PROCEDURE FOR CROSS-CORRELATING WAVEFORMS IN ANTELOPE DATABASES AND OBTAINING DIFFERENTIAL TIMES FOR IMPROVED LOCATION ACCURACY USING HYPODD

CONTENTS

Change history:

R0	05/20/2006	David von Seggern (vonseg@seismo.unr.edu)
R1	07/11/2006	DVS -- some major changes
R2	01/08/2008	DVS -- minor edits
R3	04/21/2008	DVS -- minor edits
R4	05/19/2008	DVS -- data retrieval now based on event table, not origin table
R5	09/10/2008	DVS -- added picked arrival-time difference to dbcorrelate output file
R6	10/26/2009	DVS -- minor edits
R7	09/30/2011	DVS -- eliminating the workaround for faulty dbopen; added argument to dbcorrelate command line; provided for direct path from event.sel and dt.ct files
R8	10/04/2012	DVS -- minor edits prior to adding to Antelope contrib

INTRODUCTION

Purpose: The procedure for using cross-correlation (CC) of waveforms to improve event locations is much more complicated than simply using catalog arrival time data, even with perhaps "master event" or similar techniques. This document describes a complete procedure, tied to Antelope databases, for generating and using CC data. It is somewhat specific to the Nevada Seismological Laboratory (NSL) but should be readily adaptable to other organizations with an Antelope database.

Assumptions: To successfully use this procedure without modification, the reader should be aware of several assumptions.

1) The source data should exist in an Antelope database which is organized by year and day-of-year (for example, "008", "079", or "236" under "2006"). Other organizations will require changes in programs and scripts used here if their file structure is different.
2) Events should be picked by analysts (automated arrival picks are not sufficient) and located to reasonable accuracy.
3) Online waveforms should exist for all picked arrivals and be referenced by the wfdisc tables.
4) Waveforms should have sample rates between 50 and 200 sps.

The essential steps in the full procedure are (see **Appendix 1** for flowchart):

1) assemble an Antelope database in a new directory, subset for the area of interest, leaving the waveforms in their original directories (optional, may not be necessary),
2) search for events in close proximity to be cross-correlated,
3) perform cross-correlations on waveforms (P and/or S),
4) convert cross-correlation output into a file readable by HYPODD,
5) convert catalog data in Antelope tables to a file readable by HYPODD
6) run HYPODD, using both catalog and cross-correlation data,
7) convert HYPODD output to familiar, usable data formats.

An alternative procedure exits for when the desired events have already been relocated with HYPODD. In this case, one can work directly from the events.sel and dt.ct files created to run HYPODD and add cross-correlation results in another HYPODD run (see **Appendix 2** for flowchart). In this case, steps 1 and 2 above can be ignored.

Preliminary step: Do a hypocenter search within the area of interest using the appropriate CSS3.0 origin table to see how many events are there and what their space-time distribution is. There are many ways to do this search, but the Antelope dbe interface is easy and usually satisfactory.

Programs to handle CC work should be stored in a single directory accessible to the user. In this documentation, the symbol $ccprog will be used to designate this directory. For convenience, one should set up this symbol in their environment if planning to do CC work. At NSL this is

% set ccprog "/data/correl/programs"

ASSEMBLING THE DATABASE

Unless one is using all the events in an existing database, efficiency requires that a local database comprising just the desired events be created. Invoke the script get_db.csh to assemble this local database. This script may need to be changed to accommodate an operational Antelope database having a different directory structure than assumed here. Invoke the program with, for example:

```
% $ccprog/get_db.csh ccdir db root dbroot circle latmin latmax lonmin lonmax
  zmin zmax tmin tmax append
```

where the arguments are:

ccdir defines the directory (full or relative path) in which the CC work is being done (likely the current directory, but use the full path to designate it).
db specifies the Antelope database name to be used in *ccdir*.
root is the root directory of the Antelope permanent database, assumed to be broken into years under *root* and then day-of-year under years.
dbroot is the database name for databases in the root directory tree (must be constant).
circle is a flag to indicate areal selection will be based on a rectangle (0) or circle (1).
circle = 0:
latmin, latmax are the minimum and maximum latitudes for the search box in decimal degrees (south latitudes are negative),
lonmin, lonmax are the same for longitude (west longitudes are negative).
circle = 1:
latmin, latmax are the latitude and longitude center point for the search disk in decimal degrees (south latitudes are negative; west longitudes are negative),
lonmin, lonmax are inner and outer radii (km) for the search disk (equivalent to a circle if the first is set to zero).
zmin, zmax are the depth range endpoints (km) (zmin = 0 typically).
tmin, tmax are the minimum and maximum epoch times (seconds) for the desired data search.
append is a flag to append to existing database (1) or to start new database (0)

The job may take a long time, on the order of hours if there are a large number of events which satisfy the criteria. The job creates a new database in the local directory with the new wfdisc simply pointing to the original files. Thus no new waveform files are created. The job retains the original evid, orid, arid, and wfid numbers; if it is important to renumber them, the script must be modified to do so. Events are selected by first joining the event to the origin table, so that the preferred origin id is used.

IDENTIFYING EVENTS TO BE CROSS-CORRELATED

If using the full set of steps (1 to 7 on p. 1 and Appendix 1), do the following:

Unless the source region is very small, it is not reasonable that every event be cross-correlated with every other. When event separations become larger than 1/5 to 1/10 the station distance, they will likely have correlation coefficients for corresponding waveforms that are statistically not different that zero. Thus a means of limiting the correlations to pairs of events where the separations are small is needed. Make sure the origin table is time sorted before proceeding. The following program operates on the database just created to identify these event pairs:

```
% $ccprog/event_pairs db xtol ytol ztol maxlinks outfile
```

where the arguments are:

db is the database name in the local directory (formed by get_db.csh above),
xtol, ytol, ztol are the tolerances (km) to use for hypocenter differences between event pairs. Normally the z tolerance should be largest to allow for the fact that depth is the least certain coordinate of the hypocenter solution.
maxlinks limits the number of events with which any given event can be paired. This prevents over-determining the event relative locations and can immensely save on the amount of computational time when events are closely packed in space (computational time savings of one or two orders of magnitude). Events are ordered spatially from the given event before the events are chosen. Thus, in approaching max_links, the nearer events are presumably caught first. To cross-correlate every event with every other, set maxlinks to the total number of events.
outfile is the name of the file to hold the event pairs. This file has only four fields: orid1, orid2, jdate1, jdate2.

If using the abbreviated set of steps (3-7 on p. 1 and Appendix 2), do the following:

Determine in what directory the event.sel and dt.ct HYPODD files are (it may be the current directory), substitute it for the argument “datadir”, and then run

```
% ccprog/event_pairs_fast.csh datadir
```

CROSS-CORRELATING WAVEFORMS

The inter-event time can be estimated for each station recording both events in a pair by cross-correlating those recordings. This estimated time is dependent on the time of the peak of the cross-correlation function; that is, the time of the maximum cross-correlation coefficient (CCC).

The next step is to identify phases for cross-correlation and compute the cross-correlations to estimate the inter-event times. The procedure for the next step depends on the size of the database. Due to the need to get information for event pairs from non-contiguous parts of the database, a large number of joins and subsets are required. This becomes very inefficient as the database gets large, say if the hypocenter list has more than a couple hundred events. It's an N-squared type operation. For instance, an origin table containing just 200 events can easily generate tens of thousands of possible correlation pairs. An origin table with 2000 closely located events could generate millions of cross-correlations. An alternative method is needed for larger databases, say > 1000 events.

For a small database, use Method A below; and, for a large database, use Method B.

Method A: Small Database

To process a small database, go to the local directory with the new database and type:

```
% $ccprog/dbcorrelate db dbpath flag winlen laglen cccmin inputfile outputfile
```

where the arguments are:

db is the database created above
dbpath is the path to that database
flag is a flag [0/1] which tells the program where to get the data, either in the main directory (0) or in the subdirectories (1). In this case, use flag = 0.
winlen is the length of waveform data, in seconds, to use for the cross-corrrelation. A value of 1 to 2 seconds is typically good. The window is cut starting 0.1 seconds before the particular arrival time.
laglen is the lag, in seconds, for the correlation function computation, or how far one waveform will be shifted, both ways, relative to the other. A value of 0.2 to 0.3 seconds is desirable.
cccmin is a parameter to control the output size. Because station-event pairs for which the |CCC| is low (say, 0.7 or less) are probably not useful, it is reasonable to set this value to eliminate many of the CC results. Note that the absolute value is used, so that “0.7” means all $-0.7 \leq CCC \leq +0.7$ are saved.
inputfile is the file created in running the "event_pairs" program above.
outputfile is the name of the file to hold the output. (For restarts, remove this file first because it is opened “append”).

The output file contains the CC results, with these fields

sta -- station code (for instance, DOM)
phase -- phase type (P or S; any others should be excluded if they appear)

orid1 -- orid for 1st event (local table number)
jdate1 -- julian date (jdate) for 1st event
orid2 -- orid for 2nd event (local table number)
jdate2 -- julian date (jdate) for 2nd event
ccc -- cross-correlation coefficient at the peak of the CC time series ($-1 \leq ccc \leq 1$)
cccp -- cross-variance coefficient at the CC peak, measuring the relative signal amplitude
tau -- absolute time difference (seconds) between signals (t2 - t1) as given by the CC peak
otdiff-- difference in catalog origin times (ot2 - ot1) in seconds
atdiff -- difference in catalog arrival times (at2 - at1) in seconds

The numbers in the last three fields can be quite large and are computed in double precision arithmetic. This run will take awhile, on the order of hours if the number of events is in the hundreds.

Method B: Large Database

To process a database with a large number of events, first one divides the local database into its daily tables and then runs dbcorrelate as for Method A, only now using "flag" = 1. To separate a large database, go to the directory with the new database and separate it by typing:

```
% $ccprog/separate_db.csh db startjdate endjdate
```

where "db" is the database created above and startjdate and endjdate are the first and last julian dates in the time-ordered origin table.

CULLING THE DBCORRELATE OUTPUT DATA FILE

The program dbcorrelate may produce multiple lines for a given phase for a given station for a given event. This is because the wfdisc table may have multiple entries which cover the phase even though sta-chan is the same. To prevent using multiple cross-correlations for a given phase, the output file should be filtered to remove all but one of these multiple instances. This syntax for "uniq" will throw out lines which have a file name different than the previous line, while all else is the same.

```
% uniq -f 4 correl_output.dat > temp.dat
% wc -l correl_output.dat temp.dat
(temp.dat should show only a small decrease of lines relative to
correl_output.dat)
% mv temp.dat correl_output.dat
```

Note that both negative and positive CCC's are supported. Negative CCC's arise when the polarity of the arrival changes between two events due to a rotation of the focal sphere (different focal

mechanisms). Note that tau may be greater or less than otdiff; but they should be close, differing by no more than a few hundredths of a second.

The output file should normally be culled for acceptable data before proceeding to relocation of the events. For instance, station-event pairs for which the P and S results differ greatly in tau or differ in sign are probably less reliable. Results where tau differs greatly from otdiff should be carefully considered for removal -- this may be due to "cycle-skipping" in determining the peak of the correlation function, for instance. Some of these criteria are incorporated in program "create_ccfile", described next.

USING THE CROSS-CORRELATION OUTPUT IN EVENT RELOCATION

The correlation output can be used in the relocation program HYPODD, after suitable reformatting with:

```
% $ccprog/create_ccfile inputfile outputfile cccmin iflag dtchop spdifmin minobs
```

where the arguments are:

inputfile will be the output file of the dbcorrelate program above,
outputfile will hold the output (arbitrary name, but “dt.cc” is consistent with HYPODD documention),
cccmin is the minimum value of |CCC| to accept (0.7 is suggested),
iflag is a flag to accept (0) or reject (1) CCC’s which are negative but have |CCC| >= *cccmin*
dtchop is a cutoff value. dt's whose absolute value exceeds this will not be accepted. A value of around 0.5 is suggested. A value of 0.0 means no check will be made on dt.
spdifmin is the minimum absolute difference between S and P CC results (for a given station pair and given event pair) to accept; otherwise both observations are discarded (0.1-0.2 is suggested),
minobs is the minimum number of CC results to have in order to accept an event pair (if one wants all event pairs to be used, no matter how few CC observations, enter “1”)

Create a file called "station.dat" to use with HYPODD. This file has three fields: station code (for example, "MPK" or "CCHS"), latitude, longitude. The coordinates are in decimal degrees. Format is free-field.

Now create a catalog phase-arrival file "phase.dat" with

```
% $ccprog/db2ph db phase.dat define_flag
```

where the first argument is the actual database name to be used, the second argument is the output file name (arbitrary name, but called "phase.dat" in HYPODD documentation), and the third argument is a flag to use only defining phases (timedef = "d") in the assoc table (1) or to use all

associated phases in the assoc table (0). This output file is the input to a preprocessor for HYPODD called PH2DT, distributed with HYPODD software. The PH2DT program is documented in the HYPODD manual (USGS OFR 01-113). It requires a parameter file, say "ph2dt.inp". A typical example consists of four lines:

```
% cat ph2dt.inp
station.dat
phase.dat
* MINWGHT MAXDIST MAXSEP MAXNGH MINLNK MINOBS MAXOBS
     0      100     5       8     8      8      15
```

where the named files are the ones just described. See the HYPODD documentation for the numeric parameters. Assuming that one has set $hypoddsrc to the "src" directory created from the installation of HYPODD, run the program with:

```
% $hypoddsrc/ph2dt/ph2dt ph2dt.inp
```

This produces a file called "dt.ct" which contains the catalog arrival times to be used in the HYPODD run. It also creates "event.sel" -- a file of catalog locations to use as starting solutions. Another created file, "event.dat", contains all the hypocenters; but generally, one wants to use "event.sel" because it has the culled hypocenter list, according to MINOBS. In general, the number of events in the culled file should not be greatly less than that in the original hypocenter list; if it does, then one may need to modify the parameters used in PH2DT. A log file (ph2dt.log) is created by running PH2DT, and it should be scanned for obvious problems.

The HYPODD run is controlled by a parameter file called "hypodd.inp". See the HYPODD documentation for an example of this file. Copy the one from the HYPODD delivery directory or from a previous run to the current directory. It is suggested that one should create a subdirectory to hold the files needed for HYPODD and the numerous ones which HYPODD will subsequently create. If one has collected all the relevant files at this point into a subdirectory, it would look similar to this:

```
% ls
dt.cc          event.dat     hypodd.inp    ph2dt.log     station.dat
dt.ct          event.sel     ph2dt.inp     phase.dat
```

Once the parameters are set in hypodd.inp (see HYPODD documentation), run HYPODD with:

```
% $hypoddsrc/hypoDD/hypoDD hypodd.inp
```

HYPODD requires a large memory allocation. It may be that a run's requirements exceed the compiled memory allocation of HYPODD. If you get the message "killed" when invoking HYPODD, it is likely due to insufficient memory. Either try another machine with more memory or recompile the program with smaller array sizes. Another type of immediate runtime failure is

when the dataset is too large for the compiled array sizes; again try adjusting array sizes and recompiling.

HYPODD creates many files, but the relocated hypocenters are in “hypodd.reloc”. Greatly improved relocation results should be obtained when compared to the original catalog locations. HYPODD output, especially the “hypodd.log” file, should be scanned for obvious problems and for assurance of proper behavior in convergence. HYPODD may create more than a single cluster of events if the original set of events is dispersed into clumps. Occasionally, clusters consist of small numbers of events, even as small as two. (Isolated, single events are not relocated by HYPODD.) All relocated events, regardless of cluster association, are contained in the file “hypodd.reloc”.

The “hypodd.reloc” file can be transformed to a CSS3.0 origin table with:

```
% $ccprog/hypodd2db hypodd.reloc reloc
```

where the first parameter is the HYPODD relocated event file and the second parameter is the database name (arbitrary).

The origin table can be used to make a 3-D point file suitable for many plotting programs with this example awk script:

```
# A script to convert (lat,lon,depth,OT) to (x,y,z,t) coordinates.
#
# $1 = latitude in decimal degrees (S is negative).
# $2 = longitude in decimal degrees (W is negative).
# $3 = depths (km) that are positive downwards.
# $4 = origin time in epoch seconds.
# Input parms are the reference location as (reflon,reflat,refdep).
# Output is in meters relative to the reference (lon, lat, depth) given on
# the command line.   It is not UTM coordinates.
# This script puts relative depths as negative below the reference point,
# positive above it.
# This script works directly with CSS3.0 origin tables.
# Run this with nawk, for instance:
#
# % nawk -v reflon=-120.0607 -v reflat=39.2230 -v refdep=25.6606 \
# % -f xyzt.awk inputfile > outputfile
#
#
{
x = ($2 - reflon)*111.1*cos($1/57.2958)*1000
y = ($1 - reflat)*111.1*1000
z = (refdep - $3)*1000
t = $4
printf("%10.4f\t%10.4f\t%10.4f\t%15.2f\n",x,y,z,t)
}
```

TEST CASES

Test cases are supplied with the delivery tar file in the subdirectories “testcase1” and “testcase2”. This first uses synthetic traces for four events in close proximity. Using the procedure in this document, one should be able to reproduce the original event locations, given in “hypocenter.info”, very accurately, to within 100-200 meters. Actually, we expect better accuracy with real data because, due to the manner in which the synthetic dataset was created, there are some inherent errors which limit the attainable accuracy. An additional test case using real signals is contained in the second subdirectory. Output of dbcorrelate can be compared to that using SAC there.

Appendix 1: Flowchart of relocation procedure, starting from scratch.