ((Math.cos(phasp[i])) * term33) -
((Math.sin(phasp[i])) * term34);
Ideshdampfrq = ((Math.sin(phasp[i])) * term33) +
((Math.cos(phasp[i])) * term34);
term35 = (devrealwrtamp*devrealwrtfreq)-((real1[k] - Rdesh[k]) *
Rdeshdampfrq);
term36 = (devimagwrtamp*devimagwrtfreq)-((imag1[k] - Idesh[k]) *
Ideshdampfrq);
gdevwamp[k] = term35 + term36;
DevFuncG[0] = DevFuncG[0] + gdevwamp[k];

```

```

double
term37=0.0,term38=0.0,Rdeshddcfrq=0.0,Ideshddcfrq=0.0,term39=0.0,term40=0.0;
term37 = -((4.0*(Math.PI)*amplp[i]*term1)/term3) +
((16.0*(Math.PI)*amplp[i]*dcp[i] * dcp[i]*term1)/term22);
term38 = -((4.0*(Math.PI)*amplp[i]*dcp[i])/term3) +
((16.0*(Math.PI)*amplp[i]*dcp[i] *term2)/term22);
Rdeshddcfrq = ((Math.cos(phasp[i])) * term37) -
((Math.sin(phasp[i])) * term38);
Ideshddcfrq = ((Math.sin(phasp[i])) * term37) +
((Math.cos(phasp[i])) * term38);
term39 = (devrealwrtdc*devrealwrtfreq)-((real1[k] - Rdesh[k]) *
Rdeshddcfrq);
term40 = (devimagwrtdc*devimagwrtfreq)-((imag1[k] - Idesh[k]) *
Ideshddcfrq);
gdevwdc[k] = term39 + term40;
DevFuncG[1] = DevFuncG[1] + gdevwdc[k];

```

```

double
term41=0.0,term42=0.0,Rdeshdfrqfrq=0.0,Ideshdfrqfrq=0.0,term43=0.0,term44=0.0;
term41 =
-((8.0*(Math.PI)*(Math.PI)*amplp[i]*dcp[i])/term3) +
((32.0*(Math.PI)*(Math.PI)*amplp[i]*dcp[i]*term2)/term22);
term42 = -((24.0*amplp[i]*term1*(Math.PI)*(Math.PI))/term3) +
((32.0*(Math.PI)*(Math.PI)*amplp[i]*term1*term1)/term22);
Rdeshdfrqfrq = ((Math.cos(phasp[i])) * term41) -

```

```

(Math.sin(phasp[i])) * term42);
Ideshdfrqfrq = ((Math.sin(phasp[i])) * term41) +
((Math.cos(phasp[i])) * term42);
term43 = (devrealwrtfreq*devrealwrtfreq)-((real1[k] - Rdes[k]) *
Rdesdfrqfrq);
term44 = (devimagwrtfreq*devimagwrtfreq)-((imag1[k] - Idesh[k]) *
Ideshdfrqfrq);
gdevwfreq[k] = term43 + term44;
DevFuncG[2] = DevFuncG[2] + gdevwfreq[k];

```

```

double
Rdesdphasfrq=0.0,Ideshdphasfrq=0.0,term45=0.0,term46=0.0;
Rdesdphasfrq = -((Math.sin(phasp[i])) * devRwrtfreq) -
((Math.cos(phasp[i])) * devIwrtfreq);
Ideshdphasfrq = ((Math.cos(phasp[i])) * devRwrtfreq) -
((Math.sin(phasp[i])) * devIwrtfreq);
term45 = (devrealwrtphase*devrealwrtfreq)-((real1[k] - Rdes[k]) *
Rdesdphasfrq);
term46 = (devimagwrtphase*devimagwrtfreq)-((imag1[k] - Idesh[k]) *
Ideshdphasfrq);
gdevwphase[k] = term45 + term46;
DevFuncG[3] = DevFuncG[3] + gdevwphase[k];

```

/******

Differentiating Functions H with respect to ampl., dc, freq,
and phase

```

double
Rdesdampphas=0.0,Ideshdampphas=0.0,term47=0.0,term48=0.0;
Rdesdampphas = -((Math.sin(phasp[i])) * devRwrtamp) -
((Math.cos(phasp[i])) * devIwrtamp);
Ideshdampphas = ((Math.cos(phasp[i])) * devRwrtamp) -
((Math.sin(phasp[i])) * devIwrtamp);
term47 = (devrealwrtamp*devrealwrtphase)-((real1[k] - Rdes[k])
* Rdesdampphas);
term48 = (devimagwrtamp*devimagwrtphase)-((imag1[k] - Idesh[k])
* Ideshdampphas);
hdevwamp[k] = term47 + term48;
DevFuncH[0] = DevFuncH[0] + hdevwamp[k];

```

```

double
Rdesddcphas=0.0,Ideshddcphas=0.0,term49=0.0,term50=0.0;
Rdesddcphas = -((Math.sin(phasp[i])) * devRwrtdc) -
((Math.cos(phasp[i])) * devIwrtdc);
Ideshddcphas = ((Math.cos(phasp[i])) * devRwrtdc) -
((Math.sin(phasp[i])) * devIwrtdc);
term49 = (devrealwrtdc*devrealwrtphase)-((real1[k] - Rdes[k]) *
Rdesddcphas);
term50 = (devimagwrtdc*devimagwrtphase)-((imag1[k] - Idesh[k]) *

```



```

double[][] delta1 = new double[4][4];
double[][] delta2 = new double[4][4];
double[][] delta3 = new double[4][4];
double[][] delta4 = new double[4][4];

```

```

for(int m=0; m<4; m++)
{
for(int n=0; n<4; n++)
{
delta1[m][n] = Func[m][n];
delta1[m][n+1] = J[m][n+1];
delta1[m][n+2] = J[m][n+2];
delta1[m][n+3] = J[m][n+3];
}
}

```

```

for(int m=0; m<4; m++)
{
for(int n=0; n<4; n++)
{
delta2[m][n] = J[m][n];
delta2[m][n+1] = Func[m][n];
delta2[m][n+2] = J[m][n+2];
delta2[m][n+3] = J[m][n+3];
}
}

```

```

for(int m=0; m<4; m++)
{
for(int n=0; n<4; n++)
{
delta3[m][n] = J[m][n];
delta3[m][n+1] = J[m][n+1];
delta3[m][n+2] = Func[m][n];
delta3[m][n+3] = J[m][n+3];
}
}

```

```

for(int m=0; m<4; m++)
{
for(int n=0; n<4; n++)
{
delta4[m][n] = J[m][n];
delta4[m][n+1] = J[m][n+1];
delta4[m][n+2] = J[m][n+2];
delta4[m][n+3] = Func[m][n];
}
}

```

Help in Java swings(internal Frame)

```
Determinant4 de1 = new Determinant4(J);
double determinant1 = de1.returvalue();

Determinant4 de2 = new Determinant4(delta1);
double determinant2 = de2.returvalue();

Determinant4 de3 = new Determinant4(delta2);
double determinant3 = de3.returvalue();

Determinant4 de4 = new Determinant4(delta3);
double determinant4 = de4.returvalue();

Determinant4 de5 = new Determinant4(delta4);
double determinant5 = de5.returvalue();

deltaamp[i] = determinant2/determinant1;
deltadcp[i] = determinant3/determinant1;
deltafrd[i] = determinant4/determinant1;
deltaphas[i] = determinant5/determinant1;

amplp[i+1] = amplp[i] + deltaamp[i];
dcp[i+1] = dcp[i] + deltadcp[i];
frdp[i+1] = frdp[i] + deltafrd[i];
phasp[i+1] = phasp[i] + deltaphas[i];

x1= amplp[i+1];
x2 = dcp[i+1];
x3 = frdp[i+1];
x4 = phasp[i+1];
countloop++;
}

for(int s=0; s<totalpoint; s++)
{
double term55 = 0.0, term56 = 0.0;
//term55 = 2.0*(Math.PI)*(frdp[i+1] + freq[s]);
term55 = 2.0*(Math.PI)*(x3 - freq[s]);
term56 = term55 * term55;
itRp[s] = (x1 * x2)/((x2 * x2) + term56);
itIp[s] = (x1 * term55)/((x2 * x2) + term56);
itratpeaks[s]= ((Math.cos(x4)) * itRp[s]) - ((Math.sin(x4)) *
itIp[s]);
itiatpeaks[s]= ((Math.sin(x4)) * itRp[s]) + ((Math.cos(x4)) *
itIp[s]);
iterreal[s] = (realdata[s] - itratpeaks[s]);
iterimag[s] = (imagdata[s] - itiatpeaks[s]);
}

System.out.println("Loop Complete: " + countloop);

output[0]= x1;
```

Help in Java swings(internal Frame)

```
output[1]= x2;  
output[2]= x3;  
output[3]= x4;
```

```
fileHandling fw = new fileHandling();  
fw.filewrite(savepath, iterreal, "iterreal", foldername);  
fw.filewrite(savepath, iterimag, "iterimag", foldername);  
fw.filewrite(savepath, output, "Output", foldername);  
}
```