GPGN438 Senior Design:
Exploratory Seismic Data Analysis in the Field

Advisor/Client: Dave Hale
Team Members: Colton Kohnke

December 9th, 2013

Contents

List of Figures

List of Tables

1 Executive Summary

Seismic surveys take a significant amount of time and money to complete correctly. This project aims to develop software for the Colorado School of Mines (CSM) to aid with viewing seismic survey data and geometry in the field. The software will also be able to help catch errors in the field. It will also serve as a teaching tool for the students of the Colorado School of Mines Geophysics Field Camp to help them better understand seismic surveys.

The software will accomplish this by reading the SEGD files in real time from the recording truck along with the observation notes and the flag locations from a GPS unit. This data will then be plotted in map and section view in order to give a snapshot of the seismic survey. Basic processing can then be done quickly including gain, amplitude balancing, lowpass filtering, normal-moveout correction and surface-wave attenuation to create a dynamic stacked section.

The project will be completed by May 2014 and will be used as a replacement for ProMAX in the Summer 2014 Geophysics Field Camp.

2 Problem Statement

This project seeks to develop software in JAVA that aids with the exploration of seismic data in the field. The survey exploration includes the interactive display of survey geometry and seismographs. This software will also be able to perform simple processing tasks in the field. The program will serve as a bridge from the seismic crew to the students of Field Camp and serve as a teaching tool for the student's understanding of the seismic survey.

3 Introduction & Background

Geophysical data collection is an expensive operation that costs both time and money. In the field, it is in the data collector's best interest to make sure that the data collected is good quality.

ProMAX is the current system used in the field to view the seismic data coming in from the recording station. This system, while good for in depth processing, is not robust enough to quickly display survey data and interactively display the survey geometry. The CSM Field Camp is unique in that they surveys are not conventional, that is, the surveyors are trying parameters or setups that aren't typical of a corporate survey.

There can also be a lack of understanding between the students and the seismic survey with regards to how the data is collected in the field. This software will seek to help the students at Field Camp better understand the seismic survey while they are in the field. The software can also be used as a teaching tool to the students in Field Camp to help them better understand the aspects of a seismic surveys.

4 Deliverables to Client

The deliverables of this project include two main items. The first is a program written in JAVA that accomplishes the design objectives outlined in the next section. The second deliverable will be a documentation of the code in both standard notation and as a PDF "how-to" style guide.

5 Design Objectives

5.1 Interactive Display of Survey Geometry

The first objective is to be able to import station (flag) locations. This needs to be accomplished from a variety of GPS sources including CSV, Excel, GPX and tab-delimited text files. The GPS coordinates need to then be converted to UTM for easier processing. Once these stations have been imported and converted, they need to have the option of being exported so the conversion does not need to be applied again. The suggested columns are listed in Table 1. Once the conversion is applied, the next step is to plot the locations in map view.

Table 1: GPS Export Spreadsheet Fields

StationID	UTM Easting	UTM Northing	Elevation

The second objective is to be able to compute source and receiver locations. This is done for each valid shot FFID (Field File ID) by using the observer files to determine the source station number, live recording channels and receiver station numbers. The station locations are then used to find the location of the source and the elevation is read from a USGS Digital Elevation Map. All of this data is then stored into a spreadsheet file with columns denoted in Table 2. Some FFIDs correspond to bad shots and will need to be ignored dynamically by the program routines.

Table 2: Source-Receiver Locations Spreadsheet Fields

FFID	SEGD Filename	Source	StationID	Channel Number	SourceXYZ	ReceiverXYZ

Once the data has been extracted from the observation files, the program needs to be able to plot source, receiver and midpoint locations in map view. In map view, there needs to be able to plot a piecewise curve that represents the seismic line. The program also needs to be able to plot elevation profiles along the source, receiver and midpoint lines. Finally, there needs to be a slider to select and show points corresponding to each FFID that will provide a graphical history of the seismic survey and help to catch mistakes.

5.2 Interactive Display of Seismograms

The first step to displaying the seismograms is to convert the SEGD files (bytes) to an array of an array of floats (2D flaot array). Once this is accomplished, the seismograms need to be interactively displayed by using the FFID slider. This will display all seismograms for a particular FFID.

Another method of display will be to display all seismograms within a circle that the user draws in map view.

Two more sliders, called the min-max sliders, that are populated with the minimum and maximum offsets will be implemented. As these sliders move, the program will plot all seismograms with offsets within the range of the min-max sliders.

All of this is useless unless the data can actually be seen. Therefore, interactive controls for gain, amplitude balancing and lowpass filtering (Butterworth) need to be implemented and the

results plotted. This will be done by using sliders, or interactive graphical menus.

5.3 Simple Processing

In the field, there is often the need to do basic processing in order to see a better picture of what is happening in the subsurface. Tools for basic processing of the data will be included in the developed software. The guaranteed processing tools will be for surface-wave attenuation and normal-moveout correction, but more are being investigated.

5.4 Other Requirements

Speed is a top priority when working in the field. Decisions are typically made quickly in order to move the survey along at a reasonable rate. If this software is fast enough, it can be used as a supplement to those decisions.

The user-interface must also be easy to use. The layout of the controls must be logical and follow a sequential order. These different features must be documented in the end-user documentation and be consistent in their implementation.

6 Decision-making & Assessment of Alternative Approaches

The decisions during the main software development section of this project deal with efficiency. It is possible that choices in the field need to be made very quickly and as a result, the software needs to be able to perform operations efficiently to keep pace. If the software is called to display a seismic trace, it needs to be able to find the data and display it in the least amount of operations. Similarly, if an operation is applied to a set of data, it should be applied in the least amount of compute time.

The software is not currently optimized for speed because that would slow down development. However, specific places in the software have been marked for efficiency improvements during the testing phase. One area that has been marked for optimization is the storage of the data from the SEGD files. Currently, shots are stored as a JAVA ArrayList and is traversed linearly. This causes an $O(n^2)$ search time where n is the length of the array. A much faster implementation would be a Binary Search Tree which has $O(n)$ traversal time. Another option is implementing a hast table, which returns a value in near constant time, $O(1)$.

The decisions during the later stages will revolve around end user usability. These choices will include aesthetic placement of tools in the software to create a logical progression of tasks for the end user. The decisions in this section will also revolve around how to lay out the documentation for the end user. These decisions will be made during the final stages of the project and will likely involve bringing in fresh eyes to test the documented work flows.

7 Design Solution

The tool that is being used in this design project is simulation. Each method that is written is tested by calling the method on the 2013 Field Camp data. These simulations provide the developer

with real data to test the software against and with speed/efficiency benchmarks to try to improve.

In addition, before a feature is integrated into the larger code, it is tested against itself for errors and correct implementation. This speeds up the development process because it limits the area where bugs can occur at any given time.

8 Implementation Plan

8.1 Safety

The main safety concern of this project is the ergonomics of the developer. This project does not require field work or manual labor in the traditional sense. However, it does require a significant amount of time to be spent coding at a computer which may cause irritation of the wrist or forearm. The developer will mitigate the risk of encountering this problem by taking breaks, stretching and using correct posture while at the workstation.

8.2 Timeline

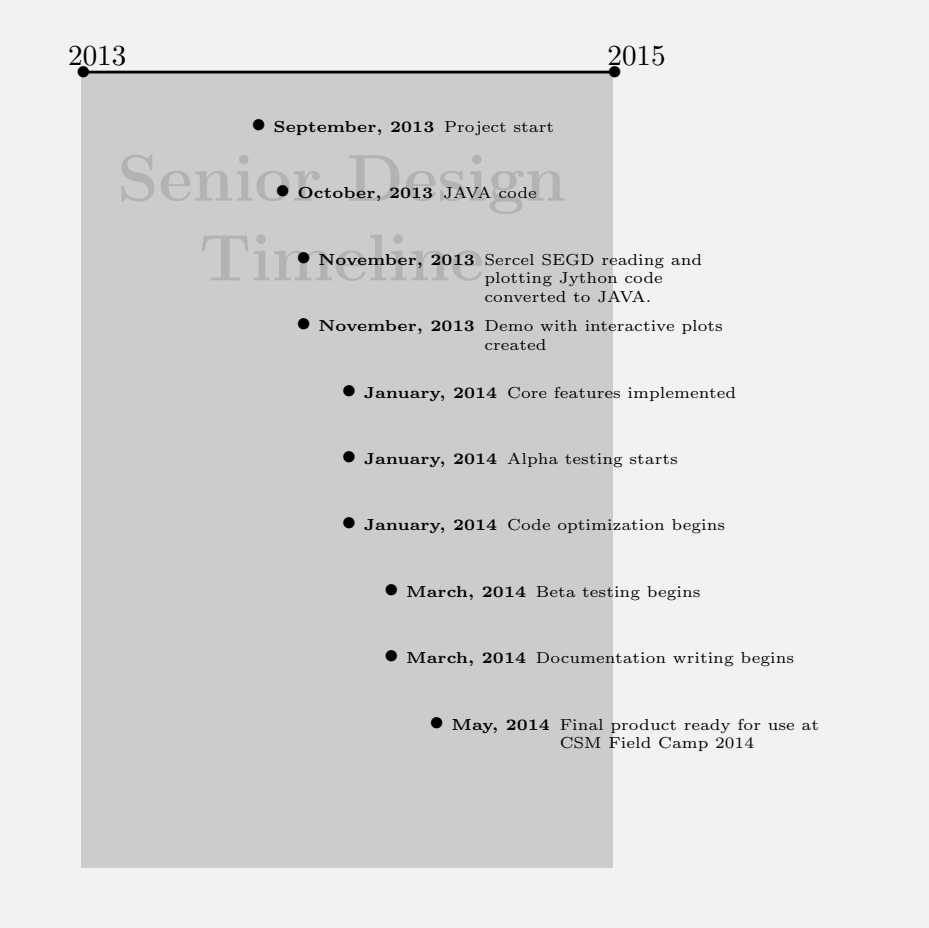

Figure 1: Senior Design Project Timeline

8.3 Division of Responsibility & Effort

The project team is composed of a single developer, so the whole responsibility is on the only member of the team.

8.4 Budget 1: Actual

No budget is required for the actual project as the project does not require any traveling or other expenditures outside the realm of a standard class at CSM.

8.5 Budget 2: Professional

Ideally to get a working product to the client, the following budget would be used in a professional setting.

Table 3: The Professional Budget

Item	Base Cost	Quantity	Total	Notes
Developers	\$20.00/hr	360	\$7,200	2 Devs for 4 work weeks.
Continued Support	\$15.00/hr	260	\$3,900	1 Dev at 5 hours/week for 1 year.
Base Total			\$11,100	
Overshoot			\$1,110	10% addition
Total			\$12,210	

9 Implementation

9.1 Data Acquisition

The data used to help develop the software is the 2012 CSM Geophysics Field Camp seismic data provided by Dr. Dave Hale. No further data acquisition is planned for this project.

9.2 Data Reduction, Analysis, Interpretation, Integration

This project revolves more around the display of field data than the analysis and active interpretation of the data. Certain reductions will be implemented to the data in an interactive way. These reductions include gain, lowpass filtering and amplitude balancing. The software will also serve as a hub for a place to view all the seismic data for a survey and integrate it together to make plots of the seismic line, such as a brute stack.

9.3 Error Analysis

The error analysis of this project will be related to the processing and display of imported data. These processes include making sure tools like the gain and filtering are correct, and the plotting of the data is accurate.

Currently the code for the gaining and filtering the data has been written, but it has not been tested against a known working algorithm. The shot plotting function has been tested against the known working Jython code and has been confirmed to work as expected.

9.4 Results

The results so far of the project have been the implementation of software that imports GPS points from Lat/Lon and transforms them into UTM coordinates before plotting them in map view. The software also imports SEGD files from the Sercel format and plots the nearest shot interactively as the user moves a mouse around the map view screen. The GPS and SEGD data can be imported and exported to files for easier imports later. Basic tools for manipulating the data have also been implemented, including gaining the data and passing the data through a lowpass filter.

9.5 Discussion and Conclusions

So far, everything has been running well and the setbacks have been mitigated by meetings with the advising professor.

The main pitfall the team has faced on this project has been time. As a single developer, I have not been able to dedicate enough time to this project and I should be much farther along at this point in time. The second half of the project will be more strenuous, but the results will come faster because my own coding ability has increased throughout this project.

10 Recommendations for Future Work

The next step for the project is to read the observation reports and finish implementing the main core features to create an alpha product. From there, code optimization can begin to occur and more efficient testing can begin.

After the software has been optimized for speed, it then needs to be optimized for usability. This will allow for the end-user to efficiently use the software.

11 References

References

Hale, Dave (2013). *Mines JTK Documentation*. URL: `http://dhale.github.io/jtk/api/index.html`.

Sercel (2011). *428XL User's Manual Vol. 2*.

Wikipedia (2013). *Binary Search Tree*. URL: `http://en.wikipedia.org/wiki/Binary_search_tree`.

12 Resumes

13 Appendices

13.1 Appendix A: JAVA Code

1. MPoint.java

```
1  import java.util.*;
2
3  public class MPoint {
4    // from xyz coord
5    MPoint(double x, double y, boolean UTM){
6      this.x = x;
7      this.y = y;
8    }
9
10   // from xyz coord
11   MPoint(int stationID, double x, double y, double z, boolean UTM){
12     this.stationID = stationID;
13     this.x = x;
14     this.y = y;
15     this.z = z;
16   }
17
18   MPoint(int stationID, double x, double y, int UTMzone, boolean UTM){
19     this.stationID = stationID;
20     this.x = x;
21     this.y = y;
22     this.UTMzone = UTMzone;
23   }
24
25   MPoint(int stationID, double x, double y, double z, int UTMzone, boolean UTM)
         {
26     this.stationID = stationID;
27     this.x = x;
28     this.y = y;
29     this.z = z;
30     this.UTMzone = UTMzone;
31   }
32
33   // from xy coord
34   MPoint(int stationID, double x, double y, boolean temp, boolean UTM){
35     this.stationID = stationID;
36     this.x = x;
37     this.y = y;
38   }
39
40   MPoint(int stationID, double lat, double lon){
41     this.stationID = stationID;
42     this.lat = lat;
43     this.lon = lon;
44   }
45
46   MPoint(int stationID, double lat, double lon, double z){
47     this.stationID = stationID;
48     this.lat = lat;
49     this.lon = lon;
```

```
50      this.z = z;
51    }
52
53
54    public double xyDist(MPoint p){
55      return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));
56    }
57
58    public double xDist(MPoint p){
59      return Math.sqrt((x-p.x)*(x-p.x));
60    }
61
62    public double yDist(MPoint p){
63      return Math.sqrt((y-p.y)*(y-p.y));
64    }
65
66    public double zDist(MPoint p){
67      return Math.sqrt((z-p.z)*(z-p.z));
68    }
69
70    public double xyzDist(MPoint p){
71      return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y)+(z-p.z)*(z-p.z));
72    }
73
74    public int stationID;
75    public double x, y, z;
76    public double lat, lon;
77    public int UTMzone;
78    public boolean selected;
79  }
80
81  class MPointComp implements Comparator<MPoint>{
82
83    //@Override
84    public int compare(MPoint p1, MPoint p2) {
85      if(p1.stationID > p2.stationID){
86         return 1;
87      } else {
88         return -1;
89      }
90    }
91  }
```

2. PlotTest.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.io.*;
4  import java.util.ArrayList;
5  import javax.swing.*;
6  import java.util.Scanner;
7
8  import edu.mines.jtk.awt.*;
9  import edu.mines.jtk.dsp.*;
10 import edu.mines.jtk.util.Cdouble;
11 import edu.mines.jtk.mosaic.*;
12 import static edu.mines.jtk.util.ArrayMath.*;
13
14 public class PlotTest{
```

```
15
16    public static void main(String[] args){
17      SwingUtilities.invokeLater(new Runnable() {
18        public void run() {
19          new PlotTest();
20        }
21      });
22    }
23
24    // Location and size of overlay plot.
25    private static final int M_X = 100;
26    private static final int M_Y = 0;
27    private static final int M_WIDTH = 520;
28    private static final int M_HEIGHT = 550;
29
30    // Location and size of response plot.
31    private static final int RP_X = M_X+M_WIDTH;
32    private static final int RP_Y = 0;
33    private static final int RP_WIDTH = 520;
34    private static final int RP_HEIGHT = 550;
35
36    // Plot of source/receivers
37    // private ArrayList<MPoint> _shots;
38    private ArrayList<MPoint> _recs;
39    public ArrayList<MPoint> _gps;
40    public ArrayList<Segdata> _segd;
41    private BasePlot _bp;
42    private ResponsePlot _rp;
43    private Waypoints wPoints;
44    private Segd seg;
45
46
47    private PlotTest(){
48      // _shots = new ArrayList<MPoint>(0);
49      _gps = new ArrayList<MPoint>(0);
50      _segd = new ArrayList<Segdata>(0);
51      _bp = new BasePlot();
52      _rp = new ResponsePlot();
53    }
54
55    private void addMPoint(MPoint p) {
56      _recs.add(p);
57      _bp.updateBPView();
58    }
59
60
61  ////////////////////////////////////////////////////////////////////////
62
63    private class BasePlot {
64
65      private PlotFrame _plotFrame;
66      private PlotPanel _plotPanel;
67      private PointsView _baseView;
68
69      private BasePlot() {
70
71        // The plot panel.
72        _plotPanel = new PlotPanel();
73        _plotPanel.setTitle("Base Plot Test");
```

```
74          _plotPanel.setHLabel("Easting (UTM)");
75          _plotPanel.setVLabel("Northing (UTM)");
76          _plotPanel.setHLimits(317600,320600); //TODO: plot displays E+06 for
                large ints
77          _plotPanel.setVLimits(4121800,4123600);    //TODO: plot displays E+06 for
                large ints
78
79          // A grid view for horizontal and vertical lines (axes).
80          _plotPanel.addGrid("H0-V0-");
81
82          // A plot frame has a mode for zooming in tiles or tile axes.
83          _plotFrame = new PlotFrame(_plotPanel);
84          TileZoomMode tzm = _plotFrame.getTileZoomMode();
85
86          // We add two more modes for editing poles and zeros.
87          ModeManager mm = _plotFrame.getModeManager();
88          RoamMode rm = new RoamMode(mm); // roam and plot
89          // PoleZeroMode zm = new PoleZeroMode(mm,false);  // for zeros
90
91          // The menu bar includes a mode menu for selecting a mode.
92          JMenu fileMenu = new JMenu("File");
93          fileMenu.setMnemonic('F');
94          fileMenu.add(new SaveAsPngAction(_plotFrame)).setMnemonic('a');
95          fileMenu.add(new ExitAction()).setMnemonic('x');
96
97          JMenu modeMenu = new JMenu("Mode");
98          modeMenu.setMnemonic('M');
99          modeMenu.add(new ModeMenuItem(tzm));
100         modeMenu.add(new ModeMenuItem(rm));
101
102         JMenu toolMenu = new JMenu("Tools");
103         toolMenu.setMnemonic('T');
104         toolMenu.add(new GetFlagsFromHH()).setMnemonic('f');
105         toolMenu.add(new GetDEM(_plotPanel)).setMnemonic('g');
106         toolMenu.add(new ExportFlagsToCSV()).setMnemonic('e');
107         toolMenu.add(new ImportSegdDir()).setMnemonic('s');
108
109         JMenuBar menuBar = new JMenuBar();
110         menuBar.add(fileMenu);
111         menuBar.add(modeMenu);
112         menuBar.add(toolMenu);
113
114         _plotFrame.setJMenuBar(menuBar);
115
116         // The tool bar includes toggle buttons for selecting a mode.
117         JToolBar toolBar = new JToolBar(SwingConstants.VERTICAL);
118         toolBar.setRollover(true);
119         toolBar.add(new ModeToggleButton(tzm));
120         toolBar.add(new ModeToggleButton(rm));
121         _plotFrame.add(toolBar,BorderLayout.WEST);
122
123         // Initially, enable editing of poles.
124         // pm.setActive(true);
125
126         // Make the plot frame visible.
127         _plotFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
128         _plotFrame.setLocation(M_X,M_Y);
129         _plotFrame.setSize(M_WIDTH,M_HEIGHT);
130         _plotFrame.setFontSizeForPrint(8,240);
```

```
131         _plotFrame.setVisible(true);
132
133       }
134
135       // Makes poles view consistent with the list of poles.
136       private void updateBPView() {
137         int np = wPoints._gps.size();
138         float[] xp = new float[np];
139         float[] yp = new float[np];
140         for (int ip=0; ip<np; ++ip) {
141           MPoint p = wPoints._gps.get(ip);
142           xp[ip] = (float)p.x;
143           yp[ip] = (float)p.y;
144         }
145         if (_baseView==null) {
146           _baseView = _plotPanel.addPoints(xp,yp);
147           _baseView.setMarkStyle(PointsView.Mark.CROSS);
148           _baseView.setLineStyle(PointsView.Line.NONE);
149         } else {
150           _baseView.set(xp,yp);
151         }
152       }
153
154   }
155
156   ////////////////////////////////////////////////////////////////////////
157
158   private class ResponsePlot {
159
160     private PlotPanel _plotPanelH;
161     private PlotFrame _plotFrame;
162     private SequenceView _hView;
163     private PointsView _pView;
164     public SimplePlot sp;
165
166
167     // The amplitude response can be in decibels (db).
168     private ResponsePlot() {
169
170       // One plot panel for the impulse response.
171       _plotPanelH = new PlotPanel();
172       _plotPanelH.setHLabel("Station");
173       _plotPanelH.setVLabel("Time (s)");
174       _plotPanelH.setTitle("Shot");
175
176       // This first update constructs a sequence view for the impulse
177       // response, and a points view for amplitude and phase responses.
178       // updateViews();
179
180       _plotFrame = new PlotFrame(_plotPanelH);
181
182       // The menu bar.
183       JMenu fileMenu = new JMenu("File");
184       fileMenu.setMnemonic('F');
185       fileMenu.add(new SaveAsPngAction(_plotFrame)).setMnemonic('a');
186       fileMenu.add(new ExitAction()).setMnemonic('x');
187       JMenuBar menuBar = new JMenuBar();
188       menuBar.add(fileMenu);
189
```

```
190         _plotFrame.setJMenuBar(menuBar);
191
192         // Make the plot frame visible.
193         _plotFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
194         _plotFrame.setLocation(RP_X,RP_Y);
195         _plotFrame.setSize(RP_WIDTH,RP_HEIGHT);
196         _plotFrame.setFontSizeForPrint(8,240);
197         _plotFrame.setVisible(false);
198         sp = new SimplePlot(SimplePlot.Origin.UPPER_LEFT);
199         sp.setSize(900,900);
200         sp.setVLabel("Time (s)");
201
202       }
203
204       public void updateRP(Segdata seg){
205         int n1 = seg.f[0].length;
206         int n2 = seg.f.length;
207         Sampling s1 = new Sampling(n1, 0.001, 0.0);
208         Sampling s2 = new Sampling(n2, 1.0, seg.rpf);
209         if(s2.getDelta() ==1.0)
210           sp.setHLabel("Station");
211         else
212           sp.setHLabel("Offset (km)");
213         sp.setHLimits(seg.rpf, seg.rpl);
214         sp.setTitle("Shot "+seg.sp);
215         PixelsView pv = sp.addPixels(s1,s2,seg.f);
216         pv.setPercentiles(1,99);
217       }
218
219     }
220
221     ////////////////////////////////////////////////////////////////////////
222
223     private class RoamMode extends Mode {
224       public RoamMode(ModeManager modeManager) {
225         super(modeManager);
226           setName("Roaming Mode");
227           // setIcon(loadIcon(PolesAndZerosDemo.class,"Poles16.png"));
228           setMnemonicKey(KeyEvent.VK_R);
229           setAcceleratorKey(KeyStroke.getKeyStroke(KeyEvent.VK_R,0));
230           setShortDescription("Roaming Mode");
231         }
232
233       // When this mode is activated (or deactivated) for a tile, it simply
234       // adds (or removes) its mouse listener to (or from) that tile.
235       protected void setActive(Component component, boolean active) {
236         if (component instanceof Tile) {
237           if (active) {
238             component.addMouseListener(_ml);
239           } else {
240             component.removeMouseListener(_ml);
241           }
242         }
243       }
244
245       private boolean _moving; // if true, currently moving
246       private Tile _tile; // tile in which editing began
247
248       private MouseListener _ml = new MouseAdapter() {
```

```
249         public void mousePressed(MouseEvent e) {
250           if(beginMove(e)){
251             _moving = true;
252             _tile.addMouseMotionListener(_mml);
253           }
254         }
255         public void mouseReleased(MouseEvent e) {
256           _tile.removeMouseMotionListener(_mml);
257           endMove(e);
258           _moving = false;
259         }
260       };
261       // Handles mouse dragged events.
262       private MouseMotionListener _mml = new MouseMotionAdapter() {
263         public void mouseDragged(MouseEvent e) {
264           if (_moving)
265             duringMove(e);
266         }
267       };
268
269       private boolean beginMove(MouseEvent e){
270         _tile = (Tile)e.getSource();
271         int x = e.getX();
272         int y = e.getY();
273         MPoint nearest = getNearestGPS(x,y);
274         return true;
275       }
276
277       private void duringMove(MouseEvent e) {
278         int x = e.getX();
279         int y = e.getY();
280         //System.out.println("x: " + x + " y: " + y);
281         MPoint gpsNear = getNearestGPS(x,y);
282         //System.out.println(gpsNear.stationID);
283         Segdata segNear = getNearestSegdata(gpsNear.stationID);
284         //System.out.println(segNear.sp);
285         _rp.updateRP(segNear);
286       }
287
288       private void endMove(MouseEvent e) {
289         duringMove(e);
290       }
291
292       private MPoint getNearestGPS(int x, int y){
293         Transcaler ts = _tile.getTranscaler();
294         Projector hp = _tile.getHorizontalProjector();
295         Projector vp = _tile.getVerticalProjector();
296         double xu = ts.x(x);
297         double yu = ts.y(y);
298         double xv = hp.v(xu);
299         double yv = vp.v(yu);
300         MPoint test = new MPoint(xv, yv, true);
301         MPoint near = wPoints._gps.get(0);
302         MPoint fin = wPoints._gps.get(0);
303         double d = near.xyDist(test);
304         for(int i = 1; i<wPoints._gps.size(); ++i){
305           near = wPoints._gps.get(i);
306           if(near.xyDist(test) < d){
307             fin = wPoints._gps.get(i);
```

```
308             d = fin.xyDist(test);
309           }
310         }
311         return fin;
312       }
313
314       private Segdata getNearestSegdata(int stationID){
315         Segdata seg1 = seg._segd.get(0);
316         Segdata seg2 = seg._segd.get(0);
317         int d1 = abs(seg1.sp-stationID);
318         for(int i=1; i<seg._segd.size(); ++i){
319           seg2 = seg._segd.get(i);
320           int d2 = abs(seg2.sp-stationID);
321           if(d2 < d1){
322             seg1 = seg2;
323             d1 = abs(seg1.sp-stationID);
324           }
325         }
326         return seg1;
327       }
328
329     }
330
331     ////////////////////////////////////////////////////////////////////////
332
333     // Actions common to both plot frames.
334     private class ExitAction extends AbstractAction {
335       private ExitAction() {
336         super("Exit");
337       }
338       public void actionPerformed(ActionEvent event) {
339         System.exit(0);
340       }
341     }
342     private class SaveAsPngAction extends AbstractAction {
343       private PlotFrame _plotFrame;
344       private SaveAsPngAction(PlotFrame plotFrame) {
345         super("Save as PNG");
346         _plotFrame = plotFrame;
347       }
348       public void actionPerformed(ActionEvent event) {
349         JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
350         fc.showSaveDialog(_plotFrame);
351         File file = fc.getSelectedFile();
352         if (file!=null) {
353           String filename = file.getAbsolutePath();
354           _plotFrame.paintToPng(300,6,filename);
355         }
356       }
357     }
358     private class GetDEM extends AbstractAction {
359       private GetDEM(PlotPanel plotPanel){
360         super("Get USGS Elevation");
361
362       }
363       public void actionPerformed(ActionEvent event){
364         //TODO
365       }
366     }
```

```
367   private class GetFlagsFromHH extends AbstractAction {
368     private GetFlagsFromHH(){
369       super("Get HandHeld GPS");
370     }
371     public void actionPerformed(ActionEvent event) {
372       JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
373       fc.showOpenDialog(null);
374       File f = fc.getSelectedFile();
375       wPoints = new Waypoints(f);
376       _bp.updateBPView();
377     }
378   }
379   private class ExportFlagsToCSV extends AbstractAction {
380     private ExportFlagsToCSV(){
381       super("Export GPS to CSV");
382
383     }
384     public void actionPerformed(ActionEvent event) {
385         JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
386         fc.showSaveDialog(null);
387         File f = fc.getSelectedFile();
388         wPoints.exportToCSV(f);
389       }
390     }
391   private class ImportSegdDir extends AbstractAction {
392     private ImportSegdDir(){
393       super("Import Segd Directory");
394
395     }
396     public void actionPerformed(ActionEvent event) {
397         JFileChooser fc = new JFileChooser(System.getProperty("user.dir"));
398         fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
399         fc.showSaveDialog(null);
400         File f = fc.getSelectedFile();
401         seg = new Segd(f.getAbsolutePath());
402         System.out.println("SEGD IMPORTED");
403       }
404     }
405
406
407   ////////////////////////////////////////////////////////////////////////
408
409
410
411 }
```

3. Segd.java

```
1  import java.awt.*;
2  import java.io.*;
3  import java.lang.*;
4  import java.nio.*;
5  import java.util.*;
6  import javax.swing.*;
7
8  import edu.mines.jtk.awt.*;
9  import edu.mines.jtk.dsp.*;
10 import edu.mines.jtk.interp.*;
11 import edu.mines.jtk.io.*;
```

```
12 import edu.mines.jtk.mosaic.*;
13 import edu.mines.jtk.ogl.Gl.*;
14 import edu.mines.jtk.util.*;
15 import static edu.mines.jtk.util.ArrayMath.*;
16
17 public class Segd{
18
19   public Segd(String segdDir){
20     this.segdDir = segdDir;
21     _segd = new ArrayList<Segdata>(0);
22     readLineSegd();
23   }
24
25   public void readLineSegd(){
26     try{
27     File[] segdList = (new File(segdDir)).listFiles();
28     //File[] segdList = new File[1];
29     //segdList[0] = new File(segdDir+"/00000001.00000293.segd");
30     int nshot = segdList.length;
31     for(int i=0; i<segdList.length; ++i){
32       System.out.println(segdList[i].getName());
33       Segdata seg = readSegd(segdList[i]);
34       //System.out.println("sln ="+sln+" spn ="+spn+" rpf ="+rpf+" rpl ="+rpl);
35       int n1 = seg.f[0].length;
36       int n2 = seg.f.length;
37       Sampling s1 = new Sampling(n1, 0.001, 0.0);
38       Sampling s2 = new Sampling(n2, 1.0, seg.rpf);
39       //lowpass2(seg.f);
40       //tpow2(seg.f);
41       //gain2(seg.f);
42       //plot(s1,s2,seg,"Shot "+seg.sp);
43       if(!(seg.sp < 0))
44         _segd.add(seg);
45     }
46       Collections.sort(_segd, new SegdataComp());
47     }catch(IOException e){
48       System.out.println(e);
49     }
50   }
51
52   public Segdata readSegd(File segdFile) throws IOException{ //return tiltdata-
       esque
53     byte[] gh = zerobyte(32); // general header
54     byte[] th = zerobyte(20); // trace header
55     byte[] the = zerobyte(32); // trace header extension
56     byte[] csh = zerobyte(32); // channel set header
57     ArrayInputStream ais = new ArrayInputStream(segdFile,ByteOrder.BIG_ENDIAN);
58     ais.readBytes(gh); // general header 1
59     int fn = bcd2(gh,0); // file number
60     ais.readBytes(gh); // general header 2
61     ais.readBytes(gh); // general header 3
62     int sln, spn;
63     sln = bin5(gh,3); // source line number
64     spn = bin5(gh,8); // source point number
65     System.out.println("fn=" + fn + ", sln=" + sln + ", spn=" + spn);
66     int cns = 0; // channel set number for seismic trace
67     int nct = 0; // total number of channels, including aux channels
68     int ncs = 0; // number of seismic channels
69     int cn, ct, nc, ic, rln, rpn;
```

```
70      for(int i=0; i<16; ++i){ // for each channel set header, ...
71        ais.readBytes(csh); // read channel set header
72        cn = csh[1]; // channel set number
73        ct = (csh[10]>>4)&0xf; // channel type (high 4 bits)
74        nc = bcd2(csh,8); // number of channels
75        if(nc>0){ // if we have channels of this type, ...
76          System.out.println("cn =" + cn + " nc =" + nc + " ct =" + ct);
77          if(ct==1){ // if seismic, ...
78            cns = cn; // remember channel set number for seismic
79            ncs = nc; // remember number of seismic channels
80          }
81
82        nct += nc; // count total number of channels
83        }
84      }
85      System.out.println("nct =" + nct + " cns =" + cns + " ncs =" + ncs);
86      ais.skipBytes(1024); // skip extended header
87      ais.skipBytes(1024); // skip external header
88      int rpf = 1;
89      int rpl = 1;
90      int n1 = 0; // # samples
91      int n2 = ncs; // #traces
92      float[][] f = null;
93      for(int j=0; j<nct; ++j){ // for all channels (including aux channels)
94        ais.readBytes(th); // trace header
95        cn = th[3]; // channel set number
96        ic = bcd2(th,4); // channel (trace) number
97        ais.readBytes(the); // trace header extension 1
98        rln = bin3(the,0); // receiver line number
99        rpn = bin3(the,3); // receiver point number
100       n1 = bin3(the,7); // number of samples
101       //System.out.println("n1 = "+n1 + " the[7-9]: " + the[7] +" "+ the[8] +"
              "+the[9]);
102       //System.out.println("ic =" + ic + " rln =" + rln + " rpn =" + rpn + " n1
              =" + n1);
103       if(ic==1){
104         rpf = rpn;
105       } else if(ic == n2){
106         rpl = rpn;
107       }
108       ais.skipBytes(6*the.length); // skip trace header extensions 2-7
109       if(cn==cns){ // if seismic channel, ...
110         if(f == null)
111           f = new float[n2][n1];
112         //System.out.println("ic =" + ic + " rln =" + rln + " rpn =" + rpn);
113         ais.readFloats(f[ic-1]); // get the seismic trace
114       } else{
115         ais.skipBytes(4*n1); // skip the aux trace
116       }
117     }
118     ais.close();
119     f = mul(1.0e-14f,f); // scale values to approx. range [-10,10]
120     return new Segdata(sln,spn,rpf,rpl,f);
121   }
122
123   public void plot(Sampling s1, Sampling s2, Segdata seg, String title){
124     SimplePlot sp = new SimplePlot(SimplePlot.Origin.UPPER_LEFT);
125     sp.setSize(900,900);
126     sp.setVLabel("Time (s)");
```

```
127       if(s2.getDelta() ==1.0)
128         sp.setHLabel("Station");
129       else
130         sp.setHLabel("Offset (km)");
131       sp.setHLimits(seg.rpf, seg.rpl);
132       sp.setTitle(title);
133       PixelsView pv = sp.addPixels(s1,s2,seg.f);
134       pv.setPercentiles(1,99);
135     }
136
137     public void tpow2(float[][] f){
138       int n1 = f[0].length;
139       int n2 = f.length;
140       float[][] t = rampfloat(0.0f,0.002f,0.0f,n1,n2);
141       mul(t,t,t);
142       mul(t,f);
143     }
144
145     public void gain2(float[][] f){
146       RecursiveExponentialFilter ref = new RecursiveExponentialFilter(40.0);
147       for(int m = 0; m<f.length; ++m){
148         if(max(abs(f))>0.0f){
149           float[][] g = mul(f,f);
150           ref.apply1(g,g);
151           div(f,sqrt(g),f);
152         }
153       }
154     }
155
156     public void lowpass2(float[][] f){
157       double f3db = 25.0*0.002;
158       ButterworthFilter bf = new ButterworthFilter(f3db,6,ButterworthFilter.Type.
              LOW_PASS);
159       bf.apply1ForwardReverse(f,f);
160     }
161
162     public int bcd2(byte[] b, int k){
163       return (1000*((b[k  ]>>4)&0xf)+100*(b[k  ]&0xf)+
164           10*((b[k+1]>>4)&0xf)+  1*(b[k+1]&0xf));
165     }
166
167     public int bin3(byte[] b, int k){
168       byte b0 = b[k  ];
169       byte b1 = b[k+1];
170       byte b2 = b[k+2];
171       return (b2 & 0xFF) | ((b1 & 0xFF) << 8) | ((b0 & 0x0F) << 16);
172     }
173
174     public int bin5(byte[] b, int k){
175       byte b0 = b[k  ];
176       byte b1 = b[k+1];
177       byte b2 = b[k+2];
178       byte b3 = b[k+3];
179       byte b4 = b[k+4];
180       return (int)(256.0+b0*65536.0+b1*256.0+b2+b3/256.0+b4/65536.0);
181     }
182
183     //public String segdDir = "/gpfc/ckohnke/fc2013/segd/140/"; // Linux Lab
184     public String segdDir = "/home/colton/Documents/School/SrDesign/fc2013/segd
```

```
      /139/"; // Laptop
185
186   public ArrayList<Segdata> _segd;
187
188 }
```

4. Segdata.java

```
 1 import static edu.mines.jtk.util.ArrayMath.*;
 2 import java.util.*;
 3
 4 public class Segdata{
 5   public Segdata(int sl,int sp,int rpf,int rpl,float[][] f){
 6     this.sl = sl;
 7     this.sp = sp;
 8     this.rpf = rpf;
 9     this.rpl = rpl;
10     this.f = ccopy(f);
11   }
12   public int sl,sp,rpf,rpl;
13   public float[][] f;
14 }
15
16 class SegdataComp implements Comparator<Segdata>{
17
18   //@Override
19   public int compare(Segdata p1, Segdata p2) {
20     if(p1.sp > p2.sp){
21        return 1;
22     } else {
23        return -1;
24     }
25   }
26 }
```

5. Waypoints.java

```
 1 ////////////////////////////////////////////////////////////////////////////
 2
 3 import java.io.*;
 4 import java.util.*;
 5
 6 public class Waypoints{
 7   public ArrayList<MPoint> _gps;
 8
 9   public static void main(String[] args){
10     System.out.println("GPS TEST START");
11     //File input = new File("/home/colton/Documents/School/SrDesign/PlotTest/
           gps_input_test1.txt");
12     File input = new File("./gps_input_test1.txt");
13     Waypoints w = new Waypoints(input);
14     //for(int i=0; i< w._gps.size(); ++i){
15       //MPoint p = w._gps.get(i);
16       //System.out.println("Station: " + p.stationID+ " lat: " + p.lat + " lon:
             " + p.lon + " x: " + p.x + " y: " + p.y + " z: " + p.z);
17     //}
18     for(int i=0; i< w._gps.size(); ++i){
19       MPoint p = w._gps.get(i);
```

```
20        System.out.println("Station: " + p.stationID+ " lat: " + p.lat + " lon: "
              + p.lon + " x: " + p.x + " y: " + p.y + " z: " + p.z);
21      }
22
23      System.out.println("GPS TEST FINISH");
24    }
25
26    public Waypoints(File f){
27      _gps = new ArrayList<MPoint>(0);
28      readLatLonFromTSV(f);
29      latLonToUTM();
30      extrapolateGPS();
31    }
32
33    public double degToRad(double deg){
34      return (deg / 180 * PI);
35    }
36
37    public double radToDeg(double rad){
38      return (rad / PI * 180);
39    }
40
41    public void readUTMFromTSV(File f){
42      try{
43        Scanner s = new Scanner(f);
44        s.nextLine(); // header skip = 1
45        while(s.hasNext()){
46          int stationID = s.nextInt();
47          double x = s.nextDouble();
48          double y = s.nextDouble();
49          double z = s.nextDouble();
50          MPoint p = new MPoint(stationID, x, y, z);
51          _gps.add(p);
52        }
53        s.close();
54      } catch(IOException ex){
55        System.out.println(ex);
56      }
57    }
58
59    public void readLatLonFromTSV(File f){
60      try{
61        Scanner s = new Scanner(f);
62        s.nextLine(); //header skip=1;
63        while(s.hasNext()){
64          int stationID = s.nextInt();
65          double lat = s.nextDouble();
66          double lon = s.nextDouble();
67          MPoint p = new MPoint(stationID, lat, lon);
68          _gps.add(p);
69        }
70        s.close();
71      } catch(IOException ex){
72        System.out.println(ex);
73      }
74    }
75
76    public void readLatLonFromXML(File f){
77      try {
```

```
78        Scanner s = new Scanner(f);
79        String current = "";
80        String[] c = null;
81        while(s.hasNext()){
82          while(!current.contains("lat")){
83            current = s.next();
84          }
85          c = current.split("\"");
86          double lat = Double.parseDouble(c[1]);
87          current = s.next();
88          c = current.split("\"");
89          double lon = Double.parseDouble(c[1]);
90          c = current.split("[><]");
91          double elev = Double.parseDouble(c[3]);
92          int name = Integer.parseInt(c[11]);
93          MPoint p = new MPoint(name, lat, lon, elev);
94          _gps.add(p);
95          s.next();
96        }
97        s.close();
98      } catch(IOException ex){
99        System.out.println(ex);
100     }
101
102   }
103
104   public void latLonToUTM(){
105     for(int i=0; i<_gps.size(); ++i){
106       MPoint p = _gps.get(i);
107       double lat = p.lat;
108       double lon = p.lon;
109       int UTMzone = (int)(Math.floor((lon+180.0)/6)+1);
110       double a = 6378.137;
111       double f = 1.0/298.257223563;
112       double n = f/(2.0-f);
113       double e0 = 500.0;
114       double n0 = 0;
115       double k0 = 0.9996;
116       double aa = a/(1.0+n)*(1.0+n*n*(1.0/4.0+n*n/64.0));
117       double a1 = n*(0.5-n*(2.0/3.0-n*5.0/16.0));
118       double a2 = n*n*(13.0/48.0-n*3.0/5.0);
119       double a3 = n*n*n*61.0/240.0;
120       double st = 2.0*Math.sqrt(n)/(1.0+n);
121       double lon0 = -183.0+(UTMzone*6.0); // reference longitude for arbitrary
              UTM Zone
122       lon -= lon0;
123       lat *= PI/180.0;
124       lon *= PI/180.0;
125       double t = Math.sinh(atanh(Math.sin(lat))-st*atanh(st*Math.sin(lat)));
126       double ep = Math.atan(t/Math.cos(lon));
127       double np = atanh(Math.sin(lon)/Math.sqrt(1.0+t*t));
128       double sx  = a1*Math.cos(2.0*ep)*Math.sinh(2.0*np);
129       sx += a2*Math.cos(4.0*ep)*Math.sinh(4.0*np);
130       sx += a3*Math.cos(6.0*ep)*Math.sinh(6.0*np);
131       double sy  = a1*Math.sin(2.0*ep)*Math.cosh(2.0*np);
132       sy += a2*Math.sin(4.0*ep)*Math.cosh(4.0*np);
133       sy += a3*Math.sin(6.0*ep)*Math.cosh(6.0*np);
134       double x = e0+k0*aa*(np+sx);
135       double y = n0+k0*aa*(ep+sy);
```

```
136         x *= 1000.0;
137         y *= 1000.0;
138         p.x = x; p.y = y; p.UTMzone = UTMzone;
139       }
140     }
141
142     public void UTMToLatLong(){
143
144     }
145
146     public void readUTMFromCSV(File f){
147       try{
148         Scanner s = new Scanner(f);
149         s.useDelimiter(",");
150         s.nextLine(); // header skip = 1
151         while(s.hasNext()){
152           int stationID = s.nextInt();
153           double x = s.nextDouble();
154           double y = s.nextDouble();
155           double z = s.nextDouble();
156           MPoint p = new MPoint(stationID, x, y, z);
157           _gps.add(p);
158         }
159       } catch(IOException ex){
160         System.out.println(ex);
161       }
162
163     }
164
165     public void exportToCSV(File f){
166       try{
167         if (f!=null) {
168           String filename = f.getAbsolutePath();
169           BufferedWriter w = new BufferedWriter(new FileWriter(f));
170           w.write("Station,Easting,Northing,Elevation");
171           w.newLine();
172           for(int i=0; i<_gps.size(); ++i){
173             MPoint p = _gps.get(i);
174             w.write(p.stationID + "," + p.x + "," + p.y + "," + p.z);
175             w.newLine();
176           }
177           w.close();
178         }
179       } catch(IOException ex){
180         System.out.println(ex);
181       }
182     }
183
184     public void extrapolateGPS(){ //assumes
185       Collections.sort(_gps, new MPointComp());
186       int start, end, dn;
187       double dx, dy, dz, r;
188       double x, y, z;
189       ArrayList<MPoint> gnew = new ArrayList<MPoint>(0);
190       for(int i=0; i<_gps.size()-1; ++i){
191         MPoint p1 = _gps.get(i);
192         MPoint p2 = _gps.get(i+1);
193         start = p1.stationID;
194         end = p2.stationID;
```

```
195         dx = p1.xDist(p2);
196         dy = p1.yDist(p2);
197         dz = p1.zDist(p2);
198         r =  p1.xyzDist(p2);
199         dn = end-start-1;
200         for(int m=1; m<=dn; ++m){
201           x = p1.x + dx*m/dn;
202           y = p1.y + dy*m/dn;
203           z = p1.z + dz*m/dn;
204           MPoint a = new MPoint(start+m, x, y, z, true);
205           gnew.add(a);
206         }
207       }
208       for(int i=0; i<gnew.size(); ++i)
209         _gps.add(gnew.get(i));
210       Collections.sort(_gps, new MPointComp());
211     }
212
213     protected double atanh(double x){
214       return 0.5*Math.log((1.0+x)/(1.0-x));
215     }
216
217     private final double PI = Math.PI;
218
219 }
```

13.2 Appendix B: Reference Jython Code

1. segd.py

```
1  """
2  Reads segd files from CSM field camp, assuming these are in Sercel's
3  SEG-D Rev 1 format, and writes a dat file containing a 3D array of
4  floats. The byte order for floats in the dat file is BIG_ENDIAN.
5
6  Author: Dave Hale, Colorado School of Mines
7  Version: 2012.05.09
8  """
9  from imports import *
10
11 """
12 Line10, 2011
13 station spacing is 30 m
14 line was shot east to west
15 Vibrator A is 3 stations to the east of channel 1
16 Vibrator B is 81 stations to the west of channel 120
17 """
18 """
19 Line10, 2012
20 station spacing is 15 m
21 line was shot SW to NE
22 shots from 1003 - 1217
23 Vibrator A is 3 stations to the east of channel 1
24 Vibrator B is 81 stations to the west of channel 120
25 """
26 s1 = Sampling(4001,0.002,0.000) # time sampling
27 s2 = Sampling(342,1,954) # station sampling, sweep 1
28 s3 = Sampling(215,1.0,1003) # first shotpoint is 1003
```

```
#s2 = Sampling(277,0.015,-0.030) # channel sampling
#s3 = Sampling(1,1.0,1001.0) # shotpoint station sampling A
#shotDir = "/data/seis/csm/fc2012/"
#segdDir = "/data/seis/csm/fc2012/segd/test139/"

#shotDir = "/data/seis/csm/fc2013/" #Linux Lab
#segdDir = "/gpfc/ckohnke/fc2013/segd/141/" #Linux Lab

shotDir = "/home/colton/Documents/School/SrDesign/fc2013/" #Laptop
segdDir = "/home/colton/Documents/School/SrDesign/fc2013/segd/141/" #Laptop
##############################################################################
def main(args):
  readLine141Segd()
  #displayLine141()
  #displayLine140S1()
  #readLine140Segd()
  #readTestSegd()

def readLine141Segd():
  #global s3
  csegdList = File(segdDir).listFiles() # list of segd files
  nshot = len(segdList)-3 # ignore first 3 shots
  s3 = Sampling(nshot,1,1003) # first shotpoint is 1003
  g = zerofloat(s1.count,s2.count,s3.count)
  print "s1.count ", s1.count
  print "s2.count ", s2.count
  print "s3.count ", s3.count
  #print segdList
  for segdFile in segdList[3:]:
    print segdFile
    sl,sp,rpf,rpl,f = readSegd(segdFile)
    print "sl =",sl," sp =",sp," rpf =",rpf," rpl =",rpl
    i3 = int(sp-s3.first)
    zero(f[42]) # no geophone string at station 996
    copy(f,g[i3])
    #lowpass2(f)
    #tpow2(f)
    #gain2(f)
    plot(s1,s2,f,title="Shot "+str(sp))
    #plotAmp(s1,s2,f,title="Shot "+str(sp))
  writeData(g,shotDir+"shotsp.dat")

def displayLine141():
  f = readData(s1,s2,s3,shotDir+"shotsp.dat")
  lowpass3(f)
  tpow3(f)
  gain3(f)
  sf = SimpleFrame()
  ip = sf.addImagePanels(f)
  #ip.setClips(-2.5,2.5)

def displayLine140S1():
  f = readData(s1,s2,s3,shotDir+"shotsp.dat")
  lowpass3(f)
  tpow3(f)
  gain3(f)
  sf = SimpleFrame()
  ip = sf.addImagePanels(f)
  #ip.setClips(-2.5,2.5)
```

```
88
89 def readLine140Segd():
90   segdList = File(segdDir).listFiles() # list of segd files
91   nshot = len(segdList)
92   g = zerofloat(s1.count,s2.count,nshot)
93   #print segdList
94   ishot = 0
95   for segdFile in segdList[:]:
96     print segdFile
97     sl,sp,rpf,rpl,f = readSegd(segdFile)
98     #s2 = Sampling(len(f),0.015,-0.030) # offset sampling
99     #s2 = Sampling(len(f),1,954) # station sampling
100    #tpow2(f)
101    #lowpass2(f)
102    #gain2(f)
103    print "sl =",sl," sp =",sp," rpf =",rpf," rpl =",rpl
104    copy(f,g[ishot])
105    ishot += 1
106    #plot(s1,s2,f,title="Shot "+str(ishot))
107    #plotAmp(s1,s2,f,title="Test "+str(ishot))
108  #sf = SimpleFrame()
109  #ip = sf.addImagePanels(g)
110  #ip.setPercentiles(2,98)
111  writeData(g,shotDir+"shots.dat")
112  writeData(g,shotDir+"shotsp.dat",bo=ByteOrder.LITTLE_ENDIAN)
113
114 def readTestSegd():
115   segdList = File(segdDir).listFiles() # list of segd files
116   #print segdList
117   itest = 0
118   for segdFile in segdList:
119     print segdFile
120     sl,sp,rpf,rpl,f = readSegd(segdFile)
121     s1 = Sampling(len(f[0]),0.002,0.000) # time sampling
122     #s2 = Sampling(len(f),0.015,-0.030) # offset sampling
123     s2 = Sampling(len(f),1,1001) # station sampling
124     tpow2(f)
125     lowpass2(f)
126     gain2(f)
127     print "sl =",sl," sp =",sp," rpf =",rpf," rpl =",rpl
128     itest += 1
129     plot(s1,s2,f,title="Test "+str(itest))
130     #plotAmp(s1,s2,f,title="Test "+str(itest))
131
132 def readData(s1,s2,s3,fileName,bo=ByteOrder.LITTLE_ENDIAN):
133   n1,n2,n3 = s1.count,s2.count,s3.count
134   f = zerofloat(n1,n2,n3)
135   ais = ArrayInputStream(fileName,bo)
136   ais.readFloats(f)
137   ais.close()
138   return f
139
140 def writeData(flist,fileName,bo=ByteOrder.LITTLE_ENDIAN):
141   n3 = len(flist)
142   print "writing",n3," shot records to",fileName
143   aos = ArrayOutputStream(fileName,bo)
144   for f in flist:
145     aos.writeFloats(f)
146   aos.close()
```

```
147
148 def tpow2(f):
149   n1,n2 = len(f[0]),len(f)
150   t = rampfloat(0,0.002,0.0,n1,n2) # time
151   mul(t,t,t) # time squared
152   return mul(t,f)
153
154 def tpow3(f):
155   n1,n2,n3 = len(f[0][0]),len(f[0]),len(f)
156   t = rampfloat(s1.first,s1.delta,0.0,n1,n2) # time
157   mul(t,t,t) # time squared
158   for f3 in f:
159     mul(t,f3,f3)
160
161 def gain2(f):
162   ref = RecursiveExponentialFilter(40.0)
163   for f2 in f:
164     if max(abs(f2))>0.0:
165       g = mul(f2,f2)
166       ref.apply1(g,g)
167       div(f2,sqrt(g),f2)
168
169 def gain3(f):
170   ref = RecursiveExponentialFilter(40.0)
171   for f3 in f:
172     if max(abs(f3))>0.0:
173       g = mul(f3,f3)
174       ref.apply1(g,g)
175       div(f3,sqrt(g),f3)
176
177 def lowpass2(f):
178   f3db = 25.0*0.002
179   #f3db = 35.0*0.002
180   bf = ButterworthFilter(f3db,6,ButterworthFilter.Type.LOW_PASS)
181   bf.apply1ForwardReverse(f,f)
182
183 def lowpass3(f):
184   bf = ButterworthFilter(0.05,6,ButterworthFilter.Type.LOW_PASS)
185   bf.apply1ForwardReverse(f,f)
186
187 def plot(s1,s2,f,title=None):
188   print "plot f: min =",min(f),"max =",max(f)
189   sp = SimplePlot(SimplePlot.Origin.UPPER_LEFT)
190   #sp.setSize(750,1000)
191   sp.setSize(900,900)
192   sp.setVLabel("Time (s)")
193   if s2.delta==1.0:
194     sp.setHLabel("Station")
195   else:
196     sp.setHLabel("Offset (km)")
197   sp.setVLimits(0.0,8.0)
198   if title:
199     sp.setTitle(title)
200   pv = sp.addPixels(s1,s2,f)
201   #pv.setColorModel(ColorMap.BLUE_WHITE_RED)
202   pv.setPercentiles(1,99)
203   #pv.setClips(-2.5,2.5)
204
205 def plotAmp(s1,s2,f,title=None):
```

```
206   fft = Fft(s1)
207   sf = fft.getFrequencySampling1()
208   ff = zerofloat(sf.count,s2.count)
209   for i2 in range(s2.count):
210     ff[i2] = cabs(fft.applyForward(f[i2]))
211   sp = SimplePlot(SimplePlot.Origin.UPPER_LEFT)
212   #sp.setSize(750,1000)
213   sp.setSize(900,900)
214   sp.setVLabel("Frequency (Hz)")
215   if s2.delta==1.0:
216     sp.setHLabel("Station")
217   else:
218     sp.setHLabel("Offset (km)")
219   sp.setVLimits(0.0,120.0)
220   if title:
221     sp.setTitle(title)
222   pv = sp.addPixels(sf,s2,ff)
223   pv.setColorModel(ColorMap.JET)
224   pv.setPercentiles(1,99)
225   #pv.setClips(-2.5,2.5)
226
227 def readSegd(segdFile):
228   #n1,n2 = 4001,230 # number of samples, number of traces
229   #n1,n2 = 4001,277 # number of samples, number of traces (1 sweep)
230   n1,n2 = 4001,342 # number of samples, number of traces (5 sweeps)
231   gh = zerobyte(32) # general header
232   th = zerobyte(20) # trace header
233   the = zerobyte(32) # trace header extension
234   csh = zerobyte(32) # channel set header
235   ais = ArrayInputStream(segdFile,ByteOrder.BIG_ENDIAN)
236   ais.readBytes(gh) # general header 1
237   fn = bcd2(gh,0) # file number
238   ais.readBytes(gh) # general header 2
239   ais.readBytes(gh) # general header 3
240   sln = bin5(gh,3) # source line number
241   print "gh[3-7] = ",gh[3]," ",gh[4]," ",gh[5]," ",gh[6]," ",gh[7]
242   spn = bin5(gh,8) # source point number
243   print "file =",segdFile
244   print "fn = ",fn," sln =",sln," spn =",spn
245   cns = 0 # channel set number for seismic traces
246   nct = 0 # total number of channels, including aux channels
247   for ics in range(16): # for each channel set header, ...
248     ais.readBytes(csh) # read channel set header
249     cn = csh[1] # channel set number
250     ct = (csh[10]>>4)&0xf # channel type (in high 4 bits)
251     nc = bcd2(csh,8) # number of channels
252     if nc>0: # if we have channels of this type, ...
253       print "cn =",cn," nc =",nc," ct =",ct
254       if ct==1: # if seismic, ...
255         cns = cn # remember channel set number for seismic
256         ncs = nc # remember number of seismic channels
257       nct += nc # count total number of channels
258   print "nct =",nct,"cns =",cns
259   ais.skipBytes(1024) # skip extended header
260   ais.skipBytes(1024) # skip external header
261   rpf = 1
262   rpl = 1
263   f = None
264   for ict in range(nct): # for all channels (including aux channels)
```

```
265      ais.readBytes(th) # trace header
266      cn = th[3] # channel set number
267      ic = bcd2(th,4) # channel (trace) number
268      ais.readBytes(the) # trace header extension 1
269      rln = bin3(the,0) # receiver line number
270      rpn = bin3(the,3) # receiver point number
271      n1 = bin3(the,7) # number of samples
272      #print "n1 = ",n1," the[7-9]: ",the[7]," ",the[8]," ",the[9]
273      print "ic =",ic," rln =",rln," rpn =",rpn," n1 =",n1
274      if ic==1:
275        rpf = rpn
276      elif ic==n2:
277        rpl = rpn
278      ais.skipBytes(6*len(the)) # skip trace header extensions 2-7
279      if cn==cns: # if seismic channel, ...
280        #print "ic =",ic," rln =",rln," rpn =",rpn
281        if not f:
282          f = zerofloat(n1,n2) # the traces
283        ais.readFloats(f[ic-1]) # get the seismic trace
284      else:
285        ais.skipBytes(4*n1) # skip the auxiliary trace
286    ais.close()
287    f = mul(1.0e-14,f) # scale values to approximate range [-10,10]
288    return sln,spn,rpf,rpl,f
289
290  def readLine10Segd():
291    segdList = File(segdDir).listFiles() # list of segd files
292    n1,n2 = s1.count,s2.count
293    fzeros = zerofloat(n1,n2)
294    spfa = int(s3a.first) # first shot point for vib A
295    spfb = int(s3b.first) # first shot point for vib B
296    faList,fbList = [],[] # lists of shot records
297    spa,spb = spfa-1,spfb-1 # shots last appended
298    for segdFile in segdList:
299      sl,sp,rpf,rpl,f = readSegd(segdFile)
300      #print segdFile
301      print "sl =",sl," sp =",sp," rpf =",rpf," rpl =",rpl
302      if sl==10: # vibrator A
303        if sp==spa: # if same station as before, count the last one
304          faList.pop()
305        for i in range(spa+1,sp): # if necessary, insert zero records
306          faList.append(fzeros)
307          print "a: zero ",i
308        print "a: append ",sp
309        spa = sp
310        faList.append(f)
311      elif sl==20: # vibrator B
312        if sp==spb:
313          fbList.pop()
314        for i in range(spb+1,sp):
315          fbList.append(fzeros)
316          print "b: zero ",i
317        print "b: append ",sp
318        spb = sp
319        fbList.append(f)
320    na = len(faList)
321    nb = len(fbList)
322    print "na =",na," nb =",nb
323    writeData(faList,shotDir+"shota.dat")
```

```
  writeData(fbList,shotDir+"shotb.dat")

def readSegd2011(segdFile):
  n1,n2 = 3001,120 # number of samples, number of traces
  f = zerofloat(n1,n2) # the traces
  gh = zerobyte(32) # general header
  th = zerobyte(20) # trace header
  the = zerobyte(32) # trace header extension
  csh = zerobyte(32) # channel set header
  ais = ArrayInputStream(segdFile,ByteOrder.BIG_ENDIAN)
  ais.readBytes(gh) # general header 1
  fn = bcd2(gh,0) # file number
  ais.readBytes(gh) # general header 2
  ais.readBytes(gh) # general header 3
  sln = bin5(gh,3) # source line number
  spn = bin5(gh,8) # source point number
  #print "file =",segdFile
  #print "fn = ",fn," sln =",sln," spn =",spn
  cns = 0 # channel set number for seismic traces
  nct = 0 # total number of channels, including aux channels
  for ics in range(16): # for each channel set header, ...
    ais.readBytes(csh) # read channel set header
    cn = csh[1] # channel set number
    ct = (csh[10]>>4)&0xf # channel type (in high 4 bits)
    nc = bcd2(csh,8) # number of channels
    if nc>0: # if we have channels of this type, ...
      #print "cn =",cn," nc =",nc," ct =",ct
      if ct==1: # if seismic, ...
        cns = cn # remember channel set number for seismic
        ncs = nc # remember number of seismic channels
      nct += nc # count total number of channels
  #print "nct =",nct,"cns =",cns
  ais.skipBytes(1024) # skip extended header
  ais.skipBytes(1024) # skip external header
  for ict in range(nct): # for all channels (including aux channels)
    ais.readBytes(th) # trace header
    cn = th[3] # channel set number
    ic = bcd2(th,4) # channel (trace) number
    ais.readBytes(the) # trace header extension 1
    rln = bin3(the,0) # receiver line number
    rpn = bin3(the,3) # receiver point number
    if ic==1:
      rpf = rpn
    elif ic==120:
      rpl= rpn
    ais.skipBytes(6*len(the)) # skip trace header extensions 2-7
    if cn==cns: # if seismic channel, ...
      #print "ic =",ic," rln =",rln," rpn =",rpn
      ais.readFloats(f[ic-1]) # get the seismic trace
    else:
      ais.skipBytes(4*n1) # skip the auxiliary trace
  ais.close()
  f = mul(1.0e-14,f) # scale values to approximate range [-10,10]
  return sln,spn,rpf,rpl,f

def bcd2(b,k):
  """ Returns binary-coded decimal integer from bytes k,k+1 in b."""
  return (1000*((b[k  ]>>4)&0xf)+100*(b[k  ]&0xf)+
             10*((b[k+1]>>4)&0xf)+  1*(b[k+1]&0xf))
```

```
383
384 def displayLine10(vib):
385   if vib=="A":
386     f = readData(s1,s2,s3a,shotDir+"shota.dat")
387   elif vib=="B":
388     f = readData(s1,s2,s3b,shotDir+"shotb.dat")
389   lowpass3(f)
390   tpow3(f)
391   gain3(f)
392   sf = SimpleFrame()
393   ip = sf.addImagePanels(f)
394   ip.setPercentiles(2,98)
395
396 def bin3(b,k):
397   """ Returns binary integer from bytes k,k+1,k+2 in b."""
398   b0 = b[k  ]
399   b1 = b[k+1]
400   b2 = b[k+2]
401   if b0<0: b0 += 256
402   if b1<0: b1 += 256
403   if b2<0: b2 += 256
404   return (b0<<16)|(b1<<8)|(b2)
405
406 def bin5(b,k):
407   """ Returns binary integer from bytes k,k+1,...,k+4 in b."""
408   b0 = b[k  ]
409   b1 = b[k+1]
410   b2 = b[k+2]
411   b3 = b[k+3]
412   b4 = b[k+4]
413   if b0<0: b0 += 256
414   if b1<0: b1 += 256
415   if b2<0: b2 += 256
416   if b3<0: b3 += 256
417   if b4<0: b4 += 256
418   return b0*65536.0+b1*256.0+b2+b3/256.0+b4/65536.0
419
420 ##########################################################################
421 class RunMain(Runnable):
422   def run(self):
423     main(sys.argv)
424 SwingUtilities.invokeLater(RunMain())
```