Data Analysis in Geophysics (CERI 7104/8104)
Final Project – Due 12/12/18

This project will tie together the pieces of code that you have been writing throughout the semester in order to perform seismic backprojection to locate a large earthquake.

1. Download seismograms from the IRIS website (in the same way as for HW 3) for an earthquake of your choice. You should use stations for the US Transportable Array (code TA, the same as you used on HW 3) for an event that occurred somewhere in the Pacific with at least a magnitude of 6. The method we are using tends to work best for these types of teleseismic events that are a certain distance away from the array, so you will have the best success if you pick such an event. You should download at least 20 stations on an approximately square portion of the array (the stations are roughly arranged on a grid). You only need to download the vertical component for this analysis.

   You should download the signals starting at the event time and choose an end time that ensures that you have a time series that contains all of the p-wave signal. This may be as much as 30 minutes of data in total, depending on your choice of event and stations. You can get a reasonable estimate of this by trying different values on the preview screen that comes up when you select a station from the map.

   **Note:** It is *extremely* important that you download your seismograms relative to the *event* time, and not the p-wave arrival time (which is the default for the IRIS web interface). The backprojection method requires all of the signals to be synchronized in time, and thus you need to have all your recordings start from the same time value. If you do not do this, your calculations will not work correctly!

   Additionally, you should download the Centroid Moment Tensor solution for your earthquake from the `https://www.globalcmt.org/` website. You can download the moment tensor in the correct format for `psmeca` directly from the Global CMT site.

2. Once you download your data, perform the analysis as follows:

   (a) Extract all receivers from the SAC header files using AWK, writing the station information to file.

   (b) Define the grid of source lat/lon locations and write all possible source coordinates to a text file. This can be done with AWK, or if the source points are integers, using `seq` as described in the class on shell scripting. I used a grid spacing of 1 degree, and you should pick the limits of your grid to be large enough that you can clearly see the location (see the file provided for HW 4 for an example of how your results should look once you are finished).

(c) Use your Python code to compute the travel times from all source points to all receiver points. Save the output of your Python program as a text file names "traveltimes.txt" using output redirection in the shell.

(d) Filter all seismograms using your SAC macro from HW 3 between 0.25 and 1 Hz. Write all filtered seismograms to disk, but do not overwrite the originals. You should call your SAC macro directly from the shell script, or embed your SAC commands directly in the shell script (either is fine with me).

(e) Read all waveform data and travel time information into MATLAB using your code from HW 2 (you will need to modify your driver script and replace the travel time function in the backprojection function with the one that I provide, but the other functions will work with no modification). Perform the backprojection and write the resulting summed amplitudes for all possible source locations to a text file with the format `sourcelat sourcelon amplitude` for each possible point on a separate line.

(f) Use your GMT script to plot the summed amplitudes, with the beach ball for the CMT solution on top using your script from HW 4. Your script should work directly with the output of your MATLAB code without having to modify anything, and it should be called from your driver shell script directly.

Note that the entire process from start to finish needs to be automated from your shell script. It should take 15 minutes or so to perform the full analysis if you pick 1 degree spacing for the source points (test your code using a coarser resolution, then increase the density of source points when you have everything working). Ideally, you should be able to run your code on *any* earthquake with little modification, just changing the latitude and longitude values of the source points, and the location of the SAC files to be analyzed (all of these parameters should be shell variables defined at the beginning of your script that can be easily changed). I will not enforce this too strictly when grading as long as the entire process runs automatically and is reasonably organized, but if I notice that you hard code too many things that you can easily determine in some other way automatically, I will take off points.

3. This will require writing a few additional pieces of code beyond what you have already done for the homework assignments:

   - You will need to write an AWK program to extract all station locations from the header files that are automatically downloaded when you obtain data from IRIS. Once you extract all of the station locations, you should save them into a text file following the format that works with your Python program to calculate travel times.

- You will need to change your MATLAB driver script from HW 2 to carry out the backprojection analysis. In particular, you will need to change the script to read the SAC files using the MATLAB function that I provide, and collect them into a matrix in the correct format for your other functions, and then perform the analysis. Additionally, you will need to replace the `travel_time` function with the one that I provide below to query the travel times computed with your Python program.
- Finally, you need to write a driver shell script to tie all of the above pieces together in your analysis. The shell script should generate the text files containing all potential source points, use AWK to generate the text file for all receiver points, and then call all of the programs that you have already written. Note that you will need to run a MATLAB program from the shell in order to do this; the following shows how you can call MATLAB from the command line:

  ```
  /Applications/MATLAB_R2016b.app/bin/matlab -nojvm -nodisplay
        -nosplash < matlabscript.m
  ```

  You will need to include an `exit` command in your MATLAB script in order to ensure that your entire script runs to completion automatically. I found that I needed to modify the MATLAB path to put the working directory at the beginning. You can modify the MATLAB path by entering `path(<newdirectory>, path)`, where `<newdirectory>` is the directory where you have all of your codes and data files for analysis.

  I found it useful to use a cell array to read the SAC files into MATLAB. You can do this by writing all of the SAC files to a text file, then reading that text file into a cell array. Then you can loop over the cell array to read in the appropriate files and place them into the correct place in the data matrix.

4. I am providing two additional MATLAB functions to help you perform your analysis. The first is `readsac`, a function for reading SAC files into MATLAB. Usage is `[t, a, p] = readsac(filename)`, which will load the time values into `t`, the signal amplitudes into `a`, and header information into `p`. The other function replaces the `travel_time` function that you wrote in order to look up the travel times computed with your Python program. It accepts the same arguments, but instead of computing the travel times directly, it loads the text file "traveltimes.txt" and looks up the results. It accepts the same arguments as before, so you just need to replace your original function with this one for it to work. The function will raise an error if it cannot find the file "traveltimes.txt" or if it does not contain one of the requested travel times.