SANDIA REPORT
SAND2023-05865
Printed June 2023

Sandia National Laboratories

PCalc User's Manual

Andrea Conley[1], Nathan J. Downey[1], Sanford Ballard[1], James R. Hipp[1], Patrick Hammond[1], Kathy Davenport[1], and Michael E. Begnaud[2]

[1]*Sandia National Laboratories*
[2]*Los Alamos National Laboratory*

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

Prediction Calculator (PCalc) is a software tool that computes travel-time predictions, ray path geometry, and model queries. This software has a rich set of features, including the ability to use custom 3D velocity models known as GeoTess models (see www.sandia.gov/geotess for details) to compute predictions using a variety of geometries. The PCalc software is especially useful for research related to seismic monitoring applications.

The PCalc software, User's Manual, and examples are available on the web at:

https://github.com/sandialabs/Salsa3DSoftware

This page left blank.

CONTENTS

1. Introduction 9
   1.1. Pseudo-bending in PCalc 10
   1.2. seismicBaseData 10
   1.3. Seismicity Depth Model 11
2. PCalc Installation 13
3. PCalc Tutorial and Examples 14
   3.1. PCalc Model Query Examples 16
      3.1.1. Example 1 – PCalc Model Query Using MJAR.coords.xyz File 16
      3.1.2. Example 2 – PCalc Model Query Using a Great Circle 21
      3.1.3. Example 3 – PCalc Model Query Using a Grid 26
   3.2. PCalc Travel Time Prediction Examples 29
      3.2.1. Example 4 – PCalc Travel Time Prediction Using MJAR.coords.xyz File 29
      3.2.2. Example 5 – PCalc Travel Time Prediction Using Great Circles 34
      3.2.3. Example 6 – PCalc Travel Time Prediction Using a Grid 37
      3.2.4. Example 7 – PCalc Travel Time Prediction Using a Database 40
      3.2.5. Example 8 – PCalc Raypath Generation Using Great Circles 45
   3.3. PCalc LibCorr3d Surface Generation 48
      3.3.1. Example 9 – LibCorr3d Surface Generation with a Custom Grid 48
      3.3.2. Example 10 – LibCorr3d Surface Generation with a Common Grid 55
      3.3.3. Example 11 – LibCorr3d Surface Generation with a Rotated Common Grid 59
4. Summary 65

Appendix A. PCalc Properties 67
   A.1. Setting Properties 67
   A.2. Property Descriptions 67
      A.2.1. General 67
      A.2.2. Input 68
      A.2.3. Input from File 70
      A.2.4. Input from Great Circle 71
      A.2.5. Input from Grid 73
      A.2.6. Input/Output from/to Database 74
      A.2.7. Input Depth Specification 77
      A.2.8. Output Properties 79
      A.2.9. Predictors 82
      A.2.10. LibCorr3d 86
      A.2.11. Model Queries 87
      A.2.12. Ray Path Geometry 88

Appendix B. LibCorr3d grids 89
   B.1. Custom Grid for Each Station 90
   B.2. Uniform Grid Shared Among Multiple Stations 91
   B.3. Non-Uniform Grid Shared Among Multiple Stations 91

Appendix C. PCalc SiteFiles 93
   C.1. Usage 93
   C.2. Site File Description 93
   C.3. Effects of Aborted PCalc Runs 94

LIST OF FIGURES

Figure 1. Seismicity depth model. (Top) Upper boundary based on elevation. (Bottom) Lower boundary based on historic seismicity. 12
Figure 2. PCalc Model Query Properties File pcalc_query_file.properties. 17
Figure 3. PCalc Model Query Properties File pcalc_query_greatcircle.properties. 22
Figure 4. PCalc Model Query Properties File pcalc_query_grid.properties. 27
Figure 5. PCalc Prediction Properties File pcalc_predictions_file.properties. 30
Figure 6. PCalc Prediction Properties File pcalc_predictions_greatcircle.properties. 35
Figure 7. PCalc Prediction Properties File pcalc_predictions_grid.properties. 38
Figure 8. PCalc Prediction Properties File pcalc_predictions_db.properties. 41
Figure 9. PCalc Prediction Properties File pcalc_raypaths_greatcircle.properties. 46
Figure 10. PCalc LibCorr3d Custom Grid Model Generation Properties File pcalc.properties. 49
Figure 11. PCalc LibCorr3d Common Grid Model Generation Properties File geotessbuilder.properties. 55
Figure 12. PCalc LibCorr3d Common Grid Model Generation Properties File pcalc.properties. 57
Figure 13. PCalc LibCorr3d Rotated Common Grid Model Generation Properties File geotessbuilder.properties. 60
Figure 14. PCalc LibCorr3d Rotated Common Grid Model Generation Properties File pcalc.properties. 62
Figure 15. a) A custom grid for a station in Iceland. b) A custom grid centered on the North Pole that can be shared among multiple stations when appropriate rotations are applied at run time. 90

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
PCalc	Prediction Calculator
CSS	Center for Seismic Studies
SALSA3D	Sandia-Los Alamos 3D velocity model
GeoTess	Geographical Tessellation Software
LocOO3D	Object-Oriented 3D Location Software
DBIO	Database input/output
3D	Three dimensional
RSTT	Regional seismic travel time (RSTT and SLBM are synonymous)
SLBM	Seismic location base model (RSTT and SLBM are synonymous)

1. INTRODUCTION

Prediction Calculator (PCalc) is a software package used for raytracing and travel-time computation using 3D Earth velocity models. The software uses the pseudo-bending algorithm of *Um and Thurber* (1987) and *Zhao and Lei* (2004). It is fully compatible with velocity models stored in GeoTess format (*Ballard, et al.,* 2016a) such as the SALSA3D model (*Ballard et al.,* 2016b) available on the webpage www.sandia.gov/salsa3d.

PCalc is distributed through GitHub at https://github.com/sandialabs/Salsa3DSoftware. In addition to PCalc, the tools GeoTess (used to build and interact with GeoTess velocity models) and LocOO3D (used to do event locations based on GeoTess velocity models) are provided through the same GitHub distribution.

PCalc is packaged with the latest version of this user's manual and a set of run examples described in Section 3. To compile PCalc from the source code, the user will need to have the Maven software package (https://maven.apache.org/index.html) and Java ver. ≥ 10 installed.

PCalc has three primary run modes:

1. Compute predictions of travel time, azimuth, slowness, and other predicted values at user specified source-receiver positions.

2. Extract model values from GeoTess models at user specified positions.

3. Compute ray path geometries through GeoTess models.

PCalc has many useful features, especially for monitoring applications, including:

- The ability to trace phases P, PP, pP, PKPdf, PKPbc, and Pn through 3D models of Earth's compressional-wave velocity structure and the ability to trace phases S, pS, SS, sS, SKS, and Sn through 3D models of Earth's shear-wave velocity structure.

- Computation of travel-time and travel-time uncertainty for the above phases using any model stored in GeoTess format. In addition, travel-times can be computed using the AK135 model, which is stored within the Pcalc software.

- The ability to directly interact with CSS3.0 format data tables stored in an Oracle database for both input and output, including the insertion of newly computed origins into existing tables.

- Geographic positions at which models can be queried or travel-times computed can be specified in a variety of formats, including a grid contained in an ascii text file, a 2D grid of points distributed in depth along a great circle path or a 3D grid specified by regular vertices on the earth and in depth.

In the following sections, an introduction to relevant PCalc concepts will be provided, followed by installation instructions, and a thorough tutorial of PCalc functionalities detailed in 11 examples.

1.1. Pseudo-bending in PCalc

The ray tracing algorithm in PCalc, aka "Bender", is an implementation of the pseudo-bending algorithm of *Um and Thurber* (1987) and *Zhao and Lei* (2004). The algorithm has been adapted to work with GeoTess models, which can contain interfaces separating regions of smoothly varying velocity, by ensuring that Snell's law is satisfied at these interfaces. This algorithm is described in detail in *Ballard, et al.* (2009). Bender is the same ray tracer/travel time predictor used in the construction of the SALSA3D velocity models.

Some advantages of the pseudo-bending approach used in Bender include:

- The algorithm finds a ray path between specified source and receiver locations by finding paths that locally minimize the travel time between these locations. This ability to specify the starting and ending points of each ray is convenient for tomography studies.

- This algorithm is computationally efficient, requiring only modest computational resources to quickly calculate travel times for rays through 3D models of the Earth's seismic velocity structure.

- Pseudo-bending is a mature approach to ray path computation. The Bender algorithm has been validated against several other approaches to ray path computation. See *Ballard, et al.*, (2009) for more details.

The compatibility of PCalc with GeoTess models means that a user can compute travel times and ray paths through any Earth velocity model (e.g., SALSA3D) that can be represented using the GeoTess format. The companion software package LocOO3D can also use Bender to compute seismic event locations using ray paths computed for arbitrary Earth models in GeoTess format.

1.2. seismicBaseData

PCalc often needs access to 2-dimensional lookup tables for travel time, azimuth, and slowness information. The 2 dimensions refer to distance and depth. Lookup tables for azimuth and slowness are rarely used but travel time is used very frequently. Travel time lookup tables for a great variety of radially symmetric velocity models have been generated but tables for models ak135 and iasp91 are the most common.

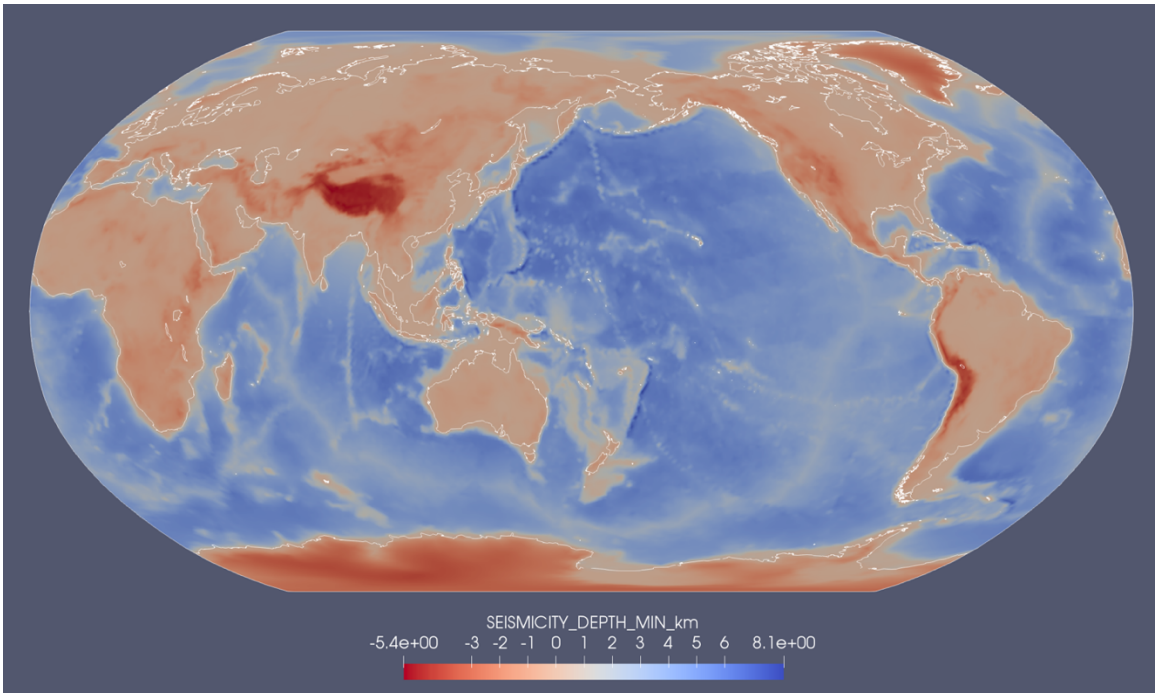
PCalc expects 2D lookup tables to be stored in a directory structure called seismicBaseData where a directory called seismicBaseData contains subdirectories tt (travel time), az (azimuth), sh (horizontal slowness), and el (ellipticity corrections). Within each of those subdirectories would be another level of subdirectories with names that correspond to a particular model, e.g., ak135, iasp91, etc. Within each of those subdirectories would be files with names that correspond to a particular phase. For example, a very commonly used lookup table would reside in seismicBaseData/tt/ak135/Pn.

While PCalc can access seismicBaseData in an external directory on the user's file system, PCalc includes a copy of seismicBaseData in the PCalc jar file. These include travel time and ellipticity corrections for model ak135 and travel-time tables for iasp91.

PCalc property seismicBaseData (Appendix A.2.9.6) can be used to specify the path to a directory on the user's file system where lookup tables reside. If property seismicBaseData is set to 'seismic-base-data.jar' or is omitted altogether, the lookup tables stored in the PCalc jar file generated in Section 2 will be used.

1.3. Seismicity Depth Model

PCalc includes a seismicity depth GeoTess model in the PCalc jar file that specifies the allowable depth range of traveltime calculations as a function of geographic location. The upper limit (seismicity_depth_min) is defined as the topography/bathymetry of the Earth and ranges from -5.4 to 8.1 km depth. The lower limit (seismicity_depth_max) is based on average depths of historic seismicity and tectonics and ranges from 50 to 700 km depth. Maps of the upper and lower limits of the default seismicity depth model are shown below:



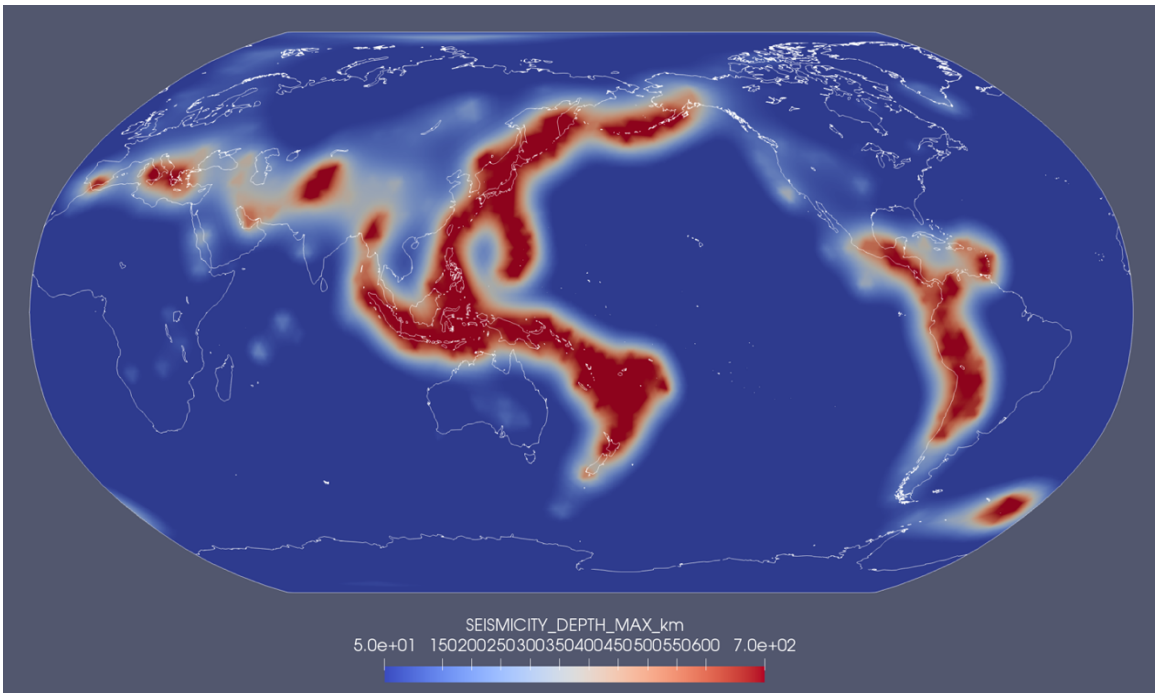


Figure 1. Seismicity depth model. (Top) Upper boundary based on elevation. (Bottom) Lower boundary based on historic seismicity.

The seismicity depth model is enforced by default. To use another seismicity depth model, the user can use the property seismicity_depth_model (Appendix A.2.1.5) to point to another seismicity depth model in GeoTess format.

2. PCALC INSTALLATION

To install PCalc, clone the Salsa3DSoftware package located at https://github.com/sandialabs/Salsa3DSoftware to the desired location. This will create a Salsa3dSoftware directory. Once the directory is cloned, run the following steps:

1. cd into the Salsa3dSoftware directory and build the executable jar file using Maven by running:

   mvn clean package

   This command creates the jar file **salsa3d-software-1.2023.4-jar-with-dependencies.jar** in the target folder (~/Salsa3DSoftware/target). Note that this jar file contains the classes for all three Salsa3DSoftware packages (LocOO3D, GeoTess, PCalc).
2. Run the **configure.sh** script located in ~/Salsa3DSoftware. This script creates executable shell scripts that can be used to access and run the desired tool contained within the jar file. In this case, the necessary executables are the **pcalc** and **geotessbuilder** shell scripts.
3. The configure.sh script also outputs a recommendation on how to update a **.bash_profile** or similar shell configuration file to include the generated executables on the user's path. For example:

   Add this line to your .bash_profile
   export PATH=/Users/username/Salsa3DSoftware:$PATH

   It is highly recommended the user follow this recommendation for ease of use.

3. PCALC TUTORIAL AND EXAMPLES

The input files for a PCalc run are:

1. properties file – A file containing the property settings, i.e., properties, used as input by PCalc for a particular run. Properties files are labeled with a ***.properties** extension.

    a. Available PCalc properties are described in detail in Appendix A.

    b. Properties can be written in any order in the properties file.

    c. Properties are case-sensitive, e.g., workdir ≠ workDir.

    d. A properties file must be provided as an argument. Otherwise, a warning will be printed to screen asking the user to specify a properties file.

    e. Example properties files are provided in ~/Salsa3DSoftware/Examples/PCalc and are described in Section 3.1.

2. An optional GeoTess file containing a desired Earth velocity model. If this file is not specified, PCalc will compute travel times with the AK135 lookup tables included within the PCalc software. GeoTess files are labeled with a ***.geotess** extension.

    In Sections 3.1.1 and 3.1.2 the user will use the velocity model SALSA3Dv2.1.geotess to run the examples. Click here to download the SALSA3Dv2.1 model.

    As the SALSA3D model will be used frequently throughout this text, a softlink to the model file can be created so the full path does not need to be entered in each property file requiring it. To do this, run:

    ln -s /path/to/model/file/salsa3d_v2.1.geotess salsa3d_model

3. An optional text file containing the latitude, longitude, and depth coordinates to which travel times, ray paths, or model queries are to be computed. This file is not required if a regular grid or grid along a great circle is used.

    a. In Sections 3.1.1 and 3.1.2, the user will use the data file MJAR.coords.xyz to run the example. MJAR.coords.xyz consists of a list of 100 locations specified in latitude (degrees), longitude (degrees), and depth (km).

    b. MJAR.coords.xyz is included with the PCalc software package.

4. An optional, specially configured folder containing the necessary files for the generation of LibCorr3d surfaces (see Section 3.3 for details on LibCorr3d surfaces). Due to the size of this folder (on the order of ~100-200 GB disk space), it will be made available upon request. Place requests for this folder to acconle@sandia.gov.

PCalc will automatically set relevant properties equal to the necessary files in this folder. The user only needs to input the path to the folder in the properties file (Section 3.3).

A brief definition for each of the folder contents is given below. More detailed definitions are beyond the scope of this guide.

a. prediction_model.geotess – the slowness model refined to a relatively dense resolution, typically 1 degree, suitable for calculating travel time using PCalc.

b. tomo_model.geotess – a relatively coarse, variable resolution version of the slowness model computed by tomography. Less suitable for computing travel times but required for the computation of path dependent travel time uncertainty using the covariance matrix.

c. activenodeIndexMap – a binary file detailing which GeoTess nodes in the model were modified when the tomography solution was calculated. Nodes that were inactive, such as the crust and core in SALSA3D models, were not included in inversion and do not differ from the starting model.

d. diagonalModel.geotess – a GeoTess file containing the diagonal elements of the Cholesky, covariance, and resolution matrices for the GeoTess slowness model. It has the same geometry as the tomo_model.geotess. Note that this file is not used as input for PCalc but provides useful model information that can be queried using GeoTess (see ~/Salsa3Dsoftware/documents/GeoTessUsersManual.pdf). In particular, this file can be used to generate figures of the slowness uncertainty and model resolution.

e. layer_standard_deviations.properties – a properties file containing the property slownessLayerStandardDeviation_P, which defines uncertainties for inactive nodes (see bullet c) for each layer.

    i. The property is set equal to a semicolon delimited list of layer/uncertainty pairs (e.g., UPPER_CRUST 0.1 sets the upper_crust slowness uncertainty to 0.1 seconds/km)

    ii. Change property name to slownessLayerStandardDeviation_S if using an S model.

f. ginv – a folder containing binary files that together make up the full covariance matrix necessary to generate the LibCorr3d path dependent uncertainties. This folder will be large, on the order of ~100-200 GB disk space.

g. distance_dependent_uncertainty – contains 1D distance dependent uncertainty information for various phases. Used when the benderUncertaintyType property is set to DistanceDependent (see Section 3.3).

h. readme.txt – a text file that provides relevant information about the model.

Several example inputs are included in the "Examples" directory of the PCalc distribution. In the following section the properties files and expected outputs for each example will be described. Due to the number of available properties and potential properties combination, we cannot provide an exhaustive list of examples that can cover all potential situations and uses. However, these examples should be sufficient to comfortably begin using PCalc.

Once the software has been installed and the environmental variables have been set (Section 2), all examples will be run on the command line as:

pcalc *property_file_name*.properties.

The examples in Section 3.1 detail using PCalc to query input models for attributes such as P-wave slowness. The examples in Section 4.2 detail using PCalc to predict travel-times. The examples in Section 4.3 detail using PCalc to generate ray paths via raytracing through an input velocity model.

Finally, Section 4.4 details an example of generating LibCorr3d surfaces using PCalc. The SALSA3D covariance matrix needed to run this example is not provided on Salsa3dSoftware GitHub due to its size. Please email acconle@sandia.gov if you would like the covariance matrix sent over massive file transfer.

3.1. PCalc Model Query Examples

3.1.1. Example 1 – PCalc Model Query Using MJAR.coords.xyz File

To execute Example 1, cd into ~/Salsa3DSoftware/Examples/PCalc/query_file. In this directory, the properties file **pcalc_query_file.properties** (Figure 2) and the input file **MJAR.coords.xyz** are provided.

```
#================================================================
#
# Property file for application PCalc v. 3.0
#
#================================================================
application=model_query
workDir=.
#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================
geotessModel=<property:workDir>/../../salsa3d_model
#================================================================
#
# INPUT PARAMETERS: GENERAL
#
#================================================================
#inputType must be one of [file | greatcircle | grid]
inputType=file
#================================================================
#
# INPUT PARAMETERS: FILE INPUT
#
#================================================================
# inputFile only applies if inputType=file
inputFile=<property:workDir>/MJAR.coords.xyz

# If input_header_row = true then first line of the input file that is not blank
# and not a comment will be interpreted as column headings that describe what
# each column contains.
#
# The default value is 'lat lon depth'
inputAttributes=latitude longitude depth

# input records will be processed in batches of this size (default is 10,000)
batchSize=10
#================================================================
#
# OUTPUT PARAMETERS
#
#================================================================
# if outputFile is specified then output is written to the specified file
# output is written to stdout
outputFile=<property:workDir>/pcalc_query_file_output.dat
#optional log file
logFile=<property:workDir>/pcalc_log.txt
# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput=true
# character to use to as field separator in the output file
# options are tab, space, comma
separator=space
# the following list of ouput attributes will be computed using each predictor
# and appended to the end of the input record.  If there is a header row in the
# output, then the attribute name is prepended with the predictor name in the
# header.
outputAttributes=pslowness
# List of available outputAttributes:
# pvelocity
```

Figure 2. PCalc Model Query Properties File pcalc_query_file.properties.

pcalc_query_file.properties (Figure 2) contains input properties to query a model at locations specified in the file MJAR.coords.xyz (see Section 3, bullet 3).

The properties file in Figure 2 begins with general properties that will be common across all examples in this manual. These properties are required in any properties file – if they are not specified, an error will be thrown and the user will be asked to specify the needed properties. These properties are:

- **application** – specifies the type of application for PCalc to perform. The types can be model_query or prediction (see Appendix A.2.1.1)
- **workDir** – path to the working directory where PCalc output are to be stored. If the specified working directory does not exist, PCalc will automatically generate the directory at the specified path (also see Appendix A.2.1.2)

In this example, these properties are set to the following:

```
application = model_query

workDir = .
```

The model_query option of the application property specifies that PCalc will be used to extract, i.e., query, a specified input model for any user-requested data or metadata information. In the example shown in Figure 2, the user requests that PCalc query the SALSA3Dv2.1 model via the softlink salsa3d_model using the geotessModel property:

```
geotessModel= <property:workDir>../../salsa3d_model
```

If the salsa3d_model softlink was not created in Section 3, bullet 2, the path must be modified to point to another valid GeoTess model or softlink.

Note the special notation <property:workDir> in the geotessModel path. In PCalc, <property:*property_name*> specifies that the value of a property with name *property_name* specified elsewhere in the property file is substituted in that location when PCalc is running. Thus, in this example, the workDir path is substituted for <property:workDir> such that the full geotessModel path is:

/Users/username/Salsa3DSoftware/Examples/PCalc/query_file/../../salsa3d_model

This type of notation can be used to quickly substitute any property value.

To query a model, the user must specify which model locations are to be queried. To provide locations, the user specifies the inputType property. Here, inputType is specified as:

```
inputType = file
```

inputType can be defined as a file, database, great circle, or a grid (see Appendix A.2.2.1). Here we will describe the file option, with other options discussed in later sections. When the user sets inputType equal to file, an input text file will be used to specify the event location coordinates at which to query the model. The input text file is specified using the property inputFile:

```
inputFile = <property:workdDir>/MJAR.coords.xyz
```

The exact contents of the text file, here MJAR.coords.xyz, are specified using the inputAttributes property. In Figure 2, inputAttributes are set to:

```
inputAttributes = latitude longitude depth
```

Thus, MJAR.coords.xyz must contain a latitude (degrees), longitude (degrees), and depth (km) column separated by white space. To modify what attributes to use or the order in which the attributes are listed in the file, the user can modify the inputAttributes property (Appendix A.2.3.3). For instance, if the user wants to specify proposed event epicenters rather than hypocenters, they can specify inputAttributes to equal latitude and longitude only. Or if the user wants longitude listed first, they can set inputAttributes = longitude latitude depth. If the text file does not match the defined inputAttributes, PCalc will throw an error, either because it does not get the values it expects, or the number of attributes does not match the number of columns. Note that when performing a model query, the default attributes are "longitude latitude depth" if inputAttributes is not defined.

The property inputFile is used to define the path to the desired input file, in this case MJAR.coords.xyz (Section 3, bullet 3):

```
inputFile = <property:workDir>/MJAR.coord.xyz
```

MJAR.coords.xyz contains latitude, longitude, depth columns, thus inputAttributes is set to "latitude longitude depth" in this example.

The next property, batchSize, is an optional property that allows the user to define the size of record batches read in from the file. In this example, batchSize is defined as:

```
batchSize = 10
```

Thus, PCalc will read in 10 records, i.e., rows, from MJAR.coords.xyz at a time. By default, batches are of size 10,000 (see Appendix A.2.9.4). batchSize can only be applied when using a file or database (Section 3.2.4) for input.

Now that the input properties have been set, the remainder of the properties file sets properties dealing with file output.

The property outputFile (Appendix A.2.8.1) defines the path and file to write the PCalc results to. Here, the path is defined as:

```
outputFile = <property:workDir>/pcalc_query_file_output.dat
```

When PCalc is run, the output will be written in the text file pcalc_query_file_output.dat. Note that the outputFile property is required unless the inputType property is set to database (see Appendix A.2.2.1), where it will be ignored if defined.

In addition to the output file above, the user can also optionally save a log file by setting the logfile (Appendix A.2.1.3) property equal to the path and filename to save the logfile to. In this example, the path is defined as:

```
logFile = <property:workDir>/pcalc_log.txt
```

The next property, terminalOutput (Appendix A.2.1.4) echoes the general information about the PCalc run to screen when it is set to true. This same information is written in the log file when it is requested. In this example, terminalOutput is set to true:

```
terminalOutput = true
```

The terminalOutput is true by default, so setting the property to true in Figure 2 is for demonstration purposes only.

The property separator (Appendix 0) is used to define what type of separator will be used for the PCalc output text file. The separator can be defined as a space (white space), comma, or tab. In this example, separator is defined as:

```
separator = space
```

The separator is set to space by default, so as with terminalOutput setting the property to space in Figure 2 is for demonstration purposes only.

The final property, outputAttributes (Appendix A.2.8.6) is used to define what output will be written to the file defined by the outputFile property. Currently, the only outputAttributes available when the application property is set to model_query are pslowness (P-wave slowness), pvelocity (P-wave velocity), sslowness (S-wave slowness), and svelocity (S-wave velocity). In turn, the availability of these options also depends on the model being queried – if the input model has no S-wave data, sslowness and svelocity will not be available outputAttributes.

In this example, outputAttributes is set to pslowness:

```
outputAttributes = pslowness
```

With all properties set, the properties file in Figure 2 can be run as **pcalc pcalc_query_file.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_query_file_output.dat). If successful, the output file should contain 100 rows and four columns separated by white space. The first four rows are shown below.

```
13.710173 144.000000 80.018 0.1213
13.710173 144.000000 75.018 0.1214
13.710173 144.000000 70.018 0.1217
13.710173 144.000000 65.019 0.1221
…         …          …      …
```

The columns in this case represent the latitude, longitude, and depth being queried (these should be the same values as the original MJAR.coords.xyz file) and the P-wave slowness in s/km at that queried event hypocenter.

3.1.2. Example 2 – PCalc Model Query Using a Great Circle

To execute Example 2, cd into ~/Salsa3DSoftware/Examples/PCalc/query_greatcircle. In this directory, the properties file **pcalc_query_greatcircle.properties** (Figure 3) is provided.

```
#================================================================
#
# Property file for application PCalc v. 3.0
#
#================================================================
application = model_query
workDir = .

#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================

geotessModel = <property:workDir>/../../salsa3d_model

#================================================================
#
# INPUT PARAMETERS: GENERAL
#
#================================================================

#inputType must be one of [file | greatcircle | grid]
inputType = greatcircle

#================================================================
#
# INPUT PARAMETERS: GREATCIRCLE
#
#================================================================

# latitude and longitude of start of greatcircle path, in degrees
gcStart = 0 0

# specify the distance and azimuth to move, in degrees relative to gcStart
gcDistance = 180
gcAzimuth = 0

# maximum spacing of points along greatcircle path, in degrees.
# Actual spacing will generally be smaller so that spacing is even.
gcSpacing = 10

# output position parameters.
# any subset of [x, y, z, latitude, longitude, distance, depth]
gcPositionParameters = x, y, distance, depth

#================================================================
#
# INPUT PARAMETERS: DEPTHS
#
#================================================================

# all depths are specified in km.  The various methods of specifying
# depth are searched until one is found that is valid.
# Search order is depths, depthRange, depthLevels, maxDepthSpacing.

depthSpecificationMethod = maxDepthSpacing

# specify that there should be a depth at the top and bottom of each
# layer and additional depths in the interior of each layer such that
# the maximum spacing of depths is no greater than maxDepthSpacing.
maxDepthSpacing = 100

# if maxDepthSpacing was specified, then maxDepth can also be used to
# specify some maximum depth for the bottom of the grid.
# Default value is center of the earth.
maxDepth = top of m660
```

Figure 3. PCalc Model Query Properties File pcalc_query_greatcircle.properties.

```
#============================================================
#
# OUTPUT PARAMETERS
#
#============================================================

# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile = <property:workDir>/pcalc_query_greatcircle_output.dat

#optional log file
logFile = <property:workDir>/pcalc_log.txt

# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput = true

# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader = true

# character to use to as field separator in the output file
# options are tab, space, comma
separator = tab

# the following list of ouput attributes will be computed using each predictor
# and appended to the end of the input record.  If there is a header row in the
# output, then the attribute name is prepended with the predictor name in the
# header.
outputAttributes = pslowness

# List of available outputAttributes:
# pvelocity
# pslowness
```

Figure 2 (cont). PCalc Model Query Properties File pcalc_query_greatcircle.properties

The example in **pcalc_query_greatcircle.properties** is nearly identical to Example 1, except that rather than querying the model using event location coordinates in a file, the model is queried using coordinates along a great circle defined in the properties file.

Due to this similarity, only new or modified properties will be discussed in this section. For properties in common with Example 1, refer to Section 3.2.1.

First, inputType (defined in Section 3.1.1) must now be set equal to greatcircle.

```
inputType = greatcircle
```

When the inputType is set to greatcircle, several additional properties are required to define the great circle along which the input model will be queried.

The first property, gcStart (Appendix A.2.4.1) is used to set the beginning of the great circle. It is defined as a latitude and longitude separated by a white space. For instance, in this example gcStart is set to:

```
gcStart = 0 0
```

Thus, this great circle will begin at 0 degrees latitude, 0 degrees longitude. Although not shown in this example, there is an equivalent property called gcEnd (Appendix A.2.4.2) that can be used to set the end of the great circle in the same way.

The next properties, gcDistance (Appendix A.2.4.3) and gcAzimuth (Appendix A.2.4.4), specify the distance and azimuth the great circle will traverse. The gcDistance value is defined as the epicentral distance in degrees from gcStart to the end of the great circle. The gcAzimuth value is defined as the azimuthal direction in degrees the great circle must go. Note that these properties will be ignored if gcEnd is specified.

In this example, gcDistance and gcAzimuth are defined as:

```
gcDistance = 180
```

```
gcAzimuth = 0
```

Thus, the defined great circle will travel 180 degrees from a start of 0 degrees lat, 0 degrees lon in an azimuthal direction of 0 degrees.

The user next needs to specify the spacing between points of the great circle. This is done by specifying the property gcSpacing (Appendix A.2.4.6). Here gcSpacing is defined as the approximate spacing, in degrees, between adjacent points. The actual spacing may be reduced somewhat from the specified value in order for an integer number of equally spaced points to span the length of the great circle.

In this example, gcSpacing is defined as:

```
gcSpacing = 10
```

Thus, the great circle will consist of 19 points (18 points in addition to the 0 lat, 0 lon point), with adjacent points separated by approximately 10 degrees distance.

The final great circle property that is defined in this example is gcPositionParameters (Appendix A.2.4.8). This property is used to set how the great circle geometry should be output. The available options are:

- latitude – the latitude of the point in degrees.
- longitude – the longitude of the point in degrees.
- distance – the epicentral distance from the beginning of the great circle (gcStart) to the point, in degrees.
- depth – the depth of the point in km relative to sea level.
- radius – the radius of the point in km.
- x, y, z – Consider the plane of the great circle and consider each point to be a vector from the center of the earth to the point. The y direction is a unit vector from the center of the earth to a point halfway along the great circle path. The z direction is a unit vector that is normal to the plane of the great circle, pointing in the direction of the observer. x is a unit vector defined by y cross z. This coordinate system is useful for plotting points in a manner that shows the curvature of the surface of the earth and the various seismic discontinuities within it. z will always be zero in this application. To convert x, y, z to distances in km, multiply the x, y, z values by the Earth radius at that point.

The user can specify any combination of the above parameters for output. For instance, in this example gcPositionParameters is defined as:

```
gcPositionParameters = x, y, distance, depth
```

Thus, the output file will define each point of the great circle with unit vectors x and y, epicentral distance in degrees, and depth in km.

Finally, the user needs to define properties that specify the depth of the great circle. In this example, we begin with the depthSpecificationMethod property (Appendix A.2.7.1). This property defines the method that will be used to specify depths at which the model queries are to be calculated. The available options are:

1. depths – specify a list of depths, in km, that will be used for every latitude-longitude position (Appendix A.2.7.2).
2. depthRange – specify a minimum and maximum depth, in km, and the number of desired depths (Appendix A.2.7.3).
3. depthLevels – depth will be determined above and below a specified layer in the model (see Appendix A.2.7.4 for details).
4. maxDepthSpacing – unique depth profiles will be generated at each geographic position such that:
    a. each profile has the same number of depths.
    b. there are two depth nodes at each major layer interface in the model, one of which records model properties above the interface and the other below the interface.
    c. the maximum spacing of depth nodes is no greater than maxDepthSpacing.
    d. See Appendix A.2.7.5 for details.

In this example, depthSpecificationMethod is set to maxDepthSpacing:

```
depthSpecificationMethod = maxDepthSpacing
```

Since depthSpecificationMethod is set to maxDepthSpacing, the user also specifies the properties maxDepthSpacing (Appendix A.2.7.5) and maxDepth (Appendix A.2.7.6). maxDepthSpacing is defined in bullet 4. maxDepth sets the deepest point returned in each profile, where the maximum depth can be specified as a depth in km or as a model layer/interface name such as "moho" or "cmb" (see Appendix A.2.7.6 for details).

In this example, these properties are set to:

```
maxDepthSpacing = 100

maxDepth = top of m660
```

Thus, at each point of the great circle the requested outputAttribute (here pslowness; see Section 3.1.1 for details on outputAttribute) is calculated at depths separated by a maximum value of 100 km until the top of the M660 layer is reached. In cases where a depth is at a major layer interface, the outputAttribute directly above and below the layer is reported.

The remaining properties in this example, with the exception of outputHeader, were covered in Section 3.1.1. outputHeader generates a column heading for each column of output on the first line of the file. By default, outputHeader is set to false. Here, it has been set to true:

```
outputHeader = true
```

With all properties set, the properties file in Figure 3 can be run as **pcalc pcalc_query_greatcircle.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_query_greatcircle_output.dat). If successful, the output file should contain 343 rows and five columns (x, y, distance in degrees, depth in km, pslowness in s/km) separated by tabs. The first four rows are shown below.

x	y	distance	depth	pslowness
-1.000000000	0.000000000	0.000000000	4.755652	0.4726
-1.000000000	0.000000000	0.000000000	5.898230	0.4726
-1.000000000	0.000000000	0.000000000	5.898230	0.2000
-1.000000000	0.000000000	0.000000000	7.597937	0.2000
…	…	…	…	…

3.1.3. Example 3 – PCalc Model Query Using a Grid

To execute Example 3, cd into ~/Salsa3DSoftware/Examples/PCalc/query_grid. In this directory, the properties file **pcalc_query_grid.properties** (Figure 4) is provided.

```
#================================================================
#
# Property file for application PCalc v. 3.0
#
#================================================================

workDir = .

application = model_query

#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================

geotessModel = <property:workDir>/../../salsa3d_model

#================================================================
#
# INPUT PARAMETERS: GENERAL
#
#================================================================

#inputType must be one of [file | greatcircle | grid]
inputType = grid

#================================================================
#
# INPUT PARAMETERS: GRID
#
#================================================================

# gridRangeLat specifies lat1, lat2, number of points
# gridRangeLon specifies lon1, lon2, number of points
gridRangeLat = 15 45 16
gridRangeLon = 70 110 21

# if outputHeaderRow is true then a column heading will be generated for
# each column of output and appear as the first line of the output file.
outputHeaderRow = true

#================================================================
#
# INPUT PARAMETERS: DEPTHS
#
#================================================================

# the depth of all grid points will be set to
# value specified by depths = comma or space deliniated list of depths

depthSpecificationMethod = depths
depths = 100.0, 200.0
```

Figure 4. PCalc Model Query Properties File pcalc_query_grid.properties.

```
#================================================================
#
# OUTPUT PARAMETERS
#
#================================================================

outputType = file

# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile = <property:workDir>/pcalc_query_grid_output.dat

#optional log file
logFile = <property:workDir>/pcalc_log.txt

# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput = true

# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader = true

outputFormat = %1.4f

# character to use to as field separator in the output file
# options are tab, space, comma
separator = tab

# the following list of ouput attributes will be computed using each predictor
# and appended to the end of the input record.  If there is a header row in the
# output, then the attribute name is prepended with the predictor name in the
# header.
outputAttributes = pslowness

# List of available outputAttributes:
# pvelocity
# pslowness
```

Figure 3 (cont). PCalc Model Query Properties File pcalc_query_grid.properties.

The example in **pcalc_query_grid.properties** is nearly identical to Examples 1 and 2, except that the model is now queried using coordinates along a grid defined in the properties file.

Due to this similarity, only new or modified properties will be discussed in this section. For properties in common with Examples 1 or 2, refer to Sections 3.1.1, 3.1.2.

First, inputType (defined in Section 3.1.1) must now be set equal to grid.

```
inputType = grid
```

When the inputType is set to grid, several additional properties are required to define the grid along which the input model is queried.

The two main properties needed are gridRangeLat and gridRangeLon (Appendices A.2.5.1, A.2.5.2). These define the minimum lat/lon, maximum lat/lon, and the number of lats/lons at which the model should be queried. For instance, in this example gridRangeLat and gridRange Lon are defined as:

```
gridRangeLat = 15 45 16
```

```
gridRangeLon = 70 110 21
```

Thus, the grid will range in latitude from 15 to 45 degrees, with 16 latitudes calculated within that latitude range, and the grid will range in longitude from 70 to 110 degrees, with 21 longitudes calculated within that longitude range.

Note that the depthSpecificationMethod discussed in Section 3.1.2 is set to depths in this example. Thus, a list of depths is provided using the depths parameter:

```
depthSpecificationMethod = depths
```

```
depths = 100.0, 200.0
```

The depths are specified in a comma-separated list. In this example only two depths of 100.0 and 200.0 km are requested.

Note that there is no equivalent to the gcPositionParameters property in Section 3.1.2. Thus, the output file will always have columns of longitude (in deg), latitude (in deg), depth (in km), and the requested outputAttribute(s) (here pslowness).

With all properties set, the properties file in Figure 4 can be run as **pcalc pcalc_query_grid.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_query_grid_output.dat). If successful, the output file should contain 673 rows and four columns (longitude in deg, latitude in deg, depth in km, pslowness in s/km) separated by tabs. The first four rows are shown below.

longitude	latitude	depth	pslowness
70.000000000	15.000000000	100.000000	0.1267
70.000000000	15.000000000	200.000000	0.1231
72.000000000	15.000000000	100.000000	0.1270
72.000000000	15.000000000	200.000000	0.1235

3.2. PCalc Travel Time Prediction Examples

3.2.1. Example 4 – PCalc Travel Time Prediction Using MJAR.coords.xyz File

To execute Example 4, cd into ~/Salsa3DSoftware/Examples/PCalc/predictions_file. In this directory, the properties file **pcalc_predictions_file.properties** (Figure 5) and the input file **MJAR.coords.xyz** are provided.

```
#============================================================
#
# Property file for application Predictions v. 3.0
#
#============================================================

# specify max number of cores this process is allowed use. Default is all.
maxProcessors = 4

application = predictions

workDir = .

#============================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#============================================================

# predictors may include any combination of (lookup2d, bender)

predictors = bender

benderModel = <property:workDir>/../../salsa3d_model

#============================================================
#
# INPUT PARAMETERS: GENERAL
#
#============================================================

#inputType must be one of [file | database | greatcircle | grid]

inputType = file

#============================================================
#
# INPUT PARAMETERS: FILE INPUT
#
#============================================================

# inputFile only applies if inputType=file
inputFile = <property:workDir>/MJAR.coords.xyz

# input records will be processed in batches of this size (default is 10,000)
batchSize = 10

# inputAttributes must include: site_lat, site_lon, [site_elev | site_depth], origin_lat, origin_lon, origin_depth and phase.
# If inputAttributes includes sta and [jdate | origin_time | arrival_time] then site terms can be computed.
# The default value is sta, jdate, site_lat, site_lon, site_elev, origin_lat, origin_lon, origin_depth, phase
inputAttributes = origin_lat, origin_lon, origin_depth

# phase, and lat, lon, elev of station are required
phase = P
site = 36.524717, 138.24718, .6617

# station name and jdate of the 'arrivals' are optional but if specified
# then bender will use the information to include tt_site_correction
sta = MJAR
jdate = 2011001
```

Figure 5. PCalc Prediction Properties File pcalc_predictions_file.properties.

```
#============================================================
#
# OUTPUT PARAMETERS
#
#============================================================

#outputType must = file | database
outputType = file

# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile = <property:workDir>/pcalc_predictions_file_output.dat

# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader = true

#optional log file
logFile = <property:workDir>/pcalc_log.txt

# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput = true

# character to use to as field separator in the output file
# options are tab, space, comma
separator = space

outputFormat = %8.4f

# the following list of ouput attributes will be computed
# and appended to the end of each input record.
outputAttributes = travel_time

# List of available outputAttributes. If a predictor does not support a
# specified outputAttribute, -999999.0 will be output.
#
# travel_time
# tt_model_uncertainty
# tt_site_correction
# tt_ellipticity_correction
# dtt_dlat
# dtt_dlon
# dtt_dr
# slowness
# slowness_degrees
# slowness_model_uncertainty
# slowness_model_uncertainty_degrees
# dsh_dlat
# dsh_dlon
# dsh_dr
# azimuth
# azimuth_degrees
# azimuth_model_uncertainty
# azimuth_model_uncertainty_degrees
# daz_dlat
# daz_dlon
# daz_dr
# backazimuth
# backazimuth_degrees
# turning_depth
# out_of_plane
# distance
# distance_degrees
# average_ray_velocity
# tt_elevation_correction
# tt_elevation_correction_source
# tt_ellipticity_correction
# calculation_time
```

Figure 4 (cont). PCalc Prediction Properties File pcalc_predictions_file.properties.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with Examples 1, 2, or 3, refer to Sections 3.1.1, 3.1.2, 3.1.3.

This is the first of a few examples detailing the prediction application of PCalc. All predictions are computed in concurrent parallel mode (multi-threaded) by default; predictions can also be computed sequentially by setting the property parallelMode (Appendix A.2.9.3) to sequential. When parallelMode is concurrent, the maximum number of processors PCalc can use to compute predictions must be defined using the property maxProcessors (Appendix A.2.9.2). By default, all processors available will be used if this property isn't specified. In this example, maxProcessors is defined as:

```
maxProcessors = 4
```

The application property (see Section 3.1.1) is now set to predictions:

```
application = predictions
```

When application is set to predictions, the property predictors (Appendix A.2.9.1) is required. The predictors property consists of a list of predictors that are to be used during the run. The available predictors are:

- lookup2d – perform predictions with the AK135 model included in the PCalc jar file.
- bender – perform predictions through an input model using the Bender ray-tracing algorithm. See Section 1.1 for details on Bender.
- rstt – perform predictions through an input model using the Regional Seismic Travel Time (RSTT) method (see link).

Different predictors can be applied to different seismic phases. For instance, if predictors is set equal to "lookup2d, bender(P, Pn), rstt(Pn, Pg)" then lookup2d will be used as the predictor for all phases not specified later in the list, Bender will be used for phase P, and RSTT will be used for phases Pn and Pg. Note that if a phase is included in multiple predictors (Pn is included in both bender and rstt in this example), the final predictor in the list takes precedence, hence why Pn is only estimated using rstt.

In this example, predictors is set to bender for all phases:

```
predictors = bender
```

When the predictors property is set to bender, the benderModel property (Appendix A.2.9.13) needs to be defined to provide an input model. This property provides the path to the GeoTess model that Bender will use to calculate predictions of seismic observables. In the example shown in Figure 5, the user requests that Bender perform predictions using the SALSA3Dv2.1.geotess model via the softlink salsa3d_model:

```
benderModel= <property:workDir>../../salsa3d_model
```

If the salsa3d_model softlink was not created in Section 3, bullet 2, the path must be modified to point to another valid GeoTess model or softlink.

As in Section 3.1.1, the MJAR.coords.xyz file provides the proposed hypocentral locations of seismic events as input. Note that while inputType and inputFile are defined in the same way as in Section 3.1.1, inputAttributes is now defined as:

```
inputAttributes = origin_lat, origin_lon, origin_depth
```

whereas in Section 3.1.1, inputAttributes was defined as "latitude longitude depth". This change is because the application is now predictions rather than model_query.

When the PCalc application is predictions, by default inputAttributes (Appendix A.2.3.3) is set to "sta jdate site_lat site_lon site_elev origin_lat origin_lon origin_depth phase", where:

- sta is the recording station – used to apply site corrections.
- jdate is the Julian date when the event origin was recorded – used to apply site corrections.
- Site_lat and site_lon are the latitude and longitude in degrees of the recording station.
- site_elev is the elevation of the recording site in km.
- origin_lat and origin_lon are the latitude and longitude in degrees of the event.
- phase is the seismic phase recorded.

Thus, when the default inputAttributes is used, the text file must contain columns corresponding to each of the above attributes. At a minimum, the inputAttributes origin_lat and origin_lon must be included. Note that the inputAttribute origin_depth is also optional – if origin_depth is not defined in the text file, it must be defined using depth properties (see Appendix A.2.7). The inputAttributes site_lat, site_lon, site_elev, and phase must also be defined using properties (see Appendices A.2.2.3, A.2.2.5).

In this example, inputAttributes sets the required text file columns to correspond to event hypocenters. Because the required attributes site_lat, site_lon, and phase were not specified in the input text file, we now set these properties in the following lines:

```
phase = P
```

```
site = 36.524717, 138.24718, .6617
```

In this example, Bender performs raytracing for first-arrival P from the input event locations to a station located at 36.524717 degrees latitude, 138.24718 degrees longitude, and 0.6617 km elevation.

Because this example is designed to predict travel-times, the additional properties sta and jdate are defined so travel-time site corrections can be applied to the travel-time predictions. In this example, we apply corrections for station MJAR that were valid on January 1, 2011:

```
sta = MJAR
```

```
jdate = 2011001
```

Finally, at the bottom of the properties file in Figure 5, the outputAttributes property is set. The predictions application has far more outputAttributes options than the model_query application seen in Section 3.1.1. These outputAttributes are listed in Appendix A.2.8.6 and can also be seen in the pcalc_predictions_file.properties file (Figure 5). In this example, outputAttributes is set to travel_time. Thus, the final output file will contain origin_lat, origin_lon, origin_depth, and travel_time columns, with each row indicating an event hypocenter and the travel-time (in sec) for first P to travel from that hypocenter to station MJAR.

Finally, in this example we see an additional property outputFormat (Appendix A.2.8.3) that can be used to format the output file.

With all properties set, the properties file in Figure 5 can be run as **pcalc pcalc_predictions_file.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_predictions_file_output.dat). If successful, the output file should contain 101 rows and four columns (origin_lat in deg, origin_lon in deg, origin_depth in km, travel_time in sec) separated by spaces. The first four rows are shown below.

```
origin_lat origin_lon origin_depth travel_time
13.710173 144.000000 80.018 299.3240
13.710173 144.000000 75.018 299.7400
13.710173 144.000000 70.018 300.1580
13.710173 144.000000 65.019 300.5790
…          …           …      …
```

3.2.2. Example 5 – PCalc Travel Time Prediction Using Great Circles

To execute Example 5, cd into ~/Salsa3DSoftware/Examples/PCalc/predictions_greatcircle. In this directory, the properties file **pcalc_predictions_greatcircle.properties** (Figure 6) is provided.

```
#================================================================
#
# Property file for application PCalc v. 3.0
#
#================================================================

workDir = .

application = predictions

#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================

# predictors may include any combination of (tauptoolkit, bender, slbm)
predictors =  lookup2d

#================================================================
#
# INPUT PARAMETERS: GENERAL
#
#================================================================

#inputType must be one of [file | greatcircle | grid]
inputType = greatcircle

#================================================================
#
# INPUT PARAMETERS: GREATCIRCLE
#
#================================================================

# phase, and sta, refsta, lat, lon, elev of station are required
phase = P
site = ARCES, ARCES, 10, 0, .2

# station name and jdate of the 'arrivals' are optional but if specified
# then bender will use the information to include tt_site_correction
sta = ARCES
jdate = 2011001


# latitude and longitude of start of greatcircle path, in degrees
gcStart = 0 0

# latitude and longitude of end of greatcircle path, in degrees
#gcEnd = 30 30

# specify the distance and azimuth to move, in degrees relative to gcStart
gcDistance = 180
gcAzimuth = 0

# maximum spacing of points along greatcircle path, in degrees.
# Actual spacing will generally be smaller so that spacing is even.
gcSpacing = 10

# output position parameters.
# any subset of [x, y, z, latitude, longitude, distance, depth]
gcPositionParameters = longitude, latitude, x, y, distance, depth
```

Figure 6. PCalc Prediction Properties File pcalc_predictions_greatcircle.properties.

```
#================================================================
#
# INPUT PARAMETERS: DEPTHS
#
#================================================================

# all depths are specified in km.  The various methods of specifying
# depth are searched until one is found that is valid.
# Search order is depths, depthRange, depthLevels, maxDepthSpacing.

depthSpecificationMethod = depths

# the depth of all grid points will be set to value specified by
# depths = comma or space deliniated list of depths
depths = 50 60 70

# specify that there should be a depth at the top and bottom of each
# layer and additional depths in the interior of each layer such that
# the maximum spacing of depths is no greater than maxDepthSpacing.
maxDepthSpacing = 100

# if maxDepthSpacing was specified, then maxDepth can also be used to
# specify some maximum depth for the bottom of the grid.
# Default value is center of the earth.
maxDepth = 500

#================================================================
#
# OUTPUT PARAMETERS
#
#================================================================

outputType = file

# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile = <property:workDir>/pcalc_predictions_greatcircle_output.dat

#optional log file
logFile = <property:workDir>/pcalc_log.txt

# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput = true

# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader = true

# character to use to as field separator in the output file
# options are tab, space, comma
separator = space

# the following list of ouput attributes will be computed
# and appended to the end of each input record.
outputAttributes = travel_time
```

Figure 5 (cont). PCalc Prediction Properties File pcalc_predictions_greatcircle.properties.

This properties file contains input properties to generate predictions for locations specified by a great circle.

With all properties set, the properties file in **Error! Reference source not found.** can be run as **p calc pcalc_predictions_greatcircle.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_predictions_greatcircle_output.dat). If successful, the output file should contain 58 rows and seven columns (longitude in deg, latitude in deg, x, y, distance in degrees, depth in km, travel_time in sec) separated by spaces. The first four rows are shown below.

```
longitude latitude x y distance depth travel_time
0.000000000 0.000000000 -1.000000000 0.000000000 0.000000000 50.000000 140.2321
0.000000000 0.000000000 -1.000000000 0.000000000 0.000000000 60.000000 140.1395
0.000000000 0.000000000 -1.000000000 0.000000000 0.000000000 70.000000 140.0606
0.000000000 10.066021190 -0.984807753 0.173648178 10.000000000 50.000000 8.0451
…           …            …             …           …            …         …
```

3.2.3. Example 6 – PCalc Travel Time Prediction Using a Grid

To execute Example 6, cd into ~/Salsa3DSoftware/Examples/PCalc/predictions_grid. In this directory, the properties file **pcalc_predictions_grid.properties** (Figure 7) is provided.

```
#================================================================
#
# Property file for application PCalc v. 3.0
#
#================================================================

workDir = .

application = predictions

#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================

# predictors may include any combination of (tauptoolkit, bender, slbm)
predictors =  bender

benderModel = <property:workDir>/../../salsa3d_model

#================================================================
#
# INPUT PARAMETERS: GENERAL
#
#================================================================

#inputType must be one of [file | greatcircle | grid]
inputType = grid

#================================================================
#
# INPUT PARAMETERS: GRID
#
#================================================================

# phase, and lat, lon, elev of station are required
phase = P
site = 36.524717, 138.24718, .6617

# station name and jdate of the 'arrivals' are optional but if specified
# then bender will use the information to include tt_site_correction
sta = MJAR
jdate = 2011001

# gridRangeLat specifies lat1, lat2, number of points
# gridRangeLon specifies lon1, lon2, number of points
gridRangeLat = 15 45 16
gridRangeLon = 70 110 21

#================================================================
#
# INPUT PARAMETERS: DEPTHS
#
#================================================================

# all depths are specified in km.  The various methods of specifying
# depth are searched until one is found that is valid.
# Search order is depths, depthRange, depthLevels, maxDepthSpacing.

depthSpecificationMethod = depthLevels

depthLevels = top of MOHO
```

Figure 7. PCalc Prediction Properties File pcalc_predictions_grid.properties.

```
#================================================================
#
# OUTPUT PARAMETERS
#
#================================================================

outputType = file

# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile = <property:workDir>/pcalc_predictions_grid_output.dat

#optional log files
logFile = <property:workDir>/pcalc_log.txt

# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput = true

# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader = true

# character to use to as field separator in the output file
# options are tab, space, comma
separator = space

# the following list of ouput attributes will be computed
# and appended to the end of each input record.
outputAttributes = travel_time
```

Figure 6 (cont). PCalc Prediction Properties File pcalc_predictions_grid.properties.

This properties file contains input properties to generate predictions for locations specified by a grid.

The properties in this file have been discussed in prior sections. Refer to Sections 3.1.1, 3.1.2, 3.1.3, 3.2.1, and Appendix A for information on these properties. Note that the bender predictor is used in this example.

With all properties set, the properties file in **Error! Reference source not found.** can be run as **p calc pcalc_predictions_grid.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_predictions_greatcircle_output.dat). If successful, the output file should contain 337 rows and four columns (longitude in deg, latitude in deg, depth in km, travel_time in sec) separated by spaces. The first four rows are shown below.

longitude	latitude	depth	travel_time
70.000000000	15.000000000	12.999901	632.1370
72.000000000	15.000000000	14.271083	621.1570
74.000000000	15.000000000	25.269918	608.8010
76.000000000	15.000000000	33.222848	596.6070
…	…	…	…

3.2.4. Example 7 – PCalc Travel Time Prediction Using a Database

To execute Example 7, cd into ~/Salsa3DSoftware/Examples/PCalc/predictions_db. In this directory, the properties file **pcalc_predictions_db.properties** (Figure 8) is provided.

```
#=====================================================
#
# Property file for application Predictions v. 3.0
#
#=====================================================

application = predictions
workDir = .

#=====================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#=====================================================

# predictors may include any combination of (lookup2d, bender, slbm)
predictors = lookup2d

#=====================================================
#
# INPUT PARAMETERS: GENERAL
#
#=====================================================

#inputType must be one of [file | database | greatcircle | grid]
inputType = database

# if dataLoaderType = oracle then specify database information
dbInputInstance = jdbc:oracle:thin:@database_instance.sandia.gov:port_number:database_instance
dbInputUserName = user_name
dbInputPassword = password

dbInputOriginTable = account_name.origin
dbInputAssocTable = account_name.assoc
dbInputArrivalTable = account_name.arrival
dbInputSiteTable = account_name.site

dbInputWhereClause = origin.orid = 15433650

# additional components of the where clause used to limit assocs.
dbInputAssocClause = assoc.phase in ('P', 'S', 'Pn', 'Sn', 'Pg','Lg')

#=====================================================
#
# OUTPUT PARAMETERS
#
#=====================================================

#outputType must = file | database
outputType = database

#optional log file
logFile = <property:workDir>/db_test_log.txt

# if terminalOutput is true then log information is written to stdout
terminalOutput = true

dbOutputInstance = <property:dbInputInstance>
dbOutputUserName = <property:dbInputUserName>
dbOutputPassword = <property:dbInputPassword>

dbOutputAssocTable = pcalc_assoc

dbOutputAutoTableCreation = true
dbOutputPromptBeforeTruncate = false
dbOutputTruncateTables = true
```

Figure 8. PCalc Prediction Properties File pcalc_predictions_db.properties.

This properties file contains input properties to generate predictions using lookup2d for locations specified in CSS3.0 schema type database tables.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 3.1.1 and 3.2.1.

In this example, inputType (defined in Section 3.1.1) must now be set equal to database.

```
inputType = database
```

There are also several new properties used to connect and interact with the database. The first new property is dbInputInstance (Appendix A.2.6.3), which is used to mount the database. In this example, a generic instance is provided.

```
dbInputInstance =
jdbc:oracle:thin:@database_instance.sandia.gov:port_number:database_instance
```

In addition to dbInputInstance, the properties dbInputUserName (Appendix A.2.6.1) and dbInputPassword (Appendix A.2.6.2) must be provided so the user can connect to their database account.

```
dbInputUserName = user_name
dbInputPassword = password
```

The user must update these properties prior to running this example.

The next properties in this example, dbInputOriginTable (Appendix A.2.6.7), dbInputAssocTable (Appendix A.2.6.8), dbInputArrivalTable (Appendix A.2.6.9), and dbInputSiteTable (Appendix A.2.6.10), are used as inputs to PCalc. They specify the database account and table name containing the origin, assoc, arrival, and site data to be used, respectively.

```
dbInputOriginTable = account_name.origin
dbInputAssocTable = account_name.assoc
dbInputArrivalTable = account_name.arrival
dbInputSiteTable = account_name.site
```

With the input tables specified, the user can specify which data to extract from the tables using the dbInputWhereClause (Appendix A.2.6.11).

```
dbInputWhereClause = origin.orid = 15433650
```

This property operates identically to the where portion of a SQL statement, with the SQL query being created in the background by PCalc based on the information input by the user. For instance, in this example dbInputWhereClause is set such that only data corresponding to the origin with orid 15433650. In SQL, these properties are equivalent to the SQL statement:

```
select origin.*, assoc.*, arrival.*, site.*
from account_name.origin origin, account_name.assoc assoc, account_name.arrival
arrival, account_name.site site
where origin.orid=assoc.orid
and assoc.arid=arrival.arid
and arrival.sta=site.sta
and arrival.jdate >= site.ondate
and (site.offdate = -1 or arrival.jdate <= site.offdate)
and site.sta=affiliation.sta
and origin.orid = 15433650
```

where the dbInputWhereClause entry is shown in italics. A dbInputWhereClause must be specified in the properties file.

To further refine the SQL statement, the user can also specify an optional dbInputAssocClause (Appendix A.2.6.12). In this example, the dbInputAssocClause is used to select certain phases.

```
dbInputAssocClause = assoc.phase in ('P', 'S', 'Pn', 'Sn', 'Pg','Lg')
```

This clause is further appended to the SQL statement such that it is executed in addition to the dbInputWhereClause. In this example, the equivalent SQL statement is:

```
select origin.*, assoc.*, arrival.*, site.*
from account_name.origin origin, account_name.assoc assoc, account_name.arrival
arrival, account_name.site site
where origin.orid=assoc.orid
and assoc.arid=arrival.arid
and arrival.sta=site.sta
and arrival.jdate >= site.ondate
and (site.offdate = -1 or arrival.jdate <= site.offdate)
and site.sta=affiliation.sta
and origin.orid = 15433650
and assoc.phase in ('P','S','Pn','Sn','Pg','Lg')
```

where the dbInputWhereClause and dbInputAssocClause entries are shown in italics.

The next few properties define how to output the PCalc results. When using a database as input, the type of output must be specified using the outputType property (Appendix A.2.8.7).

```
outputType = database
```

When using PCalc on databases, outputType must be defined and must be set to either file or database. If outputType is set to database, as in this example, the user must specify the properties dbOutputInstance, dbOutputUserName, dbOutputPassword to connect to the database the output will be written to. If the output database is the same as the input database, the user can use PCalc's substitution capability (Section 3.1.1) to copy the information from the equivalent input properties as shown below.

```
dbOutputInstance = <property:dbInputInstance>
dbOutputUserName = <property:dbInputUserName>
dbOutputPassword = <property:dbInputPassword>
```

Additionally, the tables to output need to be defined. In this example, only an assoc table called pcalc_assoc is output using the dbOutputAssocTable property (Appendix A.2.6.13).

```
dbOutputAssocTable = pcalc_assoc
```

The final database properties in this example are used to define how to create the table. The property dbOutputAutoTableCreation (Appendix A.2.6.14) will create the output database table automatically when set to true as in this example. By default, this property is set to false.

```
dbOutputAutoTableCreation = true
```

dbOutputPromptBeforeTruncate (Appendix A.2.6.16) is used in conjunction with the property dbOutputTruncateTables (Appendix A.2.6.15). When set to true, dbOutputTruncateTables will truncate, i.e., delete, the data within previously existing tables, which are then overwritten by the data from the most recent PCalc run. If dbOutputPromptBeforeTruncate is also set to true, the user will be prompted to confirm that the table is to be truncated. In this example, dbOutputTruncateTables is true while dbOutputPromptBeforeTruncate is false. Note that dbOutputTruncateTables is false by default while dbOutputPromptBeforeTruncate is true by default.

Here these are defined as:

```
dbOutputPromptBeforeTruncate = false
dbOutputTruncateTables = true
```

Thus, the tables will be truncated and overwritten without a warning to the user.

With all properties set, the properties file in **Error! Reference source not found.** can be run as **p calc pcalc_predictions_db.properties**. The run should result in a log file (pcalc_log.txt) and an output CSS3.0 database table (pcalc_assoc) in the specified output database account. As results will vary depending on the input database tables, example output is not shown here.

3.2.5. Example 8 – PCalc Raypath Generation Using Great Circles

To execute Example 8, cd into ~/Salsa3DSoftware/Examples/PCalc/raypaths_greatcircle. In this directory, the properties file **pcalc_raypaths_greatcircle.properties** (Figure 9) is provided.

```
#============================================================
#
# Property file for application PCalc v. 3.0
#
#============================================================
application = predictions

workDir= .

#============================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#============================================================
# predictors may include any combination of (tauptoolkit, bender, slbm)
predictors = bender

benderModel= <property:workDir>/../../salsa3d_model
#============================================================
#
# INPUT PARAMETERS: GENERAL
#
#============================================================
#inputType must be one of [file | greatcircle | grid]
inputType=greatcircle
#============================================================
#
# INPUT PARAMETERS: GREATCIRCLE
#
#============================================================
# station name and jdate of the 'arrivals' are optional but if specified
# then bender will use the information to include tt_site_correction
sta=ARCES
jdate=2011001
phase=P
# lat, lon, elev of station are required.
site=10, 0, .2
# latitude and longitude of start of greatcircle path, in degrees
gcStart=0 -90
# latitude and longitude of end of greatcircle path, in degrees
#gcEnd = 0 30
# specify the distance and azimuth to move, in degrees relative to gcStart
gcDistance=180
gcAzimuth=90
# maximum spacing of points along greatcircle path, in degrees.
# Actual spacing will generally be smaller so that spacing is even.
#gcSpacing = 2
# number of evenly points along greatcircle path.
gcNpoints=19
gcOnCenters=false
# output position parameters.
# any subset of [x, y, z, latitude, longitude, distance, depth]
gcPositionParameters=latitude longitude distance depth
rayPathNodeSpacing=10
#============================================================
#
# INPUT PARAMETERS: DEPTHS
#
#============================================================
# all depths are specified in km.  The various methods of specifying
# depth are searched until one is found that is valid.
# Search order is depths, depthRange, depthLevels, maxDepthSpacing.
depthSpecificationMethod=maxDepthSpacing

# specify that there should be a depth at the top and bottom of each
# layer and additional depths in the interior of each layer such that
# the maximum spacing of depths is no greater than maxDepthSpacing.
maxDepthSpacing=100
# if maxDepthSpacing was specified, then maxDepth can also be used to
# specify some maximum depth for the bottom of the grid.
# Default value is center of the earth.
maxDepth=500
#============================================================
```

Figure 9. PCalc Prediction Properties File pcalc_raypaths_greatcircle.properties.

```
#============================================================
#
# OUTPUT PARAMETERS
#
#============================================================
outputType=file
# if outputFile is specified then output is written to the specified file
# otherwise, output is written to stdout
outputFile=<property:workDir>/pcalc_raypaths_greatcircle_output.dat
#optional log file
logFile=<property:workDir>/pcalc_log.txt
# if terminalOutput is true then log information is written to stdout
# otherwise application is silent
terminalOutput=true
# if outputHeader is true then a header describing each column is output
# to the top of the output file.
outputHeader=true

# character to use to as field separator in the output file
# options are tab, space, comma
separator=space
# the following list of ouput attributes will be computed
# and appended to the end of each input record.
outputAttributes=ray_path
```

Figure 8 (cont). PCalc Prediction Properties File pcalc_raypaths_greatcircle.properties.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 3.1.1 and 3.2.2.

Like Example 5 (Section 3.2.2), this properties file is used to generate predictions for locations specified by a great circle. Unlike Example 5, the returned predictions are not travel times but rather raypath geometries. To return raypath geometries, the property outputAttributes (Appendix A.2.8.6, Section 3.1.1) is now set to ray_path.

```
outputAttribute = ray_path
```

These output raypath geometries represent rays calculated by the predictor (here Bender) for the specified phase (here P) that travel through the input model (here SALSA3Dv2.1.geotess) from the specified origins (here defined by a great circle) to the specified receiver (here ARCES).

Note that while both Example 8 uses a great circle to provide input locations to PCalc, similar to Example 5, the great circle in Example 8 is defined using properties not shown previously. These properties are gcNpoints (Appendix A.2.4.5), which defines the number of points that will be placed along the great circle path, and gcOnCenters (Appendix A.2.4.7), which causes the points along the great circle to reside at the centers of line segments rather than span the length of the great circle if true. If false, the first and last points coincide with the beginning and end of the great circle; gcOnCenters is false by default.

Here these are defined as:

```
gcNpoints = 19
gcOnCenters = false
```

Thus, the great circle will consist of 19 points and the first and last points will coincide with the beginning and end of the great circle.

Finally, to generate raypaths, the point spacing along the computed raypath in km must be specified. This action is done by defining the property rayPathNodeSpacing (Appendix A.2.12.1). In this example, points along the raypath are set to occur every 10 km.

```
rayPathNodeSpacing = 10
```

With all properties set, the properties file in **Error! Reference source not found.** can be run as **p calc pcalc_raypaths_greatcircle.properties**. The run should result in a log file (pcalc_log.txt) and an output file (pcalc_raypaths_greatcircle_output.dat). If successful, the output file should contain 251,194 rows and four columns (latitude in deg, longitude in deg, distance in deg, depth in km) separated by spaces. One raypath consists of all rows bracketed between > symbols at the beginning and end of the raypath. For instance, the first four rows and final row of one raypath generated in this example are shown below.

```
latitude longitude distance depth
>
0.000000 -90.000000 0.000000 1.868934
0.000012 -89.999932 0.000069 1.969040
0.000012 -89.999932 0.000069 1.969040
0.000976 -89.994679 0.005409 4.669786
. . .          …          …          …
10.000000 0.000000 90.000000 -0.200000
>
```

3.3. PCalc LibCorr3d Surface Generation

LibCorr3d surfaces are special GeoTess models containing two attributes, tt_delta_ak135 and tt_model_uncertainty. tt_delta_ak135 represents the difference between travel times predicted using a 3D GeoTess slowness model and travel times predicted with the standard AK135 model. tt_model_uncertainty represents either 1D distance dependent travel time uncertainty or path dependent travel time uncertainty computed from the 3D model covariance matrix that results from tomographic inversion. Details on LibCorr3d are available in Ballard et al. (2016a) [2].

To perform LibCorr3d surface generation, the special salsa3d input folder defined in Section 3, bullet 4 is required. Due to the size of this folder (~100-200 GB), it is only available upon request. Send requests to acconle@sandia.gov.

In the following three sections, examples of LibCorr3d surface generation with different grid types (custom grid, common grid, rotated common grid) will be explored. Note that the surfaces generated using a common grid or rotated common grid are not compatible with legacy C++ LibCorr3d code.

3.3.1. Example 9 – LibCorr3d Surface Generation with a Custom Grid

To execute Example 9, cd into ~/Salsa3DSoftware/Examples/PCalc/libcorr3d_examples/custom_grids. In this directory, the

properties file **pcalc.properties** (Figure 10) is provided along with a text file containing site information for stations ASAR and MJAR (**siteFile.txt**).

```
#============================================================
#
# Property file for computing LibCorr3D models using PCalc
#
#============================================================

application = libcorr3d

# parallelMode can be either sequential or concurrent.
parallelMode = concurrent

# specify max number of threads this process is allowed use when parallelMode is concurrent.
# Default is all.
# maxProcessors = 32

# workDir is  not really a pcalc property.  It is specified for convenience and
# referenced elsewhere in this file.

workDir = .

logFile = <property:workDir>/logs/<property:sta>_log.txt

#============================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#============================================================

# predictors may include any combination of (lookup2d, bender, and slbm)
predictors = bender

# can specify a specific geotess model file, or a salsa3d model directory
benderModel = <property:workDir>/../../../salsa3d_model_directory

# benderUncertaintyType can be [ DistanceDependent | PathDependent ]
benderUncertaintyType = PathDependent

# when benderUncertaintyType = PathDependent, you must specify a directory that PCalc can
# write lots of temporary files to when computing path dependent uncertainty.
# The directory will be created if it does not exist but will not be deleted at the end of the run.
# Information written in the directory will be deleted at the end of the run but may get left if the run
# fails or is aborted.  Feel free to delete or empty this directory anytime that PCalc is not using it.
benderUncertaintyWorkDir = <property:workDir>/BenderUncertaintyWorkDir

allowCMBDiffraction = true

fixAnomaliesThreshold = 15

#============================================================
#
# PHASE SPECIFICATION
#
#============================================================

phase = P
supportedPhases = P,Pn
```

Figure 10. PCalc LibCorr3d Custom Grid Model Generation Properties File pcalc.properties.

```
#=============================================================
#
# GEOTESS GRID SPECIFICATION
#
#=============================================================


# these properties will generate a separate, custom grid for each station.
# Because property geotessOutputGrid is not specified, the station-specific grids
# will be stored in the output libcorr3d model files.

# rotate the grid so that grid vertex 0 is located at the location of the station
geotessRotateGridToStation = true

# all grid points past 'geotessActiveNodeRadius' (degrees) from the station will be populated with NaN.
geotessActiveNodeRadius = 100

# Refine the grid in a spherical cap around the station.
# The parameters of the spherical_cap definitions are:
#     1 - latitude  of center of the cap, in degrees.  (<site.lat> is replaced with latitude of current site)
#     2 - longitude of center of the cap, in degrees.  (<site.lon> is replaced with longitude of current site)
#     3 - radius of the spherical cap in degrees.
#     4 - tessellation index.  Always 0 for pcalc applications.
#     5 - grid resolution inside the cap, in degrees.
geotessPolygons = spherical_cap, <site.lat>, <site.lon>, <property:geotessActiveNodeRadius>, 0, 1; \
                  spherical_cap, <site.lat>, <site.lon>, 25, 0, 0.5

#=============================================================
#
# DEPTH SPECIFICATION
#
#=============================================================

# 2 km until 10, then 5 km until 50, then 10 until 100, 20 until 200, 50 until 400, then 100.
geotessDepths = -6, -4, -2, -1, 0, 1, 2, 4, 6, 8, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300
, 350, 400, 500, 600, 700

# When property geotessDepths is used to specify a list of depths, and a seismicity depth model
# is being used to constrain model depths (true by default), the following procedure is implemented.
# At each grid vertex, the minimum and maximum depths in the seismicity depth model at the geographic
# location of the grid vertex are identified.  The list of depths specified by property geotessDepths
# is truncated to ignore depths above the minimum depth and depths below the maximum depth.  Then either
# additional depths are added at the minimum and maximum depths or the first and last depths in the
# modified list of depths are adjusted such that the minimum and maximum depths in the depth list correspond
# to the minimum and maximum depth retrieved from the seismicity depth model.

#=============================================================
#
# SITE SPECIFICATION
#
#=============================================================

# There are two ways to specify sites.  They can be specified in a separte file  or they can be
# specified in this pcalc properties file.  See appendix in the PCalc User's Manual for discussion.

siteFile = <property:workDir>/siteFile.txt
```

Figure 10 (cont). PCalc LibCorr3d Custom Grid Model Generation Properties File pcalc.properties.

```
#================================================================
#
# OUTPUT PARAMETERS: LIBCORR3D OUTPUT
#
#================================================================

# the default outputFile is
# <property:benderModel>/libcorr3d_delta_ak135/<property:sta>_<property:phase>_TT.libcorr3d
# but a different destination can be specified if desired.
outputFile = <property:workDir>/models/<property:sta>_<property:phase>_TT.libcorr3d

# (optional) if vtkFile is specified, then vtk files will be produced which allow display
# of model geometry and data using ParaView
vtkFile = <property:workDir>/vtk/<property:sta>_<property:phase>_TT.vtk

# If overwriteExistingOutputFile is true, and the outputFile already exists, then the existing file is
# overwritten.  If overwriteExistingOutputFile is false, and the outputFile already exists, then the
# surface is not computed and the existing file is not overwritten.
overwriteExistingOutputFile = false

# include these two file format properties only if the resulting libcorr3d models will be read
# by older versions of GeoTess and LibCorr3D.  Otherwise, comment them out.
geotessFileFormat = 2
libcorr3dFileFormat = 1
```

Figure 10 (cont). PCalc LibCorr3d Custom Grid Model Generation Properties File pcalc.properties.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 3.1.1 and 3.2.1.

In this example, each LibCorr3d surface calculated for each station will have a unique grid contained within the output surface GeoTess model. To begin, the application property (Appendix A.2.1.1, Section 3.1.1) is now set to libcorr3d and predictors is set to bender.

```
application = libcorr3d
predictors = bender
```

These properties are required to have the above values to perform LibCorr3d surface generation.

Next, the property benderModel must point to the special salsa3d input folder defined in Section 3, bullet 4.

```
benderModel = <property:workDir>/../../../salsa3d_model_directory
```

As LibCorr3d surfaces require travel time uncertainties to be calculated, a new property, benderUncertaintyType property (Appendix A.2.9.16), must be defined. benderUncertaintyType sets the type of travel time uncertainty desired.

If set to DistanceDependent, the user must provide files containing 1D distance dependent uncertainties to PCalc. These can be provided either using the special salsa3d input folder defined in in Section 3, bullet 4 or by using the properties benderUncertaintyDirectory (Appendix A.2.9.15) and benderUncertaintyModel (Appendix A.2.9.16) to define the path to the desired distance dependent uncertainty files.

Alternatively, if benderUncertaintyType is set to PathDependent, the velocity model's full covariance matrix is used to calculate path dependent uncertainties. In this case, benderModel must point to the special salsa3d input folder, which contains the needed covariance matrix. Further, the property benderUncertaintyWorkDir must be defined when benderUncertaintyType is PathDependent. This property sets the path where temporary files can be stored by PCalc as path dependent uncertainties are calculated. This directory can be removed or emptied anytime it is not in use by PCalc.

Next, the user can specify whether to allow for rays impinging on the core-mantle boundary (CMB) to diffract using the allowCMBDiffraction property (Appendix A.2.9.19). By default, this property is set to false such that rays impinging on the CMB are invalid. In this example, it is set to true.

```
allowCMBDiffraction = true
```

Bender will in very rare instances (~once in 1 million calculations) return an inaccurate traveltime that differs significantly from traveltimes calculated for neighboring nodes in the LibCorr3d model. To address these anomalous traveltimes, the user can specify a threshold at which to fix the anomalies using fixAnomaliesThreshold (Appendix A.2.10.7). Here it is set to:

```
fixAnomaliesThreshold = 15
```

When fixAnomaliesThreshold is set, PCalc will compute the difference between a traveltime value at a node and the average traveltime value of neighboring nodes. If the difference exceeds the threshold specified (15 sec in this example), the anomalous traveltime value is replaced by the average traveltime value of neighboring nodes.

When defining which sites, i.e., stations, to generate a LibCorr3d surface for, the user can either define site in the same way as prior prediction examples (see Sections 3.2.1,3.2.2, 3.2.3, 3.2.4) or they can input a list of sites within a text file using the siteFile property (Appendix A.2.2.4). In this example, the siteFile property is used and the path points to the necessary text file, siteFile.txt.

```
siteFile = <property:workDir>/siteFile.txt
```

See Appendix B for more details on the usage of site files.

When defining phases, in addition to the phase property (Appendix A.2.2.5) used in previous examples (see Sections 3.2.1,3.2.2, 3.2.3, 3.2.4) the user must also define the supportedPhases property. supportedPhases defines which phases the LibCorr3d surface can be used for. For instance, here supportedPhases is defined as:

```
supportedPhases = P,Pn
```

Thus, the LibCorr3d surfaces generated by this example can be used to retrieve travel times and travel time uncertainties for phases P and Pn.

Because a LibCorr3d surface is contained in a GeoTess model, the model must be built and populated with GeoTessBuilder. GeoTessBuilder and its properties are described extensively in the

GeoTess user's manual. Thus, the GeoTess properties in this example will only be briefly defined; the user is referred to the GeoTess manual for more detail.

GeoTess is used to perform grid specification and depth specification. The properties to specify the grid are geotessRotateGridToStation, geotessActiveNodeRadius, and geotessPolygons. geotessRotateGridToStation, when set to true, will rotate the grid such that vertex 0 in the grid scheme is located at the location of the specified station. In this example, this property is set to true.

```
geotessRotateGridToStation = true
```

geotessActiveNodeRadius defines which nodes in the grid are active, i.e., populated with data, by distance in degrees from the station. Any nodes beyond the geotessActiveNodeRadius value are populated with NaN. Here, the geotessActiveNodeRadius is defined as:

```
geotessActiveNodeRadius = 100
```

Thus, in this example, nodes located ≥ 100 degrees from the specified station (either ASAR or MKAR in this case) are populated by NaN values.

geotessPolygons is used to refine the grid in a "spherical cap" around the station. This property allows the user to make the grid resolution around the station up to a user-specified radius more refined than in areas whose distance exceeds the user-specified radius. Multiple spherical caps up to different radii can be made using this property. Here, geotessPolygons is defined as:

```
geotessPolygons = spherical_cap, <site.lat>, <site.lon>,
<property:geotessActiveNodeRadius>, 0, 1; \
spherical_cap, <site.lat>, <site.lon>, 25, 0, 0.5
```

where two spherical caps are made. The first row defines a cap centered at the location of the station that extends up to 100 degrees and has a 1-degree grid resolution. The second row defines a cap centered at the location of the station that extends up to 25 degrees and has a 0.5-degree resolution. Thus, the grid made in this example will be 0.5 degrees resolution up to 25 degrees away from the station and will be 1 degree resolution from 25 degrees to 100 degrees. Beyond 100 degrees, the grid will have an 64-degree resolution.

The depth specification property used in this example is geotessDepths. geotessDepths specifies a list of depths at which to perform travel time and uncertainty calculations. Note that by default, PCalc uses the seismicity depth model described in Section 1.3 to constrain allowed depths. The constraint is implemented by truncating the list of depths in geotessDepths such that depths above the minimum depth and below the maximum depth are ignored. Then either 1) additional depths are added to geotessDepths such that the list includes the seismicity depth model's minimum and maximum depths or 2) the first and last depths in geotessDepths are adjusted such that the minimum and maximum depths in geotessDepths correspond to the seismicity depth model's minimum and maximum depths. Option 2 is applied when the first and/or last depths differ from the seismicity depth model minimum and/or maximum depths by ≤ 10 %.

Here, geotessDepths is defined as:

```
geotessDepths = -6, -4, -2, -1, 0, 1, 2, 4, 6, 8, 10, 15, 20, 25, 30, 35, 40, 50, 60,
70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400, 500, 600, 700
```

Thus, predictions will be made using "events" at a 2 km spacing up to 10 km depth, a 5 km spacing up to 50 km depth, a 10 km spacing up to 100 km depth, a 20 km spacing up to 200 km depth, a 50 km spacing up to 400 km depth, and a 100 km spacing up to the final 700 km depth. Because the minimum depth is outside the minimum depth range, it will be adjusted to correspond to -5.4 km depth, the minimum depth of the seismicity depth model. The maximum depth falls within the allowed maximum depth range of the seismicity depth model, and thus is not modified.

Finally, the following output properties are defined. First, the outputFile property (Appendix A.2.8.1) is provided a desired output path and file name. In this example, outputFile is set to:

```
outputFile = <property:workDir>/models/<property:sta>_<property:phase>_TT.libcorr3d
```

This outputFile path will result in a GeoTess file being saved under a folder called models in the current directory. If the models folder does not already exist, it will be created by PCalc. The saved GeoTess file will be named sta_phase_TT.libcorr3d (e.g., MJAR_P_TT.libcorr3d), where the libcorr3d extension indicates that this GeoTess model is a LibCorr3d surface.

Optionally, the user can also output a ParaView vtk file using the property vtkFile; see the GeoTess user's manual for details on the vtkFile property. The output vtk file provides a visual representation of the generated LibCorr3d surface that can be viewed in ParaView; refer to ParaView documentation for information on usage of vtk files. Here, vtkFile is set to:

```
vtkFile = <property:workDir>/vtk/<property:sta>_<property:phase>_TT.vtk
```

This vtkFile path will save a file named sta_phase_TT.vtk, e.g., MKAR_P_TT.vtk, under folder vtk, similar to the property outputFile.

The property overwriteExistingOutputFile (Appendix A.2.8.2) will allow output files to be overwritten if they already exist when set to true. When set to false, existing files will not be overwritten; if file names are not changed in this case, nothing is saved. By default, this property is set to true. In this example, overwriteExistingOutputFile is set to false.

```
overwriteExistingOutputFile = false
```

Finally, the two properties geotessFileFormat (Appendix A.2.8.4) and libcorr3dFileFormat (Appendix A.2.8.5) are used together to make the output Libcorr3D surfaces from this example compatible with older versions of GeoTess and LibCorr3d code. This action is done by setting geotessFileFormat = 2 and libcorr3dFileFormat = 1.

```
geotessFileFormat = 2
libcorr3dFileFormat = 1
```

With all properties set, the properties file in Figure 10 can be run as **pcalc pcalc.properties**. The run should result in a logs folder containing two log files (ASAR_log.txt, MKAR_log.txt), a models folder containing two LibCorr3d surfaces (ASAR_P_TT.libcorr3d, MKAR_P_TT.libcorr3d), and a vtk folder containing two vtk files (ASAR_P_TT.vtk, MKAR_P_TT.vtk). Depending on the number of processors available, LibCorr3d surface generation may take up to an hour or more.

3.3.2. Example 10 – LibCorr3d Surface Generation with a Common Grid

To execute Example 10, cd into ~/Salsa3DSoftware/Examples/PCalc/libcorr3d_examples/common_grid. In this directory, the properties files **geotessbuilder.properties** (Figure 11) and **pcalc.properties** (Figure 12) are provided along with a text file containing site information for stations ASAR and MKAR (**siteFile.txt**).

```
# specify GeoTessBuilder grid construction mode.
gridConstructionMode = scratch

# number of multi-level tessellations to build
nTessellations = 1

# grid resolution in degrees
baseEdgeLengths = 1

# file to receive the GeoTessGrid definition
outputGridFile = ./models/_grid_01000.geotess

# file to receive the vtk  file for visualization with ParaView. (Optional)
vtkFile = ./vtk/_grid_01000.vtk
#vtkRobinsonFile = ./_grid_01000.vtk
#kmlFile = ./_grid_01000.kml
#kmzFile = ./_grid_01000.kmz
#gmtFile = ./_grid_01000.gmt
```

Figure 11. PCalc LibCorr3d Common Grid Model Generation Properties File geotessbuilder.properties.

In this example, an external grid is made to be shared between all generated LibCorr3d surfaces. Now, each LibCorr3d surface will take in this external grid as input in the properties file.

The properties to make this grid listed in geotessbuilder.properties (Figure 11) are described extensively in the GeoTess user's manual. Thus, the GeoTess properties in this example will only be briefly defined; the user is referred to the GeoTess manual for more detail.

In this example, the user specifies that the GeoTess grid is to be built from scratch using the property gridConstructionMode.

```
gridConstructionMode = scratch
```

Then the number of multi-level tessellations to build is defined using the property nTessellations. Here it is set to 1.

```
nTessellations = 1
```

The grid resolution in degrees is then set to 1 degree throughout the model using the property baseEdgeLengths.

```
baseEdgeLengths = 1
```

The grid will be output to the path specified in outputGridFile.

```
outputGridFile = ./models/_grid_01000.geotess
```

In this example, a models folder will be generated in the current directory and the grid will be saved as a GeoTess file, _grid_01000.geotess, where 01000 indicates the grid resolution in millidegrees, here 1000 millidegrees.

```
outputGridFile = ./models/_grid_01000.geotess
```

Finally, optional files can be output. These include the vtk file discussed in the previous section as well as a vtk file in a Robinson projection (vtkRobinsonFile), Google Earth kml and kmz files (kmlFile, kmzFile), and a gmt file (gmtFile). In this example, a vtk file is output in the folder vtk. To generate the other file types, uncomment the corresponding properties. As these are GeoTess properties, refer to the GeoTess user's manual for details.

```
vtkFile = ./vtk/_grid_01000.vtk
#vtkRobinsonFile = ./_grid_01000.vtk
#kmlFile = ./_grid_01000.kml
#kmzFile = ./_grid_01000.kmz
#vtkRobinsonFile = ./_grid_01000.gmt
```

With all properties set, the properties file in Figure 11 can be run as **geotessbuilder geotessbuilder.properties**. The run should result in a models folder containing a 1-degree grid in GeoTess format and a vtk folder containing a vtk file of the grid that can be read in ParaView. If the user also uncommented the other file properties, a 1) vtk file of the grid in a Robinson projection (_grid_01000.vtk) along with its coastlines centered at 12 degrees longitude (map_coastlines_centerLon_12.vtk) and a map edge (map_edge.vtk) also saved as vtks, 2) a kml file (_grid_01000.kml), 3) a kmz file (_grid_01000.kmz), and 4) a gmt file (_grid_01000.gmt) are also output in the current directory.

Once the grid file _grid_01000.geotess is created, the LibCorr3d surfaces can be generated with the properties file pcalc.properties (Figure 12).

As several properties in this properties file have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 3.1.1, 3.2.1, and 3.3.1.

```
#============================================================
#
# Property file for computing LibCorr3D models using PCalc
#
#============================================================

application = libcorr3d

# parallelMode can be either sequential or concurrent.
parallelMode = concurrent

# specify max number of threads this process is allowed use when parallelMode is concurrent.
# Default is all.
# maxProcessors = 32

# workDir is  not really pcalc property.  It is specified for convenience and
# referenced elsewhere in this file.

workDir = .

logFile = <property:workDir>/logs/<property:sta>_log.txt

#============================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#============================================================

# predictors may include any combination of (lookup2d, bender, and slbm)
predictors = bender

# can specify a specific geotess model file, or a salsa3d model directory
benderModel = <property:workDir>/../../../salsa3d_model_directory

# benderUncertaintyType can be [ DistanceDependent | PathDependent ]
benderUncertaintyType = PathDependent

# when benderUncertaintyType = PathDependent, you must specify a directory that PCalc can
# write lots of temporary files to when computing path dependent uncertainty.
# The directory will be created if it does not exist but will not be deleted at the end of the run.
# Information written in the directory will be deleted at the end of the run but may get left if the run
# fails or is aborted.  Feel free to delete or empty this directory anytime that PCalc is not using it.
benderUncertaintyWorkDir = <property:workDir>/BenderUncertaintyWorkDir

allowCMBDiffraction = true

fixAnomaliesThreshold = 15

#============================================================
#
# PHASE SPECIFICATION
#
#============================================================

phase = P
supportedPhases = P,Pn

#============================================================
#
# GRID SPECIFICATION
#
#============================================================

# These properties specify both an input grid and an output grid.  This means that grids will
# not be stored in each model output file.  All the models will share access to the same externally
# stored grid.

geotessInputGridFile = <property:workDir>/models/_grid_01000.geotess
geotessOutputGridFile = <property:geotessInputGridFile>

# all grid points past 'geotessActiveNodeRadius' (degrees) from the station will be populated with NaN.
geotessActiveNodeRadius = 100
```

Figure 12. PCalc LibCorr3d Common Grid Model Generation Properties File pcalc.properties.

```
#============================================================
#
# DEPTH SPECIFICATION
#
#============================================================

# 2 km until 10, then 5 km until 50, then 10 until 100, 20 until 200, 50 until 400, then 100.
geotessDepths = -6, -4, -2, -1, 0, 1, 2, 4, 6, 8, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400
, 500, 600, 700

# When property geotessDepths is used to specify a list of depths, and a seismicity depth model
# is being used to constrain model depths (true by default), the following procedure is implemented.
# At each grid vertex, the minimum and maximum depths in the seismicity depth model at the geographic
# location of the grid vertex are identified.  The list of depths specified by property geotessDepths
# is truncated to ignore depths above the minimum depth and depths below the maximum depth.  Then either
# additional depths are added at the minimum and maximum depths or the first and last depths in the
# modified list of depths are adjusted such that the minimum and maximum depths in the depth list correspond
# to the minimum and maximum depth retrieved from the seismicity depth model.

#============================================================
#
# SITE SPECIFICATION
#
#============================================================

# There are two ways to specify sites.  They can be specified in a separte file  or they can be
# specified in this pcalc properties file.  See appendix in the PCalc User's Manual for discussion.

siteFile = <property:workDir>/siteFile.txt

#============================================================
#
# OUTPUT PARAMETERS: LIBCORR3D OUTPUT
#
#============================================================

# the default outputFile is
# <property:benderModel>/libcorr3d_delta_ak135/<property:sta>_<property:phase>_TT.libcorr3d
# but a different destination can be specified if desired.
outputFile = <property:workDir>/models/<property:sta>_<property:phase>_TT.libcorr3d

# (optional) if vtkFile is specified, then vtk files will be produced which allow display
# of model geometry and data using ParaView
vtkFile = <property:workDir>/vtk/<property:sta>_<property:phase>_TT.vtk

# If overwriteExistingOutputFile is true, and the outputFile already exists, then the existing file is
# overwritten.  If overwriteExistingOutputFile is false, and the outputFile already exists, then the
# surface is not computed and the existing file is not overwritten.
overwriteExistingOutputFile = false

# include these two file format properties only if the resulting libcorr3d models will be read
# by older versions of GeoTess and LibCorr3D.  Otherwise, comment them out.
# geotessFileFormat = 2
# libcorr3dFileFormat = 1
```

Figure 12 (cont). PCalc LibCorr3d Common Grid Model Generation Properties File pcalc.properties.

This example is identical to the Example 9 except for the grid specification properties. The properties geotessInputGridFile and geotessOutputGridFile replace all grid specification properties in Example 9 (Figure 10) except for geotessActiveNodeRadius, which is again set to populate all nodes beyond 100 degrees with NaN values.

geotessInputGridFile is set to the path and filename of the grid file generated by geotessbuilder earlier in this section. In this example, it is set to:

```
geotessInputGridFile = <property:workDir>/models/_grid_01000.geotess
```

If desired, the grid file of the LibCorr3d surface(s) generated can also be output separately using the property geoTessOutputGridFile. Here, this property is set to:

```
geotessOutputGridFile = <property:geotessInputGridFile>
```

This setting will result in the input grid file being written out again in the same place; thus, its use is for demonstration purposes only. For more details on geotessInputGridFile and geotessOutputGridFile, refer to the GeoTess user's manual.

Finally, note that while the properties geotessFileFormat (Appendix A.2.8.4) and libcorr3dFileFormat (Appendix A.2.8.5) are included in this properties file, they are commented out. Uncomment these if desired.

With all properties set, the properties file in Figure 12 can be run as **pcalc pcalc.properties**. The run should result in two LibCorr3d surfaces (ASAR_P_TT.libcorr3d, MKAR_P_TT.libcorr3d) within the models folder generated during the creation of the external grid. As the geotessOutputGridFile property will result in the grid file being overwritten, there should still be only one grid file (_grid_01000.geotess) in the models folder, serving as the common grid to the two LibCorr3d surfaces. Depending on the number of processors available, LibCorr3d surface generation may take up to an hour or more.

3.3.3. Example 11 – LibCorr3d Surface Generation with a Rotated Common Grid

To execute Example 11, cd into ~/Salsa3DSoftware/Examples/PCalc/libcorr3d_examples/common_grid_rotated. In this directory, the properties files **geotessbuilder.properties** (Figure 13) and **pcalc.properties** (Figure 14) are provided along with a text file containing site information for stations ASAR and MKAR (**siteFile.txt**).

```
# specify GeoTessBuilder grid construction mode.
gridConstructionMode = scratch

# number of multi-level tessellations to build
nTessellations = 1

# grid resolution in degrees
baseEdgeLengths = 64

# Refine the grid in a spherical cap around the north pole.
# The parameters of the spherical_cap definitions are:
#    1 - latitude  of center of the cap, in degrees.
#    2 - longitude of center of the cap, in degrees.
#    3 - radius of the spherical cap in degrees.
#    4 - tessellation index.  Always 0 for pcalc applications.
#    5 - grid resolution inside the cap, in degrees.
polygons = spherical_cap, 90, 0,  100, 0, 1; spherical_cap, 90, 0, 25, 0, 0.5

# file to receive the GeoTessGrid definition
outputGridFile = ./models/_grid_<gridID>.geotess


# file to receive the vtk  file for visualization with ParaView. (Optional)
vtkFile = ./vtk/_grid_<gridID>.vtk
#vtkRobinsonFile = ./_grid_<gridID>.vtk
#kmlFile = ./_grid_<gridID>.kml
#kmzFile = ./_grid_<gridID>.kmz
#gmtFile = ./_grid_<gridID>.gmt
```

Figure 13. PCalc LibCorr3d Rotated Common Grid Model Generation Properties File geotessbuilder.properties.

As several properties in this properties file have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 3.1.1, 3.2.1, 3.3.1, and 3.3.2.

In this example, the generated grid will have a base grid resolution of 64°. The property polygons, effectively identical to the property geotessPolygons in Section 3.3.1, is used to create two spherical caps centered at 90° latitude and 0° longitude. The first cap has a grid resolution of 1° out to 100 degrees distance and the second cap has a grid resolution of 0.5° out to 25 degrees distance. Thus, this grid will have a resolution of 0.5° out to 25 degrees distance, 1° out to 100 degrees distance, and 64° beyond 100 degrees distance.

Note that the output grid file and optional files are now named after their grid ID as the naming scheme described in Example 10 does not apply to a variable grid. For instance, outputGridFile is now defined as:

```
outputGridFile = ./models/_grid_<gridID>.geotess
```

Otherwise, this properties file is identical to the previous example's grid properties file (Figure 11).

With all properties set, the properties file in Figure 13 can be run as **geotessbuilder geotessbuilder.properties**. The run should result in a models folder containing a variable

resolution grid in GeoTess format (here _grid_FA7CFC70F0A67B90E08662E9613176C7.geotess) and a vtk folder containing a vtk file (_grid_FA7CFC70F0A67B90E08662E9613176C7.vtk) of the grid that can be read in ParaView. If the user also uncommented the other file properties, a 1) vtk file of the grid in a Robinson projection along with its coastlines centered at 12 degrees longitude (map_coastlines_centerLon_12.vtk) and a map edge (map_edge.vtk) also saved as vtks, 2) a kml file, 3) a kmz file, and 4) a gmt file are also output in the current directory.

```
#============================================================
#
# Property file for computing LibCorr3D models using PCalc
#
#============================================================

application = libcorr3d

# parallelMode can be either sequential or concurrent.
parallelMode = concurrent

# specify max number of threads this process is allowed use when parallelMode is concurrent.
# Default is all.
# maxProcessors = 32

# workDir is  not really pcalc property.  It is specified for convenience and
# referenced elsewhere in this file.

workDir = .

logFile = <property:workDir>/logs/<property:sta>_log.txt

#============================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#============================================================

# predictors may include any combination of (lookup2d, bender, and slbm)
predictors = bender

# can specify a specific geotess model file, or a salsa3d model directory
benderModel = <property:workDir>/../../../salsa3d_model_directory

# benderUncertaintyType can be [ DistanceDependent | PathDependent ]
benderUncertaintyType = PathDependent

# when benderUncertaintyType = PathDependent, you must specify a directory that PCalc can
# write lots of temporary files to when computing path dependent uncertainty.
# The directory will be created if it does not exist but will not be deleted at the end of the run.
# Information written in the directory will be deleted at the end of the run but may get left if the run
# fails or is aborted.  Feel free to delete or empty this directory anytime that PCalc is not using it.
benderUncertaintyWorkDir = <property:workDir>/BenderUncertaintyWorkDir

allowCMBDiffraction = true

fixAnomaliesThreshold = 15

#============================================================
#
# PHASE SPECIFICATION
#
#============================================================

phase = P
supportedPhases = P,Pn

#============================================================
#
# GRID SPECIFICATION
#
#============================================================

# These properties specify both an input grid and an output grid.  This means that grids will
# not be stored in each model output file.  All the models will share access to the same externally
# stored grid.  Because geotessRotateGridToStation is true, rotations will be applied at run time
# that will mimic having the input grid rotated such that grid vertex 0 is located at station location.

geotessInputGridFile = <property:workDir>/models/_grid_FA7CFC70F0A67B90E08662E9613176C7.geotess
geotessOutputGridFile = <property:geotessInputGridFile>

# rotate the grid so that grid vertex 0 is located at the location of the station
geotessRotateGridToStation = true

# all grid points past 'geotessActiveNodeRadius' (degrees) from the station will be populated with NaN.
geotessActiveNodeRadius = 100
```

Figure 14. PCalc LibCorr3d Rotated Common Grid Model Generation Properties File pcalc.properties.

```
#=============================================================
#
# DEPTH SPECIFICATION
#
#=============================================================

# 2 km until 10, then 5 km until 50, then 10 until 100, 20 until 200, 50 until 400, then 100.
geotessDepths = -6, -4, -2, -1, 0, 1, 2, 4, 6, 8, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400
, 500, 600, 700

# When property geotessDepths is used to specify a list of depths, and a seismicity depth model
# is being used to constrain model depths (true by default), the following procedure is implemented.
# At each grid vertex, the minimum and maximum depths in the seismicity depth model at the geographic
# location of the grid vertex are identified.  The list of depths specified by property geotessDepths
# is truncated to ignore depths above the minimum depth and depths below the maximum depth.  Then either
# additional depths are added at the minimum and maximum depths or the first and last depths in the
# modified list of depths are adjusted such that the minimum and maximum depths in the depth list correspond
# to the minimum and maximum depth retrieved from the seismicity depth model.

#=============================================================
#
# SITE SPECIFICATION
#
#=============================================================

# There are two ways to specify sites.  They can be specified in a separte file  or they can be
# specified in this pcalc properties file.  See appendix in the PCalc User's Manual for discussion.

siteFile = <property:workDir>/siteFile.txt

#=============================================================
#
# OUTPUT PARAMETERS: LIBCORR3D OUTPUT
#
#=============================================================

# the default outputFile is
# <property:benderModel>/libcorr3d_delta_ak135/<property:sta>_<property:phase>_TT.libcorr3d
# but a different destination can be specified if desired.
outputFile = <property:workDir>/models/<property:sta>_<property:phase>_TT.libcorr3d

# (optional) if vtkFile is specified, then vtk files will be produced which allow display
# of model geometry and data using ParaView
vtkFile = <property:workDir>/vtk/<property:sta>_<property:phase>_TT.vtk

# If overwriteExistingOutputFile is true, and the outputFile already exists, then the existing file is
# overwritten.  If overwriteExistingOutputFile is false, and the outputFile already exists, then the
# surface is not computed and the existing file is not overwritten.
overwriteExistingOutputFile = false

# include these two file format properties only if the resulting libcorr3d models will be read
# by older versions of GeoTess and LibCorr3D.  Otherwise, comment them out.
# geotessFileFormat = 2
# libcorr3dFileFormat = 1
```

Figure 14 (cont). PCalc LibCorr3d Rotated Common Grid Model Generation Properties File pcalc.properties.

This example is identical to Example 10 except for the geotessInputGridFile path, which now points to the variable grid file generated earlier this section and the addition of the property geotessRotateGridToStation, detailed in Section 3.3.1, which is set to true.

```
geotessInputGridFile =
<property:workDir>/models/_grid_FA7CFC70F0A67B90E08662E9613176C7.geotess

geotessRotateGridToStation = true
```

As in Example 9 (Section 3.3.1), specifying geotessRotateGridToStation as true results in the grid being rotated such that the station the LibCorr3d model represents is located at vertex 0 on the grid.

Finally, note that while the properties geotessFileFormat (Appendix A.2.8.4) and libcorr3dFileFormat (Appendix A.2.8.5) are included in this properties file, they are commented out. This exclusion is because these properties are not available for LibCorr3d surfaces generated with a

rotated common grid. If uncommented, the property geotessFileFormat in particular will result in the exception "*Cannot write this model with file format 2 because eulerRotationAngles != null*".

With all properties set, the properties file in Figure 14 can be run as **pcalc pcalc.properties**. The run should result in two LibCorr3d surfaces (ASAR_P_TT.libcorr3d, MKAR_P_TT.libcorr3d) within the models folder generated during the creation of the external grid. As the geotessOutputGridFile property will result in the grid file being overwritten, there should still be only one grid file (_grid_FA7CFC70F0A67B90E08662E9613176C7.geotess) in the models folder, serving as the common grid to the two LibCorr3d surfaces. Depending on the number of processors available, LibCorr3d surface generation may take up to an hour or more.

4. SUMMARY

PCalc is a feature-rich software application that can compute travel-time predictions, ray path geometries, LibCorr3d surfaces, and perform model-queries in 3D models of Earth's velocity structure. PCalc has many useful features, especially for monitoring applications and when used with models specified in the GeoTess format:

- The ability to trace the phases P, PP, pP, PKPdf, PKPbc, and Pn through 3D models of Earth's compressional-wave velocity structure and S, SS, sS, and SKS through 3D models of Earth's shear-wave velocity structure.

- Computation of travel-time and travel-time uncertainty for the above phases.

- Seismic phase travel-times can be computed using any model stored in GeoTess format. In addition, travel-times can be computed using the AK135 model, which is stored within the PCalc Software.

- LibCorr3d surfaces can be generated on the fly with either 1D distant dependent or path dependent uncertainties.

- The ability to directly interact with CSS3.0 format data tables stored in an Oracle database for both input and output, including the insertion of newly computed origins into existing tables.

- Geographic positions at which models can be queried or travel-times computed can be specified in a variety of formats, including a list of locations contained in an ascii text file, a 2D grid of points distributed in depth along a great circle path or a 3D grid specified by regular vertices on the earth and in depth.

When PCalc is used in conjunction with the GeoTess and LocOO3D tools it becomes one part of a powerful toolkit for monitoring applications. The software and the examples specified in this manual can be downloaded through GitHub at:

https://github.com/sandialabs/Salsa3DSoftware

REFERENCES

[1] Ballard, S., J. R. Hipp and C. J. Young (2009) Efficient and accurate calculation of ray theory seismic travel time through variable resolution 3D Earth models. *Seismological Research Letters* **80** (6), 989-999.

[2] Ballard, S., J. Hipp, B. Kraus, A. Encarnacao and C. Young (2016a) GeoTess: A generalized Earth model software utility, *Seismological Research Letters* **87** (3), 719-725.

[3] Ballard, S., J. R. Hipp, M. L. Begnaud, C.J. Young, A. V. Encarnacao, E. P. Chael and W. S. Phillips (2016b) SALSA3D: A tomographic model of compressional wave slowness in the Earth's mantle for improved travel-time prediction and travel-time prediction uncertainty. *Bulletin of the Seismological Society of America* **106** (6), 2900-2916.

[4] Um, J. and C. Thurber (1987) A fast algorithm for two-point seismic ray tracing. *Bulletin of the seismological society of America* **77** (3), 972-986.

[5] Zhao, D., and J. Lei (2004). Seismic ray path variations in a 3D global velocity model, *Phys. Earth Planet. In.* 141, 153–166.

APPENDIX A. PCALC PROPERTIES

A.1. Setting Properties

The properties required by PCalc are preset to default values as the application is started. These defaults are given below in the property description section. Users may apply a different property value by using a property file (e.g., test.property). Only properties whose values differ from their defaults need to be listed in the properties file, since the defaults will be activated for any property not found in the file.

NOTES:

- PCalc properties are case sensitive.
- All properties in the properties file must contain an '=' character, separating the property name from the property value (e.g., inputType = grid). White space around the '=' sign is optional (ignored).
- Properties can be recursive. If a property value contains a string '<property:xyz>' then the phrase '<property:xyz>' is replaced with the value of property 'xyz'. For example, if the following records appear in the property file:

  testDirectory = /home/testDir
  io_log_file = <property:testDirectory>/log.txt

  then the actual value of property 'io_log_file' will be '/home/testdir/log.txt'.
- If a property value contains the string '<env:xyz>' then the phrase '<env:xyz>' is replaced with the value returned by System.getenv(xxx).

A.2. Property Descriptions

A.2.1. General

The general properties, with the exception of logfile, terminalOutput, and seismicity_depth_model, must be specified in all properties files. Otherwise, PCalc will throw an error.

A.2.1.1. application

<string> [Default = none] (model_query | predictions | libcorr3d)

Specifies the application that PCalc is to perform. The *model_query* application specifies that PCalc will extract requested data or metadata from an input model. The *predictions* application specifies that PCalc will generate travel-time or raypath predictions using input locations. Locations can be input as a file, grid, or great circle, respectively. Finally, the *libcorr3d* application specifies that PCalc will generate LibCorr3d surfaces using the provided velocity model input (see Section 3, bullet 4).

A.2.1.2. workDir

<string> [Default = none]

A path to a directory where PCalc can store output files. The directory will be generated automatically by PCalc if it does not exist.

A.2.1.3. logFile

<string> [Default = null: no text output]

Full path to log file. General information about the PCalc run is sent to this file. If property *terminalOutput* = true, the same information is sent to the screen.

A.2.1.4. terminalOutput

<boolean> [Default = true]

Echo general information about the PCalc run. This is the same information that is sent to the log file. If false, PCalc is silent.

A.2.1.5. seismicityDepthModel

<string> [Default = 'default'] ('null' | '`default`' | valid file name)

If string 'null' is specified, no seismicity depth model will be loaded. If string 'default' is specified or this property is not specified then the internal, default seismicity depth model described in Section 1.3 is loaded.

If a path and filename of a valid GeoTess file is specified, the model is loaded from the specified file.

A.2.2. Input

A.2.2.1. inputType

<string> [Default = none] (file | database | greatcircle | grid | geotess)

String indicating how the geometry of the predictions/model queries is to be specified. This document contains a section for each *inputType* that describes the properties that are pertinent to that *inputType*.

The remaining input properties listed below are used by multiple *inputTypes*.

A.2.2.2. sta

<String> [no Default]

The name of the station. If *sta* and *jdate* are supplied then Bender will include tt_site_corrections in total travel times, regardless of whether tt_site_corrections is one of the requested *outputAttributes* or not.

When one or more sites are specified with the site property, property sta is overridden with the site.sta as each site is processed.

A.2.2.3. site

<String> [no Default]

Used to specify the location of one or more sites. Several formats are supported:

1) sta, ondate, offdate, lat, lon, elevation, "staname" (in quotes), statype, refsta, dnorth, deast
2) sta, refsta, lat, lon, elevation
3) lat, lon, elevation (sta must be specified separately using the sta property, Appendix A.2.2.2)

Several sites can be specified with the site property, delimited by a semicolon. When more than one site is specified, they are processed one at a time.

Note that if *sta* is defined as a separate property (Appendix A.2.2.2), it will overwrite the *sta* value specified in the site property, i.e., *site.sta*.

A.2.2.4. siteFile

<String> [no Default]

Path to a file containing a list of sites, with each site consisting of one row. Site formats are identical to those needed for the site property (Appendix A.2.2.3). Unlike the site property, rows in the text file do not need to be delimited by a semicolon.

A.2.2.5. phase

<String> [no Default]

The seismic phase to use in PCalc calculations.

A.2.2.6. supportedPhases

<String> [no Default]

Comma-separated list of phases that are supported by the LibCorr3d surface. Only used when generating LibCorr3d surfaces.

A.2.2.7. jdate

<int> [2286324]

The jdate of predicted arrivals. If sta and jdate are supplied then Bender will include tt_site_corrections in total travel times, regardless of whether tt_site_corrections is one of the requested *outputAttributes* or not. tt_site_corrections are stored in GeoTess slowness models used by Bender to compute total travel times.

A.2.3. Input from File

If *inputType* = file, then this section defines properties that further define the file inputs.

A.2.3.1. inputFile

<String> [no Default]

The name of the file that is to be input.

A.2.3.2. inputHeaderRow

<boolean> [Default = false]

If *inputHeaderRow* = true, then the first line of the input file that is not blank and not a comment (lines that start with # are comments) will be interpreted as column headings that describe what each column contains.

If *inputHeaderRow* = false, then column heading information is obtained from property *inputAttributes*.

A.2.3.3. inputAttributes

<String>

Ignored if *inputHeaderRow* is true.

inputAttributes consists of a number of column headings separated by space(s). Each column heading may not contain any spaces and there must be exactly one for each column of input data.

When *application* = predictions

If predictions are to be computed, then the default value of *inputAttributes* is "sta jdate site_lat site_lon site_elev origin_lat origin_lon origin_depth phase".

When computing predictions, PCalc must be able to determine the origin_lat, origin_lon, origin_depth, site_lat, site_lon, site_elev, and the phase for each requested prediction. If sta and jdate columns are also supplied, then predictions will also include site corrections for predictors capable of supplying them.

At a minimum, *inputAttributes* must include origin_lat and origin_lon.

inputAttributes may also include origin_depth. If *inputAttributes* does not include origin_depth, then depth information must be supplied using the properties described in Appendix A.2.7.

inputAttributes may also include site_lat, site_lon and [site_elev | site_depth]. If *inputAttributes* does not include these quantities, then the site position information must be specified with property *site* described elsewhere and that station location will be used for all origin positions.

inputAttributes may also include 'phase'. If phase is not included in the *inputAttributes* then phase must be specified with property *phase* and the same phase will be used for all predictions.

inputAttributes may also include 'sta'. If 'sta' is not included in the *inputAttributes* then 'sta' may be specified with property *sta* and the same sta will be used for all predictions. If 'sta' is not specified, it defaults to '-'.

When *application* = model_query

If model queries are being requested, then the default value of *inputAttributes* is "longitude latitude depth".

When performing model queries, PCalc must be able to determine the latitude, longitude, and depth where the queries are to be performed.

At a minimum, *inputAttributes* must include latitude and longitude.

inputAttributes may also include depth. If *inputAttributes* does not include depth, then depth information must be supplied using the properties described in Appendix A.2.7.

A.2.4. Input from Great Circle

If *inputType* = greatcircle then this section defines properties that further define the great circle inputs.

This section describes how to define the 1D array of points distributed along a great circle path. As defined, the points have depth set to NaN (not-a-number). See Appendix A.2.7 for how to specify the depth(s) of the points along the great circle.

Property *gcStart* defines the position of one end of the great circle and is a required property. There are two ways to specify the other end of the great circle:

1. use *gcEnd* to specify the latitude and longitude of the other end,

2. use *gcDistance* and *gcAziumth* to specify the distance and azimuth to the other end of the great circle. *gcEnd* takes precedence if both are specified.

There are two ways to define the number of points that will be positioned along the great circle path:

1. use *gcNpoints* to explicitly define the number of equally spaced points,

2. use *gcSpacing* to specify the approximate spacing, in degrees, between adjacent points. In this instance, the actual spacing may be reduced somewhat from the specified value for an integer number of equally spaced points to span the length of the great circle. If both are specified, *gcSpacing* takes precedence.

A.2.4.1. gcStart

<2 doubles> [no Default]

The latitude in degrees and longitude in degrees, of the beginning of the great circle.

A.2.4.2. gcEnd

<2 doubles> [no Default]

The latitude in degrees and longitude in degrees of the end of the great circle. Takes precedence over *gcDistance*/*gcAzimuth* if both methods are specified.

A.2.4.3. gcDistance

<double> [no Default]

Epicentral distance in degrees from *gcStart* to the end of great circle. Ignored if *gcEnd* is specified.

A.2.4.4. gcAzimuth

<double> [no Default]

The azimuthal direction in degrees to move from *gcStart* in order to arrive at the end of the great circle. Ignored if *gcEnd* is specified.

A.2.4.5. gcNpoints

<int> [no Default]

The number of points that will be positioned along the great circle path. Ignored if *gcSpacing* is also specified.

A.2.4.6. gcSpacing

<double> [no Default]

The approximate spacing, in degrees, between adjacent points. The actual spacing may be reduced somewhat from the specified value in order for an integer number of equally spaced points to span the length of the great circle. Takes precedence over *gcNpoints* if both are specified.

A.2.4.7. gcOnCenters

<boolean> [false]

When *gcOnCenters* is true, the points along the great circle will reside at the centers of line segments that span the length of the great circle. When *gcOnCenters* is false, the first and last points will coincide with the beginning and end of the great circle.

A.2.4.8. gcPositionParameters

<String> [empty string] (any subset of [latitude, longitude, x, y, z, distance, depth])

Defines how the geometry of each point should be defined in the output file. Available parameters are:

- latitude – the latitude of the point in degrees.
- longitude – the longitude of the point in degrees.
- distance – the epicentral distance from the beginning of the great circle (*gcStart*) to the point, in degrees.
- depth – the depth of the point in km relative to sea level.
- radius – the radius of the point in km.
- x, y, z – Consider the plane of the great circle and consider each point to be a vector from the center of the earth to the point. The y direction is a unit vector from the center of the earth to a point halfway along the great circle path. The z direction is a unit vector that is normal to the plane of the great circle, pointing in the direction of the observer. X is a unit vector defined by y cross z. This coordinate system is useful for plotting points in a manner that shows the curvature of the surface of the earth and the various seismic discontinuities within it. z will always be zero in this application.

A.2.4.9. depthFast

<boolean> [true]

The order in which distance-depth information is written to output. When true, depths vary fastest. When false, distances vary fastest.

A.2.5. Input from Grid

If *inputType* = grid then this section defines properties that further define the grid input.
This section describes how to define the 2D array of grid points in map view. As defined, the points have depth set to NaN (not-a-number). See Appendix A.2.7 for how to specify the depth(s) of the points on the grid.

A.2.5.1. gridRangeLat

<2 doubles, 1 int> [no Default]

The minimum latitude, maximum latitude, and number of latitudes.

A.2.5.2. gridRangeLon

<2 doubles, 1 int> [no Default]

The minimum longitude, maximum longitude, and number of longitudes.

A.2.5.3. gridCenter

<2 doubles> [no Default]

Latitude and longitude, in degrees, of the center of the grid. Ignored if *gridRangeLat* and *gridRangeLon* are specified, required otherwise.

A.2.5.4. gridPole

<string> [no Default] (northPole, 90DegreesNorth, or 2 doubles)

The pole of rotation. If *gridPole* = northPole then the pole of rotation is the North Pole. If *gridPole* = 90DegreesNorth, then pole of rotation is the point found by moving 90 degrees away from *gridCenter* moving in a northerly direction. If *gridPole* = (2 doubles), then the doubles are interpreted to be the latitude and longitude of the pole of rotation, in degrees.

Ignored if *gridRangeLat* and *gridRangeLon* are specified; required if *gridCenter* is specified.

A.2.5.5. gridHeight

<1 double, 1 int> [no Default]

The size of the grid in the direction from *gridCenter* to *gridPole*, in degrees. Ignored if *gridRangeLat* and *gridRangeLon* are specified; required if *gridCenter* is specified.

A.2.5.6. gridWidth

<1 double, 1 int> [no Default]

The size of the grid in the direction perpendicular to the direction from *gridCenter* to *gridPole*, in degrees.
Ignored if *gridRangeLat* and *gridRangeLon* are specified; required if *gridCenter* is specified.

A.2.5.7. depthFast

<boolean> [true]

The order in which distance-depth information is written to output. When true, depths vary fastest. When false, distances vary fastest.

A.2.5.8. yFast

<boolean> [true]

The order in which geographic information is written to output. When true, y or latitude variable varies fastest. When false, x or longitude information varies fastest.

A.2.5.9. gridPositionParameters

<string> [longitude latitude depth]

The geographic information that is to be included in the output. The order of the position parameters in the output can be controlled with this property.

A.2.6. Input/Output from/to Database

If property *inputType* is equal to *database*, then information is loaded from tables origin, assoc, arrival and site and a new assoc table is populated with new values for timeres, azres, slores and vmodel, using the specified predictors.

A.2.6.1. dbInputUserName, dbOutputUserName

<string> [Default = user's environment variable DB_USERNAME]

Database account usernames.

A.2.6.2. dbInputPassword, dbOutputPassword

<string> [Default = user's environment variable DB_PASSWORD_<username>]

Database input/output account passwords.

A.2.6.3. dbInputInstance, dbOutputInstance

<string> [Default = user's environment variable DB_INSTANCE]

Database instance for input/output.

A.2.6.4. dbInputDriver, dbOutputDriver

<string> [Default = user's environment variable DB_DRIVER, or oracle.jdbc.driver.OracleDriver]

Database driver for input/output. Generally equals oracle.jdbc.driver.OracleDriver.

A.2.6.5. dbIntputTableTypes

<string> [Default =]

If the dbInputTableTypes property is specified, then the input table types specified with this property will default to the value of the dbInputTablePrefix property with the appropriate table type appended on the end.

A.2.6.6. dbInputTablePrefix

<string> [Default none]

If this property is specified then the four input tables (dbInputOriginTable, dbInputAssocTable, dbInputArrivalTable, dbInputSiteTable) will default to the value of this property with the appropriate table type (ORIGIN, ASSOC, ARRIVAL, SITE) appended on the end. If any of the four tables are also explicitly specified, then the explicitly specified name has precedence.

A.2.6.7. dbInputOriginTable

<string> [Default not allowed]

Name of the input origin table. Specifying this property will override any default values set by other properties.

A.2.6.8. dbInputAssocTable

<string> [Default not allowed]

Name of the input assoc table. Specifying this property will override any default values set by other properties.

A.2.6.9. dbInputArrivalTable

<string> [Default not allowed]

Name of the input arrival table. Specifying this property will override any default values set by other properties.

A.2.6.10. dbInputSiteTable

<string> [Default not allowed]

Name of the input site table. Specifying this property will override any default values set by other properties.

A.2.6.11. dbInputWhereClause

<string> [Default not allowed]

A string used to modify the default SQL query below to pull in specific data from a database. This property must be specified.

```
select origin.*, assoc.*, arrival.*, site.*
from account_name.origin origin, account_name.assoc assoc, account_name.arrival
arrival, account_name.site site, account_name.affiliation affiliation
where origin.orid=assoc.orid
and assoc.arid=arrival.arid
and arrival.sta=site.sta
and arrival.jdate >= site.ondate
and (site.offdate = -1 or arrival.jdate <= site.offdate)
and site.sta=affiliation.sta
and [dbInputWhereClause]
```

The affiliation table is optional and is only implemented if the Schema has an affiliation table specified.

A.2.6.12. dbInputAssocClause

<string> [Default not allowed]

A string used to modify the default SQL query below to pull in specific data from a database. This property is optional and is done in addition to the dbInputWhereClause.

```
select origin.*, assoc.*, arrival.*, site.*
from account_name.origin origin, account_name.assoc assoc, account_name.arrival
arrival, account_name.site site, account_name.affiliation affiliation
where origin.orid=assoc.orid
and assoc.arid=arrival.arid
and arrival.sta=site.sta
and arrival.jdate >= site.ondate
and (site.offdate = -1 or arrival.jdate <= site.offdate)
and site.sta=affiliation.sta
and [dbInputWhereClause]
and [dbInputAssocClause]
```

The affiliation table is optional and is only implemented if the Schema has an affiliation table specified.

A.2.6.13. dbOutputAssocTable

<string> [Default = none]

Name of the assoc table where output is to be written.

A.2.6.14. dbOutputAutoTableCreation

<boolean> [Default = false]

If set to true, automatically create output database tables if they do not already exist.

A.2.6.15. dbOutputTruncateTables

<boolean> [Default = false]

Boolean flag should be set to true if output database tables should be automatically truncated at the start of the run. Unless the *dbOutputPromptBeforeTruncate* property has been set to false, the user will be prompted before table truncation occurs.

A.2.6.16. dbOutputPromptBeforeTruncate

<boolean> [Default = true]

If *dbOutputTruncateTables* is true and this property is true, then the user is prompted before output table truncation occurs. If *dbOutputTruncateTables* is true and this property is false, table truncation occurs without warning.

A.2.7. Input Depth Specification

This section describes various ways in which one or more depths can be specified. These depth(s) will be applied to a whole range of latitude-longitude positions as described elsewhere.

A.2.7.1. depthSpecificationMethod

<string> [no default] (depths | depthRange | depthLevels | maxDepthSpacing)

Specified which method will be used to specify the depths at which predictions / model queries are to be calculated. Each depth specification method requires another property specification as described below.

A.2.7.2. depths

<list of doubles> [no default]

A list of depths, in km, that will be used for every latitude-longitude position.

A.2.7.3. depthRange

<2 doubles and 1 int> [no default]

Minimum and maximum depths, in km, and the number of desired depths.

A.2.7.4. depthLevels

<list of strings> [no default]

Depth will be determined at one or more major layer interfaces in the model. Example values include:

- top of upper_crust
- bottom of lower_crust
- above moho
- below moho
- etc.

A comma separated list of these values will generate multiple depths.
SALSA3D_v2.1 has the following layers/interfaces defined:

- SURFACE
- UPPER_CRUST
- MIDDLE_CRUST
- LOWER_CRUST
- MOHO
- M410
- M660
- CMB
- ICB

These can be thought of either as layers or as interfaces. For example, MOHO can refer to the interface or to the layer that includes the upper mantle between the 410 discontinuity and the Moho. Some layers/interfaces have names that sound more like interfaces (MOHO) while others have names that sound more like layers (UPPER_CRUST). To facilitate dealing with this, there are two ways to refer to each desired depth:

- Top/bottom of <layer name>
- Above/below <interface>

For example, "below moho" and "top of moho" would produce the same result, even though "below moho" is probably more natural. Same goes for 'bottom of middle_crust' and 'above lower_crust'. The former is more natural, but the latter is valid and produces the same result.

Specifying just a layer name, e.g., 'moho', is equivalent to specifying 'top of moho' or 'below moho'.

It is valid to specify multiple depth levels, separated by commas, e.g.:

"depthLevels = surface, top of upper_crust, top of middle_crust, top of lower_crust, above moho"

would return the depths of the tops of the specified layers and the model values at the top of each.

A.2.7.5. maxDepthSpacing

<double> [no default]

Unique depth profiles will be generated at each geographic position such that:

- each profile has the same number of depths,
- there are two depth nodes at each major layer interface in the model, one of which records model properties above the interface and the other below the interface.
- the maximum spacing of depth nodes is no greater than *maxDepthSpacing*.

A.2.7.6. maxDepth

<double or string> [default = infinity (center of the Earth)]

Optional if *maxDepthSpacing* is defined, ignored otherwise. When *maxDepthSpacing* is specified, this property defines the deepest point returned in each profile.

There are two ways to specify the maximum depth:

1. the maximum depth in km (a value of type double)
2. a model layer/interface name such as 'moho' or 'cmb'

SALSA3Dv2.1 has the following layers/interfaces defined:

- SURFACE
- UPPER_CRUST
- MIDDLE_CRUST
- LOWER_CRUST
- MOHO
- M410
- M660
- CMB
- ICB

A.2.8. Output Properties

A.2.8.1. outputFile

<string> [no Default]

Full path to output file where results are sent. Ignored if inputType = database, required otherwise.

If outputAttributes equals 'ray_path' and the outputFile name ends with 'vtk' then a vtk file is written with the ray geometries.

A.2.8.2. overwriteExistingOutputFile

<boolean> [default true]

If the file defined by the outputFile property exists, overwrite the file. If overwriteExistingOutputFile is false and the output exists, the libcorr3d surface is not computed.

separator
<string> [Default = space] (space | comma | tab)

Specify the character that should be used to separate information in each record of the output.

A.2.8.3. outputFormat

<string> [Default = %1.4f] (java format specifier for values of type double)

The first digit specifies the total width of the field and the second the number of digits to the right of the decimal point. For exponential notation, replace 'f' with 'e'. See
http://download.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html#syntax
for information about java format specifiers.

A.2.8.4. geotessFileFormat

<int> [Default = -1]

Specifies which GeoTess format to save a LibCorr3d surface to. By default, the surface is saved with the most recent version of GeoTess.

To save a surface that can be read by legacy GeoTess code, set geotessFileFormat = 2.

A.2.8.5. libcorr3dFileFormat

<int> [Default = -1]

Specifies which LibCorr3d format to save a LibCorr3d surface to. By default, the surface is saved with the most recent version of LibCorr3d.

To save a surface that can be read by legacy LibCorr3d code, set libcorr3dFileFormat = 1.

A.2.8.6. outputAttributes

<string> [no Default]

The attributes that should be sent to output by PCalc.

For model queries, PCalc supports whatever attributes are stored in the relevant GeoTessModel. SALSA3D GeoTessModels can return:

- pvelocity
- pslowness
- svelocity
- sslowness

For predictions, the following attributes are supported:

- travel_time (total travel time, including all applicable corrections, in seconds)
- tt_model_uncertainty (in seconds)
- tt_site_correction (in seconds)
- tt_ellipticity_correction (in seconds)

- tt_elevation_correction (travel time elevation correction at the station, in seconds)
- tt_elevation_correction_source (travel time elevation correction at the source, in seconds)
- dtt_dlat (derivative of travel time wrt latitude, seconds/radian)
- dtt_dlon (derivative of travel time wrt longitude, seconds/radian)
- dtt_dr (derivative of travel time wrt radius, seconds/km)
- slowness (in seconds/radian)
- slowness_degrees (in seconds/degree)
- slowness_model_uncertainty (in seconds/radian)
- slowness_model_uncertainty_degrees (in seconds/degree)
- dsh_dlat (derivative of horizontal slowness wrt latitude, in sec/radian^2)
- dsh_dlon (derivative of horizontal slowness wrt longitude, in sec/radian^2)
- dsh_dr (derivative of horizontal slowness wrt radius, in sec/radian/km)
- azimuth (receiver-source azimuth, in radians)
- azimuth_degrees (receiver-source azimuth, in degrees)
- azimuth_model_uncertainty (uncertainty of receiver-source azimuth, in radians)
- azimuth_model_uncertainty_degrees (in degrees)
- daz_dlat (derivative of receiver-source azimuth wrt latitude, unitless)
- daz_dlon (derivative of receiver-source azimuth wrt longitude, unitless)
- daz_dr (derivative of receiver-source azimuth wrt radius, degrees/km)
- backazimuth (source-receiver azimuth, in radians)
- backazimuth_degrees (source-receiver azimuth, in degrees)
- turning_depth (deepest point on the ray, in km)
- out_of_plane (The maximum amount by which a seismic ray deviates from the great circle plane containing the source and the receiver, in km. Considering source and receiver to be 3 component vectors in Earth centered coordinate system, the sign of out_of_plane is the same as the sign of source cross receiver.)
- distance (source-receiver epicentral distance, in radians)
- distance_degrees (source-receiver epicentral distance, in degrees)
- ray_type (a string indicating the type of ray produced: REFRACTION, REFLECTION, etc.)
- calculation_time (time required to compute the predicted values, in seconds)

To generate ray path geometries, specify 'ray_path'

For LibCorr3d surface generation, the following attributes are supported and are written to the LibCorr3d GeoTess model:

- tt_delta_ak135 – represents the difference between 3D GeoTess model predicted travel-times and base model (e.g., ak135) predicted travel-times.
    - Note that despite the name, the model being compared to does not need to be ak135. For instance, base model to compare to may be RSTT.
- tt_model_uncertainty – represents either the distance-dependent or the path-dependent travel-time uncertainty of the input 3D GeoTess model in seconds.

A.2.8.7. outputType

<string> [no Default]

Defines the type of output to write the data to. Can be set to: file, database, geotess, or libcorr3d.

A.2.8.8. outputHeader

<boolean> [default false]

If true, then a column heading will be generated for each column of output and appear as the first line of the output file.

A.2.9. Predictors

If model queries are to be returned, all the properties in this section are ignored. If predictions are to be computed, then property *predictors* (Appendix A.2.9.1) is required and other properties in this section that pertain to one of the predictors listed in *predictors* are also required.

A.2.9.1. predictors

<string> [Default = none] (lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor)

String indicating list of predictors that are to be used. For example, if value is "lookup2d, bender(P, Pn), rstt(Pn, Pg)" then lookup2d will be used for all phases not specified later in the list, Bender will be used for phase P and RSTT will be used for phase Pn and Pg. Even though Pn is specified by bender, it will be computed by rstt since rstt(Pn) comes later in the list than bender(Pn).

A.2.9.2. maxProcessors

<int> [Default = all available processors]

Limits the number of processors that PCalc will use to compute predictions in concurrent mode to the user-specified number.

A.2.9.3. parallelMode

<string> [Default = concurrent]

Sets whether prediction calculations are done concurrently on multiple threads or sequentially. Concurrent is the recommended setting.

A.2.9.4. batchSize

<int> [Default = 10000]

Records will be read from the input file, processed, and output to the output file in batches of this size. Applies only when input is from file or database. For greatcircle and grid input, this property is ignored.

A.2.9.5. lookup2dModel

<string> [Default = ak135] (ak135)

Name of the 1D model that Lookup2D should use to calculate predictions of seismic observables.

A.2.9.6. seismicBaseData

<string> [Default = seismic-base-data.jar]

Path to the seismicBaseData directory. If omitted or set equal to the default value, then distance-depth lookup tables stored in the PCalc jar files are used. These default tables support ak135 and iasp91 travel time lookup tables.

If this property is specified then the next two properties, *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory*, are not required.

A.2.9.7. lookup2dTableDirectory

<string> [Default = none]

Name of the directory where the travel time lookup tables reside. This directory will contain a separate file for each phase that will be supported. The file names can be names like 'PKP' or 'ak135.PKP'.

A.2.9.8. lookup2dEllipticityCorrectionsDirectory

<string> [Default = none]

Path of the directory where ellipticity correction coefficients are located for use with the Lookup2D predictor.

A.2.9.9. lookup2dUseEllipticityCorrections

<boolean> [Default = true] (true | false)

If false, then ellipticity corrections are not applied.

lookup2dOmitEllipticityCorrectionsForPhases

<String> [Default = empty] (list of phases)

If a comma or space delimeted list of phases is specified, then ellipticity corrections will not applied to observations with that phase.

A.2.9.10. lookup2dUseElevationCorrections

<boolean> [Default = true] (true | false)

If false, then elevation corrections are not applied.

A.2.9.11. lookup2dSedimentaryVelocityP

<double> [Default = 5.8 km/sec] ()

Seismic velocity used in the calculation of elevation corrections for all P phases.

A.2.9.12. lookup2dSedimentaryVelocityS

<double> [Default = 3.4 km/sec] ()

Seismic velocity used in the calculation of elevation corrections for all S phases.

A.2.9.13. benderModel

<string> [Default = none]

Path to GeoTessModel that Bender should use to calculate predictions of seismic observables. This can either be a file or a special salsa3d directory (see Section 3, bullet 4 for details about salsa3d directories).

A.2.9.14. benderUncertaintyType

<string> [Default = DistanceDependent] (DistanceDependent, PathDependent)

Type of travel time uncertainty desired.

If DistanceDependent is selected and *benderModel* (Appendix A.2.9.13) is not set to a special input salsa3d folder (Section 3, bullet 4), use *benderUncertaintyDirectory* (Appendix A.2.9.15) and *benderUncertaintyModel* (Appendix A.2.9.16) to define what subdirectory contains the needed distance dependent uncertainty files.

If PathDependent is selected, the user must specify a working directory using *benderUncertaintyWorkDir* (Appendix A.2.9.17).

A.2.9.15. benderUncertaintyDirectory

<string> [Default = none]

Directory where distance dependent uncertainty files can be found for use with Bender predictions.

Expecting to find subdirectories such as:

<benderUncertaintyDirectory>/<attribute>/<benderUncertaintyModel>

For example: if uncertainty information is in:
/index/SNL_tool_Root/seismicBaseData/tt/ak135

then specify:

benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135

A.2.9.16. benderUncertaintyModel

<string> [Default = none]

Subdirectory where distance dependent uncertainty files can be found for use with Bender predictions. Expecting to find subdirectories such as:

<benderUncertaintyDirectory>/<attribute>/<benderUncertaintyModel>

For example, if uncertainty information is in file:

/index/SNL_tool_Root/seismicBaseData/tt/ak135

then specify:

benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135

A.2.9.17. benderUncertaintyWorkDir

<string> [Default = none]

A path to a directory where PCalc can store temporary files when computing path dependent uncertainties. The directory will be generated automatically by PCalc if it does not exist. This directory can be removed or emptied any time it is not in use by PCalc. Property is ignored for DistanceDependent uncertainty.

A.2.9.18. benderAllowMOHODiffraction

<boolean> [Default = false]

If a crustal ray (Pg, Lg) impinges on the Moho, and this property is false, then the ray will be invalid.

A.2.9.19. allowCMBDiffraction

<boolean> [Default = false]

If a mantle ray impinges on the core-mantle boundary and this property is set to false, then the ray will be invalid.

A.2.9.20. use_tt_model_uncertainty

<boolean> [Default = true]

If true, travel time residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.21. use_az_model_uncertainty

<boolean> [Default = true]

If true azimuth residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.22. use_sh_model_uncertainty

<boolean> [Default = true]

If true slowness residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.23. use_tt_site_terms

<boolean> [Default = true]

If true, then travel time site terms computed for each station during tomography are applied to computed values. The site terms are stored in GeoTessModels read by Bender.

A.2.10. LibCorr3d

A.2.10.1. <predictor>PathCorrectionsType

<string> [Default = none] (libcorr)

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

Set the value to 'libcorr' to apply libcorr3d corrections. If this property is omitted, then corrections will not be applied.

A.2.10.2. <predictor>LibCorrPathCorrectionsRoot

<string> [Default = none]

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

The name of the directory where all the libcorr3D correction surfaces reside. This directory should contain a separate file for each correction surface.

A.2.10.3. <predictor>LibCorrPathCorrectionsRelativeGridPath

<string> [Default = "."]

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

The relative path from the directory where the correction surface files reside to the directory where the grid files reside.

A.2.10.4. <predictor>LibCorrInterpolatorType

<string> [Default = "linear"] (linear | natural_neighbor)

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

Type of horizontal interpolation to use.

A.2.10.5. <predictor>LibCorrPreloadModels

<boolean> [Default = false]

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

Whether all libcorr models should be loaded at startup or loaded on an 'as needed' basis.

A.2.10.6. <predictor>UsePathCorrectionsInDerivatives

<boolean> [Default = false]

<predictor> can be any predictor, [lookup2d, bender, benderlibcorr3d, ak135rays, rstt, infrasound_radial2d, hydro_radial2d, surface_wave_predictor]

Whether or not path corrections should be included in total values when computing derivatives of travel time with respect to source location.

A.2.10.7. fixAnomaliesThreshold

<float> [Default = 0.0] (15 sec is the recommended value)

Sets threshold at which a LibCorr3d traveltime is considered anomalous, i.e., it differs significantly from the traveltime of neighboring nodes. The difference of a node's traveltime value and the traveltimes of its neighbors is determined. If this difference exceeds the fixAnomaliesThreshold value, the anomalous traveltime is replaced with the average value of the neighboring nodes.

A.2.11. Model Queries

A.2.11.1. benderModel

<string> [Default = none]

Path to GeoTessModel that Bender should query for model values. This can either be a file or a special salsa3d directory (see Section 3, bullet 4 for details about salsa3d directories).

A.2.12. Ray Path Geometry

PCalc can compute and output the geometry of ray paths through the SALSA3D model using Bender. For this action to succeed, the following properties must be specified:

- application = predictions
- predictors = bender
- inputType = greatcircle or file
- outputAttributes = ray_path
- rayPathNodeSpacing = Point spacing along the computed ray paths, in km.
- If inputType is greatcircle, properties *site* and *gcStart* must both be specified, and their latitudes and longitudes must be equal.

A.2.12.1. rayPathNodeSpacing

<double> [Default = -1]

Point spacing along the computed ray paths, in km. If ≤ 0, then an error is thrown.

APPENDIX B. LIBCORR3D GRIDS

Quick Summary

GeoTessGrids for LibCorr3D models can be configured in 3 basic ways.

1. To build a custom grid for each station:
    a. Specify the parameters that control grid geometry directly in the PCalc properties file.
    b. Do not specify *geotessIntputGridFile* or *geotessOutputGridFile.*
    c. See Appendix B.1 for further detail.
2. To build a common grid that can be shared by multiple LibCorr3D models:
    a. Use GeoTessBuilder to construct the common grid.
    b. In the PCalc properties file specify *geotessInputGridFile* but do not specify *geotessRotateGridToStation*.
    c. See Appendix B.2 for further detail.
3. To build a custom grid that can be shared by multiple LibCorr3d models:
    a. Use GeoTessBuilder to construct the common grid.
    b. In the GeoTessBuilder properties file, do not specify any rotation parameters; this will result in a grid with vertex 0 located at the North Pole.
    c. In the PCalc properties file specify *geotessInputGridFile* and *geotessRotateGridToStation*.
    d. These grids will not be compatible with older versions of LibCorr3D C++.
    e. See Appendix B.3 for further detail.

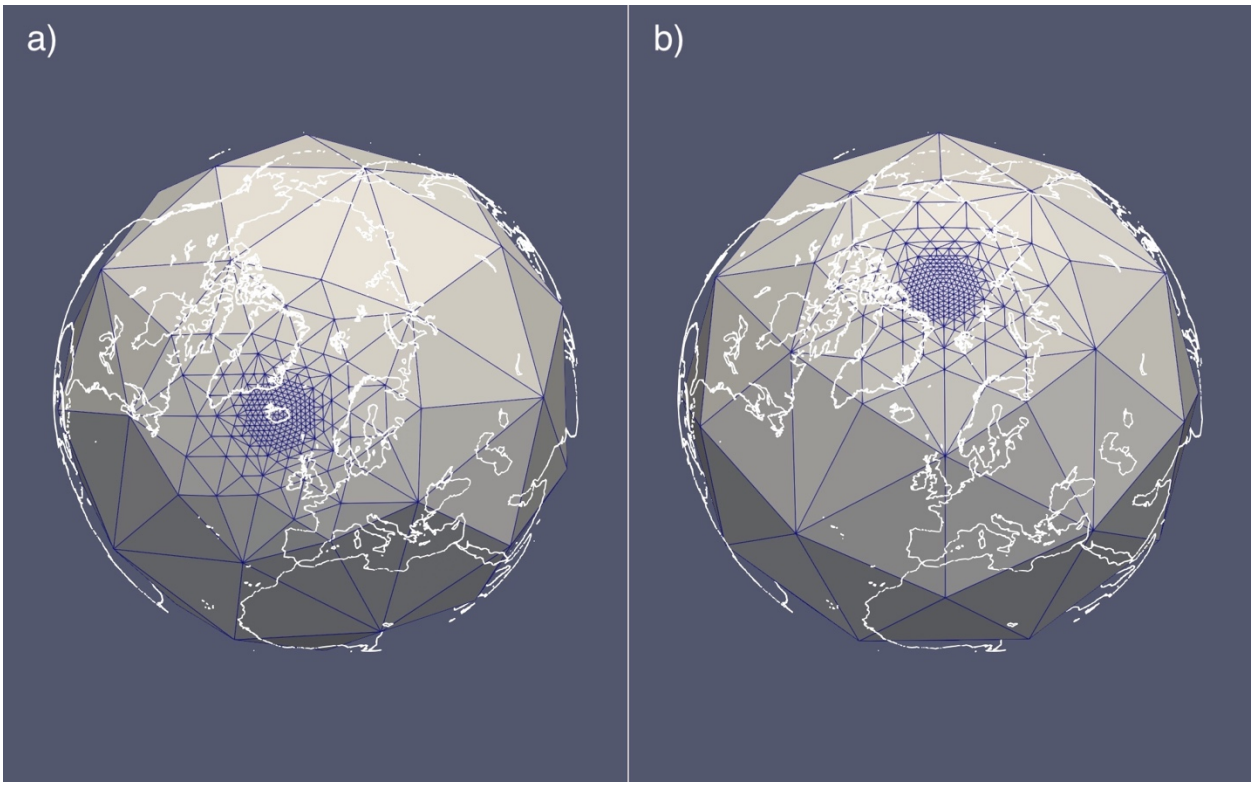


Figure 15. a) A custom grid for a station in Iceland. b) A custom grid centered on the North Pole that can be shared among multiple stations when appropriate rotations are applied at run time.

B.1. Custom Grid for Each Station

To create a custom grid for a station, include the following properties in the PCalc properties file:

```
# rotate the grid so that grid vertex 0 is located at the location
# of the station
geotessRotateGridToStation = true

# Refine the grid in spherical caps centered on the station.
# The parameters of the spherical_cap definitions are:
#    1 - latitude  of center of the cap, in degrees.
#        (<site.lat> is replaced with latitude of current site)
#    2 - longitude of center of the cap, in degrees.
#        (<site.lon> is replaced with longitude of current site)
#    3 - radius of the spherical cap in degrees.
#    4 - tessellation index.  Always 0 for pcalc applications.
#    5 - grid resolution inside the cap, in degrees.
geotessPolygons = spherical_cap, <site.lat>, <site.lon>, 20, 0, 4
```

This grid (e.g., Figure 15a), will have a default grid resolution of 64 degrees far from the site and 4 degrees resolution within 20 degrees of the site

Note that PCalc property *geotessOutputGridFile* is not specified. This means that the GeoTessGrid information will be stored in the same file as the LibCorr3D model information. Different stations in the same LibCorr3D model set could have grids built with different *geotessPolygons* parameters.

B.2. Uniform Grid Shared Among Multiple Stations

In this case, there will be a single, uniform grid where grid vertex 0 is located at the North Pole. The grid is stored in a file separate from the LibCorr3D model information, and that grid will be used by multiple LibCorr3D models. It is necessary to build the grid using the GeoTessBuilder application. Example GeoTessBuilder properties for building a typical grid would be:

```
# specify GeoTessBuilder grid construction mode.
gridConstructionMode = scratch

# number of multi-level tessellations to build
nTessellations = 1

# grid resolution in degrees
baseEdgeLengths = 4

# file to receive the GeoTessGrid definition
outputGridFile = /path/grid_4deg.geotess

# file to receive the vtk file for visualization with ParaView. (Optional)
vtkFile = /path/grid_4deg.vtk
```

The preceding are properties for use with GeoTessBuilder to build a uniform 4-degree grid with grid vertex 0 at the North Pole. Then in the PCalc properties file include the following property:

```
geotessInputGridFile = /path/grid_4deg.geotess
```

This property specifies that PCalc should use the grid stored in `/path/grid_4deg.geotess` when it builds the LibCorr3D model. Property `geotessOutputGridFile` will default to the same file with the result that the grid will not be copied into the output libcorr3d model file. Only a reference to the external grid will be included in the model file.

B.3. Non-Uniform Grid Shared Among Multiple Stations

It is also possible to construct a non-uniform grid that is shared among multiple stations, so long as the basic geometry is the same for each station. It is necessary to build a grid centered on the North Pole using the GeoTessBuilder application. Example GeoTessBuilder properties for building a typical grid like the one in Figure 15b would be:

```
# specify GeoTessBuilder grid construction mode.
gridConstructionMode = scratch

# number of multi-level tessellations to build
nTessellations = 1

# grid resolution in degrees
baseEdgeLengths = 64

# Refine the grid in a spherical cap around the North Pole.
# The parameters of the spherical_cap definitions are:
#    1 - latitude of center of the cap, in degrees.
#    2 - longitude of center of the cap, in degrees.
#    3 - radius of the spherical cap in degrees.
#    4 - tessellation index. Always 0 for pcalc applications.
#    5 - grid resolution inside the cap, in degrees.
polygons = spherical_cap, 90, 0, 20, 0, 4
```

```
# file to receive the GeoTessGrid definition
outputGridFile = /path/grid.geotess

# file to receive the vtk file for visualization with ParaView. (Optional)
vtkFile = /path/grid.vtk
```

The preceding are properties for use with GeoTessBuilder to build a non-uniform grid where grid vertex zero is located at the North Pole (Figure 15b). No grid rotations should be applied. Then in the PCalc properties file include the following properties:

```
geotessInputGridFile = /path/grid.geotess
geotessRotateGridToStation = true
```

The first property specifies that PCalc should use the grid stored in `/path/grid.geotess` when building the LibCorr3D model. Property `geotessOutputGridFile` will default to the same file with the result that the grid will not be copied into the output libcorr3d model file. Only a reference to the external grid will be included in the model file. The second property specifies that interpolation points should be rotated into grid coordinates, where vertex 0 is located at the North Pole, prior to performing the interpolation calculations. This scenario will behave as though each station had a custom grid centered on the station location but will enjoy the memory savings of having many stations share a reference to the same grid.

APPENDIX C. PCALC SITEFILES

C.1. Usage

PCalc site files were developed to allow the user to produce LibCorr3d surfaces for a large number of stations in a shortened amount of time by using as many computers as available to perform calculations. This system thus allows the user to generate LibCorr3d surfaces in parallel. A description of how site files enable this capability is provided below.

C.2. Site File Description

A site file is a text file containing the site information for each station the user wants to calculate a LibCorr3d surface for. Specifically, each row is identical to the site property (Appendix A.2.2.3), containing entries for sta, ondate, offdate, lat, lon, elevation, staname, statype, refsta, dnorth and deast, separated by tabs or white space (lddate is optional). For example, a few records from such a file may look like:

```
MSEY   -1     2286324      -4.67370     55.47920     0.4750 Mahe, Seychelles ...
MSKU   -1     2286324      -1.65570     13.61160     0.2870 Masuku ...
MSVF   -1     2286324      -17.74470    178.05270    0.8683 Monasavu, Viti Levu ...
NEW    -1     2286324      48.26333     -117.12000   0.7600 Newport, WA ...
NIL    -1     2286324      33.65000     73.25120     0.5360 Nilore, Pakistan ...
NNA    -1     2286324      -11.98730    -76.84220    0.5750 NANA, PERU ...
NOA    -1     2286324      61.03970     11.21480     0.7170 NORSAR Array, Norway ...
```

Note that station definition records have been truncated in this document for clarity.

To use a site file in place of the typical site property (Appendix A.2.2.3), specify the siteFile property such that it points to the path of the site file. There is no need to specify the site property in this case. Examples 9-11 (Sections 3.3.1, 3.3.2, 3.3.3) make use of site files.

The user is encouraged to generate just one site file containing all desired station information – there is no limit to the number of sites/rows the user can enter. It is recommended that users make a backup copy of the site file prior to launching PCalc because PCalc will modify the contents of the file during execution. PCalc can then be launched on multiple computers, all using this master site file. To prevent the creation of redundant surfaces, each instance of PCalc launched reads the first row in the master site file that *does not* begin with the comment character '#'. Once that row is read, the specific PCalc instance inserts a '#' character at the beginning of that row in the file. While a PCalc instance is making these file modifications, the site file is locked such that other PCalc instances are unable to access the site file. By making these file modifications, no two instances of PCalc will attempt to process the same site.

An example of a modified site file following ~2 hours of calculation on 4 computers is shown below:

```
#MSEY  -1     2286324      -4.67370     55.47920     0.4750 Mahe, Seychelles ...
#MSKU  -1     2286324      -1.65570     13.61160     0.2870 Masuku ...
#MSVF  -1     2286324      -17.74470    178.05270    0.8683 Monasavu, Viti Levu ...
#NEW   -1     2286324      48.26333     -117.12000   0.7600 Newport, WA ...
NIL    -1     2286324      33.65000     73.25120     0.5360 Nilore, Pakistan ...
NNA    -1     2286324      -11.98730    -76.84220    0.5750 NANA, PERU ...
NOA    -1     2286324      61.03970     11.21480     0.7170 NORSAR Array, Norway ...
```

```
# MSEY   computer01 began    2022-10-19 09:48:56 -0600
# MSKU   computer02 began    2022-10-19 09:49:26 -0600
# MSVF   computer03 began    2022-10-19 09:49:39 -0600
# MSVF   computer03 finished 2022-10-19 11:41:49 -0600 ( 1.869535 hours)
# NEW    computer03 began    2022-10-19 11:41:49 -0600
```

The stations MSEY, MSKU, MSVF, and NEW all have surfaces being generated as indicated by the # symbol next to their rows. Additional comments on which computer calculated a station's LibCorr3d surface, the status of the surface (i.e., whether the surface has finished generating), and the times a surface began and finished generating, respectively, are provided at the bottom of the site file.

The same site file is shown below once more after all instances of PCalc have completed processing (in ~10 hours). To make it clear when each station's surface began and finished generation, the file has been alphabetically sorted.

```
# MSEY   computer01 began    2022-10-19 09:48:56 -0600
# MSEY   computer01 finished 2022-10-19 12:16:04 -0600 ( 2.452057 hours)
# MSKU   computer02 began    2022-10-19 09:49:26 -0600
# MSKU   computer02 finished 2022-10-19 12:58:38 -0600 ( 3.153214 hours)
# MSVF   computer03 began    2022-10-19 09:49:39 -0600
# MSVF   computer03 finished 2022-10-19 11:41:49 -0600 ( 1.869535 hours)
# NEW    computer03 began    2022-10-19 11:41:49 -0600
# NEW    computer03 finished 2022-10-19 13:54:23 -0600 ( 2.209253 hours)
# NIL    computer01 began    2022-10-19 12:16:04 -0600
# NIL    computer01 finished 2022-10-19 15:41:49 -0600 ( 3.429111 hours)
# NNA    computer02 began    2022-10-19 12:58:38 -0600
# NNA    computer02 finished 2022-10-19 17:07:06 -0600 ( 4.141127 hours)
# NOA    computer03 began    2022-10-19 13:54:23 -0600
# NOA    computer03 finished 2022-10-19 13:54:23 -0600 ( 0.012000 seconds)
#MSEY  -1     2286324      -4.67370     55.47920     0.4750 Mahe, Seychelles ...
#MSKU  -1     2286324      -1.65570     13.61160     0.2870 Masuku ...
#MSVF  -1     2286324      -17.74470    178.05270    0.8683 Monasavu, Viti Levu ...
#NEW   -1     2286324      48.26333     -117.12000   0.7600 Newport, WA ...
#NIL   -1     2286324      33.65000     73.25120     0.5360 Nilore, Pakistan ...
#NNA   -1     2286324      -11.98730    -76.84220    0.5750 NANA, PERU ...
#NOA   -1     2286324      61.03970     11.21480     0.7170 NORSAR Array, Norway ...
```

All rows in the master site file are now commented out with the # symbol. If the user desires to reuse the site file, they must either add additional rows with new station information or uncomment the rows in the site file to regenerate the surfaces. Note that to regenerate existing surfaces, the user must either remove the existing surfaces or set the property overwriteExistingOutputFile (Appendix A.2.8.2) to true prior to the new PCalc runs.

C.3. Effects of Aborted PCalc Runs

If one or more PCalc instances are aborted during LibCorr3d surface calculations, either by a system failure or by a user, surfaces that were being processed by the aborted PCalc instances will not be generated. When all PCalc instances have finished, the user should ensure that property overwriteExistingOutputFile (Appendix A.2.8.2) is set to false, restore the contents of the site file to its original state and relaunch PCalc on as many computers as are available. Since overwriteExistingOutputFile is false, surfaces that already exist on the file system will not be

recomputed. Only surfaces that do not already exist on the file system will be computed and written to disk.

DISTRIBUTION

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name
Mike Begnaud	mbegnaud@lanl.gov	Los Alamos National Laboratory
Sanford Ballard	sballard999@gmail.com	Retired
Jorge Roman-Nieves	jorge.roman-nieves.1@us.af.mil	AFTAC
Gregory Wagner	gregory.wagner@us.af.mil	AFTAC

Email—Internal

Name	Org.	Sandia Email Address
Stephanie Teich-McGoldrick	06756	steichm@sandia.gov
John Merchant	06752	bjmerch@sandia.gov
Daniel Gonzales	06752	dgonza2@sandia.gov
Andrea Conley	06752	acconle@sandia.gov
Kathy Davenport	06756	kdavenp@sandia.gov
Robert Porritt	06756	rwporri@sandia.gov
Julia Sakamoto	06752	jsakomo@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.