SANDIA REPORT
SAND2021-10206
Printed August 2021

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/LocOO3D

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

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

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 as long as appropriate column labels are included in a header file (see Appendix A).

The LocOO3D software and source code is distributed via GitHub at:
https://github.com/sandialabs/LocOO3D
Additional information can be found on the SALSA3D website at:
www.sandia.gov/salsa3d/Software.html

LocOO3D is packaged with a copy of this manual and the associated datasets described in the section below. The LocOO3D software is formatted as a Java jar file compiled with JDK10. If the user wishes to use LocOO3D with a database the Oracle ojdbc*.jar file (version ≥ 7) is required. See Appendix A for more details.

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 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 (available at https://github.com/sandialabs/PCalc), 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 has the ability to 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.

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 parameters. 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 parameters 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.

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

2. LOCOO3D THEORY

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

This page left blank

3. LOCOO3D TUTORIAL AND EXAMPLES

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

1. A properties (*.properties) file which contains the parameter settings used by LocOO3D for a particular run. The format of the properties file and a list of 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 is capable of locating 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 SALSA3D model available from www.sandia.gov/salsa3d.

EXAMPLE 3:
Locating an event with travel time predictions based on libcorr3D corrections to AK135 travel time tables. Requires travel time 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 the same problem as Example 1, but this version uses input data from a database. To run this example the user will need to edit the example05.properties file to specify correct database connection parameters and read input from CSS3.0 formatted database tables accessible to the user. Requires Oracle ojdbc*.jar.

3.1. Setting up LocOO3D

3.1.1. Create directory structure

To execute the examples provided, the user should set up their working directory based on the following file structure.

To explore GeoTess models like SALSA3D, RSTT, and LibCorr3D models, the user may wish to install the `geotess` software. See https://github.com/sandialabs/GeoTessJava for details.

1. Clone the LocOO3D software package from https://github.com/sandialabs/LocOO3D
   - If you do not have git installed on your computer, install it from https://git-scm.com/downloads
   - If you do not have a github directory on your computer, create one, e.g., /Users/$USER/github
   - From a terminal window, cd into the github directory and execute:

     ```
     git clone git@github.com:sandialabs/LocOO3D.git
     ```

     This will download the contents of the LocOO3D repository, including the software and example files.
   - See Section 3.1.2 to set up executables for running LocOO3D.

2. Create a working directory on your system (e.g. snl_location_software).

3. Copy the Examples directory from /Users/$USER/github/locoo3d to the working directory.

4. Download and uncompress the SALSA3D version 3 model and 1D uncertainty tables into the working directory from www.sandia.gov/salsa3d/velocity-models.html:

   - SALSA3D_Model.tgz

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

   - pdu202009Du.zip

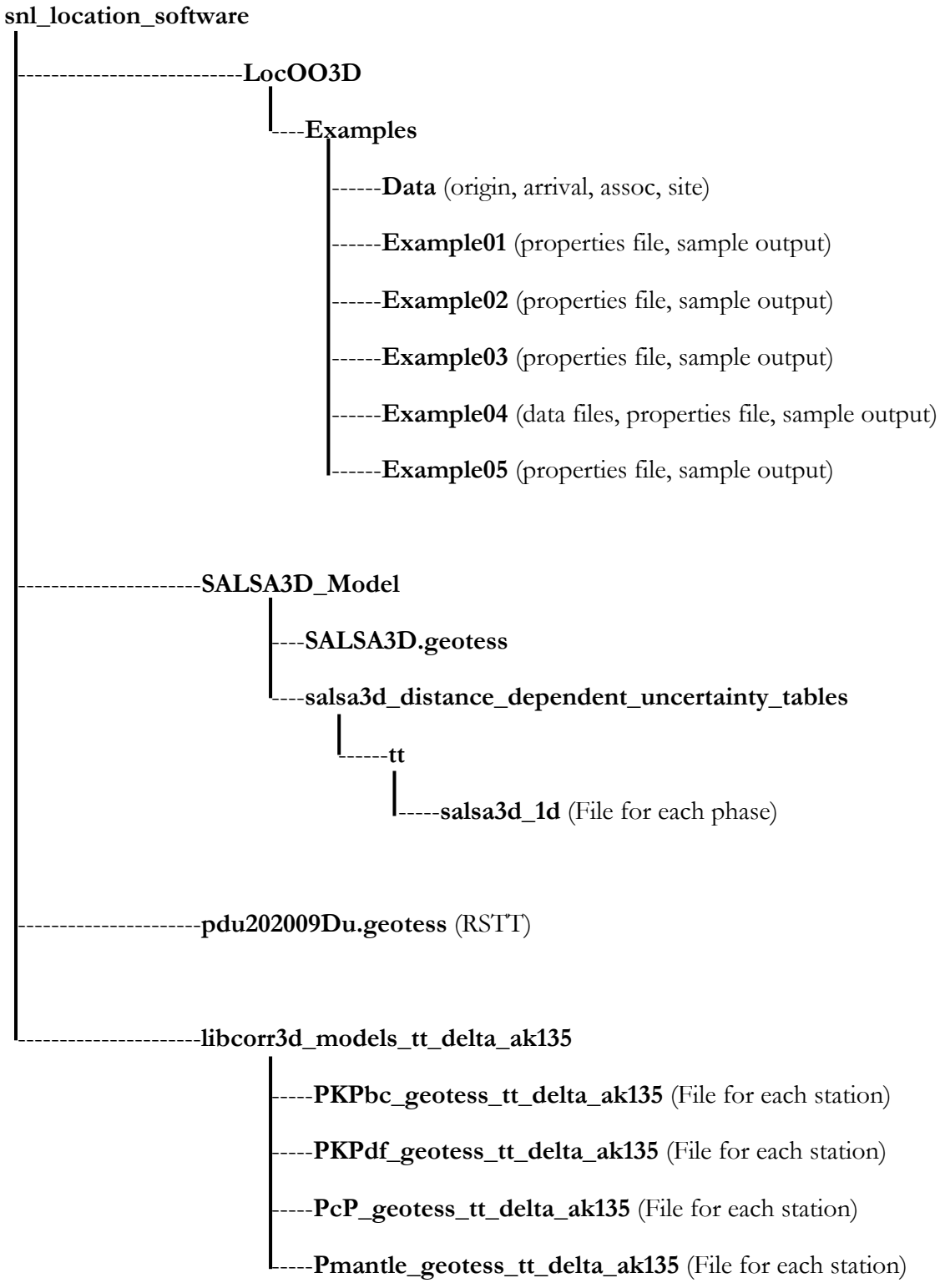
   Follow the installation instructions at www.sandia.gov/rstt to install dependencies and configure RSTT.

6. Uncompress the LibCorr3D surfaces in Examples/Examples_libcorr3d.tgz and move libcorr3d_models_tt_delta_ak135 to the working directory.

   This should produce subdirectoris for Pmantle, PcP, PKPbc, aand PKPdf phases. Each subdirectory contains a model for each station in Example03.

   Travel time tables for additional stations are available under "Depth-dependent travel time table for [phase] phase relative to AK135 in GeoTess Format" at https://www.sandia.gov/salsa3d/Travel%20Time%20Tables.html.

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



3.1.2. Create executables

With the directories appropriately set up, the user can now run the configuration script included with LocOO3D to create executables for locoo3d and supportMap.

Note: If the user wishes to run Example 5 using an Oracle database, they will need to add the path to their Oracle ojdbc*.jar file to the configuration script OJDBC variable. If the user wishes to use RSTT, they will need to add the path to their RSTT libraries to the RSTT variable. See www.sandia.gov/rstt for details on RSTT. Do not allow any spaces around the '=' signs for these variables.

In a Mac/Linux/Unix environment, go to the command line and navigate into the locoo3d directory. Execute:

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

The configure.sh script will generate a 'set path' or 'export' command. This line should be added to your .cshrc, .bashrc, .bash_profile, or .profile file. This will create executable files for locoo3d and supportmap. Source this file or restart terminal before continuing.

3.1.3. Set up support map

Next, the user should set up a support map for the libCorr3D models. For background information about support maps and why they are needed, see Appendix D. Here we will just generate the support map file.

OPTIONAL: If the user is going to access Oracle databases, first update the parameters in supportMap.properties to have specific database information (see DBIO section in Appendix C for details).

Navigate into the working directory Examples directory and run:

```
$supportmap supportMap.properties
```

3.1.4. Run examples

Now the user is ready to run the examples located in the "Examples" directory of the LocOO3D distribution. Within this directory are five directories containing unique examples (Example01, Example02, etc.) as well as the "Data" directory, which contains the common origin, association, site and arrival files used in the first three examples. Each example can be run by navigating into the example subdirectory and running:

```
$locoo3d example01.properties
```

with the appropriate properties file name. Details on each example are provided in the following sections.

3.2. Example 1 – Internal Lookup Tables

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

example01.properties

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

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
#
#####################################################################

dataLoaderType = 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 1. The properties file for Example01.

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

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

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

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

```
executionPath = .
```

specifies 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. For example, we reference `executionPath` in order to specify the directory for the output (`io_log_file`) and error (`io_error_file`) log files:

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

The parameters in Lines 11 and 12:

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

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

The next parameter:

```
loc_predictor_type = lookup2d
```

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

In LocOO3D, any of the four components of the event location (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 (see Appendix C). Here we will solve for a fixed depth solution with origin latitude, longitude, and time as free parameters. We perform this action by specifying the following parameter on Line 18:

```
gen_fix_depth = true
```

This will fix depth to the value reported in "Data/origins.txt" (in this case 0 km). See Appendix C for other fixed depth options.

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

```
dataLoaderType = file

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

Finally, we define the output path and file names. 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` parameters specified for each desired table type.

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

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

With the properties file appropriately set, we will now run Example01. If the configure.sh script has not been run to generate the `locoo3d` executable, please see section 3.1 before proceeding.

To run Example01, cd to the "Examples/Example01/" subdirectory in the working directory and run the following command in a terminal window:

```
$locoo3d example01.properties
```

This command will call locoo3d and execute all of the parameters described in the above properties file. During the run, several lines of output will be printed out to the screen. Following the run, the user should see several output text files including the `locoo3d_output.txt` file. This log file contains a large amount of information on the input parameters used in the run, 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. In order to verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example01 directory for comparison.

```
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 2. Iteration table section of the LocOO3D output for Example01.

The output for this location computation shows that the location algorithm converged to a solution in 4 iterations. The iteration table (Figure 2) summarizes the location and inversion information for each iteration during the relocation, including how much the location changed relative to the initial input origin. Columns in the iteration table are described in detail in Appendix E.

The solution summary table (Figure 3 and Appendix E) includes the final computed event location, time, and misfit information. Since this was a fixed-depth location, the final solution will only report the parameters of a 2D epicentral uncertainty ellipse. 1D linear uncertainties are also printed out; note that the depth uncertainty (`depth_se`) will be invalid for a fixed-depth solution. If a free-depth solution was chosen by the user, the summary would show both a 2D epicentral uncertainty ellipse and a 3D hypocentral uncertainty ellipsoid, along with valid depth and time 1D linear uncertainties.

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

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

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

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

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

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   79.8311   52.9151    4.0541  13270.94

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2458


Time to compute this location =  0.167391 seconds
```

Figure 3. Summary section of the LocOO3D output for Example01.

3.3. Example 2 – Ray Tracing Through a GeoTess Model

LocOO3D contains an internal set of travel time lookup tables that can be used to locate events in the absence of an external velocity model, as was shown in Section 3.2. However, locations computed using the SALSA3D model can improve upon the locations computed with the internal tables. The LocOO3D input files contained in "Examples/Example02/" are set up to use the ray tracer (called "Bender") included with LocOO3D along with the SALSA3D model available at:

http://www.sandia.gov/salsa3d

In preparation to run this example, please see section 3.1 on downloading the model. The "SALSA3D_model" directory should include the SALSA3D.geotess model and the "salsa3d_distance_dependent_uncertainty_tables" directory and files needed to run this example.

To specify that we wish to use an external velocity model to locate this event, the following parameters are added to the "example02.properties" file, in addition to the parameters described in Example 1:

```
# 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:salsa3d_model_directory>/SALSA3D.geotess
benderAllowCMBDiffraction = true
benderAllowMOHODiffraction = true

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

```

Figure 4. Additions to the properties file for Example02.

The addition of the Bender specification to the end of the `loc_predictor_type` property states that we are going to trace rays through the model specified by the `benderModel` property for the phases indicated in parentheses. For all other phases we will use the internal 2D lookup tables based on AK135 described in Section 3.22. The next two properties, `benderAllowCMBDiffraction` and `benderAllowMOHODiffraction` tell Bender to mark phases that diffract off the CMB and Moho as valid. Setting these properties to "false" would disallow the use of these phases in the location computation.

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

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

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

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

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

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

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   73.3660   52.3736    3.7114  12071.39

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2885


Time to compute this location = 23.030108 seconds
```

Figure 5. Summary section of the LocOO3D output for Example02.

Run the example by navigating to the "Examples/Example02" directory and entering the following command in a terminal window:

```
$locoo3d example02.properties
```

To verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example02 directory for comparison. Note the improvement in the solution as compared to Example01:

- Standard deviation from observations (`sdobs`): decrease from 0.5287 to 0.4587
- RMS of weighted residuals (`rms_wr`): decrease from 0.4834 to 0.4187
- Area of 2D uncertainty ellipse: decrease from 13270.94 to 12071.39

3.4. Example 3 – Using GeoTess Lookup Tables

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

If the user has not already done so, download the zip files and place the unzipped travel-time lookup table directories in the directory "libcorr3d_models_tt_delta_ak135" under the working directory (refer to Section 3.1). The directory should contain several lookup tables stored as *.geotess files, where the root file name is the abbreviation for a station and the phase the file supports, e.g. `ZAL_PcP_TT.geotess`. Models for each available seismic phase may be stored together in the root directory or in sub-directories beneath the root directory. Each model supports a single station, one

or more seismic phases, and one or more seismic attributes (such as travel time, travel time uncertainty, etc.). Currently, only P mantle phases (P, Pn) and P core phases (PcP, PKPbc, PKPdf) are included. The user must also have generated and run the supportmap executable at this stage (refer to Section 3.1 for details on this step).

The Example03 properties file `example03.properties` is the same as the Example01 properties with the addition of the following parameters:

```
# 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

```

Figure 6. Additions to the properties file for Example03.

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

With the properties file appropriately defined, we perform the run by navigating to the "Examples/Example03" directory and entering the following command in a terminal window:

```
$locoo3d example03.properties
```

To verify that LocOO3D has run correctly, a log output file run by the authors of this user manual (`locoo3d_output_original.txt`) is included in the Example03 directory for comparison. The output solution summary is shown in Figure 7 below.

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

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

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

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

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

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   41.5095   14.5179   31.9740   1893.22

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    1.3073


Time to compute this location =  0.486393 seconds
```

Figure 7. Summary section of the LocOO3D output for Example03.

3.5. Example 4 – Using RSTT

LocOO3D can also use the 3D Regional Seismic Travel Time model (RSTT, www.sandia.gov/rstt) to predict travel times for Pg, Pn, Sg, and Lg seismic phases, as demonstrated in Example04. The "Examples/Example04/" directory contains the properties file for this example, as well as a "Data" directory that contains information for Pn and Sn observations for a seismic event in central Europe.

This example requires the model file "pdu202009Du.geotess" to be located in the working directory (refer to Section 3.1 for detailed instructions). If needed, follow the installation instructions at www.sandia.gov/rstt to install dependencies and configure RSTT.

The Example03 properties file `example03.properties` is the same as the Example01 properties with the addition of the following parameters:

```
14  # predictorType = [ lookup2d | bender | slbm ]
15  # types that come later in list supersede previous
16  loc_predictor_type = lookup2d, slbm(Pn,Pg,Sn,Lg)
17
18  # SLBM/RSTT settings
19  slbmModel = <property:executionPath>/../../../pdu202009Du.geotess
20
21  slbm_max_distance = 15
22  use_tt_model_uncertainty = true
```

Figure 8. Additions to the properties file for Example04.

The addition of the slbm specification to the end of the `loc_predictor_type` property states that we are going to use the RSTT model specified by the `slbmModel` property for the phases indicated in parentheses. For all other phases we will use the internal 2D lookup tables based on AK135 described in Section 3.22. For RSTT, `maxProcessors` should always be set to 1.

The next property, `slbm_max_distance`, indicates the maximum source-receiver distance, in degrees, at which SLBM will return valid Pn/Sn predicted travel times. Observations greater than this distance will be set to non-defining.

The RSTT model supports path-dependent uncertainty for Pn, Sn, Pg, and Lg phases. The `use_tt_model_uncertainty` flag indicates if this information should be included in travel time residuals and derivatives.

With the properties file appropriately defined, we perform the run by navigating to the "Examples/Example04" directory and entering the following command in a terminal window:

```
$locoo3d example04.properties
```

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 Example04 directory for comparison. The output solution summary is shown in Figure 9 below.

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

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   47.6228   20.2403    13.000  1366669726.378   2013-04-22 22:28:46.378            2013112

 converged  loc_min   Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false     2     4     3    12     0    12    12    0.8444    0.8181

    az_gap  az_gap_2 station  Nsta   N30  N250
   93.9477  159.1739    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:

     majax     minax     trend      area
   20.9929   17.3569  156.7433   1144.70

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    1.5671


Time to compute this location =  0.334205 seconds
```

Figure 9. Summary section of the LocOO3D output for Example04.

3.6. Example 5 – Using Oracle I/O

As previously mentioned, if the user has the necessary Oracle ojdbc*.jar file available (see Appendix A.2 for details), LocOO3D can use Oracle databases for I/O rather than text files. Directory "Examples/Example05" contains the same properties file as Example01, but with the original input/output text file parameters replaced by database parameters, as shown in Figure below.

If the user has access to database tables for the International Data Center (IDC) Reviewed Event Bulletin (REB), then Example05 runs the exact same problem as Example01.

```
20  ######################################################################
21  #
22  # IO Database
23  #
24  ######################################################################
25
26  dataLoaderType = oracle
27
28  # if dataLoaderType = oracle then specify database information
29  dbInputInstance = jdbc:oracle:thin:@domain:port:database
30  dbInputUserName = username
31  dbInputPassword = password
32
33  dbInputOriginTable = gnem_idcreb.origin
34  dbInputAssocTable = gnem_idcreb.assoc
35  dbInputArrivalTable = gnem_idcreb.arrival
36  dbInputSiteTable = gnem_usndc_static.usndc_dec2019_site
37
38  dbInputWhereClause = where origin.orid = 15433650
39
40  # additional components of the where clause used to limit assocs.
41  dbInputAssocClause = assoc.arid in (129798118, 129979918, \
42      129796973, 129796914, 129797143, 129973843)
43
44  # output database
45  dbOutputInstance = jdbc:oracle:thin:@domain:port:database
46  dbOutputUserName = username
47  dbOutputPassword = password
48
49  # destination tables can be specified with prefix and table types
50  dbOutputTablePrefix = deleteme_
51  dbOutputTableTypes = origin, assoc, origerr, azgap
52
53  dbOutputAutoTableCreation = true
54  dbOutputTruncateTables = true
55  dbOutputPromptBeforeTruncate = false
56
```

Figure 10. Database IO parameters for Example05.

The user must update the example05.properties file to provide working database access information for `@domain:port:database`, `username`, and `password` for both the input and output parameters. The input table names should be replaced with database tables to which the user has access. Note that LocOO3D is designed to interact with the CSS3.0 schema or similar format (Appendix A.1).

The user may limit input data read from database tables with "where" clauses that define specific subsets of data. The parameter:

```
dbInputWhereClause = where orid = 15433650
```

limits the origin input to events with orid = 15433650. The parameter:

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

limits the association table to the arrival IDs (arids) listed in parentheses.

Database output table names may be specified by listing a name prefix (`dbOutputTablePrefix`) and table types (`dbOutputTableTypes`), or by naming the individual files. The parameters:

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

produce the same output table names as:

```
dbOutputOriginTable = deleteme_origin
dbOutputAssocTable = deleteme_assoc
dbOutputOrigerrTable = deleteme_origerr
dbOutputAzgapTable = deleteme_azgap
```

The last three parameters, `dbOutputAutoTableCreation`, `dbOutputTruncateTables`, and `dbOutputPromptBeforeTruncate`, control table creation and truncation parameters. See Appendix C for more details on these options.

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 Example05 directory for comparison. The output log file generated after running this example, `locoo3d_output.txt`, should be exactly the same as the log file generated for Example01 with the exception that the log file will now contain messages indicating that I/O is coming from a database, e.g. "Schema: dbInput". The requested output tables should also be written to the database location specified by the `dbOutputInstance` parameter with the prefix "deleteme_" in front of the specific table type, e.g. "deleteme_origin".

This page left blank

4. SUMMARY

LocOO3D is a 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 (available at https://github.com/sandialabs/PCalc) 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 has the ability to 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 distributed as a Java jar file and requires the Java Runtime Environment ≥ 9 to run. Additionally, if the user wishes to use LocOO3D with a database the oracle Ojdbc*.jar file (version ≥ 7) is required. LocOO3D allows users to explore a range of hypotheses affecting event location, especially for monitoring tasks. When used in combination with the related software packages pCalc and GeoTess, it forms an essential piece of a software suite capable of being used to construct, interrogate and test models of Earth's seismic wave velocity. The examples in this manual are intended to be useful for getting users of LocOO3D up and running with their own datasets.

The LocOO3D software is distributed via the SALSA3D website at:

www.sandia.gov/salsa3d/Software.html

and the source code is available via GitHub at:

https://github.com/sandialabs/LocOO3D

This page left blank

REFERENCES

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

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

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

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

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

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

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

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

This page left blank

APPENDIX A. INPUT/OUTPUT SETTINGS FOR LOCOO3D

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

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 parameters for each file are described below. The full specification of the CSS3.0 input file formats and linkages between tables can be found at:

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

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. ONDATE - station turn on date in Julian date format YEARJUL
6. OFFDATE - station turn off date in Julian date format YEARJUL

Arrival input:

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

4. JDATE - Julian date in YEARJUL format
5. DELTIM – Pick uncertainty in seconds
6. AZIMUTH – Station to event azimuth (geographic, only required if azimuth observation is defining)
7. DELAZ – Azimuth uncertainty (only required if azimuth observation is defining)
8. SLO - Slowness observed at station (seconds per degree, only used if slowness is defining)
9. 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`; 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`.

A.2. Oracle Database I/O

LocOO3D can interact with an Oracle Database containing database tables in CSS3.0 format if an appropriate oracle ojdbc*.jar file is specified during configuration (`configure_locoo3d.sh`, `configure_supportmap.sh`) or included on the java classpath. An example command that successfully ran on mac is:

```
java -classpath LocOO3D.jar:ojdbc8.jar gov.sandia.gmp.locoo3d.LocOO example.properties
```

Users must specify the database instance, username, and password in the properties file. There may be separate definitions for input and output. For input, LocOO3D can read **`origin, assoc, arrival`** and **`site`** database tables. LocOO3D can output results to **`origin, assoc, arrival, site, origerr`** and **`azgap`** tables.

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

```
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` and `dbInputAssocClause`) that limit the data that is 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 for additional details.

This page left blank

APPENDIX B. USING LOCOO3D IN PARALLEL

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

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

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

This page left blank

APPENDIX C. PARAMETER DESCRIPTIONS FOR LOCOO3D

C.1. Setting Parameters

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

Notes:

- LocOO parameters are case sensitive.
- All parameters in the parameter file must contain an '=' character, separating the parameter name from the parameter value (e.g. `io_print_to_screen = true`). White space around the '=' sign is optional (ignored).
- Properties can be recursive. If a property value contains a string '`<property:xyz>`' then the phrase '`<property:xyz>`' is replaced with the value of property '`xyz`'. For example, if the following records appear in the property file:

  ```
  testDirectory = /home/abc
  io_log_file = <property:testDirectory>/testdir
  ```

  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. *loc_predictor_type*

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

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), slbm(Pn, Pg)`" then SLBM 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 `slbm(Pn)` comes later in the list then `bender(Pn)`.

Use `lookup2d` for AK135, other travel time tables, or 3D correction files. Use `bender` to ray trace through a 3D GeoTess model like SALSA3D. Use `slbm` to calculate travel times through the RSTT model. All options except lookup2D with default AK135 require additional parameters as described in this Appendix.

C.2.2. *seismicBaseData*

<string> [Default = none]

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

seismicBaseData delivered in the locoo3d jar files includes lookup tables for velocity models ak135 and iasp91.

C.2.3. Lookup2d

Predictor lookup2d defaults to AK135, which is built into LocOO3D. If the user wants to specify other travel time tables, the following parameters may be used.

C.2.3.1. *lookup2dModel*

<string> [Default = ak135] (`ak135 | iasp91`)

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

C.2.3.2. *lookup2dTableDirectory*

<string> [Default = none]

Path of the directory where the travel time lookup tables are located for use with the Lookup2D predictor. This directory will contain a separate file for each phase that will be supported. The file names can be names like 'PKP' or 'ak135.PKP'.

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

C.2.3.3. *lookup2dEllipticityCorrectionsDirectory*

<string> [Default = none]

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

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

C.2.3.4. *lookup2dUseEllipticityCorrections*

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

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

C.2.3.5. *lookup2dUseElevationCorrections*

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

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

C.2.3.6. *lookup2dSedimentaryVelocity*

<double> [Default = 5.8 km/sec]

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

C.2.3.7. *lookup2dTTUncertaintyType*

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

When `lookup2dTTUncertaintyType` is `hierarchical` and a libcorr3d correction surface is available for a particular station-phase, the travel time uncertainty will be retrieved from the libcorr3d correction surface. When `lookup2dTTUncertaintyType` is `hierarchical` and a libcorr3d correction surface is *not* available for a particular station-phase, then DistanceDependent travel time uncertainty will be computed from information in seismicBaseData.

When `lookup2dTTUncertaintyType` is `DistanceDependent` then DistanceDependent travel time uncertainty will be computed from information in seismicBaseData, regardless of whether or not a libcorr3d correction surface is available.

C.2.3.8. *lookup2dTTModelUncertaintyScale*

<2 doubles: scale and offset> [Default = 1.0, 0.0]

Travel time model uncertainty scale and offset. If one value is specified, it will be used to scale the travel time model uncertainty. If two values are specified, the second will be added to the travel time model uncertainty:

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

C.2.4. Bender

C.2.4.1. *benderModel*

<string> [Default = none]

Path to GeoTess-format 3D velocity model that Bender should use 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.2.4.2. *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.2.4.3. *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.2.4.4. *benderPhaseInterfaceToModelInterfaceRemap*

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

Allows remapping 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.2.4.5. *benderUncertaintyType*

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

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

C.2.4.6. *benderUncertaintyDirectory*

<string> [Default = none]

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

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

For example, if uncertainty information is in /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` will be ignored.

C.2.4.7. *benderUncertaintyModel*

<string> [Default = none]

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

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

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

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

C.2.4.8. *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 parameter `benderModel`.

C.2.5. SLBM RSTT

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

C.2.5.1. *slbmModel*

<string> [Default = none]

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

C.2.5.2. *slbmUncertaintyType*

<string> [Default = null] (`SLBM_PATH_DEPENDENT` | `SLBM_DISTANCE_DEPENDENT` | `SLBM_HIERARCHICAL_PATH_DEPENDENT` | `SLBM_HIERARCHICAL_DISTANCE_DEPENDENT`)

When `slbmUncertaintyType` is one of the hierarchical types, and a libcorr3d correction surface is available for a particular station-phase, the travel time uncertainty will be retrieved from the libcorr3d correction surface. When `slbmUncertaintyType` is one of the hierarchical types and a libcorr3d correction surface is *not* available for a particular station-phase, then travel time uncertainty computed by SLBM will be returned.

When `slbmUncertaintyType` is *not* one of the hierarchical types then either path_dependent or distance_dependent travel time uncertainty computed by SLBM will be returned, regardless of whether or not a libcorr3d correction surface is available.

C.2.5.3. *slbm_ch_max*

<double> [Default = 0.2]

Specify the maximum value of c*h where c is the slbm zhao c parameter and h is the turning depth of the ray below the moho. See RSTT documentation for details.

C.2.5.4. *slbm_max_depth*

<double> [Default = 200]

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

C.2.5.5. *slbm_max_distance*

<double> [Default = 15]

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

C.2.5.6. *slbmTTModelUncertaintyScale*

<2 doubles: scale and offset> [Default = 1.0, 0.0]

Travel time model uncertainty scale and offset. If one value is specified, it will be used to scale the travel time model uncertainty. If two values are specified, the second will be added to the travel time model uncertainty:

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

C.2.6. LibCorr3D

Option to apply libcorr3d corrections to predicted travel times. libcorr3d corrections can be applied to lookup2d or rstt/slbm predicted travel times. Corrections are defined in a separate GeoTess-formatted file for each station/phase pair.

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

C.2.6.1. *<predictor>PathCorrectionsType*

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

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

C.2.6.2. *<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.2.6.3. *<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.2.6.4. *<predictor>LibCorrInterpolatorTypeHorizontal*

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

Type of horizontal interpolation to use.

C.2.6.5. *<predictor>LibCorrInterpolatorTypeRadial*

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

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

C.2.6.6. *<predictor>LibCorrPreloadModels*

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

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

C.2.6.7. *<predictor>LibcorrMaxSiteSeparation*

<double> [Default = 10 km]

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

C.2.6.8. *<predictor>LibcorrMatchOnRefsta*

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

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

C.2.6.9. *<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.2.7. *use_tt_path_corrections*

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

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

C.2.8. *use_az_path_corrections*

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

Flag to turn off the use of azimuth path corrections.

C.2.9. *use_sh_path_corrections*

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

Flag to turn off the use of slowness path corrections.

C.2.10. *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.2.11. *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.2.12. *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.3. Master Event Relative Relocation

If parameter `masterEventWhereClause` is specified, then LocOO3D will apply master event relocation corrections to all input origins that it relocates. A single masterEvent origin 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`. 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.3.1. *masterEventWhereClause*

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

If this parameter 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` database schema. The "where" clause must return only one origin row. If the clause returns anything other than one origin row an exception will be thrown.

C.3.2. *masterEventAssocClause*

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

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

C.3.3. *masterEventUseOnlyStationsWithCorrections*

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

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

C.3.4. *masterEventSchema*

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

If this parameter is specified, then all the properties of a valid schema must be supplied (see description of `dbInput` properties in section *DBIO Utility*). 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 parameter is not specified, then it is assumed that the information about the masterEvent origin, assoc, arrival and site will come from the same database tables as the input origins which are to be relocated (`dbInput` schema).

C.4. General

C.4.1. *lsq_max_iterations*

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

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

C.4.2. *gen_initial_location_method*

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

Specifies the method for setting the initial event location.

Options:

`data_file` = Use the initial location given in the origin file/table.

`properties_file` = Use values given in the *.properties file. Requires all of the following 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.4.3. *gen_lat_init*

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

Initial latitude estimate for the algorithm (degrees).

C.4.4. *gen_lon_init*

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

Initial longitude estimate for the algorithm (degrees).

C.4.5. *gen_depth_init*

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

Initial depth estimate for the algorithm (km).

C.4.6. *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.4.7. *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`.

C.4.8. *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`.

C.4.9. Depth Constraints

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

C.4.9.1. *gen_fix_depth*

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

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

If false, 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`.

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`. In this case, property `seismicity_depth_model` is required.

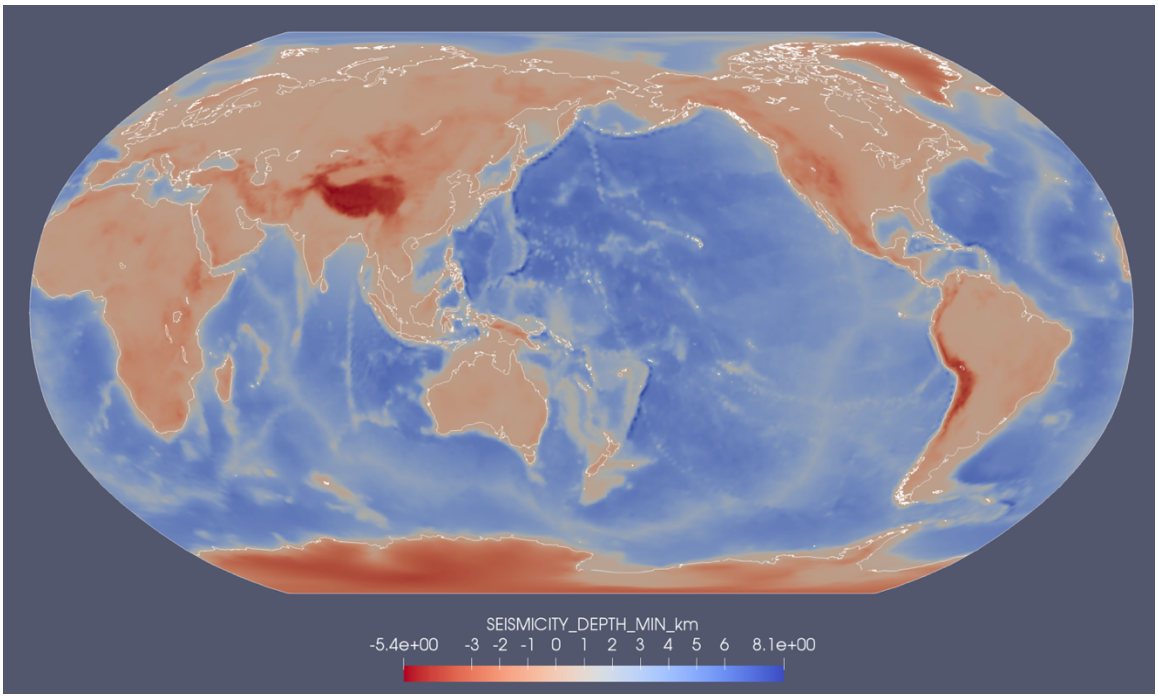
C.4.9.2. *seismicity_depth_model*

<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 average depths of historic seismicity and tectonics. Maps of the upper and lower limits of the default seismicity depth model are shown below:



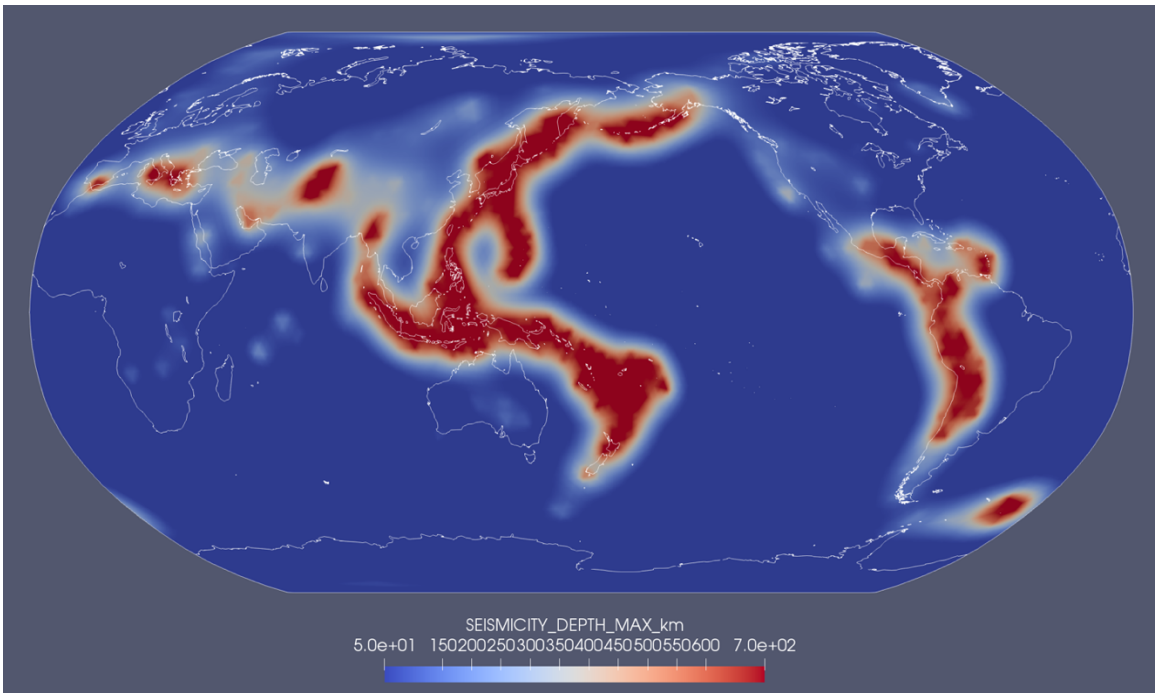


Figure 11. Seismicity depth model. (Top) Upper boundary based on elevation. (Bottom) Lower boundary based on historic seismicity.

If fixed depth solutions are requested by specifying `gen_fix_depth = topography`, 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` and `gen_max_depth`.
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` and `depth_constraint_uncertainy_offset`.

C.4.9.3. *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` is specified.

C.4.9.4. *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` is specified.

C.4.9.5. *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 or not a computed depth is within the allowable range defined by properties `gen_min_depth, gen_max_depth`, or `seismicity_depth_model).`

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.4.9.6. *depth_constraint_uncertainy_offset*

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

Depth uncertainty offset in km. See property `depth_constraint_uncertainy_scale`.
This property is ignored when a fixed depth solution is requested (`gen_fix_depth != false`).

C.4.10. 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`, 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`.

C.4.10.1. *gen_allow_big_residuals*

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

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

C.4.10.2. *gen_big_residual_threshold*

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

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

C.4.10.3. *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 is false and there are observations with large residuals.

C.4.11. *nObservationFlipFlops*

<int> [Default = 10]

There are instances when LocOO3D will drop and then later reintroduce observations (see `gen_allow_big_residuals`). 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.4.12. *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.4.13. *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.4.14. *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.4.15. *gen_defining_observations_filter*

<string> [Default = none]

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

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

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

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

C.4.16. *gen_error_ellipse_type*

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

Type of error ellipse desired:

`coverage`	Dimensions of error ellipse depend only on apriori 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 aposteriori information. The Jordan-Sverdrup K parameter is set to the value of this parameter.

C.4.17. *gen_jordan_sverdrup_K*

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

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

C.4.18. *gen_apriori_standard_error*

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

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

C.4.19. *gen_confidence_level*

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

Uncertainty confidence level desired.

C.4.20. *allowCorePhaseRenamingP*

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

If `true` then core phase renaming will occur at the distance specified by `corePhaseRenamingThresholdDistanceP`.

C.4.21. *corePhaseRenamingThresholdDistanceP*

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

If `allowCorePhaseRenamingP = true`, then this is the distance in degrees beyond which core phase renaming will occur.

C.4.22. *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).

4.1. Correlated Observation Parameters

C.4.23. *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` parameter.
`function`	Function parameters must be specified using the `gen_correlation_scale` parameter.

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

C.4.24. *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.4.25. *gen_correlation_scale*

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

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

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

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

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

C.5. Levenberg-Marquardt Non-Linear Least Squares Solver

C.5.1. *lsq_print_iteration_table*

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

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

C.5.2. *lsq_convergence_n*

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

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

C.5.3. *lsq_applied_damping_multiplier*

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

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

C.5.4. *lsq_convergence_criterion*

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

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

C.5.5. *lsq_damping_dkm_threshold*

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

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

C.5.6. *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.5.7. *lsq_initial_applied_damping*

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

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

C.5.8. *lsq_singular_value_cutoff*

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

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

C.6. Gridded Residuals

C.6.1. *grid_output_file_name*

<string> [Default = none]

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

C.6.2. *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.6.3. *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.`

C.6.4. *grid_origin_lat*

<double> [Default = none]

Latitude in degrees of the center of the grid. Ignored if `grid_origin_source` equals either `epicenter` or `hypocenter`.

C.6.5. *grid_origin_lon*

<double> [Default = none]

Longitude in degrees of the center of the grid. Ignored if `grid_origin_source` equals either `epicenter` or `hypocenter`.

C.6.6. *grid_origin_depth*

<double> [Default = 0. km]

Depth in km of the center of the grid. Ignored if `grid_origin_source` equals either `epicenter` or `hypocenter`.

C.6.7. *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.6.8. *grid_map_width*

<double> [Default = none]

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

C.6.9. *grid_map_height*

<double> [Default = none]

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

C.6.10. *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.6.11. *grid_map_nwidth*

<int> [Default = none]

Map will have this many nodes in the x direction.

C.6.12. *grid_map_nheight*

<int> [Default = none]

Map will have this many nodes in the y direction.

C.6.13. *grid_map_ndepth*

<int> [Default = 1]

Map will have this many nodes in the z direction.

C.7. General Input/Output Parameters

C.7.1. *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.7.2. *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.7.3. *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.7.4. *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.7.5. *io_print_errors_to_screen*

<boolean> [Default = io_verbosity > 0]

Write error messages to the screen.

C.7.6. *io_max_obs_tables*

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

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

C.7.7. *io_observation_sort_order*

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

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

C.7.8. *io_iteration_table*

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

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

C.7.9. *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 parameter does not impact the final computed location.

C.8. DataLoader Utility

C.8.1. *dataLoaderType*

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

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

C.9. DataFileLoader Utility

C.9.1. *dataLoaderFileOrigins*

<string> [Default = none]

Name of file containing the input origins. Required if `dataLoaderType = file`.

C.9.2. *dataLoaderFileAssocs*

<string> [Default = none]

Name of file containing the input assocs. Required if `dataLoaderType = file`.

C.9.3. *dataLoaderFileArrivals*

<string> [Default = none]

Name of file containing the input arrivals. Required if `dataLoaderType = file`.

C.9.4. *dataLoaderFileSites*

<string> [Default = none]

Name of file containing the input sites. Required if `dataLoaderType = file`.

C.9.5. *dataLoaderFileOrids*

<string> [Default = none]

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

C.9.6. *dataLoaderFileOutputOrigins*

<string> [Default = none]

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

C.9.7. *dataLoaderFileOutputOrigerrs*

<string> [Default = none]

Name of file to receive the output origerrs.

C.9.8. *dataLoaderFileOutputAssocs*

<string> [Default = none]

Name of file to receive the output assocs.

C.9.9. *dataLoaderFileOutputAzgaps*

<string> [Default = none]

Name of file to receive the output azgaps.

C.9.10. *dataLoaderFileTokenDelimiter*

<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 tokenDelimiter is read from a properties file.

C.9.11. *dataLoaderFileOutputTokenDelimiter*

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

Assembles the token delimiter for output from the specified value. The specified value is a space delimited string that uses "`tab`" for "\t", "`comma`" for ",", and "`space`" for " ". Any other character can be represented also. For example, if the desired delimiter is ",\t *" then the input string would be

"`comma tab space *`". Using the long names for whitespace characters is beneficial since the input tokenDelimiter is read from a properties file.

C.10. DBIO Utility

C.10.1. *Default database connection parameters*

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

```
DB_INSTANCE = jdbc:oracle:thin:@dwpr2.sandia.gov:1523:dwpr2
DB_WALLET = /Users/$USER/wallet
DB_DRIVER = oracle.jdbc.driver.OracleDriver
DB_USERNAME = GNEM_SBALLAR
DB_PASSWORD_GNEM_SBALLAR = ********
DB_PASSWORD_GNEM_SALSA3D = ********
DB_PASSWORD_GNEM_GMP = ********
```

Alternatively, those same default parameters 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 parameters specified in an application properties file, if present, will override the default values.

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

If both *wallet* and *instance* 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.10.2. *Database Instances and Wallets*

To use wallets at AFTAC:

- All you must do is specify *dbInputUserName/dbOutputUserName* in your properties file, e.g., "*dbInputUserName = global_ro@archive.susndc*". This will work provided the following are true:
  - The location of your wallet is correctly specified in your user environment with environment variable TNS_ADMIN.
  - *dbInputInstance / dbOutputInstance* 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).

To use wallets at SNL and LANL:

- Use one of the following methods to specify the location of your wallet:
  - Specify *dbInputWallet / dbOutputWallet* in your properties file.
  - Specify variable DB_WALLET in your *database.properties* file in your root directory.
  - Specify environment variable DB_WALLET or TNS_ADMIN
- 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:/@gnem_gmp*".
- Do not specify *dbInputIntance/dbOutputInstance* in your properties file.

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.10.3. *dbInputUserName, dbOutputUserName*

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

Database input account username.

C.10.4. *dbInputPassword, dbOutputPassword*

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

Database input/output account passwords.

C.10.5. *dbInputInstance, dbOutputInstance*

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

Database instance for input/output.

C.10.6. *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.10.7. *dbInputDriver, dbOutputDriver*

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

Database driver for input/output.

C.10.8. *dbInputTableTypes*

<string> [Default = none]

If the `dbInputTableTypes` parameter is specified then the input table types specified with this parameter will default to the value of the `dbInputTablePrefix` parameter with the appropriate table type appended on the end. Currently recognized table types include: **origin, assoc, arrival, site.**

C.10.9. *dbInputTablePrefix*

<string> [Default = none]

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

C.10.10. *dbInputOriginTable*

<string> [Default not allowed]

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

C.10.11. *dbInputAssocTable*

<string> [Default not allowed]

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

C.10.12. *dbInputArrivalTable*

<string> [Default not allowed]

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

C.10.13. *dbInputSiteTable*

<string> [Default not allowed]

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

C.10.14. *dbInputWhereClause*

<string> [Default = No default value]

An orid query "where" clause that specifies the origins that should be processed as input for LocOO. This where clause is executed against the origin table. LocOO3D executes the following sql statement in order to load data from the database:

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

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

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

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


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

C.10.15. *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` for details of how this is used.
For example, to include only data from stations within distance of 40 degrees from the origin, specify `dbInputAssocClause = assoc.delta < 40`. To limit the data to include only stations ABC and XYZ, specify `dbInputAssocClause = site.sta in ('ABC', 'XYZ')`. This where clause becomes part of a sql statement that is executed against an assoc, arrival and site table.

C.10.16. *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` for more details.

C.10.17. *dbOutputTablePrefix*

<string> [Default = none]

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

C.10.18. *dbOutputTableTypes*

<string> [Default = none]

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

C.10.19. *dbOutputOriginTable*

<string> [Default = none]

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

The value of orid in output origin rows depends on the value of `dbOutputConstantOrid`. 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 locoo 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.10.20. *dbOutputArrivalTable*

<string> [Default = none]

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

C.10.21. *dbOutputAssocTable*

<string> [Default = none]

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

C.10.22. *dbOutputAzgapTable*

<string> [Default = none]

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

C.10.23. *dbOutputOrigerrTable*

<string> [Default = none]

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

C.10.24. *dbOutputAuthor*

<string> [Default = '-']

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

C.10.25. *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 locoo 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.10.26. *dbOutputAutoTableCreation*

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

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

C.10.27. *dbOutputTruncateTables*

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

Boolean flag should be set to true if output database tables should be automatically truncated at the start of the run. Unless the `dbOutputPromptBeforeTruncate` parameter has been set to false, the user will be prompted before table truncation actually occurs.

C.10.28. *dbOutputPromptBeforeTruncate*

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

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

This page left blank

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 12. 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: -1   Orid: 15433650

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

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

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

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

2D Epicentral uncertainty ellipse:

     majax     minax     trend      area
   79.8311   52.9151    4.0541  13270.94

1D linear uncertainties:

  depth_se   time_se
-9999999999.9990    2.2458


Time to compute this location =  0.167391 seconds
```

Figure 13. Example 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 with largest 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 variance 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 intersection axes for the 3D uncertainty ellipsoid

2D Epicentral uncertainty ellipse:
Major and minor axes, trend, and area of the 2D uncertainty ellipse

1D linear uncertainties:

`depth_se`	depth uncertainty
`time_se`	time uncertainty

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
Leslie Casey	leslie.casey@nnsa.doe.gov	National Nuclear Security Administration
Sanford Ballard	sballard999@gmail.com	Retired

Email—Internal

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