LLNL-Earth3D (version 5.4.3+) User Manual

Nathan Simmons, Doug Knapp, Eric Matzel, Steve Myers
Lawrence Livermore National Laboratory
LLNL-SM-652345
Revised August 29, 2018

Table of Contents

1. Introduction

LLNL-Earth3D is a computer code designed to support the use of the LLNL-G3D series of models (see *Simmons et al*. 2011, 2012). The primary purpose of the code is to compute 3-D ray paths for various body waves and calculate travel times between a seismic source and a seismic station. The code navigates the hierarchical spherical tessellation framework of the LLNL-G3D models and is multi-threaded so that a number of ray paths and travel times can be computed concurrently. LLNL-Earth3D is also a more general model interface allowing for model property output, such as seismic velocity and the depth (or radius) of velocity discontinuities.

The LLNL-Earth3D code is written in Java and will work on any system with Java version 1.8.0 *(updated for version 5.4.3)* or higher.

1.1 General Background

LLNL-Earth3D allows for fast calculation of 3-D seismic travel times and ray paths through a 3-D model of the Earth specifically represented with the LLNL-G3D global-scale model architecture. The model architecture is node-based and consists of a series of surfaces that may undulate (*not necessarily spherical*). Therefore, Earth's ellipticity and undulating discontinuity surfaces (such as the Moho) are

explicitly represented. Each model surface is defined by a set of nodes evenly distributed around the globe, and multiple model properties (e.g. Vp, Vs, Q, etc.) may be defined at each node for a given surface. The positioning (latitude and longitude) of the model nodes are defined by the vertices (or intersections) of the spherical tessellation grids (Figure 1). The spherical tessellation grids are created by recursive subdivision of triangular faces on a spherical surface. Each recursive subdivision of a tessellation grid generates higher resolution surfaces and a new level in the tessellation hierarchy which may be exploited for fast model referencing (see *Simmons et al.*, 2011).

To represent undulating (aspherical) surfaces, nodes are placed at arbitrary radii along geocentric vectors defined from the center of the Earth through the spherical tessellation vertices (Figure 2). Three-dimensional piecewise linear interpolation between nodes and surfaces is used to determine the value of model parameters at any arbitrary location. Discontinuities (e.g. the Moho) are represented with two surfaces defined at the same location, but with differing values (such as velocity) assigned to the top-side and bottom-side sets of nodes.
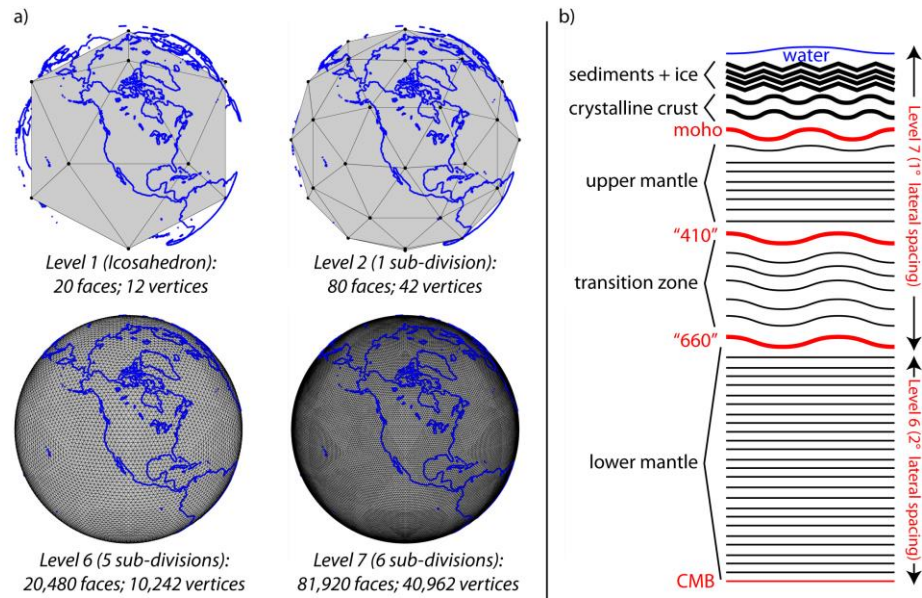

Figure 1. Summary of the LLNL-G3D model architecture. a) Selected levels of the spherical tessellation grids that define the location of nodes in the lateral extent. Nodes are placed at arbitrary radii in the direction of geocentric vectors pointing from the center of the Earth to the vertices. b) Description of the model surfaces in the LLNL-G3Dv3 P-wave model (*Simmons et al.*, 2012). Wavy lines correspond to surfaces that undulate and thick lines correspond to double surfaces needed to honor discontinuities. Flat lines correspond to surfaces that do not undulate, but note that all surfaces conform to the expected hydrostatic shape of the Earth (none of the surfaces are flat or spherical). The figure is from *Simmons et al.* (2012).
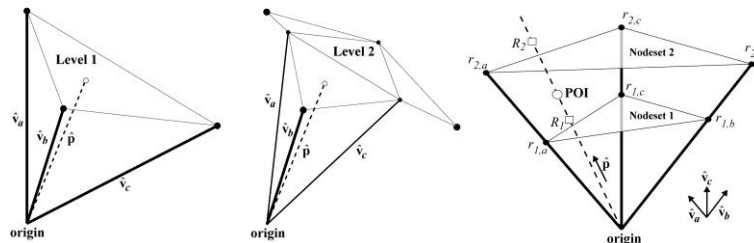
Figure 2. Model referencing and node architecture. **(a-b)** A hierarchical triangle searching algorithm is used to establish the vertices that surround a unit vector in the direction of the point-of-interest, $\hat{\mathbf{p}}$. Barycentric coordinates (triangular weights) are inherently determined at each step in the hierarchical search providing lateral interpolation weights at all tessellation levels. **(c)** Model nodes are placed at arbitrary radii in the direction of the vertices allowing for representation of irregular surfaces. To determine radial interpolation weights, radial profiles are determined along $\hat{\mathbf{p}}$ by lateral interpolation of radii for surrounding

points (R_1 interpolated from $r_{1,a-c}$ and R_2 interpolated from $r_{2,a-c}$). The distances of R_1 and R_2 from the point-of-interest (POI) provide the simple radial interpolation weights, and any model property can then be determined. The figure is from *Simmons et al.* (2011).

Three-dimensional ray tracing is performed through the complex model using a modified version of the *Zhao et al.* (1992) method that uses pseudobending within the continuous part of the media while honoring Snell's law at discontinuous interfaces (Figure 3). The ray tracing modifications include methods to overcome local travel time minima issues and the ability to find multiple paths for regional phases that arrive within some time tolerance ("multi-pathing"). For more detailed information regarding model architecture and modeling procedures see *Myers et al.* (2011), *Simmons et al.* (2011), and *Simmons et al.* (2012).

LLNL-Earth3D continues to evolve to include broader functionalities including covariance matrix handling, error estimation, surface wave dispersion calculations, and…TBD.
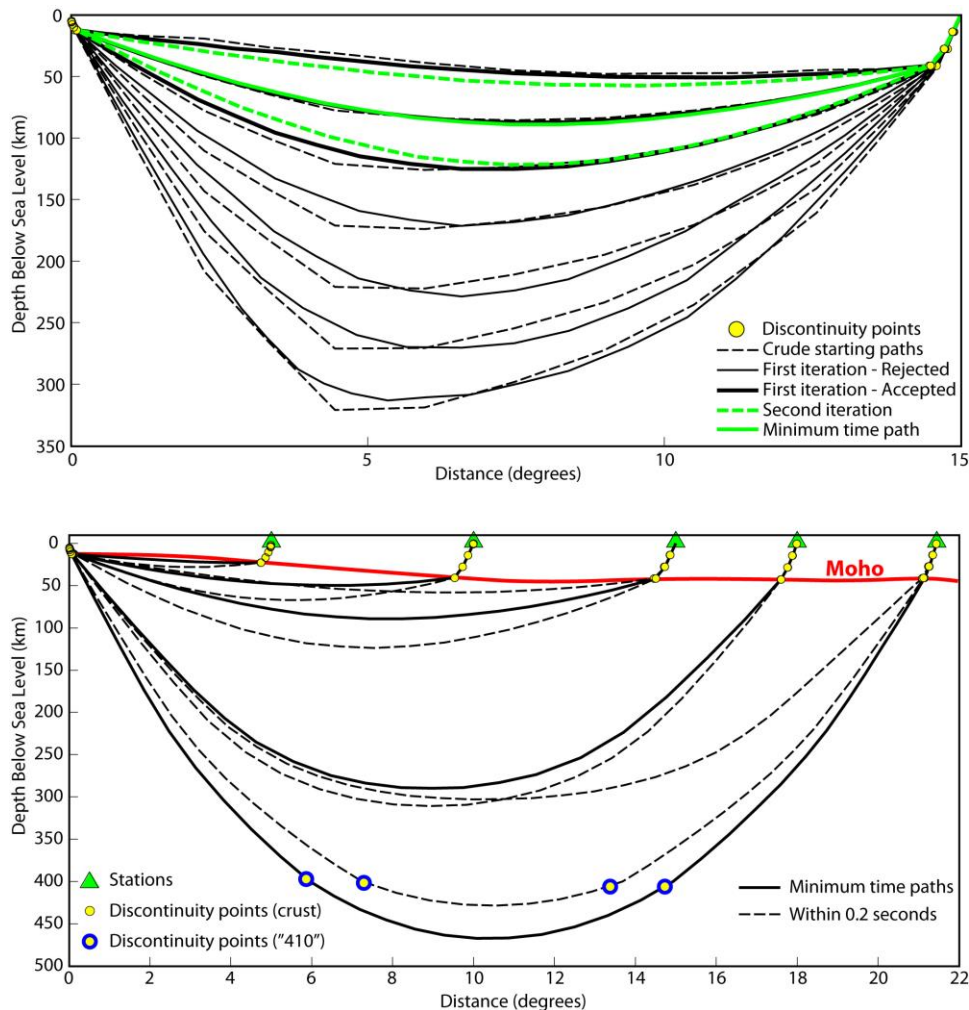
Figure 3. Three-dimensional ray tracing procedure adapted from Zhao et al. [1992] and seismic sensitivity definitions. (top) Several trial ray paths are tested to seek out the global minimum travel time. We perform a limited number of pseudo-bending [*Um and Thurber* 1987] and piercing point adjustment iterations on a set of simple starting paths (black dashed lines). Ray paths that provide travel times within some time tolerance (black solid lines) are further refined through bending and piercing point adjustments until only slight improvement (reduction) in travel times is observed (green dashed and solid lines). The minimum

3

time path can then be selected from the set (green solid line). (bottom) Example 3-D ray paths calculated through the global velocity model constructed in *Simmons et al*. (2011).

1.2 References:

Myers, S.C., G. Johannesson, and N.A. Simmons (2011). Global-scale P-wave tomography optimized for prediction of teleseismic and regional travel times for Middle East events: 1. Data set Development, *J. Geophys. Res*., **116**, B04304, doi:10.1029/2010JB007967.

Ritzwoller, M. et al. "Research Products from CU-Boulder" http://ciei.colorado.edu/Products/. Last accessed July 26, 2018.

Simmons, N.A, S.C. Myers, G. Johannesson (2011). Global-scale P-wave tomography optimized for prediction of teleseismic and regional travel times for Middle East events: 1. Tomographic inversion. *J. Geophys. Res*., **116**, B04305, doi:10.1029/2010JB007969.

Simmons, N.A, S.C. Myers, G. Johannesson, and E. Matzel (2012). LLNL-G3Dv3: Global P wave tomography model for improved regional and teleseismic travel time prediction. *J. Geophys. Res*., **117**, B10302, doi:10.1029/2012JB009525.

Simmons, N. A., S. C. Myers, G. Johannesson, E. Matzel, and S. P. Grand (2015). Evidence for long-lived subduction of an ancient tectonic plate beneath the southern Indian Ocean, *Geophys. Res. Lett*., **42**, 9270-9278, doi:10.1002/2015GL066237.

Thomsen, L. (1986). Weak elastic anisotropy. *Geophysics*, **51**(10), 1954-1966.

Um, J. and C. Thurber (1987). A fast algorithm for two-point seismic ray tracing, *Bull. Seis. Soc. Amer*., **77**, 972-986.

Zhao, D., A. Hasegawa, and S. Horiuchi (1992). Tomographic imaging of P and S wave velocity structure beneath northeastern Japan, *J. Geophys. Res*., **97**, 19,909–19,928, doi:10.1029/92JB00603.

2. Getting Started

The LLNL-Earth3D code is written in Java and will work on any system with Java version 1.6.0 or higher. LLNL-Earth3D has been tested at LLNL on Windows, Linux, OS-X, and SUN systems. The LLNL-Earth3D .jar file may be executed by:

> java -jar LLNL-Earth3D.x.x.jar

where "x.x" is a generic version number. If memory allocations for your platform are insufficient, you may need to execute by adding flags:

> java -jar -ms1500m -mx1500m LLNL-Earth3D.x.x.jar

Executing the .jar without any arguments will create a user prompt:

> E3D (input command):

After which you will enter command line options, described below. The program will continue to take user input until the user types the command "exit" or "quit", e.g.:

> E3D (input command): quit

Alternatively, a single command can be run by including the arguments in the command line. For example:

> java -jar LLNL-Earth3D.x.x.jar -getlayerinfo *earthmodel*

In the above command, the major operation is to list information about the layers in in the binary earth model, which may be a complete path pointing to a binary formatted earth model file. The code will execute this command, then exit LLNL-Earth3D. This option is used to facilitate LLNL-Earth3D being called by external programs.

All major operations are executed with switches (such as -getlayerinfo shown above) immediately following the .jar file name or at the beginning of the command prompt, depending on your execution style.

NOTE: *Working from the prompt is more efficient if multiple operations are to be performed using a single earth model since the command line version requires loading a model into memory each time. Alternatively, a list of arguments from a text file could be passed simultaneously with the* -arglist *switch.*

To get help regarding usages and the utilities available in the local .jar version, simply type -help at the E3D user prompt or, if LLNL-Earth3D is not initiated, type:

> java -jar LLNL-Earth3D.x.x.jar -help

3. Ray tracing

Basic usage (output travel times only)

LLNL-Earth3D was originally designed to compute 3-D ray paths and travel times for many source-receiver pairs simultaneously, and as efficient as possible. Therefore, a plain text file containing rows with event locations, station locations, and phases is passed to the code. The event-station-phase rows are then distributed using multi-threading techniques.

To ray trace with default options, the general command is:

java -jar LLNL-Earth3D.x.x.jar -raytrace   *earthmodel   events_stations_phases*

or, from the command prompt:

E3D (input command): -raytrace   *earthmodel   events_stations_phases*

The variables are filenames pointing to the binary earth model (*earthmodel*) and the event-station-phase list (*events_stations_phases*). The event-station-phase list can have as many rows (entries) as desired and each row is formatted as follows:

[*event_lat   event_lon   event_depth   station_lat   station_lon   station_elevation   phase   extra...*]

The *event_depth* and *station_elevation* are relative to a reference surface defined in the model (mean sea level for LLNL-G3Dv3 and other LLNL models). *phase* is a phase name such as P or Pn, etc. *extra* can be any additional columns of information the user wishes to include.

IMPORTANT NOTE FOR ANISOTROPIC MODELS ONLY: **Models such as SPiRaL may have additional parameters describing Vertical Transverse Isotropy (VTI), including "delta", "epsilon", and "gamma" defined in Thomsen (1986) for weak anisotropy. To distinguish between horizontally polarized S-waves (Sh waves) and vertically polarized S-waves (Sv waves), simply add and "h" or "v" to the phase names in the** *events_stations_phases* **file. For example, instead of** *S* **denoting a direct S-wave, use** *Sv* **or** *Sh* **for the phase name. Similarly,** *Sn* **may be** *Snv* **or** *Snh***. If the "h" or "v" are excluded from the phase name, the ray tracer will assume vertical polarity. Core phases such as SKS do not need a designation and are always assumed to be Sv in the mantle.**

The default output will be a file named *events_stations_phases* with the added extension ".TT" which stands for <u>T</u>ravel <u>T</u>imes. The .TT file will contain all of the information in each row of the input file plus additional columns of output like the following:

[*...input* EARTH3D *traveltime   water_corr_ev   water_corr_st   tt_uncert   arc_distance   azimuth   uniqueID*]

Each line of output will have the word "EARTH3D" after the input columns if the code was able to ray trace the path, and the word "NULL" if it could not. The travel time (*traveltime*) follows EARTH3D and then an event-side water correction (*water_corr_ev*) and a station-side water correction (*water_corr_st*). Water corrections arise when an event or station appear to be in the water due to the resolution limitations of a given model. This happens quite often for ocean-island stations. If it is known that the event was not in the water, simply **add** the *water_corr_ev* term to the *traveltime* to get the total travel time. Likewise, if it is known that the station is in fact on solid land, add the *water_corr_st* term to the *traveltime* to get the total travel time.

For Earth3D versions 5.4.3 and later, an additional term for the travel time uncertainty (*tt_uncert*) follows

the water correction terms. By default, the uncertainty value will be zero unless a model covariance matrix is first loaded into the environment. See Section 6 in this manual for more information.

Additional outputs include the event-station arc distance in degrees (*arc_distance*), the event-to-station azimuth (*azimuth*) and a unique identifier (*uniqueID*). The *uniqueID* are integers starting with zero that correspond to the row number of the input since the row ordering provided in the input file may not be preserved. The reordering occurs due to the multi-threading process which distributes each of the rows to multiple processors that complete the tasks at differing rates.

3.1 Ray tracing input options

Seismic phases
LLNL-Earth3D will attempt to compute travel times for a variety of phases. The user simply must identify the desired phase in each line of the *events_stations_phases* file. To get a current list of available seismic phases along with descriptions, type -listphases at the command prompt or:

>   java -jar LLNL-Earth3D.x.x.jar -listphases

Note: *While some phases are relatively simple and have been tested more thoroughly than others (e.g. first arriving P and S waves), some other phases are more complex and may be unrealistic in extreme model regions and with difficult event-receiver configurations.*

Travel time tolerances
For some seismic phases, the ray tracing algorithm begins with a ***single*** crude estimate of the path, then refines it according the pseudobending/Snell's law technique to find the minimum-time path. However, some phases (usually regional) require ***several*** initial paths to test and refine. For these more complex phases, the set of initial paths are partially optimized and down-selected to a smaller set of paths based on some initial time tolerance (*tolerance1*). The remaining paths are further refined and down-selected a final time based on a secondary time tolerance (*tolerance2*). See Figure 3 for a visual description. If the second tolerance (*tolerance2*) is greater than zero, it is possible that multiple paths are found to be within the given tolerance of the minimum time. In these cases, multiple paths are returned to the .TT file and path files if requested.

To adjust these tolerances, the user may put two numbers representing the two tolerances (in seconds) on the input line **immediately after** the event-station file:

>   E3D (input command): -raytrace *earthmodel   events_stations_phases tolerance1   tolerance2*

The default tolerances are 2 and 0 seconds, respectively. Increasing the first tolerance (*tolerance1*) will allow more rays to be passed to the second phase for further optimization. Increasing the second tolerance (*tolerance2*) above zero will possibly result in multiple paths and travel times for some phases, whereas a second tolerance equal to 0 will return the minimum-time path.

Multi-threading
The default behavior when ray tracing is seek out the number of available processors (Nproc) on the working platform and create Nproc+2 threads since this is found to be most efficient. This default can be changed by adding the –nthreads switch followed by the number of threads desired:

> E3D (input command): -raytrace *earthmodel events_stations_phases tolerance1 tolerance2* -nthreads *N*

where *N* is the number of threads. Note that the tolerance values do not need to be specified if the default values are satisfactory:

> E3D (input command): -raytrace *earthmodel events_stations_phases* -nthreads *N*

3.2 Ray tracing output options

Screen output
Simply include the switch -verbose or -v to output travel time results to the screen.

Output ray paths
When ray tracing, LLNL-Earth3D always outputs a travel time file (*.TT) by default. In addition, there are options for outputting ray paths. To output the full ray paths to files, add the switches -p or -paths after the ray tracing command:

> E3D (input command): -raytrace *earthmodel events_stations_phases* -paths

Note that multiple switches (such as -nthreads *N*) can be included as well. The above command will output an ASCII file for each row of the *events_stations_phases* unless a particular path calculation was unsuccessful for any reason.

Each of the path file names will begin with input file name (*events_stations_phases*) with the added extension that includes the unique identifier (row number from the input file beginning with 0) followed by ".ascii.path". For example, if the input file name is "myRays.txt", the first output path file name will be "myRays.txt.0.ascii.path" and the 2nd file name will be "myRays.txt.1.ascii.path", etc.

Each of the path files will contain a header containing the output travel time information, the number of paths computed, the number of points in each path, and the path points. Each point along the path (each row after the header) will have the following format:

> [*point_lat   point_lon point_radius point_distance point_depth*]

where *point_distance* is the distance from the event origin to each point along the path (in km). If the actual coordinates of the paths are not needed, another option is to output only the *point_distance* and *point_radius* using the -distradius or -dr flag:

E3D (input command): -raytrace *earthmodel events_stations_phases* -distradius

Similar to the -paths output, each distradius file will have a header followed by points that are represented by only 2 columns in this case ([*point_distance   point_radius*]). Filenames will have indexes just like the .path files, but end with ".distradius" as an extension.

3.3 Examples

Example 1a: Travel times

In this example, travel times for a small suite of P-wave phases will be computed using the LLNL-G3Dv3 model from *Simmons et al.* (2012).

Suppose we have created a text file named "myRays.txt" which contains:

```
37.68   -121.77    4.5   33.61   -116.46  1.28  Pn      user data
37.68   -121.77    4.5   57.78   -152.58  0.15  PcP     extra info
37.68   -121.77    4.5  -17.74    178.05  0.801 P       event1
33.90   -117.05   12.1   33.61   -116.46  1.28  Pg      event2
-29.90    25.13   35.7   33.61   -116.46  1.28  PKPdf   ev3
```

The first 7 columns contain the required elements including earthquake latitude/longitude and depth (first 3 elements), the required station latitude/longitude and elevation (next 3 elements), and the seismic phase. The user may add additional columns that will not be used for ray tracing.

Here is an example -raytrace command to find the minimum-time paths:

java -jar Earth3D.5.3.jar -raytrace LLNL-G3Dv3.e3d.binary G:\PROGRAMS\EARTH3D\myRays.txt -verbose

Since we are not running from the prompt, the model (which also contains the hierarchical tessellation grids and pre-computed data used for ray tracing) will be loaded. The loading process requires some wait time and you will see:

```
Calculating Cross Products and Arc Distances...
Finding Triangle Centroids...
…
1: Water (top) added
reading layer[0] 1: Water (top)
2: Water (bottom) added
reading layer[1] 2: Water (bottom)
3: Sediment 1 (top) added
reading layer[2] 3: Sediment 1 (top)
4: Sediment 1 (bottom) added
…
Inner Core (6371km) added
reading layer[94] Inner Core (6371km)
completed earthmodel file read
```

Once the model is loaded, the lines of the "myRays.txt" will be read and ray tracing will commence. If the -verbose flag is included, the travel time output will be displayed on the screen. In any case, a file named "myRays.txt.TT" will now exist in the current working directory. The .TT file will have the contents:

```
Earth3D.5.3  -raytrace  LLNL-G3Dv3.e3d.binary  G:\PROGRAMS\EARTH3D\myRays.txt  2.0  0.0 -v
   33.9000 -117.0500 12.1000  33.6100 -116.4600 1.2800  Pg   event2   EARTH3D  10.756   0.000  0.000   0.0 0.57034    120.31098  3
   37.6800 -121.7700  4.5000 -17.7400  178.0500 0.8010  P    event1   EARTH3D  726.094  0.000 -0.886  0.0 78.94891    237.40205  2
   37.6800 -121.7700  4.5000  33.6100 -116.4600 1.2800  Pn   user data  EARTH3D   87.598   0.000  0.000 0.0 5.93075    131.63000  0
   37.6800 -121.7700  4.5000  57.7800 -152.5800 0.1500  PcP extra info  EARTH3D   547.156  0.000 -0.191 0.0 28.47414    324.86188  1
  -29.9000   25.1300 35.7000  33.6100 -116.4600 1.2800  PKPdf  ev3  EARTH3D  1177.306  0.000  0.000   0.0 147.25546    286.55142  4
```

The top line of the .TT file will have the ray tracing command line inputs that were entered (default travel time tolerances 2 and 0 will appear if none were entered). Each row will contain the original rows of the input file along with the LLNL-Earth3D results after the "EARTH3D" text. The elements of output include the travel time, event- and station-side water corrections, arc distance, azimuth, and unique identifiers (last column). See the beginning of the Ray tracing section for more description. In this case, the first row of the input was the Pn phase, but output is written on the 3rd data row due to the multi-threading. If desired, the rows can be resorted to the original order using the unique identifier.

Example 1b: Minimum-time path output
The previous example did not save the actual ray paths. This example will be an extension of the previous example and demonstrate how to output ray paths to ASCII files. We will work from the command prompt to avoid loading in the model again:

E3D (input command): -raytrace   LLNL-G3Dv3.e3d.binary   G:\PROGRAMS\EARTH3D\myRays.txt -paths

Note that we have added the -paths flag. This will generate 5 files, 1 path file for each row in the input file. The path file corresponding with the first row of the input file will be named "myRays.txt.0.ascii.path". The contents of this file will look like:

```
>
>      37.6800  -121.7700  4.5000  33.6100   -116.4600 1.2800  Pn  user data EARTH3D  87.598 0.000 0.000 0.0 5.93075 131.63000  0
> npaths: 1    thispath: 1
> 221
     37.6800     -121.7700     6365.6574        0.0000        4.5000
     37.6757     -121.7642     6365.0248        0.7009        5.1342
     37.6713     -121.7585     6364.3923        1.4020        5.7683
     37.6670     -121.7527     6363.7598        2.1033        6.4023
     37.6626     -121.7469     6363.1274        2.8046        7.0362
     37.6583     -121.7412     6362.4951        3.5061        7.6701
     …omitted…
     33.6104     -116.4604     6371.8688      659.4132       -0.2792
     33.6102     -116.4602     6371.9291      659.4410       -0.3394
     33.6100     -116.4600     6371.9894      659.4689       -1.2800
```

The first character of each header line is ">" so that all path files can be concatenated together and plotted with one GMT psxy command. The travel time output is included in the header along with the number of ray paths that were found. There are 221 points in this path and the columns include latitudes, longitudes, radii, distances along the path, and depth.

Example 1c: Multi-paths
The previous example only output the minimum-time path for each row of the event-station-phase input file. This example will be an extension of the previous example and demonstrate how to output multiple ray paths with travel times within some tolerance of the minimum-time path. Working from the prompt, type:

  E3D (input command): -raytrace   LLNL-G3Dv3.e3d.binary   G:\PROGRAMS\EARTH3D\myRays.txt 4 0.5 -distradius

In the above command, we have changed the first travel time tolerance to 4 seconds and the second time tolerance to 0.5 seconds. In this case, ray paths that arrive within 0.5 seconds of the minimum-time path found will be returned for some local and regional seismic phases such as Pg and Pn. The file named "myRays.txt.TT" will be replaced with:

```
Earth3D.5.3  -raytrace  LLNL-G3Dv3.e3d.binary  G:\PROGRAMS\EARTH3D\myRays.txt  4.0  0.5 -dr
   33.9000 -117.0500  12.1000  33.6100  -116.4600  1.2800  Pg  event2  EARTH3D  10.800  0.000  0.000  0.0 0.57034   120.31098   3
   33.9000 -117.0500  12.1000  33.6100  -116.4600  1.2800  Pg  event2  EARTH3D  10.817  0.000  0.000  0.0 0.57034   120.31098   3
   33.9000 -117.0500  12.1000  33.6100  -116.4600  1.2800  Pg  event2  EARTH3D  10.756  0.000  0.000  0.0 0.57034   120.31098   3
   33.9000 -117.0500  12.1000  33.6100  -116.4600  1.2800  Pg  event2  EARTH3D  10.819  0.000  0.000  0.0 0.57034   120.31098   3
   37.6800 -121.7700   4.5000  -17.7400  178.0500  0.8010   P  event1  EARTH3D  726.094  0.000 -0.886  0.0 78.9489  237.40205   2
   37.6800 -121.7700   4.5000  57.7800 -152.5800  0.1500 PcP extrainfo   EARTH3D  547.156 0.000 -0.230 0.0 28.47414 324.86188   1
   37.6800 -121.7700   4.5000  33.6100  -116.4600  1.2800  Pn  userdata  EARTH3D  87.598 0.000  0.000  0.0  5.93075  131.63000   0
   37.6800 -121.7700   4.5000  33.6100  -116.4600  1.2800  Pn  user data  EARTH3D  87.633 0.000  0.000 0.0  5.93075  131.63000   0
   37.6800 -121.7700   4.5000  33.6100  -116.4600  1.2800  Pn  user data  EARTH3D  87.750 0.000  0.000 0.0  5.93075  131.63000   0
   37.6800 -121.7700   4.5000  33.6100  -116.4600  1.2800  Pn  user data  EARTH3D  87.951 0.000  0.000 0.0  5.93075  131.63000   0
  -29.9000   25.1300  35.7000  33.6100  -116.4600  1.2800 PKPdf  ev3  EARTH3D  1177.306 0.000  0.000 0.0 147.25546  286.55142   4
```

Note that the Pg and Pn phases have multiple entries with slightly different travel times. These multiple entries possibly indicate "true" mutli-pathing, but *there is no guarantee that the multi-paths are significantly different*. In other words, multiple starting paths could have converged to essentially the same path.

Rather than requesting paths with the -paths flag, we requested simplified path outputs with the -distradius command (which may be shortened to "-dr"). Similar to the full path output, this will generate 5 files (1 for each event-station-phase entry). The first file will be named "myRays.txt.0.distradius" and will look like:

```
>
>     37.6800  -121.7700 4.5000  33.6100  -116.4600   1.2800 Pn   userdata  EARTH3D 87.598 0.000  0.000  0.0 5.93075  131.63000
0
> npaths: 4    thispath: 1
> 221
     0.0000      6365.6574
     0.7009      6365.0248
     1.4020      6364.3923
     2.1033      6363.7598
…omitted…
   659.3853      6371.8085
   659.4132      6371.8688
   659.4410      6371.9291
   659.4689      6371.9894
>
>
>     37.6800  -121.7700 4.5000  33.6100  -116.4600   1.2800 Pn   user data EARTH3D  87.633  0.000  0.000 0.0 5.93075   131.63000
0
> npaths: 4    thispath: 2
> 221
     0.0000      6365.6574
     0.7102      6365.0248
     1.4206      6364.3922
     2.1310      6363.7597
…omitted…
```

In this example, the Pn phase produced 4 distinct paths, all of which are included in the single file. The path points are now represented only by two columns containing the distances along the path from the event (in km) and radii of the points.

4. Querying a model

LLNL-Earth3D has a number of utilities to examine or extract properties from a given LLNL-G3D formatted binary model. These include tools for general model queries, extracting profiles, extracting layer information and maps.

4.1 General queries

Model layer information
Basic information about a particular model layer structure can be output to the screen with the command:

> E3D (input command): **-**getlayerinfo *earthmodel*

Here is an example for the case where the LLNL-Earth3D prompt has not been established and the full model path is included:

> java -jar LLNL-Earth3D.5.3.jar **-**getlayerinfo   G:\PROGRAMS\EARTH3D\MODELS\LLNL-G3Dv3.e3d.binary

Note that the full model path is not needed if it resides in the working directory. Also, if the *earthmodel* of interest is already loaded into memory, simply execute with "-getlayerinfo" without the model name. Otherwise, the model will be loaded into memory. Example output:

```
Layer[0],   LayerName= '1: Water (top)',        Max.Tess.Level= 6,   Mean Radius= 6371.2212,      Properties: Vp,
Layer[1],   LayerName= '2: Water (bottom)',     Max.Tess.Level= 6,   Mean Radius= 6368.6106,      Properties: Vp,
Layer[2],   LayerName= '3: Sediment 1 (top)',   Max.Tess.Level= 6,   Mean Radius= 6368.6107,      Properties: Vp,
Layer[3],   LayerName= '4: Sediment 1 (bottom)', Max.Tess.Level= 6,  Mean Radius= 6368.3758,      Properties: Vp,
Layer[4],   LayerName= '5: Sediment 2 (top)',   Max.Tess.Level= 6,   Mean Radius= 6368.3758,      Properties: Vp,
Layer[5],   LayerName= '6: Sediment 2 (bottom)', Max.Tess.Level= 6,  Mean Radius= 6367.8412,      Properties: Vp,
    …omitted…
Layer[90], LayerName= 'Inner Core (5971km)',    Max.Tess.Level= 4,   Mean Radius=  400.0002,      Properties: Vp,
Layer[91], LayerName= 'Inner Core (6071km)',    Max.Tess.Level= 4,   Mean Radius=  299.9996,      Properties: Vp,
Layer[92], LayerName= 'Inner Core (6171km)',    Max.Tess.Level= 4,   Mean Radius=  199.9997,      Properties: Vp,
Layer[93], LayerName= 'Inner Core (6271km)',    Max.Tess.Level= 4,   Mean Radius=   99.9999,      Properties: Vp,
Layer[94], LayerName= 'Inner Core (6371km)',    Max.Tess.Level= 4,   Mean Radius=    0.0000,      Properties: Vp,
```

Tessellation node locations
The latitudes and longitudes of the spherical tessellation grids may be output to a file with the command:

> E3D (input command): **-**writelatlons *maxlevel*

where *maxlevel* is an integer of the maximum tessellation level. Note that if a model containing a tessellation grid is not already loaded into memory, the above command will need to construct a spherical tessellation grid. For example, the LLNL-G3Dv3.e3d.binary model is defined up to level 6, which consists of 40,962 vertices. To get the latitude and longitude locations of each of nodes, type:

> E3D (input command): **-**writelatlons 6

which will create a file named "SphericalTessellationCoordinates.UpTo.Level.6.txt" with the contents that look like the following:

```
Vertex    Latitude        Longitude
0         90.000000        0.000000
1         26.719301       72.000000
2         26.719301        0.000000
3         26.719301      -72.000000
```

```
4          26.719301          144.000000
5          26.719301         -144.000000
6         -26.719301           36.000000
7         -26.719301          -36.000000
8         -26.719301          108.000000
9         -26.719301          180.000000
10        -26.719301         -108.000000
11        -90.000000            0.000000
…omitted…
40957       -50.418391        -144.925818
40958       -49.400876        -144.000000
40959       -52.462817        -143.032003
40960       -53.490721        -144.000000
40961       -52.462817        -144.967997
```

General property query

Any *model_property* of interest (e.g. Vp, Vs, etc.) can be extracted using the -modelquery switch and passing a file containing a set of points to the code:

E3D (input command): -modelquery *earthmodel   model_property   points_file*

where *points_file* is a user-generated plain text file with 3 columns containing the points of interest. Each row of the *points_file* should be [*latitude longitude depth*]. The query will return a new text file that contains the original input with an addition column with values corresponding to the *model_property* value. The output file name will begin with the input file name with an added extension that corresponds to the property name. For example, we may have a file named "MyPoints.txt" which contains the entries:

```
30.95          40.51          10.9
40.01          30.79          2400
-29.92        -31.42          1000
```

The command

E3D (input command): -modelquery LLNL-G3Dv3.e3d.binary   Vp   MyPoints.txt

will generate a file named "MyPoints.txt.Vp" with a 4th column containing Vp values for each point like the following:

```
30.9500         40.5100          10.9000          6.2003
40.0100         30.7900        2400.0000         13.1993
-29.9200       -31.4200        1000.0000         11.4631
```

4.2 Profiles and cross sections

Extracting 1-D profiles
LLNL-Earth3D will create 1-D profiles by interpolating any model property such as Vp at any specified latitude and longitude. This is executed using the -create1dprofile command:

E3D (input command): -create1dprofile *earthmodel   model_property   latitude longitude*

The above command will output a text file named 1D_Profile_*.xy where the wildcard will consist of the model name, property and coordinates. The *.xy file will contain 2 columns which are the radii and model values.

For example, we can extract a P-wave velocity profile at latitude/longitude location of (30.5, -114.2) by typing:

    E3D (input command): -create1dprofile   LLNL-G3Dv3.e3d.binary   Vp   30.5   -114.2

This will output a file named "1D_Profile_LLNL-G3Dv3.e3d.binary_Vp_30.5_-114.2.xy" which will look like:

```
6372.7738        1.5000
6372.7679        1.5000
6372.7679        3.8117
6372.7679        3.8117
6372.7679        2.4663
6372.1503        2.4663
6372.1503        4.2999
…omitted…
1100.2086       11.0725
1000.1895       11.1054
 900.1710       11.1352
 800.1516       11.1619
 700.1323       11.1854
 600.1135       11.2058
 500.0947       11.2230
 400.0761       11.2371
 300.0565       11.2481
 200.0377       11.2559
 100.0188       11.2606
   0.0000       11.2622
```

Extracting 2-D profiles (cross sections) - default usage

LLNL-Earth3D will create 2-D profiles (cross sections) by interpolating any model property such as Vp along a great circle path between 2 specified latitude/longitude points. This is executed using the -create2dprofile command:

    E3D (input command): -create2dprofile *earthmodel   model_property   lat1   lon1   lat2   lon2*

The above command will output a text file named 2D_Profile_*.xyz where the wildcard will consist of the model name, property and coordinates. The *.xyz file will contain 3 columns containing:

    [*profile_distance   radii   model_property_value*]

where *profile_distance* is the distance (in km) from the starting point defined by *lat1* and *lon1*. The profiles will be evenly sampled in radius and in distance for compatibility with making images with GMT commands. With the default inputs, the code calculates the maximum resolution of the 3-D model and then samples the path 5 times more densely. The model is sampled at every ~5 km in radius from 6370.0 to ~3800.0 km (the crust and mantle).

For example, we can extract a P-wave velocity cross section from the latitude/longitude location of (30.5, -114.2) to (35, -116.9):

    E3D (input command): -create2dprofile   LLNL-G3Dv3.e3d.binary   Vp   30.5   -114.2   35.0   -116.9

This will output a file named "2D_Profile_LLNL-G3Dv3.e3d.binary_Vp_30.5_-114.2_35.0_-116.9.xyz" which will look like:

```
       0.0000      6370.0000        6.0949
       0.0000      6365.0097        6.0949
       0.0000      6360.0194        6.2045
       0.0000      6355.0291        6.2045
       0.0000      6350.0388        6.6562
       0.0000      6345.0485        6.6562
   …omitted…
       0.0000      3814.9709       13.4572
       0.0000      3809.9806       13.4631
       0.0000      3804.9903       13.4689
      23.3504      6370.0000        6.0980
      23.3504      6365.0097        6.0980
      23.3504      6360.0194        6.2039
      23.3504      6355.0291        6.2039
   …omitted…
     560.4085      3814.9709       13.4601
     560.4085      3809.9806       13.4668
     560.4085      3804.9903       13.4735
```

In addition, files containing the radii of each model layer along the great circle path will be output to files named:

"Layer_0_LLNL-G3Dv3.e3d.binary_30.5_-114.2_35.0_-116.9.xy"
"Layer_1_LLNL-G3Dv3.e3d.binary_30.5_-114.2_35.0_-116.9.xy"
…
"Layer_94_LLNL-G3Dv3.e3d.binary_30.5_-114.2_35.0_-116.9.xy"

Each of the above files will have 2 columns containing [*profile_distance   radii*] for plotting overlays of the undulating model surfaces if desired (possibly included in a GMT script).

Extracting 2-D profiles (cross sections) - set radial limits
The default radius range (in the example above) spans crust and mantle. If it is desired to expand or restrict the radius, simply add 2 additional terms with the minimum radius (*minrad*) and maximum radius (*maxrad*):

   E3D (input command): -create2dprofile *earthmodel   model_property   lat1 lon1 lat2 lon2 minrad maxrad*

Extracting 2-D profiles (cross sections) - set radial sample rate
The cross section output can be further refined by changing the vertical (radial) sample rate by including an additional number (*delta*) which defines the distance between points in km:

E3D (input command): -create2dprofile *earthmodel model_property   lat1 lon1 lat2 lon2 minrad maxrad delta*

Extracting 2-D profiles (cross sections) - return percent perturbations
The default values returned are returned in an absolute sense (e.g. absolute Vp, Vs, etc.). An alternative output is in terms of percent perturbation relative to the mean value in each layer. This is done by adding the flag "-dpct" to the end of the -create2dprofile command:

   E3D (input command): -create2dprofile *earthmodel model_property   lat1 lon1 lat2 lon2* -dpct

Note that values in the sedimentary and crustal layers can be very high and some values may be meaningless if there is zero thickness.

4.3 Map output

LLNL-Earth3D may be used to extract map data for a specified *model_property* using the -createmap command:

  E3D (input command): -createmap *earthmodel model_property   minlat minlon maxlat maxlon layerindex*

where *minlat/minlon* defines the lower left-hand corner of the map region, *maxlat/maxlon* defines the upper right-hand corner of the bounded region, and *layerindex* is the model layer index (beginning with index 0). ***Hint:*** Use the -getlayerinfo command to get the index number of the layer desired.

The above command will output a text file named Map_*.xyz where the wildcard will consist of the model name, property and bounding coordinates. The *.xyz file will contain 3columns which are:

       [*latitudes   longitudes model_values*].

For example, the following command creates a map data file containing absolute P-wave velocity for layer index 16 in the LLNL-G3Dv3 model:

       E3D (input command): -createmap   LLNL-G3Dv3.e3d.binary   Vp   15.0   30.0   45.0   90.0   16

This will create a file named "Map_ LLNL-G3Dv3.e3d.binary_Vp_16_15.0_30.0_45.0_90.0.xyz" which will have 3 columns with the latitudes, longitudes and Vp values:

```
        15.0000      30.0000        8.0701
        15.0000      30.2166        8.0707
        15.0000      30.4332        8.0714
   …omitted…
        15.0000      89.5668        8.0778
        15.0000      89.7834        8.0787
        15.2158      30.0000        8.0719
        15.2158      30.2166        8.0726
   …omitted…
        45.0000      89.3502        8.1448
        45.0000      89.5668        8.1401
        45.0000      89.7834        8.1376
```
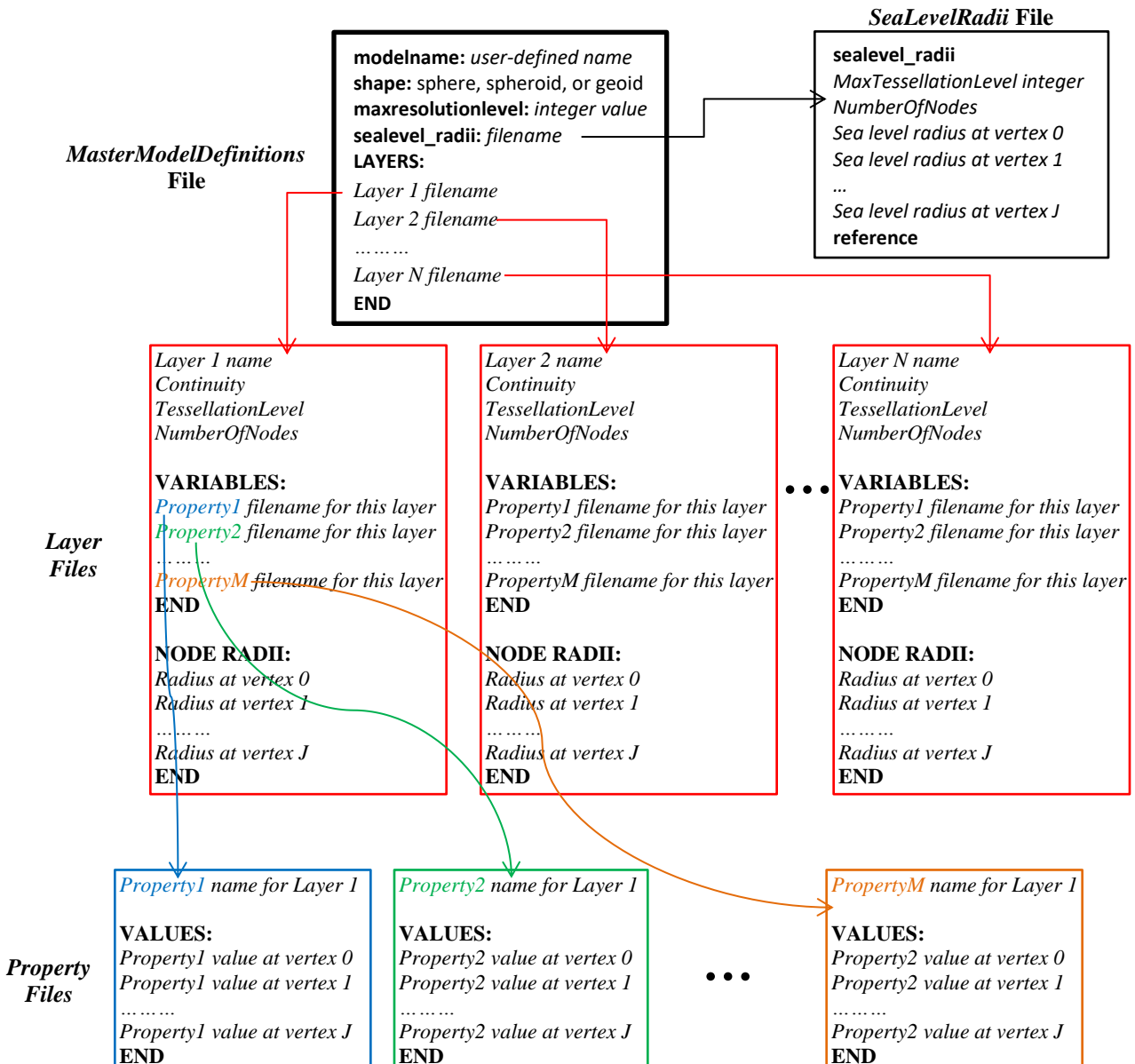
5. Model importation

LLNL-Earth3D is specifically designed to work with LLNL-G3D formatted Earth models stored in a binary stream. It is possible to import alternative Earth models and/or potentially use LLNL-Earth3D as a lookup table for user-defined variables with a quasi-spherical spatial arrangement. Since our primary focus is on structural models of the Earth, the following text specifically describes how to import an Earth model.

Hierarchical file scheme
Constructing a LLNL-G3D formatted binary file is achieved by first constructing several plain text files. The text files include i) a *MasterModelDefinitions* file with the top-level information about a model, ii) *LayerFiles* containing information about specific layers defined in the master file, iii) *PropertyFiles* with model values for a specific property (e.g. Vp) in a specific layer, and iv) *SeaLevelRadii* file containing the radii corresponding to sea level at nodes defined by the spherical tessellation grids. Below is a schematic of the file structure:

SeaLevelRadii **File**

sealevel_radii
MaxTessellationLevel integer
NumberOfNodes
Sea level radius at vertex 0
Sea level radius at vertex 1
...
Sea level radius at vertex J
reference

modelname: *user-defined name*
shape: sphere, spheroid, or geoid
maxresolutionlevel: *integer value*
sealevel_radii: *filename*
LAYERS:
Layer 1 filename
Layer 2 filename
....
Layer N filename
END

MasterModelDefinitions **File**

Layer Files

Layer 1 name
Continuity
TessellationLevel
NumberOfNodes

VARIABLES:
Property1 filename for this layer
Property2 filename for this layer
....
PropertyM filename for this layer
END

NODE RADII:
Radius at vertex 0
Radius at vertex 1
....
Radius at vertex J
END

Layer 2 name
Continuity
TessellationLevel
NumberOfNodes

VARIABLES:
Property1 filename for this layer
Property2 filename for this layer
....
PropertyM filename for this layer
END

NODE RADII:
Radius at vertex 0
Radius at vertex 1
....
Radius at vertex J
END

Layer N name
Continuity
TessellationLevel
NumberOfNodes

VARIABLES:
Property1 filename for this layer
Property2 filename for this layer
....
PropertyM filename for this layer
END

NODE RADII:
Radius at vertex 0
Radius at vertex 1
....
Radius at vertex J
END

Property Files

Property1 name for Layer 1

VALUES:
Property1 value at vertex 0
Property1 value at vertex 1
....
Property1 value at vertex J
END

Property2 name for Layer 1

VALUES:
Property2 value at vertex 0
Property2 value at vertex 1
....
Property2 value at vertex J
END

PropertyM name for Layer 1

VALUES:
Property2 value at vertex 0
Property2 value at vertex 1
....
Property2 value at vertex J
END

MasterModelDefinitions file content

In the *MasterModelDefinitions* file shown in the schematic (as well as the other files), the values in **bold** font are needed verbatim and the values in *italics* are user-defined inputs. The text needed in the *MasterModelDefinitions* file is simply the name of the model to construct, the general model shape, the maximum tessellation level and file names. The model name can be anything the user chooses, but the shape of the model being created must be one of the 3 option listed in the schematic. In any case, the user must define the radii corresponding to sea level and the file name should be listed. Here is an example for the LLNL-G3Dv3 model build:

```
modelname:  LLNL-G3Dv3
shape:  geoid
maxresolution:   6
sealevel_radii:  sealevel_radii.ascii.dat
LAYERS:
LAYER.1.1.ascii.dat
LAYER.1.2.ascii.dat
LAYER.1.3.ascii.dat
LAYER.1.4.ascii.dat
LAYER.1.5.ascii.dat
…omitted…
LAYER.3.35.ascii.dat
LAYER.3.36.ascii.dat
LAYER.3.37.ascii.dat
LAYER.3.38.ascii.dat
END
```

The layer file names must be in order from the top down (surface to the bottom of the model) and the radii of any defined layer may not be greater than the radii of any layer listed above it (no crossing surfaces). However, the layers may "pinch out" and have the same radii as the layer above. Note that the layer file names may be anything the user chooses. The layer names in the example above are our personal nomenclature. Additionally, file system paths may be included if the files reside in different locations or sub-directories (e.g. G:\MyModel\Layer1.1\Layer1.1.ascii.dat).

Node ordering

Each model property (e.g. Vp, Vs, surface radii) must be defined at a specific set of latitude-longitude points determined by the spherical tessellation recursion process. The model values listed in the *SeaLevelRadii* file and the *PropertyFiles* must be interpolated by the user to each of these points and written to the text files in a precise sequence. To get a list of the sequence of hard-wired latitude-longitude points, type:

E3D (input command): -writelatlons *maxlevel*

This will create a text file with the latitude-longitude locations of the tessellation node points for a grid defined at the requested maximum tessellation level. The number of node points for a particular tessellation level can also be determined from the output.

Note that the sequence of the latitude-longitude points never changes, regardless of the tessellation level requested. Instead, points belonging to higher tessellation level grids are appended to the bottom of the list of points from the lower level grids.

SeaLevelRadii file

The *SeaLevelRadii* file is a user-defined file containing the radii corresponding to a reference whereby depth is measured (the reference radius is often sea level). The text file must contain a header line, followed by the maximum tessellation level, and the number of nodes. The radius values are then listed in the sequence of latitude-longitude points defined by the spherical tessellation recursion process.

Here is an example for the LLNL-G3Dv3 build where the sealevel_radii file was named "sealevel_radii.ascii.dat" according the *MasterModelDefinitions* file:

```
sealevel_radii
6
40962
6356.765920
6373.798596
…omitted…
6364.343637
6364.713855
reference
```

Layer files

The *LayerFiles* contain information about each of the layers in the model. The header includes a user-defined layer name and a *Continuity* descriptor that identifies whether the layer (or surface) is continuous or discontinuous. The next 2 header entries are the tessellation level and number of points, similar to the entries in the *SeaLevelRadii* file. ***Note that the resolution (or tessellation) level may differ for each of the layers.***

There are 3 possible values for *Continuity*: i) "continuous" if the surface is not at a discontinuity, ii) "top" if the surface is the top-side of discontinuity, or iii) "bottom" if the surface is the underside of a discontinuity. An example of a "top" is a layer defined at the Moho and with model properties corresponding to the bottom of the crust (possible model property might be Vp with values near 6.5 km/s). An example of a "bottom" is a layer defined at the Moho and with model properties corresponding to the top of the mantle (possible model property might be Vp with values near 8.0 km/s). Note that to properly form a discontinuity, the radii of each of the "top" and "bottom" surfaces should be exactly the same.

Note: *In order to specify some particular phases for 3-D ray tracing, certain layer names (designated in the first line of header)* **must** *contain specific strings that identify them as special layers. The strings can be upper or lower case, and may be anywhere in the name of the layer. The current list of special layer names currently include: "water", "upper crust", "lower crust", "moho", "transition zone", "CMB", and "ICB". These special strings can be in the layer name of a discontinuity top or bottom (or both). Currently there is no specific designation for the "410" or "660", just identify all layers in the transition zone with "transition zone" in the name.*

Following the header information is a list of filenames containing the model properties (or variables) associated with a given layer. The user may include any properties (Vp, Vs, Q, etc.), but note that it is expected that each layer have the same collection of associated properties.

Following the list of model property file names is a list of node radii values for each of the vertices. The radii are defined from the center of the Earth to the latitude-longitude points defined in the -writelatlons output file described above.

Below, we will show some example layer file contents for the LLNL-G3Dv3 P-wave model where we are also including some S-wave velocities for demonstration. The LLNL-G3Dv3 model is defined at tessellation level 6 (~1° node spacing) in the crust and upper mantle. In the lower mantle however, the maximum tessellation level is 5 (~2° node spacing).

Here is an example layer file contents for the first layer in the model which defines the top of the water layer:

```
1: Water (top)
bottom
6
40962

VARIABLES:
LAYER.1.1.Vp.ascii.dat
LAYER.1.1.Vs.ascii.dat
END

NODE_RADII:
6356.765920
6374.044134
6374.112587
…omitted…
6364.343637
6364.713855
END
```

In the example shown above, notice that we include the string "Water" in the name so that particular seismic phases can be ray traced (e.g. we can compute both pwP and pP). The top of the water is a discontinuity due to the air-water interface and it thus a "bottom" since it is the underside of that discontinuity. The file names for the variables (properties) are clearly pointing to files with Vp and Vs values, but there is no file naming convention required by the code.

The next example is a layer file for the bottom of the lower crust:

```
14: Lower Crust (bottom)
top
6
40962

VARIABLES:
LAYER.1.14.Vp.ascii.dat
LAYER.1.14.Vs.ascii.dat
END

NODE_RADII:
6343.987562
6336.362607
…omitted…
6354.162211
6354.238545
END
```

In the example above, the bottom of the lower crust is a discontinuous "top" since it is the top side of the Moho discontinuity.

The next example is the layer file contents for the layer also at the Moho, but representing the top of the upper mantle:

```
15: Top of mantle (Moho)
bottom
6
40962

VARIABLES:
LAYER.1.15.Vp.ascii.dat
LAYER.1.15.Vs.ascii.dat
END

NODE_RADII:
6343.987562
6336.362607
…omitted…
6354.162211
6354.238545
END
```

Notice that in the above example, we include the string "Moho" in the layer name so that this surface may be identified when computing certain seismic phases such as Pn and PmP. Also notice that the radii are exactly the same as the previous example which is the layer file for the bottom of the lower crust.

The next example is the layer file contents for a lower mantle surface at approximately 971 km depth:

```
4: Lower Mantle (971km)
continuous
5
10242

VARIABLES:
LAYER.2.4.Vp.ascii.dat
LAYER.2.4.Vs.ascii.dat
END

NODE_RADII:
5389.017000
5402.192897
…omitted…
5394.890116
5395.456085
END
```

In the example above, the layer is not a discontinuity and is therefore marked "continuous". In addition, this layer is defined at a lower resolution level than the previous example (tessellation level 5 rather than 6). The layer name contains a "4" since this is the 4th lower mantle layer. This is our personal layer naming convention and thus has no particular significance to the user.

Property (or Variable) files

The *PropertyFiles* are simple text files containing the name of the properties and actual model values such as Vp. The top line of the file should be the name of the property which is user-defined. It should

be noted that, if the user wishes to execute ray tracing utilities, Vp and/or Vs must be one of the properties and the name given to these properties must be exactly "Vp" and/or "Vs".

Here is an example property for the LLNL-G3Dv3 mantle P-wave velocity at the Moho:

```
Vp

VALUES:
8.146360
8.198282
8.119254
8.172410
…omitted…
8.035326
8.017037
8.016469
8.020075
END
```

Again, the model property values are listed in the precise sequence of the latitude-longitude points listed by the -writelatlons command.

> **IMPORTANT NOTE FOR ANISOTROPIC MODELS ONLY:** **Models such as SPiRaL may have additional parameters describing Vertical Transverse Isotropy (VTI), including "delta", "epsilon", and "gamma" defined in Thomsen (1986) for weak anisotropy. To construct and import similar models, these 3 variables must be included in addition to Vp and Vs and exactly labeled as "delta", "epsilon" and "gamma". In this case, Vp and Vs will describe compressional and shear speeds for vertically traveling waves (0 degrees incidence).**

Execution

Once the collection of text files are completed, the model importation process may be initiated using the LLNL-Earth3D -importmodel command as follows:

> E3D (input command): -importmodel *MasterModelDefinitions*

where *MasterModelDefinitions* is a user-defined filename. The process will compute a spherical tessellation grid at the highest tessellation level needed and populate a binary stream with complete model information. The code will also compute and store normal vectors to surfaces that are listed as discontinuous for the purpose of more efficient ray tracing.

If Vp and/or Vs are included as properties, velocity gradients will be computed at each point in the model and stored with the model. This pre-computation increases the efficiency of the ray tracing algorithms. Note that the pre-computation steps could take substantial amounts of time, which is a function of the total number of model nodes. For example, the LLNL-G3Dv3 P-wave velocity model consists of ~1.6 million points and takes ~1.5 hours to complete on a moderate PC workstation.

6. Covariance Matrices and Uncertainty

Travel time uncertainty can be estimated from a model covariance matrix, if one exists. For Earth3D versions 5.4.3 and later, the travel time output lines described in Section 3 will include a travel time uncertainty term which will default to zero. However, if a covariance model exists and is loaded into the environment prior to ray tracing, travel time uncertainty will be computed for each ray path.

Loading a covariance matrix

The argument -loadCovariance *<metadata json file>* directs Earth3D to load a covariance matrix into the working environment. The *<metadata json file>* is a JavaScript Object Notation (JSON) formatted text file with metadata that identifies several parameters needed to load and construct the covariance matrix, as well as rules for computing travel time error estimates for rays traced through the underlying Earth model.

Here are the contents of an example metadata JSON file:

```
{
   "description" : "P-wave Covariance Matrix (May 3, 2018)",
   "waveType" : "P",
   "pWaveScaling" : 1.0,
   "sWaveScaling" : 2.0,
   "minVariance" : 0.01,
   "rowIndexesFile" : "LLNL-G3D-JPS_Covariance_Matrix.i.bi",
   "columnIndexesFile" : "LLNL-G3D-JPS_Covariance_Matrix.j.bi",
   "valuesFile" : "LLNL-G3D-JPS_Covariance_Matrix.k.bi",
   "maxLevel" : [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5],
   "minModelLayer" : [0, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59],
   "maxModelLayer" : [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59]
 }
```

There are (3) files identified in the JSON file that contain the actual covariance matrix (`rowIndexesFile`, `columnIndexesFile`, `valuesFile`). These binary files contain the matrix row number integers (`rowIndexesFile`), the matrix column number integers (`columnIndexesFile`) and double values (`valuesFile`) for non-zero entries of the sparse covariance matrix.

The matrix will be loaded into the Java virtual machine (JVM) on the first call and it will stay loaded until Earth3D is shutdown. The memory requirements to load a covariance matrix can be quite large. For example, the covariance matrix for the LLNL-G3D-JPS model (from *Simmons et al.* 2015) requires up to 50 GB while loading and will utilize about 30GB of memory once loaded. Therefore, java command line flags must be used to allocate enough memory for the JVM (see the execution example below).

Execution example

The following example loads the P-wave slowness covariance model for the LLNL-G3D-JPS model (*Simmons et al.* 2015), performs 3-D ray tracing (as shown in Section 3 above), and finally computes travel time error for each ray:

> java -jar -Xms30g -Xmx50g LLNL-Earth3D.5.4.3.jar argsList.txt
>
> argsList.txt:
>   -loadcovariance LLNL-G3D-JPS_Covariance_Matrix.json

```
                    -raytrace    LLNL-G3D-JPS.e3d.binary  myRays.txt
```

Example output when loading the covariance matrix:

```
-sh-4.2$ java -jar -Xms30g -Xmx50g LLNL-Earth3D.5.4.3.jar  argsList.txt
File:   argsList.txt    true
line:   -loadcovariance  P-wave_Covariance_Matrix_Metadata.json
Covariance Matrix Metadata file: P-wave_Covariance_Matrix_Metadata.json
P-wave_Covariance_Matrix_Metadata.json
INFO: Elapsed 0.023 Starting Covariance Maxtix Reads for Datafile: ~/ LLNL-G3D-
JPS_Covariance_Matrix.k.bi
INFO: Elapsed 0.416 Submitted 265 Read Tasks.
INFO: Elapsed 17.976 Read in: 31 million elements; Read Tasks remaining: 250
<…>
INFO: Elapsed 157.443 Read in: 555 million elements; Read Tasks remaining: 0
INFO: Thread pool shut down.
INFO: Elapsed 157.450 Read in: 555 million elements with MaxColumn: 1003608
line:   -raytrace    LLNL-G3D-JPS.e3d.binary  myRays.txt
LLNL-G3D-JPS.e3d.binary myRays.txt
Calculating Cross Products and Arc Distances...
<…>
reading layer[96] Inner Core 97: 6371.0km
completed earthmodel file read
Adding P phase to 1D path segments database
Adding P phase to 1D path segments database (uses the P path segments)
26 375.580000
done in 375.788 seconds
```

In this case, the time to load the covariance matrix was 157.450 seconds and the time to perform the ray tracing for thousands of ray paths was 375.788 seconds. Remember that many ray tracing tasks could be requested to avoid reloading the covariance matrix and Earth model. The above example will produce a file called "myRays.txt.TT" with the input path information followed by the calculated travel times and path-dependent error estimates like the following:

```
41.3100 129.1000 0.0000 74.2421 144.0000 0.0000 P  EARTH3D 402.720 0.000 -0.044 0.835 33.74277   7.26734 0
41.3100 129.1000 0.0000 42.6155  72.0000 0.0000 P  EARTH3D 472.295 0.000  0.000 0.824 41.78068 291.53119 1
...
```

Where the travel times and water corrections are highlighted in green, and the travel time error (standard deviation, seconds) is highlighted in red.

7. Surface wave dispersion curves

Earth3D version 5.4.3 and later allows for calculation of surface wave dispersion curves based on defined physical properties of an Earth3D-formatted model that consists of all necessary properties including:

Vertical wave speeds:
      Vp   (vertical P-wave speed, km/s)
      Vs   (vertical S-wave speed, km/s)
Thomsen parameters for Vertical Transversely Isotropic (VTI) media (see *Thomsen* 1986):
      delta   (see Equation 17 in *Thomsen* 1986, unitless)
      epsilon (see Equation 8a in *Thomsen* 1986, unitless)
      gamma(see Equation 8b in *Thomsen* 1986, unitless)
Other parameters:
      Rho   (density, g/cm^3)
      Q   (intrinsic shear Q factor)

The dispersion calculations are performed using the "**senskernel 1.0**" software package available from the Colorado University at Boulder (CU-Boulder) available here: http://ciei.colorado.edu/Products/senskernel-1.0.tgz. The **senskernel 1.0** package is designed for Linux-based machines, but the underlying utility used for this purpose (SURF_PERTURB) is a Fortran program that must be installed. See their product for installation instructions.

Earth3D will extract model properties at requested locations around the globe, construct a model file formatted for **senskernel 1.0** programs, and execute the dispersion calculation in one step.

Dispersion at a specified point (1D)
Surface wave dispersion can be computed at a specific location(s) around the globe using the LLNL-Earth3D -dispersion1D command as follows:

      E3D (input command): -dispersion1D *earthmodel latitude longitude wavetype m1 m2 T1 T2 dT*

where *earthmodel* is the name of the Earth3D-formatted model with all of the necessary variables described above, *latitude* and *longitude* are the position coordinates, wavetype is either "R" for Rayleigh waves or "L" for Love waves, *m1* and *m2* are the minimum and maximum wave modes (use "0 0" for only the fundamental mode), *T1* and *T2* are the minimum and maximum wave periods in seconds, and *dT* is the period sampling increment.

For example, suppose we want to calculate dispersion curves for a fundamental mode (*m1*=0) and 1st mode (*m2*=1) Rayleigh wave at (30°N, 110°W) provided an *earthmodel* file named "MyModel.e3d.binary". The execution from the command prompt could be:

      E3D (input command): -dispersion1D MyModel.e3d.binary *30.0 -110.0 R 0 1 10 100 10*

where we have indicated that we want velocities between the periods of 10-100s at 10s increments. The above command will result in a 1D model file suitable for the **senskernel 1.0** software named "CUBinput.model.30.0_-110.0.CUBinputModel.txt" and files with the results.

A text file named "MyModel.e3d.binary_30.0_-110.0_dispersion.R.grv" will contain the <u>group</u> velocities at each discrete period, and for each specified mode like the following:

```
PERIOD                    GROUP VELOCITY
10.000000000000000        3.1558685068399428
20.000000000000000        3.1735029469690348
30.000000000000000        3.5262948796528018
40.000000000000000        3.6823579600186389
50.000000000000000        3.7211622160793518        (for mode=0)
60.000000000000000        3.7119937293650893
70.000000000000000        3.6828438101658207
80.000000000000000        3.6464660993200053
90.000000000000000        3.6095045717809526
100.00000000000000        3.5755558012102173


10.000000000000000        4.1130722233747496
20.000000000000000        4.0413447168769343
30.000000000000000        3.9730522452177062
40.000000000000000        4.0045297498865198
50.000000000000000        4.1000319418360887        (for mode=1)
60.000000000000000        4.1799839696879362
70.000000000000000        4.2340588144648574
80.000000000000000        4.2748212623523392
90.000000000000000        4.3132951051905390
100.00000000000000        4.3651193345339596
```

In addition, a text file named "MyModel.e3d.binary_30.0_-110.0_dispersion.R.phv" will contain the phase velocities at each discrete period, and for each specified mode like the following:

```
PERIOD                 PHASE VELOCITY (method1)   PHASE VELOCITY (method2)
10.000000000000000        3.3381117069658801         3.3381110411628554
20.000000000000000        3.5968041921047225         3.5968049691901545
30.000000000000000        3.7439458444135765         3.7439465309952600
40.000000000000000        3.7896819932064316         3.7896828653416148
50.000000000000000        3.8102667615791397         3.8102676217728173
60.000000000000000        3.8280706089761960         3.8280713076414736
70.000000000000000        3.8494755573570627         3.8494764120784111
80.000000000000000        3.8762601240555377         3.8762607977855108
90.000000000000000        3.9086320075163998         3.9086325270980820
100.00000000000000        3.9462084709789984         3.9462090928680249


10.000000000000000        4.2114320222193289         4.2114325636521182
20.000000000000000        4.3176985456466079         4.3175565253548180
30.000000000000000        4.4873338464114232         4.4873347200190929
40.000000000000000        4.6823521156359327         4.6823526785480087
50.000000000000000        4.8699110704239228         4.8699118925253639
60.000000000000000        5.0449384126991283         5.0449392662743380
70.000000000000000        5.2155274069154069         5.2155274402681968
80.000000000000000        5.3871456386500283         5.3871463237235968
90.000000000000000        5.5622363677111029         5.5622370477192504
100.00000000000000        5.7405606098356206         5.7405612654961011
```

*See the **senskernel 1.0** information pages for details on the 2 phase velocity calculation methods.*

TIP: **You can use the -argslist option to calculate dispersion at multiple points, for multiple wave types, and over multiple period ranges…without reloading the Earth model. For example, try:**

    java -jar -ms1500m -mx1500m LLNL-Earth3D.5.4.3.jar -argslist MyArgs.txt

where MyArgs.txt is a text file containing multiple requests:

    -dispersion1D MyModel.e3d.binary *30.0 -110.0 R 0 1 10 100 10*
    -dispersion1D MyModel.e3d.binary *45.0 -20.0 L 0 1 10 100 10*
    -dispersion1D MyModel.e3d.binary *17.0 -142.0 R 0 2 10 200 5*

Dispersion along a profile (2D)

Average surface wave dispersion can be computed along a great circle path between 2 points along the surface of the Earth using the LLNL-Earth3D -dispersion2D command as follows:

> E3D (input command): -dispersion2D *earthmodel lat1 lon1 lat2 lon2 dL wavetype m T1 T2 dT*

where *earthmodel* is the name of the Earth3D-formatted model with all of the necessary variables described above, (*lat1*, *lon1*) are the coordinates of the starting point, (*lat2*, *lon2*) are the coordinates of the end point, *dL* is the spatial sampling increment in arc degrees, *wavetype* is either "R" for Rayleigh waves or "L" for Love waves, *mode* is *the* mode number (use "0" for the fundamental mode), *T1* and *T2* are the minimum and maximum wave periods in seconds, and *dT* is the period sampling increment.

The command will compute dispersion curves at discrete positions along a great circle path between (*lat1*, *lon1*) and (*lat2*, *lon2*) at evenly spaced increments according to *dL*, and average the curves.

For example, suppose we want to calculate an average dispersion curve for a fundamental mode (*m*=0) Rayleigh wave travelling between (30°N, 110°W) and (50°N, 90°W) provided an *earthmodel* file named "MyModel.e3d.binary". The execution from the command prompt could be:

> E3D (input command): -dispersion2D MyModel.e3d.binary *30.0 -110.0 50.0 -90.0 1 R 0 10 100 10*

where we have indicated that we want to spatially sample the model every 1 degree and return velocities between the periods of 10-100s at 10s increments.

The above command will result in a text file named "MyModel.e3d.binary_30.0_-110.0_50.0_-90.0_dispersion.R.grv" that will contain the average <u>group</u> velocities at each discrete period along the profile like the following:

```
PERIOD                 GROUP VELOCITY
 10.000000000000000 3.0596028420661776
 20.000000000000000 3.0324205050710864
 30.000000000000000 3.2961319079669513
 40.000000000000000 3.6045844977716586
 50.00000000000000  3.7690661999999864
 60.000000000000000 3.840169208434363
 70.000000000000000 3.8643657497933144
 80.000000000000000 3.8648673907158706
 90.00000000000000  3.8530706091759175
 100.00000000000000 3.8348350608189303
```

In addition, a text file named " MyModel.e3d.binary_30.0_-110.0_50.0_-90.0_dispersion.R.phv" will contain the average <u>phase</u> velocities at each discrete period along the profile like the following:

```
PERIOD              PHASE VELOCITY    PHASE VELOCITY (method2)
 10.000000000000000 3.270428383987599 3.2704384157919506
 20.000000000000000 3.534736142713886 3.5347349431881487
 30.000000000000000 3.786173321364234 3.786173719150992
 40.00000000000000  3.916417802730909 3.9164184517653973
 50.00000000000000  3.976670295263056 3.9766710180902525
 60.000000000000000 4.011315828327944 4.011316555121768
 70.000000000000000 4.037704151280009 4.037704875374668
 80.000000000000000 4.062479460733059 4.062480220596497
 90.000000000000000 4.088353542678938 4.088354206232497
 100.00000000000000 4.116476444423049 4.116477155164966
```

*See the **senskernel 1.0** information pages for details on the 2 phase velocity calculation methods.*