References

1. Schlumberger, "UBI Ultrasonic Borehole Imager", Wireline&Testing, December 1997.

2. Schlumberger Engineering Department "Multifinger Caliper Tool Manual, Theory of Operation" Schlumberger Engineering (Paris).

3. Pilu, M., Fitzgibbon, A. W., and Fisher, R. B. " Ellipse-specific Direct least-square Fitting", IEEE International Conference on Image Processing, Lausanne, September 1996.

4. Pilu, M., Fitzgibbon, A. W., and Fisher, R. B. " Direct least-square fitting of Ellipses " International Conference on Pattern Recognition, Vienna, August 1996.

Appendices

Appendix A

MainWindow.java

```
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;

import java.io.*;

public class  MainWindow extends JFrame {
      final File path=new File("C:/MyCSVfile");
      JFrame frame;
      JTextField ab;
      JTextField phaseShift1, phaseShift2 ;
      JRadioButton rb1, rb2;
      final String abratio="One Data";
      final String ps="Two Data";

      public MainWindow() {
            super("MainWindow");

            //Create JTextarea
            final String n[]=path.list();
            String s=null;
            for (int i=0; i<n.length ;i++ ){
                  s=s+n[i]+"\n";
            }

            JTextArea ta=new JTextArea(s);
            ta.setEditable(false);
            JScrollPane sp=new JScrollPane(ta);


           sp.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
            sp.setPreferredSize(new Dimension(250,250));
            sp.setBorder(
                  BorderFactory.createCompoundBorder(
                        BorderFactory.createCompoundBorder(
                              BorderFactory.createTitledBorder("CSD file"),

           BorderFactory.createEmptyBorder(5,5,5,5)),
                        sp.getBorder()));

            JPanel p1=new JPanel();
            p1.add(sp);

            //Create JRadiobutton
            rb1=new JRadioButton(abratio);
            rb1.setActionCommand(abratio);
            rb1.setSelected(true);
            rb2=new JRadioButton(ps);
            rb2.setActionCommand(ps);

            final ButtonGroup group=new ButtonGroup();
```

```
group.add(rb1);
group.add(rb2);


ab=new JTextField(10);
phaseShift1=new JTextField(10);
phaseShift1.setEditable(false);
phaseShift2=new JTextField(10);
phaseShift2.setEditable(false);

JLabel data1=new JLabel("Data 1");
JLabel data2=new JLabel("Data 2");

JPanel p2=new JPanel();
GridBagLayout gridBag=new GridBagLayout();
GridBagConstraints gc=new GridBagConstraints();
p2.setLayout(gridBag);

gc.gridx=0;      gc.gridy=0;      //add JRadioButton for a/b at grid 0,0
gc.anchor=GridBagConstraints.WEST;
                        gridBag.setConstraints(rb1, gc);
                        p2.add(rb1);

gc.gridx=1;      gc.gridy=0;      //add JTextField for a/b at grid 1,0
                        gridBag.setConstraints(ab, gc);
                        p2.add(ab);

gc.gridx=0;      gc.gridy=1;      //add JRadioButton for a/b at grid 0,1
gc.anchor=GridBagConstraints.WEST;
                        gridBag.setConstraints(rb2, gc);
                        p2.add(rb2);

gc.gridx=0;      gc.gridy=3;      //add JLabel for Data at grid 3,0
gc.anchor=GridBagConstraints.EAST;
                        gridBag.setConstraints(data1, gc);
                        p2.add(data1);

gc.gridx=1;      gc.gridy=3;      //add JTextField for phaseShift at grid 3,1
                        gridBag.setConstraints(phaseShift1, gc);
                        p2.add(phaseShift1);

gc.gridx=0;      gc.gridy=4;      //add JLabel for Data at grid 4,0
gc.anchor=GridBagConstraints.EAST;
                        gridBag.setConstraints(data2, gc);
                        p2.add(data2);

gc.gridx=1;      gc.gridy=4;      //add JTextField for phaseShift at grid 4,1
                        gridBag.setConstraints(phaseShift2, gc);
                        p2.add(phaseShift2);

p2.setBorder(
     BorderFactory.createCompoundBorder(
          BorderFactory.createCompoundBorder(
               BorderFactory.createTitledBorder("One or Two data"),

BorderFactory.createEmptyBorder(5,5,5,5)),
          p2.getBorder()));
```

```
        //Create JButton
        JButton bOK=new JButton("OK");
        JButton bClose=new JButton("Close");
        JButton bClear=new JButton("Clear");

        JPanel p3=new JPanel();
        p3.add(bOK);
        p3.add(bClose);
        p3.add(bClear);

        rb1.addItemListener(new ItemListener(){//Action for JRadioButton of rb1
            public void itemStateChanged(ItemEvent e){
                        phaseShift1.setEditable(false);
                        phaseShift2.setEditable(false);
                        ab.setEditable(true);
            }
        });

        rb2.addItemListener(new ItemListener(){//Action for JRadioButton of rb2
            public void itemStateChanged(ItemEvent e){
                        phaseShift1.setEditable(true);
                        phaseShift2.setEditable(true);
                        ab.setEditable(false);
                    }
        });

        //Action for the button of OK, Close and Clear
        bOK.addActionListener(new ActionListener(){//Action for OK Button
            public void actionPerformed(ActionEvent e){
                String txtab=ab.getText();
                String txtps1=phaseShift1.getText();
                String txtps2=phaseShift2.getText();
                String filenameOnedata=path+ "\\" +txtab;
                String filenameTwodata[]=new String[2];
                filenameTwodata[0]=path+ "\\" +txtps1;
                filenameTwodata[1]=path+ "\\" +txtps2;
                String command=group.getSelection().getActionCommand();
                if (command==abratio) {
                    if(txtab.length() ==0 )

            JOptionPane.showMessageDialog(frame,"Please type your data file.CSD from CSD
Window",

            "Information Dialog", JOptionPane.PLAIN_MESSAGE);
                        else {
                            CheckFile checkFileForOneData=new CheckFile(path, txtab);
                            try {

                ProgramAoverB pr=new ProgramAoverB(filenameOnedata);

            }
                            catch (Exception ex) {

                System.err.println("Error at ProgramAB of MainWindow");

                System.err.println(e);
                            }
                                                                                    }
```

```
                }
                else if (command==ps){
                    if (txtps1.length() !=0 & txtps2.length() ==0){

        JOptionPane.showMessageDialog(frame,"You're not type the file of Data2 yet",

        "Information Dialog", JOptionPane.PLAIN_MESSAGE);
                    }
                    else if (txtps1.length() ==0 & txtps2.length() !=0){

        JOptionPane.showMessageDialog(frame,"You're not type the file of Data1 yet",

        "Information Dialog", JOptionPane.PLAIN_MESSAGE);
                    }
                    else if (txtps1.length() !=0 & txtps2.length() !=0){
                        CheckFile checkFileForTwoData1=new CheckFile(path, txtps1);
                        CheckFile checkFileForTwoData2=new CheckFile(path, txtps2);

        //System.out.println(filenameTwodata[0]+"\t"+ filenameTwodata[1]+"\t"+"at MainWindow");
                        try {
                            ProgramPhaseShift pps=new
ProgramPhaseShift(filenameTwodata);
                        }
                        catch (Exception ex) {

                System.err.println("Error at ProgramPhaseShift of MainWindow");

                System.err.println(e);
                        }


                    }
                    else if (txtps1.length() ==0 & txtps2.length() ==0){

        JOptionPane.showMessageDialog(frame,"You're not type Both file of Data1 and 2",
                                                    "Information Dialog",
JOptionPane.PLAIN_MESSAGE);
                    }
                }
            }
        });

        bClose.addActionListener(new ActionListener(){//Action for Close Button
            public void actionPerformed(ActionEvent e){
                System.exit(0);
            }
        });

        bClear.addActionListener(new ActionListener(){//Action for Clear Button
            public void actionPerformed(ActionEvent e){
                ab.setText("");
                phaseShift1.setText("");
                phaseShift2.setText("");
                rb1.setSelected(true);
            }
        });

        //add everything into the JFrame
```

```
        JPanel contentPane=new JPanel();
        BoxLayout bl=new BoxLayout(contentPane, BoxLayout.Y_AXIS);
        contentPane.setLayout(bl);
        contentPane.add(p1);
        contentPane.add(p2);
        contentPane.add(p3);
        setContentPane(contentPane);
    }

    public static void main(String[] args) throws IOException{
        JFrame frame = new MainWindow();
        frame.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });

        frame.pack();
        frame.setVisible(true);
    }//End of Main()
}//End of Class MainWindow
```

CheckFile.java

```
import java.io.*;
import javax.swing.*;

class CheckFile{
	File path;
	CheckFile(File path, String nameOfFile){
		this.path=path;
		String n[]=path.list();
		String selectedFile=nameOfFile;
		String filename=path+ "\\" +selectedFile;
		int m=0;

		for (int i=0; i<n.length ;i++ ){
			if(n[i].equals(selectedFile)){
				m+=1;
			}
			else{
				m+=0;
			}
			//System.out.println("i value "+i+"\t"+"m value "+ m);
		}
			//System.out.println();
			//System.out.println(m);

		if (m>0){
			//System.out.println(selectedFile+" "+"at CheckFileForTwoData");
			//Program pr=new Program(path, selectedFile);
		}
		else{
			JOptionPane.showMessageDialog(new JFrame(),"Please check your type and
extendtion",
			"Information Dialog", JOptionPane.PLAIN_MESSAGE);
		}
	}//End of Constructor CheckFile

}//End of Class CheckFile
```

InputData.java

```
import java.io.*;
import java.util.StringTokenizer;
import java.text.*;

class InputData{
	public double databuf[][];
	private BufferedReader buf;
	private int row;

			NumberFormat formatter = NumberFormat.getNumberInstance();

	//class constructor
	InputData(String f) {
		row=0;

		try{
			buf=new BufferedReader(new FileReader(f));
		}catch(FileNotFoundException e){
			System.out.println("File \""+f+"\" not found!, please specify the existent
file.");
			System.out.println("Usage: java program {filename}");
			System.exit(0);
		}

		databuf=new double[1][73];

		String s=null;
		try{
		while((s=buf.readLine())!=null){
			if(s!=null){
				add(s);
				row++;
			}
		}
		}catch(IOException e){
			System.out.println("Can not read the file :"+f);
			System.out.println("Usage: java program {filename}");
			System.exit(0);
		}
					double [][] h = new double[row][74];
					System.arraycopy(databuf,0,h,0,h.length);
					databuf=h;

	}//End of Constructor Program

	private void add(String s){
		StringTokenizer st = new StringTokenizer(s,",");
		String token=null;
		int column=0;

		while (st.hasMoreTokens()) {
```

```
                token=(st.nextToken()).trim();

                if(row >= databuf.length){
                    double d[][] = new double[2*databuf.length][73];
                    System.arraycopy(databuf,0,d,0,databuf.length);
                    databuf=d;
                }

                try{
                    databuf[row][column]=Double.parseDouble(token);
                }catch(NumberFormatException e){
                    System.out.println("Invalid input number!");
                    System.out.println("Usage: java program {filename}");
                    System.exit(0);
                }
                column++;
            }
        }//End of add()

            public double getData(int row,int col) {
                if (row<databuf.length && col<databuf[0].length) {
                    return databuf[row][col];
                }
                return 0;
            }//End of getData()

}//End of Class InputData
```

ProgramAoverB.java

```
import java.io.*;

class ProgramAoverB{

      ProgramAoverB(String nameOfFile) throws IOException {
            String filename=nameOfFile;
            if(filename != null){
                                                InputData d=new InputData(filename);
                                                PrintData print=new PrintData("InputData",
d.databuf);
                                                CalCoeff e=new CalCoeff(d);
                                                e.cal();

                                                System.out.println("End of File");
            }
      }

}//End of  Class program
```

ProgramPhaseShift.java

```
import java.io.*;
import java.util.StringTokenizer;

class ProgramPhaseShift{

    ProgramPhaseShift(String nameOfFile[]) throws IOException{
                                    Runtime r=Runtime.getRuntime();
                                    r.gc();
                                    //System.out.println("Start Run");
                                    //System.out.println("Total Memory: " + r.totalMemory() + '\n' +
"Free Memory: " + r.freeMemory() );
        InputData d1=new InputData(nameOfFile[0]);
                                    InputData d2=new InputData(nameOfFile[1]);

        PrintData printInput=new PrintData("InputData 1", d1.databuf, "InputData 2", d2.databuf);
                                    r.gc();
        PhaseShiftOfCoeff d11=new PhaseShiftOfCoeff(d1, d2);
                                    r.gc();
                                    PrintData printCoeff=new PrintData(d11.index, d11.coeffDif,
d11.bpost, "Coeff");
                                    PrintData printSummary=new PrintData(d11.keyForward,
d11.bpost1, d12.bpost, d13.bpost,

                        d11.index, d11.coeffDif, d12.sumDif, d13.sumDif);
                                    //System.out.println("Finished Run");
                                    //System.out.println("Free Memory: " + r.freeMemory() );
    }//End of Constructor ProgramPhaseShift

}//End of Class ProgramPhaseShift
```

CalCoeff.java

```
import java.io.*;
import java.lang.Double;
import java.util.Vector;
import java.text.*;

class CalCoeff{
        InputData d;
        private int   i;
        private Vector points = new Vector(16,4);
        private static double FPFConstraint[][] = new double[7][7];
        NumberFormat formatter = NumberFormat.getNumberInstance();
        double databuf[][];
        double coeff[][];
        double D[][];
        double xy[][]=new double[73][2];
        double disdiff[]=new double [73];

        CalCoeff(InputData d) {
                        databuf=new double[d.databuf.length][d.databuf[0].length];
                        for(int i=0;i<databuf.length;i++) {
              for(int j=0;j<databuf[0].length;j++)
                   databuf[i][j]=d.databuf[i][j];
                        }
        }// End of Constructor CalCoeff()

        CalCoeff(double temp[][]) {
                        databuf=new double[temp.length][temp[0].length];
                        for(int i=0;i<databuf.length;i++) {
              for(int j=0;j<databuf[0].length;j++)
                   databuf[i][j]=temp[i][j];
                        }
        }// End of Constructor CalCoeff()

        private void multMatrix(double m[][], double g[][], double mg[][]) {


    // First clear the return array
    for(int i=0; i<4; i++)
       for(int j=0; j<2; j++)
          mg[i][j]=0;

    // Perform the matrix math
    for(int i=0; i<4; i++)
       for(int j=0; j<2; j++)
          for(int k=0; k<4; k++)
             mg[i][j]=mg[i][j] + (m[i][k] * g[k][j]);
  }


   private void ROTATE(double a[][], int i, int j, int k, int l, double tau, double s){
          double g,h;
          g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);
          a[k][l]=h+s*(g-h*tau);
         }//End of ROTATE
```

```
private void jacobi(double a[][], int n, double d[] , double v[][], int nrot){
        int j,iq,ip,i;
        double tresh,theta,tau,t,sm,s,h,g,c;

        double b[] = new double[n+1];
        double z[] = new double[n+1];

        for (ip=1;ip<=n;ip++) {
              for (iq=1;iq<=n;iq++) v[ip][iq]=0.0;
        v[ip][ip]=1.0;
        }
        for (ip=1;ip<=n;ip++) {
              b[ip]=d[ip]=a[ip][ip];
              z[ip]=0.0;
        }
        nrot=0;
        for (i=1;i<=50;i++) {
              sm=0.0;
              for (ip=1;ip<=n-1;ip++) {
                    for (iq=ip+1;iq<=n;iq++)
                          sm += Math.abs(a[ip][iq]);
              }
    if (sm == 0.0) {
     /*    free_vector(z,1,n);
          free_vector(b,1,n);  */
     return;
    }
    if (i < 4)
     tresh=0.2*sm/(n*n);
    else
     tresh=0.0;
    for (ip=1;ip<=n-1;ip++) {
     for (iq=ip+1;iq<=n;iq++) {
      g=100.0*Math.abs(a[ip][iq]);
      if (i > 4 && Math.abs(d[ip])+g == Math.abs(d[ip])
        && Math.abs(d[iq])+g == Math.abs(d[iq]))
       a[ip][iq]=0.0;
      else if (Math.abs(a[ip][iq]) > tresh) {
       h=d[iq]-d[ip];
       if (Math.abs(h)+g == Math.abs(h))
        t=(a[ip][iq])/h;
       else {
        theta=0.5*h/(a[ip][iq]);
        t=1.0/(Math.abs(theta)+Math.sqrt(1.0+theta*theta));
        if (theta < 0.0) t = -t;
       }
       c=1.0/Math.sqrt(1+t*t);
       s=t*c;
       tau=s/(1.0+c);
       h=t*a[ip][iq];
       z[ip] -= h;
       z[iq] += h;
       d[ip] -= h;
       d[iq] += h;
       a[ip][iq]=0.0;
       for (j=1;j<=ip-1;j++) {
        ROTATE(a,j,ip,j,iq,tau,s);
```

```
        }
        for (j=iq+1;j<=n;j++) {
          ROTATE(a,ip,j,iq,j,tau,s);
        }
        for (j=1;j<=n;j++) {
          ROTATE(v,j,ip,j,iq,tau,s);
        }
        ++nrot;
      }
    }
  }
  for (ip=1;ip<=n;ip++) {
    b[ip] += z[ip];
    d[ip]=b[ip];
    z[ip]=0.0;
  }
 }
 //printf("Too many iterations in routine JACOBI");
}//End of jacobi()


// Perform the Cholesky decomposition
// Return the lower triangular L such that L*L'=A
private void choldc(double a[][], int n, double l[][]){
  int i,j,k;
  double sum;
  double p[] = new double[n+1];

  for (i=1; i<=n; i++) {
   for (j=i; j<=n; j++) {
    for (sum=a[i][j],k=i-1;k>=1;k--) sum -= a[i][k]*a[j][k];
    if (i == j) {
     if (sum<=0.0)
      // printf("\nA is not poitive definite!");
      {}
     else
      p[i]=Math.sqrt(sum); }
    else
     {
      a[j][i]=sum/p[i];
     }
   }
  }
  for (i=1; i<=n; i++)
   for (j=i; j<=n; j++)
    if (i==j)
     l[i][i] = p[i];
    else
     {
      l[j][i]=a[j][i];
      l[i][j]=0.0;
     }
 }//End of choldc()
int inverse(double TB[][], double InvB[][], int N) {
 int k,i,j,p,q;
 double mult;
 double D,temp;
 double maxpivot;
```

```
int npivot;
double B[][] = new double [N+1][N+2];
double A[][] = new double [N+1][2*N+2];
double C[][] = new double [N+1][N+1];
double eps = 10e-20;


for(k=1;k<=N;k++)
 for(j=1;j<=N;j++)
  B[k][j]=TB[k][j];

for (k=1;k<=N;k++)
 {
  for (j=1;j<=N+1;j++)
   A[k][j]=B[k][j];
  for (j=N+2;j<=2*N+1;j++)
   A[k][j]=(float)0;
  A[k][k-1+N+2]=(float)1;
 }
for (k=1;k<=N;k++)
 {
  maxpivot=Math.abs((double)A[k][k]);
  npivot=k;
  for (i=k;i<=N;i++)
   if (maxpivot<Math.abs((double)A[i][k]))
     {
      maxpivot=Math.abs((double)A[i][k]);
      npivot=i;
     }
  if (maxpivot>=eps)
   {    if (npivot!=k)
         for (j=k;j<=2*N+1;j++)
           {
            temp=A[npivot][j];
            A[npivot][j]=A[k][j];
            A[k][j]=temp;
           } ;
        D=A[k][k];
        for (j=2*N+1;j>=k;j--)
         A[k][j]=A[k][j]/D;
        for (i=1;i<=N;i++)
         {
          if (i!=k)
           {
            mult=A[i][k];
            for (j=2*N+1;j>=k;j--)
              A[i][j]=A[i][j]-mult*A[k][j] ;
           }
         }
       }
  else
   { // printf("\n The matrix may be singular !!") ;
     return(-1);
    };
 }
/**  Copia il risultato nella matrice InvB  ***/
for (k=1,p=1;k<=N;k++,p++)
 for (j=N+2,q=1;j<=2*N+1;j++,q++)
```

```
      InvB[p][q]=A[k][j];
   return(0);
  }         /*  End of INVERSE   */


  private void AperB(double _A[][], double _B[][], double _res[][],
              int _righA, int _colA, int _righB, int _colB) {
          int p,q,l;
          for (p=1;p<=_righA;p++)
               for (q=1;q<=_colB;q++){
                    _res[p][q]=0.0;
                    for (l=1;l<=_colA;l++)
                         _res[p][q]=_res[p][q]+_A[p][l]*_B[l][q];

               }
  }//End of AperB()


  private void A_TperB(double _A[][], double  _B[][], double _res[][],
               int _righA, int _colA, int _righB, int _colB) {
   int p,q,l;
   for (p=1;p<=_colA;p++)
               for (q=1;q<=_colB;q++){
                    _res[p][q]=0.0;
                    for (l=1;l<=_righA;l++)
                         _res[p][q]=_res[p][q]+_A[l][p]*_B[l][q];

     }
  }//End of A_TperB()


  private void AperB_T(double _A[][], double _B[][], double _res[][],
               int _righA, int _colA, int _righB, int _colB) {
   int p,q,l;
   for (p=1;p<=_colA;p++)
          for (q=1;q<=_colB;q++){
               _res[p][q]=0.0;
               for (l=1;l<=_righA;l++)
                    _res[p][q]=_res[p][q]+_A[p][l]*_B[q][l];
          }
  }//End of AperB_T()


public void pv(double v[], java.lang.String  str) {
          formatter.setMaximumFractionDigits(7);
          formatter.setMinimumFractionDigits(7);
          System.out.println();
          System.out.println("---------------------------- " + str + " ------------------------------");
          System.out.println(" " + formatter.format(v[1]) + "\t" + formatter.format(v[2]) + "\t" +
formatter.format(v[3])
               + "\t" + formatter.format(v[4])+ "\t" + formatter.format(v[5]) + "\t" + formatter.format
(v[6]) );
          System.out.println("---------------------------------" + " End " + str + " ------------------------------
--------");
          System.out.println();
  }//End of pv()
```

```
   public void pm(double S[][], java.lang.String  str) {
            formatter.setMaximumFractionDigits(5);
            formatter.setMinimumFractionDigits(5);
            System.out.println();
            System.out.println("---------------------------------------- " + str + " ----------------------------------
----------");
            System.out.println(" " + formatter.format(S[1][1])+ "\t\t\t" + formatter.format(S[1][2])+ "\t\t\t"
+ formatter.format(S[1][3]) + "\t\t\t" + formatter.format(S[1][4]) + "\t\t\t" + formatter.format(S[1][5])+
"\t\t\t" + formatter.format(S[1][6]) );

            System.out.println(" " + formatter.format(S[2][1]) + "\t\t\t" + formatter.format(S[2][2]) +
"\t\t\t" + formatter.format(S[2][3]) + "\t\t\t" + formatter.format(S[2][4]) + "\t\t\t" + formatter.format(S
[2][5]) + "\t\t\t" + formatter.format(S[2][6]) );

            System.out.println(" " + formatter.format(S[3][1]) + "\t\t\t" + formatter.format(S[3][2]) +
"\t\t\t" + formatter.format(S[3][3]) + "\t\t\t" + formatter.format(S[3][4]) + "\t\t\t" + formatter.format(S
[3][5]) + "\t\t\t" + formatter.format(S[3][6]) );

            System.out.println(" " + formatter.format(S[4][1]) + "\t\t\t" + formatter.format(S[4][2]) +
"\t\t\t" + formatter.format(S[4][3]) + "\t\t\t" + formatter.format(S[4][4]) + "\t\t\t" + formatter.format(S
[4][5]) + "\t\t\t" + formatter.format(S[4][6]) );

            System.out.println(" " + formatter.format(S[5][1]) + "\t\t\t" + formatter.format(S[5][2]) +
"\t\t\t" + formatter.format(S[5][3]) + "\t\t\t" + formatter.format(S[5][4]) + "\t\t\t" + formatter.format(S
[5][5]) + "\t\t\t" + formatter.format(S[5][6]) );

            System.out.println(" " + formatter.format(S[6][1]) + "\t\t\t" + formatter.format(S[6][2]) +
"\t\t\t" + formatter.format(S[6][3]) + "\t\t\t" + formatter.format(S[6][4]) + "\t\t\t" + formatter.format(S
[6][5]) + "\t\t\t" + formatter.format(S[6][6]) );

            System.out.println("------------------------------------------------------------------ " + " End  " + str +
" ------------------------------------------------------------");
            System.out.println();
   }   //End of pm()

public void cal() throws IOException {
            FileOutputStream fos = new FileOutputStream("AlgeDis.txt");
            PrintStream prnt = new PrintStream(fos);
            coeff=new double[databuf.length][13];
            int column=databuf[0].length;
            int row=databuf.length;
            int count, x=1;
            double min, max;
            D=new double[column+1][7];
            double S[][]=new double[7][7];
            double Const[][]=new double[7][7];
              double temp[][]=new double [7][7];
              double L[][]=new double [7][7];
              double C[][]=new double [7][7];

              double invL[][] = new double [7][7];
              double dd[] = new double [7];
              double V[][] = new double [7][7];
              double sol[][] = new double [7][7];
              double tx,ty;
              int nrot=0;
              int npts=50;
```

```
 double XY[][] = new double[3][npts+1];
 double pvec[] = new double[7];

 double tan, aa, bb, cc, x1, x2, y1, y2;

//System.out.println("FPF mode");
        Const[1][3]=-2;
        Const[2][2]=1;
        Const[3][1]=-2;

 // Now first fill design matrix
formatter.setMaximumFractionDigits(4);
formatter.setMinimumFractionDigits(4);
for (int z=0; z < row; z++){
          //System.out.println(z+1+" round");
          count=0;
          for (int i=1; i <= (column-1); i++) {
                    if (databuf[z][i] < 0) {
                    //if databuf[z][i] are less than zero, we will skip that point
                    count++;
                    }
                    else{
                    double c=(Math.PI/180)*(360/(column-1))*i;

                    tx=(databuf[z][i])*(Math.cos(c));

                    ty=(databuf[z][i])*(Math.sin(c));

                    D[i][1]=tx*tx;
                    D[i][2]=tx*ty;
                    D[i][3]=ty*ty;
                    D[i][4]=tx;
                    D[i][5]=ty;
                    D[i][6]=1.0;
                                                                  }
      }


                    coeff[z][0]=databuf[z][0] ;
                    coeff[z][1]=count ;
                    for (int i=1; i <databuf[0].length; i++) {
                              if (databuf[z][i] < 0) {
                              }
                              else {
                                        x=i;
                                        break;
                                        }
                    }

                    min=databuf[z][x];
                    max=databuf[z][x];

                    for (int i=1; i <databuf[0].length; i++) {
                              if (databuf[z][i] < 0) {
                              }
                              else {
                                        if (min > databuf[z][i]) {
                                        min=databuf[z][i];
                                        }
```

```
                    if (max < databuf[z][i]) {
                        max=databuf[z][i];
                    }
                }
            }
//System.out.print("Max " + max + "\t" + "Min " + min + "\t");
            coeff[z][2]=max ;
            coeff[z][3]=min ;
if (count > 67){
System.out.println("We can't fit becouse the data is below 5 point");
}
else{
            //pm(Const,"Constraint");
            // Now compute scatter matrix  S
            A_TperB(D,D,S,column,6,column,6);
            //pm(S,"Scatter");
            choldc(S,6,L);
            //pm(L,"Cholesky");

            inverse(L,invL,6);
            //pm(invL,"inverse");

            AperB_T(Const,invL,temp,6,6,6,6);
            AperB(invL,temp,C,6,6,6,6);
            //pm(C,"The C matrix");

            jacobi(C,6,dd,V,nrot);
            //pm(V,"The Eigenvectors");  // OK
            //pv(dd,"The eigevalues");

            A_TperB(invL,V,sol,6,6,6,6);
            //pm(sol,"The GEV solution unnormalized");  // SOl

            // Now normalize them
            for (int j=1;j<=6;j++){  // Scan columns
                double mod = 0.0;
            for (int i=1;i<=6;i++)
                mod += sol[i][j]*sol[i][j];

            for (int i=1;i<=6;i++)
                sol[i][j] /=  Math.sqrt(mod);
            }
            //pm(sol,"The GEV solution");  // SOl

            double zero=10e-20;
            double minev=10e+20;
            int  solind=0;

            for (i=1; i<=6; i++)
            if (dd[i]<0 && Math.abs(dd[i])>zero)

            solind = i;

            // Now fetch the right solution
            for (int j=1;j<=6;j++)
            pvec[j] = sol[j][solind];
            //pv(pvec,"the solution");
            TransRota transRota=new TransRota(pvec[1], pvec[2],
```

```
                pvec[3], pvec[4], pvec[5], pvec[6], z, coeff);
//AlgeDis ad=new AlgeDis(pvec[1], pvec[2], pvec[3], pvec[4], pvec[5], pvec[6], d);

                prnt.println("The depth is " + coeff[z][0]);
                prnt.println("X Ellip" + "\t" + "Y Ellip" + "\t" +
                "X Data" + "\t" + "Y Data" + "\t" + "Dis. Diff");
                for (int i=5 ; i<=360 ; i=i+5 ) {
                tan=Math.tan(i*Math.PI/180);
                aa=pvec[1] + (pvec[2]*tan) + (pvec[3]*tan*tan);
                bb=pvec[4] + (pvec[5]*tan);
                cc=pvec[6];
                x1=(-bb + Math.sqrt((bb*bb) - (4*aa*cc))) / (2*aa);
                x2=(-bb - Math.sqrt((bb*bb) - (4*aa*cc))) / (2*aa);
                y1=tan*x1;
                y2=tan*x2;

                if (i>=5 & i<=90) {
                if (x1>0) {

                xy[i/5][0]=x1;          xy[i/5][1]=y1;

                //System.out.println("(" + x1 + "," + y1 + ")" );
                }
                else {

                xy[i/5][0]=x2;          xy[i/5][1]=y2;

                //System.out.println("(" + x2 + "," + y2 + ")" );
                }
                }
                else if (i>90 & i<=180) {
                if (x1<0) {

                xy[i/5][0]=x1;          xy[i/5][1]=y1;

                //System.out.println("(" + x1 + "," + y1 + ")" );
                }
                else {

                xy[i/5][0]=x2;          xy[i/5][1]=y2;

                //System.out.println("(" + x2 + "," + y2 + ")" );
                }
                }
                else if (i>180 & i<=270) {
                if (x1<0) {

                xy[i/5][0]=x1;          xy[i/5][1]=y1;

                //System.out.println("(" + x1 + "," + y1 + ")" );
                }
                else {

                xy[i/5][0]=x2;          xy[i/5][1]=y2;

                //System.out.println("(" + x2 + "," + y2 + ")" );
                }
```

```
}
else if (i>270 & i<=360) {
if (x1>0) {

xy[i/5][0]=x1;                  xy[i/5][1]=y1;

//System.out.println("(" + x1 + "," + y1 + ")" );
}
else {

xy[i/5][0]=x2;                  xy[i/5][1]=y2;

//System.out.println("(" + x2 + "," + y2 + ")" );
}
}
//System.out.println();

}
```

```
for(int i=1 ; i < disdiff.length ; i++) {
double algeDis=(pvec[1]*D[i][4]*D[i][4]) + (pvec[2]*D[i][4]*D[i][5]) + (pvec[3]*D[i][5]*D[i][5]) +
                (pvec[4]*D[i][4]) + (pvec[5]*D[i][5]) +(pvec[6]) ;
double dummy=Math.sqrt( Math.pow( (xy[i][0] - D[i][4]), 2 )+ Math.pow( (xy[i][1] - D[i][5]), 2) );
if (algeDis > 0) {
        disdiff[i]=dummy;
}
else if (algeDis == 0) {
        disdiff[i]=0;
}
else if (algeDis < 0) {
        disdiff[i]= -dummy;
}
}
for(int i=1 ; i < xy.length ; i++) {
        prnt.println(xy[i][0] + "\t" + xy[i][1] + "\t" + D[i][4] + "\t" + D[i][5]+ "\t" + disdiff[i]);

        //System.out.print(xy[i][j]+"\t");
        }
        prnt.println();
        //System.out.println();

}//End of else (count > 66)
}//End of  for loop of z
                PrintData print=new PrintData(coeff);

    }//End of cal()

}//End of Class CalCoeff
```

CalCoeffPS.java

```
import java.io.*;
import java.lang.Double;
import java.util.Vector;
import java.text.*;

class CalCoeffPS{
        InputData d;
        private int   i;
        private Vector points = new Vector(16,4);
        private static double FPFConstraint[][] = new double[7][7];
        NumberFormat formatter = NumberFormat.getNumberInstance();
        double databuf[][];
        double coeff[][];
        double D[][];
        double xy[][]=new double[72][2];

        CalCoeffPS(InputData d) {
                        databuf=new double[d.databuf.length][d.databuf[0].length];
                        for(int i=0;i<databuf.length;i++) {
              for(int j=0;j<databuf[0].length;j++)
                   databuf[i][j]=d.databuf[i][j];
                        }
        }// End of Constructor CalCoeff()

        CalCoeffPS(double temp[][]) {
                        databuf=new double[temp.length][temp[0].length];
                        for(int i=0;i<databuf.length;i++) {
              for(int j=0;j<databuf[0].length;j++)
                   databuf[i][j]=temp[i][j];
                        }
        }// End of Constructor CalCoeff()

        private void multMatrix(double m[][], double g[][], double mg[][]) {

    // First clear the return array
    for(int i=0; i<4; i++)
       for(int j=0; j<2; j++)
          mg[i][j]=0;

    // Perform the matrix math
    for(int i=0; i<4; i++)
       for(int j=0; j<2; j++)
          for(int k=0; k<4; k++)
             mg[i][j]=mg[i][j] + (m[i][k] * g[k][j]);
  }


  private void ROTATE(double a[][], int i, int j, int k, int l, double tau, double s){
          double g,h;
          g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);
          a[k][l]=h+s*(g-h*tau);
         }//End of ROTATE


  private void jacobi(double a[][], int n, double d[] , double v[][], int nrot){
          int j,iq,ip,i;
```

```
    double tresh,theta,tau,t,sm,s,h,g,c;

    double b[] = new double[n+1];
    double z[] = new double[n+1];

    for (ip=1;ip<=n;ip++) {
        for (iq=1;iq<=n;iq++) v[ip][iq]=0.0;
    v[ip][ip]=1.0;
    }
    for (ip=1;ip<=n;ip++) {
        b[ip]=d[ip]=a[ip][ip];
        z[ip]=0.0;
    }
    nrot=0;
    for (i=1;i<=50;i++) {
        sm=0.0;
        for (ip=1;ip<=n-1;ip++) {
            for (iq=ip+1;iq<=n;iq++)
                sm += Math.abs(a[ip][iq]);
        }
if (sm == 0.0) {
 /*   free_vector(z,1,n);
      free_vector(b,1,n);  */
 return;
}
if (i < 4)
 tresh=0.2*sm/(n*n);
else
 tresh=0.0;
for (ip=1;ip<=n-1;ip++) {
 for (iq=ip+1;iq<=n;iq++) {
  g=100.0*Math.abs(a[ip][iq]);
  if (i > 4 && Math.abs(d[ip])+g == Math.abs(d[ip])
    && Math.abs(d[iq])+g == Math.abs(d[iq]))
   a[ip][iq]=0.0;
  else if (Math.abs(a[ip][iq]) > tresh) {
   h=d[iq]-d[ip];
   if (Math.abs(h)+g == Math.abs(h))
    t=(a[ip][iq])/h;
   else {
    theta=0.5*h/(a[ip][iq]);
    t=1.0/(Math.abs(theta)+Math.sqrt(1.0+theta*theta));
    if (theta < 0.0) t = -t;
   }
   c=1.0/Math.sqrt(1+t*t);
   s=t*c;
   tau=s/(1.0+c);
   h=t*a[ip][iq];
   z[ip] -= h;
   z[iq] += h;
   d[ip] -= h;
   d[iq] += h;
   a[ip][iq]=0.0;
   for (j=1;j<=ip-1;j++) {
    ROTATE(a,j,ip,j,iq,tau,s);
   }
   for (j=iq+1;j<=n;j++) {
    ROTATE(a,ip,j,iq,j,tau,s);
```

```
          }
          for (j=1;j<=n;j++) {
            ROTATE(v,j,ip,j,iq,tau,s);
          }
          ++nrot;
        }
      }
    }
    for (ip=1;ip<=n;ip++) {
      b[ip] += z[ip];
      d[ip]=b[ip];
      z[ip]=0.0;
    }
  }
  //printf("Too many iterations in routine JACOBI");
}//End of jacobi()


// Perform the Cholesky decomposition
// Return the lower triangular L such that L*L'=A
private void choldc(double a[][], int n, double l[][]){
  int i,j,k;
  double sum;
  double p[] = new double[n+1];

  for (i=1; i<=n; i++) {
    for (j=i; j<=n; j++) {
      for (sum=a[i][j],k=i-1;k>=1;k--) sum -= a[i][k]*a[j][k];
      if (i == j) {
        if (sum<=0.0)
          // printf("\nA is not poitive definite!");
          {}
        else
          p[i]=Math.sqrt(sum); }
      else
        {
          a[j][i]=sum/p[i];
        }
    }
  }
  for (i=1; i<=n; i++)
    for (j=i; j<=n; j++)
      if (i==j)
        l[i][i] = p[i];
      else
        {
          l[j][i]=a[j][i];
          l[i][j]=0.0;
        }
}//End of choldc()


int inverse(double TB[][], double InvB[][], int N) {
  int k,i,j,p,q;
  double mult;
  double D,temp;
  double maxpivot;
  int npivot;
```

```
double B[][] = new double [N+1][N+2];
double A[][] = new double [N+1][2*N+2];
double C[][] = new double [N+1][N+1];
double eps = 10e-20;


for(k=1;k<=N;k++)
 for(j=1;j<=N;j++)
  B[k][j]=TB[k][j];

for (k=1;k<=N;k++)
 {
  for (j=1;j<=N+1;j++)
   A[k][j]=B[k][j];
  for (j=N+2;j<=2*N+1;j++)
   A[k][j]=(float)0;
  A[k][k-1+N+2]=(float)1;
 }
for (k=1;k<=N;k++)
 {
  maxpivot=Math.abs((double)A[k][k]);
  npivot=k;
  for (i=k;i<=N;i++)
   if (maxpivot<Math.abs((double)A[i][k]))
    {
     maxpivot=Math.abs((double)A[i][k]);
     npivot=i;
    }
  if (maxpivot>=eps)
   {    if (npivot!=k)
         for (j=k;j<=2*N+1;j++)
          {
           temp=A[npivot][j];
           A[npivot][j]=A[k][j];
           A[k][j]=temp;
          } ;
        D=A[k][k];
        for (j=2*N+1;j>=k;j--)
         A[k][j]=A[k][j]/D;
        for (i=1;i<=N;i++)
         {
          if (i!=k)
           {
            mult=A[i][k];
            for (j=2*N+1;j>=k;j--)
             A[i][j]=A[i][j]-mult*A[k][j] ;
           }
         }
       }
  else
   {  // printf("\n The matrix may be singular !!") ;
     return(-1);
    };
 }
/**  Copia il risultato nella matrice InvB  ***/
for (k=1,p=1;k<=N;k++,p++)
 for (j=N+2,q=1;j<=2*N+1;j++,q++)
  InvB[p][q]=A[k][j];
```

```
 return(0);
}        /*  End of INVERSE   */


private void AperB(double _A[][], double _B[][], double _res[][],
            int _righA, int _colA, int _righB, int _colB) {
        int p,q,l;
        for (p=1;p<=_righA;p++)
            for (q=1;q<=_colB;q++){
                _res[p][q]=0.0;
                for (l=1;l<=_colA;l++)
                    _res[p][q]=_res[p][q]+_A[p][l]*_B[l][q];

            }
}//End of AperB()


private void A_TperB(double _A[][], double _B[][], double _res[][],
             int _righA, int _colA, int _righB, int _colB) {
 int p,q,l;
 for (p=1;p<=_colA;p++)
             for (q=1;q<=_colB;q++){
                 _res[p][q]=0.0;
                 for (l=1;l<=_righA;l++)
                     _res[p][q]=_res[p][q]+_A[l][p]*_B[l][q];

    }
}//End of A_TperB()


private void AperB_T(double _A[][], double _B[][], double _res[][],
             int _righA, int _colA, int _righB, int _colB) {
 int p,q,l;
 for (p=1;p<=_colA;p++)
        for (q=1;q<=_colB;q++){
            _res[p][q]=0.0;
            for (l=1;l<=_righA;l++)
                _res[p][q]=_res[p][q]+_A[p][l]*_B[q][l];
        }
}//End of AperB_T()


public void pv(double v[], java.lang.String  str) {
formatter.setMaximumFractionDigits(7);
formatter.setMinimumFractionDigits(7);
        System.out.println();
        System.out.println("---------------------------- " + str + " ------------------------------");
        ystem.out.println(" " + formatter.format(v[1]) + "\t" + formatter.format(v[2]) + "\t" +
formatter.format(v[3]) + "\t" + formatter.format(v[4])+ "\t" + formatter.format(v[5]) + "\t" +
formatter.format(v[6]) );
        System.out.println("---------------------------------" + " End " + str + " ------------------------------
--------");
        System.out.println();
}//End of pv()


public void pm(double S[][], java.lang.String  str) {
        formatter.setMaximumFractionDigits(5);
```

```
        formatter.setMinimumFractionDigits(5);
        System.out.println();
        System.out.println("---------- " + str + " ----------");
        System.out.println(" " + formatter.format(S[1][1])+ "\t\t\t" + formatter.format(S[1][2])+ "\t\t\t"
+ formatter.format(S[1][3]) + "\t\t\t" + formatter.format(S[1][4]) + "\t\t\t" + formatter.format(S[1][5])+
"\t\t\t" + formatter.format(S[1][6]) );

        System.out.println(" " + formatter.format(S[2][1]) + "\t\t\t" + formatter.format(S[2][2]) +
"\t\t\t" + formatter.format(S[2][3]) + "\t\t\t" + formatter.format(S[2][4]) + "\t\t\t" + formatter.format(S
[2][5]) + "\t\t\t" + formatter.format(S[2][6]) );

        System.out.println(" " + formatter.format(S[3][1]) + "\t\t\t" + formatter.format(S[3][2]) +
"\t\t\t" + formatter.format(S[3][3]) + "\t\t\t" + formatter.format(S[3][4]) + "\t\t\t" + formatter.format(S
[3][5]) + "\t\t\t" + formatter.format(S[3][6]) );

        System.out.println(" " + formatter.format(S[4][1]) + "\t\t\t" + formatter.format(S[4][2]) +
"\t\t\t" + formatter.format(S[4][3]) + "\t\t\t" + formatter.format(S[4][4]) + "\t\t\t" + formatter.format(S
[4][5]) + "\t\t\t" + formatter.format(S[4][6]) );

        System.out.println(" " + formatter.format(S[5][1]) + "\t\t\t" + formatter.format(S[5][2]) +
"\t\t\t" + formatter.format(S[5][3]) + "\t\t\t" + formatter.format(S[5][4]) + "\t\t\t" + formatter.format(S
[5][5]) + "\t\t\t" + formatter.format(S[5][6]) );

        System.out.println(" " + formatter.format(S[6][1]) + "\t\t\t" + formatter.format(S[6][2]) +
"\t\t\t" + formatter.format(S[6][3]) + "\t\t\t" + formatter.format(S[6][4]) + "\t\t\t" + formatter.format(S
[6][5]) + "\t\t\t" + formatter.format(S[6][6]) );

        System.out.println("---------- " + " End  " + str +
" ----------");
        System.out.println();
    }   //End of pm()


    public void cal() throws IOException {
        coeff=new double[databuf.length][13];
        int column=databuf[0].length;
         int row=databuf.length;
         int count, x=1;
         double min, max;
         D=new double[column+1][7];
         double S[][]=new double[7][7];
         double Const[][]=new double[7][7];
         double temp[][]=new double [7][7];
         double L[][]=new double [7][7];
         double C[][]=new double [7][7];

         double invL[][] = new double [7][7];
         double dd[] = new double [7];
         double V[][] = new double [7][7];
         double sol[][] = new double [7][7];
         double tx,ty;
         int nrot=0;
         int npts=50;

         double XY[][] = new double[3][npts+1];
         double pvec[] = new double[7];
```

```
 double tan, aa, bb, cc, x1, x2, y1, y2;

//System.out.println("FPF mode");
        Const[1][3]=-2;
        Const[2][2]=1;
        Const[3][1]=-2;

 // Now first fill design matrix
formatter.setMaximumFractionDigits(4);
formatter.setMinimumFractionDigits(4);
for (int z=0; z < row; z++){
        //System.out.println(z+1+" round");
count=0;
 for (int i=1; i <= (column-1); i++) {
if (databuf[z][i] < 0) {
//if databuf[z][i] are less than zero, we will skip that point
count++;
}
else{
double c=(Math.PI/180)*(360/(column-1))*i;
                                                                tx=(databuf[z][i])*
(Math.cos(c));
ty=(databuf[z][i])*(Math.sin(c));

D[i][1]=tx*tx;
D[i][2]=tx*ty;
D[i][3]=ty*ty;
D[i][4]=tx;
D[i][5]=ty;
D[i][6]=1.0;
}
 }

coeff[z][0]=databuf[z][0] ;
coeff[z][1]=count ;
for (int i=1; i <databuf[0].length; i++) {
if (databuf[z][i] < 0) {
}
else {
        x=i;
        break;
        }
        }
        min=databuf[z][x];
        max=databuf[z][x];
for (int i=1; i <databuf[0].length; i++) {
        if (databuf[z][i] < 0) {
        }
        else {
        if (min > databuf[z][i]) {
                min=databuf[z][i];
                }
                if (max < databuf[z][i]) {
                max=databuf[z][i];
                }
        }
        }
        //System.out.print("Max " + max + "\t" + "Min " + min + "\t");
```

```
                coeff[z][2]=max ;
                coeff[z][3]=min ;

                if (count > 67){
                System.out.println("We can't fit becouse the data is below 5 point");
                }
                else{
                //pm(Const,"Constraint");
                // Now compute scatter matrix  S

                A_TperB(D,D,S,column,6,column,6);
                //pm(S,"Scatter");

                choldc(S,6,L);
                //pm(L,"Cholesky");
                inverse(L,invL,6);
                //pm(invL,"inverse");

                AperB_T(Const,invL,temp,6,6,6,6);
                AperB(invL,temp,C,6,6,6,6);
                //pm(C,"The C matrix");
                jacobi(C,6,dd,V,nrot);
                //pm(V,"The Eigenvectors");  // OK
                //pv(d,"The eigevalues");

                A_TperB(invL,V,sol,6,6,6,6);
                //pm(sol,"The GEV solution unnormalized");  // SOl
                // Now normalize them
                for (int j=1;j<=6;j++){  // Scan columns
                        double mod = 0.0;
                for (int i=1;i<=6;i++)

                        mod += sol[i][j]*sol[i][j];

                for (int i=1;i<=6;i++)

                        sol[i][j] /=  Math.sqrt(mod);
                }
                //pm(sol,"The GEV solution");  // SOl

                double zero=10e-20;
                double minev=10e+20;
                int  solind=0;
                for (i=1; i<=6; i++)
                if (dd[i]<0 && Math.abs(dd[i])>zero)
                solind = i;
                // Now fetch the right solution
                for (int j=1;j<=6;j++)
                pvec[j] = sol[j][solind];
                //pv(pvec,"the solution");
                TransRota transRota=new TransRota(pvec[1], pvec[2],
                                        pvec[3], pvec[4], pvec[5], pvec[6], z, coeff);
                }//End of else (count > 66)
        }//End of  for loop of z
                                PrintData print=new PrintData(coeff);

    }//End of cal()
}//End of Class CalCoeff
```

PhaseShiftOfCoeff.java

```
import java.io.*;

class PhaseShiftOfCoeff{
        InputData d1, d2;
   double keyForward;
   double keyBackward;
   int index1Forward;
   int index2Forward;
   int index1Backward;
   int index2Backward;
   double coeffDif[][][];
        double temp[][];
        int index;
        double bpost1[];
        double bpost[];

   PhaseShiftOfCoeff(InputData d1, InputData d2) throws IOException{
     this.d1=d1;
     this.d2=d2;
     System.out.println();
     System.out.println("The Method for checking Index is");

     //Find the forward index
     for(int i=0;i<d1.databuf.length;i++){
                        keyForward=d1.databuf[i][0];
                                for(int j=0; j<d2.databuf.length; j++) {
                                        double dif=d1.databuf[i][0]-d2.databuf[j][0];
            if ( dif != 0.0 ) {
                                                index1Forward= -1 ;
                                                index2Forward= -1 ;
                                        }
                                        else{
                                                index1Forward= i;
                                                index2Forward= j;
                                                //System.out.println(d1.databuf[i][0]+"\t"
+d2.databuf[j][0]) ;
                                                //System.out.println("The index is "+index+"\t" +
"check i value " + i) ;
                                                break;
                                        }
                                }//End of for loop j
                                if ( (index1Forward !=-1)&(index2Forward !=-1) )
                                        break;
                }//End of for loop i

                //Find the backward index
                for(int x=(d1.databuf.length-1); x>=0; x--) {
                keyBackward=d1.databuf[x][0];
                        for(int y=(d2.databuf.length-1); y>=0; y--) {
                                double dif=d1.databuf[x][0]-d2.databuf[y][0];
                                if (  dif != 0.0 ){
                                        index1Backward= -1 ;
                                        index2Backward= -1 ;
                                }
                                else{
                                        index1Backward= x;
```

```
                                        index2Backward= y;
                                        break;
                                }
                        }//End of for loop y
                        if ( (index1Backward !=-1)&(index2Backward !=-1) )
                                break;
                }//End of for loop x

                        System.out.println();
                        System.out.println("The key forwad value from Data1 is "+ keyForward);
                        System.out.println("The forward index of Data1 is "+index1Forward) ;
                        System.out.println("The forward index of Data2 is "+index2Forward) ;

                        System.out.println();
                        System.out.println("The key backward value from Data1 is "+
keyBackward);
                        System.out.println("The backward index of Data1 is "+index1Backward) ;
                        System.out.println("The backward index of Data2 is "+index2Backward) ;

                        //Method of rotation
                        rotateOfCoeff();

        }//End of Constructor PhaseShiftOfCoeff()

        public void rotateOfCoeff() throws IOException {
                index=( Math.abs(index1Forward-index1Backward) + 1);
                coeffDif=new double[index][73][7];
                int col2;

                //Find the difference of Co-Efficient parameter
                CalCoeffPS calCoeff1=new CalCoeffPS(d1);
                calCoeff1.cal();

                temp=new double[index][d2.databuf[0].length];
                for(int rowDiff=0; rowDiff<(coeffDif[0].length-1); rowDiff++) {//row of coeffDif
                        for(int row=0 ; row < index ; row++) {//row of Temp data for
rotate
                                int col3=1;
                                temp[row][0]=d2.databuf[index2Forward+row][0];
                                for(int col=1 ; col < d2.databuf[0].length ; col++)
{//column of Temp data for rotate
                                        col2=col + (rowDiff);
                                        if (col2 <= 72)
                                                temp[row][col]=d2.databuf
[index2Forward+row][col2];
                                        else {
                                                temp[row][col]=d2.databuf
[index2Forward+row][col3];
                                                col3++;
                                        }
                                }//End of column of Temp data for rotate
                        }//End of row of Temp data for rotate

                        //PrintData printTemp=new PrintData("Temporary Data", rowDiff,
temp);
                        CalCoeffPS calCoeff2=new CalCoeffPS(temp);
                        calCoeff2.cal();
                        for(int depDiff=0; depDiff<index; depDiff++) {//depth of coeffDif
```

```
                                        for(int colDiff=1; colDiff<coeffDif[0][0].length;
colDiff++) {//column of coeffDif

        //coeffDif[depDiff][rowDiff+1][colDiff]=calCoeff1.coeff[index1Forward+depDiff]
[colDiff+6] - calCoeff2.coeff[depDiff][colDiff+6];
                                                coeffDif[depDiff][rowDiff+1][colDiff]=(
calCoeff1.coeff[index1Forward+depDiff][colDiff+6] - calCoeff2.coeff[depDiff][colDiff+6] ) /
(calCoeff1.coeff[index1Forward+depDiff][colDiff+6]);
                                        }//End of column of coeffDif
                                }//End of depth of coeffDif
                }//End of row of coeffDif

                //Sum the difference of (A-A)^2 + (B-B)^2 + (C-C)^2 + (D-D)^2 + (E-E)^2 + (F-
F)^2
                for(int dep=0; dep<index; dep++) {//depth of sumDif
                        for(int row=1; row<coeffDif[0].length; row++) //row of sumDif
                                for(int col=1; col<coeffDif[0][0].length; col++)//column of sumDif
                                        coeffDif[dep][row][0]=coeffDif[dep][row][0] + Math.abs
(coeffDif[dep][row][col]);
                }

                //Find the best positon of the fit of each depth
                double min;
                int rowNumber=1;
                int dummy[]=new int[6];
                for(int dep=0; dep<index; dep++) {//depth of sumDif
                        dummy[0]=0; dummy[1]=0; dummy[2]=0; dummy[3]=0; dummy[4]=0;
dummy[5]=0;
                        for (int best5=0 ; best5 < 5 ; best5++) {
                                min=999.99;
                                for(int row=1; row<coeffDif[0].length; row++) { //row of sumDif
                                        if (row!=dummy[1]&&row!=dummy[2]&&row!=dummy
[3]&&row!=dummy[4]) {
                                                if (min > coeffDif[dep][row][0]){
                                                        min=coeffDif[dep][row][0];
                                                        rowNumber=row;
                                                }
                                        }
                                }
                                coeffDif[dep][0][best5]=rowNumber;
                                dummy[best5+1]=rowNumber;
                        }
                }

                //Find the best positon of the fit
                bpost=new double[73];
                bpost1=new double[5];
                int dummy2[]=new int[6];
                for(int col=1; col<bpost.length; col++) {
                                for(int dep=0; dep<index; dep++)
                                        bpost[col]=bpost[col] + coeffDif[dep][col][0];
                }
                int colNumber=1;
                for (int best5=0 ; best5 < 5 ; best5++) {
                        double minbpost=9999999999999.99;
                        for(int col=1; col<bpost.length; col++) {
```

```
                                        if (col!=dummy2[1]&&col!=dummy2[2]&&col!=dummy2
[3]&&col!=dummy2[4]&&col!=dummy2[5]) {
                                                if (minbpost > bpost[col]){
                                                        minbpost=bpost[col];
                                                        colNumber=col;
                                                }
                                        }
                                }
                                bpost1[best5]=colNumber;
                                dummy2[best5+1]=colNumber;
                        }
                        System.out.println("bpost1 " + bpost1[0] + " bpost2 " + bpost1[1] + " bpost3 " +
bpost1[2] + " bpost4 " + bpost1[3] +" bpost5 " + bpost1[4] );
                }//End of rotate()


}//End of Class PhaseShiftOfCoeff
```

TransRota.java

```
import java.text.*;

class TransRota{
    private double a, b, c, d, e, f;
                private double A, B, C, D, E, F;
    private double cot2, cos2, sin, cos, arccos, arcsin;
    private double Aprime, Bprime, Cprime, Dprime, Eprime, Fprime;
    private double h, k;
    private double aPower2, bPower2;
    private double aPower2Sqrt, bPower2Sqrt;

        NumberFormat formatter = NumberFormat.getNumberInstance() ;

    //TransRota(double pvec[]){
        TransRota(double a, double b, double c, double d, double e, double f, int z, double coeff[][]){

         A=a;           //a=pvec[1];
         B=b;           //b=pvec[2];
         C=c;           //c=pvec[3];
         D=d;           //d=pvec[4];
         E=e;           //e=pvec[5];
         F=f;           //f=pvec[6];

         formatter.setMaximumFractionDigits(7);
         formatter.setMinimumFractionDigits(7);

         //Rotation of Axis
         cot2=(a-c)/b;
         cos2=(a-c)/(Math.sqrt((b*b)+((a-c)*(a-c))));
                                sin=Math.sqrt((1-cos2)/2);
                                cos=Math.sqrt((1+cos2)/2);
                                arcsin=Math.asin(sin) * 180/(Math.PI);

         Aprime=(a*cos*cos)+(b*sin*cos)+(c*sin*sin);
         Bprime=(b*((cos*cos)-(sin*sin)))+(2*(c-a)*sin*cos);
         Cprime=(a*sin*sin)-(b*sin*cos)+(c*cos*cos);
         Dprime=(d*cos)+(e*sin);
         Eprime=(e*cos)-(d*sin);
         Fprime=(f);

         //Simplification of Translation
         h=2*Cprime*Dprime/(-4*Aprime*Cprime);
         k=2*Aprime*Eprime/(-4*Aprime*Cprime);

         //Maximum direction and Minimum direction
         aPower2=(-Fprime+(Aprime*h*h)+(Cprime*k*k))/Aprime;
         bPower2=(-Fprime+(Aprime*h*h)+(Cprime*k*k))/Cprime;
         aPower2Sqrt=Math.sqrt(aPower2);
         bPower2Sqrt=Math.sqrt(bPower2);

        coeff[z][4]=aPower2Sqrt ;
        coeff[z][5]=bPower2Sqrt ;
        //System.out.print("Degree " + formatter.format(arcsin) );
        coeff[z][6]=arcsin ;
        //System.out.println();
        formatter.setMaximumFractionDigits(9);
```

```
        formatter.setMinimumFractionDigits(9);

                                        coeff[z][7]=A ;
                                        coeff[z][8]=B ;
                                        coeff[z][9]=C ;
                                        coeff[z][10]=D ;
                                        coeff[z][11]=E ;
                                        coeff[z][12]=F ;
    }


    public void setA(double a){A=a;}
    public double getA(){return A;}
    public void setB(double b){B=b;}
    public double getB(){return B;}
    public void setC(double c){C=c;}
    public double getC(){return C;}
    public void setD(double d){D=d;}
    public double getD(){return D;}
    public void setE(double e){E=e;}
    public double getE(){return E;}
    public void setF(double f){F=f,}
    public double getF(){return F;}

}
```

PrintData.java

```
import java.text.*;
import java.io.*;

class PrintData {
        NumberFormat formatter = NumberFormat.getNumberInstance() ;
        FileOutputStream fos;
        PrintStream prnt;
        String name;

        //Print input data of ProgramAoverB
PrintData (String name, double data[][]) throws IOException{
                fos = new FileOutputStream("InputDataAoverB.txt");
                prnt = new PrintStream(fos);
                //System.out.println();
                prnt.println();
                //System.out.print(name + " in the array are:");
                prnt.print(name + " in the array are:");
                //System.out.println();
                prnt.println();
                for(int i=0;i<data.length;i++)      {
                        for(int j=0;j<data[0].length;j++)
                                //System.out.print(data[i][j]+"\t");
                                prnt.print(data[i][j]+"\t");
                        //System.out.println();
                        prnt.println();
                }
                //System.out.println();
                prnt.println();
}//End of Constructor printData()

//Print input data of ProgramPhaseShift
PrintData (String name1, double data1[][], String name2, double data2[][]) throws IOException{
                fos = new FileOutputStream("InputDataPhaseShift.txt");
                prnt = new PrintStream(fos);
                //System.out.println();
                prnt.println();
                //System.out.print(name1 + " in the array are:");
                prnt.print(name1 + " in the array are:");
                //System.out.println();
                prnt.println();
                for(int i=0;i<data1.length;i++)      {
                        for(int j=0;j<data1[0].length;j++)
                                //System.out.print(data1[i][j]+"\t");
                                prnt.print(data1[i][j]+"\t");
                        //System.out.println();
                        prnt.println();
                }
                //System.out.println();
                prnt.println();

                //System.out.print(name2 + " in the array are:");
                prnt.print(name2 + " in the array are:");
                //System.out.println();
                prnt.println();
                for(int i=0;i<data2.length;i++)      {
                        for(int j=0;j<data2[0].length;j++)
```

```
                                        //System.out.print(data2[i][j]+"\t");
                                        prnt.print(data2[i][j]+"\t");
                                }
                                //System.out.println();
                                prnt.println();
                        }
                        //System.out.println();
                        prnt.println();

}//End of Constructor printData()

//Print temporary data for testing
PrintData(String name, int toggle, double data[][]) throws IOException{
                        fos = new FileOutputStream("TempData.txt");
                        prnt = new PrintStream(fos);
                        //System.out.println();
                        prnt.println();
                        //System.out.println(name + " in the array are:" + toggle);
        prnt.println(name + " in the array are:" + toggle);
                         for(int i=0;i<data.length;i++)       {
                                for(int j=0;j<data[0].length;j++)
                                        //System.out.print(data[i][j]+"\t");
                                        prnt.print(data[i][j]+"\t");
                                //System.out.println();
                                prnt.println();
                        }
                        //System.out.println();
                        prnt.println();
}//End of Constructor printData()

//Print output data
PrintData(int index, double data[][][], double bpost[], String filename)  throws IOException{
                        name="PhaseShiftData" + filename + ".txt";
                        fos = new FileOutputStream(name);
                        prnt = new PrintStream(fos);
                        formatter.setMaximumFractionDigits(5);
                        formatter.setMinimumFractionDigits(5);
                        //System.out.println();
                        prnt.println();
                        prnt.println("The best position is");
                        prnt.println("bpost " + bpost[0]);
                        for(int col=1; col<bpost.length; col++) {
                                //System.out.println(bpost[col]);
                                prnt.print("sum dif [" + col + "] "+ formatter.format(bpost[col]) + "\t\t");
                                if (col==9|col==18|col==27|col==36|col==45|col==54|col==63)
                                        prnt.println();
                        }
                        //System.out.println();
                        prnt.println();
                        prnt.println();
                        formatter.setMaximumFractionDigits(9);
                        formatter.setMinimumFractionDigits(9);
                        //System.out.println("Sum the difference in the array are:");
                        prnt.println("Sum the difference in the array are:");
                        for(int i=0; i<index; i++) {
                                for(int j=0; j < data[0].length; j++) {
                                        for(int k=0; k< data[0][0].length; k++) {
          prnt.print("["+i+"]["+j + "]["+k+"]"+" "+" "+ formatter.format(data[i][j][k])+"      \t");
                                        }
```

```
                                        prnt.println();
                                }
                                prnt.println();
                        }
                        prnt.println();
}//End of Constructor printData()

//Print output data
PrintData(int index, double data[][][])  throws IOException{
                        fos = new FileOutputStream("PhaseShiftData.txt");
                        prnt = new PrintStream(fos);
                        formatter.setMaximumFractionDigits(9);
                        formatter.setMinimumFractionDigits(9);
                        prnt.println();
                        prnt.println("Sum the difference in the array are:");
                        for(int i=0; i<index; i++) {
                                for(int j=0; j < data[0].length; j++) {
                                        for(int k=0; k< data[0][0].length; k++) {
        prnt.print("["+i+"]["+j + "]["+k+"]"+" "+" "+ formatter.format(data[i][j][k])+"     \t");
                                        }
                                        prnt.println();
                                }
                                prnt.println();
                        }
                        prnt.println();
}//End of Constructor printData()

//Print Coefficient data which is A, B, C, D, E and F
PrintData(double data[][])  throws IOException{
                        fos = new FileOutputStream("CoeffData.txt");
                        prnt = new PrintStream(fos);
                        prnt.println();
                        prnt.println("Coeff in the array are: " );
                        prnt.println("Depth" + "\t\t" + "Count" + "\t" + "Max"  + "\t\t\t" + "Min"
                        + "\t\t\t" + "a" + "\t\t\t\t\t\t" + "b" + "\t\t\t\t\t\t" + "Degree"
                        + "\t\t\t\t\t"          + "A" + "\t\t\t\t\t\t" + "B" + "\t\t\t\t\t\t" + "C"
                        + "\t\t\t\t\t\t" + "D" + "\t\t\t\t\t\t" + "E" + "\t\t\t\t\t\t" + "F");
                        for(int i=0;i<data.length;i++) {
                                for(int j=0;j<data[0].length;j++)
                                        prnt.print(data[i][j]+"\t\t");
                                prnt.println();
                        }
}//End of Constructor printData()

//Print Summary Data
PrintData(double keyForward, double bpoCoeff[], double bpoDia[], double bpoRad[],
                        int index, double coeff[][][], double diameter[][][], double radii[][][])  throws
IOException{
                        fos = new FileOutputStream("SummaryData.txt");
                        prnt = new PrintStream(fos);
                        prnt.println();
                        prnt.println("Summary  Data: " );
                        prnt.println("Depth" + "\t\t" + "Code Index" + "\t\t" +
                                "Coefficient" + "\t\t" + "Diameter"  + "\t\t" + "Radii" +
                                "\t\t" + "Same Value");
                        prnt.println("\t\t\t" + "The Best Position" + "\t" +
                                bpoCoeff[0] + "\t" + bpoCoeff[1] + "\t" + bpoCoeff[2] + "\t" +
                                bpoCoeff[3] + "\t" + bpoCoeff[4] + "\t\t\t\t" + bpoDia[0]  + "\t\t\t" +
```

```
                        bpoRad[0]);
                prnt.println();
                double x=keyForward;
                for(int i=0; i<index; i++) {
                prnt.println( x + "\t\t" + "[" + i + "][0][0]" + "\t\t\t" + coeff[i][0][0] + "\t" +
                        coeff[i][0][1] + "\t" + coeff[i][0][2] + "\t" + coeff[i][0][3] + "\t" +
                        coeff[i][0][4] + "\t\t\t\t" + diameter[i][0][0] + "\t\t\t" + radii[i][0][0]);
                x=x-0.5;
                }
}//End of Constructor printData()
}//End of Class printData
```

Vitae

Sopon Leehiranyapong was born on November 6, 1973 in Bangkok, Thailand. He received his B.Eng. in Mechanical Engineering from the Faculty of Engineering, King Mongkut's Institute of Technology North Bangkok (KMIT'NB) in 1995. He has been a graduated student in the master degree program in Petroleum Engineering of the Department of Mining and Petroleum Engineering, Chulalongkorn University since 2001.