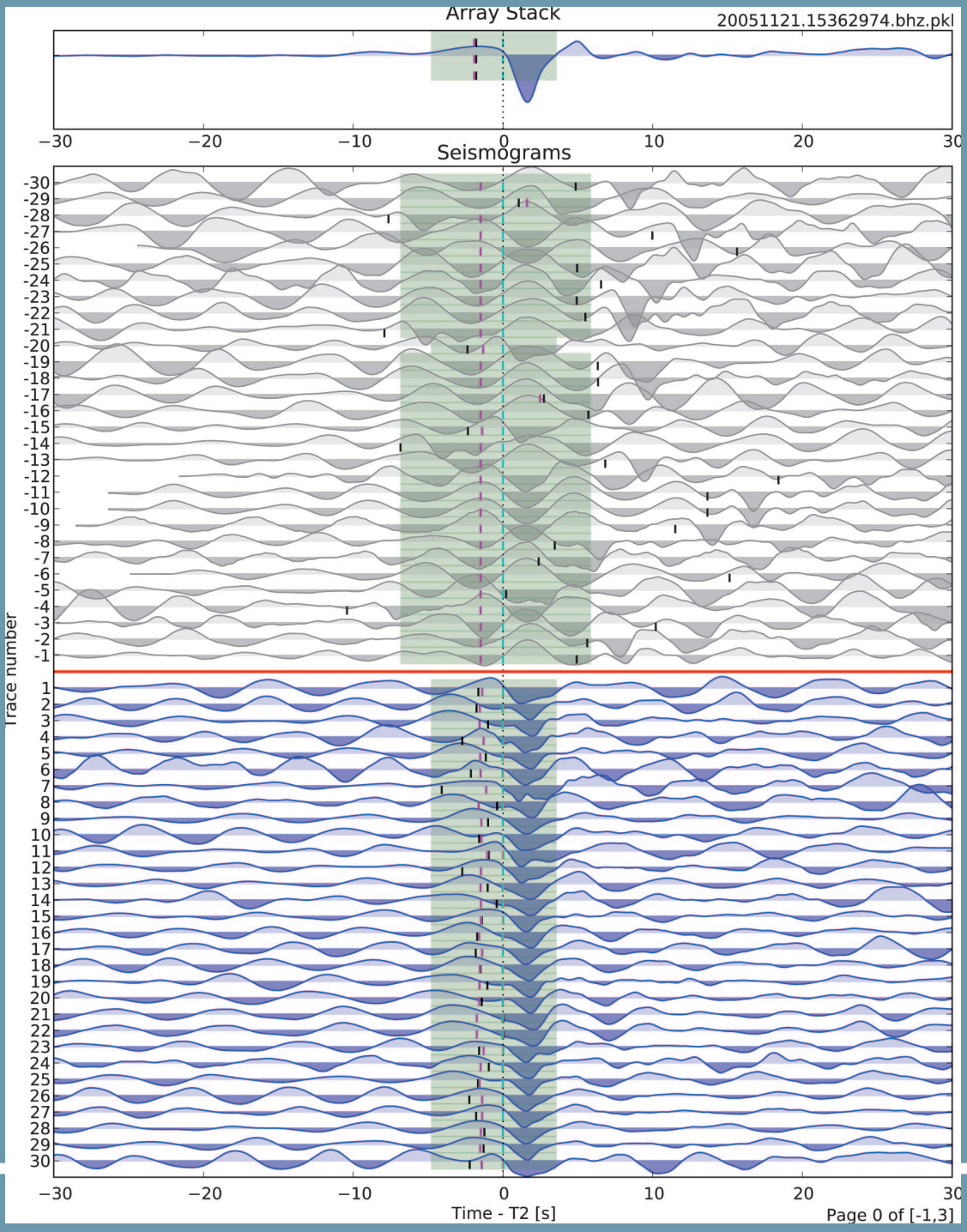
SEISMOLOGICAL RESEARCH LETTERS

Volume 84, Number 1 January/February 2013



SEISMOLOGICAL SOCIETY OF AMERICA

ELECTRONIC SEISMOLOGIST

AIMBAT: A Python/Matplotlib Tool for Measuring Teleseismic Arrival Times

by Xiaoting Lou, Suzan van der Lee, and Simon Lloyd

INTRODUCTION

Seismic travel-time tomography, a common method for studying the Earth's three-dimensional seismic velocity variations, relies on robust measurement of seismic phase arrival times. Absolute arrival times are estimated by visually picking the emergence of the seismic phase of interest on seismograms. Subtracting theoretical arrival times (calculated using a known source location and origin time) from these picked absolute times yields absolute delay times. A widely used method whereby careful picking of absolute arrival times is not required is multi-channel cross-correlation (MCCC), developed by VanDecar and Crosson (1990). In MCCC, phase arrival times relative to an unknown average time are obtained by finding the maximum of the cross-correlation function between each possible pair of seismograms. The preprocessing time of marking the phase to be cross-correlated in MCCC has increased with the tremendous growth in seismic data volume over the past decade. In this paper, we introduce an efficient and robust computer tool, Automated and Interactive Measurement of Body-wave Arrival Times (AIMBAT), to measure teleseismic body-wave arrival times for phases for which arrival time predictions exist. The tool is based on MCCC, but an iterative cross-correlation and stack (ICCS) algorithm replaces the initial phase marking part of the MCCC procedure, which significantly reduces the need for early user labor. The tool is written in Python (http://www.python.org) and utilizes its open-source packages Numpy (http://numpy.scipy.org) and Scipy (http://scipy.org) for numerical array computation and Matplotlib (Hunter, 2007) for two-dimensional plotting and graphical user interface (GUI) applications. We also transcribed the Fortran version of MCCC from VanDecar and Crosson (1990) into Python, which was validated by running both versions on the same data. Python is an efficient, high-level, object-oriented, and platform-independent (Linux, Mac, and Windows) scripting computer language with clean and intuitive syntax (Langtangen, 2004). Its native nested and heterogeneous data structures and comprehensive open-source extensions facilitated the development of this tool.

METHODOLOGY

The ICCS Algorithm

The MCCC procedure requires initial arrival-time estimates for the target seismic phase for the determination of the cross-correlation time window. Because lateral velocity variations are common, windows around the theoretical times predicted with a one-dimensional velocity model do not always capture the desired phase arrival unless a relatively large time window is used. However, a large time window can include signals unrelated to the target phase. Therefore, manual picking of individual arrivals allows a more consistent and appropriately tighter time window around the target signal. All possible pairs of the N stations are then cross-correlated to yield $\frac{N(N-1)}{2}$ time lags from finding the maxima of the crosscorrelagrams. This generates an overdetermined and sparse linear system of $\frac{N(N-1)}{2}$ equations for N arrival times that are relative to an unknown average. It is solved by least-squares inversion to give an optimized set of N relative arrival times (VanDecar and Crosson, 1990). Shifting each seismogram by its respective relative arrival time aligns the seismograms on the target phase.

For large groups of stations, initially aligning tens or hundreds of seismograms for each earthquake by manual phase picking is time consuming. The ICCS algorithm can initially align traces nearly automatically in preparation for the MCCC algorithm. ICCS takes an input time pick I_i $(i = 1, ..., N)$ for the target arrival at the i-th station and a time window W around it, adjusts it iteratively, and saves it to output time pick O_i. This iterative process can be summarized as follows (j is the iteration number):

1. For $j = 0$: Calculate array stack $\mathbf{S}^0$ by stacking all waveforms within the time window W around time pick t_i^0 (the input I_i).
2. For $j > 0$:
   i. Cross-correlate each trace with the array stack $\mathbf{S}^{j-1}$ from the previous iteration to obtain the time lag l_i^j at peak correlation. Time pick t_i^j is updated $(t_i^j = t_i^{j-1} + l_i^j)$.
   ii. Calculate new array stack $\mathbf{S}^j$ by stacking all waveforms within time window W around time pick t_i^j.
   iii. Compare the change of $\mathbf{S}^j$ from $\mathbf{S}^{j-1}$. Stop iterating either when a specified convergence criterion is satisfied or when the number of iterations exceeds a user-defined maximum. Otherwise, go to i.
3. Save the final time pick t_i^j from the last iteration as the output time pick O_i.

doi: 10.1785/0220120033

The time window W around time pick t_i^j, consists of a prepick duration, a postpick duration and a user-defined taper width. It has the same length for each seismogram and each iteration.

The user can choose one of two convergence criteria:

$$1 - \text{corrcoef}(\mathbf{S}^j, \mathbf{S}^{j-1}) < \epsilon \tag{2.1a}$$

or

$$\frac{|\mathbf{S}^j - \mathbf{S}^{j-1}|}{||\mathbf{S}^{j-1}||} < \epsilon \tag{2.1b}$$

where ϵ is a user-defined small number. When formula (2.1a) is applied, a value of $\epsilon = 0.001$ usually produces convergence in less than five iterations.

The input and output time picks of ICCS are stored in user-defined Seismic Analysis Code (SAC; Goldstein *et al.*, 2003) header variables. The ICCS algorithm also calculates a cross-correlation coefficient (CCC), a signal-to-noise ratio (SNR), and a time-domain coherence (COH) for each seismogram and saves them to user-defined SAC header variables. A new SAC file is created to store the waveform of the array stack $\mathbf{S}^j$ with the input (I_s) and output (O_s) time pick means:

$$I_s = \frac{1}{N}\sum_{i=1}^{N} I_i \tag{2.2a}$$

and

$$O_s = \frac{1}{N}\sum_{i=1}^{N} O_i. \tag{2.2b}$$

Combining ICCS with MCCC

We integrate ICCS with MCCC through a four-step procedure using four anchoring time picks ${}_0T_i$, ${}_1T_i$, ${}_2T_i$, and ${}_3T_i$ for each trace ($i = 1, ..., N$) and the array stack ($i = s$). The procedure can be summarized as

1. **Coarse alignment by ICCS.** Using the precalculated theoretical arrival time ${}_0T_i$ at the i-th station and a relatively large time window W_a, the ICCS algorithm (The ICCS Algorithm) iteratively aligns the waveforms by ${}_1T_i$.
2. **Pick the phase arrival time in the array stack.** Time pick ${}_1T_i$ is relative to the predicted arrival time. In order to relate it to an absolute reference time, the user needs to pick the phase emergence time ${}_2T_s$ in the array stack, and ${}_2T'_i$ is defined as ${}_1T_i$ shifted by a constant amount such that

$${}_2T'_i = {}_1T_i + ({}_2T_s - {}_1T_s), \tag{2.3}$$

   where ${}_1T_s$ is the mean of ${}_1T_i$. The user also needs to set a new time window W_b, narrower than W_a, to center on the emerging energy of the target phase.
3. **Refined alignment by ICCS.** The second instance of ICCS refines the coarse alignment using the input time pick ${}_2T'_i$ and the time window W_b. The output time pick is ${}_2T_i$, which is at the phase emergence.
4. **Final alignment by MCCC.** Using ${}_2T_i$ as the input time pick, MCCC is applied within the time window W_b to produce an optimized set of zero-mean relative arrival times R_i ($i = 1, ..., N$). The relative delay time r_i at the i-th station is the residual between the measured (R_i) and predicted relative arrival times:

$$r_i = R_i - ({}_0T_i - {}_0T_m), \tag{2.4}$$

   where ${}_0T_m = \frac{1}{N}\sum_{i=1}^{N} {}_0T_i$ and the ${}_0T_i - {}_0T_m$ term represents the predicted zero-mean relative arrival time. Because ${}_2T_i$ is at the phase emergence, its mean ${}_2T_m = \frac{1}{N}\sum_{i=1}^{N} {}_2T_i$ can be used to estimate the absolute arrival time ${}_3T_i$:

$${}_3T_i = {}_2T_m + R_i. \tag{2.5}$$

   The absolute delay time a_i is the residual between the measured and predicted absolute arrival times:

$$a_i = {}_3T_i - {}_0T_i = r_i + ({}_2T_m - {}_0T_m). \tag{2.6}$$

   The difference term in absolute and relative delay times, ${}_2T_m - {}_0T_m$, represents the event mean delay time for all N traces. It has the potential to recover velocity heterogeneity that would be underestimated in traditional relative delay time tomography.

The procedure described above is an ideal case when all traces are used. Seismic data quality control is an important step in practical data processing so that traces with incoherent signal or low SNR are removed. As we discuss later, our tool's GUI allows efficient seismogram quality control. ICCS steps 1–3 described in this section can be performed multiple times during the quality control process or while testing different time windows. In the procedure, the more computationally intensive step 4 needs to be run only once after quality control. The user phase picking on the array stack in step 2 allows the measuring of absolute phase arrival times. Generally, the alignment in step 4 is not significantly different from step 3, but has the advantage of providing quantitative uncertainty estimates.

Cross correlation is a natural way to measure time lag between two similar waveforms. Bungum and Husebye (1971) developed an iterative method to measure travel-time delays for an array of traces. Each trace is cross-correlated with the array beam, which is summed over all traces and updated using time lags obtained in the previous iteration step. The dbxcor program of Pavlis and Vernon (2010) uses a similar iterative procedure with a better array beam estimate. A representative trace is selected by the user for cross-correlating with each trace to calculate a median stack that is used as the initial array beam. Then the array beam is estimated by a robust stacking algorithm that uses a weighting scheme to penalize traces according to the residual vector of each trace and updated through

iteration. As a result, teleseismic body-wave arrival times and relative amplitudes are measured (Pavlis and Vernon, 2010). The ICCS algorithm is similar but not identical to the methods of Bungum and Husebye (1971) and Pavlis and Vernon (2010). The user chooses whether the array stack (beam) is an averaging of normalized traces weighted either evenly or by correlation coefficients. MCCC minimizes any remaining inconsistencies in relative arrival times between each possible pair of traces using a least-squares approach, which also provides quantitative estimates of uncertainties in the delay time for each trace (VanDecar and Crosson, 1990).

In our processing procedure, the required user interaction is limited to quality control and picking the phase arrival and setting the time window in the array stack in step 2. Compared to manually picking the target phase on each seismogram, the automated phase alignment achieved by steps 1–3 of ICCS dramatically reduces user processing time and user error during the preparation for MCCC. Pavlis and Vernon (2010) have found that their method can save about 80% of even a skilled analyst's time. The user time commitment for using our method is similar to that when using the method of Pavlis and Vernon (2010). The latter method is implemented as an extension to database software typically run in seismic network operation centers, while our method is stand alone and highly portable, making it convenient to use by beginning and advanced graduate students.

IMPLEMENTATION

Seismic Data Access

The basis of any seismic data analysis tool is data access, including reading and writing data files. The Python toolbox for seismology, ObsPy, provides useful functions to read and write seismograms in GSE2, SEED/MiniSEED, and SAC formats (Beyreuther *et al.*, 2010). Here we use our own Python package named pysmo.sac to read and write evenly spaced SAC files. The Python class sacfile opens a SAC file and returns an object including data and all SAC header variables as its attributes. Modifications of object attributes are saved to the file. It is written purely in Python so that it also runs with Jython (http://www.jython.org).

Graphical User Interface

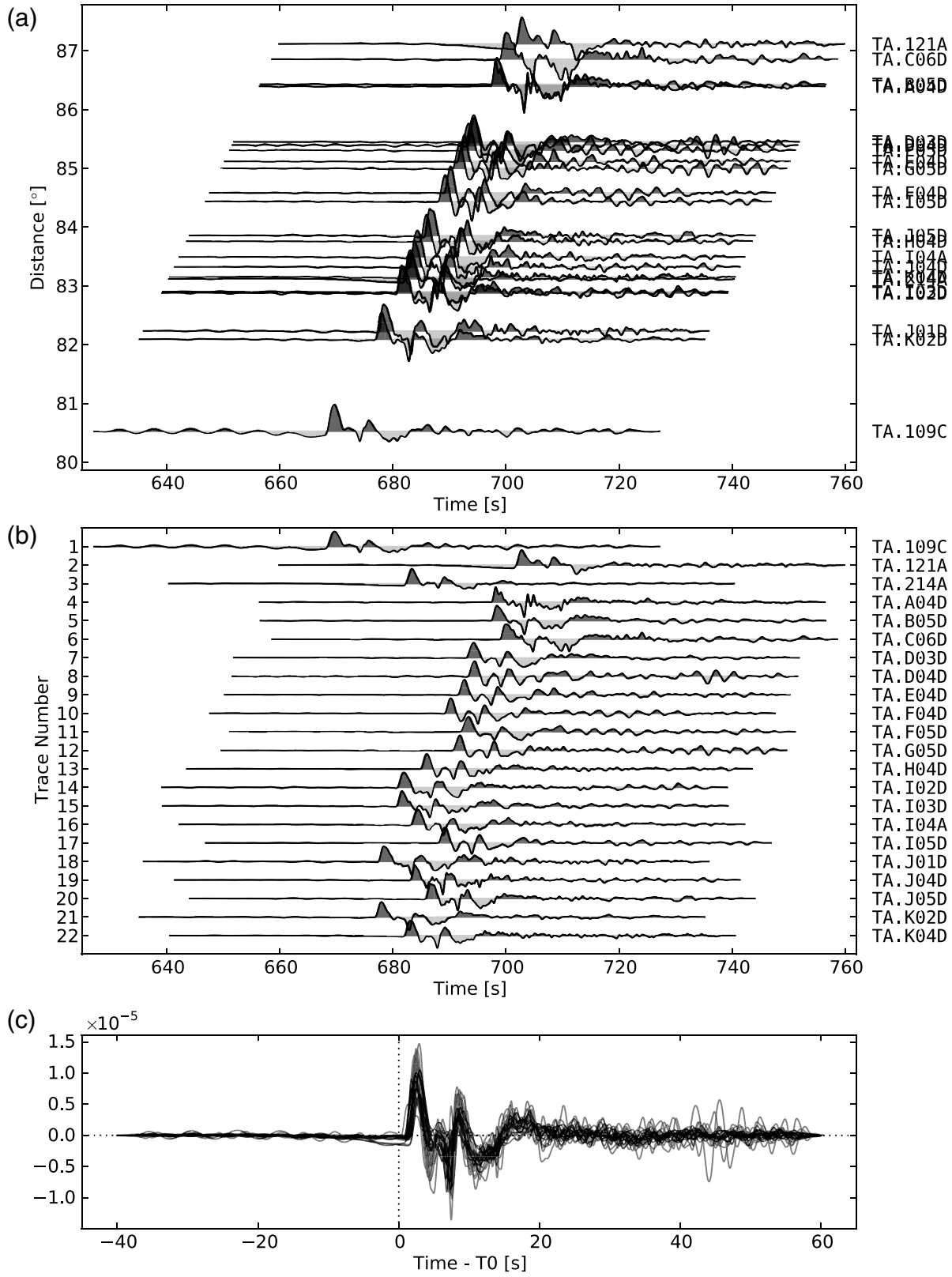
Waveform visualization is also important in seismic data analysis. Users of the MCCC method (VanDecar and Crosson, 1990) tend to use custom SAC macros and shell scripts to sort and align their seismograms in different, useful ways. A large part of quality control in the MCCC method is the result of visual inspection of (a) the data, and (b) the correlation and alignment results. Pavlis and Vernon (2010) developed a graphic interface based on a custom Motif (http://www.openmotif.org) seismic display widget for interactively sorting and (de)selecting seismograms in their dbxcor application within Antelope (http://www.brtt.com/software.html). We use Matplotlib (Hunter, 2007), an open-source two-dimensional plotting library of Python, to plot seismograms. Matplotlib uses a similar syntax to MATLAB (http://www.mathworks.com/products/matlab) in line, marker, font, and axes control.

Matplotlib has GUI-neutral widgets such as Button and SpanSelector and GUI-neutral event-handling application programming interface (API) to support interactive plotting (http://matplotlib.org/api/widgets_api.html and http://matplotlib.org/users/event_handling.html). The event handling API can interpret keyboard and mouse events (such as key_press_event and mouse_motion_event) and receive callbacks (Hunter, 2007). Using these features we have developed several Python/Matplotlib classes and scripts to plot multiple SAC files (Fig. 1). An interactive GUI (Fig. 2) has also been designed for processing seismograms, including running ICCS and MCCC in the procedure described in the Combining ICCS with MCCC section, and, importantly, for quality control. The scripts are executed from the command line with arguments parsed by the Python module optparse. It enhances the scripts' reusability and modularization by allowing multiple command line options, such as how many traces are displayed on the current screen view. Another module ConfigParser is incorporated to set default options and parameters. For example, the user can specify the default colors for waveform, waveform fill, time window, and time picks in a configuration file.

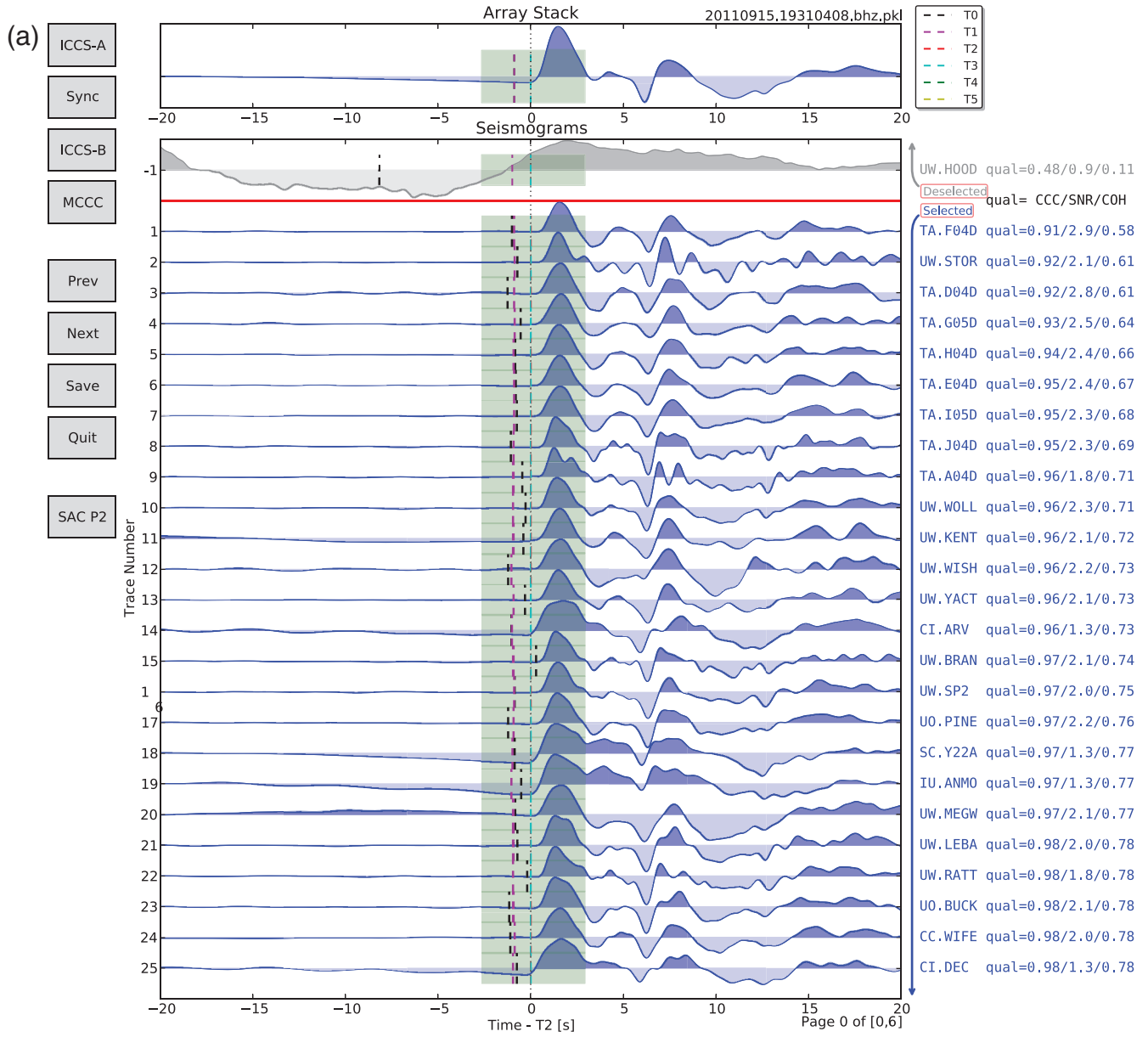
As illustrated by Figure 1, we have replicated SAC's p1 and p2 styles of plotting, as well as record section plotting by SAC's signal stacking subprocess. Similarly, seismograms can be plotted and shifted by azimuth and backazimuth, functions not provided by SAC. Moreover, the event handling API of Matplotlib allows the reproduction of SAC's phase-picking functionality. The user can set a time pick by pressing the *t* key and number keys 0–9. The *x* location of the mouse position is saved to corresponding SAC headers *t*0–*t*9. To zoom in a waveform section, mouse selection of a time span is enabled by the SpanSelector widget instead of SAC's combination of the *x* key and mouse click. The user can press the *z* key to zoom out to the previous range. We do not replicate SAC's *o* key functionality because it is used for another purpose by Matplotlib. An improvement over SAC is that the user can set a time window by pressing the *w* key, which saves the current *x* axis range to two user-defined SAC header variables. A transparent green span is plotted within the time window (Fig. 2). Another improvement is that the program outputs the file name of the seismogram when its waveform is clicked using the mouse. This is enabled by the event-handling API of Matplotlib and was mostly introduced for use in SAC p2 style plotting when seismograms are plotted on top of each other. It is especially useful when a large number of seismograms create challenges in labeling.

Quality Control

Figures 3 and 4 clearly show improved *P*-wave alignment after quality control and processing for seismograms in Figure 2a. Because of the phase arrivals being better aligned after processing (Fig. 4d) than before (Fig. 4a), the narrower and sharper phase emergence (Fig. 4d) leads to a more robust set of absolute arrival-time picks. The high-quality phase arrivals allowing this



▲ **Figure 1.** Example of using the Python/Matplotlib scripts to plot multiple seismograms in: (a) record section, (b) SAC p1 style, and (c) SAC p2 style. Data are from an earthquake that occurred near the Fiji Islands on 15 September 2011 (M_w 7.3) and was recorded by broadband seismic stations in the United States. In the configuration file, the user can change the default color for waveform and waveform fill. This color can also be generated randomly through a command line option. The event handling API of Matplotlib allows the script to display the file name of a seismogram if its waveform is clicked by mouse.



▲ **Figure 2.** (a) Snapshot of the GUI showing aligned teleseismic *P* waves on vertical-component seismograms from the same earthquake as in Figure 1. The seismograms are sorted by cross-correlation coefficient. The page here only shows the first 25 selected and 1 deselected seismograms out of a total of 163 seismograms. The user can click on the Prev and Next buttons to navigate through all seven pages. Clicking the Save button saves changes to files and the Quit button quits processing. The four buttons on the upper left correspond to the procedure described in the Combining ICCS with MCCC section: the ICCS-A button for step 1; the Sync button for step 2; the ICCS-B button for step 3; and the MCCC button for step 4. The screenshot shows the GUI after step 3. The time axis is relative to time pick $_2T_i$, and the time window W_b is shaded with a transparent green. Clicking the SACP2 button produces Figure 4. (b) Similar snapshot but for another earthquake that occurred in Kyushu, Japan, on 21 November 2005 (M_w 6.2). Seismograms are sorted by user-defined weighted average of the three quality factors. Negative signals of the waveform are filled with less transparency. This illustrates a case in which a large portion of seismograms must be removed and how the GUI and seismogram sorting parameters aid in trace selection. (*Continued*)

improvement (Figs. 3 and 4) are the result of quality control in the form of removing traces with low quality and incoherent arrivals. In practice, there is an ongoing need to remove such traces from virtually all earthquake-based sets of seismograms. Figure 2b shows an example in which half of all seismograms from an earthquake are to be discarded. Therefore, we implement a mechanism to conveniently (de)select seismograms utilizing the event-handling API of Matplotlib.

In the processing interface (Fig. 2a or 2b), there are two divisions of selected and deselected seismograms. Selected seismograms with a positive trace number are displayed with blue lines and fills, while deselected seismograms are gray and have a

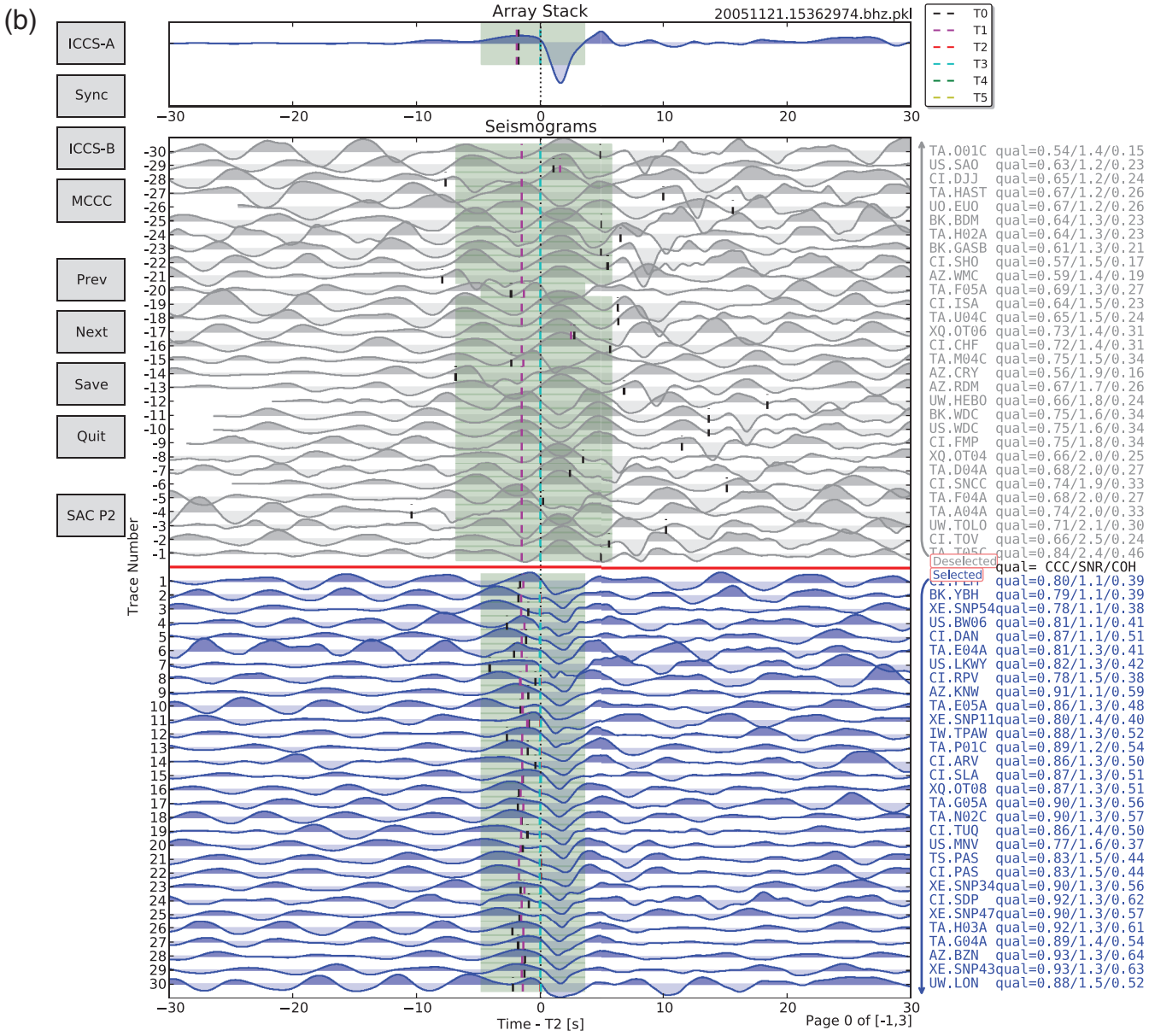


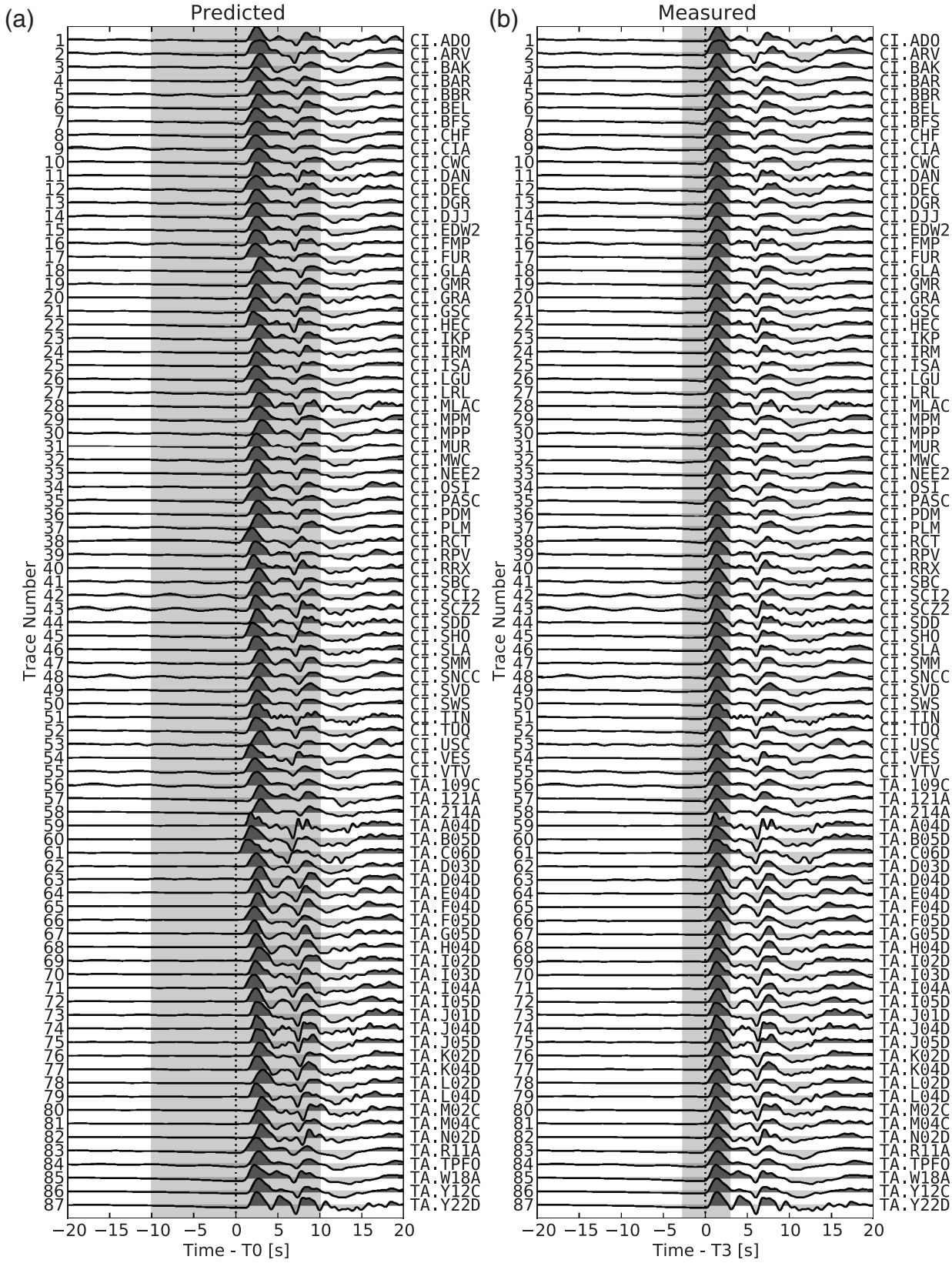
Figure 2. Continued.

negative trace number. The user can simply click on a certain seismogram with low quality to switch the selection status, either to exclude it or bring it back for inclusion.

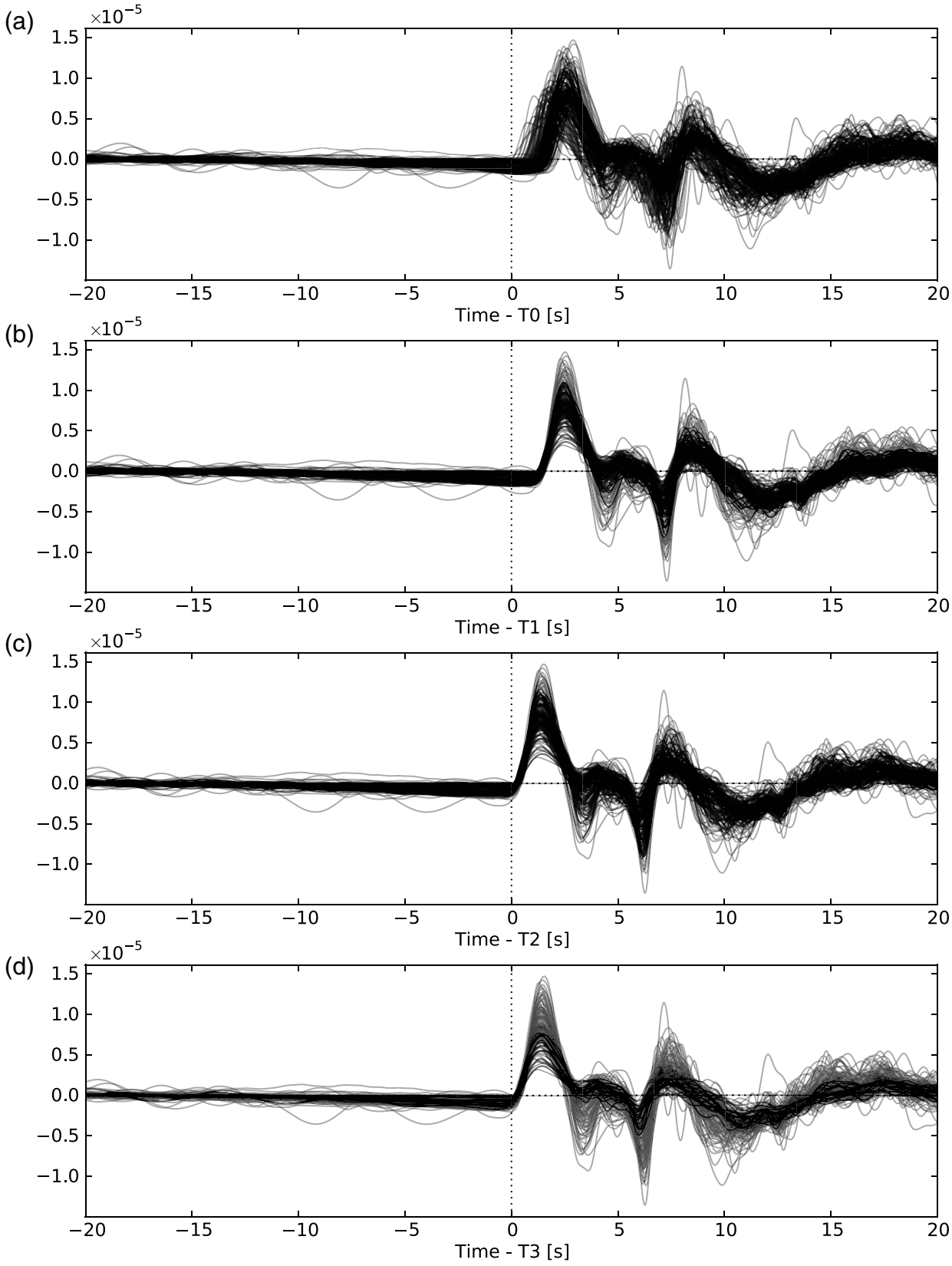
Sorting seismograms by multiple parameters is helpful for efficient trace selection. Similar to Pavlis and Vernon (2010), three parameters (CCC, SNR, and COH) calculated by the ICCS algorithm are used as quality factors to sort the display order of seismograms. By default, seismograms are sorted by a user-defined weighted average of the three factors. Seismograms with low qualities are clustered at the beginning, which makes it easier to deselect them. Optionally, seismograms can be sorted by individual quality factor or another parameter ${}_2T_i - {}_0T_i$, which is the difference between predicted and ICCS-measured arrival times. This parameter can help detect cycle skipping and thus prevent false alignment during cross correlation, especially in medium-quality cases. Other ways to sort are more traditional and include sorting by epicentral distance, azimuth, and backazimuth.

DISCUSSION AND CONCLUSION

The motivation behind developing the tool presented here was to reduce user processing time and user errors in the measurement of teleseismic body-wave arrival times while retaining valuable input based on a user's expertise and a developing seismologist's intuition. The quality control part is interactive, while the phase-picking part is automated. The ICCS algorithm aligns seismograms efficiently in a nearly automated way before applying MCCC. It replaces time-consuming manual phase picking with computer work. Both absolute and relative arrival



▲ **Figure 3.** Comparison of teleseismic *P*-wave alignment by (a) 1D-predicted and (b) measured arrival times from the processing illustrated in Figure 2a. Only the first 87 traces are displayed. Two time windows, for the coarse and refined ICCS alignments described in the Combining ICCS with MCCC section ($W_a(-10, 10$ s) and $W_b(-2.7, 2.9$ s)), are plotted with transparent gray shades in (a) and (b), respectively.



▲ **Figure 4.** Comparison of teleseismic *P*-wave alignment by four time picks: (a) ${}_0T_i$, (b) ${}_1T_i$, (c) ${}_2T_i$, and (d) ${}_3T_i$, respectively processed in Figure 2a. Alignment by ${}_1T_i$ clearly improves compared to ${}_0T_i$ which is the 1D-predicted arrival time. ${}_2T_i$ is shifted from ${}_1T_i$ and refined to be positioned at the phase emergence for the purpose of measuring absolute arrival times. ${}_3T_i$ is obtained by MCCC. The difference between ${}_3T_i$ and ${}_2T_i$ is not significant and thus not visible at this scale.

times are obtained based on cross correlation. This tool significantly improves efficiency and quality in measuring teleseismic body-wave arrival times. User interaction is needed only to pick the target phase arrival and to set a time window in the array stack. A GUI with a palette of options facilitates quality control.

We chose Python to implement the algorithms because it is a powerful platform-independent scripting language, and its extensive scientific modules precluded our needing to code every process from scratch. The open-source nature of the language and its modules eliminates licensing issues and makes our tool, AIMBAT, highly portable and accessible to a wide audience. This also allows other users to modify and enhance this module-structured tool. Our tool's GUI relies on the object-oriented and interactive features of Python and Matplotlib. GUI-neutral widgets and the event-handling API of Matplotlib allow efficient, interactive seismic data processing and quality control. As a byproduct, the SAC's phase picking and plotting functionalities are replicated and enhanced.

PACKAGE DISTRIBUTION

AIMBAT is released as a subpackage of pysmo in the name of pysmo.aimbat along with another subpackage pysmo.sac, which is for SAC file access and manipulation. The latest releases of pysmo.sac and pysmo.aimbat are available for download at http://www.earth.northwestern.edu/~xlou/aimbat.html and at github (https://github.com/pysmo/sac and https://github.com/pysmo/aimbat).

ACKNOWLEDGMENTS

Development of the GUI part of AIMBAT was inspired and initiated at the 2009 EarthScope USArray Data Processing and Analysis Short Course (http://www.iris.edu/hq/es_course/). We thank Gary Pavlis for co-organizing this short course, for sharing the work of Pavlis and Vernon (2010) at an early stage, and for thoughtful comments on the manuscript that helped us clarify this paper. We also thank Trevor Bollmann and Mike Witek for testing AIMBAT and Carl Ebeling, Trevor Bollmann, and Emily Wolin for comments on the manuscript. This research was supported by NSF Grant EAR-0645752 to Suzan van der Lee.

REFERENCES

Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann (2010). ObsPy: A Python toolbox for seismology, *Seismol. Res. Lett.* **81,** no. 3, 530–533, doi: 10.1785/gssrl.81.3.530.

Bungum, H., and E. Husebye (1971). Errors in time delay measurements, *Pure Appl. Geophys.* **91,** no. 8, 56–70, doi: 10.1007/BF00879557.

Goldstein, P., D. Dodge, M. Firpo, and L. Minner (2003). 85.5 SAC2000: Signal processing and analysis tools for seismologists and engineers, in *International Handbook of Earthquake and Engineering Seismology*, International Geophysics, Vol. 81, Part B, W. Lee, H. Kanamori, P. C. Jennings, and C. Kisslinger (Editors), Elsevier, Amsterdam, The Netherlands, 1613–1614, doi: 10.1016/S0074-6142(03)80284-X.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment, *Comput. Sci. Eng.* **9,** no. 3, 90–95, doi: 10.1109/MCSE.2007.55.

Langtangen, H. (2004). *Python Scripting for Computational Science*, Second Ed., Springer Verlag, Berlin, Germany.

Pavlis, G. L., and F. L. Vernon (2010). Array processing of teleseismic body waves with the USArray, *Comput. Geosci.* **36,** no. 7, 910–920, doi: 10.1016/j.cageo.2009.10.008.

VanDecar, J., and R. Crosson (1990). Determination of teleseismic relative phase arrival times using multi-channel cross-correlation and least squares, *Bull. Seismol. Soc. Am.* **80,** no. 1, 150–169.

Xiaoting Lou
Suzan van der Lee
Simon Lloyd[1]
Department of Earth and Planetary Sciences
Northwestern University
2145 Sheridan Road
Evanston, Illinois 60208 U.S.A.
xlou@earth.northwestern.edu

[1] Now at the Department of Meteorology and Geophysics, University of Vienna, UZA II, Althanstraße 14, 1090 Vienna, Austria.