SANDIA REPORT
SAND2019-9542
Printed August 2019

LocOO3D User's Manual

Nathan J. Downey[1], Sanford Ballard[1], James R. Hipp[1] and Mike Begnaud[2]

[1]*Sandia National Laboratories*
[2]*Los Alamos National Laboratory*

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@osti.gov
Online ordering:	http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.gov
Online order:	https://classic.ntis.gov/help/order-methods/

ABSTRACT

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

www.sandia.gov/salsa3d/Software.html

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

This page left blank

CONTENTS

LIST OF FIGURES

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
CSS	Center for Seismic Studies
SALSA3D	Sandia-Los Alamos 3D velocity model
SVD	Singular value decomposition
DBIO	Database input/output
3D	Three dimensional

This page left blank

1. INTRODUCTION

LocOO3D (Object-Oriented 3 Dimensional Location Software) is a software package used for locating (and re-locating) single seismic events using a variety of seismic velocity models. Reflecting its origin in the seismic monitoring community, LocOO3D is compatible with the CSS3.0 data format commonly used by monitoring agencies. These data tables can either be stored in an Oracle database or input as flat files. Additionally, data can be input as custom formatted text as long as appropriate column labels are included in a header file (see Appendix 1).

The LocOO3D software is distributed via 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 "Tutorial and Examples" section below. The software is formatted as a JAVA jar file and requires the java runtime environment 1.8 to run. Additionally, if the user wishes to use LocOO3D with a database the oracle OJDBC.jar is required. See Appendix 1 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 locate events using a variety of velocity models, including AK135, which is directly built into the software. Additionally, users can construct a 3D model in GeoTESS format (see www.sandia.gov/geotess), use the SALSA3D GeoTESS models (www.sandia.gov/salsa3d) or travel-time lookup surfaces for event location.

- In conjunction with the software pCalc (www.sandia.gov/salsa3d) users can construct their own travel-time look up surfaces in GeoTESS format for use in event location.

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

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

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

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

A typical run of LocOO3D consists of the following steps:

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

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

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

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

See the "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).

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

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

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

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

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

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

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

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

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

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

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

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

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

Using this equation, we have

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

We apply a weighting to this equation so it is rewritten

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

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

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

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

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

The solution of which yields the normal equation

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

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

2.1. Convergence Criteria

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

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

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

2.2. Poorly Constrained Events

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

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

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

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

2.3. Non-linear Effects

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

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

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

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

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

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

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

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

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

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

    c. return to Step 2.

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

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

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

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

This algorithm seeks to keep λ as small as possible by initializing it to a negligible value and by reducing it by factor λ_x at the conclusion of each successful iteration. Keeping λ small keeps the step size as large as possible. The algorithm is willing to increase λ as much as necessary, however, to ensure that $\|\boldsymbol{r}\|^2$ is never allowed to increase from one iteration to the next.
It is possible that the event location $\mathbf{m}_i$ at the start of some iteration i is characterized by a value of $\|\boldsymbol{r}\|^2_i$ which is very close to $\|\boldsymbol{r}\|^2_{min}$ and a value of $\|\boldsymbol{r}\|^2_{i+1}$ that is significantly lower than $\|\boldsymbol{r}\|^2_i$ will never be found. This situation is handled by monitoring the length of $\delta\mathbf{m}$, an approximation of which is given by

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

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

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

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

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

The format of the properties file and a list of all the possible properties that can be set is described in Appendix 3. Appendix 1 describes the format of the origin, association, site and arrival files.

```
parallelMode = sequential
maxProcessors = 1

executionName = Example01
#User will need to use proper path separator for their system
executionPath = ~/Locoo3d/Users_Manual/Examples/Example01/

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

# predictorType = [ lookup2d | bender ]
loc_predictor_type = lookup2d

# General
gen_fix_depth = false

outputTableTypes = origin,assoc,origerr

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

dataLoaderFileOutputOrigins = <property:executionPath>origins_output.txt
dataLoaderFileOutputAssocs = <property:executionPath>assocs_output.txt
dataLoaderFileOutputOrigerrs = <property:executionPath>origerrs_output.txt

dataLoaderFileOutputType = file
```

Figure 3. ***The properties file for Example01 as included in the directory "Examples/Example01"***

The series of example runs is located in the "Examples" directory of the LocOO3D distribution. Here we will describe these runs in some detail and hope that these examples will serve as a starting point from which users can create their own location datasets.

3.1. Example 1 – Internal Lookup Tables

"Examples/Example01" contains a dataset for an event that occurred in Nevada. The starting origin is contained in "origins_input.txt". For this example, we are relocating only one event so there is only one origin specified in this file, but LocOO3D has the capability to locate several events during a single software run. The file "arrivals_sub.txt" contains the arrivals used to relocate this event, and the file "assocs_input.txt" contains information that references each arrival to an origin. For this example, we have only included the arrivals and association information for this particular event, but in general these files will be much larger and contain arrival and association information for many events. LocOO3D parses through these files and only uses the information required to locate a particular event. Finally, the file "sites_sub.txt" contains information about the stations at which the arrivals for this event were recorded.

The parameters used by LocOO3D during a run are specified in a properties file. For this first example, the properties are in a file called "example01.properties" (See Figure 1). This example represents a minimum set of parameters required for LocOO3D to run (See Appendix 3 for a more exhaustive list with descriptions).

LocOO3D has the ability to run in parallel on suitably equipped machines (See Appendix 2 for details). The lines:

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

Specify that we are going to run this example in serial mode on a single processor. The block of text:

```
executionName = Example01
executionPath = ~/LocOO3D/Examples/Example01/
```

Specify the run name for this example, along with the directory in which the example is to be run. The syntax `<property:propertyName>` can be used to reference properties defined higher in the input file. As done, for example, in the next block of lines:

```
io_log_file = <property:executionPath>locoo3d_output-<property:executionName>.txt
io_error_file = <property:executionPath>locoo3d_error-<property:executionName>.txt
io_print_to_screen = true
io_verbosity = 4
```

which specify the output and error file names. In this case, the `io_log_file` will be written to: `~/LocOO3D/Examples/Example01/locoo3d_output-Example01.txt`. The last two lines in this block specify if the output will also be print to the screen and defines the io verbosity level. See Appendix 3 to see the different options for these parameters. The next parameter:

```
loc_predictor_type = lookup2d
```

Specifies the type of travel-time predictor used during the location process. For this example, we are using the 2D (event depth of zero km) travel-time lookup tables included with the LocOO3D software. These lookup travel times were computed using the tau-p method and the AK135 model. LocOO3D supports many different methods to calculate travel times, see Appendix 3 and the examples below to see other options.

```
gen_fix_depth = false
```

This parameter sets whether we are computing free or fixed-depth solutions for the event locations. "false" indicates that we are going to compute a free-depth location.

```
outputTableTypes = origin,assoc,origerr
```

Specifies which output tables are to be written to file or to a database, if one is used. The list parameters:

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

Specify that the origin, assoc, site and arrival tables are to be read from files and specifies the file names using the `executionPath` property. Finally,

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

Specifies that the output origin, association and origin errors will be written to named files.
In order to run this example, the Java runtime environment 1.8 (or newer) needs to be installed, and the software archive unzipped in a user directory. After opening a terminal session and changing to the "LocOO3D/Examples/Example01/" directory, the command:

```
$java -jar ../../LocOO3D.jar example01.parameters
```

Will run LocOO3D using this example dataset.

The output from LocOO3D for this run is printed to the screen and also to the `locoo3d_output-Example01.txt` file. The contents of this file contain a lot of information, but the summary info is shown in Figure 2.

```
Final location for evid: 4587038   Orid: 9043717

  latitude longitude     depth     origin_time          origin_date_gmt       origin_jdate
   36.0839 -114.6321    -0.000  1319569118.619   2011-10-25 18:58:38.619            2011298

 converged  loc_min    Nit Nfunc     M  Nobs  Ndel  Nass  Ndef     sdobs    rms_wr
      true    false      4     6     3    14     1    15    14    0.4493    0.4333

    az_gap  az_gap_2 station  Nsta    N30  N250
  197.8413  235.5228    R11A    10      0     7

      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
   19.2813   11.4932   82.7578    696.19

1D linear uncertainties:

  depth_se    time_se
-9999999999.9990    2.2089
```

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

The output for this location computation shows that the location algorithm converged to a solution in 4 iterations. Since this was a fixed-depth location, the depth uncertainty is invalid, however the uncertainty of the epicenter is calculated, and the result displayed in Figure 2.

3.2. Example 2 – Ray Tracing Through a GeoTESS Model

The LocOO3D input files contained in Examples/Example02/ are set up to compute a location for an event in Utah. This example run uses the ray tracer (called "bender") included within the LocOO3D *.jar file along with the SALSA3D model available at:

http://www.sandia.gov/salsa3d/SALSA3D_Model1.tgz

In preparation to run this example, download the above archive and unzip it in the Examples/Examples02 directory. 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. However, the locations computed using the SALSA3D model can be much better than those computed with these internal tables. To specify that we wish to use an external velocity model to locate this event the following lines are present to the "example02.properties" file:

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

# Bender
benderModel = <property:executionPath>/SALSA3D_Model/SALSA3D.geotess
benderAllowCMBDiffraction = true
benderAllowMOHODiffraction = true
```

The addition of the bender specification to the end of the "loc_predictor_type" property states that we are going to trace rays through the model specified by the "benderModel" property for the phases indicated in the parentheses. For all other phases we are going to use the internal 2D lookup tables. The final two properties, "benderAllowCMBDiffraction" and "benderAllowMOHODiffraction" tell bender to mark phases that diffract off the CMB and Moho as valid. Setting these properties to "false" will disallow these phases' use in the location computation. The rest of this property file is very similar to that for Example 1 described above.

Note that the input tables for this example contain much more data than those included in example 1. This is because they contain the full suite of data used in all these examples. You can locate all these events in a single LocOO3D run, using whatever predictor (lookup2d, bender,…) you want by uncommenting all the lines after the header in the "origins_input.txt" file.

3.3. Example 3 – Using GeoTESS Lookup Tables

It is possible, within LocOO3D, to apply a set of travel-time corrections to the internal AK135-based lookup tables that allow travel-times computed using an arbitrary model to be used to locate events, without having to trace rays through the model. Example 3 demonstrates this capability using the travel-time lookup tables distributed for the SALSA3D model on the SALSA3D website at www.sandia.gov/salsa3d.

This example computes a location for the September 2017 North Korea nuclear test (note that the starting origin for this event is that reported by the USGS Earthquake Hazards Program at earthquake.usgs.gov/earthquakes/eventpage/us2000aert/origin/detail on May 20, 2019). This example computes a new location for this event using the P-arrival corrections contained in the archive at www.sandia.gov/salsa3d/Pmantle_geotess_AK135.tgz. To use these corrections, download and unzip this archive in the Examples/Example03 directory. The lookup tables will be stored in files with a *.geotess file name, where the root file name is the abbreviation for a station in the IMS network. These files contain corrections and uncertainties for travel-times of the first-arriving P phase at each IMS station. The uncertainties contained in these files is that computed using the full covariance matrix for the SALSA3D model and provide the best constraint on event locations computed with the SALSA3D model.

The additional lines in the properties file required to use these tables are:

```
lookup2dPathCorrectionsType=libcorr
lookup2dLibCorrPathCorrectionsRoot=<property:executionPath>Pmantle_geotess_AK135
lookup2dLibCorrPreloadModels=false
```

The property "lookup2dPathCorrectionsType=libcorr" tells LocOO3D to apply the corrections contained in the directory specified by the "lookup2dLibCorrPathCorrectionsRoot" property. The final property specifies that we are only going to load the models that we need and not all the models at the beginning of the run. Setting this option to "false" preserves memory.

Commenting out these three lines in the properties file will force LocOO3D to use the internal AK135-based travel time tables. By comparing the location for this event both with and without using the SALSA3D tables shows that the SALSA3D model results in a much better-constrained location.

Caution:

The SALSA3D travel time look up tables were created with a slightly different site definition than is contained in the sites.txt input file. This will cause LocOO3D to only use the SALSA3D tables for sites whose ondate, offdate and location match that in "sites.txt". A way to fix this problem is to rewrite the site table using the information contained in the "_supportMap.txt" file contained in the lookup table directory (for this example in "Pmantle_geotess_AK135"). This file is automatically generated when LocOO3D attempts to locate an event using the travel time tables, so you will need to run LocOO3D with "example03.properties" at least once to see this file. We have included a python utility script ("fixSites.py") that shows how the input site table can be corrected to work with the SALSA3D tables in the "Examples/Examples03" directory. This script generates a new site table, called "sites_mod.txt", which, when used in place of "sites.txt" will allow LocOO3D to load all the tables for this phase.

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 locate events using a variety of velocity models, including AK135, which is directly built into the software. Additionally, users can construct a 3D model in GeoTESS format (see www.sandia.gov/geotess), use the SALSA3D GeoTESS models (www.sandia.gov/salsa3d) or travel-time lookup surfaces for event location.

- In conjunction with the software pCalc (www.sandia.gov/salsa3d) users can construct their own travel-time look up surfaces in GeoTESS format for use in event location.

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

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

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

- Sophisticated error estimations for locations. 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 1.8 to run. Additionally, if the user wishes to use LocOO3D with a database the oracle OJDBC.jar 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, described above, 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

This page left blank

REFERENCES

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

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

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

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

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

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

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

APPENDIX A. INPUT/OUTPUT SETTINGS FOR LOCOO3D

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

A.1. Flat File I/O

The user specifies the names of files from which input data will be loaded and to which output results will be written. Users can limit which origins in the input origins table are processed with "dataLoaderFileOrids". Flat files are read/written in CSS 3.0 or similar format. See Section "*DataFileLoader Utility*" in Appendix 3 for additional information.

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

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

LocOO3D is compatible with this format when entries are **tab-delimited** to allow quotes and commas in station names. Examples 2 and 3 include input files specified in this format.

However, since LocOO3D does not use all of the fields specified in the CSS3.0 format, the LocOO3D file reader is compatible with a simplified format. For each of the input files, the order of the required fields is specified in a single header line located at the top of the file, whose first character is a hash "#" and whose entries are tab-delimited. The required fields can be specified in any order. This header line is followed by the data in a series of tab-delimited lines with an order matching that of the header. The input files included with Example 1 represent a minimal set of parameters specified in this format.

The fields required for each input file are:

Origin input file:

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

Assoc input file:

1. ARID - arrival identifier. links to the arrival and stamag tables
2. ORID - origin identifier. links to the origin , origerr , origaux , netmag, and stamag tables
3. PHASE - final phase name used in event location

4. TIMEDEF – time used (d) or not (n) in event location

5. AZDEF – azimuth used (d) or not (n) in event location

6. SLODEF - slowness used (d) or not (n) in event location

Site input file:

1. STA - station code. links to the arrival , assoc , and stamg tables.

2. LAT - station latitude

3. LON - station longitude

4. ELEV - station elevation in km

5. ONDATE - station turn on date in Julian date

6. OFFDATE - station turn off date in Julian date

Arrival input file:

1. STA - stations code; links to the site table

2. TIME - arrival time (epoch time in second)

3. ARID - arrival identifier; links to the assoc and stamag tables

4. JDATE - Julian date

5. DELTIM – Pick uncertainty in seconds

6. AZIMUTH – Station to event azimuth (geographic, only used if azimuth is defining)

7. DELAZ – Azimuth uncertainty (only required if azimuth observation is defining)

8. SLO - Slowness observed at station (s/deg, only used if slowness is defining)

9. DELSLO – Slowness uncertainty (only required if slowness observation is defining)

A.2. Oracle Database I/O

LocOO3D can interact with an Oracle Database containing database tables in CSS 3.0 format. For this to work, users must specify the database instance, username, password and driver. There must be separate definitions for input and output. For input, LocOO3D can read origin, assoc, arrival and site database tables. LocOO3D can output results to origin, assoc, arrival, site, origerr and azgap tables.

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

dbOutputTableTypes = origin, assoc, origerr, azgap

*dbOutputTablePrefix = prefix_*
dbOutputTableSuffix = _suffix

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

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

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

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

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

See Section DBIO in Appendix 3 for additional details.

Note that the Oracle jar file that supports interaction with Oracle databases (ojdbc*.jar) is not supplied in the LocOO3D distribution. That jar file must be included on the java classpath in order for database interactions to be successful.

An example that successfully ran on mac is:

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

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 3 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(xxx).

C.2. Predictors

C.2.1. *loc_predictor_type*

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

String indicating list of predictors that are to be used. For example, if value is "lookup2d, bender(P, Pn), slbm(Pn, Pg)" then lookup2d will be used for all phases not specified later in the list, Bender will be used for phase P and SLBM will be used for phase Pn and Pg. Even though Pn is specified by bender, it will be computed by slbm since slbm(Pn) comes later in the list then bender(Pn).

C.2.2. *seismicBaseData*

<string> [Default = none] ()

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

C.2.3. *lookup2dModel*

<string> [Default = ak135] (ak135)

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

C.2.4. *lookup2dTableDirectory*

<string> [Default = none] ()

Name of the directory where the travel time lookup tables reside. This directory will contain a separate file for each phase that will be supported. The file names can be names like 'PKP' or 'ak135.PKP'.

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

C.2.5. ***lookup2dEllipticityCorrectionsDirectory***

<string> [Default = none] ()
Path of the directory where ellipticity correction coefficients are located for use with the Lookup2D predictor.
If *lookup2dTableDirectory* and *lookup2dEllipticityCorrectionsDirectory* are both specified, then they will dictate the locations of the table directory and ellipticity corrections directories, as advertised. If either of them is not specified then the locations of both directories will be deduced from property *seismicBaseData*.

C.2.6. ***lookup2dUseEllipticityCorrections***

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

C.2.7. ***lookup2dUseElevationCorrections***

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

C.2.8. ***lookup2dSedimentaryVelocity***

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

C.2.9. ***lookup2d TTModelUncertaintyScale***

<2 doubles: scale and offset> [Default = null] ()
Travel time model uncertainty scale and offset. If one value is specified, it will be used to scale the travel time model uncertainty. If two values are specified, the second will be added to the travel time model uncertainty. In other words,
ttModelUncertainty = ttModelUncertainty * scale + offset.

C.2.10. ***benderModel***

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

C.2.11. ***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.2.12. ***benderUncertaintyDirectory***

<string> [Default = none] ()
Directory where distance dependent uncertainty values can be found for use with Bender predictions. Expecting to find subdirectories such as
<benderUncertaintyDirectory>/<attribute>/< <benderUncertaintyModel>

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

C.2.13. *benderUncertaintyModel*

<string> [Default = none] ()
Subdirectory where distance dependent uncertainty values can be found for use with Bender predictions. Expecting to find subdirectories such as
<benderUncertaintyDirectory>/<attribute>/< <benderUncertaintyModel>
For example: if uncertainty information is in file
/index/SNL_tool_Root/seismicBaseData/tt/ak135 then specify
benderUncertaintyDirectory = /index/SNL_tool_Root/seismicBaseData
benderUncertaintyModel = ak135

C.2.14. *use_tt_site_terms*

<boolean> [Default = true]
If true then travel time site terms computed for each station during tomography are applied to computed values. The site terms are stored in the GeoTessModel specified with parameter *benderModel*.

C.2.15. *use_tt_model_uncertainty*

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

C.2.16. *use_az_model_uncertainty*

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

C.2.17. *use_sh_model_uncertainty*

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

C.3. Master Event Relative Relocation

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

C.3.1. *masterEventWhereClause*

<string>   [Default = null]   (valid sql; eg. 'orid = 100')
If this parameter is present, then master event relocation option will be implemented. Must provide a valid sql where clause that will be executed against the *masterEventSchema* database schema. The where clause must return one and only one origin row. It the where clause returns anything other than one origin row, an exception is thrown.

C.3.2. *masterEventAssocClause*

<string>   [Default = null]   (valid sql; eg. 'phase = 'P'')
Optional parameter to limit the assoc rows returned with the master event origin row.

C.3.3. *masterEventUseOnlyStationsWithCorrections*

<boolean>   [Default = false]   (true or false)
If true then the only assocs that will be used to locate input origins are those that have valid master event corrections.

C.3.4. *masterEventSchema*

<string>   [Default = dbInput]   (schemaName, eg. 'dbMaster')
If this parameter is specified, then all the properties of a valid Schema must be supplied (see description of *dbInput* properties in the *DBIO Utility* section). For example, if property *masterEventSchema = dbMaster*, then properties *dbMasterUserName, dbMasterPassword, dbMasterOriginTable*, etc, will be recognized and all information about the *masterEvent* origin will be retrieved from database tables as specified.
If this parameter is not specified, then it is assumed that the information about the masterEvent origin, assoc, arrival and site will come from the same database tables as the input origins which are to be relocated (*dbInput* schema).

C.4. LibCorr3D

C.4.1. *lookup2dPathCorrectionsType*

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

C.4.2. *lookup2dLibCorrPathCorrectionsRoot*

<string> [Default = none]
The name of the directory where all the libcorr3D correction surfaces reside. This directory should contain a separate file for each correction surface.

C.4.3. *lookup2dLibCorrPathCorrectionsRelativeGridPath*

<string> [Default = "."]

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

C.4.4. *lookup2dLibCorrInterpolatorType*

<string> [Default = “linear”] (linear | natural_neighbor)
Type of horizontal interpolation to use.

C.4.5. *lookup2dLibCorrPreloadModels*

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

C.4.6. *lookup2dUsePathCorrectionsInDerivatives*

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

C.5. General

C.5.1. *lsq_max_iterations*

<int> [Default = 100] (0 <= X <= 100000)
Maximum allowable number of iterations. If this number is set to 0, the LocOO3D simply computes the residuals and location uncertainty information at the initial location and outputs the results.

C.5.2. *gen_initial_location_method*

<string> [Default = data_file] (data_file | properties_file | internal)
Specifies the method for setting the initial event location.

Options:

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

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

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

C.5.3. *gen_lat_init*

<double> [Default = 0.0000] (-90.00 <= X <= 90.000)
Initial latitude estimate for the algorithm (degrees). If this parameter is specified, then the initial latitude is set to the value of this parameter, regardless of the setting of gen_initial_location_method. For initial location to be specified in properties file, all four parameters gen_lat_init, gen_lon_init, gen_depth_init and gen_origin_time_init must be specified in the properties file.

C.5.4. ***gen_lon_init***

<double> [Default = 0.0000] (-180.0 <= X <= 360.00)
Initial longitude estimate for the algorithm (degrees). If this parameter is specified, then the initial longitude is set to the value of this parameter, regardless of the setting of gen_initial_location_method.
For initial location to be specified in properties file, all four parameters gen_lat_init, gen_lon_init, gen_depth_init and gen_origin_time_init must be specified in the properties file.

C.5.5. ***gen_depth_init***

<double, or "topography"> [Default = Globals.NA_Value] (-10000 <= X <= 10000.)
Initial depth estimate for the algorithm (km). If this parameter is specified, then the initial depth is set to the value of this parameter. If the value of this parameter is a string that starts with "topo" (case insensitive) then initial depth will be set to the depth of the topographic/bathymetric surface interpolated from the topography model at the latitude, longitude position of the initial location. If gen_depth_init is set to 'topography' then parameter topo_model must specify the topography model to use.
For initial location to be specified in properties file, all four parameters gen_lat_init, gen_lon_init, gen_depth_init and gen_origin_time_init must be specified in the properties file.

C.5.6. ***gen_origin_time_init***

<string> [Default = 1970-01-01 00:00:00.000] (any reasonable date or epoch time)
Initial origin time estimate for the algorithm. The specified value can be either a date (yyyy-mm-dd hh:mm:ss.sss) or an epoch time (seconds since 1970-01-01 00:00:00.000). If this parameter is specified, then the initial origin time is set to the value of this parameter, regardless of the setting of gen_initial_location_method.
LocOO3D first tries to interpret the value of this parameter as a date in the format yyyy-mm-dd hh:mm:ss.sss. If that fails, it tries to convert the value to an integer (value may not include '.'). If successful, the value is interpreted as a jdate. If unsuccessful, LocOO3D attempts to convert the value to a number of type double (must include a '.'). If successful then the value is interpreted as an epoch time (seconds since 1970). If that also fails, LocOO3D throws an exception.
For initial location to be specified in properties file, all four parameters gen_lat_init, gen_lon_init, gen_depth_init and gen_origin_time_init must be specified in the properties file.

C.5.7. ***topo_model***

<string> [Default = null] (none)
If the values of properties gen_depth_init or gen_min_depth start with 'topo' (case insensitive) then locoo will interrogate the specified GeoModel object to determine the local topography at the latitude, longitude position of the event. The value must specify the name of a file that contains a valid topography model in GeoModel format (these files usually have a .bin extension).
LocOO3D version 1.3 was delivered with 4 topography models named topo_000250.bin, topo_000500.bin, topo_001000.bin and topo_002000.bin. The numbers in the model names indicate the resolution of the model in millidegrees. The models have resolutions of 0.25°, 0.5°, 1°, 2°, respectively.

C.5.8. ***gen_fix_lat_lon***

<bool> [Default = false]

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

C.5.9. ***gen_fix_depth***

<bool> [Default = false]
Hold (true) / don't hold (false) the depth fixed during the solution algorithm. If held fixed, then the initial value specified by gen_depth_init is used. See also gen_initial_location_method, gen_depth_init.

C.5.10. ***gen_fix_origin_time***

<bool> [Default = false]

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

C.5.11. ***gen_allow_big_residuals***

<bool> [Default = true]
Allow (true) / don't allow (false) observations that result in 'big' residual values.
Here is how this works: The event is first located using all defining observations. If any observations have weighted residuals greater than the value specified with gen_big_residual_threshold (default value is 3), then some subset of those observations are set to non-defining and the event is relocated. This process continues until either there are no observations whose weighted residuals exceed the threshold or N = M where N is the number of defining observations and M is the number of degrees of freedom in the problem (4 for free depth solutions, 3 for fixed depth, etc). If there are observations with large residuals, then the subset of them that is set to non-defining is determined as follows. The minimum number that will be set to non-defining is 1. The maximum number that can be set to non-defining is (N-M) * gen_big_residual_max_fraction. gen_big_residual_max_fraction defaults to 0.1.
See gen_big_residual_threshold and _big_residual_max_fraction.

C.5.12. ***gen_big_residual_threshold***

<double> [Default = 3.0000] (0.0000 <= X <= 100000)
Threshold weighted residual value above which observations are flagged as 'big'. See gen_allow_big_residuals.

C.5.13. ***gen_big_residual_max_fraction***

<double> [Default = 0.10] (0.0 <= X <= 1.0)
A constraint on the maximum number of observations that can be set to non-defining when gen_allow_big_residuals is false and there are observations with large residuals.

C.5.14. ***gen_max_depth***

<double> [Default = 700.00] (-1e6 <= X <= 1e6.)
Maximum depth constraint (km).

C.5.15. ***gen_min_depth***

<double, or "topography"> [Default = -999999.] (-10000 <= X <= 10000.)
Minimum depth constraint (km). If the value of this parameter is a string that starts with "topo" (case insensitive) then the depth of the event location is constrained to be no less than the depth of the local surface topography/bathymetry.

C.5.16. ***gen_defining_phases***

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

C.5.17. ***gen_defining_stations***

<string> [Default = all]
The subset of defining stations to use for the location algorithm. This overrides the assoc table values. Setting the value to 'all' is the default behavior and causes all previously indicated defining stations to be used. A comma-delimited list of station names overrides this behavior 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.5.18. ***gen_defining_attributes***

<string> [Default = all] (t, a, s)
The subset of defining attributes (travel time, azimuth, slowness) to use for the location algorithm. This overrides the data file or assoc table values. Setting the value to 'all' is the default behavior and causes all previously indicated defining attributes to be used. A comma-delimited list overrides this behavior and down-selects from the set of defining stations. 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.5.19. ***gen_defining_observations_filter***

<string> [Default = none]
A set of filters that can be used to toggle individual observations from defining to non-defining and vice versa. Each filter is of the form: ±ORID/STATION/PHASE/ATTRIBUTE. The set of filters consists of a number of individual filters, separated by commas. Each observation starts out as either defining or non-defining (after application of parameters gen_defining_stations, gen_defining_phases and gen_defining_attributes). Then the observation is subjected to each filter, in order. If the observation matches the filter, then the observation is made defining if the first character of the filter is '+' or non-defining if the first character of the filter is '-'. Each component of a filter can be the '*' character. Every observation matches the component of a filter that is the '*' character.
For example, the filter +100/ILAR/P/TT will guarantee that for the origin with orid 100, the P travel time observation from station ILAR will be defining. The filter
-*/ILAR/*/*, +*/ILAR/P/t

will first make all observations from station ILAR non-defining, then turn back on just the P travel time observations from station ILAR.

C.5.20. ***gen_error_ellipse_type***

<string> [Default = coverage] (coverage|confidence|mixed)
Type of error ellipse desired:

coverage	Dimensions of error ellipse depend only on 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.5.21. ***gen_jordan_sverdrup_K***

<int> [Default = -1] (-1 <= X <= 999999)
Jordan-Sverdrup K parameter for 'mixed' type confidence ellipses. -1 is interpreted as infinity. This parameter is ignored unless gen_ellipse_type == mixed.

C.5.22. ***gen_apriori_standard_error***

<double> [Default = 1.0000] (0.0000 <= X <= 1000.0)
The a priori standard error scale factor. It represents an estimate of the ratio between the true and actual data standard errors. This parameter only applies when K > 0.

C.5.23. ***gen_confidence_level***

<double> [Default = 0.9500] (0.0000 <= X <= 1.0000)
Uncertainty confidence level desired.

C.5.24. ***allowCorePhaseRenamingP***

<boolean> [Default = false]
If corePhaseRenaming is true then this is the distance in degrees beyond which corePhaseRenaming will take place.

C.5.25. ***corePhaseRenamingThresholdDistanceP***

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

C.5.26. ***useSimplex***

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

4.1. Correlated Observation Parameters

C.5.27. *gen_correlation_matrix_method*

<string> [Default = uncorrelated] (uncorrelated | file | function)
Specifies how correlation coefficients are to be specified. If 'file', then correlation coefficients between pairs of observations are read in from a file, with the file name being specified with the gen_correlation_matrix_file parameter. If value = 'function', then function parameters must be specified using the gen_correlation_scale parameter. Regardless of how the correlation coefficients are entered, they are used to populate an N x N matrix where N is the number of observations. The correlation matrix will have ones on the diagonal and user specified values between -1 and 1 in the off-diagonal elements. The diagonal elements of this matrix will be multiplied by the square of the total uncertainty of each observation (combined model and pick error). The off-diagonal elements are multiplied by the product of the model errors for the two observations (pick error is not included for off-diagonal elements).

C.5.28. *gen_correlation_matrix_file*

<string> [Default =] (any valid file path+name)
Specifies the name of the file from which correlation coefficients are to be read. The file consists of an arbitrary number of lines, each of which defines the correlation coefficient between two observations. Each line must contain the following information:
Station_1/phase_1/type_1 station_2/phase_2/type_2 corr_coeff
Where type_1 and type_2 are the observation types (TT, AZ or SH for travel time, azimuth and slowness), and corr_coeff is the correlation coefficient (-1 <= corr_coeff <= 1). For example,
ARCES/P/TT FINES/P/TT 0.5
would specify a correlation coefficient of 0.5 between all P wave travel time observations from ARCES and FINES. Correlation coefficients between pairs of observations which are not specified in the file are set to zero.

C.5.29. *gen_correlation_scale*

<double> [Default = 10 degrees] (> 0 degrees)
Specifies the correlation scale length for calculating the correlation coefficients between pairs of observations (degrees).
If gen_correlation_matrix_method is 'function' then the correlation coefficients between pairs of stations are computed from

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

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

C.6. Levenberg-Marquardt Non-Linear Least Squares Solver

C.6.1. *lsq_print_iteration_table*

<bool> [Default = true]
If false, iteration tables are not sent to general output.

C.6.2. *lsq_convergence_n*

<int> [Default = 2] (1 <= X <= 100000)
Number of consecutive times convergence criterion must be satisfied before convergence is declared.

C.6.3. *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.6.4. *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.6.5. *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.6.6. *lsq_damping_factor*

<double> [Default = -1.000] (-1.000 <= X <= 1e6)
Damping factor to be applied to singular values. For AUTOMATIC DAMPING: lsq_damping_factor = -1 [DEFAULT]. This factor controls application of the Levenberg-Marquardt algorithm which helps the locator converge if it is oscillating around the optimum location. Set to -1.0 to have the locator automatically adjust the damping factor as necessary. Set to 0.0 or positive values to override the default behavior.

C.6.7. *lsq_initial_applied_damping*

<double> [Default = 0.0001] (0.0000 <= X <= 1e6)
When AUTOMATIC DAMPING is being applied, this is the initial damping factor applied when an increase in the sum squared weighted residuals is observed.

C.6.8. *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.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*

<bool> [Default = true]
Echo iteration progress and/or messages to the screen during the location calculation. This is the same information that is sent to the output text file.

C.7.4. *io_error_file*

<string> [Default = locoo_errors.txt]
Full path to general output error file. All error messages generated during locoo execution are sent to this file.

C.7.5. *io_print_errors_to_screen*

<bool> [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]
Whether or not to print the iteration table when io_verbosity >= 4.

C.7.9. *io_nondefining_residuals*

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

C.8. Gridded Residuals

C.8.1. *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.8.2. *grid_output_file_format*

<string> [Default = tecplot] (tecplot)
The output file format. The only currently supported output format is tecplot. The format is pretty self-explanatory. Other formats may be supported in the future. Requests to support other formats will be considered.
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.8.3. *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.8.4. *grid_origin_lat*

<double> [Default = none]

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

C.8.5. *grid_origin_lon*

<double> [Default = none]
Longitude in degrees of the center of the grid. Ignored if grid_origin_source equals either epicenter of hypocenter.

C.8.6. *grid_origin_depth*

<double> [Default = 0. km]
Depth in km of the center of the grid. Ignored if grid_origin_source equals either epicenter of hypocenter.

C.8.7. *grid_map_units*

<string> [Default = degrees] (degrees or km)
Map width and height will be interpreted with these units. Also controls the units of the output.

C.8.8. *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.8.9. *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.8.10. *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.8.11. *grid_map_nwidth*

<int> [Default = none]
Map will have this many nodes in the x direction.

C.8.12. *grid_map_nheight*

<int> [Default = none]
Map will have this many nodes in the y direction.

C.8.13. *grid_map_ndepth*

<int> [Default = 1]
Map will have this many nodes in the z direction.

C.9. DataLoader Utility

C.9.1. *dataLoaderType*

<string> [Default = None] (file | oracle)
Specifies whether to perform IO with text files or with an Oracle database. For descriptions of parameters related to file IO see section DataFileLoader Utility. For descriptions of parameters related to database IO see section DBIO Utility.

C.10. DataFileLoader Utility

C.10.1. *dataLoaderFileOrigins*

<string> [Default = None]
Name of file containing the input origins. Required.

C.10.2. *dataLoaderFileAssocs*

<string> [Default = None]
Name of file containing the input assocs. Required.

C.10.3. *dataLoaderFileArrivals*

<string> [Default = None]
Name of file containing the input arrivals. Required.

C.10.4. *dataLoaderFileSites*

<string> [Default = None]
Name of file containing the input sites. Required.

C.10.5. *dataLoaderFileOrids*

<string> [Default = None]
A list of orids to process. Optional. If not specified, then all origins in the file are processed.

C.10.6. *dataLoaderFileOutputOrigins*

<string> [Default = None]
Name of file to receive the output origins.

C.10.7. *dataLoaderFileOutputOrigerrs*

<string> [Default = None]
Name of file to receive the output origerrs.

C.10.8. *dataLoaderFileOutputAssocs*

<string> [Default = None]
Name of file to receive the output assocs.

C.10.9. *dataLoaderFileOutputAzgaps*

<string> [Default = None]
Name of file to receive the output azgaps.

C.10.10. *dataLoaderFileTokenDelimiter*

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

C.10.11. *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 was necessary since the input tokenDelimiter is read from a properties file.

C.11. DBIO Utility

C.11.1. *dbInputUserName*

<string>   [Default = user's environment variable DBTOOLS_USERNAME]
Database input account usernames.

C.11.2. *dbOutputUserName*

<string>   [Default = none]
Database output account usernames. If not specified, no output is written to the database.

C.11.3. *dbInputPassword, dbOutputPassword*

<string>   [Default = user's environment variable DBTOOLS_PASSWORD or UserName]
Database input/output account passwords. If not specified in the property file, and the property DBTOOLS_PASSWORD is specified in the user's environment then the value from the environment is used. If not specified in either the property file or the user's environment then the value of dbInputUsername/dbOutputUserName is used.

C.11.4. *dbInputInstance, dbOutputInstance*

<string>   [Default = user's environment variable DBTOOLS_INSTANCE]
Database instance for input/output.

C.11.5. *dbInputDriver, dbOutputDriver*

<string>   [Default = user's environment variable DBTOOLS_DRIVER, or oracle.jdbc.driver.OracleDriver]
Database driver for input/output. Generally equals oracle.jdbc.driver.OracleDriver.

C.11.6. *dbInputTableTypes*

<string>   [Default =]

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

C.11.7. *dbInputTablePrefix*

<string> [Default none]
If this parameter is specified then the four input tables (dbInputOriginTable, dbInputAssocTable, dbInputArrivalTable, dbInputSiteTable) will default to the value of this parameter with the appropriate table type (ORIGIN, ASSOC, ARRIVAL, SITE) appended on the end. If any of the four tables are also explicitly specified, then the explicitly specified name has precedence.

C.11.8. *dbInputOriginTable*

<string> [Default not allowed]
Name of the input origin table. Specifying this parameter will override any default values set by other parameters.

C.11.9. *dbInputAssocTable*

<string> [Default not allowed]
Name of the input assoc table. Specifying this parameter will override any default values set by other parameters.

C.11.10. *dbInputArrivalTable*

<string> [Default not allowed]
Name of the input arrival table. Specifying this parameter will override any default values set by other parameters.

C.11.11. *dbInputSiteTable*

<string> [Default not allowed]
Name of the input site table. Specifying this parameter will override any default values set by other parameters.

C.11.12. *dbInputWhereClause*

<string> [No default value]
An orid query "where" clause that specifies the origins that should be processed as input for LocOO. This where clause is executed against the origin table.
LocOO3D executes the following sql statement in order to load data from the database:
First, it executes something similar to:
select orid, ndef from <*dbInputOriginTable*> where <*dbInputWhereClause*> order by ndef desc
With the output, it forms the orids into batches where each batch will have roughly the same number of defining phases and the batches tend to be in order increasing number of orids. The number of defining phases per batch can be specified with property *batchSizeNdef*, which defaults to 1000. This arrangement is most efficient when processing predictions in parallel. Orids that have more defining phases than *batchSizeNdef* will form their own batch. Once all of those origins have been formed into batches, other batches will have no more than *batchSizeNdef* defining phases in them. No batch will have more than 1000 orids in it due to oracle limitations.
For each batch of orids generated during step 1, LocOO executes 2 sql statements like

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

C.11.13. *dbInputAssocClause*

<string> [empty string]
An optional phrase that will be appended onto the end of the where clause that selects rows from the assoc, arrival and site tables. See dbInputWhereClause for details of how this is used.
For example, to include only data from stations within distance of 40 degrees from the origin, specify dbInputAssocClause = assoc.delta < 40. To limit the data to include only stations ABC and XYZ, specify dbInputAssocClause = site.sta in ('ABC', 'XYZ'). This where clause becomes part of a sql statement that is executed against an assoc, arrival and site table.

C.11.14. *batchSizeNdef*

<int> [1000]
The approximate number of defining phases that will be included in each batch of origins that will be loaded and processed. See *dbInputWhereClause* for more details.

C.11.15. *dbOutputTablePrefix*

<string> [Default =]
If this parameter is specified then the output table types specified with the dbOutputTableTypes parameter will default to the value of this parameter with the appropriate table type appended to the end.

C.11.16. *dbOutputTableTypes*

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

C.11.17. *dbOutputOriginTable*

<string> [Default = none]
Name of the origin table where output is to be written. Specifying this parameter will override any default values set by other parameters.
In general, the value of orid in the output origin table will be set to a value that is unique in the output table. However, if parameter *dbOutputConstantOrid* is set to true, then the orid value from the input origin row is retained.
Note that evids are never changed by LocOO; evid in the output origin row will be equal to evid in the input origin row.

C.11.18. *dbOutputArrivalTable*

<string> [Default = none]

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

C.11.19. *dbOutputAssocTable*

<string> [Default = none]
Name of the assoc table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.20. *dbOutputAzgapTable*

<string> [Default = none]
Name of the azgap table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.21. *dbOutputOrigerrTable*

<string> [Default = none]
Name of the origerr table where output is to be written. Specifying this parameter will override any default values set by other parameters.

C.11.22. *dbOutputAuthor*

<string> [Default = ‘-‘]
Name of the output author. This is used to populate the AUTH field of the new origin row.

C.11.23. *dbOutputConstantOrid*

<bool> [Default = false]
If true, then the value of orid in the output table will be unchanged from the value in the input origin table. If false, the value of orid will be set to a value that is unique in the output origin table.

C.11.24. *dbOutputAutoTableCreation*

<bool> [Default = false]
Boolean flag should be set to true if output database tables should be created if they do not already exist.

C.11.25. *dbOutputTruncateTables*

<bool> [Default = false]
Boolean flag should be set to true if output database tables should be automatically truncated at the start of the run. Unless the *dbOutputPromptBeforeTruncate* parameter has been set to false, the user will be prompted before table truncation actually occurs.

C.11.26. *dbOutputPromptBeforeTruncate*

<bool> [Default = true]
If *dbOutputTruncateTables* is true and this parameter is true, then the user is prompted before output table truncation actually occurs. If *dbOutputTruncateTables* is true and this parameter is false, table truncation occurs without warning.

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
Technical Library	01177	libref@sandia.gov

This page left blank

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.