Prediction Calculator (PCalc) Version 3.2

Sandy Ballard
Sandia National Laboratories
sballar@sandia.gov

August 26, 2019

Introduction

PCalc is software to perform 3 primary functions:

1. compute predictions of travel time, azimuth, slowness and other predicted values at user specified source-receiver positions, or
2. extract model values from SALSA3D at user specified positions.
3. compute ray path geometries through SALSA3D models.

PCalc can accept user specified positions from a number of input sources:

1. an ascii text file
2. a 1D grid of points distributed along a great circle path, optionally extended to 2D by the specification of additional depth information.
3. a 2D grid of points in map view, optionally extended to 3D by the specification of additional depth information.
4. a set of parameters that describe how to construct a GeoTessModel.
5. when requesting predictions, but not model queries, the geometry can come from a set of database tables. In this special case, PCalc will read information from specified origin, assoc, arrival and site tables and produce a new assoc table that has new timeres, azres and slores information computed using a specified predictor (tauptoolkit, bender, slbm, etc).

PCalc will output results to an ascii file or to a GeoTessModel, except when input comes from a set of database tables, in which case output goes to a database table.

Setting Parameters:

Many sample property files are provided in directory PCalc/examples.

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:

1) PCalc parameters are case sensitive.

2) 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).

3) 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'.

4) If a property value contains the string '<env:xyz>' then the phrase '<env:xyz>' is replaced with the value returned by System.getenv(xxx).

General

application

<string> [] (model_query | predictions)

Set this property to the desired type of application.

logFile

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

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

terminalOutput

<boolean> [Default = true]

Echo general information about the PCacl run. This is the same information that is sent to the logFile. If false, PCacl is silent.

Input

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*:

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.

site

<3 doubles> [no Default]

The latitude in degrees, longitude in degrees, and elevation in km, of a station.

phase

<String> [no Default]

Seismic phase.

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.

Input from File

If *inputType* = file then this section defines properties that further define the input parameters.

inputFile

<String> [no Default]

The name of the file that is to be input.

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*.

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**.

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.

gcStart

<2 doubles> [no Default]

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

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.

gcDistance

<double> [no Default]

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

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.

gcNpoints

<int> [no Default]

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

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.

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.

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.

depthFast

<boolean> [true]

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

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.

gridRangeLat

<2 doubles, 1 int> [no Default]

The minimum latitude, maximum latitude and number of latitudes.

gridRangeLon

<2 doubles, 1 int> [no Default]

The first longitude, last longitude and number of longitudes.

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.

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.

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.

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.

depthFast

<boolean> [true]

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

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.

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.

Input from GeoTess

If property *inputType* is equal to *geotess* then PCalc will query the properties file for parameters that describe how to construct a GeoTessGrid and GeoTessModel and will populate the GeoTessModel with predictions.

geotessInputGridFile

<string> [ ]

If the model is to be built using an existing GeoTessGrid, specify the file containing the grid with this property. If *geotessInputGridFile* is specified then none of the geotessXXX parameters specified below are required.

geotessOutputGridFile

<string> [ ]

To save the GeoTessGrid to an output file, specify the file name here. If this parameter is specified, then the GeoTessGrid information will not be written into the GeoTessModel file specified by parameter *outputFile*; only a reference to the grid file will be included in the GeoTessModel file.

geotessModelDimensions

<int> [2 or 3]

This property is required. Specify whether to build a 2D or 3D model. If a 2D model is specified then all grid points in the model will be located at the surface of the Earth (sea level). Otherwise, there will be grid points below the surface of the Earth in regions where deep seismicity has been observed empirically. The depth of Earth seismicity is contained in an internal geotess model called seismicity_depth.geotess.

In areas of the Earth where deep seismicity has been observed, grid points will be distributed radially with density specified using property *geotessDepthSpacing*.

geotessDepthSpacing

<double> [Default none]

For 3D models (see property *geotessModelDimensions*) use this property to specify the maximum depth spacing of grid points in regions of the Earth where deep seismicity has been observed.

geotessDescription

<string> [Default = 'GeoTessModel constructed by PCalc containing predicted values']

A string specifying what will be stored in the 'description' field of the GeoTessModel output file. The specified description will be supplemented with information about the station and phase used to generate the predicitons.

geotessDataType

<string> [Default = float] (float or double)

Data type to use to store the predicted values in the output GeoTessModel.

geotessRotateGridToStation

<boolean> [Default = true]

If true then the GeoTessGrid will be rotated such that grid vertex zero will be located at the same location as the station, which is specified with property *site* defined above.

geotessInitialSolid

<string> [Default = icosahedron] (icosahedron, tetrahexahedron, octahedron, tetrahedron)

This property specifies the initial solid that defines the first level of each of the multi-level tessellations that will be included in the new grid.

geotessBaseEdgeLengths

<double> [Default none]

the background *triangle edge length*, in degrees, of the triangles in the top level tessellation. The value should be a power of 2, i.e., one of 64, 32, 16, 8, 4, 2, 1, 0.5, 0.25, 0.125, etc.

geotessPoints

<string> [Default = none]

specification of a list of geographic locations about which refinement is to take place. The supplied value is parsed as follows: First, the property value is split into substrings based on the semicolon character (';'). Each of these substrings defines a

single point about which refinement is to occur. Each substring is split on the comma character (‘,’) into a number of tokens.

- If the resulting array of strings contains 3 tokens, they are interpreted to be (1) a *file name*, (2) a *tessellation index*, and (3) a *triangle edge length*. Points are read from the specified file and the *multi-level tessellation* with the specified index will be refined around all the points to the specified *triangle edge length*.

- If the resulting array of tokens contains 5 elements, they are interpreted as follows:

  1. Either ‘*lat-lon*’ or ‘*lon-lat*’ defining the order of latitude and longitude in entries 4 and 5 below.

  2. The index of the *multi-level tessellation* to refine. (specify zero)

  3. The *triangle edge length* specifying how small the refined triangles around the point should be.

  4. Latitude or longitude of the point in degrees.

  5. Latitude or longitude of the point degrees.

geotessPaths

<string> [Default = none]

specification of lists of points that define paths. All triangles that contain any segment of the specified paths will be refined to the specified level. The property value is parsed as follows: First, the property value is split into substrings based on the semicolon character (‘;’). Each substring includes the specification of a single *path*. Each substring is split on the comma character (‘,’). The resulting array of tokens must contain 3 tokens, which are interpreted to be (1) a file name, (2) a *tessellation index*, and (3) a *triangle edge length*. Points are read from the specified *file* and the multi-level tessellation with the specified index will be refined around all the paths to the specified *triangle edge length*.

geotessPolygons

<string> [Default = none]

specification of one or more polygons. All triangles that have at least one corner inside one of the polygons will be refined. The property value is parsed as follows: First, the property value is split into substrings based on the semicolon character (‘;’). Each substring is the specification of a single *polygon*. Each substring is split on the comma character (‘,’). The resulting tokens are interpreted as follows:

- If the first token is equal to ‘*spherical_cap*’, then the remaining tokens are interpreted as:
  - latitude of the center of the spherical cap
  - longitude of the center of the spherical cap
  - horizontal radius of the spherical cap in degrees
  - tessellation index (use 0)
  - triangle edge length specifying the size of the triangles desired within the spherical cap, in degrees.

- Otherwise the tokens are interpreted as follows:
  - The name of a *file* containing the definition of a polygon. Files can be either an ascii file or a Google Earth kmz/kml file. Ascii files contain a list of points defining a closed polygon. Each point is specified as either a latitude-longitude or longitude-latitude pair, in degrees. Latitude-longitude order is the default, but if the file contains the line ‘lon-lat’, then points are assumed to be in lon-lat order. Latitude and longitude values can be

separated by either a comma or white space. Kml/kmz files contain a single polygon as defined by Google Earth.
- *tessellation* index (specify 0)
- *triangle edge length* specifying the size of the triangles desired within the polygon, in degrees.

geotessEulerRotationAngles

<string> [Default = none]

It is possible to rotate the triangular tessellation produced by GeoTessBuilder, relative to the traditional latitude, longitude grid, by providing 3 Euler rotations angles, in degrees. Given two coordinate systems xyz and XYZ with common origin, starting with the axis z and Z overlapping, the position of the second can be specified in terms of the first using three rotations with angles A, B, C as follows:

1. Rotate the xyz-system about the z-axis by A.
2. Rotate the xyz-system again about the now rotated x-axis by B.
3. Rotate the xyz-system a third time about the new z-axis by C.

Clockwise rotations, when looking in direction of vector, are positive.
Reference: http://mathworld.wolfram.com/EulerAngles.html
For example, to rotate the grid such that grid vertex 0, which normally coincides with the north pole, resides instead at position *geocentric_lat0*, *lon0* (in degrees), then supply the 3 Euler rotation angles *lon0*+90, 90 - *geocentric_lat0*, 90.

Note that if it is desired to rotate the grid such that grid vertex zero is located at the station location, it is easier to simply specify property *geotessRotateGridToStation* = true.

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.

dbInputUserName, dbOutputUserName

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

Database account usernames.

dbInputPassword, dbOutputPassword

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

Database input/output account passwords.

dbInputInstance, dbOutputInstance

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

Database instance for input/output.

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.

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.

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.

dbInputOriginTable

<string> [Default not allowed]

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

dbInputAssocTable

<string> [Default not allowed]

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

dbInputArrivalTable

<string> [Default not allowed]

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

dbInputSiteTable

<string> [Default not allowed]

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

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.

dbOutputAssocTable

<string> [Default = none]

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

dbOutputAutoTableCreation

<bool> [Default = false]

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

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.

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.

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.

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.

depths

<list of doubles> [no default]

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

depthRange

<2 doubles and 1 int> [no default]

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

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.

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*.

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

Output Parameters

outputFile

<string> [no Default]

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

separator

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

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

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, reflace '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.

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'

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.

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.

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).

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.

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.

lookup2dModel

<string> [Default = ak135] (ak135)

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

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.

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'.

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.

lookup2dUseEllipticityCorrections

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

lookup2dUseElevationCorrections

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

lookup2dSedimentaryVelocity

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

tauptoolkitModel

<string> [Default = ak135] (ak135)

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

tauptoolkitEllipticityCorrectionsDirectory

<string> [Default = none] ()

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

tauptoolkitUncertaintyType

<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.

tauptoolkitUncertaintyDirectory

<string> [Default = none] ()

Directory where distance dependent uncertainty values can be found for use with TaupToolkit predictions. Expecting to find subdirectories such as <tauptookitUncertaintyDirectory>/<attribute>/< <tauptookitUncertaintyModel>
For example: if uncertainty information is in file /index/SNL_tool_Root/seismicBaseData/tt/ak135 then specify
tauptoolkitUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
tauptoolkitUncertaintyModel = ak135

tauptoolkitUncertaintyModel

<string> [Default = none] ()

Subdirectory where distance dependent uncertainty values can be found for use with TaupToolkit predictions. Expecting to find subdirectories such as <tauptookitUncertaintyDirectory>/<attribute>/< <tauptookitUncertaintyModel>
For example: if uncertainty information is in file /index/SNL_tool_Root/seismicBaseData/tt/ak135 then specify

tauptoolkitUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
tauptoolkitUncertaintyModel = ak135

tauptoolkitSedimentaryVelocity

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

Sedimentary velocity value that TaupToolKitWrapper should use to compute source and receiver elevation corrections.

benderModel

<string> [Default = none] ()

Path to GeoTessModel that Bender should use to calculate predictions of seismic observables.

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.

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

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

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.

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.

slbmModel

<string> [Default = null]

The full path to the directory where the slbm model can be found.

slbm_max_distance

<double> [Default = 1e4]

The maximum source-receiver distance, in degrees, at which SLBM will return valid Pn/Sn predicted travel times. If a Pn or Sn travel time observation which is supposed to use SLBM for predicted travel times is more than this distance from the source, then the observation will be set to non-defining.

slbm_max_depth

<double> [Default = 1e4]

The maximum source depth, in km, for which SLBM will return valid Pn/Sn predicted travel times. If a Pn or Sn travel time prediction is requested from SLBM for a source depth greater than this depth, then the observation will be set to non-defining.

slbm_chmax

<double> [Default = 0.2]

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.

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.

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.

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.

LibCorr3D

<predictor>PathCorrectionsType

<string> [Default = none] (libcorr)

<predictor> is either lookup2d or tauptoolkit.
Set the value to 'libcorr' to apply libcorr3d corrections. If this parameter is omitted, then corrections will not be applied.

<predictor>LibCorrPathCorrectionsRoot

<string> [Default = none]

<predictor> is either lookup2d or tauptoolkit.
The name of the directory where all the libcorr3D correction surfaces reside. This directory should contain a separate file for each correction surface.

<predictor>LibCorrPathCorrectionsRelativeGridPath

<string> [Default = ".]

<predictor> is either lookup2d or tauptoolkit.
The relative path from the directory where the correction surface files reside to the directory where the grid files reside.

<predictor>LibCorrInterpolatorType

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

<predictor> is either lookup2d or tauptoolkit.
Type of horizontal interpolation to use.

<predictor>LibCorrPreloadModels

<boolean> [Default = false]

<predictor> is either lookup2d or tauptoolkit.
Whether all libcorr models should be loaded at startup or loaded on an 'as needed' basis.

<predictor>UsePathCorrectionsInDerivatives

<boolean> [Default = false]

<predictor> is either lookup2d or tauptoolkit.
Whether or not path corrections should be included in total values when computing derivatives of travel time with respect to source location.

Model Queries

benderModel

<string> [Default = none] ()

Path to geotessModel that PCalc should query for attribute values. This should point to a SALSA3D model.

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.

rayPathNodeSpacing

<double> [Default = -1]

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