IMPERIAL COLLEGE LONDON, DEPARTMENT OF EARTH SCIENCE AND ENGINEERING; NOW AT:
DEPARTMENT OF EARTH SCIENCES, UNIVERSITY OF CAMBRIDGE

From relative to absolute teleseismic travel-times: the Absolute Arrival-time Recovery Method (AARM)
AARM code user guide

Alistair Boyce

July 13, 2020

1 THE PUBLISHED MANUSCRIPT

This user guide provides support for the absolute arrival-time recovery method (AARM) code distributed upon request used in the following manuscript, when using the code please cite the work as below:

Boyce, A., Bastow, I.D., Rondenay, S., Van der Hilst, R.D., 2017. From relative to absolute teleseismic travel-times: the Absolute Arrival-time Recovery Method (AARM). Bulletin of the Seismological Society of America 107, 2511–2520. doi:10.1785/0120170021

The manuscript describes the methodology (Figure 1.1) for recovery of absolute delay-times from relative arrival-time datasets found by using the multichannel cross-correlation code of *VanDecar and Crosson* (1990). Picks are compared to those catalogued by the International Seismological Centre (ISC) (*Di Giacomo et al.*, 2014; *International Seismological Centre*, 2016). However it can also be generalized easily for use following other trace alignment methods (e.g. *Rawlinson and Kennett*, 2004).

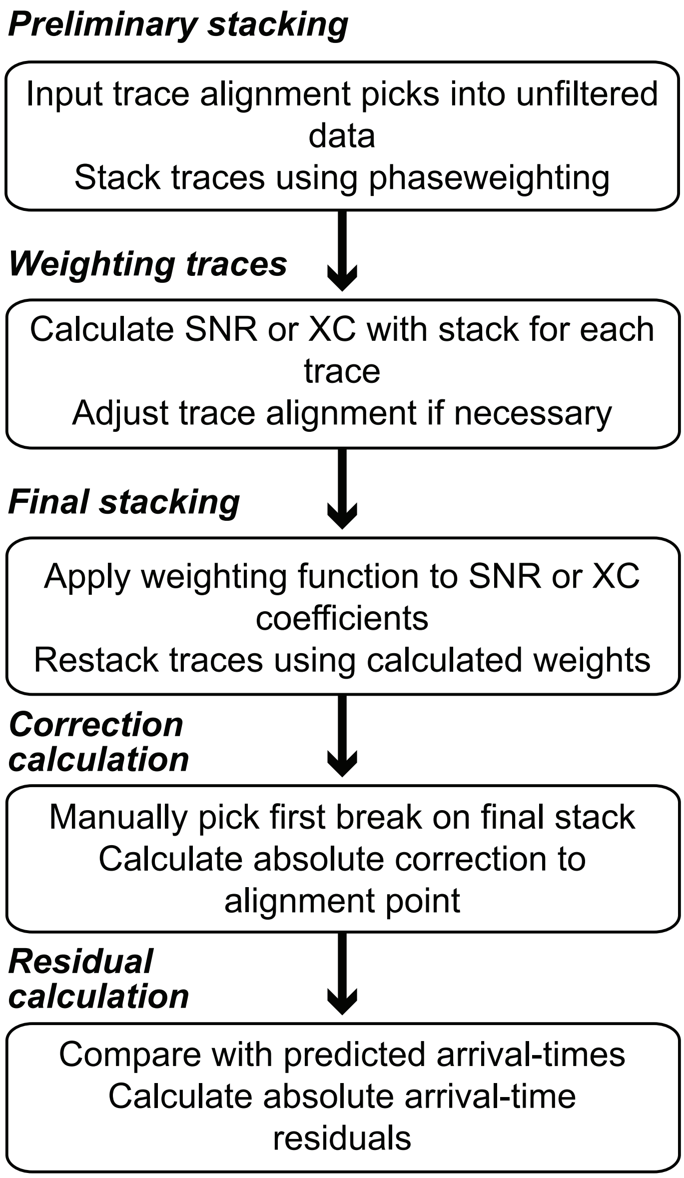

Figure 1.1: A summary of AARM, the recovery of absolute arrival-times using a relative arrival-time dataset.

2 BEFORE WE START - REQUIRED CODES AND FORMATS

A variety of seismological tools are required for use of the AARM code. Due to bugs within my current versions of MacSac (10.6d-grh110/116) and IRIS SAC for linux (101.5c) I use both versions within the code written to work in the terminal in Mac OS X 10 Yosemite, El Capitan, Sierra, Mojave etc. In instances where the variable `$mac` is used this is just to enable the codes to work across both laptop and desktop without changing the paths.
The data I use is initally processed to calculate accurate relative arrival-times using the Multi-channel cross correlation method of *VanDecar and Crosson* (1990) and the predicted arrival-times of the ak135 (*Kennett et al.*, 1995) using the TauP Toolkit (*Crotwell and Ritsema*, 1999). Please refer to *Boyce et al.* (2016) for further detail of this workflow (such as filtering parameters). However it should be easily achieveable to generalize this method for use following another relative arrival-time determination scheme such as *Rawlinson and Kennett* (2004). The codes I use in this toolkit are listed below:

- MACSAC - Version 10.6d-grh110/116 (www1.gly.bris.ac.uk/ george/sac-download.html)
- IRIS SAC - Version 101.5c (ds.iris.edu/ds/nodes/dmc/software/downloads/sac/)
- The Generic Mapping Tools, Version 5.1.1 (r12968) - (gmt.soest.hawaii.edu/projects/gmt/wiki/Download)
- Sac misallaneous tools:
  1. saclst (geophysics.eas.gatech.edu/classes/SAC/)
  2. sactosac (www1.gly.bris.ac.uk/ george/sac-bugs.html)
  3. sac2xy (web.utah.edu/thorne/software.html)

The AARM code outlined here accepts SAC data files in the following format: `YYYYMMDDHHMMSS_NETWORK_STATION.00.?HZ`, which corresponds to: `YEARMONTHDAYHOURMINUTESECOND_NETWORKCODE_STATIONCODE.LOCATION.COMPONENT`. The component BHZ is used as instrument response is removed prior to the analysis to normalize all data to the shortest period response on the network (see *Boyce et al.*, 2016). Laterly I have been normalizing waveform responses to station EKB: `http://ds.iris.edu/mda/BN/EKB/--/`. The AARM code can be changed relatively easily to accept other component choices. Event directories containing these SAC files are also assumed to be named using the `YYYYMMDDHHMMSS` 14 character format. The notable sac header placements are T0 for relative arrival-time derived alignment points (output of MCCC in my case (*VanDecar and Crosson*, 1990) and T1 for ak135 predicted arrivals. Note that these alignment points must be present in the unfiltered, instrument response corrected input data for this code.

3 GETTING STARTED

Retrieve AARM from my GitHub page: (https://github.com/alistairboyce11/AARM) or alternatively unzip the toolkit if you have a tarball:

```
>>tar -zxvf Absolute_arrival-time_toolkit.tar.gz
```

Move the contents of the resulting directory `Absolute_arrival-time_toolkit` to the location of your choice. The main script *calc_arr_times.sh* is written to work in the directory containing all event directories (YYYYMMDDHHMMSS) but others can be placed elsewhere. Open all scripts *(*.sh)* and edit all lines begining with */Users/* as these refer to your specific paths - make sure these are correct first! The SAC macro *normalize.m* must be placed within a directory which both you versions of SAC can find it. Also check the correct bash profile is being sourced in the first few lines. Finally make all script files executable using

```
>> chmod u+x *.sh
```

3.1 ISC REFERENCE PICKS

Within the directory *ISC_FILES*, the script (*TT_calculator_ISC.sh*) can be used to convert a file in ISC download format (e.g. *ISC_picks_2007-2014.txt*) using python to the format readable by the main script here (e.g. *ISC_reference_picks.txt*). This also uses a standard JWEED format event file for reference (e.g. *correct_evts_304.txt*) which are the events previously processed to obtain relative arrival-times. If this step is not done the ISC pick check part will be skipped.

3.2 WEIGHTING FUNCTION

Open the script *AARM_weighting_function.sh* and select the desired function by changing the parameter `$FUNC_NUM`. The default is 1, i.e. a linear weighting. As mentioned within the main manuscript, our testing showed that adding a non-linear weighting function to the final stack did not show any visible improvement and thus a linear weighting function is used. The functionality is left in here to aid in future applications of the method where weighting is found to be useful.

3.3 PLOTTING

In *AARM_Make_plots_Dataset.sh* edit the line `Dataset_name=` to reflect your own input dataset. This is used in the title and file name only. The script *AARM_Make_plots_Event.sh* will only produce a subplot of comparison with ISC picks if the correct files are in place following calculation in *calc_arr_times.sh*.

3.4 QUALITY CONTROL MEASURES

The four quality control (QC) parameters can be found near the top of *calc_arr_times.sh*. These values cause the AARM code to notify the user that an individual file exceeds a particular QC parameter at the end of the implementation. Each QC parameter is explained below, please adjust in your own implementation as you see fit.

- `ISC_CUTOFF` - The maximum allowable difference between available ISC picks and those generated using the AARM code.

- `SNR_CUTOFF` - The lowest signal-to-noise ratio for an individual trace that is allowed to contribute to the stack. Traces with SNR values lower than this are assumed not to give reliable arrival-time measurements.

- `XC_CUTOFF` - The maximum allowable offset (in seconds) from zero in the cross correlation of each trace with the final stack. If an individual trace has a greater `XC_CUTOFF` it is not contributing constructively to a sharp pulse onset and should thus be removed.

- `AUTO_CUTOFF` - The maximum allowable autocorrelation-estimated pick error, traces with pick errors greater than this can be easily removed.

4 Using the code

4.1 Initiating the code

Your data should now be ready to use with the AARM code. As explained in the main manuscript (*Boyce et al.*, 2017) there are two methodologies for weighting the individual traces in the final stack: Cross correlation (XC) and signal-to-noise (NOISE). The choice is input as the first argument ($1) to the code. The second argument ($2) indicates whether the plotting scripts for each event and the whole dataset are to be called (YES/NO). An optional thrid argument ($3) enables the user to specify a rerun of the code (RERUN) for only the selection of events that contained traces that were removed in QC. These events will be listed in *bad_events.txt* after an inital run. Example implementations are shown below:

```
>> calc_arr_times.sh <SCHEME> <PLOTTING> <RERUN>
>> calc_arr_times.sh XC NO
>> calc_arr_times.sh NOISE YES RERUN
```

4.2 Manual input

For each event one user input is required. This is picking the first breaking onset time on the final stack. Once initated the AARM code will run for the first event and sac will initate an X11 window showing the final stack for the first event. The user is then required to pick the onset time. To do this simply hover the mouse over the X11 window in the desired location and press “a” on the keyboard:

```
>> a
```

You can adjust your pick by pressing `a` in a different location if required. As with viewing any files in SAC using `ppk` it is possible to zoom in and out using `x` and `o`. When you are happy with the pick, press “n” on the keyboard to move the script on to the next event:

```
>> n
```

The AARM code will then calculate the absolute arrival-times and produce the plot for the first event if this funciton is enabled. The process then repeats for the remaining events until the dataset plot is formed and the code exits. If you do not wish to place a pick on the stack given high SNR or ambiguity around the location of the onset time, do not include an a on the stacked trace. This event will be moved to a `POOR` directory. Please see description of C1.10.1 below.

5 OUTPUTS

5.1 TO TERMINAL

The code outputs text to the terminal window indicating the files which have failed the quality control criteria in section 3.4. The files to be removed are contained within the file *bad_files.txt* within the directories found in *bad_events.txt*. Use the command:

```
>> while read line; do ‘echo $line‘; done<bad_files.txt
```

The files that failed the QC criteria will then be zipped and the analysis can be rerun, but only for the events which have had traces removed. To do this, rerun your inital implementation but add “RERUN” as the third argument in the command line. For example:

```
>> calc_arr_times.sh NOISE YES RERUN
```

5.2 TO .SAC FILE

There are three SAC file outputs from the AARM code for each event:

- *stack.sac* - The inital unweighted stack.
- *stack2.sac* - The final weighted stack (on which the onset time is picked).
- *stack2_auto.sac* - The autocorrelation of the final stack.

5.3 TO .TXT FILE

The AARM code outputs ten textfiles to record statistics, calculations and most importantly the absolute arrival-time residuals. Many of these are used by the plotting scripts.

- *auto_corr_errors2.txt* - The autocorrelation-estimated picking error.
- *Conv_stack-ISC_picks.txt* - The difference between calculated absolute arrival-times (from AARM) and those derived from the ISC (*Di Giacomo et al.*, 2014; *International Seismological Centre*, 2016). **Convention:** negative if calculated arrival-time is less than the ISC time.
- *correlation_sort2.txt* - The cross correlation coefficients between the stack and each trace and the offset in XC maximum from zero, sorted by XC max offset.

- *correlation_weightings.txt* - Record of calculation of stack weights for individual traces.
- *d.txt* - Absolute arrival-time residuals (s), arrival-time compared to ak135. **Convention:** negative if calculated arrival-time (from AARM) is less than predicted.
- *SNR.txt* - Estimated SNR for each trace contributing to the stack.
- *SNR_pick_error.txt* - Mean SNR and mean autocorrelation error for use in plotting.
- *stack2_SNR.txt* - Estimated SNR of the final stack.
- *TT_calc_results.txt* - Reocrd of arrival-time calculations for each trace. Columns are: Filename, omarker, rel-arr alignment, Predicted arrival, absolute arrival-time, predicted arrival-time, absolute arrival-time residual.
- *XC_means2_.txt* - Mean SNR and mean cross correlation coefficient for use in plotting.

5.4 To postscript file

The individual plot for each event will be named as `AARM_YYYYMMDDHHMMSS.ps`. The dataset plot will be contained within the directory above, from which the main code is run (`$Dataset_name.ps`).

6 Example Data

The directory `EXAMPLE_DATA` contains sample data used in the study of *Boyce et al.* (2016). This section outlines the processes required to calculate absolute arrival-times for these two events `20071008171037, 20090809105556`. Navigate into the directory containing the example data and move the main script to that location:

```
>> cd EXAMPLE_DATA
>> mv ../calc_arr_times.sh .
```

Start the AARM code using either XC or NOISE weighting schemes then press `a` followed by `n` when the final stack appears for each event (two in this case):

```
>> calc_arr_times.sh XC YES
a
n
a
n
```

You will notice (depending on parameters used) that both directories (`20071008171037, 20090809105556`) may have some traces that do not pass the default QC criteria as shown in the terminal window. These can be removed (zipped, to prevent further analysis) using:

```
>> while read line; do ‘echo $line‘; done<bad_files.txt
```

Then the AARM code can be rerun using your inital parameters but only operating on the events for which traces have been removed from the analysis:

```
>> calc_arr_times.sh NOISE YES RERUN
```

You will notice that the AARM code only reruns the analysis for the earthquakes that have bad individual traces, this is to save time when processing larger datasets. Now all data pass the QC criteria as the file *bad_file.txt* is not displayed as it is empty.

Go ahead and take a look at the text, SAC and postscript files produced by the code. You may need to adjust the X-Y scaling on some plots to make the data visible depending on dataset size.

7 Explanation of sections within each code - detailed comments

7.1 Calculate arrival-times - Shell Script - C1

`calc_arr_times.sh`

- C1.1 - Specifies path locations, removes files from a previous implementation, sets quality control limits and checks code has been called correctly by the user. The event loop criteria is also setup here, either all events, individual events or a rerun.
- C1.2 - Loop is started for each event in the $file_list, old files removed, saclst function is used to read Omarker, T0, T1 headers from original sac files *(*.?HZ)*
- C1.3 - Prepare files before stacking: (MacSac) traces are cut around their alignment point to 60 s (P-waves) in length (change to 120 s for S-waves), synchronized (optionally integrated to displacement) and saved as *.stk* files These traces are then normalized using the normalize.m macro (Files **.stk* are overwritten).
- C1.4 - (MacSac) read in normalized files from previous step and calculate RMS value for 25 s noise and signal windows. Write out *.?HZ.stk.s* files. Read **.?HZ.stk* files use signal stacking subprocesses within MacSac to form a phaseweighted stack. Normalize the output stack.
- C1.5 - (Linux Sac) read the stack and each trace *(*.?HZ.stk)*. Correlate each trace with the stack throughout a 20 s window. Store time of min and max in T8 T9 headers, write **.?HZ.stk.corr*. Use SAC headers User0 and User5 to calculate estimate for SNR for each trace - output to file *SNR.txt*.
- C1.6 - Use preset trace SNR cutoff from C1.1 to copy name of low SNR traces to *bad_files.txt* this is used later to remove traces in a rerun of the script. **Recently moved to after C1.10.1.**

- C1.7 - Collect cross correlation coefficients and location of XC maximum. This is used for weighting and calculating any shifts that can be used to improve the alignment. Use GMT to compute mean of XC coefficients.

- C1.8 - When the noise weighting scheme is used: Calculate normalized SNR values, paste to *SNR_norm.out*. Use the normalized SNR values as input into weighting function (default is a linear distribution) (*weight_function.sh*). These weightings are then written to a SAC macro (*addstack.m*) to be used in the second phaseweighted stack.

- C1.9 - When the XC weighting scheme is used: Calculate normalized XC values, paste to *correlation_norm.out*. Use the normalized XC values as input into weighting function (default is a linear distribution) (*weight_function.sh*). These weightings are then written to a SAC macro (*addstack.m*) to be used in the second phaseweighted stack. Use the T9 header from *correlation.out* to adjust the T0 header in the files **.?HZ* thus slightly adjusting the alignments where necessary (*correct_T0.m*). Normalize and write out SAC macro *norm_stk2.m*. Use the above macros to new T0 adjusted files. Output normalized updated files for stacking **.stk2*.

- C1.10 - (MacSac) Use signal stacking subprocesses on **.stk2* add files to stack using weightings in *addstack.m*. Stack as before but saving an "improved" second stack (*stack2.sac*) and normalize. Prompt manual picking of first break on second stack (**USE a**). Again input RMS values into headers for SNR calculation.

- C1.10.1 - Use sachdrinfo to check if the header within the stack is undefined, i.e. user does not wish to pick this event. Move the whole event to POOR directory. Section C1.6 moved here so that poor events are not included in the `bad_files.txt` zip prompt when the script terminates.

- C1.11 - (Linux Sac) read the second stack *(stack2.sac)* and each updated trace *(*.stk2)*. Correlate each trace with the stack throughout a 20 s window. Store time of min and max in T8 T9 headers, write **.stk2.corr2*. Collect new cross correlation coefficients and location of XC maximum. This is sorted by XC maximum offset to give file *correlation_sort2.txt*

- C1.12 - Using preset XC offset maximum ($XC_CUTOFF) paste files above this limit to *bad_files.txt* by running through *correlation_sort2.txt*.

- C1.13 - On the prior correlation of stack with itself (autocorrelation) in C1.11, mark position of the maximum cross correlation value between the stack and the trace for each station. This will be offset from zero and thus acts as a picking error estimate. Using preset autocorrelation pick error estimate limit ($AUTO_CUTOFF) paste files above this limit to *bad_files.txt* by running through *auto_corr_errors2.txt*. Use GMT to paste mean trace SNR and mean pick error to *SNR_pick_error.txt*.

- C1.14 - Collect the amarker from *stack2.sac*, this is the correction to be applied T_{corr} in Eq 7 of *Boyce et al.* (2017). Calculate the SNR for the second stack in *stack2_SNR.txt*. Collect Omarker T0, T1 headers from **.?HZ.new* to enable arrival-time calculations.

Calculate arrival-time (TT), predicted arrival-time (ak135) and absolute arrival-time residual following Eq 7 of *Boyce et al.* (2017).

- C1.15 - For each station check whether an ISC pick is available, calculate the difference between the official pick and the one calculated using this code, paste to *Conv_stack-ISC_picks.txt*. If this is above the preset value ($ISC_CUTOFF) paste the file name to *bad_files.txt*. This is often commented for networks without a permanent station against which we can compare AARM picks.

- C1.16 - use GMT to calculate statistical parameters for each event. Mean trace SNR and mean cross correlation coefficient for use in plotting.

- C1.17 - Call plotting scripts for individual event and data set.

- C1.18 - Output the contents of *bad_files.txt* to the terminal. The user can now zip those individual files by calling:

```
>> while read line; do ‘echo $line‘; done<bad_files.txt
```

7.2 THE WEIGHTING FUNCTION - SHELL SCRIPT - C2

`AARM_weight_function.sh`

- C2.1 - This specifies input and output locations and checks whether code has been called with correct usage.

- C2.2 - Specify here which weighting function ($FUNC_NUM) is to be used. Default is 1 as testing found this to be ineffective for our test datasets. The functionality remains for future use.

$$y1 = x \tag{7.1}$$
$$y2 = (sin((\pi * x) - \pi/2) + 1)/2 \tag{7.2}$$
$$y3 = exp(-0.5 * (log(x)).^2) \tag{7.3}$$
$$y4 = exp(-(log(x)).^2) \tag{7.4}$$
$$y5 = exp(-2 * (log(x)).^2) \tag{7.5}$$
$$y6 = exp(0.5 * (log(x)).^3) \tag{7.6}$$
$$y7 = exp((log(x)).^3) \tag{7.7}$$
$$y8 = exp(2 * (log(x)).^3) \tag{7.8}$$
$$y9 = -((x-1).^4) + 1 \tag{7.9}$$
$$\tag{7.10}$$

7.3 Produce summary plot for each event - GMT 5 Shell Script - C3

`AARM_Make_plots_Event.sh`

- C3.1 - Sets up plotting standards, outlines required files and formats and specifies their location.
- C3.2 - Plots weightings for all traces along the chosen weighting function, displays % of traces with weighting > 0.6 (lower left).
- C3.3 - Uses *sac2xy* code to plot primary stack (blue) and secondary, weighted stack (red) and final stack SNR (upper left).
- C3.4 - Plots histogram of absolute arrival-time residuals (upper middle).
- C3.5 - Plots distribution of Absolute arrival-time residuals with respect to epicentral distance. Station names are plotted below (lower middle).
- C3.6 - Plots a histogram of autocorrelation errors (upper right).
- C3.7 - Plots the difference in residual between code-derived pick and the ISC pick for available stations. This will not appear in cases where picks are not available. The correct file must have the appropriate format and be specified in this section under $ISC_FILE.

7.4 Produce dataset plot for all events - GMT 5 Shell Script - C4

`AARM_Make_plots_Dataset.sh`

- C4.1 - Sets up plotting standards, outlines required files and formats and specifies their location.
- C4.2 - Plots absolute arrival-time residual distribution for the entire data set including some statistics (lower left).
- C4.3 - Plots histogram and statistics for autocorrelation errors for the entire data set (upper left).
- C4.4 - Plots the difference in residual between code-derived pick and the ISC pick for available stations for the entire dataset (upper middle).
- C4.5 - Plots the mean autocorrelation error for each event with the mean trace SNR ration from each event (lower middle).
- C4.6 - Shows the mean cross-correlation with of each trace with the stack against the mean trace SNR (lower right).
- C4.7 - Plots the SNR of the stack against the mean trace SNR (upper right).
- C4.8 - Computes statistics of the data set and prints to terminal output.

7.5 Normalizes individual SAC files so the maximum amplitude is 1. - SAC macro - C5

`normalize.m`

References

Boyce, A., I. D. Bastow, F. A. Darbyshire, A. G. Ellwood, A. Gilligan, V. Levin, and W. Menke (2016), Subduction beneath Laurentia modified the eastern North American cratonic edge: Evidence from P wave and S wave tomography, *J. Geophys. Res.*, *121*(7), 5013–5030, doi: 10.1002/2016JB012838.

Boyce, A., I. D. Bastow, S. Rondenay, and R. D. Van der Hilst (2017), From relative to absolute teleseismic travel-times: the Absolute Arrival-time Recovery Method (AARM), *Bull. Seis. Soc. Am.*, *107*(5), 2511–2520, doi:10.1785/0120170021.

Crotwell, T. J. O., H. P., and J. Ritsema (1999), The TauP Toolkit: Flexible seismic travel-time and ray-path utilities, *Seis. Res. Lett.*, *70*(2), 154–160, doi:10.1785/gssrl.70.2.154.

Di Giacomo, D., D. A. Storchak, N. Safronova, P. Ozgo, J. Harris, R. Verney, and I. Bondár (2014), A New ISC Service: The Bibliography of Seismic Events, *Seis. Res. Lett.*, *85*(2), 354–360, doi: 10.1785/0220130143.

International Seismological Centre (2016), *On-line Bulletin*, Internatl. Seismol. Cent., Thatcham, United Kingdom, http://www.isc.ac.uk.

Kennett, B. L. N., E. R. Engdahl, and R. Buland (1995), Constraints on seismic velocities in the earth from traveltimes, *Geophys. J. Int.*, *122*(1), 108–124, doi:10.1111/j.1365-246X.1995.tb03540.x.

Rawlinson, N., and B. Kennett (2004), Rapid estimation of relative and absolute delay times across a network by adaptive stacking, *Geophys. J. Int.*, *157*(1), 332–340, doi:10.1111/j.1365-246X.2004.02188.x.

VanDecar, J., and R. Crosson (1990), Determination of teleseismic relative phase arrival times using multi-channel cross-correlation and least squares, *Bull. Seis. Soc. Am.*, *80*(1), 150–169.