**Reflection Processing Exercise**

*Part 1: Seismic Unix*

In this exercise we are going to do a brute stack of some high resolution seismic reflection data collected in April 2010 with a 24 channel PASSCAL system. The data were collected to study glacial sediments near Ft Wayne, Indiana, by my introductory geophysics class. This is pretty much a turnkey exercise using seismic unix.

1. Copy the directory I will point you at containing a raw data file and a set of shell scripts. Seismic Unix is a crude seismic reflection system using the unix concept of data processing pipelines implemented through what is commonly called a “pipe” or a “FIFO (First In First Out)”.
2. First we need to set the geometry. For modern systems this almost always involves a database to translate field notes and/or navigational data into attributes written to the trace headers. Run the script “setgeom.sh” and it should create a file called datawgeom.su from rawdata.su. Look at the script and you see the script runs two programs (sushw and suchw) that do arithmetic on trace headers and set new values. You can read the documentation at a later date if you want to understand these obscure parameters but the key point is this script does two things: (a) it sets trace header values sx, gx, offset, and cdp, and (b) it does this through a pipeline which passes data between two programs reading from rawdata.su and writing the result to datawgeom.su. This is a classic unix filter processing pipeline.
3. Look at a few of the shot gathers using the “showshot.sh” script. Usage is:

showshot.sh file n

where file is a variable (here use datawgeom.su) and n is a “field file id number” (fldr to seismic unix), which is a trace attribute written with the raw data by the digtizers (This is the lowest common denominator in reflection data indexing). For these data fldr ranges from 112 to 269. Note you can zoom into a seismic unix plot like this with mb1-drag, release. To exit the plot hit “q” in the seismic display window or kill the window (red x in upper right hand of window on these macs).

1. Now let’s generate a brute stack. Run the crude script here called “process.csh”. It will run silently for a few seconds and should just return control to your terminal. When it is finished look at the file and you should see that this script is another unix pipeline. This one runs four programs: sumute, susort, sunmo, and sustack. A peculiar anomaly of this example is that susort does a disk-based sort of its output. As a result susort produce a copy of the data with the data order changed from “shot gathers” to “cdp gathers”.
2. Plot the result with the little script called showstack.sh. This script plots the brute stack in an image format followed by a more conventional wiggle trace, variable area plot. (Note you have to kill the image plot before the wtva plot will appear).

Key concepts I hope this exercise illustrates are:

1. Data pipeline approach to processing
   1. Data flows in a linear way through the processing flow
   2. Processing “modules” (in this case discrete programs, but in other systems modules can have a different meaning) act on the data and alter it to produce a new version of the data.
   3. Flow is data driven
   4. Key “object” in the processing stream are seismic traces
2. A data set tends to be defined by a small number (here one) large entity (segy or related data files). This is in contrast to sac with one trace per file.
3. The processing is driven by “metadata” which in the seismic reflection world are called trace headers.

*Part 2: SIA*

Another example of a package that is “reflection like”, but which is much more is the SIA/IGEOS package available from Igor Morozov at the University of Saskatchewan. The web site for this package is: <http://seisweb.usask.ca/SIA/> , but the most useful page linked from that page is this one: <http://seisweb.usask.ca/SIA/index/index.html> . That later is the online documentation for each module.

SIA/IGeos is reflection like because it uses a similar model to a package like ProMAX. That is, data is passed through a “job” and acted on my “modules”. It is only loosely like Seismic Unix in this sense. In this exercise we will use this as an example of a command line processing flow description. Sia has a “gui” front end like most modern commercial systems, but we’ll focus on the command language for two reasons: (1) the gui is a pain to install and we’ll be using the teragrid machine, and (2) the language does a better job of illustrating how this kind of thing works. With that, let’s jump in:

1. Login to the teragrid machine called quarry.uits.indiana.edu using ssh and the guest account we assigned to you.
2. You need this setup: source /N/dc/scratch/pavlis/enable\_sia
3. In your terminal window cd to NWU\_sc2010
4. For this exercise we are going to look at two “job files: refl\_synthetic.job and refl\_nmo.job. Let’s look at the “job” files together and I’ll try my best to describe the language and what they do.
5. Run the first job that will generate a synthetic shot gather using the reflectivity method: sia refl\_synthetic.job (This takes several minutes to run).
6. When the synthetic job finishes run the second: sia refl\_nmo.job.
7. Look at the outputs in the PS directory. This is an illustration of the current way you commonly interact with a “big iron” machine in that graphics are better done locally. The outputs here are all postcript files so the best approach is:
   1. Launch a new local terminal window
   2. Use sftp to connect and transfer these files:

sftp quarry.uits.indiana.edu

cd to PS directory

get –enter ps file name to fetch-

* 1. View the files with the mac preview program

1. Peruse some of the other “job” files in this directory. They illustrate some additional examples that were the earlier versions of those used in this pair of recent papers:

Morozov, I. and G. Pavlis (2011a). Management of large seismic data sets I: Automated Building and Updating using BREQ\_FAST and NetDC, Seismological Research Letters, 82(2), p. 211-221.

Morozov, I. and G. Pavlis (2011b). Management of large seismic data sets II: Data Center-type Operations, Seismological Research Letters, 82(2), p. 222-226.