Due Date: 17:00, Wednesday, April 10th, 2013
TA: Yunyue (Elita) Li

Lab 1: Introduction to 3D geometries

Your Name: Catherine of Aragon

ABSTRACT

In this exercise, you will explore two 3D land acquisition geometries, and begin to learn how to manipulate prestack seismic data with the SEPlib3d ("SEP3d") processing system. The first geometry comes from a shallow seismic experiment, and the second from an exploration survey. There are no programming tasks required, although the lab does require you to run SEP3d programs yourself, either from the command-line or from a Makefile.

INTRODUCTION

This assignment requires you to explore the differences between acquisition geometries, and also serves as a tutorial to the SEPlib3D processing library (a.k.a. SEP3d). Among other things, you are required to:

1. Sort and bin the subset by common midpoint (CMP).
2. Compute and analyze the fold chart for the subset.
3. Explore the geometry of a shallow seismic experiment.

SEPlib3D processing library

SEPlib3D is a set of extensions to the established SEPlib environment, which provides a powerful and flexible environment for the manipulation and processing of irregularly-sampled 3D prestack seismic data. Although there are other processing systems available, commercial systems such as Promax and Focus are generally designed for batch processing, while SU was designed primarily for 2-D research. SEPlib3D was designed expressly for research and development with prestack 3D seismic data.

Before you attempt the assignment make sure you understand the structure of a SEPlib3D dataset explained below. For more detailed description check the web site about SEPlib3D: http://sep.stanford.edu/doku.php?id=sep:software:seplib and Appendix 1 in your course notes. If you still don't feel comfortable, see the TA for help.

Structure of a SEPlib3D data set

A very important and efficient feature of SEP3d is the separation of the geometry information from the seismic data. As you will notice through this exercise, this principle minimizes the need to access the bulky file of seismic data, and allows many operations to be performed on the trace headers only, or small subsets of the seismic data.

As shown in Figure 1, a "complete" SEPlib3D data set consists of 6 files: three ASCII files and three binary files. With the exception of the Header File, the existence of all the other files is optional. The six files are connected to one other through file pointers contained in the ASCII files.

The core unit of the SEP3d data structure is the **data record**; typically this is a seismic trace. Each trace contains headers, which describe such things as the receiver position or time of day. A raw dataset consists of a collection of traces and their corresponding headers.

In Figure 1, the header file, *file.H*, contains the number of data records, and a pointer to the location of the actual data value file (DVF), *file.H@*. Additionally, the header file contains a file pointer to the header format file (HFF), *file.H@@*. The HFF tells us how many headers each trace has and the names of these headers. It also contains a pointer to another binary file, the header values file (HVF), which contains the actual values of the headers of each data record.

Most later geophysical processing operations require the data to be on a regular grid, but the raw dataset described in the previous paragraph consists only of data records and their corresponding headers. SEP3d allows us to define multiple grid axes without resorting the entire dataset. A grid is uniquely defined by the number of grid points (*n*), the grid spacing (*d*), and the origin of the grid (*o*). Figure 1 shows the grid format file (GFF), *file.H@@@@*, and grid values file (GVF), *file.H@@@@@*.

Figure 2 illustrates the so-called "double-pointer" structure of SEP3d. Imagine that we wish to sort a 2-D dataset into common midpoint (CMP) gathers. We must define two grid axes, offset and midpoint, of dimension *n2* and *n3*, respectively. The GVF contains *n2* × *n3* elements. When the data are irregular, some bins in the GVF point to no valid data record–we have an empty bin, and a null pointer (-1) in the GVF. Other times, more than one data record falls into a single bin.

In the header file, the *gff* and *hff* parameters refer to the GFF and HFF, respectively. Both the GFF and HFF are completely valid SEPlib history files, and may be processed accordingly. We will exploit this fact later in the assignment.

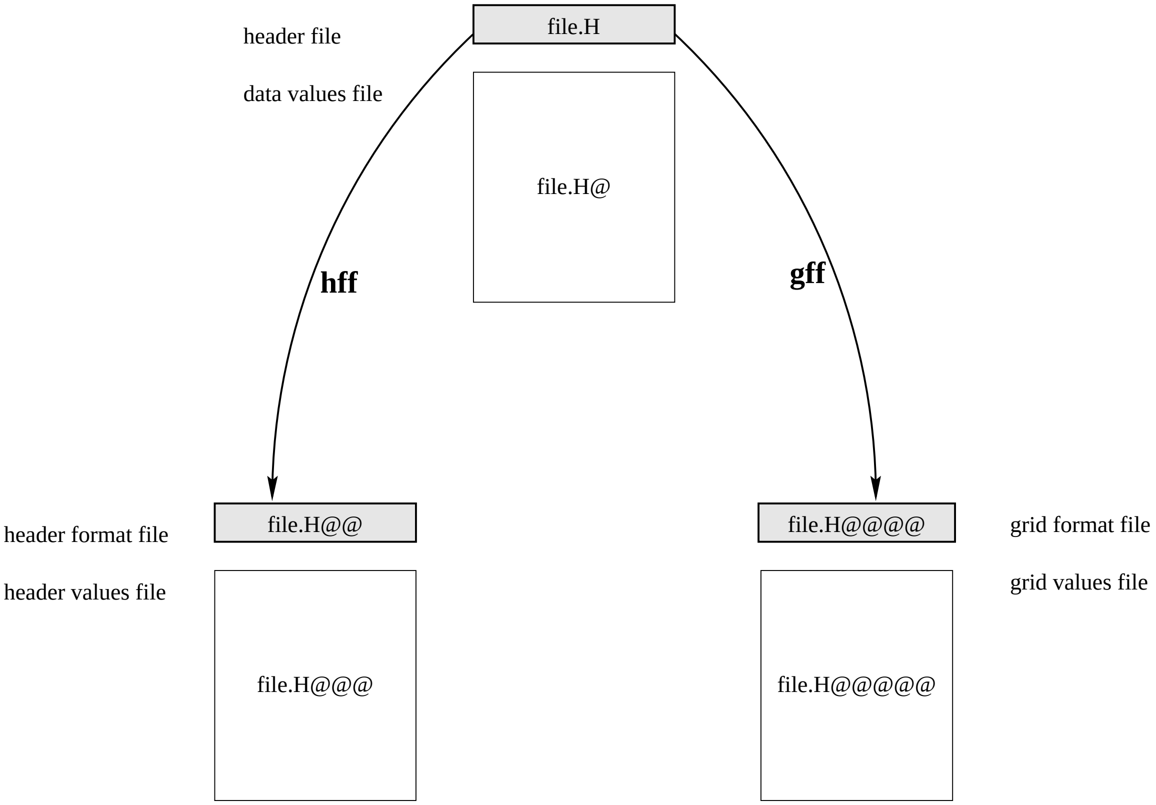

Figure 1: SEPlib3d data-file structure.

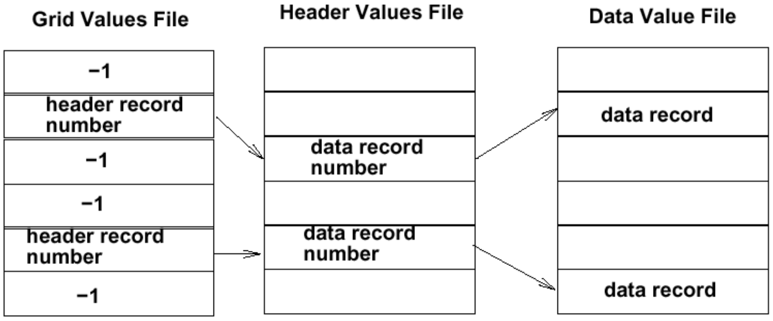

Figure 2: Double-pointer mechanism.

ASSIGNMENT

First log into an SEP workstation. From:

http://sepwww.stanford.edu/doku.php?id=sep:courses:gp280_labs,

download the gzipped tar file, *lab1.tgz*. Type tar -xvzf lab1.tgz; cd GP280_lab1.

Open the *paper.tex* file with your favorite editor, and begin the exercise by changing the name on this document to yours. In order to see the results of your editing you have to save the document and type *scons paper.pdf* again. Hopefully the document will rebuild with your changes.

At any point in this exercise, if you really mess things up and the document does not rebuild, restore the original document by typing *cp -f paper.tex.orig paper.tex*.

Most of the commands for this lab are contained in the *Makefiles*. Note that the main makefile includes other two makefiles:

- *SHLakefile* which contains the commands relevant to the shallow dataset.
- *BIGakefile* which contains the commands relevant to the bigger, exploration dataset.

Take your time to understand what is going on in the makefiles because the answers to many of the questions originate in this file. If you have questions, ask the TA.

You can see the self-documentation for any application simply by typing its name on the command line. The self-doc should give you a brief description of the program and the parameters that it takes. For example, type *In3d* at the command line and you will get a brief description of that utility.

Shallow seismic dataset

In the next few questions, you will work with an ultra-shallow 3-D seismic survey performed at Moss Beach, California, by Rickeemph (SEP-97). Build the header file for the shallow seismic dataset using the command *make SData.H*.

1. Use the command *In3d* to analyze the dataset. Answer the following questions:
   (a) How many records are in the data? **Your Answer:**
   (b) Where are the data values themselves located (be specific)? **Your Answer:**
   (c) How many header keys are associated with each data record? **Your Answer:**

(d) What key numbers describe the source and receiver locations? **Your Answer:**

(e) Have the data been associated to a grid? How can you tell? **Your Answer:**

2. **Shot and receiver geometry:** Figure 3 shows the receiver locations for this dataset, while Figure 4 shows the source locations. Burn these figures by typing *make rec.burn* or *make sou.burn*, and then rebuild them with *make rec.view* or *make sou.view*. Look at the commands that are being run to do this by opening the file *SHLakefile* in your favorite editor. To make these two figures, conventional **SEPlib** utilities are used to process the HFF and HVF.

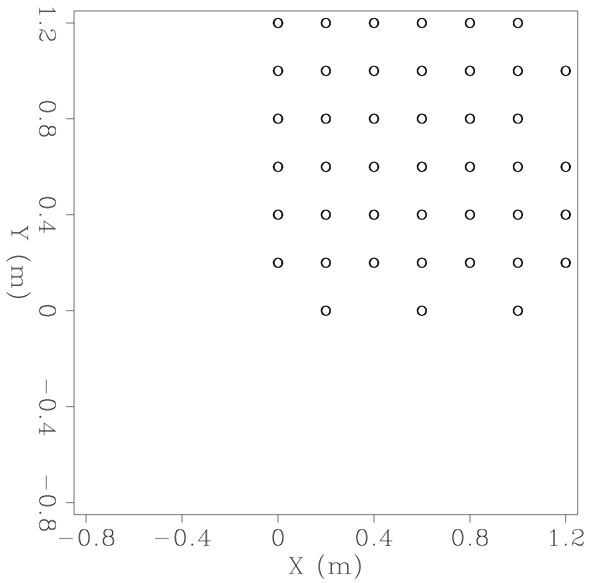

Figure 3: Receiver locations

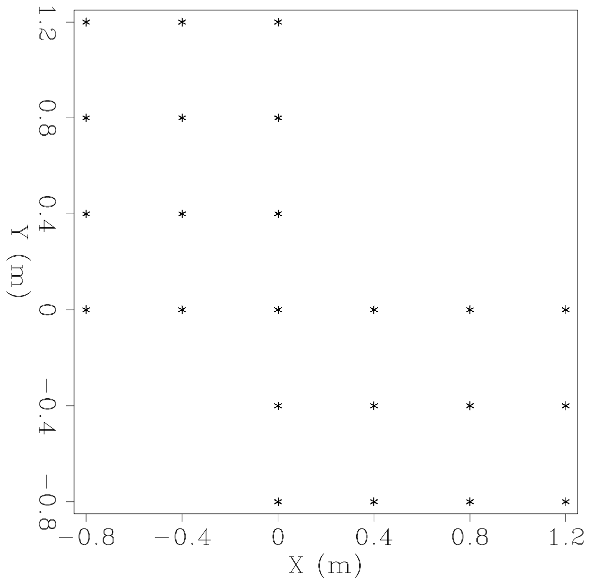

Figure 4: Source locations

(a) Explain briefly how the HFF is manipulated to obtain Figures 3 and 4. **Your Answer:**

(b) Are all the source and receiver locations shown? How can you tell? **Your Answer:**

(c) Do you expect uniform illumination of those reflectors that are flat and horizontal? Why? How about the nonflat, non-horizontal ones?**Your Answer:**

3. **CMP geometry:** Figure 5 shows the locations of the CMPs.

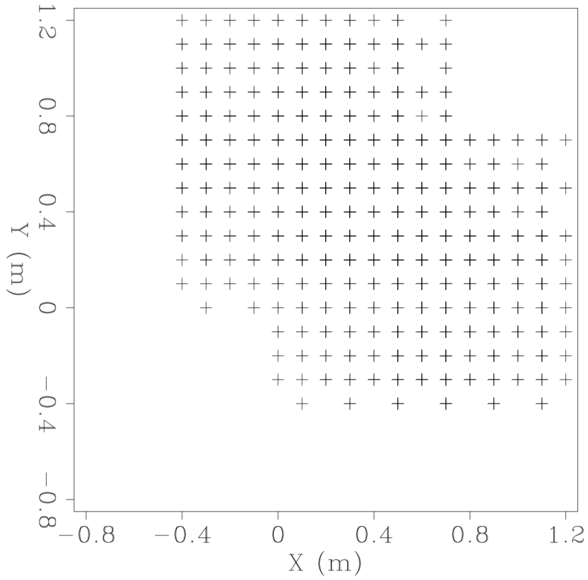

Figure 5: CMP locations

(a) Would you change your earlier answer regarding the subsurface illumination? Why? **Your Answer:**

(b) To make Figure 5, the HFF was manipulated in the same fashion as with Figures 3 and 4, because CMP keys had been created beforehand. To create these keys, use the SEP3d utility *Headermath*. Refer to the self-doc and write a command that you would use to generate a header file with CMP keys, *output.H*, from an input file, *input.H*, which contains only s_x, s_y, g_x, and g_y keys. **Your Answer:**

4. **Sorting and binning:** The subset that you have extracted is in **SEP3d** ungridded format. The seismic traces and the trace headers are in arbitrary order. Before they can be processed, you must sort and bin the trace headers on a regular grid using the *Sort3d* program.

Use the command *make grid.H* and observe how a grid is associated to the dataset. You can redefine the grid by changing parameters in the file *par/sortcmp.P*.

Use the command *make fold.H* to compute the fold for this dataset. Figure 6 shows the fold distribution.

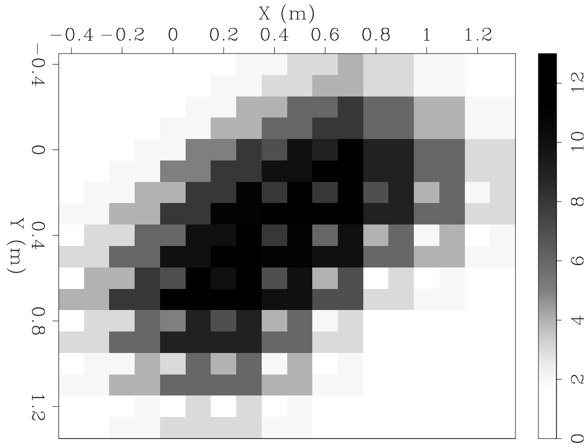

Figure 6: Fold distribution.

(a) Comment on the fold distribution for this subset. **Your Answer:**

(b) Would there be any advantages in using a different bin size than the nominal bin-size determined by the survey parameters? **Your Answer:**

(c) According to your previous answer, experiment by changing the parameters in the file *par/sortcmp.P* and rebuilding Figure 6 using the commands *make fold.burn* followed by *make fold.view*. When you are happy, do *make fold.build* so I can also see your best image. Specify here the new values of the parameters you changed. **Your Answer:**

(d) **Extra credit:** Rotate the coordinate system so that it is aligned to the direction of maximum coverage, and find the minimum bin size in the x and y directions that gives a satisfactorily uniform fold in the midpoints space.

(e) In a few sentences, describe qualitatively how *Fold3d* works. **Your Answer:**

(f) **Extra Credit:** You can apply some trickery to (almost) compute a fold map using *Stack3d* and *Math*. Hints: 1) You can do it with three or four command-line calls. 2) Use the *normalize* command to *Stack3d*. Write the necessary commands below. Also explain the limitations of your approach. **Your Answer:**

3D land geometry

In this section you will process a 3-D land dataset, “Shorncliff”, which was used by Chemingui in his Ph.D. thesis (SEP-101). The commands used to build the figures in this section are contained in the makefile called *BIGakefile*. Open it with your favorite editor and follow what happens. Your answers to the questions in this section are strongly linked to this makefile.

1. What is the plot in Figure 7? Modify the figure caption to answer the question.

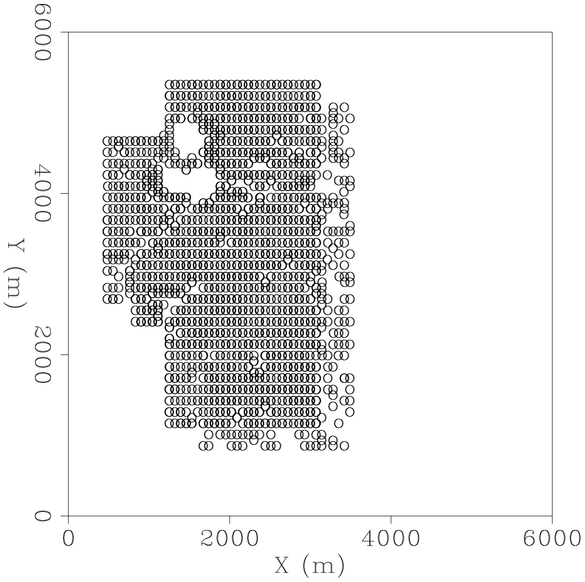

Figure 7: **Your Answer:**

2. What is the plot in Figure 8? Modify the figure caption to answer the question.

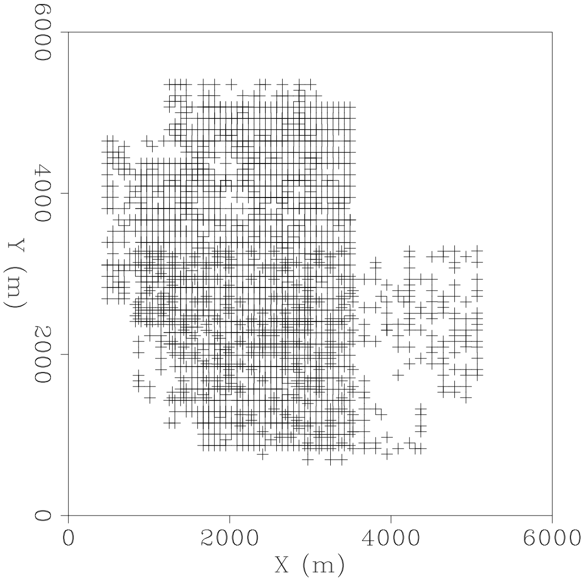

Figure 8: **Your Answer:**

3. What is the plot in Figure 9? Modify the figure caption to answer the question. Can you explain the straight line in the middle ?
4. What is the plot in Figure 10? Modify the figure caption to answer the question.
5. Use the command *make Bstk* to see a stacked cube for the grid defined earlier in the section. This operation uses the program *Sep_cube*. Familiarize yourself

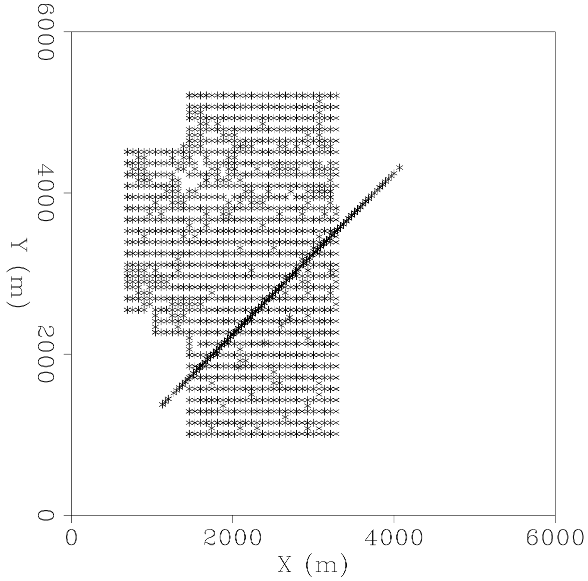

Figure 9: **Your Answer:**

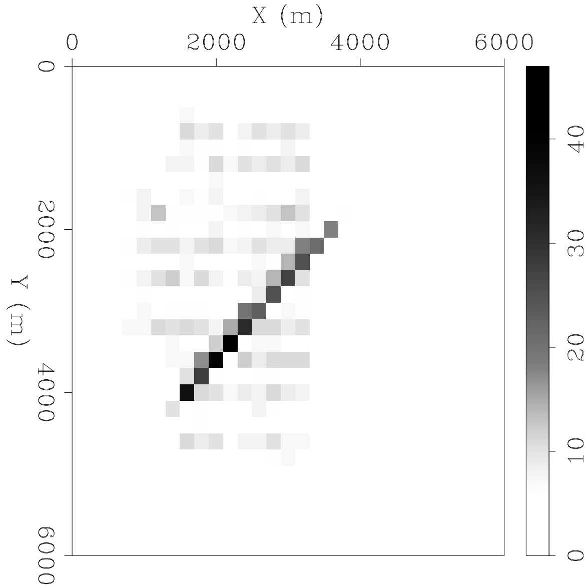

Figure 10: **Your Answer:**

with its menus, because it is going to be extremely useful in the later labs. Illustrate your answers with figures of your choice (put your rules in *BIGakefile*, but remember to keep all makefile variable definitions in alphabetical order in the same place, which for this directory is the beginning of the *Makefile*). The command *Cubeplot* is recommended for your figures.

(a) Is this a normalized stack with respect to the number of traces in each bin? How can you tell? **Your Answer:**

(b) Modify the makefile to change this. Run the command *Rm Bstk.H;make Bstk* to see the cube again. Which version do you like better, normalized or not? Why? **Your Answer:**

(c) Does it make a big difference if you regrid the data to larger bins? Double the bin size by modifying *par/bigsortcmp.P* and re-run *make Bstk*. **Your Answer:**

(d) Do you prefer larger or smaller bins? Why? **Your Answer:**

ALL DONE

When you are all done, 1) put a pdf copy in the online Dropbox and 2) clean up your directory (*make clean*).

REFERENCES