SANDIA REPORT
SAND2023-05864
Printed June 2023

Sandia National Laboratories

LocOO3D User's Manual

Kathy Davenport[1], 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 data format used in monitoring applications. The LocOO3D software, User's Manual, and Examples are available on the web at:

https://github.com/sandialabs/Salsa3DSoftware

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
RSTT	Regional seismic travel time (RSTT and SLBM are synonymous)
SLBM	Seismic location base model (RSTT and SLBM are synonymous)

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 in flat files. Additionally, data can be input as custom formatted text if appropriate column labels are included in a header file (see Appendix A).

LocOO3D is distributed through GitHub at https://github.com/sandialabs/Salsa3DSoftware. In addition to LocOO3D, the tools GeoTess (used to build and interact with GeoTess velocity models) and PCalc (used for raytracing and travel-time computation using GeoTess velocity models) are provided through the same GitHub distribution.

LocOO3D is packaged with the latest version of this user's manual and a set of run examples described in Section 4. To compile LocOO3D from the source code, the user will need to have the Maven software package (https://maven.apache.org/index.html) and Java ver. ≥ 10 installed.

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

- The ability to include observed arrival time, back-azimuth, and horizontal slowness in location calculations.

- 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 custom 3D velocity model in GeoTess format (see documentation at https://github.com/sandialabs/GeoTessJava, https://github.com/sandialabs/Salsa3DSoftware, www.sandia.gov/geotess for details), use SALSA3D or RSTT GeoTess models (www.sandia.gov/salsa3d, www.sandia.gov/rstt), or travel-time lookup surfaces for event location.

- In conjunction with the software PCalc (provided with LocOO3D at https://github.com/sandialabs/Salsa3DSoftware), 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. In addition to the standard error ellipse computation, LocOO3D can compute fully 3D error estimations using a grid-search around the computed origin location.

- Multiple events can be located in a single software run either in 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 for the event. An **arrival** is an observation of the arrival time, back-azimuth, and horizontal slowness of a seismic phase at a particular seismic station. Arrivals are associated with an origin through an association ("**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. Details about the input data formats are described in Appendix A.

A typical run of LocOO3D consists of the following steps:

1. Input an event and velocity model. If the event does not have a starting origin, LocOO3D will generate a default origin (see gen_initial_location_method in Appendix C for details.)

2. Read the association table to get the arrivals associated with that origin. Note that the choice of arrivals used in the event location, called the defining arrivals, can be modified using run properties. There must be at least one defining arrival for LocOO3D to run, and there must be at least as many defining observations (travel time, back-azimuth, and horizontal slowness) as the number of properties that can change in the new origin (latitude, longitude, depth, and time).

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.

In the following sections, an introduction to relevant LocOO3D concepts will be provided, followed by installation instructions, and a thorough tutorial of LocOO3D functionalities.

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). Details of the LocOO3D implementation and convergence criteria can be found in the accompanying document, lsq_algorithm.pdf. In that document we review the mathematical basis of the iterative linear least squares inversion algorithm and discuss the major assumptions made during its derivation. We go on to explore the utility of using Levenberg-Marquardt damping to improve the performance of the algorithm in cases where some of these assumptions are violated. We also describe how location uncertainties are calculated.

2.1. Pseudo-bending in LocOO3D

The ray tracing algorithm in LocOO3D, aka "Bender", is an implementation of the pseudo-bending algorithm of *Um and Thurber* (1987), *Zhao and Lei* (2004), and *Ballard et al* (2009). The algorithm has been adapted to work with GeoTess models, which can contain interfaces separating regions of smoothly varying velocity, by ensuring that Snell's law is satisfied at these interfaces. This algorithm is described in detail in *Ballard, et al.* (2009). Bender is the same ray tracer/travel time predictor used in the construction of the SALSA3D velocity models.

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

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

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

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

LocOO3D can use Bender to compute seismic event locations using ray paths computed for arbitrary Earth models in GeoTess format. The companion software PCalc can also use Bender to compute travel times, ray paths, and uncertainties through any Earth velocity model that can be represented using the GeoTess format.

2.2. seismicBaseData

LocOO3D often needs access to 2-dimensional lookup tables for travel time, azimuth, and slowness information. The 2 dimensions refer to distance and depth. Lookup tables for azimuth and slowness are rarely used but travel time is used very frequently. Travel time lookup tables for a great variety of radially symmetric velocity models have been generated but tables for models ak135 and iasp91 are the most common.

LocOO3D expects 2D lookup tables to be stored in a directory structure called seismicBaseData where a directory called seismicBaseData contains subdirectories tt (travel time), az (azimuth), sh (horizontal slowness), and el (ellipticity corrections). Within each of those subdirectories would be

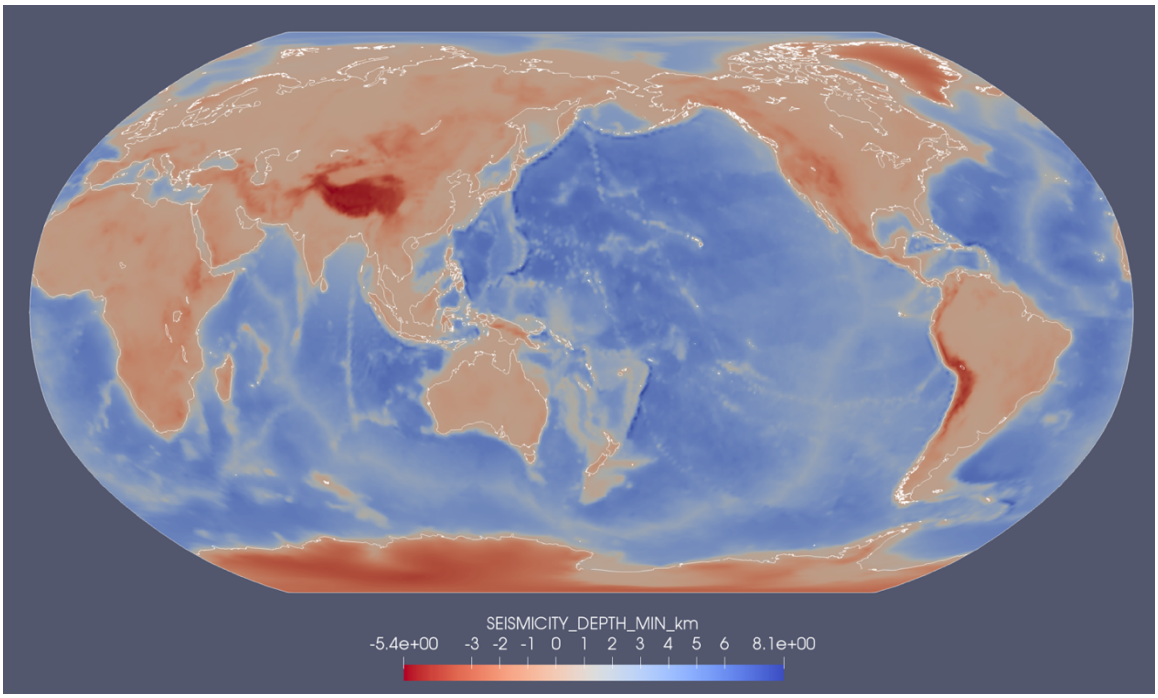
another level of subdirectories with names that correspond to a particular model, e.g., ak135, iasp91, etc. Within each of those subdirectories would be files with names that correspond to a particular phase. For example, a very commonly used lookup table would reside in seismicBaseData/tt/ak135/Pn.

While LocOO3D can access seismicBaseData in an external directory on the user's file system, LocOO3D includes a copy of seismicBaseData in the LocOO3D jar file. These include travel time and ellipticity corrections for model ak135 and travel-time tables for iasp91.

LocOO3D property seismicBaseData (Appendix **Error! Reference source not found.**) can be used t o specify the path to a directory on the user's file system where lookup tables reside. If property seismicBaseData is set to 'seismic-base-data.jar' or is omitted altogether, the lookup tables stored in the LocOO3D jar file generated in Section 3 will be used.

2.3. Seismicity Depth Model

LocOO3D includes a seismicity depth GeoTess model in the LocOO3D jar file that specifies the allowable depth range of free depth solutions as a function of geographic location. The upper limit (seismicity_depth_min) is defined as the topography/bathymetry of the Earth and ranges from -5.4 to 8.1 km depth. The lower limit (seismicity_depth_max) is based on the maximum depth of historic seismicity and ranges from 50 to 700 km depth. Maps of the upper and lower limits of the default seismicity depth model are shown below:



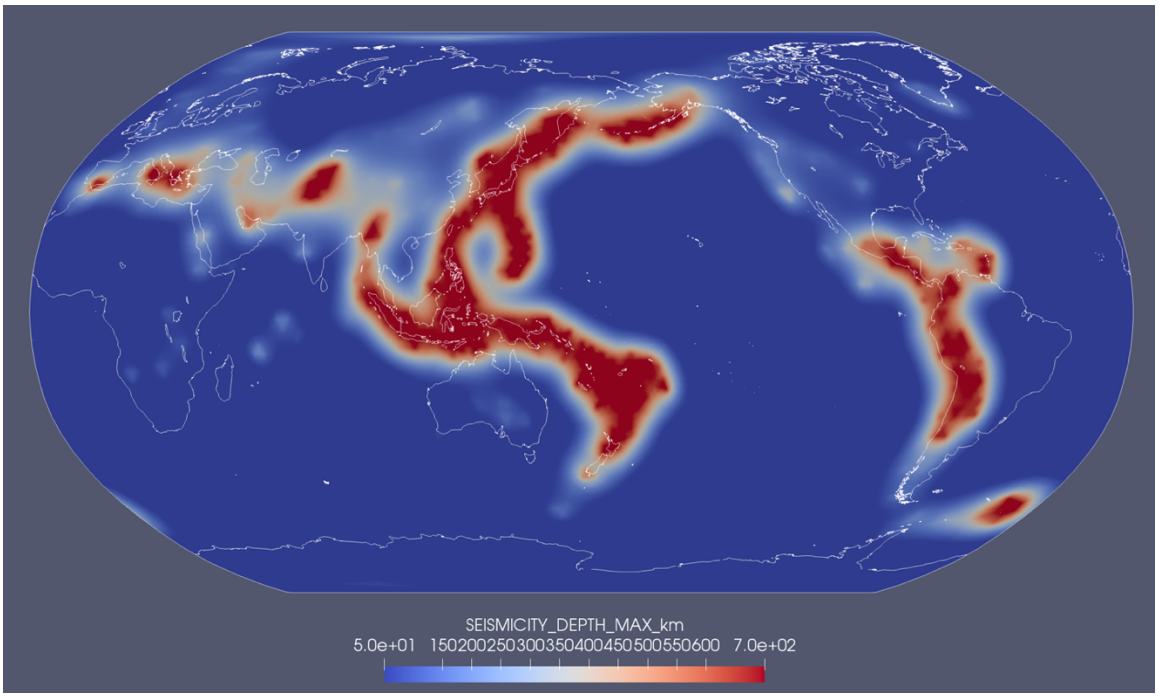


Figure 1. Seismicity depth model. (Top) Upper boundary based on elevation. (Bottom) Lower boundary based on historic seismicity.

The seismicity depth model is not enforced by default. The default behavior is to constrain event depths to be between 0 and 700 km depth. To use the default seismicity depth model shown in Figure 1 or to use another seismicity depth model, the user can use the property seismicity_depth_model (Appendix C.6.2).

3. LOCOO3D INSTALLATION

To install LocOO3D, clone the Salsa3DSoftware package located at https://github.com/sandialabs/Salsa3DSoftware to the desired location. This will create a Salsa3dSoftware directory. Once the directory is cloned, run the following steps:

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

   mvn clean package

   This command creates the jar file **salsa3d-software-1.2023.4-jar-with-dependencies.jar** in the target folder (~/Salsa3DSoftware/target). Note that this jar file contains the classes for all three Salsa3DSoftware packages (LocOO3D, GeoTess, PCalc).
2. Run the **configure.sh** script located in ~/Salsa3DSoftware. This script creates executable shell scripts that can be used to access and run the desired tool contained within the jar file. In this case, the necessary executable is the **locoo3d** shell script.
3. The configure.sh script also outputs a recommendation on how to update a **.bash_profile** or similar shell configuration file to include the generated executables on the user's path. For example:

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

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

4. 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 properties settings used by LocOO3D for a particular run. The format of the properties file and a list of properties that can be set are described in Appendix C.

2. A set of files that contain the data used by LocOO3D to constrain event locations. These files include an **origin** file, containing information about the starting origin used in the location process; an **arrival** file, which contains information about the time and type of arrival at each station; an **assoc** file, which associates the arrivals in the arrival file to the origins in the origin file; and, finally, a **site** 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.

The Examples directory includes demonstrations of five ways to run LocOO3D. The data for the examples consist of a single origin occurring in the Arctic Ocean near Greenland with arrivals recorded on permanent seismic stations. These arrivals consist of mantle (P, Pn, Sn) and core (PcP, PKPdf, PKPbc) phases. Note that while we only use one origin in our examples, LocOO3D can locate multiple events using whatever phases are supported in the selected velocity model. Thus, the origin, association, and arrival files can contain information for multiple events, each with multiple associated arrivals.

EXAMPLE 1:
Locating an event with travel time predictions from built-in ak135 travel time tables.

EXAMPLE 2:
Locating an event with travel time predictions from raytracing through SALSA3D, a 3D velocity model in GeoTess format. Requires SALSA3Dv2.1 model available from https://www.sandia.gov/salsa3d/velocity-models/.

A softlink to the model file can be created so the full path does not need to be entered in each property file requiring it. To do this, run:

ln -s /path/to/model/file/SALSA3Dv2.1.geotess salsa3d_model

EXAMPLE 3:
Locating an event with travel time predictions based on LibCorr3D corrections to ak135 traveltime tables. Requires traveltime tables from www.sandia.gov/salsa3d/Travel%20Time%20Tables.html.

EXAMPLE 4:
Locating an event with travel time predictions from the RSTT model using path-dependent uncertainty estimates. Requires RSTT model available from www.sandia.gov/rstt.

EXAMPLE 5:
Example 5 is like Example 1, but this version uses input data from a database. To run this example the user will need to edit the example.properties file to specify correct database connection properties and read input from CSS3.0 formatted database tables accessible to the user.

4.1. Example 1 – Location Using Internal Lookup Tables

To execute Example 1, cd into ~/Salsa3DSoftware/Examples/LocOO3D/file_io_simple. In this directory, the properties file **example.properties** (Figure 2) and the original output file **locoo3d_output_original.txt** are provided.

```
################################################

parallelMode = sequential
maxProcessors = 1

executionPath = .

# Regular and error logs
io_log_file   = <property:executionPath>/locoo3d_output.txt
io_error_file = <property:executionPath>/locoo3d_error.txt
io_print_to_screen = true
io_verbosity = 4

# predictorType = [ lookup2d | bender | slbm ]
loc_predictor_type = lookup2d

# General
gen_fix_depth = true

#####################################################################
#
# IO Files
#
#####################################################################

dataLoaderInputType = file
dataLoaderOutputType = file

dataLoaderFileInputOrigins = <property:executionPath>/../Data/origins.txt
dataLoaderFileInputAssocs = <property:executionPath>/../Data/assocs.txt
dataLoaderFileInputArrivals = <property:executionPath>/../Data/arrivals.txt
dataLoaderFileInputSites = <property:executionPath>/../Data/sites.txt

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

Figure 2. LocOO3D Simple File IO Example Properties File.

This example demonstrates event relocation using the 2D ak135 travel-time lookup tables included with the LocOO3D software.

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

LocOO3D can run sequentially or in parallel on suitably equipped machines (see Appendix B for details). To set the run to sequential or parallel, as well as the number of processors to use during computation, define the properties parallelMode (Appendix C.5.2) and maxProcessors (Appendix C.5.1), respectively. In this example, these properties are set to run in serial mode on a single processor.

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

The property executionPath (Appendix **Error! Reference source not found.**):

```
executionPath = .
```

specifies that the example is to be run in the current directory.

The syntax <property:propertyName> can be used to reference properties defined earlier in the input file. In this example, executionPath is referenced to specify the directory for the output (io_log_file; Appendix C.11.2) and error (io_error_file; Appendix C.11.4) log files:

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

The properties io_print_to_screen (Appendix C.11.3) and io_verbosity (Appendix 0):

```
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. Note that a high verbosity can result in memory issues if a high number of events are being located. It is recommended that lower verbosity is used when multiple events are being processed. Throughout the examples in this manual the maximum verbosity of 4 will be used for demonstration purposes.

The next property, loc_predictor_type (Appendix C.2.1.1):

```
loc_predictor_type = lookup2d
```

specifies the type of travel-time predictor to use during the location process. The available predictors are:

- lookup2d – perform predictions with the ak135 model included in the LocOO3D jar file.
- bender – perform predictions through an input model using the Bender ray-tracing algorithm. See Section 2.1 for details on Bender.
- rstt – perform predictions through an input model using the Regional Seismic Travel Time (RSTT) method (see link). rstt is the former name of RSTT.

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

In this example, the 2D internal travel-time lookup tables included with LocOO3D are applied to all phases by setting loc_predictor_type equal to lookup2d.

In LocOO3D, any of the four components of the event location (origin depth, latitude, longitude, and/or time) 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 input origin data. In this example, a fixed depth solution with origin latitude, longitude, and time as free properties is solved for. This action is performed by setting the gen_fix_depth (Appendix C.6.1) property to true:

```
gen_fix_depth = true
```

This property will fix event origin depth to the value reported in the input text file set using the following input properties.

First, the properties dataLoaderInputType (Appendix C.12.2) and dataLoaderOutputType (Appendix C.12.3) are both set to file:

```
dataLoaderInputType = file
dataLoaderOutputType = file
```

indicating the input data for LocOO3D will be read in from text files and the output data produced will also be text files.

For this example, all input text files are in ~/Salsa3DSoftware/Examples/LocOO3D/Data. These input text files are flat files containing origin (origins.txt), assoc (assocs.txt), arrival (arrivals.txt), and site (sites.txt) data in CSS3.0 format. To input these text files, the user sets the properties dataLoaderFileOrigins (Appendix C.13.1), dataLoaderFileAssocs (Appendix C.13.2), dataLoaderFileArrivals (Appendix C.13.3), and dataLoaderFileSites (Appendix C.13.4) to point to each relevant text file.

```
dataLoaderFileOrigins = <property:executionPath>/../Data/origins.txt
dataLoaderFileAssocs = <property:executionPath>/../Data/assocs.txt
dataLoaderFileArrivals = <property:executionPath>/../Data/arrivals.txt
dataLoaderFileSites = <property:executionPath>/../Data/sites.txt
```

Finally, the output path and file names are defined. Four output table types can be written by LocOO3D: 1) origin, 2) association, 3) origin error, and 4) azimuthal gap. These will be written to the path and file name specified by the dataLoaderFileOutput properties specified for each desired table type (dataLoaderFileOutputOrigins, Appendix C.13.6; dataLoaderFileOutputAssocs , Appendix C.13.8; dataLoaderFileOutputOrigerrs, Appendix C.13.7; dataLoaderFileOutputAzgaps, Appendix C.13.9).

```
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 property (Appendix C.13.11) for other options.

With all properties set, the properties file in Figure 2 can be run as **locoo3d example.properties.**

This command will call locoo3d and execute all properties described in this section. 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 including the locoo3d_output.txt file, flat files in CSS3.0 schema for each requested table type (here assocs_output.txt, azgaps_output.txt, origerrs_output.txt, and origins_output.txt), and an error log (locoo3d_error.txt).

The locoo3d_output.txt file contains a large amount of information, including default values, the input data, travel time predictions, iteration tables, and output location results. For the sake of brevity, we will only discuss the iteration table (Figure 2) and the location solution summary (Figure 3) shown near the end of the log file. 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 file_io_simple directory for comparison.

The output for this location computation shows that the location algorithm converged to a solution in 4 iterations. The iteration table (Figure 3) summarizes the location and inversion information for each iteration during the relocation, including how much the location changed relative to the initial input origin. Information on each column in the iteration table is provided in Appendix D.1

Iteration Table:

Itt	It	Comment	N	M	Lat	Lon	Depth	Time	rms_Trsd	rms_Wrsd	dNorth	dEast	dZ	dT
1	1	start	6	3	79.7625	2.4637	0.000	0.000	1.4131	0.7525	-23.683	4.266	0.000	-0.599
2	2	start	6	3	79.5503	2.6743	0.000	-0.599	0.8261	0.4848	-2.672	0.688	0.000	0.007
3	3	start	6	3	79.5264	2.7082	0.000	-0.592	0.8161	0.4834	-0.012	0.023	0.000	0.001
4	4	start	6	3	79.5263	2.7093	0.000	-0.591	0.8160	0.4834	-0.000	0.000	0.000	0.000
4	4	damped	6	3	79.5263	2.7093	0.000	-0.591	0.8160	0.4834	0.000	0.000	0.000	0.000

Itt	It	dkm	dxStart	dzStart	dtStart	azStart	nF	damp	converge
1	1	24.5369	24.0641	0.0000	-0.5992	169.7887	1	-4	0.00e+00
2	2	2.7594	26.8154	0.0000	-0.5920	169.3321	2	-5	5.85e-01
3	3	0.0267	26.8315	0.0000	-0.5914	169.2877	3	-5	5.69e-03
4	4	0.0005	26.8319	0.0000	-0.5914	169.2870	4	-5	5.37e-07
4	4	0.0000	26.8315	0.0000	-0.5914	169.2877	6	-5	0.00e+00

Figure 3. Iteration table section of the LocOO3D output for Example 1. Columns are shown here in two rows for clarity.

The solution summary table (Figure 4) includes the final computed event location, time, and misfit information. Since this was a fixed-depth location, the final solution will only report the properties 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. Information on each entry in the solution summary table is provided in Appendix D.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

  geographic region: GREENLAND SEA    seismic region ARCTIC ZONE

  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:

    smajax    sminax     trend      area
   79.8311   52.9151    4.0541  13270.94

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2458


Time to compute this location = 0.244179 seconds
```

Figure 4. Summary Section of the LocOO3D Output for Example 1.

4.2. Example 2 – Ray Tracing Through a GeoTess Model

To execute Example 2, cd into ~/Salsa3DSoftware/Examples/LocOO3D/file_io_bender. In this directory, the properties file **example.properties** (Figure 5) and the original output file **locoo3d_output_original.txt** are provided.

```
################################################

parallelMode = sequential
maxProcessors = 1

executionPath = .

# Regular and error logs
io_log_file    = <property:executionPath>/locoo3d_output.txt
io_error_file = <property:executionPath>/locoo3d_error.txt
io_print_to_screen = true
io_verbosity = 4

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

# Bender
benderModel = <property:executionPath>/../../salsa3d_model
benderAllowCMBDiffraction = true
benderAllowMOHODiffraction = true

benderUncertaintyType = DistanceDependent

# General
gen_fix_depth = true

#####################################################################
#
# IO Files
#
#####################################################################

dataLoaderInputType = file
dataLoaderOutputType = file

dataLoaderFileInputOrigins = <property:executionPath>/../Data/origins.txt
dataLoaderFileInputAssocs = <property:executionPath>/../Data/assocs.txt
dataLoaderFileInputArrivals = <property:executionPath>/../Data/arrivals.txt
dataLoaderFileInputSites = <property:executionPath>/../Data/sites.txt

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

Figure 5. LocOO3D Bender File IO Example Properties File.

The example in file_io_bender is nearly identical to Example 1, except that rather than locating with the internal ak135 tables, locations will be calculated by using Bender and the SALSA3Dv2.1 model. Due to this similarity, only new or modified properties will be discussed in this section. For properties in common with Example 1, refer to Section 4.2.

Here, loc_predictor_type is set to:

```
loc_predictor_type = lookup2d, bender(P,Pn,PcP,PKPbc,PKPdf)
```

Thus, Bender is applied to the phases P, Pn, PcP, PKPbc, and PKPdf while lookup2d will be applied to all other phases.

When the predictors property is set to bender, the benderModel property (Appendix C.2.3.1) needs to be defined to provide an input model. This property provides the path to the GeoTess model that Bender will use to calculate predictions of seismic observables. In the example shown in Figure 5, the user requests that Bender perform predictions using the SALSA3Dv2.1.geotess model via the softlink salsa3d_model:

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

If the salsa3d_model softlink was not created in the description of Example 2 in Section 4, the path must be modified to point to another valid GeoTess model or softlink.

In this example, the user specifies whether to allow for rays impinging on the core-mantle boundary (CMB) and the Moho using the properties benderAllowCMBDiffraction (Appendix C.2.3.3) and benderAllowMOHODiffraction (Appendix C.2.3.2), respectively. By default, both these properties are set to false. Here, they are set to true:

```
benderAllowCMBDiffraction = true
benderAllowMOHODiffraction = true
```

Thus, rays impinging on the CMB and Moho in this example will diffract.

Finally, the benderUncertaintyType property (Appendix C.2.3.5) is defined. benderUncertaintyType sets the type of travel time uncertainty desired. By default, the distance dependent uncertainty type is set to UncertaintyNAValue. When set to this value all returned travel time uncertainties have an NA_VALUE of -999999.

To get distance dependent uncertainty values, benderUncertaintyType must be set to DistanceDependent as in the current example:

```
benderUncertaintyType = DistanceDependent
```

When set to DistanceDependent, files containing 1D distance dependent uncertainty files are needed as input. By default, ak135 distance dependent uncertainties will be used. The user can also use the properties benderUncertaintyDirectory (Appendix C.2.3.6) and benderUncertaintyModel (Appendix C.2.3.7) to define the path to the desired distance dependent uncertainty files.

The remaining properties are identical to those discussed in Section 4.2. With all properties set, the properties file in Figure 5 can be run as **locoo3d example.properties.** 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 including the locoo3d_output.txt file, flat files in CSS3.0 schema for each requested table type (here assocs_output.txt, azgaps_output.txt, origerrs_output.txt, and origins_output.txt), and an error log (locoo3d_error.txt).

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 file_io_bender directory for comparison. The output solution summary table is shown in Figure 6.

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

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   79.7783    2.6590     0.000  1518056955.611   2018-02-08 02:29:15.611            2018039

  geographic region: GREENLAND SEA    seismic region ARCTIC ZONE

  converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     2     9     3     6     0     6     6    0.5049    0.4796

    az_gap  az_gap_2 station  Nsta   N30  N250
  276.1390  298.2379     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:

    smajax    sminax     trend      area
   44.8301   20.6078   49.0479   2902.35

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    1.9230
```

Figure 6. Summary Section of the LocOO3D Output for Example 2.

4.3. Example 3 – Using LibCorr3D Surfaces

To execute Example 3, cd into ~/Salsa3DSoftware/Examples/LocOO3D/file_io_LibCorr3d. In this directory, the properties file **example.properties** (Figure 6) and the original output file **locoo3d_output_original.txt** are provided.

```
################################################

parallelMode = sequential
maxProcessors = 1

executionPath = .

# Regular and error logs
io_log_file   = <property:executionPath>/locoo3d_output.txt
io_error_file = <property:executionPath>/locoo3d_error.txt
io_print_to_screen = true
io_verbosity = 4

# predictorType = [ lookup2d | bender | slbm ]
# types that come later in list supersede previous
loc_predictor_type = lookup2d

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

# General
gen_fix_depth = true

######################################################################
#
# IO Files
#
######################################################################

dataLoaderInputType = file
dataLoaderOutputType = file

dataLoaderFileInputOrigins = <property:executionPath>/../Data/origins.txt
dataLoaderFileInputAssocs = <property:executionPath>/../Data/assocs.txt
dataLoaderFileInputArrivals = <property:executionPath>/../Data/arrivals.txt
dataLoaderFileInputSites = <property:executionPath>/../Data/sites.txt

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

Figure 7. LocOO3D LibCorr3D Surfaces IO Example Properties File.

In this example, LocOO3D uses LibCorr3D surfaces rather than Bender to perform location calculations. LibCorr3D surfaces are special GeoTess models containing two attributes, tt_delta_ak135 and tt_model_uncertainty. tt_delta_ak135 represents the difference between travel times predicted using a 3D GeoTess slowness model and travel times predicted with the standard ak135 model. tt_model_uncertainty represents either 1D distance dependent travel time uncertainty or path dependent travel time uncertainty computed from the 3D model covariance matrix that results from tomographic inversion. Details on LibCorr3D are available in Ballard et al. (2016a). The companion software tool PCalc can be used to generate LibCorr3D surfaces (see PCalc manual).

Importantly, these LibCorr3D surfaces allow the user to locate events with pre-computed travel-times rather than travel-times calculated on the fly via Bender, significantly reducing calculation times (see Begnaud et al. 2022). Example 3 demonstrates this capability using travel-time lookup surfaces provided with the LocOO3D examples; these can be found in ~/Salsa3DSoftware/Examples/LocOO3D/libcorr3d_models_tt_delta_ak135. Within this folder a subfolder, Pmantle_geotess_tt_delta_ak135, contains six LibCorr3D surfaces for various stations

(e.g., MAW_Pmantle_TT.geotess). Both the subfolder and its contents are named after the phases supported by the surface, e.g., the Pmantle subfolder and its contents support the phases P and Pn.

To make use of these LibCorr3D surfaces, the following properties must be set in LocOO3D. First, the property lookup2dPathCorrectionsType (Appendix **Error! Reference source not found.**) is u sed to set the type of path corrections to apply to the predicted travel-times. Here it is set to:

```
lookup2dPathCorrectionsType = libcorr
```

When this property is set, the property lookup2dLibCorrPathCorrectionsRoot (Appendix **Error! R eference source not found.**) must also be set to point to the main LibCorr3D surfaces directory. In this example, it is set to:

```
lookup2dLibCorrPathCorrectionsRoot =
<property:executionPath>/../libcorr3d_models_tt_delta_ak135
```

The final property, lookup2dLibCorrPreloadModels (Appendix **Error! Reference source not f ound.**), specifies whether to load all available models at the beginning of the LocOO3D run or only those corresponding to the sites input via a sites.txt file (see Section 4.4 for details on the sites.txt file). Note that if a site specified in the sites.txt file is not included among the available LibCorr3D models, corrections from a nearby site (within 10 km by default) are applied instead. To modify the separation threshold between a user-specified site and an available modify site, set the property <predictor>LibCorrMaxSiteSeparation (Appendix C.3.7).

In this example, lookup2dLibCorrPreloadModels is set to false.

```
lookup2dLibCorrPreloadModels = false
```

The remaining properties are identical to those discussed in Sections 4.1, 4.2, 4.3. With all properties set, the properties file in Figure 9 can be run as **locoo3d example.properties.** 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 including the locoo3d_output.txt file, flat files in CSS3.0 schema for each requested table type (here assocs_output.txt, azgaps_output.txt, origerrs_output.txt, and origins_output.txt), and an error log (locoo3d_error.txt).

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 file_io_bender directory for comparison. The output solution summary table is shown in Figure 8.

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

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   79.7080    1.9288     0.000  1518056955.292   2018-02-08 02:29:15.292            2018039

  geographic region: GREENLAND SEA    seismic region ARCTIC ZONE

  converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     4     6     3     6     0     6     6    0.5260    0.4801

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.8657  300.3570     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:

    smajax    sminax     trend      area
   74.1208   36.1988   27.2713   8429.16

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2045


Time to compute this location = 0.178639 seconds
```

Figure 8. Summary Section of the LocOO3D Output for Example 3.

Note the decrease in computation time compared to Example 2, which used Bender to ray trace through the 3D SALSA model. With 3D model lookup surfaces, the location was calculated in 0.482501 seconds while with Bender, the location calculation took 6.718359 seconds.

4.4. Example 4 – Using RSTT

To execute Example 4, cd into ~/Salsa3DSoftware/Examples/LocOO3D/file_io_rstt. In this directory, the properties file **example.properties** (Figure 9) and the original output file **locoo3d_output_original.txt** are provided along with a subdirectory Data containing the CSS3.0 flat files, arrivals.txt, assocs.txt, origins.txt, and sites.txt. These flat files contain Pn and Sn observations for a seismic event in central Europe.

```
################################################

parallelMode = sequential
maxProcessors = 1

executionPath = .

# Regular and error logs
io_log_file   = <property:executionPath>/locoo3d_output.txt
io_error_file = <property:executionPath>/locoo3d_error.txt
io_print_to_screen = true
io_verbosity = 4

# predictorType = [ lookup2d | bender | slbm ]
# types that come later in list supersede previous
loc_predictor_type = lookup2d, slbm(Pn,Pg,Sn,Lg)

# SLBM/RSTT settings.  Retrieves the slbm model using
# environment variable RSTT_ROOT.  If this does not work,
# try RSTT_HOME, SLBM_ROOT or SLBM_HOME.
slbmModel = <env:RSTT_HOME>/models/pdu202009Du.geotess

slbmUncertaintyType = path_dependent

# General
gen_fix_depth = true

#####################################################################
#
# IO Files
#
#####################################################################

dataLoaderInputType = file
dataLoaderOutputType = file

dataLoaderFileInputOrigins = <property:executionPath>/Data/origins.txt
dataLoaderFileInputAssocs = <property:executionPath>/Data/assocs.txt
dataLoaderFileInputArrivals = <property:executionPath>/Data/arrivals.txt
dataLoaderFileInputSites = <property:executionPath>/Data/sites.txt

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

Figure 9. LocOO3D RSTT IO Example Properties File.

In this example, LocOO3D uses the 3D Regional Seismic Travel Time model (RSTT, www.sandia.gov/rstt) to predict travel times for Pg, Pn, Sg, and Lg seismic phases. This example requires that RSTT be installed and compiled and that the shell configuration file be updated to have the environmental variable RSTT_ROOT (or alternatively RSTT_HOME, SLBM_ROOT, or SLBM_HOME). The user is referred to the RSTT installation instructions; specifically refer to the instructions to compile the Java library slbmjni.

Additionally, this example requires the RSTT model file pdu202009Du.geotess as input. This file is installed with RSTT and can be found in ~/RSTT_v3.2.0/models.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 4.1, 4.2, 4.3. In this example, the loc_predictor_type property (Appendix C.2.1.1) property is updated to

apply rstt (the RSTT predictor) to regional phases (Pg, Pn, Sg, Lg), with all other phases using the lookup2d predictor:

```
loc_predictor_type = lookup2d, rstt(Pn,Pg,Sn,Lg)
```

When the predictors property is set to rstt, the rsttModel property (Appendix C.2.4.1) needs to be defined to provide an input model. This property provides the path to the GeoTess model that RSTT will use to calculate predictions of seismic observables. In the example shown in Figure 9, the user requests that RSTT perform predictions using the RSTT model with path dependent uncertainty, pdu202009Du.geotess.

```
rsttModel = <env:RSTT_ROOT>/models/pdu202009Du.geotess
```

Here the syntax <env:RSTT_ROOT> is used to reference the RSTT_ROOT environmental variable added to the shell script configuration during RSTT installation. If installed and configured properly, the rsttModel path will be read as ~/RSTT_v3.2.0/models/pdu202009Du.geotess.

The rsttUncertaintyType property (Appendix C.2.4.2) must also be defined. rsttUncertaintyType sets the type of travel time uncertainty desired. rsttUncertaintyType can be set to path_dependent, distance_dependent.

When a LibCorr3d correction surface(s) (Section 4.3) is available for a particular station-phase, the travel time uncertainty is retrieved from the LibCorr3d correction surface(s). If a LibCorr3d correction surface is *not* available for a particular station-phase, then travel time uncertainty computed by RSTT is returned.

Here, the uncertainty type is set to path_dependent:

```
rsttUncertaintyType = path_dependent
```

Finally, the input properties (dataLoaderFileInputOrigins, dataLoaderFileInputAssocs, dataLoaderFileInputArrivals, dataLoaderFileInputSites) have been updated to point to the Data folder under the file_io_rstt directory.

The remaining properties are identical to those discussed in Sections 4.1, 4.2, 4.3. With all properties set, the properties file in Figure 9 can be run as **locoo3d example.properties.** 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 including the locoo3d_output.txt file, flat files in CSS3.0 schema for each requested table type (here assocs_output.txt, azgaps_output.txt, origerrs_output.txt, and origins_output.txt), and an error log (locoo3d_error.txt).

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 file_io_bender directory for comparison. The output solution summary table is shown in Figure 10.

```
Final location for evid: -1   Orid: 8926220

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   47.6219   20.2392    13.000  1366669726.373   2013-04-22 22:28:46.373            2013112

  geographic region: HUNGARY    seismic region NORTHWESTERN EUROPE

  converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     2     6     3    12     0    12    12    0.8274    0.8021

    az_gap  az_gap_2 station  Nsta   N30  N250
   93.9531  159.1777    BRTR     8     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:

    smajax    sminax     trend      area
   21.4270   17.7118  156.4698   1192.27

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    1.6084


Time to compute this location = 0.824596 seconds
```

Figure 10. Summary Section of the LocOO3D Output for Example 4.

4.5. Example 5 – Using Oracle I/O

To execute Example 4, cd into ~/Salsa3DSoftware/Examples/LocOO3D/database_io_simple. In this directory, the properties file **example.properties** (Figure 11) and the original output file **locoo3d_output_original.txt** are provided.

```
################################################

# Regular and error logs
io_log_file   = ./locoo3d_output.txt
io_error_file = ./locoo3d_error.txt
io_print_to_screen = true
io_verbosity = 4

# predictorType = [ lookup2d | bender | slbm ]
loc_predictor_type = lookup2d

# General
gen_fix_depth = true

#####################################################################
#
# IO Database
#
#####################################################################

dataLoaderInputType = database

# assume that a database wallet location is specified in user's environment
dbInputUserName = user_name

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

dbInputWhereClause = where origin.orid = 15433650

# additional components of the where clause used to limit assocs.
dbInputAssocClause = assoc.arid in (129798118, 129979918, 129796973, 129796914, 129797143, 129973843)
```

Figure 11. LocOO3D Database IO Example Properties File.

This example solves for locations in the same way as Example 1, except that input data are pulled in from the database. If the user has access to the International Data Center (IDC) Reviewed Event Bulletin (REB), the user can recreate the output contained in locoo3d_output_original.txt.

As several properties have been detailed in prior sections, only new or modified properties will be discussed in this section. For properties in common with prior examples, refer to Sections 4.1, 4.2, 4.3, 4.4.

In this example, lookup2d is used to do predictions for all phases:

```
loc_predictor_type = lookup2d
```

The property dataLoaderInputType (Appendix C.12.2) is now set equal to database, indicating that input data will be pulled in from the database. The database is assumed to be in CSS3.0 format (Appendix A.2).

```
dataLoaderInputType = database
```

As no dataLoaderOutputType is defined, no output beyond the log files (locoo3d_error.txt, locoo3d_output.txt) are output.

In this example, it is assumed the user has an Oracle wallet with a particular formatting (see Appendix C.14.2). If the wallet exists and appropriately formatted, the user only needs to specify their wallet username using the property dbInputUserName (Appendix C.14.3).

```
dbInputUserName = user_name
```

Note that alternative wallet formatting and connecting to the database via an instance if no wallet exists are also options and are described in Appendix C.14.2.

The next properties in this example, dbInputOriginTable (Appendix C.14.10), dbInputAssocTable (Appendix C.14.11), dbInputArrivalTable (Appendix C.14.12), and dbInputSiteTable (Appendix C.14.13), are used as inputs to LocOO3D. They specify the database account and table name containing the origin, assoc, arrival, and site data to be used, respectively.

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

With the input tables specified, the user can specify which data to extract from the tables using the dbInputWhereClause (Appendix C.14.14).

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

This property operates identically to the where portion of a SQL statement, with the SQL query being created in the background by LocOO3D based on the information input by the user. A dbInputWhereClause must be specified in the properties file. See Appendix C.14.14 for details on how the SQL statement is structured.

To further refine the SQL statement, the user can also specify an optional dbInputAssocClause (Appendix C.14.15). In this example, the dbInputAssocClause is used to select certain arids, i.e., arrivals.

```
dbInputAssocClause = assoc.arid in (129798118, 129979918, 129796973, 129796914,
129797143, 129973843)
```

This clause is further appended to the SQL statement such that it is executed in addition to the dbInputWhereClause.

The remaining properties are identical to those discussed in Sections 4.1, 4.2, 4.3, 4.4. With all properties set, the properties file in Figure 11 can be run as **locoo3d example.properties.** During the run, several lines of output will be printed out to the screen. Following the run, the user should see two output text files the locoo3d_output.txt file and an error log (locoo3d_error.txt).

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 file_io_bender directory for comparison. The output solution summary table is shown in Figure 12.

```
Final location for evid: 15399370   Orid: 15433650

  latitude longitude     depth     origin_time           origin_date_gmt       origin_jdate
   79.5262    2.7089     0.000  1518056955.420   2018-02-08 02:29:15.420            2018039

  geographic region: GREENLAND SEA    seismic region ARCTIC ZONE

  converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     4     6     3     6     0     6     6    0.5287    0.4835

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.9591  302.7308     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:

    smajax    sminax     trend      area
   79.8353   52.9144    4.0499  13271.47

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2457


Time to compute this location = 0.350623 seconds
```

Figure 12. Summary Section of the LocOO3D Output for Example 5.

5. 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 include observed arrival time, back-azimuth, and horizontal slowness in location calculations.
- 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 custom 3D velocity model in GeoTess format (see https://github.com/sandialabs/GeoTessJava and www.sandia.gov/geotess for details), use the SALSA3D or RSTT GeoTess models (www.sandia.gov/salsa3d, www.sandia.gov/rstt), or travel-time lookup surfaces for event location.
- In conjunction with the software PCalc (included with LocOO3D) 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. In addition to the standard error ellipse computation, LocOO3D can compute fully 3D error estimations using a grid-search around the computed origin location.
- 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 available via GitHub at https://github.com/sandialabs/Salsa3DSoftware and requires a Java Runtime Environment ≥ 10 to run. 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 (included at https://github.com/sandialabs/Salsa3DSoftware), it forms an essential piece of a software suite 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 users of LocOO3D to get up and running with their own datasets.

REFERENCES

[1] Ballard, S. (2002), Seismic Event Location Using Levenberg-Marquardt Least Squares Inversion. United States: N. p., 2002. Web. doi:10.2172/803290.

[2] Ballard, S., J. R. Hipp and C. J. Young (2009). Efficient and Accurate Calculation of Ray Theory Seismic Travel Time through Variable Resolution 3D Earth Models, Seismological Research Letters v 80, 6 doi: 10.1785/gssrl.80.6.989.

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

[4] Begnaud, M. L., Davenport, K., Conley, A., Ballard, S., Hipp, J., and R.W. Porritt (2022), Developing a Consistent Travel-Time Framework for Comparing Three-Dimensional Velocity Models for Seismic Location Accuracy, *Pure and Applied Geophysics*, pp. 1-18.

[5] Geiger, L. (1910), Herdbestimmung bei erdbeden ans den ankfunftzeiten, K. Gessel. Wiss. Goett. V. 4, pp. 331-349.

[6] 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.

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

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

[9] Marquardt, D.W. (1963), An algorithm for least-squares estimation of nonlinear parameters, *Journal of the Society for Industrial and Applied Mathematics,* v. 11, pp. 431-441.

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

[11] 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.

[12] Um, J. and C. Thurber (1987) A fast algorithm for two-point seismic ray tracing. *Bull. Seis. Soc. Am.*, v. 77 (3), 972-986.

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

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 properties `dataLoaderInputType` (Appendix C.12.2) and `dataLoaderOutputType` (Appendix C.12.3) which must be set to "file" to input/output ascii flat files or "database" to input/output to database tables. Note that setting to "oracle" is equivalent to setting to database.

The user specifies the names of files/tables from which input data will be loaded and to which output results will be written using CSS3.0 or similar data format. Input files/tables required to run LocOO3D include **`origin, assoc, arrival,`** and **`site`**. Minimal properties for each file are described below.

The fields required for each input file/table are:

Origin input:

1. ORID - origin identifier; links to entries in the **`assoc`** and **`origerr`** files/tables
2. EVID – event identifier
3. LAT – origin latitude
4. LON – origin longitude
5. DEPTH – origin depth below ellipsoid (km)
6. TIME – origin time (Epoch time)

Assoc input:

1. ARID – arrival identifier; links to entries in the **`arrival`** table
2. ORID – origin identifier; links to entries in the **`origin`** and **`origerr`** 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:

1. STA – station code; links to entries in the **`arrival`** and **`assoc`** tables
2. LAT – station latitude
3. LON – station longitude
4. ELEV – station elevation above ellipsoid (km)
5. O–DATE - station turn on date in Julian date format YEARJUL
6. OF–DATE - station turn off date in Julian date format YEARJUL

Arrival input:

1. ARID - arrival identifier; links to entries in the **`assoc`** tabl–
2. STA - station code; links to entries in the **`site`** table–TIME - arrival time (epoch time in seconds)
3. –DATE - Julian date in YEARJUL format

4. DELTIM – Pick uncertainty in seconds
5. AZIMUTH – Station to event azimuth (geographic, only required if azimuth observation is defining)
6. DELAZ – Azimuth uncertainty (only required if azimuth observation is defining–
7. SLO - Slowness observed at station (seconds per degree, only used if slowness is defining)
8. DELSLO – Slowness uncertainty (only required if slowness observation is defining)

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. For each input file, the required fields can be specified in any order. The first line of the file must be a header line beginning with a hash "#" and naming the columns in the file. This header line is followed by the data in a series of lines matching the order of the header. Input files may be tab-, space-, or comma-delimited by setting the property `dataLoaderFileInputTokenDelimiter` (Appendix C.13.10); however, only tab-delimited files allow quotes and commas in station names.

Users can limit which origins in the input origins table are processed with the property `dataLoaderFileOrids` (Appendix C.13.5).

A.2. Oracle Database I/O

LocOO3D can interact with an Oracle Database containing database tables in CSS3.0 format. Users must either specify a combination of database instance, username, and password or specify a database wallet in the properties file (see Appendix C.14). There may be separate definitions for input and output. For input, LocOO3D can read **`origin, assoc, arrival`** and **`site`** database tables while outputting 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:

```
dbOtputTableTypes = 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.:

```
dbOutputOriginTable = my_origin
dbOutputAssocTable = my_assoc
dbOutputOrigerrTable = my_origerr
dbOutputAzgapTable = 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.

The user can specify SQL "where" clauses (`dbInputWhereClause`, Appendix C.14.14, `dbInputAssocClause`, Appendix C.14.15) that limit the data that are loaded from the 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.

See Section DBIO in Appendix C.14 for additional details.

APPENDIX B. USING LOCOO3D IN PARALLEL

LocOO3D can run 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 many arrivals. Concurrent mode is likely faster when computing many events with a relatively small number of arrivals.

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

APPENDIX C. PROPERTIES DESCRIPTIONS FOR LOCOO3D

C.1. Setting Properties

The properties required by LocOO3D are preset to default values as the application is started. These defaults are given below in the property description section. Users may apply a different property value by using a property file (e.g., test.property). Only properties whose values differ from their defaults need to be listed in the properties file, since the defaults will be activated for any property not found in the properties file.

Notes:

- LocOO3D properties are case sensitive.
- All properties in the properties file must contain an '=' character, separating the property name from the property 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
  ```

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

C.2. Predictors

C.2.1. *General*

C.2.1.1. *loc_predictor_type*

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

REQUIRED. String indicating list of travel time predictors that are to be used for each arrival. May restrict predictor to specific phases by listing them in parentheses after the predictor name. Later entries supersede earlier items in the list. For example, if the property value is "`lookup2d, bender(P, Pn), rstt(Pn, Pg)`" then RSTT will be used for phase Pn and Pg, Bender will be used for phase P, and lookup2d will be used for all other phases. Even though Pn is specified by Bender, it will be computed with RSTT since `rstt(Pn)` comes later in the list then `bender(Pn)`.

Use `lookup2d` when using ak135, other traveltime tables, or 3D correction files. Use `bender` to ray trace through a 3D GeoTess model like SALSA3D. Use `rstt` to calculate travel times through the RSTT model. All options, apart from lookup2D using the default ak135 model, require additional properties as described in this Appendix.

C.2.1.2. *sascLibraryDirectory*

<string> [Default = none]

If specified, then Slowness-Azimuth Stations Corrections (SASCs) are applied to azimuth and slowness predictions. Specify the path to the directory containing SASC files.
For more information see Bondar, I. (1998). Teleseismic slowness-azimuth station corrections (SASC) for the IMS network. CCB-PRO-98/01. Memo available at https://swp.ctbto.org/web/swp/manuals.

C.2.2. *lookup2d*

Predictor lookup2d defaults to ak135, which is built into LocOO3D. If the user wants to specify other traveltime tables, the following properties may be used.

C.2.2.1. *lookup2dModel*

<string> [Default = ak135]

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

C.2.2.2. *lookup2dTableDirectory*

<string> [Default = /jar/ak135.zip]

Path to the directory or zip file containing the travel time lookup tables for use with the Lookup2D predictor. Within the directory or zip file, files can have names like 'PKP' or 'ak135.PKP', etc.

Zip files containing travel time lookup tables for models ak135 and iasp91 are delivered inside the Salsa3DSoftware jar file and can be requested by specifying /jar/ak135.zip or /jar/iasp91.zip.

On Linux systems it is safe to use uncompressed directories containing travel time lookup tables but on Mac and Windows machines uncompressed directories can lead to errors because file names on these systems are case insensitive. This means that files for phases pP and PP will collide and one of the files will be ignored. The same happens for phases sS and SS; pS and PS; and sP and SP, as well as possibly other phases. To avoid these problems, users on Mac and Windows systems should use models stored in zip files instead of uncompressed directories.

It is possible, on both Windows and Macs, to create a directory that can contain files with case sensitive names but special procedures are required. If a user unzips a file in one of these special directories there will be no case sensitivity collisions.

On Windows, see
https://learn.microsoft.com/en-us/windows/wsl/case-sensitivity

On Mac, see
https://docs.spryker.com/docs/dg/dev/integrate-and-configure/switch-to-a-case-sensitive-file-system-on-mac-os.html#create-the-disk-image

C.2.2.3. *lookup2dEllipticityCorrectionsDirectory*

<string> [Default = /jar/ellipticity_corrections.zip]

Path to the directory or zip file where ellipticity correction coefficients are located for use with the Lookup2D predictor. A zip file containing ellipticity corrections for many standard phases is delivered inside the Salsa3DSoftware jar file and can be requested by specifying /jar/ellipticity_corrections.zip.

C.2.2.4. *lookup2dUseEllipticityCorrections*

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

A flag that can be used to turn off the use of ellipticity corrections.

C.2.2.5. *lookup2dUseElevationCorrections*

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

A flag that can be used to turn off the use of elevation corrections.

C.2.2.6. *lookup2dSedimentaryVelocityP*

<double> [Default = 5.8 km/sec]

The P-wave seismic velocity to be used in the calculation of elevation corrections.

C.2.2.7. *lookup2dSedimentaryVelocityS*

<double> [Default = 3.4 km/sec]

The S-wave seismic velocity to be used in the calculation of elevation corrections.

C.2.2.8. *lookup2dTTUncertaintyType*

<string> [Default = `DistanceDependent`] (`DistanceDependent`)

When a LibCorr3D correction surface is available for a particular station-phase-attribute, the attribute uncertainty will be retrieved from the LibCorr3D correction surface. Otherwise attribute uncertainty will be computed from information in seismicBaseData.

C.2.2.9. *lookup2dTTModelUncertaintyScale*

<double> [Default = 1.0]

Travel time model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.2.10. *lookup2dTTModelUncertaintyOffset*

<double> [Default = 0.0]

Travel time model uncertainty offset in seconds. Value is added the standard model uncertainty.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.2.11. *lookup2dAZModelUncertaintyScale*

<double> [Default = 1.0]

Azimuth model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.2.12. *lookup2dAZModelUncertaintyOffset*

<double> [Default = 0.0]

Azimuth model uncertainty offset in degrees. Value is added the standard model uncertainty.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.2.13. *lookup2dSHModelUncertaintyScale*

<double> [Default = 1.0]

Slowness model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.2.2.14. *lookup2dSHModelUncertaintyOffset*

<double> [Default = 0.0]

Slowness model uncertainty offset in seconds/degree. Value is added the standard model uncertainty.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.2.3. *Bender*

C.2.3.1. *benderModel*

<string> [Default = none]

Path to GeoTess-format 3D velocity model that Bender uses to calculate predictions of seismic observables.

If `benderModel` points to a salsa3d directory and the directory contains a file called 'prediction_model.geotess', then that model is used.

C.2.3.2. *benderAllowMOHODiffraction*

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

Mark phases that diffract off the Moho as valid observations. Setting to false will disallow these phases from the location computation.

C.2.3.3. *benderAllowCMBDiffraction*

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

Mark phases that diffract off the core-mantle boundary as valid observations. Setting to false will disallow these phases from the location computation.

C.2.3.4. *benderPhaseInterfaceToModelInterfaceRemap*

<2 strings: requested-interface and model-interface> [Default = none]

Allows remapping of one model interface to another to satisfy phase-specific path requirements defined in seismicBaseData. Input is name of interface in phase definition followed by name of interface based on layers present in velocity model.

C.2.3.5. *benderTTUncertaintyType*

<string> [Default = `DistanceDependent`] (`DistanceDependent | SourceDependent`)

Type of travel time uncertainty desired,

C.2.3.6. *benderTTUncertaintyDirectory*

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

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 /index/SNL_tool_Root/seismicBaseData/tt/ak135 then specify:

```
benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135
```

If `benderUncertaintyDirectory` points to a salsa3d directory that contains a subdirectory called distance_dependent_uncertainty, then the uncertainty information stored in that directory will be used. Property `benderUncertaintyModel` (Appendix C.2.3.7) will be ignored.

If no directory is specified, the path to the internal seismic-base-data.jar included with the LocOO3D jar is used.

C.2.3.7. *benderTTUncertaintyModel*

<string> [Default = ak135]

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

```
<benderUncertaintyDirectory>/<attribute>/<benderUncertaintyModel>
```

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

```
benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135
```

If no model is specified, the internal ak135 model included in the LocOO3D jar is used.

C.2.3.8. *benderUseTTSiteCorrections*

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

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 GeoTess model specified with property `benderModel`.

C.2.3.9. *benderTTModelUncertaintyScale*

<double> [Default = 1.0]

Travel time model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.3.10. *benderTTModelUncertaintyOffset*

<double> [Default = 0.0]

Travel time model uncertainty offset in seconds. Value is added the standard model uncertainty.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.3.11. *benderAZModelUncertaintyScale*

<double> [Default = 1.0]

Azimuth model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.3.12. *benderAZModelUncertaintyOffset*

<double> [Default = 0.0]

Azimuth model uncertainty offset in degrees. Value is added the standard model uncertainty.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.3.13. *benderSHModelUncertaintyScale*

<double> [Default = 1.0]

Slowness model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.2.3.14. *benderSHModelUncertaintyOffset*

<double> [Default = 0.0]

Slowness model uncertainty offset in seconds/degree. Value is added the standard model uncertainty.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.2.4. *RSTT*

RSTT is the predictor for determining travel times from the RSTT model.

C.2.4.1. *rsttModel*

<string> [Default = none]

Path to RSTT model to use for predictions of seismic observables.

C.2.4.2. *rsttTTUncertaintyType*

<string> [Default = null] (`RSTT_PATH_DEPENDENT | RSTT_DISTANCE_DEPENDENT`)

When a LibCorr3D correction surface is available for a particular station-phase, the travel time uncertainty will be retrieved from the LibCorr3D correction surface. When a LibCorr3D correction surface is *not* available for a particular station-phase, then travel time uncertainty computed by RSTT will be returned.

When `rsttUncertaintyType` is *not* one of the hierarchical types then either path_dependent or distance_dependent travel time uncertainty computed by RSTT will be returned, regardless of whether a LibCorr3D correction surface is available.

C.2.4.3. *rstt_ch_max*

<double> [Default = 0.2]

Specify the maximum value of c*h where c is the rstt Zhao c property and h is the turning depth of the ray below the Moho. See RSTT documentation for details.

C.2.4.4. *rstt_max_depth*

<double> [Default = 200]

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

C.2.4.5. *rstt_max_distance*

<double> [Default = 15]

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

C.2.4.6. *rstt_path_increment*

<double> [Default = 0.1]

The path increment in degrees used when computing the path integral from source to receiver.

C.2.4.7. *rstt_backstop_lookup2d*

<boolean> [Default = false]

If `rstt_backstop_lookup2d` is true, and rstt fails to compute a valid prediction for any reason, then a prediction computed by the lookup2d predictor is returned. All lookup2d properties such as `lookup2dModel` (Appendix C.2.2.1) and `lookup2dPathCorrectionsType` (Appendix **Error! Reference s ource not found.**) are in effect.

C.2.4.8. *rsttTTModelUncertaintyScale*

<double> [Default = 1.0]

Travel time model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.4.9. *rsttTTModelUncertaintyOffset*

<double> [Default = 0.0]

Travel time model uncertainty offset in seconds. Value is added the standard model uncertainty.

```
ttModelUncertainty = ttModelUncertainty * scale + offset
```

C.2.4.10. *rsttdAZModelUncertaintyScale*

<double> [Default = 1.0]

Azimuth model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.4.11. *rsttAZModelUncertaintyOffset*

<double> [Default = 0.0]

Azimuth model uncertainty offset in degrees. Value is added the standard model uncertainty.

```
azModelUncertainty = azModelUncertainty * scale + offset
```

C.2.4.12. *rsttSHModelUncertaintyScale*

<double> [Default = 1.0]

Slowness model uncertainty scale. Standard model uncertainty value is multiplied by this value.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.2.4.13. *rsttSHModelUncertaintyOffset*

<double> [Default = 0.0]

Slowness model uncertainty offset in seconds/degree. Value is added the standard model uncertainty.

```
shModelUncertainty = shModelUncertainty * scale + offset
```

C.3. LibCorr3D

Option to apply LibCorr3D corrections to predicted travel times, azimuth or slowness. LibCorr3D corrections can be applied to any Predictor, including lookup2d, bender or rstt. Corrections are defined in a separate GeoTess-formatted file for each station/phase pair.

In the property descriptions below, `<predictor>` can be either `lookup2d`, `bender` or `rstt`.

C.3.1. *<predictor>PathCorrectionsType*

<string> [Default = none] (`libcorr`)

Set the value to `libcorr` to apply LibCorr3D corrections. If this property is omitted, then corrections will not be applied.

C.3.2. *<predictor>LibCorrPathCorrectionsRoot*

<string> [Default = none]

The name of the directory where the LibCorr3D correction surfaces reside. This directory should contain a separate file for each correction surface.

C.3.3. *<predictor>LibCorrPathCorrectionsRelativeGridPath*

<string> [Default = "."]

The relative path from the directory where the correction surface files reside to the directory where the grid files reside. The default value of '.' specifies that either (1) the GeoTessGrid is stored in the

GeoTessModel file, or (2) the GeoTessGrid is stored in a separate file in the same directory as the GeoTessModel.

C.3.4. *<predictor>LibCorrInterpolatorTypeHorizontal*

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

Type of horizontal interpolation to use.

C.3.5. *<predictor>LibCorrInterpolatorTypeRadial*

<string> [Default = "linear"] (`linear | cubic spline`)

Type of radial interpolation to use. Use of cubic spline interpolation is strongly discouraged.

C.3.6. *<predictor>LibCorrPreloadModels*

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

If true, load all LibCorr3D models at startup. If false, load LibCorr3D models on an 'as needed' basis.

C.3.7. *<predictor>LibCorrMaxSiteSeparation*

<double> [Default = 10 km]

When separation of user Site and model Site is less than `LibcorrMaxSiteSeparation`, corrections will be applied to data from user Site. Units are km.

C.3.8. *<predictor>LibCorrMatchOnRefsta*

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

When true and a user Site and model Site have the same refsta, corrections will be applied to data from user Site regardless of the separation of the two Sites. Default is false.

C.3.9. *<predictor>UsePathCorrectionsInDerivatives*

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

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

C.3.10. *use_tt_path_corrections*

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

Flag to turn off the use of travel time path corrections.

C.3.11. *use_az_path_corrections*

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

Flag to turn off the use of azimuth path corrections.

C.3.12. *use_sh_path_corrections*

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

Flag to turn off the use of slowness path corrections.

C.3.13. *use_tt_model_uncertainty*

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

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.3.14. *use_az_model_uncertainty*

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

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.3.15. *use_sh_model_uncertainty*

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

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.4. Master Event Relative Relocation

If property `masterEventWhereClause` is specified, then LocOO3D will apply master event relocation corrections to all input origins that it relocates. A single masterEvent origin and the 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` (Appendix C.2.1.1). These residuals will become masterEventCorrections, which will be added to predictions of observations that have the same station-phase-type as the masterEvent assocs. Note that corrections to travel time, azimuth, and slowness will be applied if the corresponding master event assoc is time-defining, azimuth-defining and/or slowness-defining.

C.4.1. *masterEventWhereClause*

<string> [Default = null] (valid sql; e.g. '`orid = 100`')

If this property is present, then the master event relocation option will be implemented. Must provide a valid sql "where" clause that will be executed against the `masterEventSchema` (Appendix C.4.4) database schema. The "where" clause (Appendix C.14.14) must return only one origin row. If the clause returns anything other than one origin row an exception will be thrown.

C.4.2. *masterEventAssocClause*

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

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

C.4.3. *masterEventUseOnlyStationsWithCorrections*

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

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

C.4.4. *masterEventSchema*

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

If this property is specified, then all the properties of a valid schema must be supplied (see description of dbInput properties in Appendix C.14). 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 the specified database tables.

If this property 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.5. General

C.5.1. *maxProcessors*

<int> [Default = all available processors]

Limits the number of processors that LocOO3D will use to compute predictions in concurrent mode to the user-specified number.

C.5.2. *parallelMode*

<string> [Default = sequential]

Sets whether prediction calculations are done concurrently on multiple threads or sequentially. To perform

C.5.3. *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 = Use the initial location given in the origin file/table.

properties_file = Use values given in the *.properties file. Requires all the following properties to be defined in the properties file: 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.5.4. *gen_lat_init*

<double> [Default = Globals.NA_Value] (-90.00 <= X <= 90.000)

Initial latitude estimate for the algorithm (degrees).

C.5.5. *gen_lon_init*

<double> [Default = Globals.NA_Value] (-180.0 <= X <= 360.00)

Initial longitude estimate for the algorithm (degrees).

C.5.6. *gen_depth_init*

<double> [Default = Globals.NA_Value] (-10000 <= X <= 10000.)

Initial depth estimate for the algorithm (km).

C.5.7. *gen_origin_time_init*

<double> [Default = Globals.NA_Value] (epoch time)

Initial origin time estimate for the algorithm, in seconds since 1970-01-01.

C.5.8. *gen_fix_lat_lon*

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

If true, hold latitude and longitude fixed to their initial values during the solution algorithm. See also `gen_initial_location_method`, `gen_lat_init`, and `gen_lon_init` (Appendices C.5.3, C.5.4, C.5.5, respectively).

C.5.9. *gen_fix_origin_time*

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

If true, hold origin time fixed to the initial value during the solution algorithm. See also `gen_initial_location_method`, `gen_origin_time_init` (Appendices C.5.3 and C.5.7, respectively).

C.6. Depth Constraints

Options for constraining origin depth based on fixed values, allowable range, and uncertainty.

C.6.1. *gen_fix_depth*

<string> [Default = false] (true | false | <double> | '`topography`' | '`origin.dtype`')

If gen_fix_depth is true, then fixed depth solutions are computed with depth fixed to the value of the depth of the input origin(s).

If gen_fix_depth is false, then free depth solutions are computed. Depth may be constrained to fall within a certain range with properties gen_min_depth, gen_max_depth or seismicity_depth_model (Appendices C.6.3, C.6.4, C.6.2, respectively).

If value is a floating point number, then fixed depth solutions are computed with depth held fixed at the specified value (in km).

If value is topography or topo, then fixed depth solutions are computed with depth held fixed at the topographic/bathymetric surface as specified in seismicity_depth_model (Appendix C.6.2). In this case, property seismicity_depth_model is required.

If value is origin.dtype, then whether to compute a fixed or free depth solution is determined individually for each input origin based on the *origin.dtype* field. If *origin.dtype* is 'f' then a free depth solution is computed otherwise a fixed depth solution is computed. For more control, gen_fix_depth can be set to origin.dtype = <list of strings> or origin.dtype != <list of strings>. For example, for a dataset the only possible values of origin.dtype are a,d,f,g, then origin.dtype != f and origin.dtype = a,d,g both have the same meaning. Free depth solution will only be calculated when origin.dtype = f.

C.6.2. *seismicity_depth_model*

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

If the string default is specified then the internal, default seismicity depth model is loaded. If a valid file name is specified, the model is loaded from the specified file.

This property specifies a GeoTess model containing attributes that specify the allowable depth range of free depth solutions as a function of geographic location. The upper limit is usually defined as topography/bathymetry of the Earth. The lower limit is based on maximum depth of historic seismicity and tectonics. See Section 2.3 for details.

If fixed depth solutions are requested by specifying gen_fix_depth = topography (Appendix C.6.1), then seismicity_depth_model is required, and depth is fixed to the first attribute in the specified model.

If free depth solutions are requested (gen_fix_depth = false) and seismicity_depth_model is not specified, then the allowable depth range is determined by properties gen_min_depth (Appendix C.6.3) and gen_max_depth (Appendix C.6.4).

If free depth solutions are requested (gen_fix_depth = false) and a solution is computed whose depth is outside the allowable range, the free depth solution is replaced with a fixed depth solution with depth fixed at the minimum or maximum limit, as appropriate.

For additional information, see also properties depth_constraint_uncertainy_scale (Appendix C.6.5) and depth_constraint_uncertainy_offset (Appendix C.6.6).

C.6.3. *gen_min_depth*

<double> [Default = 0.] (-10000 <= X <= 10000.)

Minimum depth constraint (km). Only considered when a free depth solution is requested. Ignored if `seismicity_depth_model` (Appendix C.6.2) is specified.

C.6.4. *gen_max_depth*

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

Maximum depth constraint (km). Only considered when a free depth solution is requested. Ignored if `seismicity_depth_model` (Appendix C.6.2) is specified.

C.6.5. *depth_constraint_uncertainy_scale*

<double> [Default = 0.] (any valid floating point value)

Ignored for fixed depth solutions. For free depth solutions, the uncertainty of the event depth is used to determine a tolerance for whether a computed depth is within the allowable range defined by properties `gen_min_depth, gen_max_depth`, or `seismicity_depth_model` (Appendices C.6.3, C.6.4, C.6.2, respectively).

The modified depth with uncertainty is determined by:

```
origin.depth + (origerr.sdepth * depth_constraint_uncertainty_scale +
      depth_uncertainty_offset) < min_depth
```

or

```
origin.depth - (origerr.sdepth * depth_constraint_uncertainty_scale +
       depth_uncertainty_offset) > max_depth
```

If the modified value is outside the allowable range, then the free depth solution is considered invalid and is replaced with a fixed depth solution computed with depth fixed to either `min_depth` or `max_depth`, as appropriate.

C.6.6. *depth_constraint_uncertainy_offset*

<double> [Default = 0.] (any valid floating point value)

Depth uncertainty offset in km. See property `depth_constraint_uncertainy_scale` (Appendix C.6.5).

This property is ignored when a fixed depth solution is requested (`gen_fix_depth = true`, Appendix C.6.1).

C.7. Residuals

Parameters to define residual allowance. 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` (Appendix C.7.2) then a 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 minimum number that will be set to non-defining is 1. The maximum number that can be set to non-defining is determined by (N-M) * `gen_big_residual_max_fraction` (Appendix C.7.3).

C.7.1. *gen_allow_big_residuals*

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

If `true`, allow observations that result in 'big' residual values.

C.7.2. *gen_big_residual_threshold*

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

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

C.7.3. *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` (Appendix C.7.1) is false and there are observations with large residuals.

C.7.4. *nObservationFlipFlops*

<int> [Default = 10]

There are instances when LocOO3D will drop and then later reintroduce observations (see `gen_allow_big_residuals`, Appendix C.7.1). It is theoretically possible for LocOO3D to go into an infinite loop where it repeatedly drops then reintroduces one or more observations. `nObservationFlipFlops` specifies the maximum number of times that this can happen before it adopts one of the states and moves on.

C.7.5. *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.7.6. *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 and down-selects from the set of defining stations. All observations from stations that are not included in the list are set to non-defining.

C.7.7. *gen_defining_attributes*

<string> [Default = all] (`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 and down-selects from the set of defining attributes. All observations with attributes that are 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.7.8. *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 several individual filters, separated by commas. Each observation starts out as either defining or non-defining (after application of properties `gen_defining_stations`, `gen_defining_phases` and `gen_defining_attributes`; Appendices C.7.6, C.7.5, C.7.7, respectively). 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.7.9. *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 property is set to the value of this property.

C.7.10. *gen_jordan_sverdrup_K*

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

Jordan-Sverdrup K property for 'mixed' type confidence ellipses. -1 is interpreted as infinity. This property is ignored unless `gen_error_ellipse_type = mixed` (Appendix C.7.9).

C.7.11. *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 property only applies when K > 0.

C.7.12. *gen_confidence_level*

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

Uncertainty confidence level desired.

C.7.13. *allowCorePhaseRenamingP*

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

If `true,` then core phase renaming will occur at the distance specified by `corePhaseRenamingThresholdDistanceP` (Appendix C.7.14).

C.7.14. *corePhaseRenamingThresholdDistanceP*

<double> [Default = 110.] (0.0000 <= X <= 180.0000)

If `allowCorePhaseRenamingP = true` (Appendix C.7.13) then this is the distance in degrees beyond which core phase renaming will occur.

C.7.15. *useSimplex*

<boolean> [Default = false] (true | 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.8. Correlated Observation Parameters

C.8.1. *gen_correlation_matrix_method*

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

Specifies how correlation coefficients are to be specified.

`file`	Correlation coefficients between pairs of observations are read in from a file, with the file name being specified with the `gen_correlation_matrix_file` (Appendix C.8.2) property.
`function`	Function properties must be specified using the `gen_correlation_scale` property (Appendix C.8.3).

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.8.2. *gen_correlation_matrix_file*

<string> [Default =] (any valid file path + file 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.8.3. *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` (Appendix C.8.1) 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` property (Appendix C.8.3).

C.9. Levenberg-Marquardt Non-Linear Least Squares Solver

C.9.1. *lsq_convergence_n*

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

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

C.9.2. *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`, Appendix C.9.4).

C.9.3. *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.9.4. *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` (Appendix C.9.4) it can be concluded that the sum squared weighted residuals cannot be further reduced and convergence can be declared.

C.9.5. *lsq_damping_factor*

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

Damping factor to be applied to singular values. This factor controls application of the Levenberg-Marquardt algorithm, which helps the locator converge if it is oscillating around the optimum location.

For AUTOMATIC DAMPING: `lsq_damping_factor = -1` [DEFAULT].

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.9.6. *lsq_initial_applied_damping*

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

When `lsq_damping_factor = -1` (automatic damping, see Appendix C.9.5), this is the initial damping factor applied when an increase is observed in the sum squared weighted residuals.

C.9.7. *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 property will not change from its initial value.

C.10. Gridded Residuals

C.10.1. *grid_output_file_name*

<string> [Default = none]

Name of file to receive the gridded residuals. If this property is not specified then gridded residuals are not generated and all the rest of the properties in this section are ignored.

C.10.2. *grid_output_file_format*

<string> [Default = tecplot] (`tecplot`)

Output file format. The only currently supported format for the output text file is tecplot. Other formats may be supported in the future.

When NZ = 1 there will be 4 values per record:
X (either longitude in degrees, or East in km),
Y (either latitude in degrees or North in km),
R root mean squared weighted residual, unitless
C confidence level (useful for plotting a 95% contour line).
When NZ > 1 there will be 5 values per record:
X (either longitude in degrees, or East in km),
Y (either latitude in degrees or North in km),
Z depth in km,
R root mean squared weighted residual, unitless
C confidence level, in % (useful for plotting a 95% contour line).

C.10.3. *grid_origin_source*

<string> [Default = epicenter] (`epicenter | hypocenter | other`)

The center of the grid. If anything other than `epicenter` or `hypocenter` is specified then the center of the grid is determined by properties `grid_origin_lat, grid_origin_lon`, and `grid_origin_depth` (Appendices C.10.4, C.10.5, C.10.6, respectively).

C.10.4. *grid_origin_lat*

<double> [Default = none]

Latitude in degrees of the center of the grid. Ignored if `grid_origin_source` (Appendix C.10.3) equals either `epicenter` or `hypocenter`.

C.10.5. *grid_origin_lon*

<double> [Default = none]

Longitude in degrees of the center of the grid. Ignored if `grid_origin_source` (Appendix C.10.3) equals either `epicenter` or `hypocenter`.

C.10.6. *grid_origin_depth*

<double> [Default = 0. km]

Depth in km of the center of the grid. Ignored if `grid_origin_source` (Appendix C.10.3) equals either `epicenter` or `hypocenter`.

C.10.7. *grid_map_units*

<string> [Default = degrees] (`degrees | km`)

Map width and height will be interpreted with these units. Also controls the units of the output.

C.10.8. *grid_map_width*

<double> [Default = none]

Map will be this many units wide, in units specified by `grid_map_units` (Appendix C.10.7). The origin of the map will be in the center.

C.10.9. *grid_map_height*

<double> [Default = none]

Map will be this many units high, in units specified by `grid_map_units` (Appendix C.10.7). The origin of the map will be in the center.

C.10.10. *grid_map_depth_range*

<double> [Default = 0.]

Map will be this many units deep, in km. The origin of the map will be in the center.

C.10.11. *grid_map_nwidth*

<int> [Default = none]

Map will have this many nodes in the x direction.

C.10.12. *grid_map_nheight*

<int> [Default = none]

Map will have this many nodes in the y direction.

C.10.13. *grid_map_ndepth*

<int> [Default = 1]

Map will have this many nodes in the z direction.

C.11. General Input/Output Parameters

calculations concurrently, set parallelMode = concurrent.

C.11.1. *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.11.2. *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.11.3. *io_print_to_screen*

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

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.11.4. *io_error_file*

<string> [Default = locoo_errors.txt]

Full path to general output error file. All error messages generated during LocOO3D execution are sent to this file.

C.11.5. *io_print_errors_to_screen*

<boolean> [Default = io_verbosity > 0]

Write error messages to the screen.

C.11.6. *io_max_obs_tables*

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

Maximum number of observation and prediction tables to output when `io_verbosity >= 4` (Appendix 0).

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 property is ignored if io_verbosity < 4.

C.11.7. *io_observation_sort_order*

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

Specifies the order in which observations are reported in the observation and prediction tables in the text output file.

C.11.8. *io_iteration_table*

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

Whether or not to print the iteration table when `io_verbosity >= 4` (Appendix 0).

C.11.9. *io_nondefining_residuals*

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

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. This property does not impact the final computed location.

C.12. DataLoader Utility

C.12.1. *dataLoaderType*

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

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

C.12.2. *dataLoaderInputType*

<string> [Default = none] (`file | database | oracle | application`)

String indicating the input type to be used by LocOO3D. If set to file, the input will consist of text file(s) containing input data in CSS3.0 schema tables. Types oracle and database are equivalent and indicate that input will come from a database. For descriptions of properties related to file IO see Section C.13. For descriptions of properties related to database IO see Section C.14.

C.12.3. *dataLoaderOutputType*

<string> [Default = none] (`file | database | oracle | application`)

String indicating the output type to be produced by LocOO3D. If set to file, the output will consist of text file(s) containing data in CSS3.0 schema tables. Types oracle and database are equivalent and indicate that output will be written to a database. For descriptions of properties related to file IO see Section C.13. For descriptions of properties related to database IO see Section C.14.

C.13. DataFileLoader Utility

C.13.1. *dataLoaderFileOrigins*

<string> [Default = none]

Name of file containing the input origins. Required if `dataLoaderInputType = file` (Appendix C.12.2).

C.13.2. *dataLoaderFileAssocs*

<string> [Default = none]

Name of file containing the input assocs. Required if `dataLoaderInputType = file` (Appendix C.12.2).

C.13.3. *dataLoaderFileArrivals*

<string> [Default = none]

Name of file containing the input arrivals. Required if `dataLoaderInputType = file` (Appendix C.12.2).

C.13.4. *dataLoaderFileSites*

<string> [Default = none]

Name of file containing the input sites. Required if `dataLoaderInputType = file` (Appendix C.12.2).

C.13.5. *dataLoaderFileOrids*

<string> [Default = none]

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

C.13.6. *dataLoaderFileOutputOrigins*

<string> [Default = none]

Name of file to receive the output origins. Note that orids and evids in the output origin rows will equal orids and evids in the input tables. In other words, orids and evids are not changed by LocOO3D when reading and writing from flat files.

C.13.7. *dataLoaderFileOutputOrigerrs*

<string> [Default = none]

Name of file to receive the output origerrs.

C.13.8. *dataLoaderFileOutputAssocs*

<string> [Default = none]

Name of file to receive the output assocs.

C.13.9. *dataLoaderFileOutputAzgaps*

<string> [Default = none]

Name of file to receive the output azgaps.

C.13.10. *dataLoaderFileInputTokenDelimiter*

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

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 is beneficial since the input `dataLoaderFileInputTokenDelimiter` is read from a properties file.

C.13.11. *dataLoaderFileOutputTokenDelimiter*

<string> [Default = same value as `dataLoaderFileInputTokenDelimiter`]

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 is beneficial since the input `tokenDelimiter` is read from a properties file.

C.14. DBIO Utility

C.14.1. *Default database connection properties*

Default database connection properties can be included in a file called *database.properties* that resides in the user's root directory (e.g. */Users/username/database.properties*). The file should specify the following properties (users should specify appropriate values to the right of the '=' signs):

```
DB_INSTANCE =
jdbc:oracle:thin:@database_instance.sandia.gov:port_number:database_instance
DB_WALLET = /Users/$USER/wallet
DB_DRIVER = oracle.jdbc.driver.OracleDriver
DB_USERNAME = USER_NAME
DB_PASSWORD_ACCOUNT_1 = ********
DB_PASSWORD_ACCOUNT_2 = ********
DB_PASSWORD_ACCOUNT_3 = ********
```

Alternatively, those same default properties can be specified in the user's environment, which is usually accomplished by specifying them in the user's ~/*.bash_profile file* or *.cshrc* file.

Database connection properties specified in an application properties file, if present, will override the default values. These properties are:

- `dbInputInstance / dbOutputInstance`
- `dbInputWallet / dbOutputWallet`
- `dbInputUserName / dbOutputUserName`
- `dbInputPassword / dbOutputPassword`
- `dbInputDriver / dbOutputDriver`

If both *wallet* and *instance* property types are specified in the properties file, an exception is thrown.

If a *driver* is not specified by any of the methods specified above, a default value of *oracle.jdbc.driver.OracleDriver* is used.

C.14.2. *Database Instances and Wallets*

Databases can be connected to via wallets or instances. If using wallets, there are two ways that a wallet can be used depending on the setup of the user's database environment. Read through the following bullets to establish which wallet setup is needed:

- Provided the following are true, specify `dbInputUserName/dbOutputUserName` (Appendix C.14.3) in the properties file, e.g., "`dbInputUserName = global_ro@archive.susndc`".
    - The location of your wallet is correctly specified in your user environment with environment variable TNS_ADMIN.
    - `dbInputInstance/dbOutputInstance` (Appendix C.14.5) is not specified in your properties file.
    - You do not have a *database.properties* file in your root directory, or, if you do, DB_INSTANCE is not specified in it.

- DB_INSTANCE is not specified in your user's environment (check your ~/*.bash_profile* or *.cshrc* file).

- Otherwise, use one of the following methods to specify the location of your wallet:
  1. Specify `dbInputWallet/dbOutputWallet` (Appendix C.14.6) in your properties file.
  2. Specify variable DB_WALLET in your *database.properties* file in your root directory.
  3. Specify environment variable DB_WALLET or TNS_ADMIN
  - Also, specify `dbInputUserName/dbOutputUserName` in your properties file. It is likely necessary to prepend the string '*jdbc:oracle:thin:/@*' to the beginning of your usernames, e.g., "*dbInputUserName = jdbc:oracle:thin:/@account_name*".
  - Do not specify `dbInputIntance/dbOutputInstance` in your properties file.

If using an instance, provide the following properties:

- `dbInputUserName/dbOutputUserName` (Appendix C.14.3)
- `dbInputInstance/dbOutputInstance` (Appendix C.14.5)
- `dbInputPassword/dbOutputPassword` (Appendix C.14.4)C.14.3

Database applications need to discover either a database *instance* or a *wallet* location, but not both. The following procedure is followed to discover a database *instance* or *wallet*.

1. Check the properties file for properties `dbInputInstance/dbOutputInstance` and `dbInputWallet/dbOutputWallet`.
2. If both are found, throw an exception.
3. If neither is found:
   a. Look for DB_INSTANCE in the *database.properties* file.
   b. If *instance* not found, look for DB_INSTANCE in the user's environment.
   c. If *instance* not found, look for DB_WALLET in the *database.properties* file.
   d. If *wallet* not found, look for DB_WALLET in the user's environment.
   e. If *wallet* not found, look for TNS_ADMIN in the user's environment (this environment variable is used at AFTAC to store the user's wallet location.)
4. If neither *instance* nor *wallet* is found, throw an exception.

C.14.3. *dbInputUserName, dbOutputUserName*

<string> [Default = DB_USERNAME in environment or *database.properties* file]

Database input/output account username.

C.14.4. *dbInputPassword, dbOutputPassword*

<string> [Default = DB_PASSWORD_<DB_USERNAME> in environment or *database.properties* file]

Database input/output account passwords.

C.14.5. *dbInputInstance, dbOutputInstance*

<string> [Default = DB_INSTANCE in environment or *database.properties* file]

Database instance for input/output.

C.14.6. *dbInputWallet, dbOutputWallet*

<string> [Default = DB_WALLET or TNS_ADMIN in environment or *database.properties* file]

The location of the database wallet for input/output.

C.14.7. *dbInputDriver, dbOutputDriver*

<string> [Default = *oracle.jdbc.driver.OracleDriver.*]

Database driver for input/output.

C.14.8. *dbInputTableTypes*

<string> [Default = none]

If the `dbInputTableTypes` property (Appendix C.14.8) is specified, then the input table types specified with this property will default to the value of the `dbInputTablePrefix` (Appendix C.14.9) property with the appropriate table type appended on the end. Currently recognized table types include **origin, assoc, arrival, site.**

C.14.9. *dbInputTablePrefix*

<string> [Default = none]

If this property is specified then the four input tables (`dbInputOriginTable, dbInputAssocTable, dbInputArrivalTable, dbInputSiteTable`; Appendices C.14.10, C.14.11, C.14.12, C.14.13, respectively) will default to the value of this property with the appropriate table type (ORIGIN, ASSOC, ARRIVAL, SITE) appended on the end. If any of the four tables are also explicitly specified, then the explicitly specified name has precedence.

C.14.10. *dbInputOriginTable*

<string> [Default not allowed]

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

C.14.11. *dbInputAssocTable*

<string> [Default not allowed]

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

C.14.12. *dbInputArrivalTable*

<string> [Default not allowed]

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

C.14.13. *dbInputSiteTable*

<string> [Default not allowed]

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

C.14.14. *dbInputWhereClause*

<string> [Default = 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 to load data from the database:

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

With the output from this query, LocOO3D sorts the orids into batches, where each batch will have roughly the same number of defining phases and the batches are ordered to have an increasing number of orids. The number of defining phases per batch can be specified with the property `batchSizeNdef` (Appendix C.14.16), 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 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 two SQL statements, the second of which applies the input dbInputWhereClause:

1.
```
select orid, evid, lat, lon, depth, time from <dbOriginTable>
  where orid in (<list of orids in batch>)
```

2.
```
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 <dbInputWhereClause>
```

C.14.15. *dbInputAssocClause*

<string> [Default = 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` (Appendix C.14.14) for details of how it is used to create a SQL query to extract desired data.

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 dbInputAssocClause becomes part of a SQL statement that is executed against an assoc, arrival, and site table.

C.14.16. *batchSizeNdef*

<int> [Default = 1000]

The approximate number of defining phases that will be included in each batch of origins that will be loaded and processed. See `dbInputWhereClause` (Appendix C.14.14) or more details.

C.14.17. *dbOutputTablePrefix*

<string> [Default = none]

If this property is specified, then the output table types specified with the `dbOutputTableTypes` (Appendix C.14.18) property will default to the value of this property with the appropriate table type appended to the end.

C.14.18. *dbOutputTableTypes*

<string> [Default = none]

If the `dbOutputTableTypes` property is specified, then the output table types specified with this property will default to the value of the `dbOutputTablePrefix` (Appendix C.14.17) property with the appropriate table type appended on the end. Currently recognized table types include: **origin**, **assoc**, **arrival**, **site**, **origerr,** and **azgap**.

C.14.19. *dbOutputOriginTable*

<string> [Default = none]

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

The value of orid in output origin rows depends on the value of `dbOutputConstantOrid` (Appendix C.14.25). If `dbOutputConstantOrid` is true, then orids in output origin rows will be equal to orids in the input origin rows. If `dbOutputConstantOrid` is false (default) then at the beginning of the LocOO3D run the output origin table is queried for the maximum value of orid before any new origin rows are output to it. This maxOrid value is incremented by one for every output origin row.

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

C.14.20. *dbOutputArrivalTable*

<string> [Default = none]

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

C.14.21. *dbOutputAssocTable*

<string> [Default = none]

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

C.14.22. *dbOutputAzgapTable*

<string> [Default = none]

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

C.14.23. *dbOutputOrigerrTable*

<string> [Default = none]

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

C.14.24. *dbOutputAuthor*

<string> [Default = '-']

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

C.14.25. *dbOutputConstantOrid*

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

If `dbOutputConstantOrid` is true, then orids in output origin rows will be equal to orids in the input origin rows. If `dbOutputConstantOrid` is false (default) then at the beginning of the LocOO3D run the output origin table is queried for the maximum value of orid before any new origin rows are output to it. This maxOrid value is incremented by one for every output origin row.

C.14.26. *dbOutputAutoTableCreation*

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

Set to true if output database tables should be created when they do not already exist.

C.14.27. *dbOutputTruncateTables*

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

Set to true if output database tables should be automatically truncated at the start of the run. Unless the `dbOutputPromptBeforeTruncate` property (Appendix C.14.28) has been set to false, the user will be prompted before table truncation occurs.

C.14.28. *dbOutputPromptBeforeTruncate*

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

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

APPENDIX D. OUTPUT VALUES FOR LOCOO3D

D.1. Iteration Table

Iteration Table:

Itt	It	Comment	N	M	Lat	Lon	Depth	Time	rms_Trsd	rms_Wrsd	dNorth	dEast
1	1	start	6	3	79.7625	2.4637	0.000	0.000	1.4131	0.7525	-23.683	4.266
2	2	start	6	3	79.5503	2.6743	0.000	-0.599	0.8261	0.4848	-2.672	0.688
3	3	start	6	3	79.5264	2.7082	0.000	-0.592	0.8161	0.4834	-0.012	0.023
4	4	start	6	3	79.5263	2.7093	0.000	-0.591	0.8160	0.4834	-0.000	0.000
4	4	damped	6	3	79.5263	2.7093	0.000	-0.591	0.8160	0.4834	0.000	0.000

Iteration Table

Itt	It	dZ	dT	dkm	dxStart	dzStart	dtStart	azStart	nF	damp	converge
1	1	0.000	-0.599	24.5369	24.0641	0.0000	-0.5992	169.7887	1	-4	0.00e+00
2	2	0.000	0.007	2.7594	26.8154	0.0000	-0.5920	169.3321	2	-5	5.85e-01
3	3	0.000	0.001	0.0267	26.8315	0.0000	-0.5914	169.2877	3	-5	5.69e-03
4	4	0.000	0.000	0.0005	26.8319	0.0000	-0.5914	169.2870	4	-5	5.37e-07
4	4	0.000	0.000	0.0000	26.8315	0.0000	-0.5914	169.2877	6	-5	0.00e+00

…

Figure 13. Example Iteration Table Section of LocOO3D Output.

Iteration table fields:

`Itt, It`	Iteration number
`Comment`	Indicates if damping is applied
`N`	Number of observations
`M`	Number of free parameters
`Lat`	Latitude at beginning of iteration
`Lon`	Longitude at beginning of iteration
`Depth`	Depth at beginning of iteration
`Time`	Origin time at beginning of iteration
`rms_Trsd`	RMS of the time residuals
`rms_Wrsd`	RMS of the weighted residuals
`dNorth`	North/south component of the change in location (km)
`dEast`	East/west component of the change in location (km)
`dZ`	Depth component of the change in location (km)
`dT`	Change in time (s)
`dkm`	Distance of the change in location for this iteration (km)
`dxStart`	Distance of the change in x-y location from the initial origin to this iteration (km)
`dzStart`	Change in depth from the initial origin to this iteration (km)
`dtStart`	Change in time from the initial origin to this iteration
`azStart`	Azimuth from the initial origin to the location from this iteration
`nF`	Counter for number of times sum square weighted residuals are computed
`damp`	Applied damping value
`converge`	Least-squared convergence value

D.2. Solution Summary

```
Final location for evid: 15399370   Orid: 15433650

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   79.5262    2.7089     0.000  1518056955.420   2018-02-08 02:29:15.420            2018039

  geographic region: GREENLAND SEA    seismic region ARCTIC ZONE

  converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     4     6     3     6     0     6     6    0.5287    0.4835

    az_gap  az_gap_2 station  Nsta   N30  N250
  275.9591  302.7308     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:

    smajax    sminax     trend      area
   79.8353   52.9144    4.0499  13271.47

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2457


Time to compute this location = 0.350623 seconds
```

Figure 14. Example Solution Summary Section of LocOO3D Output.

Fields in final location solution summary:

`evid`	Event ID. Same value as input event
`orid`	Origin ID. Unique to each computed origin
`Latitude`	Latitude of the final location
`Longitude`	Longitude of the final location
`Depth`	Depth of the final location
`origin_time`	Origin time of the final location (epoch seconds)
`origin_date_gmt`	GMT date and time of the final location
`origin_jdate`	Julian date of the final location
`converged`	Flag to indicate if the solution met convergence criteria
`loc_min`	Possible existence of local minima
`Nit`	Number of iterations to convergence
`Nfunc`	Number of times sum square weighted residuals were calculated
`M`	Number of free parameters
`Nobs`	Number of defining observations
`Ndel`	Number of observations converted to non-defining
`Nass`	Number of arrivals associated with the event
`Ndef`	Number of time defining phases
`sdobs`	Standard deviation of weighted residuals
`rms_wr`	Weighted RMS
`az_gap`	Primary azimuthal gap (radians)
`az_gap_2`	Secondary azimuthal gap (radians)
`station`	Station removed during calculation of secondary azimuthal gap

`Nsta`	Number of stations
`N30`	Number of stations within 30 km of event
`N250`	Number of stations within 250 km of event
`conf`	(0. ≤ confidence ≤ 1.)
`type`	Type of uncertainty ellipse
`K`	K value of Jordan and Sverdrup (1981)
`apriori`	Apriori estimate of the data standard error scale factor
`sigma`	Data variance scale factor used to enhance or diminish the actual data variances for purposes of scaling the location uncertainty
`kappa 1-4`	scaling factors for uncertainty limits. One per free parameter

3D Hypocentral uncertainty ellipsoid:
Length, trend, and plunge of major, minor, and intermediate axes for the 3D uncertainty ellipsoid. Units are km and degrees, respectively.

2D Epicentral uncertainty ellipse:
Major and minor axes, trend, and area of the 2D uncertainty ellipse. Units are km, degrees, and km^2, respectively.

1D linear uncertainties:

`depth_se`	depth uncertainty in km.
`time_se`	time uncertainty in seconds.

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
Jorge Roman-Nieves	jorge.roman-nieves.1@us.af.mil	AFTAC

Email—Internal

Name	Org.	Sandia Email Address
Stephanie Teich-McGoldrick	06756	steichm@sandia.gov
John Merchant	06752	bjmerch@sandia.gov
Daniel Gonzales	06752	dgonza2@sandia.gov
Andrea Conley	06752	acconle@sandia.gov
Kathy Davenport	06756	kdavenp@sandia.gov
Robert Porritt	06756	rwporri@sandia.gov
Julia Sakamoto	06752	jsakomo@sandia.gov
Technical Library	01177	libref@sandia.gov

This page left blank

This page left blank

Sandia National Laboratories

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