SANDIA REPORT
SAND2021-10207
Printed August 2021

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

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 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/PCalc

For additional information on GeoTess, SALSA3D, and other related software, please see:

https://github.com/sandialabs/GeoTessJava
www.sandia.gov/geotess
www.sandia.gov/salsa3d

This page left blank.

CONTENTS

LIST OF FIGURES

This page left blank

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)

This page left blank

1. INTRODUCTION

PCalc (prediction Calculator) is a software package used for ray-tracing and travel-time computation in 3D Earth velocity models. The software uses the pseudobending algorithm of *Um and Thurber* (1987). 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/PCalc

It is packaged with the latest version of this user's manual, a set of run examples (the use of which is outlined in the "Tutorial and Examples" section below) and a jar file of the PCalc software. PCalc is formatted as a Java jar file compiled with JDK10 and requires a java runtime environment to be installed in a user's machine. If the user wishes to use PCalc with a database the Oracle ojdbc*.jar file (version ≥ 7) is required.

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 SALSA3D or other 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, PBPbc, and Pn through 3D models of Earth's compressional-wave velocity structure, and the ability to trace phases qS, 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.

- 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.

A typical run of PCalc requires the following:

1. Specification of a velocity model, either the AK135 tables included within PCalc, or a user-specified model in GeoTess format.

2. Specification of a mode of operation, which can be to compute model queries, compute ray-path geometry, or to compute travel time predictions from points on/in the Earth to a specified station location

3. Definition of a geometry at which the model will be queried or to which travel-time computation is to be completed.

See the "Tutorial and Examples" section below for more details on how these are specified and input into the PCalc software.

2. PSEUDOBENDING IN PCALC

The ray tracing algorithm in PCalc, aka "bender", is an implementation of the pseudobending algorithm of *Um and Thurber* (1987). 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 pseudobending 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.

- Pseudobending 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 (see https://github.com/sandialabs/GeoTessJava and www.sandia.gov/geotess) means that a user can compute travel times and ray paths through any Earth velocity model that can be represented using the GeoTess format. For example, a user can use the ray paths computed for a starting model in tomographic models of Earth's velocity structure. The companion software package LocOO3D (see https://github.com/sandialabs/LocOO3D) can also use bender to compute seismic event locations using ray paths computed for arbitrary Earth models in GeoTess format.

This page left blank

3. PCALC INSTALLATION

To run PCalc and execute the examples provided, the user should set up their target directory based on the following file structure. To explore the structure of GeoTess models like SALSA3D and RSTT, the user may wish to also install the `geotess` software. See https://github.com/sandialabs/GeoTessJava for details.

1. Create a target directory on your system (e.g. `pcalc_software`).

2. Into the working directory, download the latest version of the PCalc software, documentation, and example files from GitHub at https://github.com/sandialabs/PCalc.

   The download will include a pcalc jar file so users do not need to recompile the code. The code can be recompiled by cd'ing into the PCalc directory and executing 'mvn clean package' at the command line.

3. Download the SALSA3D model from https://www.sandia.gov/app/uploads/sites/109/2021/10/SALSA3D_v2_PS.geotess.zip:

   - SALSA3D_v2_PS.geotess or SALSA3D_Model.tgz

4. Download the RSTT model from www.sandia.gov/rstt/software/downloads/models.html:

   - pdu202009Du.zip

   If needed, follow the installation instructions at www.sandia.gov/rstt to install dependencies and configure RSTT.

5. Edit the pcalc run script `pcalc.sh` downloaded from GitHub. There are three variables in the file that specify the locations of certain required resources on the user's system:

   pcalc_jar must be set to the full path name of the pcalc jar file located in the working directory (e.g., `snl_pcalc_software`). It has a name similar to pcalc-3.2.2-jar-with-dependencies.jar

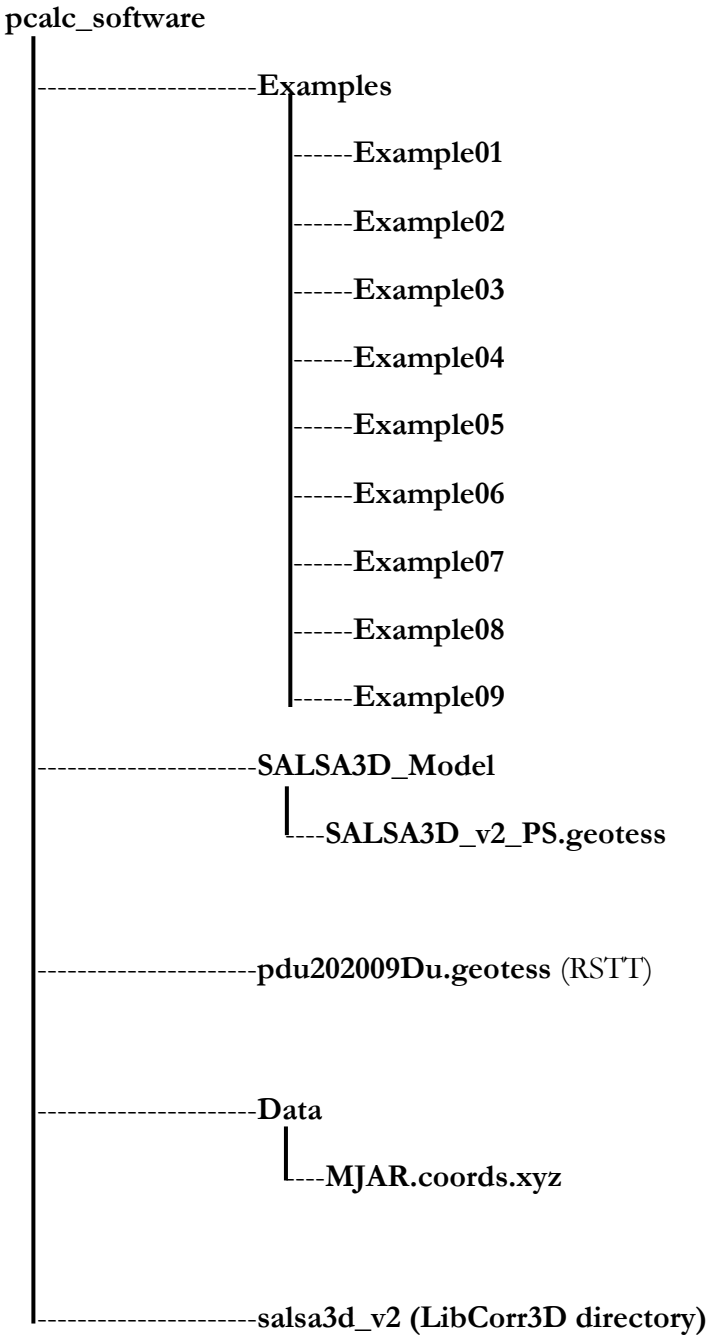
   ojdbc_jar should be set to the location of the user's Oracle ojdbc*.jar file. This is only required if the user plans to use PCalc's database features.

   slbm_libdir should be set to the path of the RSTT library directory. This is only required if the user plans to use RSTT to compute travel time predictions. If the user has specified RSTT_ROOT property in their ~/.bash_profile, then specification of slbm_libdir is not required.

   Users may wish to add the path to the directory that contains pcalc.sh to their ~/.bash_profile, ~/.cshrc, ~/.bashrc, or ~/.profile, e.g.:

   ```
   PATH=$PATH:/Users/username/github/pcalc_software
   ```

6. When complete, the user should have the following directory structure set up to run the examples in this manual:



However, the user can utilize a different structure provided all paths in the properties files are correctly set.

4. PCALC TUTORIAL AND EXAMPLES

The input files for a PCalc run are:

1. A required properties file which contains the property settings used by PCalc for a particular run. Properties files will be 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

2. An optional GeoTess file (see https://github.com/sandialabs/GeoTessJava) 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 will be labeled with a *.geotess extension.

    In Sections 4.1.1 and 4.1.2 the user will use the GeoTess file SALSA3D_v2_PS.geotess to run the examples. The SALSA3D velocity models are available from https://www.sandia.gov/salsa3d/velocity-models/.

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: the parameters describing these geometries are specified directly in the properties file (see Sections 4.1.2, 0, 4.2, 4.2.3, 4.3).

    a. In Sections 4.1.1 and 4.1.2, 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 files necessary for the generation of LibCorr3D surfaces (see Section 4.4 for details on LibCorr3D surfaces).

    An example version of this folder, `salsa3d_v2` is included with PCalc in the GitHub distribution. 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 4.4).

    A brief definition for each of the folder contents is given below. More detailed definitions are beyond the scope of this guide. Please refer to the *Ballard et al. 2016b* reference [3] and https://github.com/sandialabs/GeoTessJava documentation for more information on a particular file and how to use the GeoTess files, respectively.

    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 https://github.com/sandialabs/GeoTessJava documentation). 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 4.4).

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 0), all examples will be run on the command line as: `pcalc *.properties`.

The Examples in Section 4.1 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 PCald to generate ray paths via raytracing through an input velocity model. Finally, Section 4.4 details an example of generating LibCorr3D surfaces using PCalc.

4.1. PCalc Model Query Examples

4.1.1. PCalc Model Query Using MJAR.coords.xyz File

The properties file **Example01/pcalc_query_file.properties** (shown below in Figure 1) contains input parameters to query a model at locations specified in the file **Data/MJAR.coords.xyz** (see Section 4).

The properties file in Figure 1 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 them. These properties are:

- **application** – specifies the type of application for PCalc to perform. The types can be model_query or prediction (also see Section 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 Section A.2.1.2)

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

```
application = model_query

workDir = /Users/username/pcalc/Examples/Example01/
```

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 1, the user requests that PCalc query a GeoTess model with the geotessModel property:

```
geotessModel= <property:workDir>../SALSA3D_Model/SALSA3D_v2_PS.geotess
```

The geotessModel property specifies the path to the input model, in this case SALSA3D_V2_PS.GEOTESS.geotess. 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/pcalc/Examples/Example01/../SALSA3D_Model/SALSA3D_v2_PS.geotess.

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

```
#
# Property file for application PCalc v. 3.0
#
#================================================================
application=model_query
workDir=/Users/username/pcalc/Examples/Example01/
#================================================================
#
# PREDICTORS, MODELS, UNCERTAINTY, ETC.
#
#================================================================
geotessModel=<property:workDir>../SALSA3D_Model/SALSA3D_v2_PS.geotess
#================================================================
#
# 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>../Data/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 1. PCalc Model Query Properties File pcalc_query_file.properties.

In order to query a model, locations must be input in order for PCalc to determine where in the model to extract information from. 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 Section 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 proposed event location coordinates at which to query the model.

The exact contents of the text file are specified using the inputAttributes property. In Figure 1, inputAttributes are set to:

```
inputAttributes = latitude longitude depth
```

Thus, the input text file must contain a latitude (degrees), longitude (degrees), and depth (km) column separated by white space. To modify the order of the file columns or what each file column represents, the user can modify the inputAttributes property (Section 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 columns of the text file are in longitude, latitude, depth order, the user 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.

In this example, the inputFile is MJAR.coords.xyz, the text file provided with the PCalc software download (Section 4). 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/rows from MJAR.coords.xyz at a time. By default, batches are of size 10,000 (see Section A.2.9.3). batchSize can only be applied when using a file or database (Section *) for input.

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

The property outputFile (Section 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 this property is required unless the inputType property is set to database (see Section 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 property equal to the path and file to save the log to. In this example, the path is defined as:

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

The next property, terminalOutput (Section 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 1 is for demonstration purposes only.

The property separator (Section A.2.8.2) 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 1 is for demonstration purposes only.

The final property, outputAttributes (Section A.2.8.4) 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 this final property set, the properties file in Figure 1 can be run (see Section 4). 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 8.1246
13.710173 144.000000 75.018 8.1499
13.710173 144.000000 70.018 8.1753
13.710173 144.000000 65.019 8.2009
   …         …          …     …
```

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 km/s at that queried event hypocenter.

4.1.2. PCalc Model Query Using A Great Circle

The properties file **Example02/pcalc_query_greatcircle.properties** (shown below in Figure 2) contains input parameters to query a model at locations specified using a great circle.

Much of this example is the same as the example shown in Section 4.1.2. Thus, we will only discuss new properties here.

The first major difference between this example and the previous example is the definition of inputType. Where in Section 4.1.2 the locations to query were input via a text file, here we set inputType to greatcircle:

```
inputType = greatcircle
```

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

The first property, gcStart (Section 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 (Section A.2.4.2) that can be used to set the end of the great circle in the same way as gcStart.

In the next properties, gcDistance (Section A.2.4.3) and gcAzimuth (Section A.2.4.4), we 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 to reach the end of the defined great circle. 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.

```
#
# Property file for application PCalc v. 3.0
#
#=================================================================
application = model_query
workDir = /Users/username/pcalc/Examples/Example02/

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

geotessModel = <property:workDir>../SALSA3D_Model/SALSA3D_v2_PS.geotess

#=================================================================
#
# 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
```

Figure 2. PCalc Model Query Properties File pcalc_query_greatcircle.properties

```
# 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

#============================================================
#
# 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 user next needs to specify the spacing between points of the great circle. This is done by specifying the property gcSpacing (Section 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 (Section 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 (Section 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:

- depths – specify a list of depths, in km, that will be used for every latitude-longitude position.
- depthRange – specify a minimum and maximum depth, in km, and the number of desired depths.
- depthLevels – depth will be determined above and below a specified layer in the model (see Section A.2.7.4 for more detail)
- maxDepthSpacing – 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.

In this example, depthSpecificationMethod is set to maxDepthSpacing:

```
depthSpecificationMethod = maxDepthSpacing
```

Since maxDepthSpacing was specified, the user also specifies the properties maxDepthSpacing (Section A.2.7.5) and maxDepth (Section A.2.7.6). maxDepthSpacing is defined in the fourth bullet above. 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 Section 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 4.1.1 for details on outputAttribute), the P-wave slowness at depths directly above and below each layer in the model will be returned in the output until the top of the M660 layer is reached. These depths will be separated by a maximum value of 100 km.

The remaining properties in this example, with the exception of outputHeader were covered in Section 4.1.1. 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
```

A sample of the output when PCalc is run on this properties file is shown below:

```
x       y       distance        depth   pslowness
-1.000000000    0.000000000     0.000000000     -0.000207       0.6667
-1.000000000    0.000000000     0.000000000     4.871375        0.6667
-1.000000000    0.000000000     0.000000000     4.871375        0.5242
-1.000000000    0.000000000     0.000000000     5.832801        0.5242
```

4.1.3. PCalc Model Query Using a Grid

The properties file **Example03/pcalc_query_grid.properties** (shown below in Figure 3) contains input parameters to query a model at locations specified using a grid.

Much of this example is the same as those shown in Sections 4.1.1 and 4.1.2. Thus, we will only discuss new properties here.

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

workDir = /Users/username/pcalc/Examples/Example03/

application = model_query

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

geotessModel = <property:workDir>../SALSA3D_Model/SALSA3D_PUBLIC.geotess

#================================================================
#
# 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 3. 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 first major difference between this example and the previous examples is the definition of inputType. Whereas in Sections 4.1.2 and 4.1.2 the locations to query were input via a text file or by defining a great circle, respectively, here we set inputType to grid:

```
inputType = grid
```

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

The two main properties needed are gridRangeLat and gridRangeLon (Sections 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
```

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 4.1.2 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 4.1.2. Thus, the output file will always have locations defined as longitude, latitude, and depth columns, with the requested outputAttributes making up the remainder of the columns.

A sample of the output when PCalc is run on this properties file is shown below:

longitude	latitude	depth	pslowness
70.000000000	15.000000000	100.000000	0.1255
70.000000000	15.000000000	200.000000	0.1227
72.000000000	15.000000000	100.000000	0.1256
72.000000000	15.000000000	200.000000	0.1225

4.2. PCalc Travel Time Prediction Examples

4.2.1. PCalc Travel Time Prediction Using MJAR.coords.xyz File

The properties file **Example04/pcalc_predictions_file.properties** (shown below in Figure 4) contains input parameters to generate predictions for locations specified in a file.

Much of this example is the same as those shown in previous sections. Thus, we will only discuss new properties here.

```
#=====================================================================
#
# 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 = /Users/username/pcalc/Examples/Example04/

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

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

predictors = bender

benderModel = <property:workDir>../SALSA3D_Model/SALSA3D_PUBLIC.geotess

#=====================================================================
#
# 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>../Data/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], o
# If inputAttributes includes sta and [jdate | origin_time | arrival_time] then
# The default value is sta, jdate, site_lat, site_lon, site_elev, origin_lat, o
inputAttributes = origin_lat, origin_lon, origin_depth

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

Figure 4. PCalc Prediction Properties File pcalc_predictions_file.properties.

```
# 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

#============================================================
#
# 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
```

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

This is the first of a few examples detailing the prediction application of PCalc. All predictions are computed in concurrent parallel mode (multi-threaded). Thus, the first new property is maxProcessors (Section A.2.9.2). maxProcessors defines the maximum number of processors to be used in the PCalc run. 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 is now set to predictions:

```
application = predictions
```

When application is set to predictions, the property predictors (Section 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.
- tauptoolkit -perform predictions through an input model using the TauP Toolkit ray-tracing algorithm (see link).
- bender – perform predictions through an input model using the Bender ray-tracing algorithm. See Section 0 for details on Bender.
- slbm – perform predictions through an input model using the Regional Seismic Travel Time (RSTT) method (see link). slbm is the former name of RSTT.

Different predictors can be applied to different seismic phases. For instance, if predictors is set equal to "lookup2d, bender(P, Pn), slbm(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 SLBM, i.e., RSTT, will be used for phases Pn and Pg.

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

```
predictors = bender
```

With predictors set to bender, the benderModel property (A.2.9.11) needs to be defined. This property is the path to the GeoTessModel that Bender should use to calculate predictions of seismic observables.

In this example, we use SALSA3D_v2_PS.geotess, so the benderModel property is defined as:

```
benderModel = <property:workDir>../SALSA3D_Model/SALSA3D_v2_PS.geotess
```

As in Section 4.1.1, 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 4.1.1, inputAttributes is now defined as:

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

whereas in Section 4.1.1, 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 (Section 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

- 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, origin_lon, must be included. Note that the inputAttribute origin_depth is also an option – if origin_depth is not defined in the text file, it must be defined using depth properties (see Section A.2.7). The inputAttributes site_lat, site_lon, site_elev, and phase must also be defined using properties (see Sections A.2.2.3 and A.2.2.4).

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 4, the outputAttributes property is set. The predictions application has far more outputAttributes options than the model_query application seen in Section 4.1.1. These outputAttributes are listed in Section A.2.8.4. 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 go from that hypocenter to station MJAR.

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

A sample of the output when PCalc is run on this properties file is shown below:

```
origin_lat origin_lon origin_depth travel_time
13.710173 144.000000 80.018 299.4320
13.710173 144.000000 75.018 299.8480
13.710173 144.000000 70.018 300.2570
13.710173 144.000000 65.019 300.6660
13.710173 144.000000 60.019 301.0740
```

4.2.2. PCalc Travel Time Prediction Using Great Circles

The properties file **Example05/pcalc_predictions_greatcircle.properties** (shown below in Figure 5) contains input parameters to generate predictions for locations specified by a great circle.

The properties in this file have been discussed in prior sections. Refer to Sections 4.1.2, 4.1.3, 4.2.1, and Appendix A for information on these properties. Note that the lookup2d predictor is used in this example.

A sample of the output when PCalc is run on this properties file is 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
```

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

workDir = /Users/username/pcalc/Examples/Example05/

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
```

Figure 5. PCalc Prediction Properties File pcalc_predictions_greatcircle.properties.

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

4.2.3. PCalc Travel Time Prediction Using a Grid

The properties file **Example06/pcalc_predictions_grid.properties** (shown below in Figure 6) contains input parameters to generate predictions for locations specified by a grid.

The properties in this file have been discussed in prior sections. Refer to Sections 4.1.2, 4.1.3, 4.2.1, and Appendix A for information on these properties.

A sample of the output when PCalc is run on this properties file is shown below:

```
longitude latitude depth travel_time
70.000000000 15.000000000 11.746270 631.0310
72.000000000 15.000000000 14.652013 619.8160
74.000000000 15.000000000 34.726226 607.3200
76.000000000 15.000000000 41.667169 595.2580
```

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

workDir = /Users/username/pcalc/Examples/Example06/

application = predictions

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

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

benderModel = <property:workDir>../SALSA3D_Model/SALSA3D_PUBLIC.geotess

#=========================================================================
#
# 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
```

Figure 6. PCalc Prediction Properties File pcalc_predictions_grid.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 = depthLevels

depthLevels = top of MOHO

#=============================================================
#
# 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.

4.2.4. PCalc Travel Time Prediction Using a Database

As previously mentioned, if the user has the necessary Oracle ojdbc*.jar file available, PCalc can use Oracle databases for I/O rather than text files. Figure 7 shows an example properties file that loads origin, assoc, arrival, site information from database tables and writes new assocs where timres, azres and slowres are replaced with values computed using the specified RSTT model. This file serves as a template for database input/output parameters that can be used with other examples in this tutorial. This example properties file is in **Example07/pcalc_predictions_db.properties**

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

application = predictions
workDir = /Users/username/pcalc_software/testing_db

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

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

slbmModel = /Users/username/pcalc_software/pdu202009Du.geotess

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

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

dbInputInstance = jdbc:oracle:thin:@domain:port:database
dbInputUserName = username
dbInputPassword = password

dbInputTablePrefix = uebgt_
dbInputTableTypes = origin, arrival, assoc
dbInputSiteTable = uebgt_site
dbInputWhereClause = origin.orid = 48834027


#=================================================
#
# 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 7. PCalc Prediction Properties File pcalc_predictions_db.properties.

The user must update the properties file to provide working database access information for `@domain:port:database`, `username`, and `password`. The input table name prefix should be replaced with the name of database tables to which the user has access. Note that PCalc is designed to interact with the CSS3.0 schema or similar format.

4.3. PCalc Raypath Generation Using Great Circles

The properties file **Example08/pcalc_raypaths_greatcircle.properties** (Figure 8) contains input parameters to generate predictions for locations specified by a great circle.

Much of this example is the same as those shown in previous sections. Thus, we will only discuss new properties here. Note that as before, inputType can be file (Section 4.2.1), greatcircle (Section 4.2.2), grid (Section 4.2.3), or database (Section 4.2.4). In this example, inputType is set to greatcircle:

```
inputType = greatcircle
```

Note that this great circle is defined with the additional properties gcNpoints (Section A.2.4.5), which defines the number of points that will be placed along the great circle path, and gcOnCenters (A.2.4.7), which will cause 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 will coincide with the beginning and end of the great circle.

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

workDir=/Users/username/pcalc/Examples/Example08/

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

benderModel= <property:workDir>../SALSA3D_Model/SALSA3D_PUBLIC.geotess
#============================================================
#
# 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
```

Figure 8. PCalc Prediction Properties File pcalc_raypaths_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=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
#=============================================================
#
# 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.

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. Setting gcOnCenters to false is default and is shown here for demonstration purposes only.

In this example, the user still specifies that the application perform predictions. However, rather than setting outputAttributes to travel_time as seen in the previous predictions examples, we now set the outputAttributes property (Section A.2.8.4) to ray_path:

```
outputAttributes = ray_path
```

Thus, unlike in prior predictions examples where the output file contains a list of event locations and their travel times to a specified receiver, the output file will now contain raypath geometries. These geometries represent rays calculated by the predictor (here Bender) for the specified phase (here P) that travel through the input model (here SALSA3D_v2_PS.geotess) from the specified origins (defined by a great circle here) to the specified receiver (here ARCES).

Finally, note that the outputType (Section A.2.8.6) is explicitly set to file:

```
outputType = file
```

A sample of the output when PCalc is run on this properties file is shown below:

```
latitude longitude distance depth
>
0.000000 -90.000000 0.000000 -0.000207
0.000148 -89.999304 0.000711 1.226326
0.000184 -89.999141 0.000878 1.440761
0.001282 -89.994334 0.005807 3.644488
. . .
10.000131 -0.001974 89.998056 1.006994
>
```

One raypath consists of all rows between two arrows >, as shown in the above example. In this example, gcPositionParameters was set to "latitude longitude distance depth". These can be changed to x, y, z output or some other combination (see Sections 4.2.2, A.2.4.8 for details).

4.4. PCalc LibCorr3D Surface Generation

The properties file **Example09/pcalc_predictions_geotess.properties (shown below in Figure 9)** contains input parameters to generate LibCorr3D surfaces using PCalc. LibCorr3D surfaces are 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.

To perform LibCorr3D surface generation, a special salsa3d input folder is required. We provide an example of one of these folders in the following section. The folder and its contents are defined in Section 4. As we go through the example in Figure 9, the properties requiring this folder path will be defined.

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

application = predictions

# 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           = 196

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

workDir = /Users/username/Documents/pcalc/testing

logFile = <property:workDir>/pcalc_<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 = /Users/username/Documents/salsa3d/salsa3d_v2

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

# for DistanceDependent uncertainty, must specify the benderUncertaintyDirectory.
# A salsa3d model directory or a specific uncertainty directory can be specified.
# Ignored for PathDependent uncertainty
benderUncertaintyDirectory = <property:benderModel>
```

Figure 9. PCalc Prediction Properties File pcalc_predictions_geotess.properties.

```
# If benderUncertaintyType is PathDependent then benderUncertaintyModel is ignored.
# If benderUncertaintyType is DistanceDependent and the benderUncertaintyDirectory is
# a salsa3d model directory then a benderUncertaintyModel is ignored because the salsa3d directory
# will contain distance dependent uncertainty information for only one model.
# If benderUncertaintyType is DistanceDependent and the benderUncertaintyDirectory is not
# a salsa3d model directory then a benderUncertaintyModel must be specified and must point to a
# directory that contains valid travel time uncertainty information.

# benderUncertaintyModel = /Users/username/Documents/salsa3d/salsa3d_v2

# 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.  Feel free to delete or empty this directory anytime that PCalc is not using it.
# Ignored for DistanceDependent uncertainty

benderUncertaintyWorkDir = <property:workDir>/../BenderUncertaintyWorkDir

#============================================================
#
# INPUT PARAMETERS: station phase info
#
#============================================================

# jdate of the 'arrivals' is optional but if specified then bender will use the information to include
# tt_site_corrections in computed travel times.
jdate = 2011001

# site has to be included as a property for libcorr3d models. Many sites may be specified, separated
# by a semi-colon, and will be processed one at a time.
site = MJAR, -1, 2286324, 36.524717, 138.24718, .6617, "Japan", ar, MJAR, 0, 0

# when one or more sites are specified, sta will be automatically updated with site.sta
#sta = x

# phase
phase = P
supportedPhases = P,Pn
```

Figure 9 (cont). PCalc Prediction Properties File pcalc_predictions_geotess.properties.

```
#============================================================
#
# INPUT PARAMETERS: Input
#
#============================================================

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

# When inputType is geotess, a geotess grid and model will be constructed from scratch using
# properties defined below.  No geotess grid or model will be loaded from file.

# specify GeoTessBuilder parameters that control grid construction.
# See GeoTess User's Manual for more information about grid construction parameters
# For each grid construction parameter defined in the GeoTess Users Manual, capitalize the
# parameter name and prepend 'geotess'

geotessDataType = float

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

# the background grid resolution, in degrees.
geotessBaseEdgeLengths = 64

# 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 radius of the spherical cap
# will equal geotessActiveNodeRadius.  The tessellation index is always 0.
# Specify the grid resolution inside the spherical cap, in degrees.
# Substrings <site.lat> and/or <site.lon> are replaced with the latitude and longitude of the site
# currently being processed.
geotessPolygons = spherical_cap, <site.lat>, <site.lon>, <property:geotessActiveNodeRadius>, 0, 1

# PCalc has access to the geotess model seismicity_depth.geotess which contains the minimum and
# maximum depth of seismicity around the globe.  If depthSpacing is >= 0 then PCalc will use that
# information, along with the specified value of depthSpacing, to specify a bunch of depths that
# span the depth range from seismicityDepthMin to seismicityDepthMax.

# max depth spacing in the profile at each grid vertex, in km.
geotessDepthSpacing = 10
```

Figure 9 (cont). PCalc Prediction Properties File pcalc_predictions_geotess.properties.

```
# Seismicity Depth Model.  If not specified, internal default model is used.
#seismicityDepthModel = /sa/salsa3d/Location/LocOO3D/seismicity_depth_sm100.geotess

#============================================================
#
# OUTPUT PARAMETERS: GEOTESS OUTPUT
#
#============================================================

#outputType must = file | database | geotess | libcorr3d
outputType = libcorr3d

outputFile = <property:workDir>/<property:sta>_<property:phase>.geotess

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

# the following list of output attributes will be computed.
# For libcorr3d models, outputAttributes must = tt_delta_ak135, tt_model_uncertainty
outputAttributes = tt_delta_ak135, tt_model_uncertainty
```

Figure 9 (cont). PCalc Prediction Properties File pcalc_predictions_geotess.properties.

Several new properties can be seen in this properties file.

parallelMode can be either sequential or concurrent (the latter is recommended). When parallelModel is concurrent, maxProcessors specifies the maximum number of threads PCalc can use to compute predictions. The default is all threads.

Note that the predictors property must be set to bender to perform LibCorr3D surface generation. In addition, the benderUncertaintyType property (Section A.2.9.12) must be defined. benderUncertaintyType sets the type of travel time uncertainty desired. If set to DistanceDependent, the 1D distance dependent uncertainties in the salsa3d_v2 folder (see Section 4) will be used. Otherwise, if set to PathDependent, path dependent uncertainties will be calculated using the input velocity model's full covariance matrix. In this example, the benderModel and benderUncertaintyType are set to:

```
benderModel = /Users/username/Documents/salsa3d/salsa3d_v2

benderUncertaintyType = PathDependent
```

Note that unlike in prior examples, benderModel is set equal to the path to the special salsa3d model folder needed to generate LibCorr3D surfaces. PCalc will automatically retrieve the model necessary for Bender to perform ray tracing from the folder.

With benderUncertaintyType set to PathDependent, the benderUncertaintyDirectory property (Section A.2.9.13) does not need to be set.

Next, the benderUncertaintyWorkDir property (Section A.2.9.15) is defined. This property is only necessary when computing path dependent uncertainties. Thus, it is ignored in this example which calculates distance dependent uncertainty. But for demonstration purposes it is included in the example properties file and is set to:

```
benderUncertaintyWorkDir = <property:workDir>/../BenderUncertaintyWorkDir
```

This directory can be removed or emptied anytime it is not in use by PCalc.

Next, the inputType property is defined to be geotess:

```
inputType = geotess
```

When defining which sites to generate a LibCorr3D surface for, all parameters are the same as in prior prediction examples except for the new property supportedPhases. supportedPhases defines which phases the LibCorr3D surface can be used for. For instance, here supportedPhases is defined as:

```
supportedPhases = P,Pn
```

Thus, any surfaces generated by this example can be used for phases P and Pn.

Because a LibCorr3D surface is contained in a GeoTess model, the model must be built and populated with GeoTessBuilder, the GeoTessBuilder properties geotessBaseEdgeLengths, geotessRotateGridtoStation, geotessDataType, geotessModelDimensions, geotessBaseEdgeLengths, geotessActiveNodeRadius, geotessPolygons, and a depth spacing option (geotessDepthSpacing or geotessDepths) must be defined. The reader is referred to the GeoTess documentation to learn more about these properties.

Finally, the outputType property (Section A.2.8.6) is set to libcorr3d and the output file is set to generate a GeoTess file containing the outputAttributes tt_delta_ak135 and tt_model_uncertainty defined at the beginning of this section (also see Section A.2.8.4). Note that the property overwriteExistingOutputFile (Section A.2.8.2) is set to true:

```
outputType = libcorr3d

outputFile = <property:outputDir>/<property:sta>_<property:phase>.geotess

overwriteExistingOutputFile = true

outputAttributes = tt_delta_ak135, tt_model_uncertainty
```

The output of this example file consists of two GeoTess files named MJAR_P.geotess. These files can be queried and used via GeoTess, which includes the capability to generate .vtk figures that can be viewed in ParaView.

4.4.1. seismicBaseData

PCalc often needs access to 2-dimentional 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 sesimicBaseData 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 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 will be used.

5. SUMMARY

PCalc is a feature-rich software application that can compute travel-time predictions, ray path geometries 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 (available at https://github.com/sandialabs/GeoTessJava **Error! Hyperlink reference not valid.**and https://github.com/sandialabs/LocOO3D respectively) 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/PCalc
or
www.sandia.gov/salsa3d/Software.html

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.

APPENDIX A. PCALC PARAMETERS

A.1. Setting Parameters

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

NOTES:

- PCalc parameters are case sensitive.
- All parameters in the parameter file must contain an '=' character, separating the parameter name from the parameter 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 and terminalOutput, must be specified in all properties files. Otherwise, PCalc will throw an error.

A.2.1.1. application

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

A.2.1.2. 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.2.1.3. 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.2.2. 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 following input properties 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

< > [no Default]

Used to specify the location of one or more sites. Several format 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

Note that several sites can be specified with the site property, delimited by a semi-colon. When more than one site are specified, they are processed one at a time as though PCalc was run multiple time with each site separately.

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

A.2.2.4. phase

<String> [no Default]

Seismic phase.

A.2.2.5. supportedPhases

<String> [no Default]

Comma-separated list of phases that are supported by the LibCorr3D surface. Only used when generating LibCorr3D surfaces.

A.2.2.6. 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 input parameters.

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 Section 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 Section A.2.7.

A.2.4. Input from Great Circle

If *inputType* = greatcircle then this section defines properties that further define the input parameters. 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 Section A.2.7 for how to specify the depth(s) of the points along the greatcircle.

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 in order 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 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 input parameters. 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 Section 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 parameter.

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

A.2.6.6. dbInputTablePrefix

<string> [Default none]

If this parameter is specified then the four input tables (dbInputOriginTable, dbInputAssocTable, dbInputArrivalTable, dbInputSiteTable) will default to the value of this parameter 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 parameter will override any default values set by other parameters.

A.2.6.8. dbInputAssocTable

<string> [Default not allowed]

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

A.2.6.9. dbInputArrivalTable

<string> [Default not allowed]

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

A.2.6.10. dbInputSiteTable

<string> [Default not allowed]

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

A.2.6.11. dbInputWhereClause

PCalc will execute a sql query similar to:

```
select origin.*, assoc.*, arrival.*, site.*
   from leb_origin origin, leb_assoc assoc, leb_arrival arrival, idc_site site,
       idc_affiliation affiliation
   where origin.orid=assoc.orid and assoc.arid=arrival.arid and
       arrival.sta=site.sta
       and arrival.jdate greater than or equal to 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. Users can specify a where clause string using this property.

A.2.6.12. dbOutputAssocTable

<string> [Default = none]

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

A.2.6.13. dbOutputAutoTableCreation

<bool> [Default = false]

Boolean flag should be set to true if output database tables should be created if they do not already exist.

A.2.6.14. dbOutputTruncateTables

<bool> [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* parameter has been set to false, the user will be prompted before table truncation actually occurs.

A.2.6.15. dbOutputPromptBeforeTruncate

<bool> [Default = true]

If *dbOutputTruncateTables* is true and this parameter is true, then the user is prompted before output table truncation actually occurs. If *dbOutputTruncateTables* is true and this parameter 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 parameter 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:

- topography
- 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_2.0 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, eg. 'moho', is equivalent to specifying 'top of moho' or 'below moho'. If 'topography' is specified, then property *topographyModel* is required and should have a value that corresponds to the path to the desired topography model file.

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

"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'

SALSA3D.2.0 has the following layers/interfaces defined:

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

A.2.8. Output Parameters

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_pth' 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. True by default.

A.2.8.3. separator

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

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

A.2.8.4. 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.2.8.5. 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 model predicted travel-times)
    - Note that despite the name, the model being compared to does not need to be ak135. For instance, the difference between 3D GeoTess model predicted travel-times and RSTT model predicted travel-times can be compared.
- tt_model_uncertainty (represents either the distance-dependent or the path-dependent travel-time uncertainty of the input 3D GeoTess model in seconds)

A.2.8.6. outputType

<string> [no Default]

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

A.2.9. 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.2.10. 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* is required and other properties in this section that pertain to one of the predictors listed in *predictors* are also required.

A.2.10.1. predictors

<string> [Default = none] (lookup2d, bender, slbm)

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

A.2.10.2. maxProcessors

<int> [Default = all available processors]

All predictions are computed in concurrent parallel mode (multi-threaded). To limit the number of processors that PCalc will use to compute predictions, specify the desired number with this property.

A.2.10.3. batchSize

<int> [Default = 10,000]

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 parameter is ignored.

A.2.10.4. lookup2dModel

<string> [Default = ak135] (ak135)

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

A.2.10.5. 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 parameter is specified then the next two parameters, *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory*, are not required.

A.2.10.6. 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.2.10.7. lookup2dEllipticityCorrectionsDirectory

<string> [Default = none] ()

Path of the directory where ellipticity correction coefficients are located for use with the Lookup2D predictor. LocOO3D will throw an exception if this parameter is not specified and tauptoolkit is one of the options specified in parameter loc_predictor_type. A recommended value is <SNL_Tool_Root>/seismicBaseData/el/ak135.

A.2.10.8. lookup2dUseEllipticityCorrections

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

If false, then ellipticity corrections are not applied.

A.2.10.9. lookup2dUseElevationCorrections

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

If false, then elevation corrections are not applied.

A.2.10.10. lookup2dSedimentaryVelocity

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

Seismic velocity used in the calculation of elevation corrections.

A.2.10.11. 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 4.4 for details about salsa3d directories).

A.2.10.12. benderUncertaintyType

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

Type of travel time uncertainty desired.

A.2.10.13. benderUncertaintyDirectory

<string> [Default = none] ()

Directory where distance dependent uncertainty values 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.2.10.14. benderUncertaintyModel

<string> [Default = none] ()
Subdirectory where distance dependent uncertainty values 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.2.10.15. 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.2.10.16. 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.2.10.17. benderAllowCMBDiffraction

<boolean> [Default = false]

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

A.2.10.18. 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.2.10.19. 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.2.10.20. 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.2.10.21. 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.2.11. LibCorr3D

A.2.11.1. <predictor>PathCorrectionsType

<string> [Default = none] (libcorr)

<predictor> is lookup2d or slbm.

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

A.2.11.2. <predictor>LibCorrPathCorrectionsRoot

<string> [Default = none]

<predictor> is lookup2d or slbm.

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

A.2.11.3. <predictor>LibCorrPathCorrectionsRelativeGridPath

<string> [Default = "."]

<predictor> is lookup2d or slbm.

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

A.2.11.4. <predictor>LibCorrInterpolatorType

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

<predictor> is lookup2d or slbm.

Type of horizontal interpolation to use.

A.2.11.5. <predictor>LibCorrPreloadModels

<boolean> [Default = false]

<predictor> is lookup2d or slbm.

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

A.2.11.6. <predictor>UsePathCorrectionsInDerivatives

<boolean> [Default = false]

<predictor> is lookup2d or slbm.

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

A.2.12. Model Queries

A.2.12.1. benderModel

<string> [Default = none] ()

Path to GeoTessModel that Bender should should query for model values. This can either be a file or a special salsa3d directory (see section 4.4 for details about salsa3d directories).

A.2.13. Ray Path Geometry

PCalc can compute and output the geometry of ray paths through the SALSA3D model using Bender. In order for this 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.2.13.1. 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

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 section B.1 for explanation.
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 section B.2 for explanation.
3. To build a custom grid that can be shared by multiple libcorr3d models
    a. Use GeoTessBuilder to construct the common grid
    b. Do not specify any rotation parameters, which will result in a gird 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 section B.3 for explanation.

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

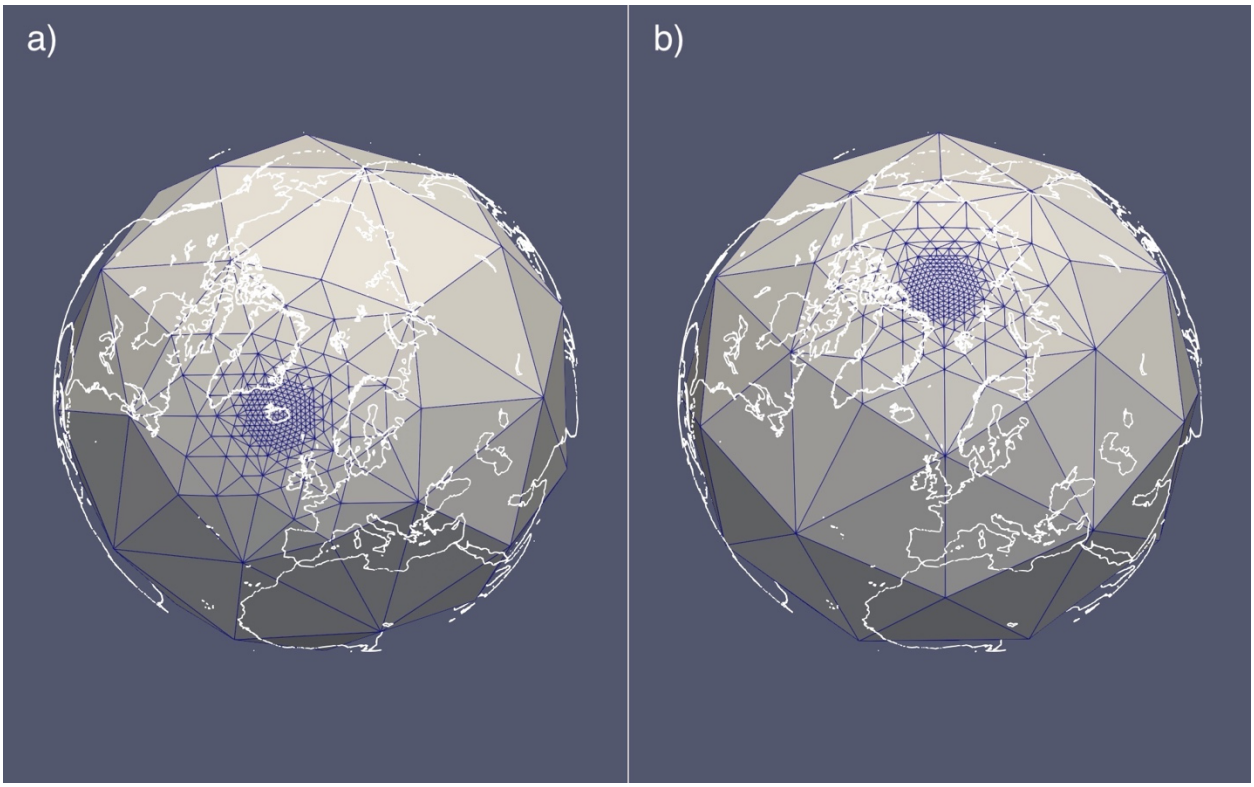


Figure B1 – a) a custom grid for a station in Iceland. b) a custom grid centered on the north pole that can be shared among multiple stations when appropriate rotations are applied at runtime.

B.1.1. 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, shown in Figure B1a, will have default grid resolution of 64 degrees far from the site and 4 degree 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.1.2. 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 GeoTessBulder 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.1.3. 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 GeoTessBulder application. Example GeoTessBuilder properties for building a typical grid like the one in Figure B1b 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 B1b). 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. Problem

The user wants to produce libcorr3d surfaces for a large number of stations. The user has a number of computers to use to compute the surfaces but, without Fabric, pcalc can only compute each surface on a single computer in concurrent mode. It typically takes multiple hours to compute a single surface in this manner. To speed things up, the user would like to launch pcalc runs on multiple computers to compute the surfaces in parallel.

C.2. Solution

The user creates a file with station information for each desired surface. 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 ...
NRIK   -1     2286324      69.33958     87.55504     0.2150 Norilsk ...
NRIS   -1     2286324      69.00610     87.99640     0.4980 Norilsk, Russia ...
NVAR   -1     2286324      38.42961     -118.30355   2.0416 MINA ARRAY ELEMENT ...
```

Each record contains sta, ondate, offdate, lat, lon, elevation, staname, statype, refsta, dnorth and deast, separated by tabs. Note that station definition records have been truncated in this document for clarity.

Note that the contents of this file will be modified during processing so the user must ensure that a copy of the information exists elsewhere. The user should also refrain from modifying the file in any way while processing is proceeding. It is ok to view the contents of the file, or copy the file to another file for inspection.

In the pcalc properties file, instead of specifying sites with the `site` property, the user specifies `siteFile` = path to the site file described above.

Then the user can launch pcalc with this properties file on multiple computers. There is no need to modify the properties file between launches on different computers.

Each instance of pcalc will read the first record in the file that does not begin with the comment character '#' and insert a '#' character at the beginning of that record in the file. It will also write some additional information to the file, each record of which begins with a '#' character. During these operations, a lock is placed on the file so that only one instance of pcalc can access the file at a time.

After pcalc has been launched on 3 computers, the file will contain

```
#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   zinger01 began    2022-10-19 09:48:56 -0600
# MSKU   twinkie1 began    2022-10-19 09:49:26 -0600
# MSVF   snoball1 began    2022-10-19 09:49:39 -0600
```

Note that the records for 3 stations, MSEY, MSKU and MSVF have been commented out with a '#' character and records have been added indicating that processing has begun for those three stations on computers `zinger01, twinkie1 and snoball1`.

A couple of hours later, processing of station MSVF on `snoball1` has completed and `snoball1` has begun processing station NEW. The siteFile looks 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 ...
# MSEY   zinger01 began    2022-10-19 09:48:56 -0600
# MSKU   twinkie1 began    2022-10-19 09:49:26 -0600
# MSVF   snoball1 began    2022-10-19 09:49:39 -0600
# MSVF   snoball1 finished 2022-10-19 11:41:49 -0600 ( 1.869535 hours)
# NEW    snoball1 began    2022-10-19 11:41:49 -0600
```

Approximately 10 hours after processing began libcorr3d surfaces for all 10 stations has been completed and the siteFile contains:

```
#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   zinger01 began    2022-10-19 09:48:56 -0600
# MSKU   twinkie1 began    2022-10-19 09:49:26 -0600
# MSVF   snoball1 began    2022-10-19 09:49:39 -0600
# MSVF   snoball1 finished 2022-10-19 11:41:49 -0600 ( 1.869535 hours)
# NEW    snoball1 began    2022-10-19 11:41:49 -0600
# MSEY   zinger01 finished 2022-10-19 12:16:04 -0600 ( 2.452057 hours)
# NIL    zinger01 began    2022-10-19 12:16:04 -0600
# MSKU   twinkie1 finished 2022-10-19 12:58:38 -0600 ( 3.153214 hours)
# NNA    twinkie1 began    2022-10-19 12:58:38 -0600
# NEW    snoball1 finished 2022-10-19 13:54:23 -0600 ( 2.209253 hours)
# NOA    snoball1 began    2022-10-19 13:54:23 -0600
# NOA    snoball1 finished 2022-10-19 13:54:23 -0600 ( 0.012000 seconds)
# NIL    zinger01 finished 2022-10-19 15:41:49 -0600 ( 3.429111 hours)
# NNA    twinkie1 finished 2022-10-19 17:07:06 -0600 ( 4.141127 hours)
```

Above, the contents of the file appear in chronological order, which is a bit difficult to interpret since the results for individual stations are jumbled up. If the file is sorted alphabetically, it will look like:

```
# MSEY   zinger01 began    2022-10-19 09:48:56 -0600
```

```
# MSEY   zinger01 finished 2022-10-19 12:16:04 -0600 ( 2.452057 hours)
# MSKU   twinkie1 began    2022-10-19 09:49:26 -0600
# MSKU   twinkie1 finished 2022-10-19 12:58:38 -0600 ( 3.153214 hours)
# MSVF   snoball1 began    2022-10-19 09:49:39 -0600
# MSVF   snoball1 finished 2022-10-19 11:41:49 -0600 ( 1.869535 hours)
# NEW    snoball1 began    2022-10-19 11:41:49 -0600
# NEW    snoball1 finished 2022-10-19 13:54:23 -0600 ( 2.209253 hours)
# NIL    zinger01 began    2022-10-19 12:16:04 -0600
# NIL    zinger01 finished 2022-10-19 15:41:49 -0600 ( 3.429111 hours)
# NNA    twinkie1 began    2022-10-19 12:58:38 -0600
# NNA    twinkie1 finished 2022-10-19 17:07:06 -0600 ( 4.141127 hours)
# NOA    snoball1 began    2022-10-19 13:54:23 -0600
# NOA    snoball1 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 ...
```

Note that station NOA completed processing in less than 1 second instead of multiple hours. This is because libcorr3d surfaces for that station had been computed in a previous run and property `overwriteExistingOutputFile = false`.

C.3. What if one or more machines go down or processing is aborted in the middle of a run

Consider the following scenario: the user wants to compute surfaces for 100 stations, S1 to S100, using 10 computers named M1 to M10. They create a siteList file with the information about the 100 stations, specify that file in a properties file using the `siteFile` property. They then visit each of the 10 computers and launch pcalc with the same properties file. Processing all 100 stations should take approximately 24 hours (100 stations * ~2.4 hours / station / 10 computers). Now, let's say that after about 10 hours, machine M2 goes down or the user aborts the pcalc job on that machine. At the time, M2 is in the middle of computing a surface for station S30. This is fine, all the other machines keep processing and process all the other stations. It is also fine to restart processing on machine M2. At the conclusion of processing, surfaces for all the stations will have been computed except the surface for station S30. If multiple machines had stopped processing (and maybe restarted), it is possible that surfaces for multiple stations would not have been processed. To recover, the user can modify the siteList file by restoring it to its original contents (including information for all 100 stations). They then set property `overwriteExistingOutputFile = false` and restart pcalc on 1 or more machines. PCalc will skip processing of all stations that already have output files in the output directory and will only process the stations that were not processed in the first round of processing.

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

Email—Internal

Name	Org.	Sandia Email Address
Stephanie Teich-McGoldrick	06756	steichm@sandia.gov
John Merchant	06752	bjmerch@sandia.gov
Rigobert Tibi	06752	rtibi@sandia.gov
Steve Vigil	06752	srvigil@sandia.gov
Andrea Conley	06752	acconle@sandia.gov
Kathy Davenport	06756	kdavenp@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.