SANDIA REPORT
SAND2019-9542
Printed August 2019

LocOO3D User's Manual

Andrea Conley[1], 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

LocOO3D is a software tool that computes geographical locations for seismic events at regional to global scales. This software has a rich set of features, including the ability to use custom 3D velocity models, correlated observations and master event locations. The LocOO3D software is especially useful for research related to seismic monitoring applications, since it allows users to easily explore a variety of location methods and scenarios and is compatible with the CSS3.0 software format used in monitoring applications. The LocOO3D 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
SVD	Singular value decomposition
DBIO	Database input/output
3D	Three dimensional

This page left blank

1. INTRODUCTION

LocOO3D (Object-Oriented 3-Dimensional Location Software) is a software package used for locating (and re-locating) single seismic events using a variety of seismic velocity models. Reflecting its origin in the seismic monitoring community, LocOO3D is compatible with the CSS3.0 data format commonly used by monitoring agencies. These data tables can either be stored in an Oracle database or input as flat files. Additionally, data can be input as custom formatted text as long as appropriate column labels are included in a header file (see Appendix A).

The LocOO3D software is distributed via the SALSA3D website at:

https://www.sandia.gov/salsa3d

LocOO3D is packaged with a copy of this manual, the associated datasets described in the LocOO3D Tutorial and Examples section below, and the jar files geotess_2.5.1.jar and libcorr3d_1.2.3.jar which can be used to interrogate input GeoTess models. The LocOO3D software is formatted as a Java jar file and requires a Java Runtime Environment ≥ 1.8 to run. Additionally, if the user wishes to use LocOO3D with a database, the Oracle ojdbc*.jar file (ver ≥ 7) is required. See Appendix A for more details.

LocOO3D has a rich set of features for event location, providing users the ability to explore a variety of scenarios affecting event location, including:

- The ability to locate events using a variety of velocity models, including AK135, which is directly built into the software. Additionally, users can construct a 3D model in GeoTESS format (see www.sandia.gov/geotess for details on use), use the SALSA3D GeoTESS models provided on the SALSA3D website, or use travel-time lookup surfaces for event location.

- In conjunction with the software pCalc (provided on the SALSA3D website), users can construct their own travel-time look up surfaces in GeoTESS format for use in event location.

- Master event location, wherein a seismic event is located relative to a fixed location of a known event.

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

- Compute event locations with specified correlations between closely spaced stations to avoid location bias.

- Sophisticated error estimations for locations, including standard error ellipse computation.

- Multiple events can be located in a single software run either in a sequential or parallel mode, making efficient location of a large number of events possible.

The terminology used throughout this document, as well as in the software self-documentation, reflects terminology commonly used in the seismic monitoring community. An **event** is defined as a single occurrence of radiated seismic energy and is given a unique ID, typically denoted "EVID". An event can have any number of **origins** which specify a location and time at which an event occurred. In a typical LocOO3D run, a new origin is computed for an event using an existing origin typically taken from a catalog of seismicity. An **arrival** is an observation of the arrival time of a seismic phase at a particular seismic station. Arrivals are associated with an origin through an **assoc** table which pairs arrival IDs ("ARID") with origin IDs ("ORID"). The arrivals form the dataset by which location algorithms compute a new or improved event location and store that location as a new origin.

A typical run of LocOO3D consists of the following steps:

1. Starting from an event which already has an origin (although LocOO3D will generate a default origin if one is not specified) and velocity model.

2. Get the arrivals associated with that origin from the assoc table. Note that the selection of which arrivals are used to locate the event, called defining arrivals, can be modified using run parameters. There must be at least one defining arrival for LocOO3D to run.

3. Compute a new location using the starting origin and the defining arrivals and create a new origin for this event.

4. Examine the quality of the new location by examining the change in travel time residuals and/or the error in the event location.

See the LocOO3D Tutorial and Examples section for more details on this process.

2. LOCOO3D THEORY

LocOO3D uses an iterative linear least squares inversion algorithm to locate seismic events. This technique was originally proposed by Geiger (1910) and is described in detail by Jordan and Sverdrup (1981) and Lay and Wallace (1995).

The seismic event location problem can be described as follows: We are presented with *N* observations of seismic arrival times, station-to-event azimuth and horizontal slowness, which we denote as a vector **d** of length *N*, and we wish to determine the location in time and space of the seismic event which produced the observations. The location is described by a vector **m**, of length *M* = 4, which contains the event latitude, longitude, depth and origin time. In order to find **m**, we need an operator **F**, which relates **m** and **d**:

$$\mathbf{F}(\mathbf{m}) \approx \mathbf{d}$$

F is a prediction operator which, given an Earth model, is capable of estimating the travel-times, azimuth and slowness observations that make up the vector **d**. This equation is a statement of the forward problem: given an Earth model, **F**, and an arbitrary seismic event location, $\mathbf{m_0}$, we can calculate a set of predictions, $\mathbf{F}(\mathbf{m_0})$, of our observations, **d**. The problem we wish to solve is the inverse problem: given an Earth model, **F**, find **m**, the location of the seismic event that produces a set of predictions **F**(**m**) which agree as well as possible with the observations **d**.

We must specify precisely what we mean by 'agree as well as possible'. To that end we define a vector **r** of weighted residuals

$$\mathbf{r} = \mathbf{w} \cdot [\mathbf{d} - \mathbf{F}(\mathbf{m_0})]$$

We seek the event location **m** that minimizes the sum of the squared weighted residuals, $||r||^2$, defined as

$$||\mathrm{r}||^2 = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{r}_\mathrm{i}^2$$

Since **F** is non-linear, we cannot solve the inverse problem directly. The standard approach is to start with an initial estimate of the event location, $\mathbf{m_0}$, and try to deduce an appropriate perturbation to that location, δ**m**, such that

$$\mathbf{m}_0 + \delta\mathbf{m} = \mathbf{m}$$

In other words, we find the change in event location δ**m** which will move the event location from the initial estimate of the event location $\mathbf{m_0}$ to the event location **m** that, when operated on by the Earth model, will produce predictions that agree as well as possible with the observations. From the above equations, we now have

$$\mathbf{F}(\mathbf{m_0} + \delta\mathbf{m}) \approx \mathbf{d}$$

This equation is nonlinear, but we make a linear approximation using a Taylor series, from which we can estimate $\delta\mathbf{m}$

$$\mathbf{F}(\mathbf{m_0} + \boldsymbol{\delta}\mathbf{m}) \approx \mathbf{F}(\mathbf{m_0}) + \left(\frac{\partial\mathbf{F}(\mathbf{m_0})}{\partial\mathbf{m_0}}\right)\boldsymbol{\delta}\mathbf{m} + \cdots$$

Using this equation, we have

$$\left(\frac{\partial \mathbf{F}(\mathbf{m_0})}{\partial \mathbf{m_0}}\right) \boldsymbol{\delta}\mathbf{m} \approx \mathbf{d} - \mathbf{F}(\mathbf{m_0})$$

We apply a weighting to this equation, so it is rewritten

$$\boldsymbol{A} \cdot \boldsymbol{\delta}\mathbf{m} \approx \mathbf{r}$$

Where **A** is a matrix of weighted partial derivatives

$$\boldsymbol{A} = \boldsymbol{w} \cdot \frac{\partial \mathbf{F}(\mathbf{m_0})}{\partial \mathbf{m_0}}$$

We now have an equation that describes how changes in a given event location will affect residuals between observed and predicted seismic quantities. What we seek is the particular value of δ**m** that leads to$\|\boldsymbol{r}\|_{min}^2$. To find that value of δ**m**, we must seek the least squares solution by minimizing

$$E^2 = \sum_{i=1}^{N} \left(r_i - \sum_{j=1}^{M} A_{ij} \delta m_j \right)^2$$

The solution of which yields the normal equation

$$\boldsymbol{A^T} \cdot \boldsymbol{A} \cdot \boldsymbol{\delta m} \approx \boldsymbol{A^T} \cdot \boldsymbol{r}$$

This equation can be solved for δ**m** in a variety of ways; LocOO3D uses a singular value decomposition approach (SVD; Menke, 1989; Lay and Wallace, 1995). Once we have solved for δ**m**, we then move the location of the event to a place where E^2 is smaller, and the procedure is iterated until convergence is reached.

2.1. Convergence Criteria

Iteration is terminated when either some maximum number of iterations have been exhausted or when $\|\boldsymbol{r}\|^2$ceases to change significantly. More specifically, convergence is declared when

$$\left| \frac{\|\boldsymbol{r}\|_i^2}{\|\boldsymbol{r}\|_{i-1}^2} - 1 \right| < tolerance$$

where subscripts *i* and *i*–1 indicate the current and previous iterations, respectively. A reasonable value for tolerance might be 1 × 10^{-3}. Convergence can also be declared when damping is being applied and small changes in location fail to reduce $\|\boldsymbol{r}\|^2$ relative to its value at the conclusion of the previous iteration.

2.2. Poorly Constrained Events

Up to this point we have assumed that the solution to our problem was well-constrained by the available data. What if this is not the case? The degree to which **A** is ill-conditioned is characterized by the ratio of the largest to the smallest singular values, a quantity that is referred to as the condition number.

One of the significant advantages of the SVD algorithm is that after computing the SVD of **A**, we can examine the singular values to assess potential difficulties and take steps to alleviate them before finally computing $\delta\mathbf{m}$ (Press et al., 2002).

If **A** is singular, then one or more of the singular values will be equal to zero and the condition number will be infinite. In cases where the condition number is extremely large, round off errors in the computer may come in to play, yielding a solution with wildly large components that send our event location off into regions that make no physical sense. Fortunately, this situation appears to be rare in overdetermined seismic event location problems. More common is for the condition number to be moderately large, in which case it is possible to solve for all of the location parameters, but it may not be particularly useful to do so. The uncertainties on the combination of parameters corresponding to the very small singular values may be extremely large (larger than the dimensions of the Earth, for example) rendering the results meaningless.

When we encounter a situation with a large condition number (greater than 10^6, say) our best option is to change one of the very small singular values to zero (Press et al., 2002). As will be described in the next section, zero singular values will be rendered ineffectual, causing the algorithm to simply not solve for the parameter corresponding manipulated singular value. Parameters with zero singular values will not deviate from their initial estimates.

2.3. Non-linear Effects

It is possible for our solution to diverge, meaning that at the conclusion of a given iteration we find that $||\boldsymbol{r}||^2$ has increased over its value at the beginning of the iteration. This indicates failure of our algorithm and can be attributed to violation of the linearity assumption.

When the linearity assumption is not valid, $||\boldsymbol{r}||^2_{\boldsymbol{min}}$ will reside at the lowest point in a non-linear trough-like feature of potentially complex topology and the position pointed to by $\delta\mathbf{m}$ may be characterized by a value of $||\boldsymbol{r}||^2$ which is higher than the value of $||\boldsymbol{r}||^2$at the current position. In this case, we would be better off just taking a small step in the direction of steepest descent.

Best of all would be if we could interpolate smoothly between taking the step specified by $\delta\mathbf{m}$ and taking a small step in the direction of steepest descent. A method to accomplish this was originally proposed by Levenberg (1944) and Marquardt (1963) and is described in Press et al. (2002). In this method, the normal equations are modified to

$$(\boldsymbol{A^T A} + \lambda \boldsymbol{I}) \cdot \boldsymbol{\delta m} \approx \boldsymbol{A^T} \cdot \boldsymbol{r}$$

where λ is a damping parameter and $\boldsymbol{I}$ is the identity matrix. The Levenberg-Marquardt transformation has two effects on $\delta\mathbf{m}$: $\delta\mathbf{m}$ is simultaneously reduced in magnitude and rotated in the direction of the direction of steepest descent. When $\lambda = 0$, we have our unmodified least squares solution. As $\lambda \rightarrow \infty$, the direction of $\delta\mathbf{m}$ will approach the direction of steepest descent and its length will approach zero. It remains only to select the appropriate value of λ such that we take as large a step as possible that still moves our solution down the gradient (i.e., such that $||\boldsymbol{r}||^2_{i+1} < ||\boldsymbol{r}||^2_{i}$).

Our algorithm proceeds in the following manner:

1) Begin by initializing the applied damping factor, λ, to a negligible initial value, λ_0, such as 10^{-8}, at the start of the first iteration.

2) Start each iteration *i* by calculating $\delta\mathbf{m}_i$, $\mathbf{m}_{i+1}$ and $\|\boldsymbol{r}\|_{i+1}^2$.

3) If $\|\boldsymbol{r}\|_{i+1}^2 \geq \|\boldsymbol{r}\|_i^2$:

    a. discard $\mathbf{m}_{i+1}$

    b. increase λ by some factor λ_x (10 for example)

    c. return to Step 2.

4) If $\|\boldsymbol{r}\|_{i+1}^2 < \|\boldsymbol{r}\|_i^2$:

    a. replace $\mathbf{m}_i$ with $\mathbf{m}_{i+1}$

    b. if $\lambda < \lambda_0$, divide it once by λ_x

    c. return to Step 2 to start the next iteration.

This algorithm seeks to keep λ as small as possible by initializing it to a negligible value and by reducing it by factor λ_x at the conclusion of each successful iteration. Keeping λ small keeps the step size as large as possible. The algorithm is willing to increase λ as much as necessary, however, to ensure that $\|\boldsymbol{r}\|^2$ is never allowed to increase from one iteration to the next.

It is possible that the event location $\mathbf{m}_i$ at the start of some iteration *i* is characterized by a value of $\|\boldsymbol{r}\|_{\boldsymbol{i}}^{\mathbf{2}}$ which is very close to $\|\boldsymbol{r}\|_{\boldsymbol{min}}^{\mathbf{2}}$ and a value of $\|\boldsymbol{r}\|_{\boldsymbol{i+1}}^{\mathbf{2}}$ that is significantly lower than $\|\boldsymbol{r}\|_{\boldsymbol{i}}^{\mathbf{2}}$ will never be found. This situation is handled by monitoring the length of $\delta\mathbf{m}$, an approximation of which is given by

$$\|\delta\mathbf{m}\| = \sqrt{(\Delta \cdot R)^2 + dz^2 + (dT \cdot v)^2}$$

where Δ is the great circle distance the epicenter moved, in radians, *R* is the radius of the earth in km , *dz* is the change in hypocenter depth, in km, *dT* is the change in origin time, in seconds and *v* is an estimate of the seismic velocity of the medium near the hypocenter (assumed to be a constant 8 km/second).

As λ increases, $\|\delta\mathbf{m}\|$will decrease. If $\|\delta\mathbf{m}\|$ decreases below some threshold value (0.01 km for example), then the algorithm is terminated. At this point, not only is the current iteration concluded but solution convergence can be declared. The reason this conclusion can be reached is that repeated applications of our algorithm have failed to cause $\delta\mathbf{m}$ to point down the $\|\boldsymbol{r}\|^2$ gradient. Further applications may ultimately achieve that end, but only after having reduced the magnitude of $\delta\mathbf{m}$ so severely that it results in only a negligible change in $\mathbf{m}$. We must therefore be at, or very near, the value of $\mathbf{m}$ where $\|r\|^2 = \|\boldsymbol{r}\|_{min}^2$.

3. LOCOO3D TUTORIAL AND EXAMPLES

Included with the LocOO3D distribution is a set of example input files that demonstrate how LocOO3D works. The required input files for a LocOO3D run are:

1. A properties (*.properties) file which contains the parameter settings used by LocOO3D for a particular run. The format of the properties file and a list of all possible properties that can be set are described in Appendix C.

2. A set of text files containing data used by LocOO3D to constrain event locations. These files include an origins file, containing information about the starting origin used in the location process; an arrivals file, which contains information about the time and type of arrival at a particular station; an association file, which associates the arrivals in the arrival file to the origins in the origin file; and finally, a sites file which contains information about the location of the stations at which the arrivals were recorded. Appendix A describes the format of the origin, association, site and arrival files.

CREATE DIRECTORY STRUCTURE:

To execute the examples provided, the user should set up their directory structure in the following way:

1. Create a working directory on your system (e.g. snl_location_software).
2. From the website (https://www.sandia.gov/salsa3d/Software.html) download the following files into the working directory and uncompress them:

    a. LocOO3D.tgz
    b. SALSA3D_Model.tgz

3. In the working directory, create a subdirectory called libcorr3d_models_tt_delta_ak135
4. From the website (https://www.sandia.gov/salsa3d/Travel%20Time%20Tables.html), download the following compressed files into the libcorr3d_models_tt_delta_ak135 subdirectory and uncompress them:

    a. Pmantle_geotess_tt_delta_ak135.zip

        i. Select Depth-dependent travel time table for P phase relative to AK135 in GeoTESS Format from website

    b. PcP_geotess_tt_delta_ak135.zip

        i. Select Depth-dependent travel time table for PcP phase relative to AK135 in GeoTESS Format from website

    c. PKPbc_geotess_tt_delta_ak135.zip

        i. Select Depth-dependent travel time table for PKPbc phase relative to AK135 in GeoTESS Format from website

   d. PKPdf_geotess_tt_delta_ak135.zip

      i. Select Depth-dependent travel time table for PKPdf phase relative to AK135 in GeoTESS Format from website

5. When complete, the user should have the following directory structure set up:

```
snl_location_software
 |
 |--------------------------LocOO3D
 |                              |
 |                              |----Examples
 |                              |        |
 |                              |        |------Data
 |                              |        |
 |                              |        |------Example01
 |                              |        |
 |                              |        |------Example02
 |                              |        |
 |                              |        |------Example03
 |                              |        |
 |                              |        |------Example04
 |                              |
 |                              |----jars
 |
 |---------------------SALSA3D_Model
 |                              |
 |                              |----SALSA3D.geotess
 |                              |
 |                              |----salsa3d_distance_dependent_uncertainty_tables
 |                                       |
 |                                       |------tt
 |                                                |
 |                                                |-----salsa3d_1d
 |
 |------------------libcorr3d_models_tt_delta_ak135
                                |
                                |-----PKPbc_geotess_tt_delta_ak135
                                |
                                |-----PKPdf_geotess_tt_delta_ak135
                                |
                                |-----PcP_geotess_tt_delta_ak135
                                |
                                |-----Pmantle_geotess_tt_delta_ak135
```

CREATE EXECUTABLES:

With the directories appropriately set up, the user should now perform the following steps:

1. cd into the LocOO3D/jars directory.
2. On your system find the Oracle ojdbc*.jar file. This jar file is not included with LocOO3D.
3. Edit the file LocOO3D/jars/configure.sh and change the value of the OJDBC variable to point to the path to your oracle ojdbc*.jar file. **Warning:** do not allow any spaces around the '=' sign.
4. Go back to the command line in the LocOO3D/jars directory and execute:

   ```
   $./configure.sh
   ```

   The configure.sh script will generate a 'set path' or 'export' command which should be copied to your .cshrc or .bash_profile file. Source your .cshrc file. This will create executable files for `locoo3d`, `supportmap`, `geotess` and `libcorr3d` programs. The `geotess` and `libcorr3d` executables allow the user to interrogate input GeoTess models. Run these executables without an input statement to print usage instructions to screen.

SET UP SUPPORT MAP:

Next, the user should set up a support map. For background information about support maps and why they are needed, see Appendix D. Here, we will just execute the necessary commands to generate the support map file.

1. OPTIONAL: If the user is going to access Oracle databases, update the parameters in supportMap.properties to have specific database information (see Example04, Appendix C.11 for details).
2. cd to the LocOO3D/Examples directory and run:

   ```
   $supportmap supportMap.properties
   ```

RUN EXAMPLES:

Now the user is ready to run the examples located in the “Examples” directory of the LocOO3D distribution. Within this directory are four directories containing each unique example (Example01, Example02, etc.) as well as the “Data” directory, which contains the common origin, association, site and arrival files used in all four examples.

The data consist of a single origin occurring in the Arctic Ocean near Greenland with six arrivals, each recorded by a unique International Monitoring System (IMS) station. These arrivals consist of mantle (P, Pn) and core (PcP, PKPdf, PKPbc) phases. Note that while we only make use of one origin and a small handful of sites and P-arrivals in our examples, LocOO3D is capable of locating several events using whatever phases are supported in the model being analyzed. Thus, the origin, association, and arrival files can contain information for multiple events.

Here we provide brief instructions to run each example. For more detail on each example, see Sections 3.1-3.4.

EXAMPLE01:

1. cd to the "Examples/Example01" directory and run `$locoo3d example01.properties`

EXAMPLE02:

1. cd to the "Examples/Example02" directory and run `$locoo3d example02.properties`

EXAMPLE03:

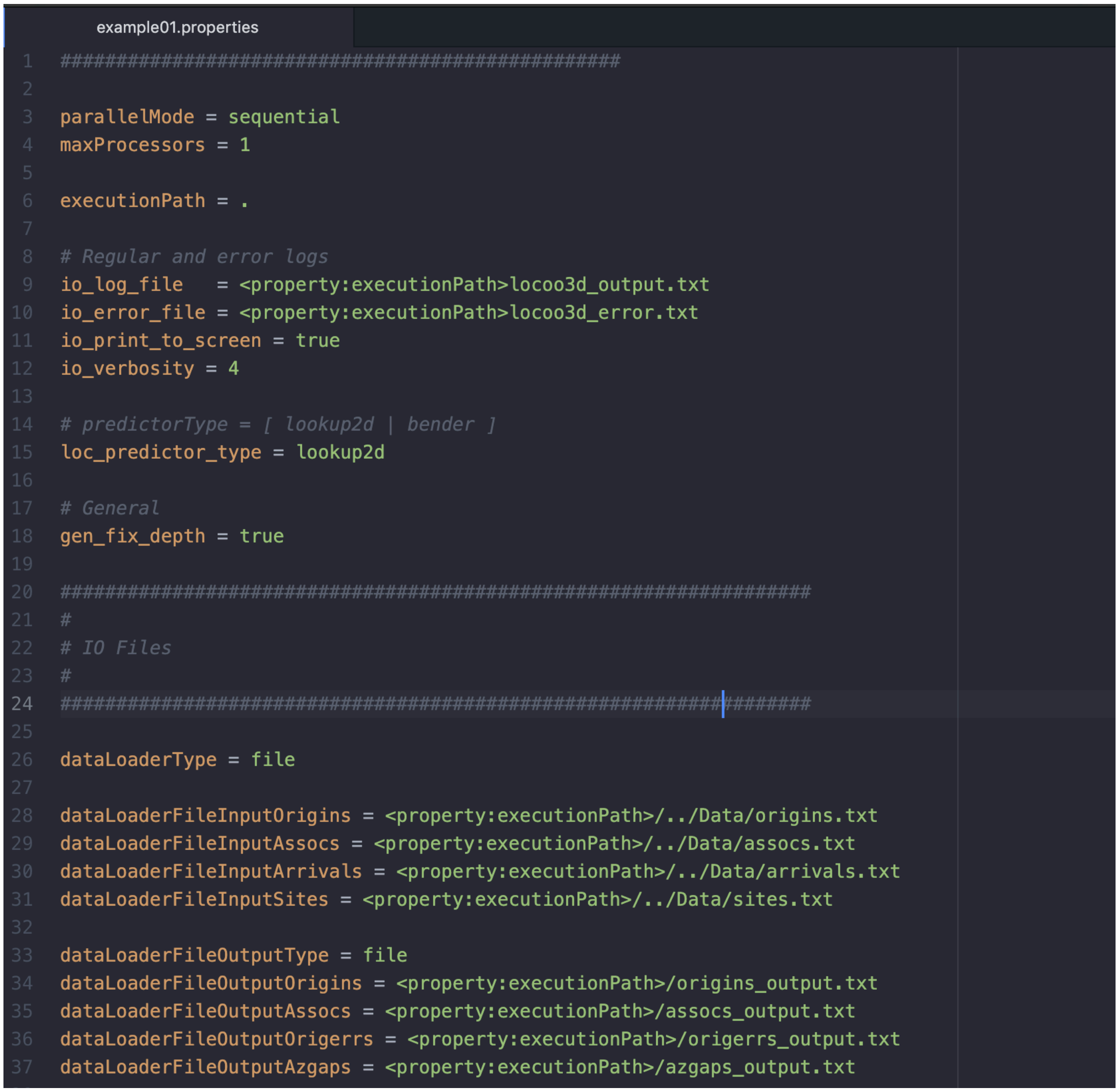
1. cd to the "Examples/Example03" directory and run `$locoo3d example03.properties`

EXAMPLE04:

1. cd to the "Examples/Example04" directory and run `$locoo3d example04.properties`
    - **a.** Example04 runs the exact same problem as Example01 but it gets data from a database. You will need to edit file example04.properties and specify correct database connection parameters. The example may not run properly because it accesses data in SNL databases.

3.1. Example 1 – Internal Lookup Tables

"Examples/Example01" demonstrates event relocation using the 2D travel-time lookup tables included with the LocOO3D software. These lookup travel-time tables were computed using the tau-p method and the AK135 model. The properties file for Example01 is labeled as "example01.properties" and is shown in Figure 1 below.



```
 1  ##################################################
 2
 3  parallelMode = sequential
 4  maxProcessors = 1
 5
 6  executionPath = .
 7
 8  # Regular and error logs
 9  io_log_file   = <property:executionPath>locoo3d_output.txt
10  io_error_file = <property:executionPath>locoo3d_error.txt
11  io_print_to_screen = true
12  io_verbosity = 4
13
14  # predictorType = [ lookup2d | bender ]
15  loc_predictor_type = lookup2d
16
17  # General
18  gen_fix_depth = true
19
20  ###################################################################
21  #
22  # IO Files
23  #
24  ###################################################################
25
26  dataLoaderType = file
27
28  dataLoaderFileInputOrigins = <property:executionPath>/../Data/origins.txt
29  dataLoaderFileInputAssocs = <property:executionPath>/../Data/assocs.txt
30  dataLoaderFileInputArrivals = <property:executionPath>/../Data/arrivals.txt
31  dataLoaderFileInputSites = <property:executionPath>/../Data/sites.txt
32
33  dataLoaderFileOutputType = file
34  dataLoaderFileOutputOrigins = <property:executionPath>/origins_output.txt
35  dataLoaderFileOutputAssocs = <property:executionPath>/assocs_output.txt
36  dataLoaderFileOutputOrigerrs = <property:executionPath>/origerrs_output.txt
37  dataLoaderFileOutputAzgaps = <property:executionPath>/azgaps_output.txt
```

Figure 1. ***The properties file for Example01 as included in the directory "Examples/Example01".***

This properties file represents the minimum set of parameters required to run LocOO3D. A brief description of each parameter in the example is described below. For more detailed descriptions, as well as a list of other potential parameters, see Appendix C.

LocOO3D has the ability to run sequentially or in parallel on suitably equipped machines (see Appendix B for details). The lines:

```
parallelMode = sequential
maxProcessors = 1
```

specify that we are going to run this example in serial mode on a single processor. Line 6:

```
executionPath = .
```

specifies the directory in which the example is to be run.

The syntax `<property:propertyName>` can be used to reference properties defined earlier in the input file. For example, we reference `executionPath` in order to specify the directory to write the output (`io_log_file`) and error (`io_error_file`) log files to:

```
io_log_file = <property:executionPath>locoo3d_output.txt
io_error_file = <property:executionPath>locoo3d_error.txt
```

The parameters in Lines 11 and 12:

```
io_print_to_screen = true
io_verbosity = 4
```

specify that the output will be printed to screen and define the I/O verbosity level, respectively. The verbosity can range from 0-4, with 0 providing no output and 4 providing all available output. Throughout our examples we will be using the maximum verbosity option. See Appendix C for more details on `io_verbosity`.

The next parameter:

```
loc_predictor_type = lookup2d
```

specifies the type of travel-time predictor to use during the location process. Several travel-time predictor methods/options are available in LocOO3D (see Appendix C). Here we will demonstrate the use of the 2D internal travel-time lookup tables included with LocOO3D by selecting `lookup2d`.

In LocOO3D, any of the four components of the event location (depth, latitude, longitude, origin time; see Section 2) can be made a free parameter to be solved for or can be fixed to either a user-defined value or the initial value in the origin data (see Appendix C). Here we will fix the event depth. We perform this action by specifying the following parameter on Line 18:

```
gen_fix_depth = true
```

By not defining a depth value using the `gen_init_depth` parameter (see Appendix C), LocOO3D will fix depth to the value reported in "Data/origins.txt" (in this case 0 km) by default.

We will now define the input files to be used by LocOO3D as well as the output tables to be written out. We first specify where the input data will be read in from using the `dataLoaderType` parameter on Line 26. Data can either be read in from a text file or from a database if the required Oracle jar file is available (see Appendix A.2). In this example, we will read in our data from the files in the common "Examples/Data" directory by setting `dataLoaderType = file` and specifying the `dataLoaderFile` parameters for each necessary table type (origins, assocs, arrivals, and sites).

```
dataLoaderType = file
dataLoaderFileOrigins = <property:executionPath>origins_input.txt
dataLoaderFileAssocs = <property:executionPath>assocs_input.txt
dataLoaderFileArrivals = <property:executionPath>arrivals_sub.txt
dataLoaderFileSites = <property:executionPath>sites_sub.txt
```

Finally, we define the output file type and the path and file name to write the output to. Similar to the input, output can be written either to text files or as database tables (if the Oracle jar is available). Once again, we will save to text files in this example by setting `dataLoaderFileOutputType = file`.

Four output table types can be written out by LocOO3D: an 1) origin, 2) association, 3) origin error, and 4) azimuthal gap table. These will be written to the path and file name specified by the `dataLoaderFileOutput` parameters specified for each desired table type.

```
dataLoaderFileOutputType = file
dataLoaderFileOutputOrigins = <property:executionPath>origins_output.txt
dataLoaderFileOutputAssocs = <property:executionPath>assocs_output.txt
dataLoaderFileOutputOrigerrs = <property:executionPath>origerrs_output.txt
dataLoaderFileOutputAzgaps = <property:executionPath>azgaps_output.txt
```

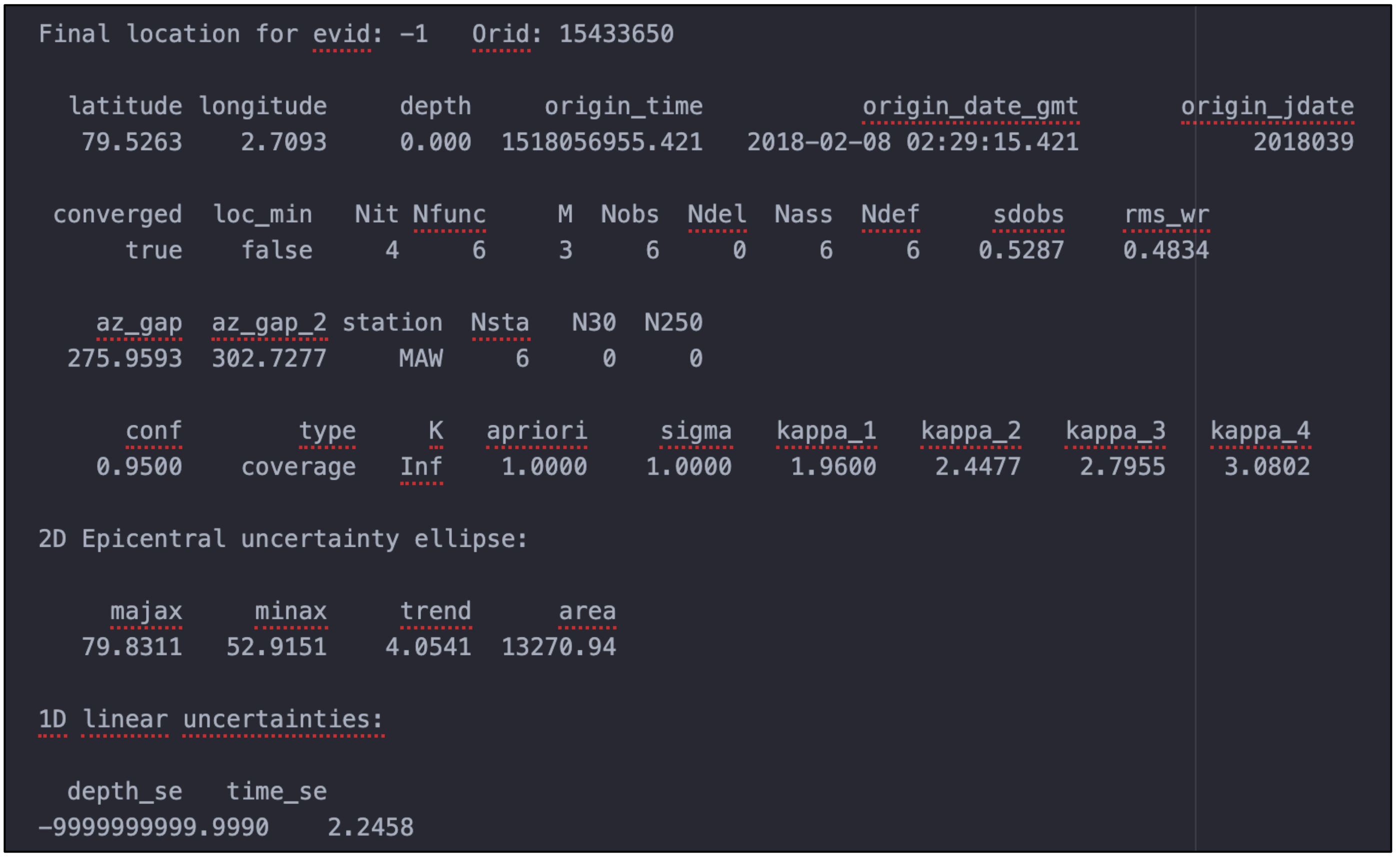
By default, the output tables will be written as tab-delimited text files. See the `dataLoaderFileOutputTokenDelimiter` parameter defined in Appendix C for other options.

With the properties file appropriately set, we will now run Example01. The user should have first run the `configure.sh` script provided with the LocOO3D distribution (see Section 3) to generate the `locoo3d` executable. If this script has not been run, please run `$./configure.sh` in the "LocOO3D/jars" directory now.

Tor run Example01, we cd to the "Examples/Example01/" directory in the working directory and run the following command:

```
$locoo3d example01.properties
```

This command will execute all of the parameters described in the above properties file. During the run, several lines of output will be printed out to the screen. Following the run, the user should see several output text files, the most important of which is the `locoo3d_output.txt` file. This log file contains a large amount of information, including every parameter that was used in the run. For the sake of brevity, we will only discuss the solution summary shown near the end of the log file (Figure 2).



```
Final location for evid: -1   Orid: 15433650

  latitude longitude     depth     origin_time           origin_date_gmt       origin_jdate
   79.5263    2.7093     0.000  1518056955.421   2018-02-08 02:29:15.421            2018039

 converged   loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true     false     4     6     3     6     0     6     6    0.5287    0.4834

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.9593  302.7277     MAW     6     0     0

      conf      type     K   apriori     sigma   kappa_1   kappa_2   kappa_3   kappa_4
    0.9500  coverage   Inf    1.0000    1.0000    1.9600    2.4477    2.7955    3.0802

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   79.8311   52.9151    4.0541  13270.94

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2458
```

Figure 2. ***Summary section of the LocOO3D output for Example01.***

The output for this location computation shows that the location algorithm converged to a solution in 4 iterations (see Nit variable). Since this was a fixed-depth location, the final solution will only report the parameters of a 2D epicentral uncertainty ellipse. 1D linear uncertainties are also printed out; note that the depth uncertainty (depth_se) will be invalid for a fixed-depth solution. If a free-depth solution was chosen by the user, the summary would show both a 2D epicentral uncertainty ellipse and a 3D hypocentral uncertainty ellipsoid, along with valid depth and time 1D linear uncertainties.

Finally, in order to verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example01 directory for comparison.

3.2. Example 2 – Ray Tracing Through a GeoTESS Model

The LocOO3D input files contained in "Examples/Example02/" are set up to use the ray tracer (called "Bender") included with LocOO3D along with the SALSA3D model available at:

http://www.sandia.gov/salsa3d

In preparation to run this example, download the model and unzip it in the "Examples" directory. The newly downloaded "SALSA3D_model" directory should include the SALSA3D.geotess model and “salsa3d_distance_dependent_uncertainty_tables” directory needed to run this example (refer back to Section 3).

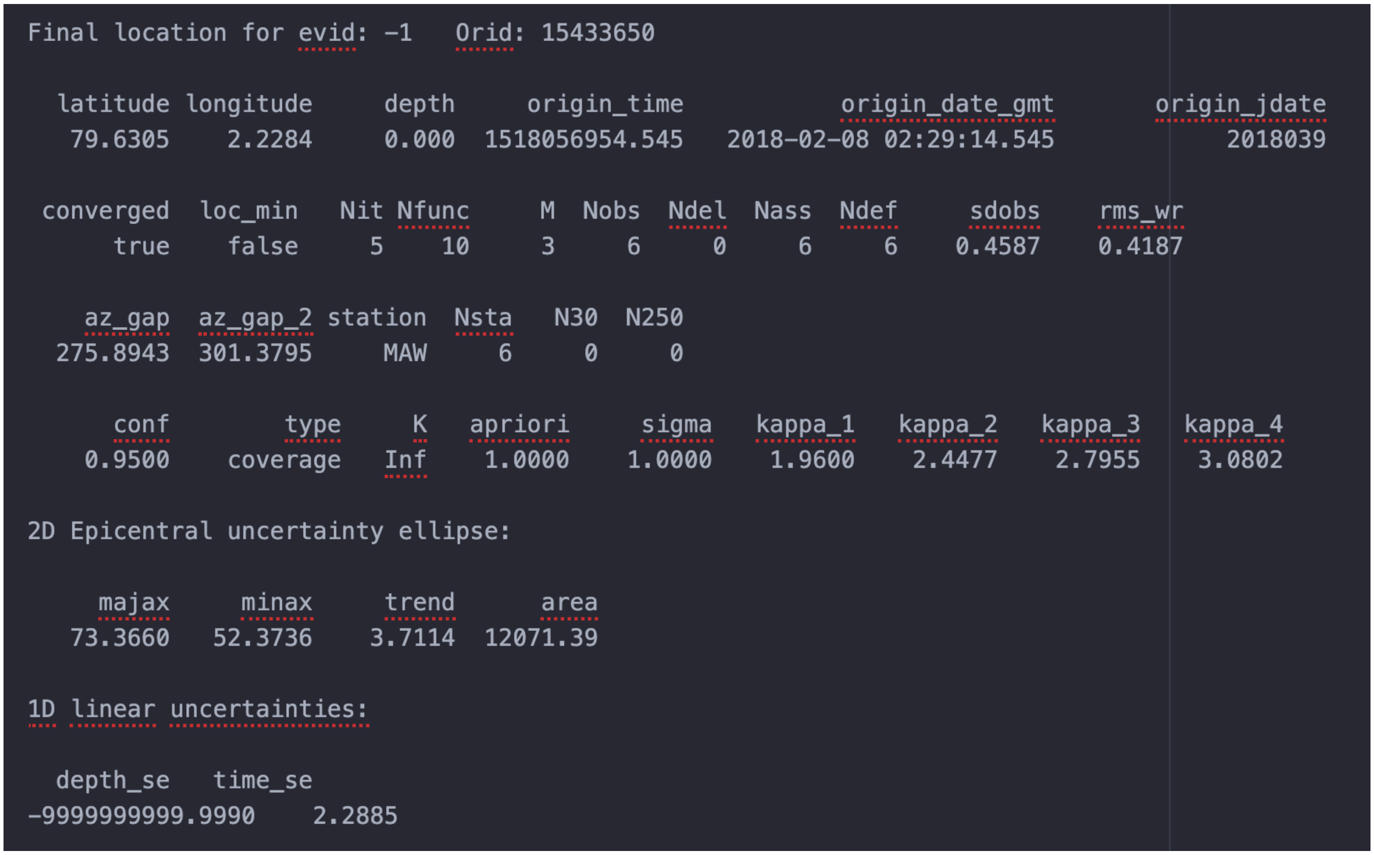
LocOO3D contains an internal set of travel time lookup tables that can be used to locate events in the absence of an external velocity model, as was shown in Section 3.1. However, the locations computed using the SALSA3D model can improve upon the locations computed with these internal tables. To specify that we wish to use an external velocity model to locate this event, the following parameters are added to the “example02.properties” file in addition to the parameters described in Section 3.1:

```
# predictorType = [ lookup2d | bender ]
# types that come later in list supersede previous
loc_predictor_type = lookup2d, bender(P,Pg,Pn,PKPdf)

# Bender
benderModel = <property:salsa3d_model_directory>/SALSA3D.geotess
benderAllowCMBDiffraction = true
benderAllowMOHODiffraction = true

benderUncertaintyType = DistanceDependent
benderUncertaintyDirectory = <property:salsa3d_model_directory>/
salsa3d_distance_dependent_uncertainty_tables
benderUncertaintyModel = salsa3d_1d
```

The addition of the Bender specification to the end of the `loc_predictor_type` property states that we are going to trace rays through the model specified by the `benderModel` property for the phases indicated in the parentheses. For all other phases we are going to use the internal 2D lookup tables described in Section 3.1. The next two properties, `benderAllowCMBDiffraction` and `benderAllowMOHODiffraction` tell Bender to mark phases that diffract off the CMB and Moho as valid. Setting these properties to “false” will disallow these phases’ use in the location computation.

We run this example with distance dependent uncertainty values for use with Bender predictions by specifying `benderUncertaintyType = DistanceDependent`. We then specify the location of the path to the tables containing the distance uncertainty values as well as the specific model to use by specifying the `benderUncertaintyDirectory` and the `benderUncertaintyModel`, respectively. Note that he uncertainty model `salsa3d_1d` is an example table populated with AK135 uncertainty values used to illustrate the expected uncertainty table structure.

This run results in the following summary file (Figure 3).



```
Final location for evid: -1   Orid: 15433650

  latitude longitude     depth     origin_time           origin_date_gmt       origin_jdate
   79.6305    2.2284     0.000  1518056954.545   2018-02-08 02:29:14.545            2018039

 converged   loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true     false     5    10     3     6     0     6     6    0.4587    0.4187

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.8943  301.3795     MAW     6     0     0

      conf      type     K   apriori     sigma   kappa_1   kappa_2   kappa_3   kappa_4
    0.9500  coverage   Inf    1.0000    1.0000    1.9600    2.4477    2.7955    3.0802

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   73.3660   52.3736    3.7114  12071.39

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2885
```

Figure 3. ***Summary section of the LocOO3D output for Example02.***

Note the improvement in the solution as compared to Example01.

Finally, in order to verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example02 directory for comparison.

3.3. Example 3 – Using GeoTESS Lookup Tables

It is possible within LocOO3D to apply a set of travel-time corrections to the internal AK135-based lookup tables, thus allowing travel-times computed with an arbitrary model to be used to locate events without having to trace rays through the model. Example03 demonstrates this capability using the travel-time lookup tables provided on the SALSA3D website (distributed as multiple zip files).

If the user has not already done so, download the zip files and place the unzipped travel-time lookup table directories in the directory "libcorr3d_models_tt_delta_ak135" under the working directory (refer back to Section 3). The directory should contain several lookup tables stored as *.geotess files, where the root file name is the abbreviation for a station in the IMS network as well as the phase it supports, e.g. `ZAL_PcP_TT.geotess`. Currently, only P mantle phases (P, Pn) and P core phases (PcP, PKPbc, PKPdf) are included.

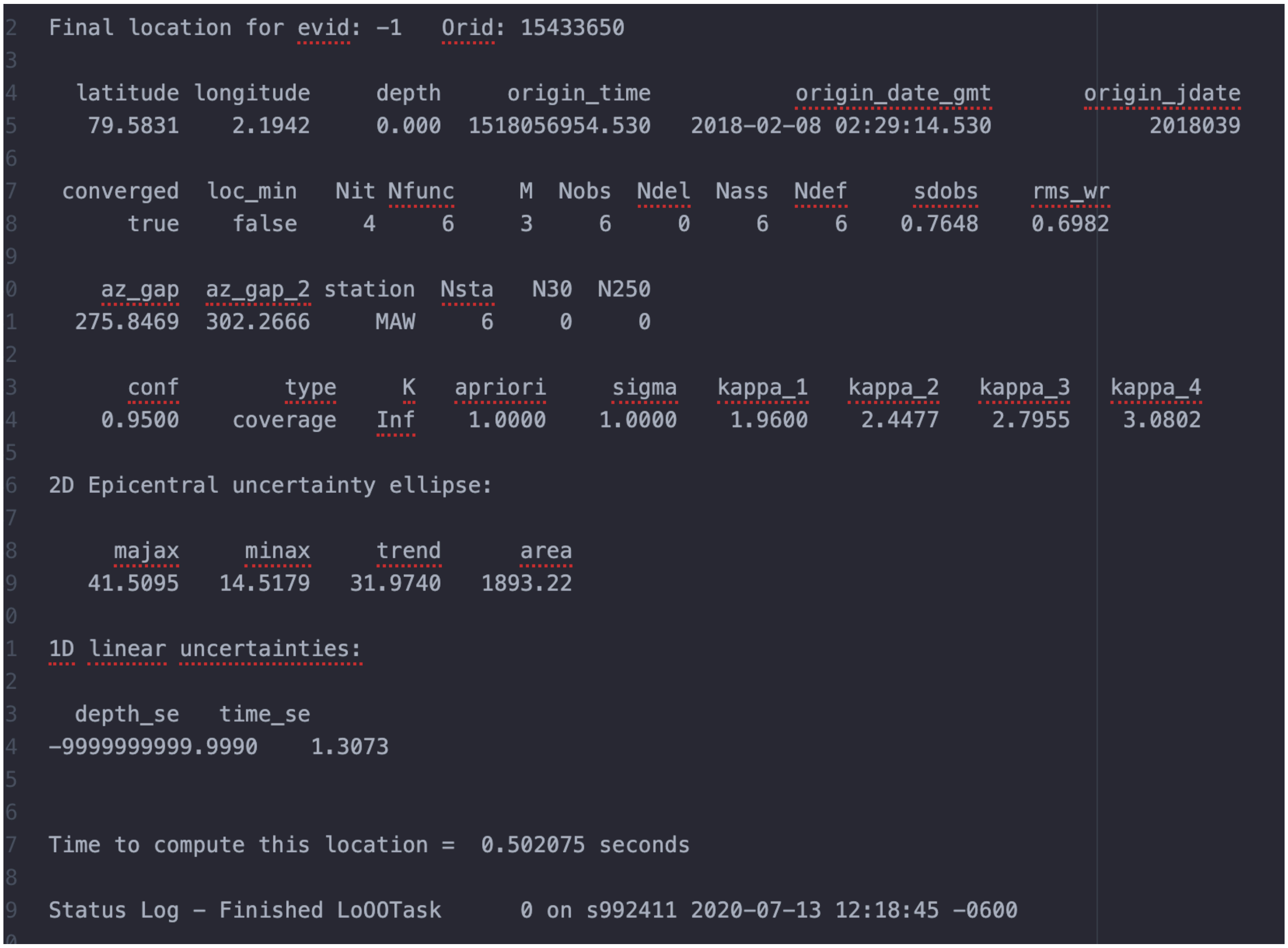
The user must also have generated and run the `supportmap` executable at this stage (refer back to Section 3). If this has not been done, run `$./config.sh,` in the "LocOO3D/jars" directory, then run `$supportmap supportMap.properties` in the "LocOO3D/Examples" directory now.

The Example03 properties file `example03.properties` is the same as the Example02 properties except the parameters specifically used for Bender (see Section 3.2) are now replaced with the following parameters:

```
lookup2dPathCorrectionsType=libcorr
lookup2dLibCorrPathCorrectionsRoot=<property:executionPath>/../../../
libcorr3d_models_tt_delta_ak135
lookup2dLibCorrPreloadModels=false
```

The parameter `lookup2dPathCorrectionsType=libcorr` tells LocOO3D to apply the corrections contained in the directory specified by the `lookup2dLibCorrPathCorrectionsRoot` parameter. The final parameter, `lookup2dLibCorrPreloadModels=false`, specifies that at the beginning of the run we are only going to load the models we need rather than all available models in order to preserve memory.

With the properties file appropriately defined, we perform the run resulting in the summary shown in Figure 4 below.



```
Final location for evid: -1   Orid: 15433650

  latitude longitude     depth     origin_time           origin_date_gmt       origin_jdate
   79.5831    2.1942     0.000  1518056954.530   2018-02-08 02:29:14.530            2018039

 converged   loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true     false     4     6     3     6     0     6     6    0.7648    0.6982

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.8469  302.2666     MAW     6     0     0

      conf      type     K   apriori     sigma   kappa_1   kappa_2   kappa_3   kappa_4
    0.9500  coverage   Inf    1.0000    1.0000    1.9600    2.4477    2.7955    3.0802

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   41.5095   14.5179   31.9740   1893.22

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    1.3073


Time to compute this location =  0.502075 seconds

Status Log - Finished LoOOTask      0 on s992411 2020-07-13 12:18:45 -0600
```

Figure 4. ***Summary section of the LocOO3D output for Example03.***

Note the much better constrained solution location as compared to Example01 and Example02.

Finally, in order to verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example03 directory for comparison.

3.4. Example 4 – Using Oracle I/O

As previously mentioned, if the user has the necessary Oracle ojdbc*.jar file available (see Appendix A.2 for details), LocOO3D can use Oracle databases for I/O rather than text files. Directory "Examples/Example04" will contain the exact same properties file as Example01, but with the original parameters reading in text files replaced by the following database parameters. Note that LocOO3D is designed to interact with the CSS3.0 schema or similar format (Appendix A.1).

```
# if dataLoaderType = oracle then specify database information
dataLoaderType = oracle
dbInputInstance = jdbc:oracle:thin:@domain:port:database
dbInputUserName = username
dbInputPassword = password

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

dbInputWhereClause = where orid = 15433650

dbInputAssocClause = assoc.arid in (12798118, 129979918, 129796973, 129796914,
129797143, 129973843)


dbOutputInstance = jdbc:oracle:thin:@domain:port:database
dbOutputUserName = username
dbOutputPassword = password


dbOutputTablePrefix = deleteme_
dbOutputTableTypes = origin, assoc, origerr, azgap


# dbOutputOriginTable = deleteme_origin

# dbOutputAssocTable = deleteme_assoc

# dbOutputOrigerrTable = deleteme_orrigerr

# dbOutputAzgapTable = deleteme_azgap

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

The output log file generated after running this example, `locoo3d_output.txt`, should be exactly the same as the log file generated for Example01 with the exception that the log file will now contain messages indicating that I/O is coming from a database, e.g. "Schema: dbInput". The requested output tables should also be written to the database location specified by the `dbOutputInstance` parameter with the prefix “deleteme_” in front of the specific table type, e.g. “deleteme_origin”.

4. SUMMARY

LocOO3D is used for computing locations of single seismic events and is compatible with a variety of seismic velocity models. The rich set of features available in LocOO3D allows users to explore a variety of scenarios:

- The ability to locate events using a variety of velocity models, including AK135, which is directly built into the software. Additionally, users can construct a 3D model in GeoTESS format (see www.sandia.gov/geotess), use the SALSA3D GeoTESS models (www.sandia.gov/salsa3d) or travel-time lookup surfaces for event location.
- In conjunction with the software pCalc (www.sandia.gov/salsa3d), users can construct their own travel-time look up surfaces in GeoTESS format for use in event location.
- Master event location, wherein a seismic event is located relative to a fixed location of a known event.
- The ability to directly interact with CSS3.0 format data tables stored in an Oracle database, including the insertion of newly computed origins into existing tables.
- Compute event locations with specified correlations between closely-spaced stations to avoid location bias.
- Sophisticated error estimations for locations, including standard error ellipse computation.
- Multiple events can be located in a single software run, either in a sequential or parallel mode, making efficient location of a large number of events possible.

LocOO3D software is distributed as a Java jar file and requires the Java Runtime Environment ≥ 1.8 to run. Additionally, if the user wishes to use LocOO3D with a database, the Oracle Ojdbc*.jar file (version ≥ 7) is required. LocOO3D allows users to explore a range of hypotheses affecting event location, especially for monitoring tasks. When used in combination with the related software packages pCalc and GeoTess, it forms an essential piece of a software suite that is capable of being used to construct, interrogate and test models of Earth's seismic wave velocity. The examples in this manual are intended to be useful for getting users of LocOO3D up and running with their own datasets.

The LocOO3D software is distributed via the SALSA3D website at: www.sandia.gov/salsa3d/Software.html

This page left blank

REFERENCES

Geiger, L. (1910), Herdbestimmung bei erdbeden ans den ankunftzeiten, K. Gessel. Wiss. Goett. v. 4, pp. 331-349.

Jordan, T. H. and K. A. Sverdrup (1981), Teleseismic Location Techniques and their Application to Earthquake Clusters in the South-Central Pacific, Bull. Seis. Soc. Am., v. 71, pp. 1105-1130.

Lay, T., and T. C. Wallace (1995), *Modern Global Seismology*, Academic Press.

Levenberg, K. (1944), A method for the solution of certain non-linear problems in least squares, Quart. Appl. Math., v. 2, pp. 164-168.

Marquardt, D. W. (1963), Journal of the Society for Industrial and Applied Mathematics, v. 11, pp. 431-441.

Menke, W. (1989), *Geophysical Data Analysis: Discrete Inverse Theory*, Academic Press.

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery (2002), *Numerical Recipes in C++, The Art of Scientific Computing*, 2nd Edition, Cambridge University Press.

APPENDIX A. INPUT/OUTPUT SETTINGS FOR LOCOO3D

LocOO3D can perform input and output operations against ascii flat files or oracle database tables. The option is specified using property `dataLoaderType`, which must be set to either "file", or "oracle".

A.1. Flat File I/O

The user specifies the names of files from which input data will be loaded and to which output results will be written. Users can limit which origins in the input origins table are processed with `dataLoaderFileOrids`. Flat files are read/written in CSS3.0 or similar format. See Section C.9 in Appendix C for additional information.

The full specification of the CSS3.0 input file formats can be found at

ftp://ftp.pmel.noaa.gov/newport/lau/tphase/data/css_wfdisc.pdf

LocOO3D is compatible with this format when entries are **tab-delimited** to allow quotes and commas in station names. The example input files included with the LocOO3D distribution are specified in this format.

However, since LocOO3D does not use all of the fields specified in the CSS3.0 format, the LocOO3D file reader is compatible with a simplified format. For each of the input files, the order of the required fields is specified in a single header line located at the top of the file, whose first character is a hash "#" and whose entries are tab-delimited. The required fields can be specified in any order. This header line is followed by the data in a series of tab-delimited lines with an order matching that of the header. The input files included with Example01 represent a minimal set of parameters specified in this format.

The fields required for each input file are:

Origin input file:

1. LAT - event latitude
2. LON - event longitude
3. DEPTH - event depth
4. TIME – origin time (Epoch time)
5. ORID - origin identifier; links to the assoc, origerr , origaux , netmag, and stamag tables
6. EVID - event identifier; links to the origaux, netmag, and stamag tables

Assoc input file:

1. ARID - arrival identifier. links to the arrival and stamag tables
2. ORID - origin identifier. links to the origin, origerr, origaux, netmag, and stamag tables

3. PHASE - final phase name used in event location
4. TIMEDEF – time used (d) or not (n) in event location
5. AZDEF – azimuth used (d) or not (n) in event location
6. SLODEF - slowness used (d) or not (n) in event location

Site input file:

1. STA - station code. links to the arrival, assoc, and stamag tables.
2. LAT - station latitude
3. LON - station longitude
4. ELEV - station elevation in km
5. ONDATE - station turn on date in Julian date
6. OFFDATE - station turn off date in Julian date

Arrival input file:

1. STA - stations code; links to the site table
2. TIME - arrival time (epoch time in second)
3. ARID - arrival identifier; links to the assoc and stamag tables
4. JDATE - Julian date
5. DELTIM – Pick uncertainty in seconds
6. AZIMUTH – Station to event azimuth (geographic, only used if azimuth is defining)
7. DELAZ – Azimuth uncertainty (only required if azimuth observation is defining)
8. SLO - Slowness observed at station (s/deg, only used if slowness is defining)
9. DELSLO – Slowness uncertainty (only required if slowness observation is defining)

A.2. Oracle Database I/O

LocOO3D can interact with an Oracle Database containing database tables in CSS3.0 format. For this to work, users must specify the database instance, username, password and driver. There must be separate definitions for input and output. For input, LocOO3D can read origin, assoc, arrival and site database tables. LocOO3D can output results to origin, assoc, arrival, site, origerr and azgap tables.

There are two basic methods to specify input and output database table names. In the first method, the user specifies

dbOutputTableTypes = origin, assoc, origerr, azgap
*dbOutputTablePrefix = prefix_*
dbOutputTableSuffix = _suffix

which will result in output to tables *prefix_origin_suffix, prefix_assoc_suffix*, etc.
The second method involves explicitly specifying each table, eg.:

dbInputOriginTable = my_origin
dbInputAssocTable = my_assoc
dbInputOrigerrTable = my_origerr
dbInputAzgapTable = my_azgap

The two methods can be combined, in which case the second method will override table names generated by the first method. There are similar properties for input tables.

For the output tables, the user can specify that the output tables are to be created if they do not already exist, that the output tables are to be truncated if they do exist, and whether the user should be prompted before truncating tables.

The user can specify SQL where clauses (*dbInputWhereClause* and *dbInputAssocClause*) that limit the data that are loaded from the input tables.

See Section C.11 in Appendix C for additional details.

Note that the Oracle jar file that supports interaction with Oracle databases (ojdbc*.jar) is not supplied in the LocOO3D distribution. This jar file must be included on the Java classpath in order for database interactions to be successful. If the user has this jar file, update the OJDBC variable in the configuration scripts (`configure_locoo3d.sh`, `configure_supportmap.sh`) to make use of it (see Section 3).

This page left blank

APPENDIX B. USING LOCOO3D IN PARALLEL

LocOO3D is capable of running in parallel mode on multi-threaded machines. There are two primary parallel modes of operation, selected using property "*parallelMode*". When *parallelMode* is *sequential*, origins are located sequentially but the predictions calculated during location calculations are computed concurrently. When *parallelMode* is *concurrent*, the locations are computed in parallel and the predictions are computed sequentially. The default is *sequential. Sequential* mode is advantageous when computing a small number of events that each have a large number of arrivals. *Concurrent* mode is likely faster when computing a large number of events with a relatively small number of arrivals.

In both *parallelModes*, property *maxProcessors* can be used to specify the number of processors LocOO3D can use. The default is all available processors.

See the description of property *dbInputWhereClause* in Appendix C for additional details.

This page left blank

APPENDIX C. PARAMETER DESCRIPTIONS FOR LOCOO3D

C.1. Setting Parameters:

The parameters required by LocOO3D 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 the parameter file.

Notes:

- LocOO3D parameters are case sensitive.
- All parameters in the parameter file must contain an '=' character, separating the parameter name from the parameter value (e.g. io_print_to_screen = true). 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\abc
  io_log_file = <property:testDirectory>\testdir
  ```

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

C.2. Predictors

C.2.1. *loc_predictor_type*

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

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).

C.2.2. *seismicBaseData*

<string> [Default = none] ()

Path to the seismicBaseData directory. If this parameter is not specified, a copy of the seismicBaseData directory that is included in the locoo3d jar file will be used.

C.2.3. *lookup2dModel*

<string> [Default = ak135] (ak135)

Name of the 1D model that Lookup2D should use to calculate predictions of seismic observables. Options include ak135 and iasp91, with ak135 acting as the default.

C.2.4. *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'.

If *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory* are both specified, then they will dictate the locations of the table directory and ellipticity corrections directories, as advertised. If either of them is not specified, then the locations of both directories will be deduced from property *seismicBaseData*.

C.2.5. *lookup2dEllipticityCorrectionsDirectory*

<string> [Default = none] ()

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

If *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory* are both specified, then they will dictate the locations of the table directory and ellipticity corrections directories, as advertised. If either of them is not specified, then the locations of both directories will be deduced from property *seismicBaseData*.

C.2.6. *lookup2dUseEllipticityCorrections*

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

C.2.7. *lookup2dUseElevationCorrections*

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

C.2.8. *lookup2dSedimentaryVelocity*

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

C.2.9. *lookup2dTTModelUncertaintyScale*

<2 doubles: scale and offset> [Default = null] ()

Travel time model uncertainty scale and offset. If one value is specified, it will be used to scale the travel time model uncertainty. If two values are specified, the second will be added to the travel time model uncertainty. In other words,
ttModelUncertainty = ttModelUncertainty * scale + offset.

C.2.10. *lookup2dUncertaintyType*

<string> [Default = “hierarchical”] (hierarchical | DistanceDependent)

The type of model uncertainty to apply. If hierarchical, the model uncertainty retrieved from the LibCorr3D model will be used if available; otherwise, the distance dependent model uncertainty from the underlying 1D model (ak135) will be used. If DistanceDependent is specified, then model uncertainty from the underlying 1D model will be used and the model uncertainty from the LibCorr3D model will be ignored.

C.2.11. ***benderModel***

<string> [Default = none] ()

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

C.2.12. ***benderUncertaintyType***

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

Type of travel time uncertainty desired. If UncertaintyNAValue is specified (default), then all requests for travel time uncertainty return the NA_VALUE (-999999.). If DistanceDependent is specified, then distance dependent uncertainty is returned.

C.2.13. ***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

specify:

benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135

C.2.14. ***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 the uncertainty information is in file:
/index/SNL_tool_Root/seismicBaseData/tt/ak135

specify:

benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135

C.2.15. *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 the GeoTessModel specified with parameter *benderModel*.

C.2.16. *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.

C.2.17. *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.

C.2.18. *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.

C.3. Master Event Relative Relocation

If parameter *masterEventWhereClause* is specified, then LocOO3D will apply master event relocation corrections to all input origins that it relocates. A single masterEvent origin along with associated assocs will be loaded into memory. For each masterEvent assoc, residuals of time, azimuth and slowness will be computed for defining observations using the Predictor defined in the properties file with property *loc_predictor_type*. These residuals will become masterEventCorrections which will be added to predictions of observations that have the same station-phase-type as the masterEventCorrection. Note that corrections to travel time, azimuth and slowness will be applied, so long as the corresponding master event assoc is time-defining, azimuth-defining and/or slowness-defining.

C.3.1. *masterEventWhereClause*

<string> [Default = null] (valid sql; eg. ‘orid = 100’)

If this parameter is present, then master event relocation option will be implemented. Must provide a valid sql where clause that will be executed against the *masterEventSchema* database schema. The where clause must return one and only one origin row. It the where clause returns anything other than one origin row, an exception is thrown.

C.3.2. ***masterEventAssocClause***

<string> [Default = null] (valid sql; eg. 'phase = 'P'')

Optional parameter to limit the assoc rows returned with the master event origin row.

C.3.3. ***masterEventUseOnlyStationsWithCorrections***

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

If true, then the only assocs that will be used to locate input origins are those that have valid master event corrections.

C.3.4. ***masterEventSchema***

<string> [Default = dbInput] (schemaName, eg. 'dbMaster')

If this parameter is specified, then all the properties of a valid Schema must be supplied (see description of *dbInput* properties in the *DBIO Utility* section). For example, if property *masterEventSchema = dbMaster*, then properties *dbMasterUserName, dbMasterPassword, dbMasterOriginTable*, etc, will be recognized and all information about the *masterEvent* origin will be retrieved from database tables as specified.

If this parameter is not specified, then it is assumed that the information about the masterEvent origin, assoc, arrival and site will come from the same database tables as the input origins which are to be relocated (*dbInput* schema).

C.4. LibCorr3D

C.4.1. ***lookup2dPathCorrectionsType***

<string> [Default = none] (libcorr)

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

C.4.2. ***lookup2dLibCorrPathCorrectionsRoot***

<string> [Default = none]

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

C.4.3. ***lookup2dLibCorrPathCorrectionsRelativeGridPath***

<string> [Default = "."]

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

C.4.4. *lookup2dLibCorrInterpolatorType*

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

Type of horizontal interpolation to use. The default linear interpolator is recommended.

C.4.5. *lookup2dLibCorrPreloadModels*

<boolean> [Default = false]

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

C.4.6. *lookup2dUsePathCorrectionsInDerivatives*

<boolean> [Default = false]

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

C.5. General

C.5.1. *lsq_max_iterations*

<int> [Default = 100] (0 <= X <= 100000)

Maximum allowable number of iterations. If this number is set to 0, LocOO3D simply computes the residuals and location uncertainty information at the initial location and outputs the results.

C.5.2. *gen_initial_location_method*

<string> [Default = data_file] (data_file | properties_file | internal)

Specifies the method for setting the initial event location.

Options:

data_file = If flat_file I/O is used, then take the initial location from the observation file summary line. For database I/O, use the location given in the origin table

properties_file = Use the values given in the *.properties file. See also gen_lat_init, gen_lon_init, gen_depth_init and gen_origin_time_init.

internal = For events with no defining azimuth observations, the initial location is set to the location of the station that observed the first arrival. If azimuth observations are available, then the initial location is based on the intersections of great circles computed from the azimuth observations. See Ballard [2002] for complete details.

C.5.3. *gen_lat_init*

<double> [Default = 0.0000] (-90.00 <= X <= 90.000)

Initial latitude value for the algorithm (degrees). Only applies if *gen_initial_location_method* = *properties_file*.

For initial location to be specified in properties file, all four parameters *gen_lat_init, gen_lon_init, gen_depth_init* and *gen_origin_time_init* must be specified in the properties file.

C.5.4. ***gen_lon_init***

<double> [Default = 0.0000] (-180.0 <= X <= 360.00)

Initial longitude value for the algorithm (degrees). Only applies if *gen_initial_location_method* = *properties_file*.

For initial location to be specified in properties file, all four parameters *gen_lat_init, gen_lon_init, gen_depth_init* and *gen_origin_time_init* must be specified in the properties file.

C.5.5. ***gen_depth_init***

<double, or "topography"> [Default = Globals.NA_Value] (-10000 <= X <= 10000)

Initial depth estimate for the algorithm (km). If this parameter is specified, then the initial depth is set to the value of this parameter. If the value of this parameter is a string that starts with "topo" (case insensitive) then initial depth will be set to the depth of the topographic/bathymetric surface interpolated from the topography model at the latitude, longitude position of the initial location. If *gen_depth_init* is set to 'topography' then parameter topo_model can optionally be used to specify the topography model to use.

C.5.6. ***gen_origin_time_init***

<string> [Default = Globals.NA_Value]

Initial origin time estimate for the algorithm expressed as epoch time (seconds since 1970). Only applies if *gen_initial_location_method* = *properties_file*.

For initial location to be specified in properties file, all four parameters *gen_lat_init, gen_lon_init, gen_depth_init* and *gen_origin_time_init* must be specified in the properties file.

C.5.7. ***topo_model***

<string> [Default = jarfile:etopo1_00500.geotess]

Path to a GeoTessModel that contains topographic/bathymetric information. By default, model etopo1_00500.geotess is loaded from the locoo3d jar file but parameter topo_model can be used to specify a different model located on the user's file system, if desired.

C.5.8. ***gen_fix_lat_lon***

<bool> [Default = false]

Hold (true) / don't hold (false) the latitude and longitude fixed during the solution algorithm. If held fixed, then the initial values specified by *gen_lat_init* and *gen_lon_init* are used. See also *gen_initial_location_method*, *gen_lat_init*, and *gen_lon_init*.

C.5.9. ***gen_fix_depth***

<bool> [Default = false]

Hold (true) / don't hold (false) the depth fixed during the solution algorithm. If held fixed, then the initial value specified by *gen_depth_init* is used. See also *gen_initial_location_method*, *gen_depth_init*.

C.5.10. ***gen_fix_origin_time***

<bool> [Default = false]

Hold (true) / don't hold (false) the origin time fixed during the solution algorithm. If held fixed, then the initial value specified by *gen_origin_time_init* is used. See also *gen_initial_location_method*, *gen_origin_time_init*.

C.5.11. ***gen_allow_big_residuals***

<bool> [Default = true]

Allow (true) / don't allow (false) observations that result in 'big' residual values.

Here is how this works: The event is first located using all defining observations. If any observations have weighted residuals greater than the value specified with *gen_big_residual_threshold* (default value is 3), then some subset of those observations are set to non-defining and the event is relocated.

This process continues until either there are no observations whose weighted residuals exceed the threshold or N = M where N is the number of defining observations and M is the number of degrees of freedom in the problem (4 for free depth solutions, 3 for fixed depth, etc). If there are observations with large residuals, then the subset of them that is set to non-defining is determined as follows. The minimum number that will be set to non-defining is 1. The maximum number that can be set to non-defining is (N-M) * *gen_big_residual_max_fraction*. *gen_big_residual_max_fraction* defaults to 0.1. See *gen_big_residual_threshold* and *gen_big_residual_max_fraction*.

C.5.12. ***gen_big_residual_threshold***

<double> [Default = 3.0000] (0.0000 < X < 100000)

Threshold weighted residual value above which observations are flagged as 'big'. See *gen_allow_big_residuals*.

C.5.13. ***gen_big_residual_max_fraction***

<double> [Default = 0.10] (0.0 <= X <= 1.0)

A constraint on the maximum number of observations that can be set to non-defining when *gen_allow_big_residuals* is false and there are observations with large residuals.

C.5.14. ***gen_max_depth***

<double> [Default = 700.00] (-1e6 <= X <= 1e6)

Maximum depth constraint (km).

C.5.15. ***gen_min_depth***

<double, or “topography”> [Default = -999999.] (-10000 <= X <= 10000)

Minimum depth constraint (km). If the value of this parameter is a string that starts with “topo” (case insensitive) then the depth of the event location is constrained to be no less than the depth of the local surface topography/bathymetry.

C.5.16. ***gen_defining_phases***

<string> [Default = all]

The subset of defining phases to use for the location algorithm. This overrides the assoc table values. Setting the value to 'all' is the default behavior and causes all previously indicated defining phases to be used. A comma-delimited list overrides this behavior and down-selects from the set of defining phases. All observations with phases that are not included in the list are set to non-defining.

C.5.17. ***gen_defining_stations***

<string> [Default = all]

The subset of defining stations to use for the location algorithm.

This overrides the assoc table values. Setting the value to 'all' is the default behavior and causes all previously indicated defining stations to be used. A comma-delimited list of station names overrides this behavior such that all observations from stations that not included in the list are set to non-defining.

C.5.18. ***gen_defining_attributes***

<string> [Default = all] (t, a, s)

The subset of defining attributes (travel time, azimuth, slowness) to use for the location algorithm.

This overrides the data file or assoc table values. Setting the value to 'all' is the default behavior and causes all previously indicated defining attributes to be used. A comma-delimited list overrides this behavior such that all observations with attributes that not included in the list are set to non-defining.

Any word starting with the letters t or T is interpreted to be travel time, a or A is interpreted to be azimuth, and s or S is interpreted to be slowness.

C.5.19. ***gen_defining_observations_filter***

<string> [Default = none]

A set of filters that can be used to toggle individual observations from defining to non-defining and vice versa. Each filter is of the form: ±ORID/STATION/PHASE/ATTRIBUTE.

The set of filters consists of a number of individual filters, separated by commas. Each observation starts out as either defining or non-defining (after application of parameters *gen_defining_stations*, *gen_defining_phases* and *gen_defining_attributes*). Then the observation is subjected to each filter, in order. If the observation matches the filter, then the observation is made defining if the first character of the filter is '+' or non-defining if the first character of the filter is '-'. Each component of a filter can be the '*' character. Every observation matches the component of a filter that is the '*' character.

For example, the filter +100/ILAR/P/TT will guarantee that for the origin with orid 100, the P travel time observation from station ILAR will be defining.

The filter -*/ILAR/*/*, +*/ILAR/P/t will first make all observations from station ILAR non-defining, then turn back on just the P travel time observations from station ILAR.

C.5.20. ***gen_error_ellipse_type***

<string> [Default = coverage] (coverage|confidence|mixed)

Type of error ellipse desired:

coverage	Dimensions of error ellipse depend only on a priori information (K=-1, interpreted as infinity).
confidence	Dimensions of error ellipse depend only on a posteriori information (K=0).
mixed	Dimensions of error ellipse depend on both a priori and a posteriori information. The Jordan-Sverdrup K parameter is set to the value of this parameter.

C.5.21. ***gen_jordan_sverdrup_K***

<int> [Default = -1] (-1 <= X <= 999999)

Jordan-Sverdrup K parameter for 'mixed' type confidence ellipses. -1 is interpreted as infinity. This parameter is ignored unless *gen_ellipse_type* == mixed.

C.5.22. ***gen_apriori_standard_error***

<double> [Default = 1.0000] (0.0000 <= X <= 1000.0)

The a priori standard error scale factor. It represents an estimate of the ratio between the true and actual data standard errors. This parameter only applies when K > 0.

C.5.23. ***gen_confidence_level***

<double> [Default = 0.9500] (0.0000 <= X <= 1.0000)

Uncertainty confidence level desired.

C.5.24. *allowCorePhaseRenamingP*

<boolean> [Default = false]

If *allowCorePhaseRenamingP* is true, core phase renaming will occur at the distance specified by *corePhaseRenamingThresholdDistanceP*.

C.5.25. *corePhaseRenamingThresholdDistanceP*

<double> [Default = 110.] (0.0000 <= X <= 180.0000)
If *allowCorePhaseRenamingP* is true, then this is the distance in degrees beyond which core phase renaming will take place.

C.5.26. *useSimplex*

<boolean> [Default = false]

If true, then after computing the best fit location in the standard manner, the Simplex algorithm is applied to the previous best fit solution. If the Simplex finds a better solution (lower rms residuals) then the Simplex location is retained; otherwise, the standard solution is retained. The advantage of the Simplex algorithm is that it does not require derivatives of predictions with respect to source position. The disadvantage is that it can be very expensive computationally (order of magnitude longer to execute is not uncommon).

C.6. Correlated Observation Parameters

C.6.1. *gen_correlation_matrix_method*

<string> [Default = uncorrelated] (uncorrelated | file | function)

Specifies how correlation coefficients are to be specified.

If 'file', then correlation coefficients between pairs of observations are read in from a file, with the file name being specified with the *gen_correlation_matrix_file* parameter. If value = 'function', then function parameters must be specified using the *gen_correlation_scale* parameter.

Regardless of how the correlation coefficients are entered, they are used to populate an N x N matrix where N is the number of observations. The correlation matrix will have ones on the diagonal and user specified values between -1 and 1 in the off-diagonal elements. The diagonal elements of this matrix will be multiplied by the square of the total uncertainty of each observation (combined model and pick error). The off-diagonal elements are multiplied by the product of the model errors for the two observations (pick error is not included for off-diagonal elements).

C.6.2. *gen_correlation_matrix_file*

<string> [Default =] (any valid file path+name)

Specifies the name of the file from which correlation coefficients are to be read. The file consists of an arbitrary number of lines, each of which defines the correlation coefficient between two observations. Each line must contain the following information:

Station_1/phase_1/type_1 station_2/phase_2/type_2 corr_coeff

where type_1 and type_2 are the observation types (TT, AZ or SH for travel time, azimuth and slowness), and corr_coeff is the correlation coefficient (-1 <= corr_coeff <= 1).

For example, ARCES/P/TT FINES/P/TT 0.5 would specify a correlation coefficient of 0.5 between all P wave travel time observations from ARCES and FINES. Correlation coefficients between pairs of observations which are not specified in the file are set to zero.

C.6.3. *gen_correlation_scale*

<double> [Default = 10 degrees] (> 0 degrees)

Specifies the correlation scale length for calculating the correlation coefficients between pairs of observations (degrees).

If *gen_correlation_matrix_method* is 'function' then the correlation coefficients between pairs of stations are computed from:

$$c = \exp(-(\Delta / scale)^2)$$

where *c* is the correlation coefficient, Δ is the separation of the two stations where the observations were made, in degrees, and *scale* is the scale length specified with the *gen_correlation_scale* parameter. The scale length is in degrees. The default is 10 degrees.

C.7. Levenberg-Marquardt Non-Linear Least Squares Solver

Note that the following default parameters are recommended for running LocOO3D. Modify with caution.

C.7.1. *lsq_print_iteration_table*

<bool> [Default = true]

If false, iteration tables are not sent to general output.

C.7.2. *lsq_convergence_n*

<int> [Default = 2] (1 <= X <= 100000)

Number of consecutive times convergence criterion must be satisfied before convergence is declared.

C.7.3. *lsq_applied_damping_multiplier*

<double> [Default = 10.000] (1.0 <= X <= 1e6)

If the initial applied damping does not reduce the sum squared weighted residuals to a level below that observed in the previous iteration, keep multiplying the applied damping by this factor until it does, or until the damping is so large that the solution stops moving (see *lsq_damping_dkm_threshold*).

C.7.4. *lsq_convergence_criterion*

<double> [Default = 0.0001] (0 <= X <= 1e6)

Threshold convergence criterion. At the conclusion of each iteration the ratio of the sum squared residual at the conclusion of the current iteration to the sum squared residual at the conclusion of the previous iteration is calculated. One is subtracted from the result and the absolute value evaluated. If the resulting quantity is less than the threshold convergence criterion, the convergence criterion is declared to have been achieved.

C.7.5. *lsq_damping_dkm_threshold*

<double> [Default = 0.0100] (0.0000 <= X <= 1e6)

During automatic damping, the applied damping will continue to increase until either the sum squared weighted residual is reduced to a level less than that observed in the previous iteration, or until the applied damping is so large that the solution stops moving. The quantity dkm is the amount, in km, that the solution will move during an iteration. When dkm becomes less than, lsq_damping_dkm_threshold, it can be concluded that the sum squared weighted residuals cannot be further reduced and convergence can be declared.

C.7.6. *lsq_damping_factor*

<double> [Default = -1.000] (-1.000 <= X <= 1e6)

Damping factor to be applied to singular values. For AUTOMATIC DAMPING: lsq_damping_factor = -1 [DEFAULT]. This factor controls application of the Levenberg-Marquardt algorithm which helps the locator converge if it is oscillating around the optimum location. Set to -1.0 to have the locator automatically adjust the damping factor as necessary. Set to 0.0 or positive values to override the default behavior.

C.7.7. *lsq_initial_applied_damping*

<double> [Default = 0.0001] (0.0000 <= X <= 1e6)

When AUTOMATIC DAMPING is being applied, this is the initial damping factor applied when an increase in the sum squared weighted residuals is observed.

C.7.8. *lsq_singular_value_cutoff*

<double> [Default = 1e-6] (0.0000 <= X <= 1e30)

Singular value cutoff. Any singular values that are less than this number times the maximum singular value will have their value set to infinity, with the result that the associated location parameter will not change from its initial value.

C.8. General Input/Output Parameters

C.8.1. *io_verbosity*

<int> [Default = 1] (0-4)

Verbosity level for progress information.
0 : no output, not even error messages. Error messages sent to output_error_file
1 : minimal output related mostly to property values, IO, and errors
2 : + basic information about the final locations
3 : + initial site and observation information
4 : + observation, prediction, and iteration tables

C.8.2. *io_log_file*

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

Full path to general output file. All header information, iteration status information, and final results are sent to the general output file. The output is in ascii text format.

C.8.3. *io_print_to_screen*

<bool> [Default = true]

Echo iteration progress and/or messages to the screen during the location calculation. This is the same information that is sent to the output text file.

C.8.4. *io_error_file*

<string> [Default = locoo_errors.txt]

Full path to general output error file. All error messages generated during locoo execution are sent to this file.

C.8.5. *io_print_errors_to_screen*

<bool> [Default = io_verbosity > 0]

Write error messages to the screen.

C.8.6. *io_observation_tables*

<int> [Default = 2] (0 <= X <= 100000)

Maximum number of observation and prediction tables to output when io_verbosity >= 4.
0 = No tables are output
1 = Tables output only on final iteration
2 = Tables output only on first and final iterations
3 = Tables output only on first, second and final iterations
4 = Tables output only on first three iterations + final iteration
etc...

This parameter is ignored if io_verbosity < 4.

C.8.7. *io_observation_sort_order*

<string> [Default = distance] (distance | station_phase | weighted_residual)

Specifies the order in which observations are reported in the observation and prediction tables in the text output file. If station_phase is selected, the columns will be sorted alphabetically descending by station, then by phase. If weighted_residual is selected, the columns will be sorted from the smallest to largest absolute value of the weighted_residual.

C.8.8. ***io_iteration_table***

<boolean> [Default = true]

Whether or not to print the iteration table when io_verbosity >= 4.

C.8.9. ***io_nondefining_residuals***

<boolean> [Default = true]

If true, then the residuals of all observations with valid observed values are computed prior to writing results to the database, regardless of whether they are defining or not. Setting this parameter to false will not impact the final computed location.

C.9. DataLoader Utility

C.9.1. ***dataLoaderType***

<string> [Default = None] (file | oracle)

Specifies whether to perform IO with text files or with an Oracle database. For descriptions of parameters related to file IO see section DataFileLoader Utility. For descriptions of parameters related to database IO see section DBIO Utility.

C.10. DataFileLoader Utility

C.10.1. ***dataLoaderFileInputOrigins***

<string> [Default = None]

Name of file containing the input origins. Required.

C.10.2. ***dataLoaderFileInputAssocs***

<string> [Default = None]

Name of file containing the input assocs. Required.

C.10.3. ***dataLoaderFileInputArrivals***

<string> [Default = None]

Name of file containing the input arrivals. Required.

C.10.4. ***dataLoaderFileInputSites***

<string> [Default = None]

Name of file containing the input sites. Required.

C.10.5. ***dataLoaderFileOrids***

<string> [Default = None]

A list of orids to process. Optional. If not specified, then all origins in the file are processed.

C.10.6. ***dataLoaderFileOutputOrigins***

<string> [Default = None]

Name of file to receive the output origins.

C.10.7. ***dataLoaderFileOutputOrigerrs***

<string> [Default = None]

Name of file to receive the output origerrs.

C.10.8. ***dataLoaderFileOutputAssocs***

<string> [Default = None]

Name of file to receive the output assocs.

C.10.9. ***dataLoaderFileOutputAzgaps***

<string> [Default = None]

Name of file to receive the output azgaps.

C.10.10. ***dataLoaderFileOutputArrivals***

<string> [Default = None]

Name of file to receive the output arrivals. The values of these arrivals are unchanged from the input arrivals, but their format may have changed.

C.10.11. ***dataLoaderFileOutputSites***

<string> [Default = None]

Name of file to receive the output sites. The values of these sites are unchanged from the input sites, but their format may have changed.

C.10.12. ***dataLoaderFileInputTokenDelimiter***

<string>   [Default = tab]

Assembles the token delimiter for input from the specified value.

The specified value is a space delimited string that uses "tab" for "\t", "comma" for ",", and "space" for " ". Any other character can be represented also. For example, if the desired delimiter is ",\t *" then the input string would be "comma tab space *". Using the long names for whitespace characters was necessary since the input tokenDelimiter is read from a properties file.

C.10.13. ***dataLoaderFileOutputTokenDelimiter***

<string>   [Default = same value as *dataLoaderFileTokenDelimiter*]

Assembles the token delimiter for output from the specified value.

The specified value is a space delimited string that uses "tab" for "\t", "comma" for ",", and "space" for " ". Any other character can be represented also. For example, if the desired delimiter is ",\t *" then the input string would be "comma tab space *". Using the long names for whitespace characters was necessary since the input tokenDelimiter is read from a properties file.

C.11. DBIO Utility

C.11.1. ***dbInputUserName***

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

Database input account usernames.

C.11.2. ***dbOutputUserName***

<string>   [Default = none]

Database output account usernames. If not specified, no output is written to the database.

C.11.3. ***dbInputPassword, dbOutputPassword***

<string>   [Default = user's environment variable DBTOOLS_PASSWORD or UserName]

Database input/output account passwords. If not specified in the property file, and the property DBTOOLS_PASSWORD is specified in the user's environment then the value from the environment is used. If not specified in either the property file or the user's environment, then the value of dbInputUsername/dbOutputUserName is used.

C.11.4. ***dbInputInstance, dbOutputInstance***

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

Database instance for input/output.

C.11.5. ***dbInputDriver, dbOutputDriver***

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

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

C.11.6. ***dbInputTableTypes***

<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. Currently recognized table types include: origin, assoc, arrival, site.

C.11.7. ***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.

C.11.8. ***dbInputOriginTable***

<string> [Default not allowed]

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

C.11.9. ***dbInputAssocTable***

<string> [Default not allowed]

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

C.11.10. ***dbInputArrivalTable***

<string> [Default not allowed]

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

C.11.11. ***dbInputSiteTable***

<string> [Default not allowed]

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

C.11.12. ***dbInputWhereClause***

<string> [No default value]

An orid query "where" clause that specifies the origins that should be processed as input for LocOO3D. This where clause is executed against the origin table.

LocOO3D executes the following sql statement in order to load data from the database. First, it executes something similar to:

select orid, ndef from <*dbInputOriginTable*> where <*dbInputWhereClause*> order by ndef desc

With the output, it forms the orids into batches where each batch will have roughly the same number of defining phases and the batches tend to be in order increasing number of orids. The number of defining phases per batch can be specified with property *batchSizeNdef*, which defaults to 1000. This arrangement is most efficient when processing predictions in parallel.

Orids that have more defining phases than *batchSizeNdef* will form their own batch. Once all of those origins have been formed into batches, other batches will have no more than *batchSizeNdef* defining phases in them. No batch will have more than 1000 orids in it due to Oracle limitations. For each batch of orids generated during step 1, LocOO3D executes 2 sql statements such as:

select origin.orid, evid, lat, lon, depth, time from <*dbOriginTable*> where orid in (<*list of orids in batch*>)

select orid, assoc.arid, -1, site.sta, site.lat, site.lon, site.elev, site.ondate, site.offdate, assoc.phase, arrival.time, arrival.deltim, assoc.timedef, arrival.azimuth, arrival.delaz, assoc.azdef, arrival.slow, arrival.delslo, assoc.slodef from <*dbInputAssocTable*> assoc, <*dbInputArrivalTable*> arrival, <*dbInputSiteTable*> site where orid in (<*orids in batch*>) and assoc.arid=arrival.arid and arrival.sta=site.sta and arrival.jdate between site.ondate and site.offdate and <*dbInputAssocWhereClause*>

C.11.13. ***dbInputAssocClause***

<string> [empty string]

An optional phrase that will be appended onto the end of the where clause that selects rows from the assoc, arrival and site tables.

See dbInputWhereClause for details of how this is used. For example, to include only data from stations within distance of 40 degrees from the origin, specify dbInputAssocClause = assoc.delta < 40. To limit the data to include only stations ABC and XYZ, specify dbInputAssocClause = site.sta in ('ABC', 'XYZ'). This where clause becomes part of a sql statement that is executed against an assoc, arrival and site table.

C.11.14. ***batchSizeNdef***

<int> [1000]

The approximate number of defining phases that will be included in each batch of origins that will be loaded and processed. See *dbInputWhereClause* for more details.

C.11.15. ***dbOutputTablePrefix***

<string>   [Default =]

If this parameter is specified, then the output table types specified with the dbOutputTableTypes parameter will default to the value of this parameter with the appropriate table type appended to the end.

C.11.16. ***dbOutputTableTypes***

<string>   [Default =]

If the dbOutputTableTypes parameter is specified, then the output table types specified with this parameter will default to the value of the dbOutputTablePrefix parameter with the appropriate table type appended on the end. Currently recognized table types include: origin, assoc, arrival, site, origerr and azgap.

C.11.17. ***dbOutputOriginTable***

<string>   [Default = none]

Name of the origin table where output is to be written. Specifying this parameter will override any default values set by other parameters.

In general, the value of orid in the output origin table will be set to a value that is unique in the output table. However, if parameter *dbOutputConstantOrid* is set to true, then the orid value from the input origin row is retained.

Note that evids are never changed by LocOO3D; evid in the output origin row will be equal to evid in the input origin row.

C.11.18. ***dbOutputArrivalTable***

<string>   [Default = none]

Name of the arrival table where output is to be copied. Specifying this parameter will override any default values set by other parameters.

C.11.19. ***dbOutputAssocTable***

<string>   [Default = none]

Name of the assoc table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.20. ***dbOutputAzgapTable***

<string>   [Default = none]

Name of the azgap table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.21. *dbOutputOrigerrTable*

<string> [Default = none]

Name of the origerr table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.22. *dbOutputAuthor*

<string> [Default = '-']

Name of the output author. This is used to populate the AUTH field of the new origin row.

C.11.23. *dbOutputConstantOrid*

<bool> [Default = false]

If true, then the value of orid in the output table will be unchanged from the value in the input origin table. If false, the value of orid will be set to a value that is unique in the output origin table.

C.11.24. *dbOutputAutoTableCreation*

<bool> [Default = false]

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

C.11.25. *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.

C.11.26. *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.

APPENDIX D. SUPPORT MAP

LibCorr3D models are generally stored in a directory with a large number of LibCorr3D models and such a directory will be referred to as a LibCorr3D root directory. LibCorr3D models for each available seismic phase can either be stored together in the root directory or in several sub-directories beneath the root directory. Each model supports a single station, one or more seismic phases, and one or more seismic attributes (such as travel time, travel time uncertainty, etc.). When an application that uses LibCorr3D wants to retrieve a particular value from one of the models, it requests the value using information about a single station, a single seismic phase and a single attribute name. In order to service such a request, LibCorr3D maintains a SupportMap, which stores the linkages between station-phase-attribute triples and the model capable of providing the requested value.

The linkages are stored in a text file in the LibCorr3D root directory called _supportMap.txt and this file is read when LibCorr3D is instantiated. Each line in the _supportMap.txt file contains the following information:

modelName [station info] phase attribute

where [*station* info] consist of *sta, refsta, ondate, offdate, latitude, longitude, elevation*

Here are a few sample records from a typical _supportMap.txt file:

```
AAK_Pmantle.geotess AAK   AAK   2007065 2286324  42.639100000   74.494200000  1.645000 P   TT_DELTA_AK135
AAK_Pmantle.geotess AAK   AAK   2007065 2286324  42.639100000   74.494200000  1.645000 P   TT_MODEL_UNCERTAINTY
AAK_Pmantle.geotess AAK   AAK   2007065 2286324  42.639100000   74.494200000  1.645000 Pn  TT_DELTA_AK135
AAK_Pmantle.geotess AAK   AAK   2007065 2286324  42.639100000   74.494200000  1.645000 Pn  TT_MODEL_UNCERTAINTY
```

These records indicate that the model *AAK_Pmantle.geotess* supports requests for *TT_DELTA_AK135* and *TRAVEL_TIME* for phases P and Pn and site AAK with *ondate* 2007065.

The first time LibCorr3D is used with a particular root directory, if the root directory does not contain a file called _supportMap.txt, one is created by scanning all the models in the root directory, and all of its subdirectories, extracting the necessary information from the model files (not the file names!), and populating the _supportMap.txt file with the information.

Problems can arise, however. The most important issue is that the default _supportMap.txt file will allow retrieval of values only if the request includes a site that is equal to the site used to generate the model. Note that equality of sites is based on the station name (sta) and the station ondate. It often occurs that the site information associated with an observation is not exactly the same as the site information stored in the model, even when it is appropriate to retrieve model attribute values from the model. For example, an observation may be associated with a site entry with a different ondate, even when the station did not move significantly. Or, an observation may be associated with a specific element of an array, but the model only contains values for the array's reference station. One more situation is when a station is renamed for some reason. In all of these cases, users can modify the _supportMap.txt file and add linkages between LibCorr3D models and station-phase-attribute triples, as they see fit.

While users can manually modify _supportMap.txt files, this can be a very cumbersome process if many modifications need to be made. To facilitate generation of custom _supportMap.txt files, an application called `supportmap` is provided. `supportmap` takes as input a LibCorr3D root directory and a site table from a database. It will associate models in the root directory with sites loaded from the site table and create support map links if the two sites are deemed to be similar, where similarity is assessed in one of two ways. In one case, the sites are similar if they share the same refsta. In the second case, the sites are similar if their locations are separated by less than a user-specified distance threshold (e.g., 10 km). When a database site and a model site are deemed to be similar, entries are added to the _supportMap.txt file for the database site and all the supported phases and attributes in the LibCorr3D model.

This page left blank

DISTRIBUTION

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name
Mike Begnaud	mbegnaud@lanl.gov	Los Alamos National Laboratory
Sanford Ballard	sballard999@gmail.com	Retired

Email—Internal

Name	Org.	Sandia Email Address
Stephanie Teich-McGoldrick	06756	steichm@sandia.gov
John Merchant	06752	bjmerch@sandia.gov
Rigobert Tibi	06752	rtibi@sandia.gov
Steve Vigil	06752	srvigil@sandia.gov
Nathan Downey	06752	njdowne@sandia.gov
Andrea Conley	06752	acconle@sandia.gov
Technical Library	01177	libref@sandia.gov

This page left blank