Antelope New User's Guide v1.2
(March 2011)

Jeff Lockridge
School of Earth and Space Exploration
Arizona State University

Introduction

This guide is intended to be a basic introduction to *Antelope* for new and inexperienced users. This guide is not all-inclusive, but it will provide basic instruction on how to begin developing an earthquake catalog with *Antelope*. By default, most *Antelope* parameters are set to deal with large teleseismic and regional earthquakes. This guide provides examples of parameters set to analyze small magnitude local events with station spacing similar to the USArray Transportable Array (70 km).

This guide assumes that the user is starting with a newly installed version of *Antelope* (v. 4.11 or later), and a complete, working *Antelope* database provided by ANF. All commands in this guide are written exactly as they should be typed into the terminal, except the user should replace *"dbname"* with the name of their main *Antelope* database. Most commands will be performed within the *dbname/* directory, which is equivalent the directory that contains all of your database files.

Additional information can always be found by viewing the MAN pages or performing searches on the FAQ at www.brtt.com.

Thanks to Kevin Eagar, Matt Fouch, and especially Jennifer Eakins for passing along their knowledge and patience. Please contact me at lockridge [at] asu.edu with any comments, corrections and/or suggestions.

Table of Contents

[1.0] Directory Structure Setup
[1.1] Backup
[1.2] Moving Static Tables
[1.3] Changing *.wfdisc* Paths
[1.4] Descriptor File
[1.5] Parameter Files and *DBPATH*
[1.6] Create a Sub-Database
[1.7] Understanding Your Database (*dbe*)

[2.0] Intro to *dbpick*
[2.1] Resource File (*.dbpickrc*)
[2.2] Opening *dbpick*
[2.3] Navigation
[2.4] Menu Buttons
[2.5] Picking Arrivals
[2.6] Command Line Operations

[3.0] Intro to *dbloc2*
[3.1] Setting up *dbevproc*
[3.2] *dbloc2* Parameter File (*dbloc2.pf*)
[3.3] Using *dbloc2*
[3.4] Velocity Models

[4.0] Database Processing
[4.1] *dbdetect*
[4.2] *ttgrid*
[4.3] *dbgrassoc*
[4.4] Developing Your Catalog

[5.0] Database Surgery/Troubleshooting/Misc
[5.1] Adding Waveform Data
[5.2] Replacing Bad Data
[5.3] Fixing Multi-Station Data Dropout

[1.0] Directory Structure Setup

To keep things organized, establish a db directory structure before any new files are created. The db should arrive with some or all of the following files in the *dbname/* directory:

a) Descriptor file (*dbname* without extension), *dbname.calibration, dbname.dlsensor, dbname.instrument, dbname.network, dbname.schanloc, dbname.sensor, dbname.sensormodel, dbname.site, dbname.sitechan, dbname.stage, dbname.wfdisc*

b) Sub-directory containing waveform data (*certified/*). This directory name may vary depending on the origin of your database (example: *seed/*).
Data in this directory should be arranged by year and Julian day. An example path and waveform filename:

certified/2006/046/Y12C.2006:045:00:00:00

Note: If your database does not appear to have this format, refer to these guides:
http://www.iris.edu/hq/es_course/content/friday/database2locate_CaseStudy3.pdf
http://www.aeic.alaska.edu/input/mgardine/*Antelope*_guide.html

c) Sub-directories *nom-response/* and *response/* containing instrument response info.

d) Your db may have several "*dbname.*_____" files that are not listed above. Assuming you are using a brand new db and no *Antelope* functions have been run yet, any other "*dbname.*_____" files are likely be static db tables similar to those listed in Part a.

[1.1] Backup

It is convenient to have a backup copy of the original db tables on the local disk too. Create a *backup/* directory and then copy all of the original db files (excluding those in the *certified/* directory) into it.

```
% cp dbname* backup/
% cp *response backup/
```

You may also want to create a backup script that copies your dynamic database files into the backup folder on a regular basis.

[1.2] Moving Static Tables

Within the *dbname/* directory, create another sub-directory called *dbmaster/*. This directory will hold all of the static db tables (i.e. the ones we will not be changing). Move the *response/* and *nom-response/* directories and all of the *dbname*.* files (exclude the descriptor file *dbname*) into the *dbmaster/* directory.

```
% mv dbname.* dbmaster/
% mv *response dbmaster/
```

After the move, your home *dbname/* directory should only contain:

dbname backup/ certified/ dbmaster/

Now is a good time to create some additional directories that will keep things organized in the future. The following should also be sub-directories within your *dbname/* directory:

bin/ Stores various scripts (*rundbevproc*, backup script, etc).

db/ Contains sub-directories to store any supplementary or test databases other than your main *dbname* database (more on this in Section 1.6).

grids/ Stores travel time grids (Section 4.2)

pf/ Stores customized *Antelope* parameter files.

The *dbname/* directory should now contain the following files and directories:

dbname backup/ bin/ certified/ db/ dbmaster/ grids/ pf/

While using this database structure, all *Antelope* commands should be executed from the *dbname/* directory. Also, any dynamic db tables created for the *dbname* database will reside in the home *dbname/* directory, while any files associated with other sub-databases will be located in separate sub-directories within the *db/* directory (See Section 1.6).

[1.3] Changing .wfdisc Paths

The *dbname.wfdisc* file tells *Antelope* where to look for the waveform data. We just moved the *.wfdisc* file, so *Antelope* will no longer be looking for the data in the correct directory. To change the path within the *.wfdisc* file, run the following command from your *dbname/* directory:

```
% dbset -cv dbmaster/dbname.wfdisc dir '*' 'patsub(dir,"certified/*","../$0")'
```

dbset will ask you to confirm the change to each entry in the *.wfdisc* table, so this is your chance to double check that the path is being changed to the correct location (Example: ../certified/2006/046). If the new path looks incorrect, then Ctrl+C out of *dbset* and make any corrections. If the new path looks correct, you can bypass pressing "y" for every *.wfdisc* entry by typing "all" when it asks if you are sure. *dbset* may take a while to finish, depending on how big your db is. When it is finished, you can quickly verify that your *.wfdisc* file is correct by checking the path listed the output of:

```
% head dbmaster/dbname.wfdisc
```

[1.4] Descriptor File

In *Antelope*, each db is defined by a descriptor file. The filename for a descriptor file is always the database name with no file extensions (Example: *dbname*). The descriptor file is used to define a schema, place locks on the db, and to tell *Antelope* where to look for any files associated with the db.

For this guide, we will only edit the *dbpath* line of descriptor files. Since the db files have recently been moved, the main db descriptor file *dbname* needs to be edited to include each new path. Use a text editor like *vi* to change the descriptor file to look like the one below:

```
% vi dbname

        #
        schema css3.0
        dblocks none
        dbidserver
        dbpath ./{dbname}:dbmaster/{dbname}
```

#NOTE: (Be sure to include a blank return line at the end of the file or else you will have problems)

The *dbpath* line tells *Antelope* to look within the specified directories for files belonging to whichever database name is found between the {} symbols. In this example, *Antelope* will recognize any *dbname* files within the same directory as the descriptor file (*dbname/*) or within the *dbmaster/* sub-directory. Additional directories can be used by adding another ":filepath/{dbname}" to this line (See Section 1.6).

Once the descriptor file has been edited, run *dbpick* from the *dbname/* directory to ensure that the database structure has been set up correctly:

```
% dbpick dbname
```

If you are unfamiliar with *dbpick* navigation controls, you can type "fw" into the *dbpick* command line or "Shift+F" in the *dbpick* window to skip ahead to the first waveform. If the waveform data is still not visible after using the scroll bar on the left to look through all traces; first, make sure that the *dbpath* line in the *dbname* descriptor files contains the correct paths (separated by a ":"), and second, re-check to be sure that the *.wfdisc* file is referring to the correct directory and use *dbset* to make any corrections. Exit *dbpick* by typing "quit" into the command line.

[1.5] Parameter Files and DBPATH

The MAN page for each *Antelope* program lists the name of its parameter file and how to edit the various parameters within. Original copies of parameter files are located in the *$ANTELOPE/data/pf/* directory, however, files in this directory should never be edited or deleted. From now on, any time you need to customize a parameter file, make a copy from the *$ANTELOPE/data/pf/* directory and edit a copy within your *dbname/pf/* directory.

Antelope uses the environment variable *PFPATH* to identify directories in which it will look for parameter files. This variable is stored within the *.cshrc* file, located within in your User home directory (the "." at the beginning of the file name indicates that it is a hidden file). Navigate to your User home directory and open your *.cshrc* file with a text editor, such as *vi*. Once opened, navigate down to the section labeled with the header:

```
        ### Antelope ###
```

You should see a line containing the environment variable *PFPATH*:

```
        setenv PFPATH $ANTELOPE/data/pf:.
```

This portion of the *PFPATH* line should not be edited, but we can add as many paths as desired to the end of the line by separating each with a ":" symbol, for example:

```
setenv PFPATH $ANTELOPE/data/pf:.:./pf
```

In the above line, *Antelope* will now look in a *pf/* sub-directory located within the current working directory.

If a parameter file exists in two or more of the specified directories, *Antelope* will always choose the parameter file located in the rightmost directory on the list. This means that when working in the *dbname/* directory, the *PFPATH* variable will tell *Antelope* to find parameter files within the *pf/* directory first, the current working directory second, and the *$ANTELOPE/data/pf/* directory third.

When finished, source the *.cshrc* file with the following command:

```
% source ~/.cshrc
```

If you are ever unsure which version of a *.pf* file antelope is using, the *pfwhich* command will list the locations of any*.pf* file that is typed in (the one *Antelope* uses is returned at the bottom of the list):

```
%pfwhich dbloc2.pf
```

[1.6] Create a Sub-Database

In order to keep the *dbname* database nice and clean while learning *Antelope*, create a separate database just for practice. Here we will use nice generic names like "test1" for our dbs. So from your *dbname/* directory, create a new directory called *test1/* as a sub-directory of *db/* :

```
% mkdir db/test1
```

Antelope recognizes dbs by their descriptor file, so copy the *dbname* descriptor file and edit it to apply to the new *test1* db:

```
% cp dbname db/test1/test1
% vi db/test1/test1
```

The *test1* descriptor file should start off identical to the *dbname* file (except for the filename). Edit the *dbpath* line to modify the directories and change the name of the database:

```
dbpath ./{test1}:../../dbmaster/{dbname}
```

Note that *Antelope* will look (1st) in the *test1/* directory for all *test1* database files, and (2nd) up two directories (to the *dbname/* directory) and down into the *dbmaster/* directory for any *dbname* files. Entry #2 will allow the *test1* db to use the static tables and waveforms of the *dbname* database without writing to its dynamic tables.

[1.7] Understanding Your Database (*dbe*)

The *dbe* program is a useful tool for viewing, editing and exporting *Antelope* database tables. There are other useful commands for editing large portions of databases (*dbset*, *dbjoin*, etc.), but *dbe* seems to be a more simple choice for making small changes. For now, open the *test1* database with *dbe* and have a look at how it is set up.

```
% dbe db/test1/test1
```

When *dbe* is first opened, it appears as a long, horizontal window with a button for every table in the database. Clicking on any button will open that table in a new window. Note that the database is simply a series of tab-delimited text files. The most useful functions in *dbe* allow the user to sort, subset, export, and edit tables. By default, *dbe* will not allow changes to any tables unless *Allow edits* is selected from the *Options* menu, so it is impossible to accidentally edit the tables until that option is selected. Click around for a bit and explore the GUI.

Viewing All Columns
When a table is opened in dbe, only some of the columns will be visible. To view all (or more) columns, select *View -> Arrange* and check the boxes of the columns you wish to view.

Sorting Tables in *dbe*
Open any db table, left-click on any column header and choose *Sort* from the drop-down menu. This sorts the selected column in alphanumeric order.

Creating a Subset
To create a data subset, select a table entry, left-click on the column header and choose *Subset* from the drop-down menu. This will create a subset of all rows that have the selected text in that column.

In *dbe*, additional rows can be deleted from view by using the *Edit -> Delete…* commands. Multiple rows of data can be selected by holding down the Shift key.

Exporting Tables
The *File -> Save…* option in *dbe* will export all or a portion of a database table to a plain text file. Try this out by creating a subset of a table, choosing *File -> Save…* and entering a filename into the text box. There are other useful options here too, such as headers, and tab separated text.

Editing Database Tables
In general, it is not a good idea to edit static database tables, so you may want to wait until you have some dynamic tables before editing any tables in *dbe*. However, if you have recently saved backup copies of your tables and are able to easily restore them, you can allow tables to be edited by selecting *Options -> Allow Edits* from the menu at the top of the table window. Once edits are enabled:

1. Select the field(s) you wish to edit. (Remember: Shift + Drag/Click highlights multiple rows)
2. Change the cell value in the text box at the top of the window.
3. Left-click on the column header button and choose *Set value* from the menu. This will change the value of each selected cell to the value entered in the text box.
4. Entire rows can be deleted by choosing *Edit -> Delete*.
5. When finished editing, uncheck the *Options -> Allow Edits* again to lock the table for editing.

Crunch Table

If an entire row is deleted, *Antelope* will leave a blank line in place of the deleted row. If many rows in a table are deleted the *Edit -> Crunch Table* option will remove all blank lines. This can also be done on the command line for an entire database or a single table at a time:

```
% dbcrunch dbname.table
```

[2.0] Intro to *dbpick*

This guide will include a basic rundown of some tips and tricks, but it is best to learn by experimentation. A *dbpick* tutorial with a practice database can be found at:

http://www.indiana.edu/~aug/g515/lesson1/lesson1.html

Otherwise, the *dbpick* MAN page is a great reference. It is also good to keep in mind that you can only add/change/delete arrivals while viewing waveforms in *dbpick*, so there is no need to worry about deleting any waveform data by accident. If *dbpick* begins to act in unexpected ways, just quit and re-launch. Don't worry about losing any work, as the db tables are updated as soon as any changes are made in *dbpick*.

[2.1] Resource File (*.dbpickrc*)

The resource file for *dbpick* is named *.dbpickrc*. This file customizes many of *dbpick*'s settings, including display options, waveform filters, phase labels and more. *dbpick* looks for *.dbpickrc* in the current working directory first, in the User home directory second, and in the default *Antelope* pf directory *($ANTELOPE/data/pf/*) third. A sample copy of a *.dbpickrc* file from ANF can be viewed here.

For this guide, simply copy the *.dbpickrc* file to the *dbname/* directory and edit it there:

```
% cp $ANTELOPE/data/pf/.dbpickrc .
% vi .dbpickrc
```

The main difference between the original *.dbpickrc* file and the ANF version is that the ANF version has added waveform filters and phase names to the file. Once all of the changes have been made, *Antelope* will use the new version of the *.dbpickrc* file every time *dbpick* is run from the *dbname/*directory. This also means that if *dbpick* is run from a different directory, the customized *dbpick* options will not be available. If you plan to run *dbpick* from directories other than your *dbname/* directory, it may be a better option to copy the *.dbpickrc* file to your User's home directory. As always, the *dbpick* MAN page gives a full list of editing options for the resource file.

[2.2] Opening *dbpick*

The *dbpick* MAN page lists the many command line options available when opening *dbpick*, but it is often easiest to just load the entire db and navigate from within *dbpick*. By opening *dbpick* without any command line flags, all waveform data from your database will be available for view. Remember that the *test1* database is located in a different directory, so path and filename must be included in the command line:

```
% dbpick db/test1/test1
```

Each time *dbpick* loads it will ask if you want to view the waveforms. This can be annoying (why else would you be opening *dbpick*?), so one way around this is to create an alias for *dbpick* in the *.cshrc* file that is located in the User's home directory. The alias entered into the *.cshrc* file should contain the *-nostarttalk* flag:

```
alias dbpick    'dbpick -nostarttalk'  #this will eliminate the "y/n" prompt at startup
```

After closing the file and sourcing the *.cshrc* file, *dbpick* will open without getting hung up on the y/n prompt:

```
% source ~/.cshrc
```

If *dbpick* seems to be taking a while to load waveforms at startup, try adjusting the amount of time displayed in the default waveform window. This can be done in the *.dbpickrc* file (see *dbpick* MAN page), or the changes can be made in the *dbpick* alias line of the User's *.cshrc* file:

```
alias dbpick    'dbpick -nostarttalk -tw 3600'  #no y/n prompt, time window of 3600 sec (1hr).
```

[2.3] Navigation

The *dbpick* MAN page gives a great explanation of navigation, so below is a quick summary of navigation with some extra tips and suggestions.

In the waveform window:

Left Click	Scroll time forward - Takes the time you clicked and moves it to the left edge of waveform window.
Right Click	Scroll time backward - Takes time you clicked and moves left edge of *dbpick* window to that spot.
Middle Click + Drag	Drag waveforms along time axis
Shift + Left Click + R/L Click	Time axis zoom in (wider yellow region = less zoom in)
Shift + Right Click + R/L Click	Time axis zoom out (wider region yellow = less zoom out)
Ctrl + Left Click + R/L Click	Trace axis zoom in (wider yellow region = less zoom in)
Ctrl + Right Click + R/L Click	Trace axis zoom out (wider region yellow = less zoom out)
Middle Click	Cancels any zoom action you started (Example: Shift + Left Click, slide mouse and Middle Click = cancel zoom before second L/R Click).

You can also navigate by using the scroll bars on the left (trace axis) and bottom (time axis). A right click on either scroll bar will give you additional options (*Fit, ZoomIn, ZoomOut, Undo*). The *Fit* option on the bottom (time) scroll bar will probably crash the *dbpick* session, as it will try to fit the entire time range of entire database onto the current display window. The *Fit* option on the left (traces) scroll bar will zoom out to display all available traces up to a maximum number of 100.

[2.4] Menu Buttons

The menu buttons at the top of the *dbpick* window are reviewed in detail within the *dbpick* MAN page.

Traces – This button selects and modifies which traces are being displayed. Click on any trace name to select it, and click on it again to de-select.

Select	Displays only the selected traces in order of selection (numbers left of trace name).
Delete	Removes selected traces from display.

Original	Displays all traces loaded into *dbpick* at startup. 100 traces is max to display on one window without use of scroll bar.
SelectAll	Selects all traces.
DeleteAll	Unselects all traces. This option will not remove any traces from view.
Zoom	Displays top and bottom selected traces, plus all traces in between.
New Win	New window displaying selected traces. All changes are synchronized among open windows. N**ote**: Any time you close a new window, be sure to close it using the
Done	button, as the [X] at the top right of the window will close the entire *dbpick* session.

AMP – This tab changes the amplitude scale for every trace in the display. Depending on how you are using *dbpick*, you may find different options useful. For me, *Fixed* and *Auto* worked just fine.

Filter – This tab controls how *dbpick* filters the waveform data before it is displayed. Experience is the best teacher for knowing which filter to use for your project. The filter parameters can be modified in the *.dbpickrc* resource file. The *dbpick* MAN page gives an example of how to add filters to the *.dbpickrc* file, or you can refer to this sample from ANF.

Add Arrivals – Places the mouse pointer in add-arrival mode. In this mode:

Left Click	Adds a single arrival and returns mouse pointer to normal mode.
Right Click	Adds an arrival and keeps mouse in add-arrival mode (good for making many picks).
Middle Click	Cancels add-arrival mode and returns mouse to normal mode.

More on picking arrivals in Section 2.5 below.

Add Time Marks – Places mouse in add-timemark mode. Timemarks are vertical time-reference lines and are only visible during the current session of *dbpick*. Timemarks are not stored in a database and disappear when *dbpick* is restarted. Left Click + Drag moves the time mark, and Shift + Left Click deletes a time mark.

[2.5] Picking Arrivals

Click on the "Add Arrivals" button at the top of the *dbpick* window to place the mouse in add-arrival mode. In this mode:

Left Click	Adds a single arrival and returns mouse pointer to normal mode.
Right Click	Adds an arrival and keeps mouse in add-arrival mode (until Left or Middle Click).
Middle Click	Cancels add-arrival mode and returns mouse to normal mode.

Once an arrival has been added and the mouse is no longer in add-arrival mode, the arrival flag will probably be marked "U" for unknown. The default phase can be changed with the command line option "ph" (example in Section 2.6). The following occur when clicking on the arrival flag:

Left Click + Drag	Slides location of arrival flag until button is released.
Right Click	Brings up "Window", "Magnify" and "Delete" menu.
Window	Opens single trace w/arrival in new window. Close window by pressing "Done".

Magnify	Creates new time-magnified trace window. Close with "Done".
Delete	Permanently deletes arrival from tables.
Middle Click	Choose phase from a drop down menu (P, S, Pn, etc). Choices on this menu can be edited in the *.dbpickrc* file (See example here).
Shift + Left + Drag	Sets arrival time uncertainty window (area within yellow box). This option gives the arrival a weight when locating, depending on size of error box.
Ctrl+Shift+Left+Drag	Changes arrival period and amplitude (see *dbpick* MAN page).
Shift + Middle Click	Allows you to set the first motion direction and override *dbpick*'s choice.

Take some time to practice navigation in *dbpick* and pick 8 to 10 first arrivals for several events in your test1 database. Remember to only place P phase picks on the vertical (Z) components, and only place S phases on horizontal components (N or E). These picks will be used during the *dbloc2* tutorial in Section 3.

After picking several arrivals, view the contents of the *db/test1* directory to note that *dbpick* has created the test1.arrival and test1.lastid tables:

```
% ls db/test1/

    test1    test1.arrival    test1.lastid
```

[2.6] Command Line Operations

Any time *dbpick* is open, type "help" into the command line to see a list of commands, associated actions and shortcut keys. All command line options are listed in the *dbpick* MAN page. I'll list some of the most useful commands as examples here:

tw 120	change time window to two minutes (in seconds)
sc T15A:*	displays all channels available for station T15A
sc *:BHZ	displays all stations, vertical channel only
sc [XYZ]1[567]A:BH.	displays all X,Y,Z 15A-17A stations, and all BH_ channels
sc .14A:*	displays all stations all _14A stations, all channels
cm 21	changes display to show only 21 traces at a time
ph P	sets default phase to "P" when selecting arrivals
ph Sn	sets default phase to "Sn" when selecting arrivals
sd on	turns detection on. "sd off" turns them off
dw	hides traces with no waveform data
ts 2008:123:00:00:00	skips to time. Format is year:julday:hr:min:sec
fw	skips to first waveform

[3.0] Intro to *dbloc2*

Once you have used *dbpick* to pick arrivals for multiple events, *dbloc2* can be used to locate event hypocenter. There are a couple of very helpful *dbloc2* tutorials available online:

http://anf.ucsd.edu/faq/operations/dbloc2/
http://www.indiana.edu/~aug/g515/lesson6/dbloc2_tutorial.html

This guide will go over how to set up an alternative magnitude calculator, edit parameter files, and provide some tips for using *dbloc2*. For a detailed intro to using *dbloc2* refer to the above links.

[3.1] Setting Up *dbevproc*

Before jumping into *dbloc2*, it is a good idea to set up *dbevproc* as the primary magnitude calculator. The default *dbloc2* magnitude calculator in *Antelope* 4.11 is *dbml*; however, this method is considered to be deprecated by *Antelope*'s developers and should be avoided.

To run *dbevproc*, use the parameter file from the real-time *orbevproc* program (since *dbevproc* and *orbevproc* are interchangeable). From the *dbname/* directory, copy over the *orbevproc.pf* parameter file located in the *$ANTELOPE/data/pf/* directory, name the copy "*dbevproc.pf*", and place it in the *dbname/pf/* directory:

```
% cp $ANTELOPE/data/pf/orbevproc.pf ./pf/dbevproc.pf
% vi pf/dbevproc.pf
```

Notice how the *dbevproc.pf* file is setup; the top table lists the available *event_processes*, while the bottom two tables define the parameters for the *Mlrichter* and *Mbusneic* processes. A complete explanation of each parameter is available at http://www.brtt.com/wiki/Mlrichter%283p%29. Below, the original version of the line is listed as a comment ("#" in front), with the example new line below it. These changes should be made in both the *Mlrichter* and *Mbusneic* table.

auth_accept

This option is matched to the *origin.author* field for each event before processing. If you are having trouble getting *dbevproc* to produce magnitudes, then double-check to make sure the author field in the *.origin* table matches this field. In the example below, any origin that has an author ID beginning with "db" or "ASU" will be accepted. The "db" will match any events located by *dbgrassoc* or *dbgenloc*, while the "ASU" matches the *Institution* field specified in the *dbloc2.pf* file (we will edit this in Section 3.2).

```
#auth_accept        oa_.|oa_. dbg
auth_accept         db.*|ASU.*
```

output_auth

This parameter describes which method was used to calculate the magnitude of an event:

```
# output_auth       orbevproc
output_auth         evproc
```

output_wfmeas

Changing this parameter from "yes" to "no" keeps *Antelope* from returning *output_wfmeas* entries for every station that contributed to the final network magnitude calculation.

Next, set up a short shell script that will run *dbevproc* as the magnitude calculator whenever you press the *Magnitudes* button in *dbloc2*. From the *dbname/* directory, create the following script within your *bin/* sub-directory by using vi:

```
% vi bin/rundbevproc

    #!/bin/sh
    dbsubset $1.origin "orid==$2" | dbevproc - $1
```

This script will be referenced in the *dbloc2* parameter file in Section 3.2 below. But first, change the file to an executable:

```
% chmod +x bin/rundbevproc
```

[3.2] *dbloc2* Parameter File (*dbloc2.pf*)

The parameter file for *dbloc2* is named *dbloc2.pf*, and an original copy is located in the *$ANTELOPE/data/pf/* directory. The *dbloc2* MAN page gives brief descriptions of what each table in the *.pf* file does. To customize the *dbloc2.pf* file, place a copy of it in the *pf/* directory:

```
% cp $ANTELOPE/data/pf/dbloc2.pf ./pf
% vi pf/dbloc2.pf
```

A copy of my edited *dbloc2.pf* file is available here. Below, the original version of the line is listed as a comment ("#" in front), with the example new line below it. Line numbers provided for reference.

Institution (Line 3)

```
#Institution      &ref(site,Institution)
Institution       ASU
```

By default, *dbloc2* assigns author to every origin using the format "institution:Username". The *author* field can only be a maximum of 15 characters, so if your login Username is long, you may want to use an abbreviated institution name. In the above example, the first line references an entry in the *site.pf* file, but it is much easier to just edit your institution directly into the *dbloc2.pf* file. This change will cause *dbloc2* to assign "ASU:Username" to every origin that is created by pressing the *Locate* button in *dbloc2*:

Note: As described in Section 3.1, any changes made to the *Institution* field will also need to be made to the *auth_accept* field in the *dbevproc.pf* file: (auth_accept db.*|ASU.*).

reference_db (Line 16)
This table associates your events to events already picked and located in other existing catalogs. To use this option, you will need to download event catalogs and reference them in this table. An example of how to do this for a real-time database is available here:

http://eqinfo.ucsd.edu/faq/antelope_retrieve.php

depth_list* and *starting_depth (Line 26)
These entries specify a list of starting depths, and the default starting depth used by the location algorithm. You may want to change the list and/or default starting depth if you are looking for earthquakes at a particular depth.

magnitude_calculators (Line 38)
The default *dbloc2* magnitude calculator in *Antelope* 4.11 is *dbml*; however, this method is considered to be deprecated by *Antelope*'s developers and should be avoided. Section 3.1 discussed how to set up *dbevproc* to calculate the local magnitude of events. Here, we reference the *rundbevproc* script that was placed in the *bin/* directory in Section 3.1:

```
magnitude_calculators  &Tbl{
#dbml -make_magtables
bin/rundbevproc
}
```

etype (Line 48)
Here you can edit the "etype" flag in *dbloc2*. This allows the user to create 1 or 2 character flags for each event that is reviewed. For our databases, I generally characterize the location stability by good (g), fair (f) or poor (p). Also, additional flags like mine blast (mi) or revisit (??) might be useful for your catalog. Default values are "qb eq me ex"

allow_direct_magnitude_calculations (Line 44)
Setting this option from default "no" to "yes" will reduce potential conflicts within *dbloc2* when calculating magnitude.

time_window (Line 68)
The default time window for *dbloc2* is set to 1500 seconds. If you are working on regional or local small-scale earthquakes or a series of aftershocks, you may want to change the default window to something much smaller so you do not have a multitude of events on the *dbpick* window at one time.

```
#time_window 1500     # time window = 1500 sec
time_window   200     # time window = 200 sec
```

coredumpsize (Line 147)
Setting this parameter from "unlimited" to "0" disables *dbloc2* saving files when a core dump occurs. It is up to you whether or not you want to make this change.

[3.3] Using DBLOC2

Use *dbloc2* to open the test1 database created in Section 2.0 above:

```
% dbloc2 db/test1/test1
```

The *dbloc2* window may get hung up when loading a db for the first time. If the arrivals you picked are not immediately displayed at the center of the *dbloc2* window and you do not see any stations or phases when it opens, simply choose *File -> Quit* from the menu at the top and re-open *dbloc2*. This may be a bug in *Antelope* 4.11 when using *dbloc2* with a new db.

Now, arrivals, stations and times should appear when *dbloc2* opens. Jon Tytell at ANF has already made a great *dbloc2* tutorial, so I will defer to his guide for descriptions of the windows, buttons, and main *dbloc2* functions.

http://anf.ucsd.edu/faq/operations/dbloc2/

Use the linked guide to help you locate some of the events with arrivals you picked in Section 2.0. Be sure to try out the magnitude calculator is well. Note that it generally takes a while (a little over 30 seconds on my system) for *rundbevproc* to return a magnitude value in *dbloc2*.

The following bullets are just some tips and tricks that I have found useful in *dbloc2*:

- Always be sure to save the events you want to keep. To do this, press the *Drop* button under the *Keep* field in the Event Origin Section until it changes to *Save*. When you close *dbloc2*, press the *Regroup* button, or press the *Save* button at the bottom of the *dbloc2* window the *Save* button in the Event Origin Section will automatically change to *Keep*, which means it has effectively been saved to your database tables.
- In most cases, a lower *sdobs* value means a more accurate location. However, for events with four or fewer arrivals, an extremely low *sdobs* value may suggest an inaccurate solution. Generally speaking, be skeptical of any locations that have an *sdobs* value less than 0.01 and keep an eye on how stable location solution is when one or two arrivals are slightly adjusted.
- Quitting a *dbpick* window opened by *dbloc2* may crash both *dbloc2* and *dbpick*. If you no longer want to see *dbpick* while in *dbloc2*, simply minimize the window.
- Always choose *File -> Quit* when closing *dbloc2*. Choosing *File -> Quit without saving* may leave a corrupt database.
- If you have many events to calculate magnitudes for, consider using running *dbevproc* for your whole database with the following command:

```
% dbevproc -p pf/dbevproc.pf -v db/test1/test1 db/test1/test1
```

- If *dbloc2* starts to return strange locations, or refuses to return a magnitude, the *Regroup* button at the top of the window will occasionally fix the problem.
- Sometimes, the *Next* or *Previous* buttons need to be pressed multiple times to move to the next or previous event.
- The *Previous* will move you to the last event viewed which is not necessarily the preceding event chronologically.
- If arrival flags picked in *dbpick* do not appear immediately in *dbloc2*, the *Regroup* button will refresh the window and update the display.
- *dbloc2* automatically creates a *dbname/tmp/* directory containing temporary files it uses during normal operations. Occasionally, these files may cause a conflict with one another. If *dbloc2* starts behaving in unexpected ways or returns repeated errors, try removing the *tmp/* directory

and its contents. This will force *dbloc2* to build a fresh *tmp/* directory and may solve your problem.

After closing *dbloc2*, view the contents of the *db/test1* directory to note that *dbloc2* has created several new tables:

```
% ls db/test1/

    New tables =   test1.assoc      test1.event      test1.origin
```

[3.4] Velocity Models

The default *dbloc2* velocity model (*iasp91*) may not be ideal for your study area. The *dbgenloc* MAN page gives a good summary of how velocity models are set up in *Antelope*. The link below contains a quick velocity model setup guide for *dbgenloc*. The link references *Antelope* 4.4, but the principles still apply to current *Antelope* versions.

http://www.indiana.edu/~aug/genloc/vmodels/genloc_vmodel_setup.html

A sample version of a customized velocity model for the Colorado Plateau in Northern Arizona is available for reference here. Note that *dbloc2* will only recognize a velocity model if the *modelname.pf* file is placed within the *$ANTELOPE/data/tables/genloc/* directory.

[4.0] Database Processing

Previous sections discussed how to manually pick arrivals with *dbpick* and locate events with *dbloc2*. This section will go over how to automatically detect events and process an event catalog using *dbdetect*, *ttgrid*, dbgrassoc, and *dbevproc*.

[4.1] *dbdetect*

Antelope's *dbdetect* program reads waveform data, runs detectors based on short-term average versus long-term average (STA/LTA) for waveform amplitude, and flags any times where the amplitude of the trace is significantly above background noise level. The *dbdetect* MAN page gives a sample parameter file and short descriptions of the function of each parameter. This guide covers the most useful command-line options and lists the parameters that are most likely to be edited when running *dbdetect* and looking for small, local earthquakes.

Parameter File (*dbdetect.pf*)

When editing a *dbdetect.pf* file, keep in mind that it might be useful to have a different *dbdetect.pf* file for each test that is performed with differing parameters. In this case, copy the *dbdetect.pf* file to the *dbname/pf* directory, rename it to *dbdetect_test1.pf* and view it:

```
% cp $ANTELOPE/data/pf/dbdetect.pf ./pf/dbdetect_test1.pf
% vi pf/dbdetect_test1.pf
```

Notice that the top portion of the original *dbdetect.pf* file contains default parameter values. Most of these values will be left unchanged. When a change is made, it is generally a good idea to comment out the original line from the parameter file with"#" and enter your edited line below. The following is a list of parameters that should be changed for a database exploring local, small magnitude earthquakes:

nodet_twin

Default value is set to 2.0 seconds, so any signal with amplitude above background noise level for less than two seconds will be ignored by *dbdetect*. This entry eliminates any high-impulse amplitude spikes, but may also eliminate signal from earthquakes directly adjacent to stations. For a dataset that is exploring small, local earthquakes, this value can be reduced to zero:

```
nodet_twin      0       # no detection if on time is less than this
```

thresh

This value represents the detection signal to noise ratio (SNR) threshold. Default value is set to 5.0, but for detecting micro-earthquakes, this parameter can be lowered to around 3.5. Trial and error is the best way to determine what value works best for your data set. A value that is too low will increase the number of false event detections. The value of 3.5 allows for the detection of most events magnitude 1.0 or larger while having relatively few false detections.

```
thresh          3.5     # detection SNR threshold
```

The next section of the default *dbdetect.pf* file is used to set up parameters for frequency band(s) used by *dbdetect* to search through the waveform data. At least one band must be set up, but multiple bands are often useful. This example sets up (1) a band for detecting local events, and (2) a band with parameters used by ANF for the QED monthly catalogs.

```
bands&Tbl{
        &Arr{
                iphase l        # this table sets flags labeled “l” for “local”
                sta_twin        1.0
                sta_tmin        1.0
                sta_maxtgap     0.5
                lta_twin        5.0
                lta_tmin        2.5
                lta_maxtgap     3.0
                pamp            500.0
                filter          BW 3.0 4 10.0 4
        }
        &Arr{
                iphase a        # this value sets flags labeled “a” for “ANF”
                sta_twin        1.5
                sta_tmin        1.5
                sta_maxtgap     0.5
                lta_twin        10.0
                lta_tmin        5.0
                lta_maxtgap     4.0
                pamp            500.0
                filter          BW 0.5 5 5 5
        }
}
```

Next, be sure to specify the stations and channels that *dbdetect* will be searching. The following examples (1) (commented out) searches only vertical channels on all stations, and (2) searches vertical channels on stations beginning with X, Y or Z and ending in 15A-18A. Note that the two “.” symbols are wild cards for any character, and the syntax for identifying stations is the same as the “sc” command in *dbpick*.

```
stachans        &Tbl{
#       sta     chan
#       .*      [SB]HZ|BHZ_..
        [XYZ]1[5678]A   BHZ|BHZ_..
}
```

Also, **be sure to include the “reject” table, or else *dbdetect* might give you errors**. The table can be empty as below, or you can specify stations to reject or ignore while running *dbdetect*.

```
reject &Tbl{
#       sta     chan
#       ABC     BHZ
}
```

Finally, there are several optional tables that can be added to a *dbdetect.pf* file in which specific stations can be ignored or specific parameters can be assigned to override parameters for one station or channel. Examples of these options are given in the *dbdetect* MAN page.

dbdetect Command Line Options

For a complete list of command line options, view the MAN page or type "dbdetect" into the terminal for command line syntax. When testing new edits to your *dbdetect* parameter files, it is best to only process a small amount of the database at a time (hours to a day), otherwise, *dbdetect* might take hours to finish:

```
% dbdetect -v -tstart 2008:143:12:00:00 -twin 720 -pf pf/dbdetect_test1.pf db/test1/test1
db/test1/test1
```

To break this command line entry down step by step:

-v	Verbose flag. Displays *dbdetect* output
-tstart	Time start flag with syntax year:julday:hr:min:sec
-twin	Time window (in minutes) (720 = 12 hours)
- pf	Specify path and filename of parameter file (*pf/dbdetect_test1.pf*)
dbin	Path and filename of input database descriptor file (*db/test1/test1*)
dbout	Path and filename of output database descriptor file (*db/test1/test1*)

If *dbdetect* is being run on a large amount of data, it may be a good idea to redirect the *dbdetect* verbose output into a separate file (type ">! filename" at end of command line). This will speed things up and make it easier to access the output and determine if *dbdetect* is behaving as expected. Once *dbdetect* has finished, the *test1* database will now have a *test1.detection* file. Open *dbpick* and navigate to the time during which *dbdetect* was run.

```
% dbpick -ts 2008:143:12:00:00 db/test1/test1
```

The "sd on" command in *dbpick* will display the detections as red markers. Notice that some of the markers are labeled with "l" and others with "a". Also, there may be two detection markers in one place. This is a result of using more than set of detection parameters in the "band" table. Any duplicate detections will be ignored later in *dbgrassoc*.

Using DBDETECT with Your Main Database

While running tests in *dbdetect*, it is a good idea to keep time windows small in order to be able to quickly and easily check the results. Once you are satisfied with the *dbdetect* settings used in the *test1* database, you are ready to copy the parameter file to a more permanent name (i.e. *dbdetect_dbname.pf*) and use it to process your entire *dbname* database. If *dbdetect* is run on a large database, it may run out of memory and crash before it can finish the job. If this happens, *dbdetect* can be broken up into multiple chunks of time while processing the entire *dbname* database. In this case, the *dbout* on the command line should be *dbname.2007a* or some similar variation. Later, *dbgrassoc* can use *dbname.2007a* as the *dbin* and can divert output for all time chunks back into the *dbname* database.

[4.2] *ttgrid*

Now that *dbdetect* has flagged all of the potential events, the next step is to use the detections to produce a list of event arrivals. The best way to do this in *Antelope* is by using *dbgrassoc* (abbreviated from db-grid-association). As the name implies, *dbgrassoc* uses a travel time grid to associate observed arrivals with hypothetical earthquake hypocenters. Each area of study should have a unique travel time grid, so it is a good idea to create a separate grid for each area that is searched. These grids should be stored in the *grids/* directory.

Parameter File (*ttgrid.pf*)

Copy the default version of *ttgrid.pf* over to your *grids/* directory, rename it, and view it:

```
% cp $ANTELOPE/data/pf/ttgrid.pf ./grids/ttgrid_test1.pf
% vi grids/ttgrid_test1.pf
```

The original *ttgrid.pf* file has examples of several different types of grids. Later, *dbgrassoc* will reference one of these grids, so be sure to pay attention to names of each grid that is used. This guide will use the first grid in the main *grids* table (labeled *local*). Comment out the *regional*, *tele_slow* and *tele_uni* since they will not be used in this example. The following is an example of a grid centered at a coordinate point rather than a station location. Adjust the *latr*, *lonr* and size of your grid to cover your area of study:

```
grids &Arr{
        local &Arr{
                mode            edp     # defines an equal-distance projection regular 3-D mesh
                #type           1d      # defines travel times base upon a 1-D structure model
                #ndist          10000   # number of distance nodes in 1-D grid
                latr            34.5    # reference latitude (origin of grid - midpoint)
                lonr            -112.0  # reference longitude (origin of grid - midpoint)
                nx              151     # Number of X-axis distance grid nodes
                ny              151     # Number of Y-axis distance grid nodes
                xmin            -0.5    # Minimum value of X-axis distance grid in degrees
                xmax            0.5     # Maximum value of X-axis distance grid in degrees
                ymin            -0.5    # Minimum value of Y-axis distance grid in degrees
                ymax            0.5     # Maximum value of Y-axis distance grid in degrees
                strike          90.0    # Angle from north clockwise in degrees to the X-axis
                compute_P       yes     # yes = Compute P travel times
                compute_S       yes     # yes = Compute S travel times
                method          tttaup  # method for computing travel times
                model           iasp91  # model for computing travel times
                pxstamin        -50.0   # minimum x-range for finding stations as percent
                pxstamax        150.0   # maximum x-range for finding stations as percent
                pystamin        -50.0   # minimum y-range for finding stations as percent
                pystamax        150.0   # maximum y-range for finding stations as percent

        #optional
        depths &Tbl{            # table of grid depths.
                        0.0     # More entries means more depths used, higher accuracy.
                        5.0
```

```
                10.0
        }

    #optional
    stations&Tbl{       # table of station names
                Y18A    # only stations on this list will be used in the grid.
                Y17A
                Z17A
                Z16A
                X16A
                X15A
        }
    }
}
```

In the example above, 1d lines are commented out because *ttgrid* uses a 3d grid type if this value is not specified (and that's what we want). Also note that the *latr* and *lonr* are the origin (or midpoint) of the grid with the length of each axis defined by *xmin*, *xmax*, *ymin* and *ymax*. The *nx* and *ny* parameters are the number of nodes along each axis, and the optional depth table represents the number of depth nodes. A larger the number of nodes will result in more precise the locations, and an increase in computation time. If the optional *stations* table is not included, then *ttgrid* will use all stations.

ttgrid Command Line Options

Once the *ttgrid.pf* file has been customized to cover the area of study, run *ttgrid* with the following command line options:

```
% ttgrid -pf pf/ttgrid_test1.pf -time all db/test1/test1> grids/test1_grid
```

Command line options:

-pf	Allows user to specify which *.pf* file to use
-time	Can enter "all", "now" or time string: "year:julday:hr:min:sec". Choose stations to use based on which stations were/are active at specified time.
dbname	Specify a database path and filename (*db/test1/test1*)
>	Must use redirect to generate grid file
gridfilename	Specifies grid path/filename (*grids/test1_grid*)

After the grid has been completed, the grid can be displayed by entering:

```
% displayttgrid grids/test1_grid local
```

The "local" at the end of the command line represents the grid name, but it is not necessary to enter this on the above command if the other grids have been commented out of the *ttgrid_test1.pf* file.

[4.3] *dbgrassoc*

So far, a list of potential event arrivals has been produced using *dbdetect*, and a travel time grid has been established with *ttgrid*. The next step is to use a database grid association (*dbgrassoc*) algorithm to search for potential hypocenter locations that best fit the detections from *dbdetect*. When working

with *dbgrassoc*, keep in mind that the MAN page for *dbgrassoc* refers to the MAN page for the real-time version of the program (*orbassoc*) because the parameter files and function of both are essentially identical. *dbgrassoc* will write to the *.arrival*, *.assoc*, *.event* and *.origin* tables.

Parameter file (*dbgrassoc.pf*)

Note that the *dbgrassoc* parameter file looks a bit different than the *orbdetect.pf* example in the *orbdetect* MAN page. This guide will explain how to edit the *.pf* file to search for small, local events. Only the parameters that should be edited from the default *dbgrassoc.pf* file will be listed below. It is a good idea to only edit a copy of each line and comment out the original line using "#". To get started, copy the default version of *dbgrassoc.pf* over to the *pf/* directory, rename it, and view it:

```
% cp $ANTELOPE/data/pf/dbgrassoc.pf ./pf/dbgrassoc_test1.pf
% vi pf/dbgrassoc_test1.pf
```

A copy of a *dbgrassoc.pf* file able to locate a cluster of events ranging from magnitude 1.0 and up is listed below:

```
#   Parameter file for dbgrassoc

process_time_window  20.0          # Main detection processing time window
process_ncycle       0             # how often to do detection processing, in detections
process_tcycle       10.0          # how often to do detection processing, in delta time

grid_params &Arr{
        local &Arr{
                nsta_thresh     &Tbl{   # Minimum allowable number of stations
                                        #expressed in degrees and number of stations
                        1.5     4       # accept only events with 4+ arrivals within 1.5 degrees
                        2.0     10      # accept only events with 10+ arrivals within 2 degrees
                }
                nxd             11      # Number of east-west grid nodes for depth scans
                nyd             11      # Number of north-south grid nodes for depth scans
                cluster_twin    1.0     # Clustering time window
                try_S           no      # yes = Try observations as both P and S
                                        # no  = Observations are P only
                associate_S     no      # yes = Try to associate observations as both P and S
                reprocess_S     no      # yes = Reprocess when new S-associations found
                phase_sifter    l|a
                auth            dbgrassoc
                P_deltim        0.1   # Default deltim value for arrival table rows for P arrivals
                S_deltim        0.2   # Default deltim value for arrival table rows for S arrivals
                priority        5
                use_dwt         yes     # turns on distance weighting
                dwt_dist_near   1.5     # distance for "near"
                dwt_wt_near     0.5     # weight for "near"
                dwt_dist_far    2.0     # distance for "far"
```

```
                dwt_wt_far      0.0     # weight for “far”
                closest_stations2.0     # zero weight for stations beyond this degree distance
                relocate        rundbgenloc     # Run this relocation script to refine the final
                                                # solution
                use_only_relocation yes  # If relocation converges, just output the relocation
                drop_if_on_edge yes
        }
}
```

No changes are made to the area below “# parameters for "smart" association”.

Detailed explanations of each parameter are provided at http://www.brtt.com/wiki/Orbassoc(1). Below is a brief clarification of key parameters to edit:

process_time_window
This parameter should be set to a value on the order of the total P-wave travel time (in seconds) across your grid. Larger grids should have larger values, while small grids should have small values. In the example provided, the value is changed from default 500.0 to 20.0 for a grid that is about 1 square degree.

process_ncycle and ***process_tcycle***
The *process_ncycle* parameter is used in real-time processing, so for the *test1* database it should be turned off by setting it to 0.0. The *process_tcycle* tells *dbgrassoc* how often to load in events for processing. The parameter can be set somewhere around 10.0 seconds for a database dealing with local events.

nsta_thresh
This parameter is probably the most important parameter for *dbgrassoc*. It lists the minimum number of stations required to create an event origin. A larger number of stations will reduce the number of false events, but it will also reduce sensitivity (since small earthquakes may not have clear signal on many stations). For locating events down to magnitude 1.5 and below with station coverage similar to USArray (70 km spacing), station threshold can be set as low as 4 without creating a large number of false events. As shown above, this parameter can also be expressed as a table. By limiting the number of arrivals to 4 or more within 1.5 degrees of the hypocenter, we eliminate most unwanted detections associated with teleseismic and false detections.

cluster_twin
This is another important parameter for *dbgrassoc*. If an observed arrival is within this time window (in seconds) from the hypothetical arrival value for an event, then it is considered to be associated with that event. If not, then it is considered to be part of a separate event. A smaller value will reduce the number of false events identified, but the value should be large enough to accommodate any potential error (pick uncertainty, velocity model, grid spacing, etc). Default value is 1.5, but 1.0 reduces spurious events effectively when searching for small earthquakes.

associate_S and ***reprocess_S***

By default, these values are set to "yes" so *dbgrassoc* will try to associate some arrivals as S arrivals. This may cause some problems since *dbdetect* was only run on vertical (BHZ) components (see Section 4.1). Therefore, set the value for each of these parameters to "no" for this database.

phase_sifter
This parameter allows the user to specify which detection flag IDs to use in *dbgrassoc* (separated by pipe "|" symbol). This guide used the "l" and "a" flags in *dbdetect*, so both should be included in the parameter line: phase_sifter l | a

auth
This is the author field that will be assigned to the events in the *.origin* table. It should be set to "dbgrassoc".

P_deltim
Value (in seconds) set as the default time uncertainty in the *.arrival* table for relocation. A good value for this parameter is 0.1.

dwt parameters
These are distance weighting parameters used by *dbgrassoc*. Values in the example above are set for analyzing small, local earthquakes.

relocate
The parameter above is set to relocate each event using *rundbgenloc* to refine the final solution.

use_only_relocation
This parameter is set to only output the final relocation computed by *dbgrassoc*. Essentially, setting this command to "yes" will reduce the number of duplicate event origins in your database.

dbgrassoc Command Line Options

Once the *dbgrassoc* parameter file has been set up, enter the following command:

```
% dbgrassoc -v -pf pf/dbgrassoc_test1.pf db/test1/test1 db/test1/test1 grids/test1_grid
```

Command line options:

-v	Verbose option
-pf	Allows user to specify which *.pf* file to use
dbin	Path and filename of input db
dbout	Path and filename of output db
gridfilename	Specifies grid path/filename

After the *dbgrassoc* is finished, the detected events can be viewed in *dbpick*:

```
% dbpick db/test1/test1
```

Pressing the "n" (for next) and "p" (for previous) keys in the trace window of *dbpick* will quickly and easily skip from event to event. The *test1* database should now be filled with events; however, it is likely that there are several teleseismic or regional events beyond your grid and multiple false events that

have been flagged by *dbgrassoc*. These unwanted events are difficult to get rid of without also losing the smallest local events. If you feel that there are simply too many unwanted events, run some new test databases by changing various parameters of the *dbdetect* and *dbgrassoc* parameter files. Trial and error is the best way to do this. Start by increasing the number of stations required to locate an event, or by adjusting *process_time_window* or *process_tcycle parameters*. This will reduce false events, but it will also reduce the number of small earthquake detections.

[4.4] Developing Your Catalog

Once you are satisfied with the results of the *dbdetect-ttgrid-dbgrassoc* process, it is time to go through the database and develop your catalog. Using the parameter files from the *test1* database to process the entire *dbname* database will result in some unwanted detections. There are a many ways to work with this issue, but it is easiest to adjust your parameters to minimize false detections and then delete any unwanted events from the *.origin* tables using *dbloc2*.

[5.0] Database Surgery/Troubleshooting/Misc

[5.1] Adding Waveform Data

This section lists the steps necessary to add waveform data to an existing database.

1. When converting miniseed files to waveform data, it is easiest to use the *miniseed2days* command. The default format for waveform file names created by *miniseed2days* does not match the format that ANF uses. It is not vital that this is changed, but it may make things easier later. Copy the *miniseed2days* parameter file from *$ANTELOPE/data/pf/miniseed2days.pf* to the *pf/* directory and edit it to match the following wfname line:

   wfname %Y/%j/%{sta}.%{chan}.%Y.%j.00.00.00.msd

2. Request the data for download from http://www.iris.edu/forms/webrequest.htm. Be sure to request these files in the miniseed format with a file extension of .mseed. Once the request is ready you should get an email.

3. Copy the .mseed files into the *certified/* directory.

4. Run the following *miniseed2days* command from within the *dbname/* directory:

   % miniseed2days -v -S certified/ -d newdb - < certified/filename.mseed

   This will unpack the miniseed file into the year/day/ directory structure that is already established for your *dbname* database. The "-d" option creates a new database with the *.wfdisc*, *.lastid*, *.schanloc*, and *.snetsta* tables. The "-S" option specifies which directory to place the waveform files. If the .mseed file is larger than 2GB, the "- <" is required for the *miniseed2days* command.

5. Make backup copies of the *dbname* database.

6. Run *dbmerge* to combine the *waveformdb* created by *miniseed2days* with existing *dbname* database:

   % dbmerge *waveformdb* dbname

   This command takes the database tables from *waveformdb* and adds them to the tables in *dbname*. There must not be any duplicate or overlapping data between the two dbs, otherwise errors will occur.

7. Open the *dbname* database using *dbpick* and make sure that the new waveform data is visible.

8. If everything looks to be merged correctly, move all of the *waveformdb* tables into the *backup/* directory or delete them.

[5.2] Replacing Bad Data

This section will explain how to delete a section of waveform data and replace it. This can be useful if you find that a station has missing data where it should be continuous.

1. Save a backup copy of your database!

2. Create a subset and delete the time range containing bad data:

   % dbsubset dbname.wfdisc "sta=='U17A' && (time >= '2008:200:00:00:00' && endtime < '2008:203:00:00:00')" | dbdelete -sv -

3. Manually delete waveform files with bad data.

4. Run *miniseed2days* and *dbmerge* for the new (good) data according to Section 5.1 above.

[5.3] Fixing Multi-Station Data Dropout

At some point, you may experience what appears to be a data dropout for multiple stations in *dbpick*. If this occurs, try checking the *datatype* column of your *.wfdisc* table in *dbe*.

```
% dbe dbname
```

To view this column in *dbe*, click on the button for the *.wfdisc* table, choose *View -> Arrange* from the drop-down menus and select *datatype*. For most databases, the *datatype* field should be "sd", which stands for "seed". To check if you have incorrect listings, click on the *datatype* column header and select *Sort* from the menu. This may take several minutes if your database is large, but it will eventually sort the column in alphanumeric order. Using the scroll-bar on the left side of the screen, check the top and bottom rows of the *.wfdisc* table to be sure that all *datatype* field entries are listed as "sd".

If all of the values are listed as "sd", then there is probably an issue with the waveform data itself. If there is a different entry in the datatype field ("MS" for example), then the data dropout in *dbpick* is probably due to an indexing issue.

To fix an indexing issue, save a backup copy of the *.wfdisc* table and then change value of all *datatype* fields to be "sd". For help on this refer to the section marked "Editing Database Tables" in Section 1.7

[5.4] Converting miniseed to SAC

Converting miniseed to SAC is easy with db2sac:

```
% db2sac -sc T14A:BH* -ts 2007:146:23:45:23 -te 2007:146:23:50:23 db/test1/test1 db/sac/sacdb
```