Slab2 – Code Documentation

Written by Ginevra Moore and Gavin Hayes (ghayes@usgs.gov)
Edited by Kirstie Haynie

This document lists the procedure for accomplishing every step in the Slab2 modeling process using a MacOS. These steps have not been tested on Windows or Linux. Begin with the installation section. If adding data to the current Slab2 data set, start with the section that best matches the data type you are adding. If creating a slab model from the data available in the Slab2 repository, or if reproducing the Slab2 models, then start at the section Creating an Input File.

Contents

Installation

1) If a version of Anaconda is not already installed, install the latest version of Anaconda. Instructions for installation can be found at https://www.anaconda.com/download/#macos. These instructions also work with miniconda (https://conda.io/miniconda.html latest version on MacOSX)
2) If not already installed, install GMT5. A resource for install instructions is found at http://gmt.soest.hawaii.edu/?id=Installing. These instructions will likely work with GMT4 as well, but have not been tested.
3) Add the path to GMT to your .bash_profile or .bashrc. For example, add: PATH="/Applications/GMT-5.1.2.app/Contents/Resources/bin:${PATH}".
4) Download slab2 from GitHub (slab2-master.zip). Unzip the folder (now called slab2-master) and move it to your preferred location referred to here as [Slab2_location].
5) Navigate to [Slab2_location]/slab2-master/slab2setup.
6) Now, we will create a virtual environment. In the terminal (using a bash Unix shell) enter: *bash slab2env.sh*
    a) This will take about 5-10 minutes to run depending on internet speed
    b) Once it has finished running, check to see if any error messages occurred during the run.
    c) If any error messages occurred, the environment just built must be removed, and pip and conda should be updated:
        i) Enter: *conda env remove -n slab2env # this removes the environment*
        ii) Enter: *y*
        iii) Enter: *conda update conda # updates conda*
        iv) Enter: *pip install --upgrade pip # updates pip*
        v) Enter: *bash slab2env.sh*
    d) If no error messages occurred, proceed
7) This created a virtual environment for Python 3.7 (slab2env).
8) To activate the environment:
    a) In the terminal enter: *conda activate slab2env*
    Note: You must be in this environment in order to run any of the slab2 code.
9) To deactivate the enniornment:
    a) In the terminal enter: *conda deactivate*

Synthetic Test

We have created a synthetic data set (exp_04-18_input.csv) and have placed it in the *[Slab2_location]*/slab2-master/*slab2code/Input* directory. A companion parameter file also exists

at *[Slab2_location]*/slab2-master*/slab2code/ library/parameterfiles/expinput.par*. This can be used to test that the code is working properly before running further models. The figures below show the synthetic data and the model that the code should produce, following the steps below in **Creating a model.**

The input data and model shown below were created with GMT tools (Project, Blockmean, Surface). One can follow the steps in the script 'makesynth.pl' in *[Slab2_location]*/slab2-master*/slab2code/SYNTH* for plotting. The input model (used as a reference surface for the Slab2 search) includes only the projected input data shown below, while the Slab2 code by default also uses an Average Active Source profile, and so differs slightly from the input model in its shallowest extent.

The output created via this process is included in *[Slab2_location]*/slab2-master*/slab2code/*Output/exp_slab2_06.14.18. The user should be able to reproduce these files.

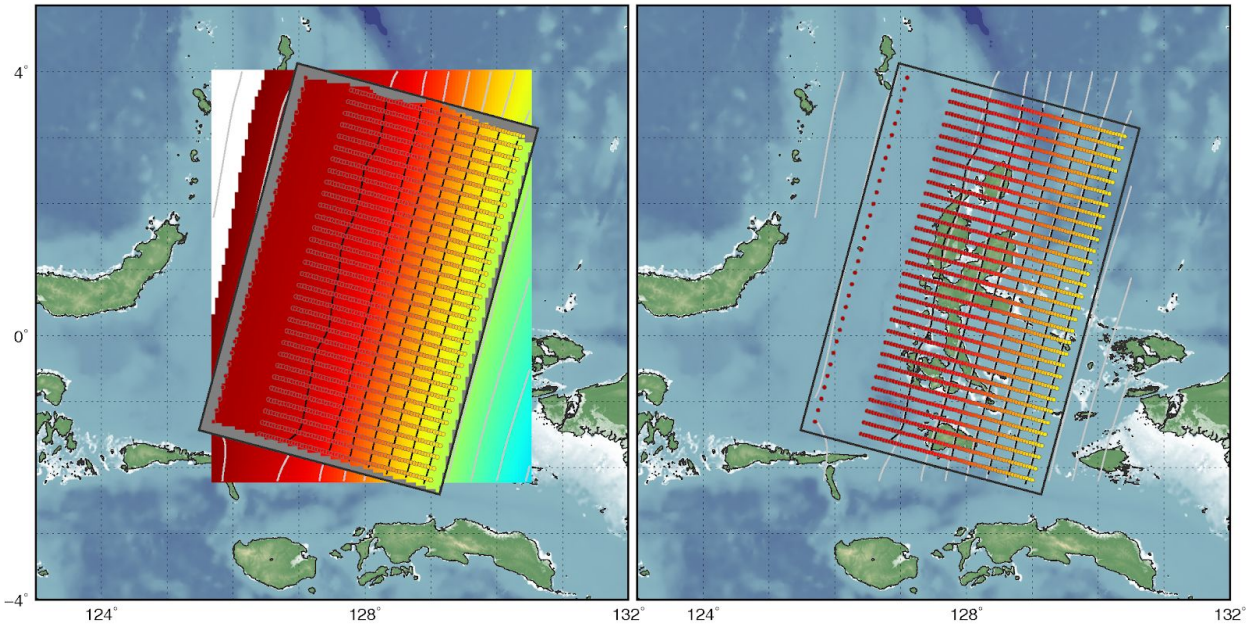

Figure 1: Left - synthetic data (circles colored by depth) and input surface (rectangular colored grid in the background), plotted versus the recovered Slab2 surface (clipped grid within black rectangle). Right - input data (circles colored by depth) plotted versus the contours of the input data surface (gray) and the recovered Slab2 surface (black). Note the input data surface includes only data shown here, while the Slab2 surface also uses the global average active source profile (see main manuscript for details). Thus, the two surfaces differ in the shallowest trench region between the trench and first input data points shown here.

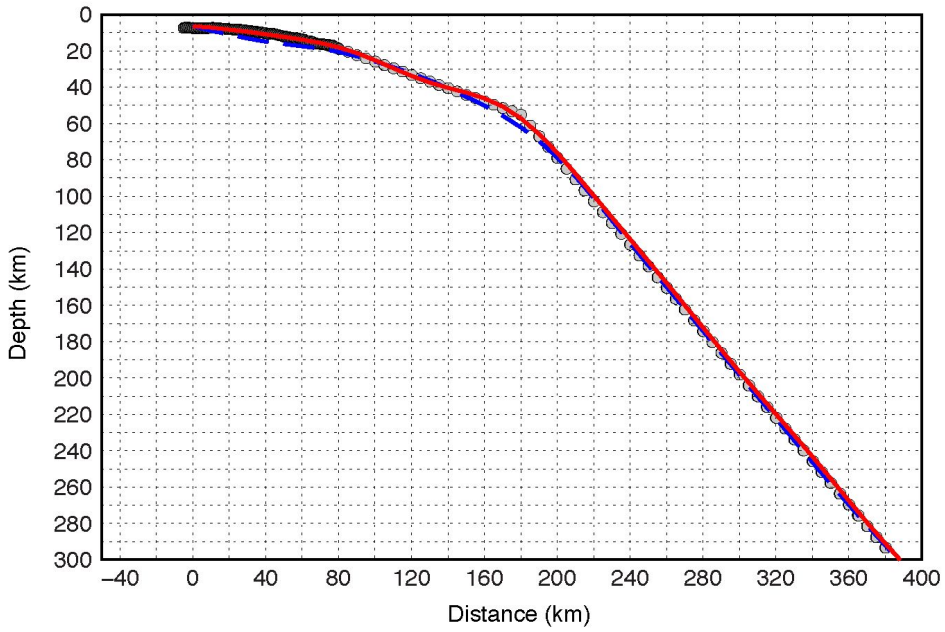

Figure 2: Cross section of the synthetic data (gray circles), the input model (blue dashed line) and the recovered Slab2 model (red line).

Setup Bathymetry and Sediment Thickness Datasets

- If you would like to include new bathymetry and/or sediment thickness datasets, then proceed with the following steps and then jump to the section **Making a bathymetry file**.

1) Navigate to [Slab2_location]/slab2-master/misc/bathymetry.
2) Activate environment: *conda activate slab2env*
3) Convert coarse sediment thickness data to a CSV file:
    a. gmt grd2xyz sedmap.grd | awk '{if(NR==1) print "lon,lat,thickness\n"$1","$2","$3; else print $1","$2","$3}' >sedmap.csv
4) Unzip and convert dense sediment thickness data to a CSV file:
    a. Gunzip sedthick.xyz.gz, then
    b. awk '{if(NR==1) print "lon,lat,thickness\n"$1","$2","$3; else print $1","$2","$3}' sedthick.xyz > sedthick2-2.csv
5) Download GEBCO bathymetry data (https://www.gebco.net/). We used the one min gird (https://www.gebco.net/data_and_products/gridded_bathymetry_data/gebco_one_minute_grid/). The GEBCO_2019 global grid is now available in a NetCDF file format. If you would like to use that, please convert from NetCDF to CSV. Else, use the included 5minx5min grid (gebco5x5.grd) and skip (a) below.

a. Downsample resulting grid to 5x5 minute data.
b. Convert the grd filed into a CSV file (gebco5x5.csv) using,

```
gmt grd2xyz gebco5x5.grd | awk '{if(NR==1) print
"lon,lat,elev\n"$1","$2","$3; else print $1","$2","$3}' >gebco5x5.csv
```

6) Run *mergest.py* by typing *python mergetest.py* in the terminal. This will create the file totalBA.csv, which combines bathymetry and sediment thickness datasets.
7) Follow the steps in **Making a bathymetry file** to update the existing bathymetry files or add a file if one does not exist for a specific subduction zone (check [Slab2_location]/slab2-master/MasterDB/bathymetry).

Making a Bathymetry File

- Before making a new bathymetry file, follow the steps in **Set Up Bathymetry & Sediment Thickness Datasets** to first generate the file totalBA.csv.

1) Navigate to [Slab2_location]/slab2-master/misc/bathymetry and activate the slab2 environment: *conda activate slab2env*
2) Open *newbathymetry.py* in text editor of choice
3) Look for line with "edit next two lines to create bathymetry files"
4) In the lines between the commented strings, edit the list of slabs to create bathymetry files for and name path to trench file that will be used to constrain the search.
    a) The trench file must have columns: lon, lat, az, bound, slab
        i) If the trench file is not already made, it would be good to first generate a file in this format using the steps/tools listed in **Adding a trench/modifying trench file**
    b) The trench file can have any number of other columns, and the order does not matter
    c) Every slab in the slablist in *newbathymetry.py* must be listed somewhere in the slab column in the trench file
5) To create a new bathymetry file for every slab in the slab list, in the terminal enter: *python newbathymetry.py*
6) Depending on how many bathymetry files are being made, how densely spaced the trench file is, and how long the trench is for each slab in the slablist the script will finish in ~1-20 minutes
7) The output files will be written to [Slab2_location]/slab2-master/misc/bathymetry/[slab]_BA_bath.csv
    a) Where [slab] is the three letter slab code associated with each region in the slablist.

8) Move the new files to the bathymetry folder in the master database ([Slab2_location]/masterDB/bathymetry)
9) Create a new input file for this slab following steps in **Creating an input file**.

Adding or Modifying a Trench File

- When adding a new slab to the database, if the subduction in this region initiates at an oceanic trench, a set of points delineating the trench segment must be acquired then processed to have the format, spacing, and details required by *slab2.py*.
- For consistency, all trench segments should be made with the tools outlined in this section.
- The first three steps discuss setting up trench sections. If a set of points delineating the trench already exists, proceed to step 4.

1. Check the USGS trench file (found in [Slab_location]/slab2code/library/forplotting/trenches_usgs_2016_old.csv) to see if the slab region has a segment (lon, lat coordinates that represent the trench location).
2. If the USGS file does not have a segment for this slab, search the literature for a segment. A good place to start this search is with Bird, 2003. To do this, go to http://peterbird.name/publications/2003_PB2002/2003_PB2002.htm to see if a plate boundary exists in this database over the margin associated with this region. If one exists, go to http://peterbird.name/oldFTP/PB2002/PB2002_boundaries.dig.txt and identify the segment(s) that delineate the trench for this slab model.
3. As a last resort, if a trench cannot be identified, digitize the trench based on topography using an appropriately colored and labeled map.
    a. If the segment comes from a source not listed in [Slab2_location]/misc/trenches/citations.txt, add this citation to that list. Include the slab code name and the trench segment source.
4. Convert the list of lon, lat points to a CSV file, with a third column listing the three letter slab code in every row, and a fourth column listing the plates on either side of the trench in every row.
    a. For example, the first two rows in the alu trench segment might be:

        lon,lat,slab,bound
        167.1909,54.0207,alu,PA/NA
        167.4184,53.9208,alu,PA/NA
5. Label the columns of the file lon, lat, slab, bound. Other columns may exist, but will be discarded throughout the initial sorting process.
6. Make sure that the last point in the file is the point that you want to start sorting from according to the right hand rule. In other words:

a. Picture yourself walking along the length of the trench
b. If the slab would be subducting to your right for the whole walk, which point on the trench would you start from? Call this [lon1,lat1].
c. Now make sure that the row containing [lon1,lat1] is listed as the last row of the file.
d. For example, the last point in the south America trench should be the most southern point, for the Aleutians, the most eastern point, for Kermadec, the most northern point, for Papua New Guinea, the most western point.
e. Don't worry about sorting the rest of the trench data. This will be done using the tools listed in the next few steps.
f. Many trenches can be made at once. Just list the segments one after another in the same CSV file with the row organization in step (4). No special marker between each trench segment is necessary.
g. For example:

```
Lon,lat,slab,bound
-137.9525,59.0277,alu,PA/NA
-137.819,58.9251,alu,PA/NA
-26.071,-60.391,sco,SA/SC
-25.8205,-60.2865,sco,SA/SC
```

h. Make sure that the last row for each slab follows the last row rule listed in step (6).

7. Save the file to a logical place, and don't forget to document the source(s) in [Slab2_location]/slab2-master/misc/trenches/citations.txt
8. Navigate to [Slab2_location]/slab2-master/misc/trenches
9. Use *sortpoints.py* to sort the list of points according to the right hand rule, and calculate the azimuth between consecutive points:
    a. In the terminal enter: *conda activate slab2env*
    b. Enter: *python sortpoints.py -n [newtrenchlist] -t [trenchfile]*
        i. [newtrenchlist] is the file you just made listing the new trench points to sort and calculate azimuths for
        ii. [trenchfile] is the output file to save the sorted trench segment to
10. After the trench segment(s) have been sorted and azimuths have been calculated for each point, use *trenchsmooth.py* to smooth over any outliers that may have slipped through the sorting process.
    a. In the terminal enter: *python trenchsmooth.py -n [newtrenchlist] -t [trenchfile] -s s*
        i. [newtrenchlist] is the file listing the sorted trench points with azimuth values to smooth
        ii. [trenchfile] is the output file to save the smoothed trench segment to.

iii. *-s s* indicates that we are smoothing the trench segment, rather than filling in (filling in explained in next step).

11. After the trench segment(s) are smoothed with *trenchsmooth.py*, use *trenchsmooth.py* to make sure that the distance between each consecutive trench point is no more than 20 km. This is necessary for regions with very linear trenches that can be plotted with very few vertices. Often the list of points describing these straight trench segments are very far apart, which is bad for the purpose that these points serve in *slab2.py*.
12. *trenchsmooth.py* linearly interpolates between these far away points until a sufficient point density is attained.
    a. In the terminal enter: *python trenchsmooth.py -n [newtrenchlist] -t [trenchfile] -s f*
        i. [newtrenchlist] is the file listing the sorted trench points with azimuth values to smooth
        ii. [trenchfile] is the output file to save the new smoothed trench segment to.
        iii. *-s f* indicates that we are filling in the trench segment, rather than smoothing the azimuth values.
    b. The file listed as [trenchfile] is in the proper format to be added to the global trench dataset in the slab2code library. Identify the currently used trench file in the library located in [Slab2_location]/slab2code/library/misc/trenches_usgs_2017_depths.csv. If no trench information for this slab exists, add the new slab trench data to end of this file. If just updating pre-existing slab trench information, replace the new data with the old data for the specific slab trench segment.

Adding a Dataset

1) Reorganize any new dataset into a CSV file format with labeled columns. Document where the dataset came from and any modification details in masterdb.xlsx. Some things to note:
    a) Available list of column names is: time, lat, lon, depth, mag, mrr, mtt, mpp, mrt, mrp, mtp, type, mlon, mlat, mdep
    b) All columns with a name not listed in (1a) will be discarded
    c) The order of the columns is inconsequential
    d) The file at *minimum* must have labeled columns of lon, lat, depth
2) Add the new file to the appropriate folder in MasterDB. The database is organized by data type. The name of the file is important:
    a) File naming convention is [slab]_[data type]_[source ID].csv
    b) [slab] is the three letter slab code (e.g. alu, sam, cam, etc.)

c) [data type] is the two letter abbreviation indicating the kind of slab related dataset. Available list of event types is: EQ, ER, AS, RF, BA, CP, TO, SR, ST
d) Abbreviations of SR and ST are merged into the same event type as AS in the modeling algorithm.
e) [source ID] can be any name referring to the source of the data.

3) If a common folder containing all datasets listed in the masterdb spreadsheet exists and is up-to-date, add this file to that folder, and proceed to step (3) in **Creating an input file**. If this folder does not exist, proceed to step (1) in **Creating an input file.**

Querying the Preliminary Determination of Epicenters (PDE) Bulletin

1) Navigate to [Slab2_location]/slab2-master/catalogquery and activate the slab2 environment: *conda activate slab2env*
2) Since the catalog takes an outrageous amount of time to query in its entirety, the querying script, *newcomcatquery.py*, prompts the user for an interval between 1 and 11 with the assumption that the user will run ten simultaneous queries and merge the files once all are finished. This drops the amount of time from > 2 weeks to ~15 hours. There is an option to just update the existing catalog as well, which is described in **Adding to existing catalog query.**
3) To query the whole catalog at the same time, open 11 terminal windows and repeat steps (1) and (2) in each. For a later step, it is important that these are new blank terminal windows.
4) In each terminal window, enter: *libcomcatquery.py -d [qdir] -i [intervalno]*
    a) [intervalno] is an integer between 1 and 11
    b) [qdir] is a folder name within this directory that all query files will be saved to
5) The entire query is designed to run overnight, and should take around 15 hours
6) After all queries are completed, save the printed terminal output for each window into a new directory. These files will be searched using *getfailedevents.py* for events that failed using the libcomcat API (due to random timeouts).
7) Once all terminal content is saved, run *getfailedevents.py*:
    a) In the terminal window enter: *python getfailedqueries.py -f [fdir] -q [qdir]*
        i) [fdir] is the directory where the terminal outputs were saved
        ii) [qdir] is the directory containing the output files of the original query.
    b) A new query of all failed searches will be saved in [qdir], which is used in the next step
8) In order to combine all of these queries and condense the amount of columns returned, use *concatcomcat.py* to concatenate all of the output files in [qdir]
    a) In the terminal window enter: *python concatcomcat.py*
    b) Enter *1*

c) Enter *[qdir]*
d) Wait ~ 5 minutes
e) The concatenated and condensed file will be written to [qdir].csv

9) Once a comprehensive query has been attained, it should be associated with the most current version of GCMT for use in Slab2. See **Associating PDE and GCMT** for next steps.

Adding to an Existing Catalog Query

1) Navigate to [Slab2_location]/Slab2_Code/catalogquery and activate the slab2 environment: *conda activate slab2env*
2) In the terminal window, enter: *python libcomcatquery.py -d [qdir] -i 100 -p [previousquery]*
    a) [qdir] is a folder name within this directory where all query files will be saved.
    b) [previousquery] is the file (e.g., pde_0618.csv) that we are adding more recent events to with this query. The script will take the date of the most recent event of this file then search from that date/time to present over the globe.
3) Wait for the query to finish. This will take longer for longer time spans.
4) Add this file to the existing query [previousquery].
    a) First we need to convert the file to the condensed slab2 format. Use concatcomcat for this:
        i) In the terminal enter: *python concatcomcat.py*
        ii) Enter: *1*
        iii) Enter: *[qdir]*
    b) A file in the correct format will be written to [qdir].csv
    c) Now add [qdir].csv to [previousquery]:
        i) In the terminal enter: *python concatcomcat.py*
        ii) Enter: *2*
        iii) Enter: *[previousquery]*
        iv) Enter: *[qdir].csv*
        v) Enter: *pde_MMYY.csv*

    Where MMYY is the month and year of the most recent event in the new query.
5) Every time the catalog is updated, it should be associated with the most current version of GCMT for use in Slab2. See **Associating PDE and GCMT** for the next steps.

Adding Data from the Global Centroid Moment Tensor (GCMT) Catalog

1) Navigate to [Slab2_location]/slab2-master/catalogquery and activate the slab2 environment: *conda activate slab2env*
2) Identify most recent event in the latest gcmt catalog on file. This will be listed in the catalogquery folder as gcmt_MMYY.csv.
3) Go to http://www.ldeo.columbia.edu/~gcmt/projects/CMT/catalog/NEW_MONTHLY/ and see if any months have been added since MMYY.
4) Copy and paste each new MMYY to individual files. Put all of these files into a single folder.
5) Use *makecolumns.py* to merge all of these files into a single column file.
    a) In the terminal enter: *python makecolumns.py*
    b) Follow prompts
    c) Two files will be written:
        i) Converted file listing all events in new folder in appropriate format.
        ii) New updated gcmt file in the appropriate format.
6) Every time that this file is updated, it should be associated with the most current version of the PDE for use in Slab2. See **Associating PDE and GCMT** for next steps.

Associating the PDE and GCMT Data

1) Navigate to [Slab2_location]/slab2-master/catalogquery and activate the slab2 environment: *conda activate slab2env*
2) Identify the most recent gcmt catalog query (named gcmt_MMYY.csv) and PDE catalog query (named pde_MMYY.csv).
3) Make sure gcmt catalog is in CSV format with the column: [year month day hour minute second lat lon depth gmlat gmlon gmdep mrr mtt mpp mrt mrp mtp smo expo mag fss fth fclvd]
4) Make sure PDE file is in .csv format with columns: [id_no, time, lat, lon, depth, mag, mag type, moment_lon, moment_lat, moment_depth, mrr, mtt, mpp, mrt, mrp, mtp, type]
5) It is okay if there are duplicates in either file, they will be removed within the association code.
6) If it does not exist, make a directory called “matchfiles”
7) In the terminal enter: *python pdegcmtdupassc.py -g [gcmtfile] -c [ccatfile] -f [assofile]*
    a) [gcmtfile] is the gcmt file to be associated
    b) [ccatfile] is the comcat/pde query file to be associated

c) [assofile] is the filename to write the associated output to

8) This could take up to ~1 hour to run
9) Extra information files will be written to [Slab2_location]/slab2-master/catalogquery/matchfiles.
10) Move the new associated file to [Slab2_location]/slab2-master/MasterDB/gcmt_pde_assc
11) Rename the file to ALL_EQ_MMDDYY.csv where MMDDYY indicates the date of the most recent event in the catalog or the date of the PDE query.
12) Remove the other existing file in this folder, as this file is a more updated version.
13) Document this change in the first line of the file: [Slab2_location]/slab2-master/MasterDB/masterdb.xlsx
14) Make a new folder as described in step (1) of **Creating an input file,** or if no other changes to that folder need to be made, replace the existing ALL_EQ_MMDDYY.csv with this newfile, and rename the folder to reflect the update in month and year.
15) To make new Slab2 input files with the updated catalog query, see **Creating an input file.**

Removing Data Points

- If an event or a cluster of events is clearly not associated with the slab but is not filtered by the broader filter specifications associated with the slab model, it may need to be added to a remove file for the slab model.
- Removing a point is the last resort option to filtering an event after the best filter parameters are identified.

1) Make a file listing the points to be removed in CSV format.
    a) The file must at minimum have the columns: lon, lat, depth, etype
    b) The file can have any number of other columns, and the order does not matter
2) Document why the points are being removed, name the file [slab]_[desc].csv, and save the file in [Slab2_location]/slab2-master/slab2code/library/points_to_remove/current_files/
    a) Where [slab] is the three letter slab code associated with this region
    b) And [desc] is a short descriptive word or phrase that hints as to why the points in the file are being removed from that slab.
3) Re-run the model using *slab2.py*. The points in the added remove file will then be filtered from the input dataset at the onset of the modeling algorithm.

Running L-Curve for a Model

1) After identifying the optimum parameters for a given model, the optimum smoothness must be determined. Navigate to [Slab2_location]/slab2-master/slab2code and activate the slab2 environment: *conda activate slab2env*
2) To generate an L-curve for a 12.22.17 suluwesi model stored in the Output directory of slab2code:
    a) In the terminal enter: *python lcurvewrap.py -p library/parameterfiles/sulinput.par -f Output/sul_slab2_12.22.17*
3) The code should be run overnight for larger models (sam, sum, kur, alu), and takes between ~10 minutes and ~12 hours to complete.
4) An L-curve plot will be saved in the folder listed after -f in the input, along with a list of filters and the associated misfits.
5) Visually interpret the x value of the curve apex in the left subplot of [slab]_slab2_lcv_[date].png
6) Find the x value from previous step in the filt column of [slab]_slab2_lcv_[date].csv, then identify the actualfilter column for that row and use it as the smoothing filter for this slab.
7) Run the model with the suggested smoothing parameter, (change the filt value in the parameter file) and visually assess whether this value makes sense for this slab.
    a) The filters are often too high, deteriorating the geometric features of slabs with steep dips at shallow extents.
    b) If the model seems too smooth, continue making models, decreasing the suggested filter width for each run, until a suitable smoothing balance is achieved.
8) A histogram of difference values between each data/node and the surface for each smoothing value is saved in the model output directory in a directory with the name diffhists
9) The two plots in the Lcurve figure that are not the L curve are the first and second derivatives of the Lcurve (derived as the slope of the lcurve, then the slope of the slope of the lcurve).
    a) These were used as an attempt to identify the inflection point of the curve, which doesn't work. A different approach is needed to quantitatively identify the L-curve apex.

Creating an Input File

1) Create a single directory located in [Slab2_location]/slab2-master/ that contains all datasets within the folders of MasterDB. To do this, simply copy and paste the contents

of each folder into your new folder. There should already be older dataset folders (e.g., 04-18database) that you can refer to for an example.

2) A good name for the new directory is MMYYdatabase, where MMYY is the month and year of the most recent query in the PDE. MM and YY can be identified by the name of the file (ALL_EQ_MMDDYY.csv) located in [Slab2_location]/slab2-master/MasterDB/gcmt_pde_assc.
3) Navigate to [Slab2_location]/slab2-master/slab2code in the terminal and activate the environment: *conda activate slab2env*
4) For example, if creating an input file for Alaska (*-p alu*), with a database and earthquake catalog that was queried on Dec 2017 (*-d [Slab2_location]/*slab2-master*/12-17database*) and saving the file using the appropriate naming convention (*-f alu_12-17_input.csv*) :
    a) In the terminal enter: *python s2d.py -p alu -d [Slab2_location]/*slab2-master*/12-17database -f alu_12-17_input.csv*
    b) This will take about 5 minutes to run for most regions.
    c) In the terminal enter: *mv alu_12-17_input.csv Input*
5) Note that the input file and output name must be in format [slab]_[MM]-[YY]_input.csv where [slab] is the three letter slab code, [MM] is the month of the associated database, and [YY] is the year of the associated database.
6) For more help and descriptions, see documentation within *s2d.py* and *s2d_fnctns.py*. For example, entering *python s2d.py -h* in the terminal will give more details on the running options and required arguments.

Creating a Slab Model

1) Navigate to [Slab2_location]/slab2-master/slab2code in the terminal and activate the slab2 environment: *conda activate slab2env*
2) If generating a complete model of the Aleutians slab using the input file with the most recent date and year (stored in [Slab2_location]/slab2-master/slab2code/Input), retaining all other predetermined parameters:
    a) In the terminal enter: *python slab2.py -p library/parameterfiles/aluinput.par*
3) To run the same model listed above, but in parallel using three cores:
    a) In the terminal enter: *python slab2.py -p library/parameterfiles/aluinput.par -c 3*
    b) This can take anywhere from 3 minutes to 3 hours to complete, depending on several variables including: how many cores are used, the grid resolution/size, the surface resolution/size, any other loads your system is under, if there is a vertical component to the slab, etc.

4) To change any parameters, navigate to [Slab2_location]/Slab2_Code/slab2code/library/parameterfiles, and edit the desired slab parameter file. It is important to maintain the same formatting.
    a) Parameters that one may want to change include:
        i) radius1 - long axis of search ellipsoid (km)
        ii) radius2 - short axis of search ellipsoid (km)
        iii) taper - depth over which to taper from no shift to full shift (km)
        iv) filt - width of smoothing filter (degrees)
        v) fracS - fraction of slab thickness to shift
        vi) grid - grid node resolution (degrees). Can range between 0.1 and 0.5.
        vii) minstk - maximum range of reference model strikes in search ellipsoid (degrees)
        viii) node - grid resolution of least squares interpolation (degrees)
    b) If changing any parameters, it would be useful to use the find command in the text editor that you are using in *slab2.py*, s*lab2_functions.py*, and *loops.py* for the string "if slab == '[slab]'" where [slab] is the three letter string associated with this region. These statements currently override any parameters in the parameter file, and are necessary to take in to consideration when modifying parameters. These are in place for:
        i) Sections within slabs that do not fit the general parameters for that region,
        ii) Regions that do not fit the general modeling approach at a specific step in the algorithm.
5) To plot general maps and cross sections, see **Creating generalized maps and cross sections**.
6) Once a general set of parameters is defined, see **Determining shift magnitude** to find the optimum shift magnitude parameter. This is found from comparing the width of the event distribution with the inferred slab thickness at each node.
7) After determining the parameters that make the post shifted nodes behave as desired, proceed to steps listed in **Running L-Curve for model** to determine the best width for the gaussian smoothing filter.
8) Several slabs can be generated concurrently, but only one can be generated in parallel at any given time.
9) The interpolation of the larger slabs (sum, alu, sam, kur) and medium sized slabs with vertical components (ker, izu) is very memory intensive, and will consume all RAM if several of them are run at the same time. It is fine to run these concurrently overnight (up to 4 of them) but this is not recommended if the machine needs to be used for anything else while they are running (e.g. in the middle of the workday).

10) For more help and descriptions, see documentation within *slab2.py* and *slab2functions.py*. Entering: *python slab2.py -h* will give more details on running options and required arguments.

Creating Generalized Plots and Cross-sections

1) Navigate to [Slab2_location]/slab2-master/slab2code
2) There are two perl scripts, *generalplot.pl* and *generalxsec.pl,* for plotting the slab models using GMT. Open each script and search for the first instance of "$gmt". If necessary, update the file path.
3) To make a series of generalized maps for the slab region, use generalplot.pl:
    a) In the terminal enter: *perl generalplot.pl*
    b) A list of required arguments will be listed that are needed to run the script. Run the script again with the required arguments, each separated by a space.
    c) For example, to plot the sam_slab2_01.03.18 model (stored in the Output folder) that was made with the 12-17 input file (stored in the Input folder) enter:
        i) *perl generalplot.pl sam 01.03.18 Output/sam_slab2_01.03.18 Input/sam_12-17_input.csv*
2) To make a series of general cross sections of the region, use generalxsec.pl:
    a) In the terminal enter: *perl generalxsec.pl*
    b) A list of required arguments, followed by a list of optional arguments will be listed. Run the script again with the required arguments, each separated by a space character. If any of the optional arguments are used, they all have to be used, or substituted by the flag "na" if the default for that option is desired.
    c) For example, to plot the ryu_slab2_01.04.18 model (stored in the Output folder) that was made with the 12-17 input file (stored in the Input folder) with a spacing of 50 km between each section and an azimuth perpendicular to the azimuth of the default trench file enter:
        i) *perl generalxsec.pl ryu 01.04.18 Output/ryu_slab2_01.04.18 Input/ryu_12-17_input.csv library/slab1grids/notused/ryu_slab1.0_clip.grd 50*
    d) To plot the same model but with a constant azimuth of 300º and include the slab guide grid enter:
        i) *perl generalxsec.pl ryu 01.04.18 Output/ryu_slab2_01.04.18 Input/ryu_12-17_input.csv library/slab1grids/notused/ryu_slab1.0_clip.grd 50 na 300 library/slabguides/ryu_11.20.17_15.50.grd na na na*

e) To plot with the same specs as in [c] but zoomed in to -10 < distance < 200 and depth < 150 enter:
   i) *perl generalxsec.pl ryu 01.04.18 Output/ryu_slab2_01.04.18 Input/ryu_12-17_input.csv library/slab1grids/notused/ryu_slab1.0_clip.grd 50 na na na 200 -10 150*

Calculating Seismogenic Zone Width and Organizing Files

- *sztcalc.py* takes a depth grid and an unfiltered Slab2 input dataset and calculates the width of the seismogenic zone based on a distribution of depths of slab related events. The events are deemed slab related based on a series of filters, using spatial and Kagan's angle information.
- This tool can be used to calculate the sz width for a model within a discrete bounding box or in an area bounded by a mask or polygon.
- The filelist within *sztcalc.py* must first be updated by the user prior to running the code.
  a) In *sztcalc.py*, search for "filelist = ["
  b) Change the contents inside the brackets of this line to the list of new models to calculate the seismogenic zone width for. Follow syntax and naming convention in previously existing list.
- The required inputs include:
  a) Input file information *(-[flag]: description of input for [flag])*
     i) -s: The path to the depth grid
     ii) -r: The model name ([slab]_slab2_[MM.DD.YY])
     iii) -i: The directory holding the input file
     iv) -t: The input file date (MM-YY)
     v) -f: The folder to save these files to
  b) Filter information:
     i) -l: Flag indicating origin or CMT lon, lat to compare with the slab (o or c)
     ii) -d: Flag indicating origin or CMT depths to compare with the slab (o or c)
     iii) -b: Flag indicating a distribution of event depths or slab depths (e or s)
     iv) -m: A depth filter to exclude points outside of in szt calculations
     v) -x: A depth extent to cut distributions off at
- Optional inputs include:
  a) -o: lonmin, lonmax, latmin, latmax
  b) -k: path to a clipping mask. Mask must have unlabled columns of lon lat, separated by white space.

1) Navigate to [Slab2_location]/slab2-master/slab2code and activate the slab2 environment: *conda activate slab2env*
2) To move a list of slab models contained in [Slab2_location]/slab2-master/Output to a new file structure in directory [newdir] and calculate the seismogenic zone width using input files stored in [Slab2_location]/slab2-master/Input:
    a) In the terminal enter: *python sztcalc.py -s [path to depth grid] -l [origin or cmt lat,lon] -d [origin or cmt depths] -b [slab or event depth histogram] -m [maxdepdiff] -x [distcutoff] -i Input -f [newdir] -t [MM-YY] -r [model]*
        i) *[path to depth grid]* is the path to the slab model
        ii) [origin or cmt lat,lon] can be "c" or "o"
            (1) "c" indicates to use the CMT lon, lat in spatial filters
            (2) "o" indicates to use the origin lon, lat in spatial filters
        iii) [origin or cmt depths] can be "c" or "o"
            (1) "c" indicates to use the CMT depth in spatial filters
            (2) "o" indicates to use the origin depth in spatial filters
        iv) [slab or event depth histogram] can be "e" or "s"
            (1) "e" indicates to make a histogram of event depths within the Kagan's angle and spatial filters. This will be a histogram of origin or CMT depths depending on the -d flag
            (2) "s" indicates to make a histogram of the slab depths queried at event lon, lat locations within the Kagan's angle and spatial filters. The slab depth will be queried at the CMT or origin lon, lat coordinates depending on the -l flag.
        v) [maxdepdiff] is the width of the spatial filter around the slab model in km.
        vi) [distcutoff] is the depth in kilometers that the histogram is cut off at
        vii) *-i Input* indicates that the input files being used to generate the depth histograms are stored in the Input directory
        viii) [newdir] is the folder where the new file structure and seismogenic zone thickness calculations will be saved
        ix) [MM-YY] is the date of the input files to use. Input file naming convention is [slab]_[MM-YY]_input.csv.
        x) [model] is in the form [slab]_slab2_[MM.DD.YY]
3) For example. to calculate the seismogenic zone width of alu_slab2_01.08.18 using event origins (*-l o*) and centroid MT depths (*-d c -b e*) with a near slab filter of 20 km depth (*-m 20*) and a distribution cutoff of 65 km depth (*-x 65*) using input files with date December 2017 (*-i Input -t 12-17*) and save them to ~/Desktop/alutest:
    c) In the terminal enter: *python sztcalc.py -s Output -f ~/Desktop/alutest -i Input -t 12-17 -l o -d c -m 20 -x 65 -b e*

d) A plot showing a histogram of slab related event depths and a best fitting double normal distribution will be plotted for the model in *~/Desktop/alutest*.
e) The file containing all events within specified Kagan's Angle and spatial filters will be saved in the same folder.
f) A table listing the upper and lower bounds of the seismogenic zone and the amount of events to constrain those calculations will be written to *~/Desktop/alutest/alu_slab2_szttable_01.08.18.csv.*

- To do the same as above, but bounded by a mask stored in *~/Desktop/alutest/alu_west_mask.txt:*
    a) In the terminal enter: *python sztcalc.py -s Output -f ~/Desktop/alutest -i Input -t 12-17 -l o -d c -m 20 -x 65 -b e -k ~/Desktop/alutest/alu_west_mask.txt*
    b) The histogram and kagan files that were saved here previously will be deleted if they were made for the same model. New tables indicating the final calculations, however, will be saved with different file names.
- To do the same as above, but with a bounding box instead of a mask with lonmin=160, lonmax=195, latmin=45, latmax=70:
    a) In the terminal enter: *python sztcalc.py -s Output -f ~/Desktop/alutest -i Input -t 12-17 -l o -d c -m 20 -x 65 -b e -b 160 195 45 70*
    b) The histogram and kagan files that were saved here previously will be deleted if they were made for the same model. New tables indicating the final calculations, however, will be saved with different file names.
- To organize the slab model output by output file type, rather than by slab model *sztcalc.py* can also be used.

4) Move all original model output folders to a common folder (if they are not already in a common folder). It is easiest to keep them in the original [Slab2_location]/slab2-master/slab2code/Output folder that they are written to from slab2.py.
5) Make a new directory [newdir] to save the new file structure and seismogenic zone thickness calculations to
6) To move a list of slab models contained in [Slab2_location]/slab2-master/slab2code/Output to the new file structure in directory [newdir] and calculate the seismogenic zone width using input files stored in [Slab2_location]/slab2-master/slab2code/Input:
    a) In the terminal enter: *python sztcalc.py -s Output -l [origin or cmt lat,lon] -d [origin or cmt depths] -b [slab or event depth histogram] -m [maxdepdiff] -x [distcutoff] -i Input -f [newdir] -t [MM-YY]*
        i) *-s Output* indicates that all of the models are in the [Slab2_location]/slab2-master/slab2code/Output directory.
        ii) [origin or cmt lat,lon] can be "c" or "o"

(1) “c” indicates to use the CMT lon,lat in spatial filters
(2) “o” indicates to use the origin lon,lat in spatial filters

iii) And [origin or cmt depths] can be “c” or “o”
(1) “c” indicates to use the CMT depth in spatial filters
(2) “o” indicates to use the origin depth in spatial filters

iv) And [slab or event depth histogram] can be “e” or “s”
(1) “e” indicates to make a histogram of event depths within the Kagan’s angle and spatial filters. This will be a histogram of origin or CMT depths depending on the -d flag
(2) “s” indicates to make a histogram of the slab depths queried at event lon,lat locations within the Kagan’s angle and spatial filters. The slab depth will be queried at the CMT or origin lon, lat coordinates depending on the -l flag.

v) [maxdepdiff] is the width of the spatial filter around the slab model in km.

vi) [distcutoff] is the depth in kilometers that the histogram is cut off at

vii) And *-i Input* indicates that the input files being used to generate the depth histograms are stored in the Input directory

viii) [newdir] is the folder where the new file structure and seismogenic zone thickness calculations will be saved

ix) [MM-YY] is the date of the input files to use. Input file naming convention is [slab]_[MM-YY]_input.csv.

b) If there are 27 slab models, the script should take less than a minute to complete.

c) A new file structure will be saved in [newdir] with folders:

i) clippingmasks: contains the polygons bounding each slab grid

ii) filtereddata: contains the list of data points that were used in constraining the slab model

iii) Grids: contains the grid formats of the slab geometry, thickness, and uncertainty. (including: depth, strike, dip, thickness, and uncertainty)

iv) inputdata: the original unfiltered input files for *slab2.py*

v) nodes: contains the lists of all original grid node locations that were used to constrain the final slab surfaces. Important information regarding filter dimensions and pre/post shifted locations is listed for each point.

vi) parameters: contains metadadata files listing the parameters that were used in creating this model.

vii) Supplement: contains ascii files representing the vertical component of the slab (for slab regions where it exists) and the associated geometry, thickness, and uncertainty information.

viii) Surfacetext: contains the ascii file versions of the geometry grids.

ix) Szt: the seismogenic thickness histograms for each new slab model will be saved in this directory, along with a general table and distribution plot.

x) Contours: contains the contour text files for vertical slab sections (where they exist).

xi) figures: contains simple postscript files showing the geometry grids, filtered and unfiltered datasets, and post shifted node locations (if the figures were generated prior to running this).

xii) crossections: contains cross sections showing the slab model and associated filtered and unfiltered datasets (if they were generated prior to running this).

7) To calculate the seismogenic zone width of a list of models inside of [Slab2_location]/slab2-master/slab2code/Output (*-s Output*) using event origins (*-l o*) and centroid MT depths (*-d c -b e*) with a near slab filter of 20 km depth (*-m 20*) and a distribution cutoff of 65 km depth (*-x 65*) using input files with date December 2017 (*-i Input -t 12-17*) and save them to [newdir] created in step 2 (*-f [newdir]*):
    a) In the terminal enter: *python sztcalc.py -s Output -f [newdir] -i Input -t 12-17 -l o -d c -m 20 -x 65 -b e*
    b) A plot showing a histogram of slab related event depths and a best fitting double normal distribution will be plotted for each slab model in [newdir]/szt.
    c) The file for each slab containing all events within specified Kagan's Angle and spatial filters will be saved in the folder [newdir]/szt.
    d) A plot of all of the summed double normal distributions will be saved as [newdir]/szt/allpdf.png.
    e) A table listing the upper and lower bounds of the seismogenic zone and the amount of events to constrain those calculations will be written to [newdir]/szt/table.csv.

- This script requires that all models have an original unfiltered input file from the same date, so multiple runs might be necessary to get around this if all input files were not made from the same database version.
- For more help and descriptions, see documentation within *sztcalc.py* and *slab2functions.py*. Entering: *python sztcalc.py -h* will give more details on running options and required arguments.

To-do List:

Determining shift magnitude - in **creating a model** section, readers are referred to this. Need to add in. Need to use centsurfhist.py ask Gavin to send to Ginevra make flow more simple
Directory /Lacie/Ginevra2017_slab2stuff/Slab2_code/centsurfhist.py

NEED TO MERGE maketrenchdepth.py WITH slab2.py !!!!
Ask Gavin to send Ginevra this script.

Adding a tomography dataset:

Modifying polygons:

Making or using a new slab guide:

- Make l curve parallel
- Make comcat query parallel
- Make slab models with new data
- Add in tests