SANDIA REPORT
SAND2019-10171
Printed August 2019

Sandia National Laboratories

pCalc User's Manual

Nathan J. Downey[1], Sanford Ballard[1], James R. Hipp[1] and Mike 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 is available on the web at:

www.sandia.gov/salsa3d/Software.html

The software is packaged with this user's manual and a set of example datasets, the use of which is described in this manual.

This page left blank

CONTENTS

LIST OF FIGURES

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
CSS	Center for Seismic Studies
SALSA3D	Sandia-Los Alamos 3D velocity model
GeoTESS	Geographical Tessellation Software
LocOO3D	Object-Oriented 3D Location Software
3D	Three dimensional

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) and 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 the SALSA3D website at

www.sandia.gov/salsa3d/Software.html

and is packaged with the latest version of this user's manual, a set of run examples (the use of which is outlined on the "Tutorial and Examples" section below) and a jar file of the pCalc software. pCalc requires the java runtime environment 1.8 or later to be installed in a user's machine.
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 the phases P, PP, pP, PKPdf, PBPbc, 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.

- 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. A 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) and is the same ray tracer/ travel time predictor used in the construction of the SALSA3D velocity models. 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)
Some advantages of the pseudobending approach used in bender include:

- The algorithm finds a raypath 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 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 www.sandia.gov/salsa3d/Software.html) 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 TUTORIAL AND EXAMPLES

Included with the pCalc distribution is a set of example input files that demonstrate how pCalc works. The required input files for a pCalc run are:

1. A required properties (*.properties) file which contains the parameter settings used by pCalc for a particular run

2. An optional GeoTess (*.geotess) file containing the Earth velocity model. If this file is not specified, then pCalc will compute travel times using the AK135 lookup tables included within the pCalc software.

3. An optional 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.

The format of the properties file, as well as a list of all the possible properties that can be set is described in the Appendix.
Several simple example inputs are included in the "Examples" directory of the pCalc distribution. Here we will describe a few of these runs in some detail, the other examples run "out of the box" and should be easily understandable once a user has worked through the next section.

3.1. pCalc "query" Example

The properties file "example_pcalc_query_file.properties" (Figure 1) contains input parameters to query a model at locations specified in the file "MJAR.coords.xyz" also included in the "Examples" directory. All of the examples in this directory can be run using the command:

```
$java -jar pCalc.jar <parameter_file_basename>.parameters
```

Where "<parameter_file_basename>" is replaced by the basename of the example you wish to run, which for this first example is "examples_pcalc_query_file". In order to run this example, you will need to download the SALSA3D model from www.sandia.gov/salsa3d, ensuring that the file "SALSA3D.geotess" is contained in the "Examples" directory
Many of the parameters specified in this file are common to all the examples, so we will discuss them here:

```
maxProcessors = 4

application = model_query

workDir = ~/Examples/
```

These three parameters set the maximum number of processors to be used in the pCalc run, the type of query, which can be set to "predictions" or "model_query" if you are predicting raypaths or travel times or, as in this case, extracting model properties. The "workDir" property sets the working directory for this example (here we assume the Examples directory is in the user's home directory.

This parameter will have to be reset to the actual location of the "Examples" directory on a user's system).

```
#================================================================
#
# Property file for application PCalc v. 2.0
#
#================================================================

maxProcessors = 4

application = model_query

workDir = ~/Examples/

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

geotessModel = <property:workDir>SALSA3D.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>MJAR.coords.xyz

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

inputAttributes = latitude longitude depth

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

outputType = file
outputFile = <property:workDir>pcalc_query_file_output.dat

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

terminalOutput = true

separator = space

outputAttributes = pslowness
```

Figure 1: The properties file "example_pcalc_query_file.properties" as included in the directory "Examples"

```
predictors = bender
```

This property specifies that for a prediction run we are going to use the pseudobending library within pCalc to make any predictions, jhowever for the "model_query" case run here this parameter is ignored.

```
geotessModel = <property:workDir>SALSA3D.geotess
```

Specifies the path to the GeoTESS model we are going to query during this run. The notation <property:*property_name*> specifies that the value for a property specified elsewhere in the property file is substituted.

```
inputType = file
```

pCalc can use several different methods to specify where a model is to be queried or to which travel time or raypath predictions are to be computed. These include "file" in which these locations are specified manually in an ascii file, "greatcircle" in which the locations are specified as a grid along a great circle through the Earth or "grid" in which locations are specified on a regular latitude, longitude, depth grid.

```
inputFile = <property:workDir>MJAR.coords.xyz
```

For this run, the coordinates are specified in the file MJAR.coords.xyz, included in the "Examples" directory.

```
batchSize = 10
```

pCalc processes input records in groups, this parameter specifies the group size.

```
inputAttributes = latitude longitude depth
```

This property specifies the format of the input coordinates contained in the inputFile.

```
outputType = file

outputFile = <property:workDir>/pcalc_query_file_output.dat

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

These three properties specify whether the output should be written to an output file instead of standard out, the name of the output file and the name of an optional log file.

```
terminalOutput = true

separator = space

outputAttributes = pslowness
```

These properties specify if log information is to be printed to the terminal, what the output separator ("space", "comma" or "tab") is and the model attribute to be output in the output file.

3.2. pCalc "predictions" Example

The file "example_pcalc_predictions_file.properties" contains the input properties for a pCalc run which computes travel times from a specified station location to a set of coordinates contained in an input file. Most of the parameters are the same as for a "query" run with some exceptions:

```
predictors = bender
```

This property specifies that we are going to use the built-in raytracer to compute predictions. Set this parameter to "lookup2d" if you want to make predictions using AK135.

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

The coordinates in the input file ("MJAR_coords.xyz" for these examples) are now interpreted to be a set of hypocenters of event origins. These form the starting point for travel time predictions computed using bender to a station location which is specified by the properties:

```
phase = P

site = 36.524717, 138.24718, .6617

sta = MJAR

jdate = 2011001
```

The first two of these properties, specifying the station coordinates (in latitude, longitude and elevation in km) and the desired phase are required, while the second two are optional, but if there are site terms for the specified station in the GeoTESS model, these will be applied to the predictions.

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

This property specifies the name of the output file to which the predictions are output. This file has the same format as that for the input file, however the predicted property is pasted as a new column at the end of each line.

3.3. pCalc "raypaths" Example

pCalc uses the bender predictor to output ray paths through a 3D velocity model. In this mode, the raypath start and endpoints can be specified in a variety of means, which are the same for those in "prediction" mode. Methods by which the coordinates are specified are:

1. "inputType=file" is that used in the above examples, where an input file is used to specify the coordinates of proposed event hypocenters. Predictions and ray paths are specified between these locations and a station location specified in the properties file

2. "inputType=greatcircle" is the input method used in this example, in which the property "gcStart" specifies one end of the ray path and either "gcEnd" or "gcDistance" and "gcAzimuth" are used to specify a second point on a great circle on which "gcNpoints" endpoints are regularly spaced.

3. "inputType=grid" is an input mode in which a regular grid is specified using the "gridRangeLat" and "gridRangeLon" properties. Each of these is set to a set of three space-delimited numbers representing the start, endpoint and number of points in each dimension of the grid.

In conjunction to these methods for specifying latitude and longitudes for endpoints the depth specification for the endpoints can also be specified in a variety of ways:

1. "depthSpecificationMethod=depths" indicates that the depths to which rays are computed for each grid point are specified directly using the "depths" property.

2. "depthSpecificationMethod=depthRange" specified depths using a minimum depth, maximum depth and number of depth points as a triplet of numbers assigned to the "depthRange" property

3. "depthSpecificationMethod=depthLevels" computes rays to depths above and below a specified layer in the model

4. "depthSpecificationMethod=maxDepthSpacing" computes rays to the top and bottom of each layer as well as within each layer at a spacing specified by the "maxDepthSpacing" property to a maximum depth set by the "maxDepth" property.

In each of these methods rays are computed between the station location and the specified points. If we are computing ray paths, bender computes the ray path geometry for rays that travel from the specified origin to the specified grid points and outputs these geometries into an output file. For the "prediction" mode these ray paths are used to compute a travel-time prediction which is output to a file instead.
An example of ray path computation using the "greatCircle" and "maxDepthSpacing" geometry specifications is given in the example file "example_pcalc_raypaths_greatcircle.properties" file included in the "Examples" directory.

This page left blank

4. SUMMARY

pCalc is a feature-rich software application that can compute travel-time prediction, 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.

- 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.

When pCalc is used in conjunction with the GeoTESS and LocOO3D tools (available at www.sandia.gov/geotess and www.sandia.gov/salsa3d/Software.html 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 the SALSA3D website at:

www.sandia.gov/salsa3d/Software.html

This page left blank

REFERENCES

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.

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.

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.

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*

A.2.1.1. application

<string> [] (model_query | predictions)

Set this property to the desired type of application.

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)
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.

A.2.2.3. site

<3 doubles> [no Default]
The latitude in degrees, longitude in degrees, and elevation in km, of a station.

A.2.2.4. phase

<String> [no Default]
Seismic phase.

A.2.2.5. 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.

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 **Depth Specification**.
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 **Depth Specification**.

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 "Depth Specification" 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.
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 "Depth Specification" 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 first longitude, last 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.1.6 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.1.6 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 application = database, required otherwise.

A.2.8.2. separator

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

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

A.2.8.3. outputFormat

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

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

A.2.8.4. outputAttributes

<string> [no Default]

The attributes that should be sent to output.
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)
For ray path geometries specify 'ray_path'

A.2.8.5. outputHeader

<boolean> [default false]

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

A.2.9. *Predictors*

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

A.2.9.1. predictors

<string> [Default = none] (lookup2d, tauptoolkit, 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.2.9.2. 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.2.9.3. 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.2.9.4. lookup2dModel

<string> [Default = ak135] (ak135)

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

A.2.9.5. seismicBaseData

<string> [Default = none] ()

Path to the seismicBaseData directory. If this parameter is specified then the next two parameters, *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory*, are not required.

A.2.9.6. 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.2.9.7. 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.2.9.8. lookup2dUseEllipticityCorrections

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

A.2.9.9. lookup2dUseElevationCorrections

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

A.2.9.10. lookup2dSedimentaryVelocity

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

A.2.9.11. benderModel

<string> [Default = none] ()
Path to GeoTessModel that Bender should use to calculate predictions of seismic observables.

A.2.9.12. benderUncertaintyType

<string> [Default = UncertaintyNAValue] (UncertaintyNAValue, UncertaintyDistanceDependent)
Type of travel time uncertainty desired. If UncertaintyNAValue is specified (default), then all requests for travel time uncertainty return the NA_VALUE (-999999.). If UncertaintyDistanceDependent is specified then distance dependent uncertainty is returned.

A.2.9.13. 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.2.9.14. 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.2.9.15. benderAllowMOHODiffraction

<boolean> [Default = false]
If a crustal ray (Pg, Lg) impinges on the Moho, and this property is false, then the ray will be invalid.

A.2.9.16. 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.2.9.17. use_tt_model_uncertainty

<boolean> [Default = true]
If true travel time residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.18. use_az_model_uncertainty

<boolean> [Default = true]
If true azimuth residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.19. use_sh_model_uncertainty

<boolean> [Default = true]
If true slowness residuals and derivatives are weighted by the total uncertainty which consists of a combination of the model uncertainty and the pick uncertainty. If false, only the pick uncertainty is used.

A.2.9.20. use_tt_site_terms

<boolean> [Default = true]
If true then travel time site terms computed for each station during tomography are applied to computed values. The site terms are stored in GeoTessModels read by Bender.

A.2.10. *LibCorr3D*

A.2.10.1. <predictor>PathCorrectionsType

<string> [Default = none] (libcorr)
<predictor> is lookup2d.
Set the value to 'libcorr' to apply libcorr3d corrections. If this parameter is omitted, then corrections will not be applied.

A.2.10.2. <predictor>LibCorrPathCorrectionsRoot

<string> [Default = none]
<predictor> is lookup2d.

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

A.2.10.3. <predictor>LibCorrPathCorrectionsRelativeGridPath

<string> [Default = "."]
<predictor> is lookup2d.
The relative path from the directory where the correction surface files reside to the directory where the grid files reside.

A.2.10.4. <predictor>LibCorrInterpolatorType

<string> [Default = "linear"] (linear | natural_neighbor)
<predictor> is lookup2d.
Type of horizontal interpolation to use.

A.2.10.5. <predictor>LibCorrPreloadModels

<boolean> [Default = false]
<predictor> is lookup2d.
Whether all libcorr models should be loaded at startup or loaded on an 'as needed' basis.

A.2.10.6. <predictor>UsePathCorrectionsInDerivatives

<boolean> [Default = false]
<predictor> is lookup2d.
Whether or not path corrections should be included in total values when computing derivatives of travel time with respect to source location.

A.2.11. *Model Queries*

A.2.11.1. benderModel

<string> [Default = none] ()
Path to geotessModel that PCalc should query for attribute values. This should point to a SALSA3D model.

A.2.12. *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. *rayPathNodeSpacing*

<double> [Default = -1]

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

DISTRIBUTION

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name
Mike Begnaud	mbegnaud@lanl.gov	Los Alamos National Laboratory
Leslie Casey	leslie.casey@nnsa.doe.gov	National Nuclear Security Administration
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
Nathan Downey	06752	njdowne@sandia.gov
Technical Library	01177	libref@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.