Relative source location based on coda wave interferometry (CWI)

MATLAB code user manual with examples

Youqian Zhao[1] and Andrew Curtis[1,2], 2018

[1] School of Geosciences, University of Edinburgh, Edinburgh

[2] Department of Earth Sciences, ETH Zurich, Switzerland

Summary:

This user manual accompanies the *coda wave interferometry (CWI) relative source location MATLAB code package*. The code estimates the relative location of a cluster of events of similar source mechanisms using inter-source separations estimated with CWI (Snieder, 2006). The advantage of this location technique is that the location result is insensitive to the number and distribution of seismic stations. The method is particularly useful where there are not a large number of seismic stations with a good event-sensor azimuthal coverage so that the performance of the conventional double-difference relative source location method of Waldhauser & Ellsworth (2000) deteriorates. It also provides an alternative, entirely independent method with which conventional methods can be compared in order to assess relative location uncertainties.

This manual provides a brief introduction of CWI and the location algorithm. This is sufficient to run the code. More details are given in the accompanying paper. The package consists of three sections, each of which contains codes that conduct one step of the location method: 1) classifying events into different clusters with given waveforms recorded by one or multiple seismic station channels, 2) estimating inter-source separations with the CWI method, and 3) estimating the relative event locations from the separation data. The code solves for the event locations as an optimization problem, where the most likely set of event locations are found where an objective function attains its minimum. This algorithm takes account of the known biases of CWI-estimated source separations, and is able to correct for them to a significant extent in location results. Examples on synthetic and real data are presented in this manual to demonstrate how to use the codes, and test-data is included in the package. For real Earth problems where the actual event locations are unknown, we propose a way to assess the quality of location results.

Contents:

Summary

1. Introduction
    1.1 Clustering
    1.2 Estimating inter-source separations
    1.3 Source location
2. Package contents
3. Codes and Examples
    3.1 Clustering
        3.1.1 Code description
            Crosscorrelation
            Identify clusters
        3.1.2 Example (New Ollerton earthquakes)
    3.2 Estimating inter-source separations
        3.2.1 Code description
            Pick first arrival of waveforms
            Determine parameters for estimating source separations
            Estimate source separations
        3.2.2 Example (New Ollerton earthquakes)
    3.3 Location
        3.3.1 Code description
        3.3.2 Synthetic example
        3.3.3 Example (New Ollerton earthquakes)

References

Appendix 1 Descriptions of codes and variables for clustering

Appendix 2 Descriptions of codes and variables for estimating inter-source separations

Appendix 3 Descriptions of codes and variables for location

1. Introduction

This manual accompanies the *coda wave interferometry (CWI) relative source location MATLAB code package*. The package implements source location estimation in three steps. First, the events are classified into different clusters based on recorded waveform similarity. Second, for each cluster, inter-source separations between all available event pairs are estimated with CWI. Finally, the relative event locations are obtained from an optimization problem. Using the code to estimate event locations does not require a thorough understanding of the technique, however users are recommended to refer to Snieder (2006), Robinson et al. (2013), and the accompanying paper Zhao and Curtis (2018) to gain a full understanding. The remainder of this section provides an introduction to the technique used in each step. For a quick start, users can implement the whole process by running a single script `Main_running_script.m` in directory `codes` using the test-date set provided, or after editing to use their own data.

1.1 Clustering

CWI uses scattered waves in seismograms to estimate the separations between pairs of source locations, which are then used jointly to determine the relative locations between all events. It has been proved that the location results of CWI-based technique are insensitive to the number and distribution of seismic stations, and it is even possible to estimate event locations with data from a single station channel (Robinson et al., 2013; Zhao et al., 2017). In order to estimate inter-source separations with CWI, it is required that the sources have identical source mechanism (Snieder, 2006). Therefore, it is essential to classify the events into clusters, within each of which,

all the events have very similar source mechanisms so that CWI can be applied. The similarity in source mechanism can be assessed by waveform similarity, which is measured by crosscorrelation between pairs of waveforms. The package classifies events in two steps: computing crosscorrelations and identifying clusters. `sort_cr.m` computes the crosscorrelation of all available waveforms recorded by the same seismic station channel, and sorts the event pairs in a descending order of their correlation coefficients. `clustering.m` then classifies the events into clusters using the sorted crosscorrelations, following the method used by Ottemoller et al. (2017), so that each cluster contains events of similar source mechanisms.

1.2 Estimating inter-source separation with CWI

Coda refers to later part of a seismogram, generated by multiply scattered waves. In spite of its complex appearance with few uniquely identifiable arrivals, coda is extremely sensitive to minute changes in the seismic system. It therefore has the potential to be used for applications involving distinguishing and monitoring small perturbations in seismic systems that are difficult for methods that only use early arrivals of seismograms. Aki (1969) uses the spectrum of coda from local earthquakes to evaluate seismic moments, and more recently Snieder et al. (2002) developed CWI that uses phase information to identify differences between pairs of sources or changes in the propagating medium. The location technique of this package uses inter-source separations between all available event pairs in a cluster as data to estimate the relative locations of the events. This subsection focuses on how CWI estimates inter-source separations.

CWI estimates the inter-source separation between a pair of events by comparing coda of the two seismograms recorded by the same seismic station channel. The theory is based on path summation of scattering waves (Snieder, 1999) - that the total wavefield at a given location can be written as the superposition of waves traveling along all possible trajectories

$$u^1(t) = \sum_T A_T(t), \tag{1}$$

where u^1 is the total wavefield from event 1, T represents a wave trajectory, and A_T is the contribution to the total wavefield of waves that travel along trajectory T. The trajectory of each scattered wave consists of the path from the source to the first scatterer encountered, and the path followed thereafter. For event 2 that is close to event 1 and with very similar source mechanisms, CWI assumes that the paths to the first scatterer change, but that subsequent paths do not because they depend mainly on the medium rather than on the source location. Thus the dominant difference of the recorded waveforms at the same seismic station is in coda arrival times (Snieder, 2006). The wavefield of event 2 can then be written as

$$u^2(t) = \sum_T A_T(t - \tau_T), \tag{2}$$

where τ_T is the travel-time difference of waves traveling along trajectory T due to the difference in source position. Due to the proximity in source locations and the similarity in source mechanisms, the two waveforms will be similar, which can be quantified by the normalized cross-correlation of the two waveforms in a time-window defined with a central time t and a half-width t_ω, computed for a sequence of time-windows in the coda

$$R(t_s) = \frac{\int_{t-t_\omega}^{t+t_\omega} u^{(1)}(t')u^{(2)}(t'+t_s)dt'}{\sqrt{\int_{t-t_\omega}^{t+t_\omega} {u^{(1)}}^2(t')dt' \int_{t-t_\omega}^{t+t_\omega} {u^{(2)}}^2(t')dt'}} \tag{3}.$$

In each time-window, the distribution of the travel-time differences τ_T contains information about the source separation δ. Snieder et al. (2006) estimates the standard deviation of the travel-time difference σ_τ from the maximum of the correlation coefficient R_{max}. σ_τ can then be related to the source separation δ by

$$\sigma_\tau{}^2 = \frac{1}{2}\frac{\delta^2}{v^2} \quad \text{for isotropic sources in a 2D scoustic medium,} \tag{4a}$$

$$\text{or} \quad \sigma_\tau{}^2 = \frac{1}{2}\frac{\delta^2}{v^2} \quad \text{for isotropic sources in a 3D scoustic medium,} \tag{4b}$$

$$\text{or} \quad \sigma_\tau{}^2 = \frac{6/\alpha^8 + 7/\beta^8}{7(2/\alpha^6 + 3/\beta^6)}\delta^2 \quad \text{for double} - \text{couple sources in an elastic medium,} \tag{4c}$$

where α and β are P- and S-wave velocity (Snieder and Vrijlandt, 2005). Since the waves arriving in different time-windows have traveled along different paths, the separation results of each time-window are independent and their variation can be used to estimate uncertainty. It has been proved that inter-source separation estimates are often highly consistent among different station channels (Snieder and Vrijlandt, 2005; Robinson et al., 2011; Zhao et al., 2017).

It is essential to be aware that the CWI technique has an increasing tendency toward underestimation of larger source separation due to the cycle-skipping effect in the correlation of coda in equation 3. This trend can be quantified by two empirical relations between the mean $\mu = \mu(\widetilde{\delta_t})$ and the standard deviation $\sigma = \sigma(\widetilde{\delta_t})$ of CWI separation estimates with the true separations $\widetilde{\delta_t}$ (Figure 1 a, b), where the tilde above separation δ_t indicates that the quantity has been normalized by the dominant wavelength λ_d in the recorded data. The applicable range of CWI is visualized in Figure 1: CWI fails to identify any increase in inter-source separation when the true separation is larger than $0.55\lambda_d$. The empirical functions are derived from a multitude of

synthetic experiments with a large range true separations in different Gaussian random media, by fitting the rational function forms

$$\mu(\widetilde{\delta_t}) = a_1 \frac{a_2 \widetilde{\delta_t}^{a_4} + a_3 \widetilde{\delta_t}^{a_5}}{a_2 \widetilde{\delta_t}^{a_4} + a_3 \widetilde{\delta_t}^{a_5} + 1}, \tag{5a}$$

$$\sigma(\widetilde{\delta_t}) = b_1 \frac{b_2 \widetilde{\delta_t}^{b_4} + b_3 \widetilde{\delta_t}^{b_5}}{b_2 \widetilde{\delta_t}^{b_4} + b_3 \widetilde{\delta_t}^{b_5} + 1} + c, \tag{5b}$$

(Robinson et al., 2011) where the value of the constants are listed in Table 1.

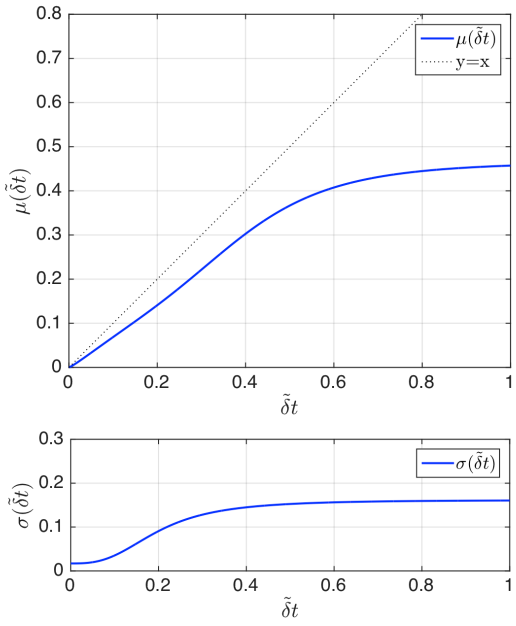


Figure 1: Empirical functions showing the bias and uncertainties of separation estimates of the CWI technique. The upper panel shows the empirical relation between the mean of the source separation $\mu = \mu(\widetilde{\delta_t})$ and the true separation $\widetilde{\delta_t}$. All separations are normalized by their dominant wavelengths. The dotted line labelled $y = x$ shows the case where the mean of the CWI separation estimates are identical to the true separations. The lower panel shows the empirical relation between the standard deviation $\sigma(\widetilde{\delta_t})$ of the separation estimates and the true separation $\widetilde{\delta_t}$.

$\mu = \mu(\widetilde{\delta_t})$	$\sigma = \sigma(\widetilde{\delta_t})$
$a_1 = 0.4661$	$b_1 = 0.1441$
$a_2 = 48.9697$	$b_2 = 101.0376$
$a_3 = 2.4693$	$b_3 = 120.3864$
$a_4 = 4.2467$	$b_4 = 2.8430$
$a_5 = 1.1619$	$b_5 = 6.0823$
	$c = 0.017$

Table 1: Constants used in the empirical relations $\mu = \mu(\widetilde{\delta_t})$ and $\sigma = \sigma(\widetilde{\delta_t})$ in equation 5 (Robinson et al., 2013).

1.3 Source location

The source location algorithm estimates relative location from the separation estimates and their uncertainties in a probabilistic framework. It accounts for the known biases of the CWI separation estimates, and is able to correct for them to a significant extent. Robinson et al. (2011) describe the probabilistic nature of the CWI estimates with a conditional probability density function (pdf) $P(\widetilde{\delta_t}|\tilde{\delta}_{CWI})$ which describes the probability of the true separation being $\widetilde{\delta_t}$ given that the estimate from CWI is $\tilde{\delta}_{CWI}$. According to Bayes' theorem, the posterior probability of $\widetilde{\delta_t}$ is proportional to the likelihood of observing $\tilde{\delta}_{CWI}$ given the true separation $\widetilde{\delta_t}$ multiplied by the prior on $\widetilde{\delta_t}$:

$$P\left(\widetilde{\delta_t}\middle|\tilde{\delta}_{CWI}\right) \propto P\left(\tilde{\delta}_{CWI}\middle|\widetilde{\delta_t}\right) \times P\left(\widetilde{\delta_t}\right) \qquad (6).$$

The prior is used to incorporate information about source separation or event location known prior to the location process, which here is considered to be a uniform distribution with wide bounds. The likelihood function $P\left(\tilde{\delta}_{CWI}\middle|\widetilde{\delta_t}\right)$ is approximated by a positively bounded Gaussian pdf, whose mean and standard deviation are represented by the empirical relations $\mu = \mu(\widetilde{\delta_t})$ and $\sigma = \sigma\left(\widetilde{\delta_t}\right)$ given true separation $\widetilde{\delta_t}$.

For a cluster of events, equation 6 holds for each event pair. Robinson et al. (2013) incorporate the separations between multiple event pairs by multiplying the formulae for all available event pairs to establish the joint posterior pdf, assuming they are independent of each other:

$$P\left(e_1, \dots, e_n\middle|\tilde{\delta}_{CWI}\right) = c\prod_{i=1}^{n} P(e_i) \times \prod_{i=1}^{n-1}\prod_{j=i+1}^{n} P\left(\tilde{\delta}_{CWI,ij}\middle|e_i, e_j\right), \qquad (7)$$

where c is a constant, n is the number of events, $\boldsymbol{e}_i = (x_i, y_i, z_i)$ is event i location, within the last term we use $\delta_{t,ij} = \left\|\boldsymbol{e}_i - \boldsymbol{e}_j\right\|_2$ for source separation $\delta_{t,ij}$ between the i'th and j'th earthquakes, and the prior $P(\boldsymbol{e}_i)$ only contains the relative event locations. Throughout the evaluation of the joint pdf, the separation quantities are used as their normalizations (divided by dominant wavelength λ_d). However, the dominant frequency, hence dominant wavelength of the events can extend over a large range, and is also subject to limitations of recording instruments; using the average dominant wavelength over different station channels therefore introduces inaccuracy to the location process. To this end, this package uses the joint pdf introduced by Zhao et al. (2017), which applies an individual likelihood for each channel when data from multiple channels are used, so that the separations computed during the evaluation of the joint pdf are normalized by the actual dominant wavelengths recorded on that channel

$$P\left(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n \middle| \tilde{\delta}_{CWI}\right) = c \prod_{i=1}^{n} P(\boldsymbol{e}_i) \times \prod_{k=1}^{m} \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} P_k\left(\tilde{\delta}_{CWI,ij} \middle| \boldsymbol{e}_i, \boldsymbol{e}_j\right), \qquad (8)$$

where m is the number of channels used and k is the index over m channels.

The maximum of the joint posterior pdf (equation 8) occurs at the most probable combination of the events locations. Hence, the event locations can be found by solving an optimization problem. Taking the negative logarithm of equation 8, the multiplications are then converted to summations that are more numerically stable:

$$-\ln\left[P\left(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n \middle| \tilde{\delta}_{CWI}\right)\right] = -\ln[c] - \sum_{i=1}^{n} \ln[P(\boldsymbol{e}_i)] - \sum_{k=1}^{m} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \ln\left[P_k\left(\tilde{\delta}_{CWI,ij} \middle| \boldsymbol{e}_i, \boldsymbol{e}_j\right)\right]. \quad (9)$$

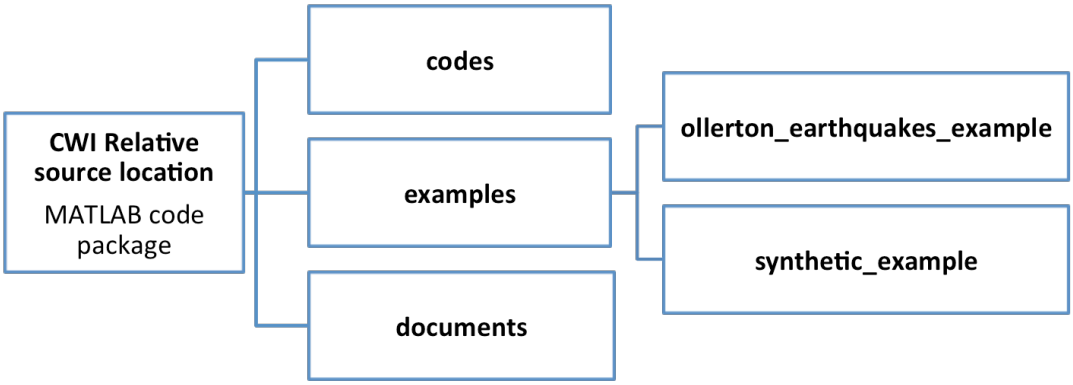
Optimization of equation 8 is equivalent to minimizing equation 9, where $\ln[c]$ and $\ln[P(\boldsymbol{e}_i)]$ (for uniform priors) can be ignored as they are constant. Thus, the event locations $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$ can be solved by minimizing the objective function

$$L(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n) = -\sum_{k=1}^{m} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \ln\left[P_k\left(\tilde{\delta}_{CWI,ij} \middle| \boldsymbol{e}_i, \boldsymbol{e}_j\right)\right]. \qquad (10)$$

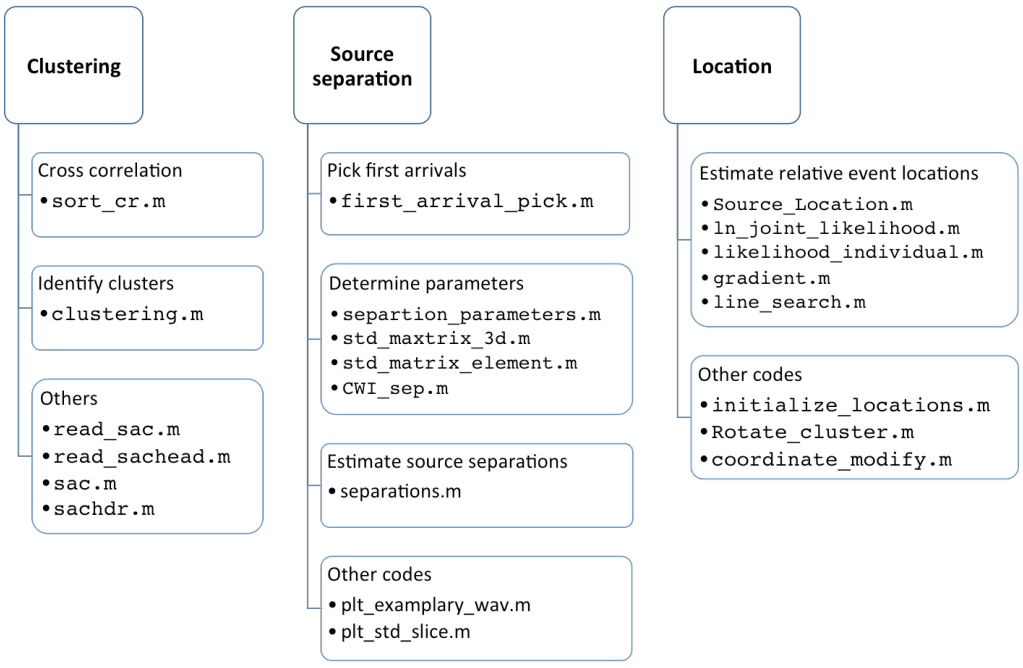
The objective function L is minimized using a conjugate gradient method, Polak-Ribiere technique (Navon and Legler, 1987; Press et al. 1987).

2. Package contents

The package include the following files and directories:



Directory `codes` contains the following scripts and functions that conduct clustering, estimating inter-source separations, and estimating relative event locations. The entire process can be implemented by running a single script `Main_running_script.m` in the directory step-by-step. The files generated will be stored in directory `codes` as the working directory. However, files generated when the authors implemented the example are stored in directories `examples/synthetic_example` and `examples/ollerton_earthquakes_example/` for the real-data example and synthetic data, respectively.



3. Codes

This section introduces the usage of the codes listed in section 2 in three subsections, Clustering, Estimating inter-source separations, and Location. A working example using microearthquake data from New Ollerton, Nottinghamshire, England is presented to demonstrate how to use the code package to process seismic data to solve for source locations.

3.1 Clustering

The theory of CWI requires the events to have identical source mechanisms, so the events need to be classified into clusters of similar source mechanisms. The similarity in sources is assessed by the similarity in waveforms recorded by the same seismic station channel, which is measured by their correlation coefficients.

The package classifies events in two steps, computing crosscorrelations and identifying clusters, with scripts `sort_cr.m` and `clustering.m`, respectively. `sort_cr.m` reads in seismic data in SAC form. It first selects events and station channels for location according to criteria set by user: `MIN_channel` is the minimum number of recording channels for an event to be considered, and `MIN_event_per_channel` is the minimum number of events that a channel records for the channel to be used. It then calculates the correlation coefficient cr of all available pairs consisting of the selected events recorded by each selected station channel, and finds the maximum of each correlation coefficient cr_{max}. For each event pair, cr_{max} are then averaged

over all selected station channels that have recorded both events, and these average maximum correlation coefficient $cravg_{max}$ are then sorted in descending order.

`clustering.m` implements the method of Ottemoller et al. (2017) to identify clusters. It starts with the event pair with the highest $cravg_{max}$, making them the first two events of the first cluster. It then searches through the sorted list of $cravg_{max}$, and adds events that are linked to the current cluster. A link to a cluster is defined as one of the events in the pair is correlated with any event that is already in the cluster (event pairs are defined to be correlated if $cravg_{max}$ is higher than `MIN_corr`, the threshold of $cravg_{max}$ for an event pair to be considered part of a cluster). The search restarts from the first unclassified pair (unclassified pair with the highest $cravg_{max}$) every time a new event is added to the cluster to avoid overlooking any linked event. The search loops until there is no event that can be added to the current cluster, after which it starts a new cluster by adding the two events of the first unclassified pair. The search ends when all events are classified. Any clusters with fewer then `MIN_E_per_CLUSTER` events will be removed from the list of clusters.

3.1.1 Code description

This subsection contains a short description of each script or function for event clustering. A thorough explanation of inputs and outputs is included in Appendix 1.

Crosscorrelation

- `sort_cr.m`:

`sort_cr.m` selects events and station channels using the SAC files according to the given criteria `MIN_channel` and `MIN_event_per_channel`. For each station channel, the correlation coefficient of all available event pairs are computed and the maximum correlation coefficient of each pair are found. The maximum correlation coefficient is then averaged over all channels that have recorded the given event pair. The average maximum correlation coefficients are then sorted in a descending order and sorted in `max_cr.xls` or `max_cr.csv` depending on the availability of Excel.

Identify clusters

- `clustering.m`:

`clustering.m` classifies the selected events into clusters according to waveform similarity, which is reflected by maximum correlation coefficient.

3.1.2 Example

In order to demonstrate how to use the codes in the package, we use the New Ollerton microearthquake data recorded by a temporary recording network deployed by British Geological Survey (BGS) in early 2014. The events have magnitudes of 0.7-1.8ML, and the waveforms are filtered to 2-20Hz. In order to show the capability of the codes, we used 118 SAC files, with 41 events recorded by 5 different channels of seismic stations. Specifically, 8 events were recorded by station channel NOLA_E, 35 by NOLC_Z, 34 by NOLF_E, 34 by NOLF_N, and 7 by NOLF_Z.

Crosscorrelation

The SAC files are stored in directory `example_data`. To acquire the sorted crosscorrelation file `max_cr.xls` (or `.csv`) using `sort_cr.m`, the following lines are executed (Session 1.1 in `Main_running_script.m`):

```
data_location='../examples/ollerton_earthquakes_example/example_data/';
MIN_channel=2;
MIN_event_per_channel=10;
xcorr_range=[10 30];
max_time_lag=10;
sort_cr
```

For these waveforms, we set the selection criteria that only events that have been recorded by at least 2 station channels should be considered for clustering and hence for location in later steps; and that only station channels that have recorded at least 10 events should be used. These criteria are to ensure the robustness of data chosen for source location, presuming fewer data of higher quality gives more reliable results than more data of inconsistent quality. We only compute the crosscorrelation of the part between 10-30 seconds of waveforms to save computation time, as most of the energy of a waveform is contained in the early arrivals which contribute most to the crosscorrelation. As all of the events are located very close to each other compared to event-station distances, the first arrival time in each waveform should not differ very much, and a search range can be set to shorten the time used searching for the maximum crosscorrelation: it is set to 10 seconds this case. The results are shown in the MATLAB workspace:

```
There are 41 events and 5 channels found.
There are 34 events and 3 channels selected.
Files generated:
events_channels_selected.txt
file_indices.csv
max_cr.csv
```

The code identified 41 events and 5 station channels from the SAC files, among which 34 events and 3 station channels were selected (not rejected), based on the criteria above. The files generated by executing `sort_cr.m` are also listed. The selected events and station channels are stored in file `events_channels_selected.txt`, as shown below:

```
1    NOLCEHZ
2    NOLFHHE
3    NOLFHHN

01    2014-02-07-04-06-05
02    2014-02-07-15-13-54
03    2014-02-09-05-33-27
04    2014-02-09-13-56-42
        …
```

The first three lines are the names of the selected station channels, and the following lines show the selected events, represented by their occurrence time. For example,

`"01 2014-02-07-04-06-05"` means the first selected event occurred at year 2014, month February, day 07, hour 04, minute 06, and second 05.

`file_indices.csv` keeps a record of the indices of the SAC files of the waveforms from the selected events recorded by the selected station channels. For instance the panel below shows the first few lines of `file_indices.csv`, where in the first line `"2 3 4"` refers to the indices of SAC files of waveforms from the first selected event `2014-02-07-04-06-05` recorded by the selected channels NOLCEHZ, NOLFHHE, and NOLFHHN, respectively, which is used by `clustering.m` in later steps:

```
2 3 4
6 7 8
11 12 13
15 16 17
   …
```

The third file generated by `sort_cr.m` is `max_cr.csv`, which stores the maximum crosscorrelations of all available pairs consisting of the selected events in descending order. The panel below shows the first few lines:

```
33 34 0.99682 0.99331 0.99816 0.999
28 29 0.99527 0.98852 0.99812 0.99918
27 28 0.99525 0.99552 0.99397 0.99627
12 13 0.9921  0.98366 0.99608 0.99657
```

…

The first line "`33 34 0.99682 0.99331 0.99816 0.999`" shows the highest average maximum crosscorrelations $cravg_{max}$ among the available channels. "`33 34`" are the indices of the selected events with highest $cravg_{max}$, "`0.99682`" is their $cravg_{max}$, and each of the following figures is the maximum crosscorrelation cr_{max} of the waveforms of these two events recorded by one selected channel.

Identify clusters

With the sorted crosscorrelations, the events are classified into clusters with similar source mechanisms by running the following lines (Session 1.2 in `Main_running_script.m`):

```
data_location='../examples/ollerton_earthquakes_example/example_data/';
max_cr_pair=load('Max_cr.csv');
load('file_indices.csv');
MIN_corr=0.9;
MIN_E_per_CLUSTER=5;
clustering
```

We choose to only consider event pairs with a high similarity in source mechanisms, hence set the threshold `MIN_corr` at 0.9, so that the event pairs in `max_cr.csv` whose $cravg_{max}$ is below 0.9 will not be searched by `clustering.m`. In the location algorithm, up to $n(n-1)$ separation data (a mean and a standard deviation for each event pair out of up to $n(n-1)/2$ pairs for n events) are used to invert for $3n$ variables (x, y and z coordinate for each event

location) for the locations of n events. Therefore, the more events are contained in a cluster, the denser information we have to solve the locations. However,since only 34 events were selected, the lower threshold to form a cluster should be relatively wide, therefore `MIN_E_per_CLUSTER` is set to 5 in this example. The following results are shown in the MATLAB workspace:

```
34 events classified into 2 clusters.
cluster1: 11 events
cluster2: 23 events

Files generated:
files_cluster1_channel1.txt
files_cluster1_channel2.txt
files_cluster1_channel3.txt
files_cluster2_channel1.txt
files_cluster2_channel2.txt
files_cluster2_channel3.txt
```

All selected events are classified into 2 clusters, with 11 and 23 events, respectively, and there are no unclassified events. The generated files store the filenames of the SAC files of waveforms of events in each cluster recorded by each station channels. For instance, files_cluster1_channel1.txt lists the SAC files of waveforms of the 11 events in cluster 1 recorded by channel 1 (NOLCEHZ):

```
2014-04-17-1750-40S.BGS___015_NOLC__EH_Z.SAC
2014-04-18-0453-16S.BGS___015_NOLC__EH_Z.SAC
2014-04-13-2058-38S.BGS___015_NOLC__EH_Z.SAC
2014-04-10-0653-33S.BGS___015_NOLC__EH_Z.SAC
                        …
```

Thus, events in each cluster have very similar source mechanism, with $cravg_{max} \geq 0.9$, and are suitable for source separation estimation using CWI.

3.2 Estimating inter-source separations

For each cluster, we pick the first arrivals of waveforms recorded by the same station channel and align them. We assume that the differences in the coda are only due to the difference in source locations, as we have ensured that the source mechanisms are similar. This is not true if seismic velocity changes between the different event occurring times. Although theoretically, the two types of changes could be discriminated (Snieder et al., 2002), this is beyond our scope. This package allows users to manually pick waveform first arrivals in an interactive graphical manner.

To apply CWI to estimate inter-source separations, waveform coda needs to be divided into non-overlapping time-windows with equal length. Seismic coda starts to become suitable for CWI where the waves are sufficiently scattered so that a time-window contains waves leaving the source from many possible directions; and it ends where the background noise level exceeds the amplitude of the signal from the events. Therefore, there is a limited length of coda that can be used. In order to obtain reliable separation standard deviations, at least four time-windows are

needed for each pair of waveforms. Deciding the lengths of time-windows involves a trade-off: the computation of equation 3 using wave representations 1 and 2 gives rise to double sums, and CWI assumes the cross term is negligible compared to the diagonal terms, whose ratio is inversely proportional to the length of time-windows (Snieder 2006); for it to be reasonable to ignore the cross terms, the time-windows need to have sufficient length. On the other hand, the time-windows cannot be too long to avoid cycle skipping in the crosscorrelation (equation 3). It is therefore not a trivial task to decide the number, length, and start of time-windows applied to implement CWI.

To avoid the vagaries of trial and error, this package applies the separation-uncertainty matrix introduced by Zhao et al. (2017) to systematically find the most suitable combination of [number, length, start-time] of time-windows. Script `separation_parameters.m` allows users to set a time `coda_start` in the waveform, a time shortly after which coda should be taken; and a time `coda_end`, at which coda is no longer suitable for CWI. At each time, the [start-time] of time-windows is given by `coda_start+n*dl_coda_start`, where `dl_coda_start` is the increment of the starting time of windowing, and `n=0,1,2….` The script then takes the part of the aligned waveforms starting from the [start-time] of time-windows, and ending at `coda_end`. For each taken length of waveform coda, the script is able to work out all the possible combinations of [number, length] of time-window allowed, given the minimum number and length of time-window (`min_num_win` and `min_l_win`), the maximum time-window needed (`max_num_win_limit`) if the total length of coda taken is sufficiently long, and the increment of window length (`dl_win`). It then computes the uncertainty of separation estimates using each possible [number, length] parameter combination for the current selected coda.

For data from each station channel, the script computes a separation-uncertainty matrix, with elements $\Omega_{i,j} = \sum_N \sigma_{i,j}/N$, where $\sigma_{i,j} = \sqrt{\sum_l (\delta_{i,j,k} - \bar{\delta}_{i,j})^2/l}$ is the standard deviation of separation estimates from l coda time windows for events i and j, $\delta_{i,j,k}$ is the separation estimate of the kth window and $\bar{\delta}_{i,j}$ is their mean, and N is the number of event pairs on that channel. The value of $\Omega_{i,j}$ reflects the uncertainty of separation estimates derived from one recording channel for each source pair. Averaging over multiple source pairs gives a final uncertainty estimate for that combination of parameters. Thus, the script searches for the combination of time-window [number, length, start-time] parameters that gives the lowest uncertainties for separation estimates.

With the number, length and start-time of time-windows systematically determined by `separation_parameters.m`, or set by the users for their specific needs, inter-source separations between all possible event pairs in a cluster recorded by the same station channel can be estimated with script `separations.m`. The script generates files storing the mean and standard deviations of separation estimates that are used in the location algorithm. For each cluster and channel combination, a file `seps_clusteri_channelj.xls (or .csv)` is generated, which contains two columns of N elements, where N is the number of event pairs. The first and second column store the means and standard deviations of the separation estimates from all time-windows used. In each column, the event pairs are ordered as $[e_1\ e_2]$, $[e_1\ e_3]$,…, $[e_1\ e_n]$, $[e_2\ e_3]$, $[e_2\ e_4]$,…, $[e_2\ e_n]$, $[e_3\ e_4]$, $[e_3\ e_5]$,… $[e_{n-1}\ e_n]$, where n is the number of events. If practitioners use their own separation data to estimate relative event locations, the separation data files should be organized in the same way.

The remainder of this subsection introduces the scripts and functions used for first arrival picking, determining separation parameters, and estimating source separations. It is then shown how they are used with the example of the New Ollerton microearthquake data.

3.2.1 Code description

This subsection contains a short description for each script or function for estimating inter-source separations. Thorough explanation of inputs and outputs are included in Appendix 2.

Pick first arrival of waveforms

- `first_arrival_pick.m`

`first_arrival_pick.m` allows users to interactively pick the first arrival of the waveforms listed in a .txt file in three simple steps:

1) On an exemple waveform of events in the cluster recorded by the same station channel (listed in the given .txt file), select a rough (initial) range containing the early arrivals, including a small interval before the first arrival.

2) The script then allows the user to pick the first arrival of each waveform within the initial searching range selected in step 1. For each waveform, two plots are generated. The first plot shows the part of waveform inside the initial range, and allows the user to select a narrower range that contains the first arrival and only a few subsequent arrivals of the waveform.

3) For each event, the second plot shows the part of waveform inside the narrow range selected in step 2, which should be narrow enough for the user to location the center of the first arrival with a high accuracy. The user then picks the first arrival at its center. If in step 2, the part of the

waveform selected is found to be inconvenient for selecting the narrow range and picking the first arrival, end the process by closing any plotted figure and rerun the script.

Determine parameters for estimating source separations

- `separation_parameters.m`

`separtion_parameters.m` finds the combination of [number, length, start-time] of time-windows for running `CWI_sep.m`. A 3D std-matrix is generated. Each element is the average standard deviation of inter-source separations over all available pairs in the given cluster recorded by the given station channel(s).

- `CWI_sep.m`

`CWI_sep.m` is one of the core functions in this package, called by multiple scripts and functions. It applies coda wave interferometry (CWI) (Snieder, 2006) to estimate the separation between one pair of sources with similar mechanisms for isotropic sources in a 2D or 3D acoustic medium, or double-couple sources in an elastic medium, assuming both sources are on the same fault plane (Snieder and Vijlandt, 2005). The two improvements introduced by Robinson et al. (2011) are applied: 1) they remove the Taylor series approximation of the autocorrelation of waveforms of the two events, and 2) appliey a restricted searching range when searching for the maximum of the correlation coefficient `Rmax` to avoid cycle skipping.

- `std_maxtrix_3d.m`

`std_maxtrix_3d.m` is called by `separation_parameters.m` generates a three-dimensional matrix. Each element is the average standard deviation of inter-source separations over all available pairs in the given cluster recorded by the given station channel(s).

- `std_matrix_element.m`

`std_matrix_element.m` is called by `std_maxtrix_3d.m`. It calculates elements of `STD_matrix_3D`, with given [number, length, start-time] of time-windows.

Estimate inter-source separations

- `Separations.m`

`separations.m` estimates the inter-source separations of all available event pairs in the given cluster recorded by the given station channel. It generates `.xls` (or `.csv`) files containing the means and standard deviations of all event pairs from the cluster. For event pairs whose waveforms do not all exist, the results are marked with `-1`.

3.2.2 Example

The 34 selected events have been classified into 2 clusters, with 11 in cluster 1 and 23 in cluster 2. Waveforms recorded by the selected station channels NOLCZ, NOLFE and NOLFN (short for NOLCEHZ, NOLFHHE, and NOLFHHN) will be used to estimate inter-source separations. For each cluster recorded by each channel, the routine consisting of first arrival picking, parameter determining, and separation estimation are conducted.

Pick first arrival of waveforms

To use `first_arrival_pick.m` to pick the first arrival of waveforms in cluster 1 recorded by channel 1 (NOLC), the following lines are executed (Session 2.1 in `Main_running_script.m`):

```
data_location='../examples/ollerton_earthquakes_example/example_data/';
txtfilename='files_cluster1_channel1.txt';
first_arrival_pick
```

A plot (Figure 2) of an exemple waveform is generated for the user to select an initial range containing the early part of the waveform. Click twice to select the two boundaries of the initial range, each indicated by the center of the cross as shown in Figure 2. Only the horizontal coordinate of the clicked positions will be recorded, as the boundaries are defined on the time axis. This range should contain the first and the early arrivals for all waveforms in the cluster recorded by the same channel.

Two plots are then generated for each waveform listed in the .txt file used above. The first plot shows the part of the waveform inside the initial range for event 1 of cluster 1 recorded by channel 1, as shown in Figure 3a. Click twice on the plot to select a narrower range that contains the first arrival and only a few subsequent arrivals (3 to 5 is suggested). The second plot is then generated, containing the part of the waveform in the narrow range, as shown in Figure 3b. Pick the first arrival of event 1 by clicking at its center. Plots for subsequent events in the cluster will then be generated for the user to pick their first arrivals. The index of the sample at which the first arrival is picked for each waveform is stored in

`first_arrivals_clusteri_channelj.xls` (or `.csv`) for cluster i recorded by channel j, which can be used to align the waveforms by their first arrivals at the same position. Figure 4a and 4b show exemples of aligned waveforms of cluster 1 and cluster 2 recorded by channel 2 (NOLFE).

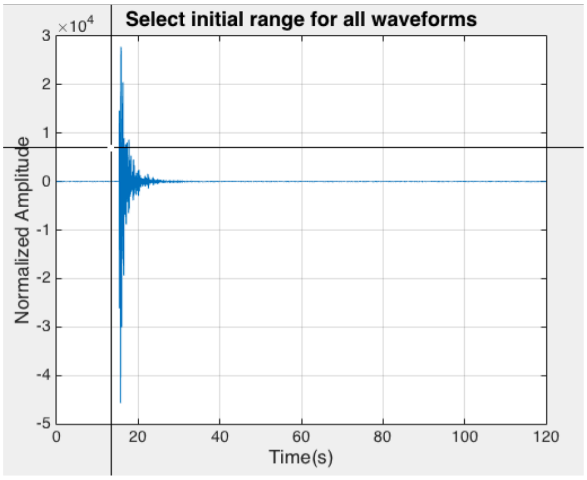


Figure 2: An exemple waveform in the first cluster displayed for the user to select the initial range that contains the first arrival.

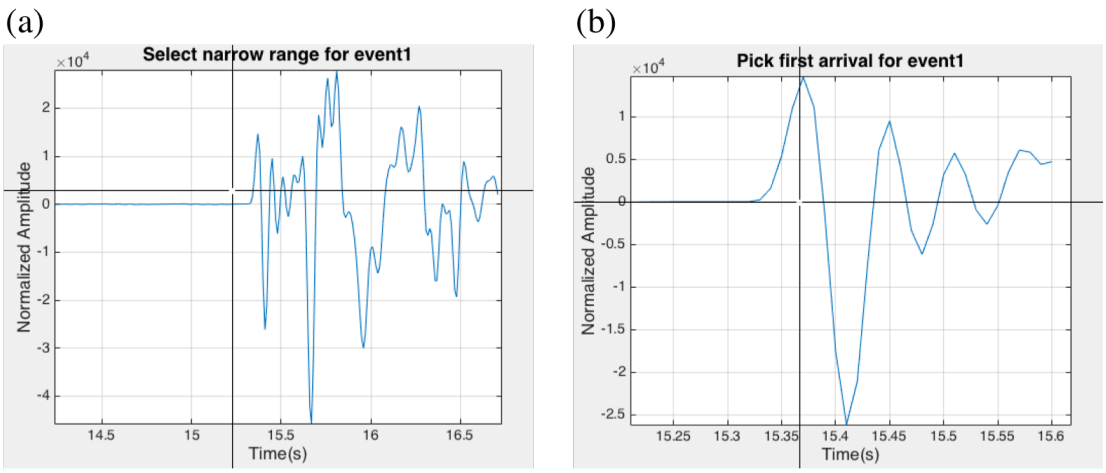


Figure 3: Two plots created for each waveform in the cluster. Panel (a) is the part of waveform in the selected initial range for the user to pick a narrower range that contains the first arrival. Panel (b) is the part of waveform in the selected narrow range, where the user picks the center of the first arrival of the waveform.

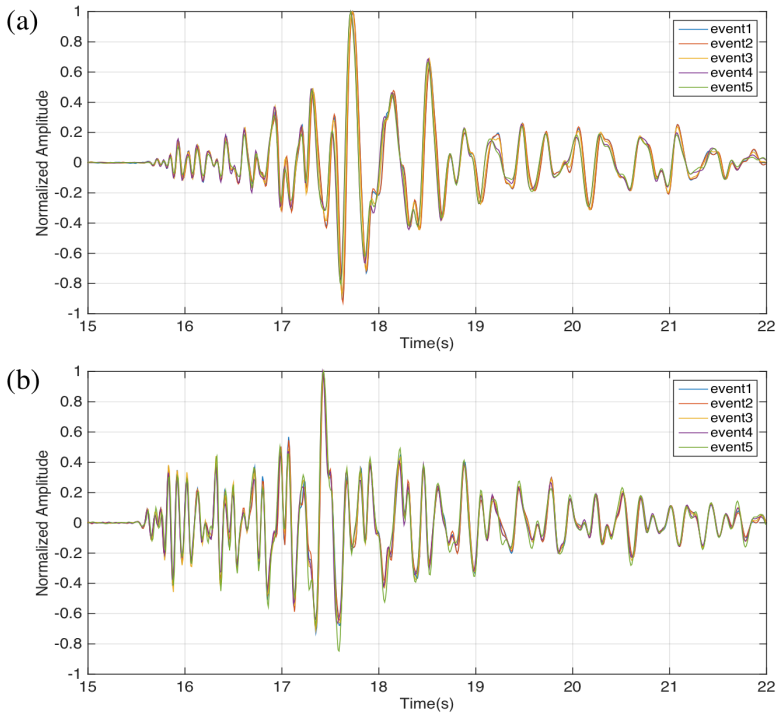


Figure 4: Exemplary waveforms with first arrivals aligned together. Panel (a) and (b) are waveforms from cluster 1 and 2, recorded by channel 2 (NOLFE), respectively.

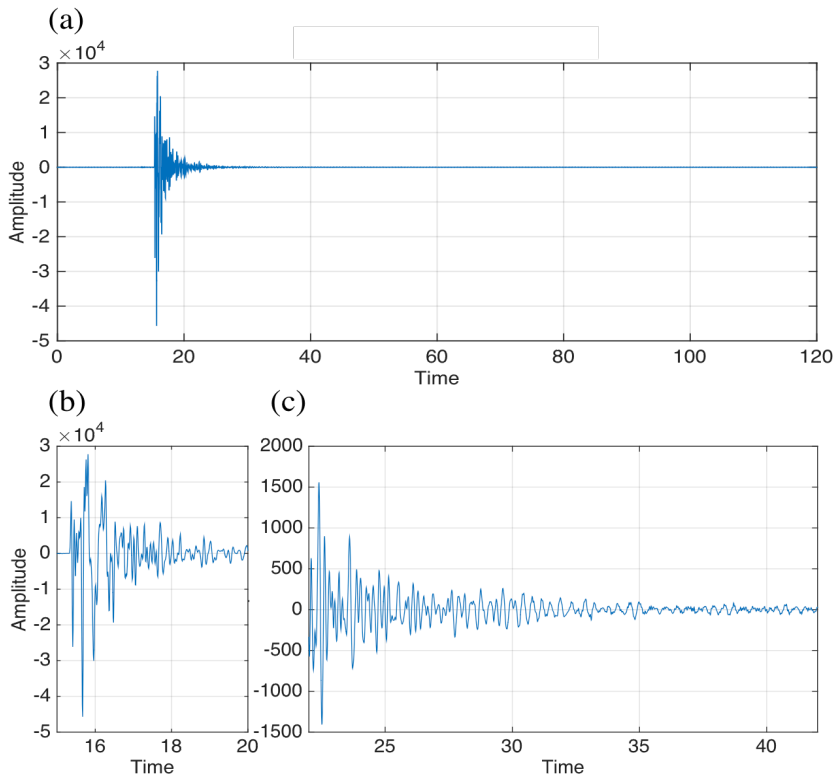


Figure 5: An exemple waveform in the cluster. Panel (a) is the full waveform; panel (b) contains the early arrivals and a small part of coda, within which the user chooses where to start searching for the start of windowing; and panel (c) contains a large part of the coda where the amplitude of coda falls significantly.

Determine parameters for estimating source separations

Before using CWI to estimate inter-source separations, we use the separation-uncertainty matrix to determine the parameters [number, length, start-time] of time-windows. First, execute the following lines (Session 2.2 in `Main_running_script.m`) to acquire the plot of an exemple waveform (Figure 5) to determine the remaining inputs of `separation_parameters.m` (unless determined otherwise):

```
first_arrival_ind=load('first_arrivals_cluster1_channel1.csv')
txtfilename='files_cluster1_channel1.txt';
data_location='../examples/ollerton_earthquakes_example/example_data/';
plt_examplary_wav
```

The exemple waveform is shown as Figure 5a, with its zoomed panels that contain the interval where the potential start for windowing lies (Figure 5b) and the interval where the amplitude of coda falls significantly, indicating the end of the search (Figure 5c). We decide to start the search from 16s (`coda_start`) when the multiply scattered waves started to be recorded, and to end the search at 40s (`coda_end`) when the amplitude of coda has fallen to noise level. We set the increment of the [start-time] of windowing to be 1s (`dl_coda_start`), the minimum length of windows `min_l_win` to 2.5s, which can increase at an increment of 0.5s (`dl_win`). We also set the minimum number of windows `min_num_win` to 4 to ensure some robustness of uncertainty estimation, and the maximum number of windows `max_num_win_limit` to 7. `max_num_win_limit` is not necessarily the maximum number of time-windows being searched, but an upper limit of the number of windows examined if the available coda is so long that the maximum number of windows allowed for some given [start-time, length] is more than that the user desires. If there isn't a limit on the number of time-windows, `max_num_win_limit` can be set to a number that is guaranteed to be larger than the maximum number of windows allowed by the length of available coda. We want the case represented by coda [start-time] to be displayed, and the running result to be saved, so `plt_flag` and `save_flag` are both set to '`y`'.

`separation_parameters.m` calls for function `CWI_seps.m`. Some of its inputs are generated inside `separation_parameters.m`, and the remainder need to be provided. The following lines are executed (Session 2.2 in `Main_running_script.m`):

```
coda_start=16;
coda_end=40;
dl_coda_start=1;
min_num_win=4;
min_l_win=2.5;
max_num_win_limit=7;
dl_win=0.5;
plt_flag='y';
save_flag='y';
% Inputs for running CWI_sep.m
veloc=[4200 2360];
source_type='doublecouple';
search_range=40;
num_interpo_point=10;
separtion_parameters
```

The MATLAB workspace shows the combination of parameters found that gives the lowest separation estimation uncertainty:

```
The smallest average standard deviation 11.08m is obtained when using:
4 2.500s time-windows, starting from 19s.
```

It is found that using 4 time-windows with a length of 2.5s starting from 19s of the waveforms to conduct source separation estimation gives the lowest average standard deviation of 11.08m, indicated by the darkest blue grid at the bottom left of Figure 6a. The separation-uncertainty matrix provides a guideline to enable us to determine the combination of parameters for CWI. However, when using other combinations of parameters, the results should not change significantly if we do not move far from those found with the matrix. For instance, using 4 time-windows with a length of 3s starting from 19s leads to an average standard deviation of 15.64 (Figure 6a), and using 5 time-windows with a length of 2.5s starting from 21s leads to an average standard deviation of 17.22 (Figure 6b). We can thus determine the [number, length, start-time] of time-windows for each cluster recorded by each channel and estimate the source separations.

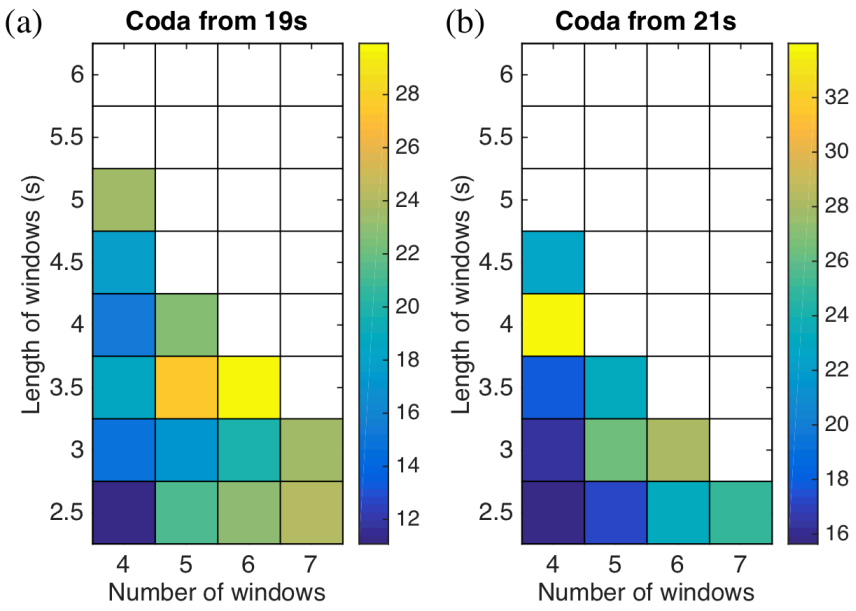


Figure 6: Slices of the 3D separation uncertainty matrix. Colors indicate the values of the average standard deviations resulting from the corresponding parameter combination of number of windows, window length and start-time of windows, used to divide the coda. White cells indicate parameter combinations that are not supported by the available data. Panel (a) is the slice of the matrix where all parameter combinations have a start time of windowing at 19s. Panel (b) is the slice where all combinations have a start time of windowing at 21s.

Estimate inter-source separations

With the combination of parameters found with the separation-uncertainty matrix, we estimate the inter-source separations for each cluster recorded by each channel separately. For cluster 1 channel 1, the following lines are executed (Session 2.3 in `Main_running_script.m`):

```
first_arrival_ind=load('first_arrivals_cluster1_channel1.csv')
txtfilename='files_cluster1_channel1.txt';
```

```
data_location='  ../examples/ollerton_earthquakes_example/example_data/
';
veloc=[4200 2360];
source_type='doublecouple';
search_range=40;
num_interpo_point=10;
plt_aligned='y';
win_length=2.5;
win_start=21;
win_num=5;
separations
```

There are 55 event pairs made up of 11 events in cluster 1. For each event pair, there are 4 estimates of source separations given by the 4 time-windows. The generated file `seps_cluster1_channel1.csv` stores the mean and standard deviation of the 4 separation estimates for each event pair, which are then used for source location.

3.3 Location

The relative source locations of events in a cluster are solved by minimizing the objective function L in equation 10. The most likely set of event locations are obtained where the function L attains its minimum. The main function in the package for source location is `Source_Location.m`, which estimates the relative source locations of a cluster of events with the mean and standard deviations of the inter-source separations estimated with CWI. To start the minimization, a set of initial locations of the events are needed; these can be generated

with function `initialize_locations.m` given in the package, or provided by the user. Function `Source_Location.m` first evaluates the objective function *L* (the negative logarithm of the joint likelihood function) at the given initial event locations (by calling `ln_joint_likelihood.m`) and computed the gradient (by calling `gradient.m`). The function searches along the star searching direction `-Gradient0` for the minimum in the given direction (by calling `line_search.m`). The event locations are thus updated to the minimum found, and the value of *L* is reduced. The function then calculates the next searching direction that is orthogonal to the gradient at the current position and conjugate to the last searching direction, conducts a line search along the new searching direction, and updates the event locations to the new minimum found. Such iterations continues until one of the three conditions is met: 1) the value for any non-zero step length is larger then that obtained in the later iteration; 2) the reduction in the value of *L* in an iteration is trivial (smaller than `L_tolerance`); or 3) the maximum number of allowed iterations `max_iteration` is reached. Starting with a different set of initial location is recommended if the iterations are terminated due to the third condition, as some initializations can lead to convergence quicker than others.

Inter-source separations estimated with CWI are required to estimate the relative locations using function `Source_Location.m`. If users are using their own separation data, the separation files for each cluster and channel combination should be generated in the format described in the introduction of section 2.2.

3.3.1 Code Description

This subsection contains a short description for each script or function for source location. Thorough explanations of inputs and outputs are included in Appendix 3.

- `Source_Location.m`

`Source_Location.m` estimates the relative location of a cluster of events, using inter-source separation data (their mean and standard deviations) estimated with CWI. The location is solved as a minimization problem, where the most probable set of event locations are found where the objective function *L* attains its minimum.

Parameters of the empirical relations (equation 5) between the true separation and the mean and standard deviation of the CWI estimates are given inside function `Source_Location.m`. Here we use the ones derived by Robinson et al. (2011), however, users can change them to those derived from their own experiments.

- `ln_joint_likelihood.m`

`ln_joint_likelihood.m` estimates the negative logarithm of the joint likelihood function *L* for a cluster of events. For each station channel, all available event pairs are considered, except for those instructed to be discarded in the input of the function.

- `likelihood_individual.m`

`likelihood_individual.m` estimates the likelihood function for an individual pair of events.

- `gradient.m`

`gradient.m` computes the gradient of *L* at a given set of event locations

- `line_search.m`

`line_search.m` conducts searches for the minimum of an objective function *L* over a given direction in three steps:

1) along the given direction, evaluate *L* in a large range with increments of `delta_lambda_coarse`, and find the (coarse) minimum among these values of *L*.

2) define a smaller range around the minimum found, evaluate *L* in the smaller range with increments of `delta_lamba_dense`, and find a new (dense) minimum among these values of *L*.

3) adjust the dense minimum by fitting a parabola using the minimum point found and two of its adjacent points, and take the vertex of the parabola as the final minimum of *L* along the given direction.

- `initialize_locations.m`

`initialize_locations.m` generates a set of initial locations for the minimization problem solved by `Source_Location.m`. It first creates a set of locations for the given event number, uniformly distributed in the given range. It then reorders the created events to ensure the least squares residuals of inter-source separations (residuals being the difference in separation of each event pair between that calculated from the created initial locations and that estimated with CWI from seismic data).

3.3.2 Synthetic example

Here we use a synthetic example to show that optimization conducted by function `Source_Location.m` is able to solve the relative location of a cluster of events using CWI separation estimates. We randomly distribute 50 sources in a homogeneous cube with side length of 300m, shown as the hollow circles in Figure 7. Dominant wavelength is $\lambda_d = 534m$, and the maximum source separation is λ_{max} is $424m$, i.e. $0.8\lambda_d$. The purpose of this synthetic experiment is to demonstrate the location of a cluster of events with CWI separation data, we therefore start by generating CWI separation with the features of CWI estimates given by equation 5. With the locations of the 50 events, we calculate their true separations δ_t, and create the CWI separation data (separation mean and standard deviation) using the empirical relations $\mu(\widetilde{\delta_t}) = a_1 \frac{a_2\widetilde{\delta_t}^{a_4} + a_3\widetilde{\delta_t}^{a_5}}{a_2\widetilde{\delta_t}^{a_4} + a_3\widetilde{\delta_t}^{a_5} + 1}$, and $\sigma(\widetilde{\delta_t}) = b_1 \frac{b_2\widetilde{\delta_t}^{b_4} + b_3\widetilde{\delta_t}^{b_5}}{b_2\widetilde{\delta_t}^{b_4} + b_3\widetilde{\delta_t}^{b_5} + 1} + c$ (equation 5). The separation data are thus consistent with the known biases and level of uncertainty from the CWI technique.

To implement the location process, initial locations of the 50 events are randomly distributed within a cube of the same size of that of the true locations of the events using `initialize_locations.m`, shown as red triangles in Figure 7a. The spread (the volume in the medium over which the sources extend) of the initial locations are not necessarily similar to that of the true locations, however in most cases doing so can lead to faster convergence to the global minimum of the objective function, and significantly reduce the chance of converging to a local minimum. Hence, when applying the location codes to real data, users are recommended to make an estimation of the spread of the event cluster for initializing the optimization.

The optimization with the given true and initial locations in this example took 27 iterations to converge, with max_iteration set to 300, L_tolerance to 10^{-5}, and no discarded pairs. The estimated locations are shown as solid circles in Figure 7b, where the improvement in event locations from optimization is readily observed by comparing with their initial locations (red triangles in Figure 7a). The optimization does not lead the estimated locations to exactly the true event locations, due to the uncertainty $\sigma(\widetilde{\delta_t})$ added. The average location error is 27m, amounting to 6.4% of δ_{max}, or 5% of λ_d. Figure 8 compares source separations calculated from the relocated event locations (red) to the separation data (blue) used as input to the optimization. Figure 8a shows that although the input separation data deviate significantly from their actual value (dashed line $y = x$) where the true separation is larger than $0.55\lambda_d$, the recovered source separations are only slightly underestimated. It is therefore proved that the location algorithm is able to correct the biases of CWI estimates to a large extent.

As optimization techniques that guarantee converge to the global minimum of a complicated non-linear objective function do not currently exist, it is suggested to implement the optimization multiple times with different random initializations. In order to examine the location results, we illustrate the change of objective function value in the optimization process for all implementations, as shown in Figure 9. The two zoomed panels show details of the start and end of the processes. The 6 implementations start with different objective function values because of different initial locations. The value decreases rapidly for the first 10 iterations following different paths, and then slows down towards the minima. All optimizations converged to the same minimum of 6738, except for the 4th, which was stuck at a local minimum at 6753. For this synthetic example, the true (global) minimum of the objective function is known as the true

event locations are known, hence we can conclude that the minimum 6738 found is correct within a small numerical error. However, when applying this technique to real events where the correct minimum is unknown, using this method can add confidence to the result at which most implementations converge. This will be shown in the next part of this subsection where the location codes are applied to New Ollerton data.

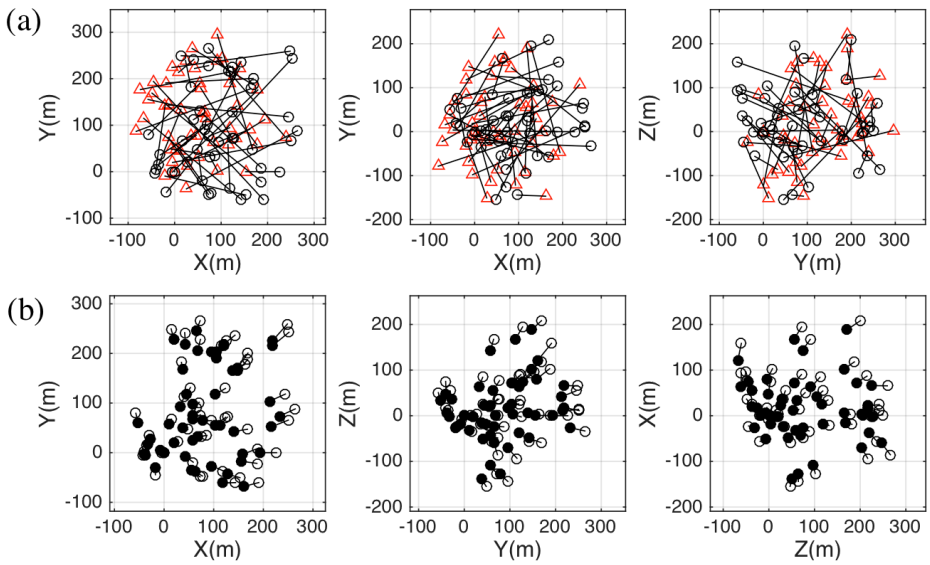


Figure 7: Planar projections of event locations, where axes X, Y and Z point in three orthogonal directions. Panels (a) compare the events' actual locations (circles) and their initial locations (triangles) before optimization, and the black bars show their differences. Panels (b) shows the events' actual locations (hollow circles) and the location results obtained (solid circles), with lines between the hollow and solid circles representing post-optimization location errors.

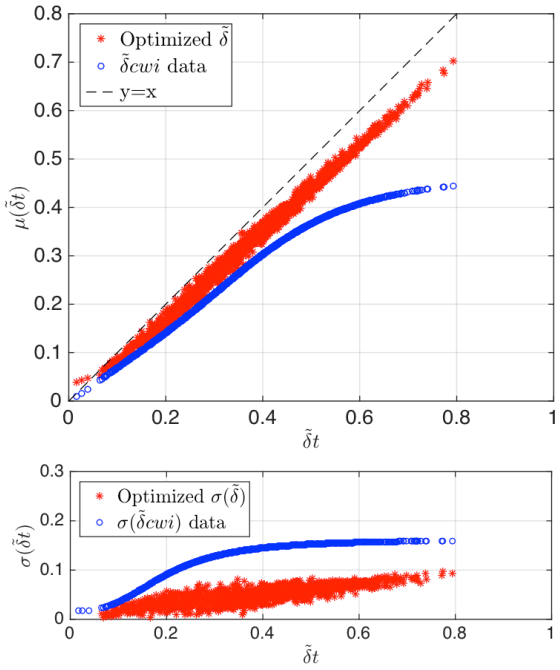


Figure 8: A comparison of input inter-source separations (blue circles) and separations calculated from the location result after optimization (red asterisks). The upper and lower panels show the separation data and their standard deviations, respectively.

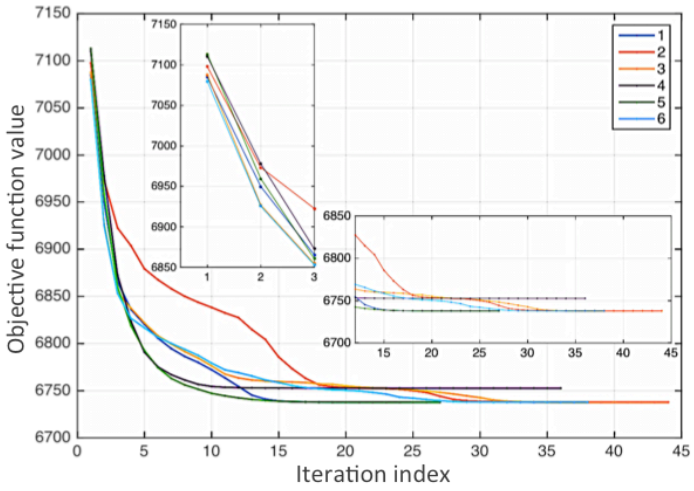


Figure 9: An illustration of the minimization process of each of 6 optimizations from different initializations of random event locations using different colors to indicate each example optimization. The magnified panels show details of how the values of the objective function change with iteration number at the beginning and end of each optimization.

3.3.3 Example with New Ollerton earthquake

With the CWI separation data obtained in 3.2.2, we conduct relative source location for each cluster using data from single station channels (NOLCZ, NOLFE, and NOLFN) and data from all three channels. The followings show the example of estimating source locations of cluster 1 with data from all three channels (Session 3 in `Main_running_script.m`):

```
% Basic parameters
veloc=[4200 2360];
domin_freq=[2.6 2.7 3.2];
domin_wavelength=veloc(2)./domin_freq;
nevent=11;
num_channel=3;
num_pair=(1+(nevent-1))*(nevent-1)/2;
max_iteration=240;
L_tolerance=1e-5;
% Separation data
seps1=load('seps_cluster1_channel1.csv');
seps2=load('seps_cluster1_channel2.csv');
seps3=load('seps_cluster1_channel3.csv');
MU_n=[seps1(:,1) seps2(:,1) seps3(:,1)];
SIGMA_n=[seps1(:,2) seps2(:,2) seps3(:,2)];
select_e_pair
% Initialization
initial_loc=load('../examples/ollerton_earthquakes_example/generated_f
iles/initial_location_cluster1.csv');
```

```
% Location
[ optm_loc, Func_L_add] = Source_Location( initial_loc,MU_n,SIGMA_n,...
   domin_wavelength,discarded_pairs,max_iteration,L_tolerance);
% Save location results
xlswrite('location_cluster1_3channels',optm_loc);
```

`discarded_pairs` are selected with script `select_e_pair.m` with criteria that 1) the mean of the source separation of all event pairs used as input is lower than $0.5\delta_d$ because of the known bias of the CWI technique, as suggested in Figure 1(a); and that 2) the standard deviations of all pairs are lower than $0.17\delta_d$, in order to avoid the impact of event pairs whose data uncertainty is larger than the largest that any CWI estimates should have empirically. Users should apply their own criteria to selected data for input, however they are suggested to include at least these two in order to obtain reliable results. The number of pairs that are discarded can also be used as a rough measurement of data quality. In this example, any standard deviations that are smaller than a certain value, conservative_std, are replaced with conservative_std to avoid including data with falsely estimated very small uncertainties.

The initial locations of 11 events were randomly distributed in a cube with a side-length of 80m using function `initialize_locations.m` as:

```
[initial_loc] = initialize_locations([0 80],[0 80],[0 80], ...
   nevent, MU_n);
```

The initial locations created were stored in file `initial_location_cluster1_1.csv`.

The optimization converged to `optm_loc` with 56 iterations, where the value of function L is reduced to 794.4637. We then repeated the location 9 times with different initializations. Figure 10 shows the progress of each implementation, with the horizontal zoomed panel showing details of convergence. All cases converged to the same level, with reasonable numerical errors. This adds to our confidence that the minimum found with different random initializations is the global minimum of function L, and therefore the relative locations of these 11 events are found.

We conduct source location for both clusters with 1) each single channel, and 2) all three channels. The recovered sources separations are consistent among individual channels (red, green, and cyan) with small residuals, and they are very similar to the result obtained using all three channels (blue), as shown in Figure 11a. The location process also corrects for the underestimation bias of the CWI technique, as illustrated in Figure 11b, which compares the recovered separations (red) and the CWI separation data (black) as input of the optimization. Figure 12 shows the location results of cluster 2 using data from single channels and all three channels, projected onto three orthogonal planes. The patterns show a high level of consistency among single channels and multiple channels. Hence, we proved once again that CWI source location technique is able to give reliable relative locations of a cluster of events using only a single recording channel.

It is worth noting that Figure 12 merely shows one way to present these relative locations, and the patterns would be different when projected to other planes. In some cases, the plotted patterns of location results of different initializations or different channels can appear different, but they can be adjusted to look the same with some rotation operations of the cluster. However,

with the criteria discussed above and shown with Figure 10 and 11, it is unnecessary to obtained similar projected patterns to assess the consistency of the location results.

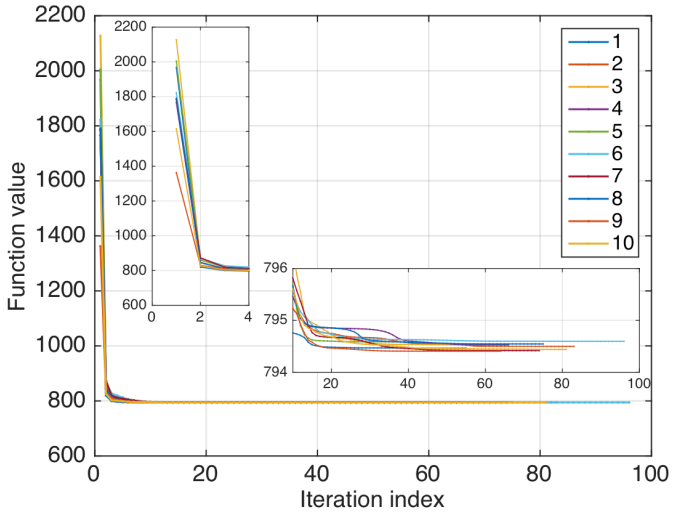


Figure10: An illustration of the minimization process of each optimization from different initializations of random event locations using different colors to indicate each example optimization for cluster 1 using data from all three channels. The magnified panels show details of how the values of the objective function change with iterations at the beginning and end of each optimization.

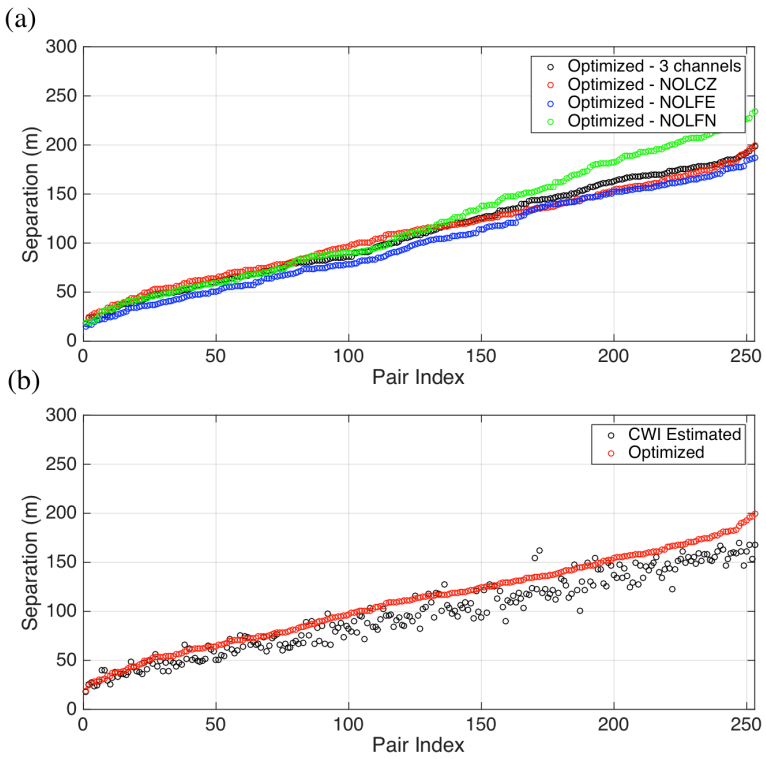


Figure 11: CWI estimates of source separations and their optimized counterparts (the latter calculated from the estimated event locations) of cluster 2. Panel (a) shows the optimized separations using all three channels (black), and using single channels NOLCZ, NOLFE and NOLFN (red, blue, and green). Panel (b) compares the optimized separations of channel NOLCZ (red) with the original CWI separation estimates (black).

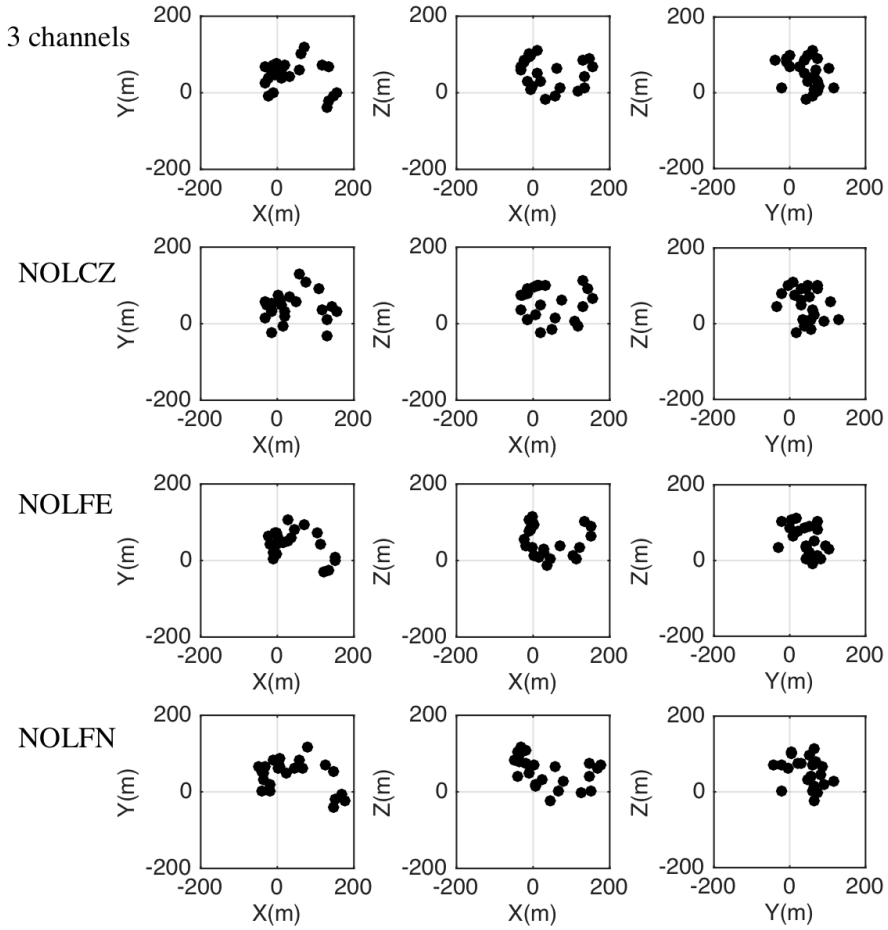


Figure 12: Planar projections of relative event location results of cluster 2, using all three channels (top row), and using single channels NOLCZ, NOLFE, and NOLFN (successive rows). Axes X, Y and Z point in orthogonal directions.

Acknowledgements

We thank Jonathan Singh (University of Edinburgh) for his constructive reviews of this document and code usability. We also thank Brian Baptie (British Geological Survey) for providing the seismic data of the New Ollerton earthquakes used in the real-data example.

References:

Aki, K., 1969, Analysis of seismic coda of local earthquake as scattered waves: Journal of Geophysics Research, **74**, 615-631.

Navon, I. M., D. M. Legler, 1987, Conjugate-Gradient methods for Large-scale minimization in meteorology: Monthly Weather Reviews, **115**, 1479-1502.

Ottemoller, Voss, and Havskov, Seisan: Earthquake Analysis Software for Windows, Solaris and Macosx, 2017, http://seisan.info, accessed 20 December 2017.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1987, Numerical Recipes: The Art of Scientific Computing: Cambridge University Press, U.S.A.

Robinson, D. J., M. Sambridge, and R. Snieder, 2011, A probabilistic approach for estimating the separation between a pair of earthquakes directly from their coda waves: Journal of Geophysical Research, **116**,1-14.

Robinson, D. J., M. Sambridge, R. Snieder, and J. Hauser, 2013, Relocating a Cluster of Earthquakes Using a Single Seismic Station: Bulletin of the Seismological Society of America, **103**, 3057-3072.

Snieder, R., 1999, Imaging and averaging in complex media in diffuse waves in complex media: Springer Netherlands.

Snieder, R., 2006, The theory of coda wave interferometry: Pure and Applied Geophysics, **163**, 455-473.

Snieder R., A. Gret, H. Douma, and J. Scales (2002), Coda wave interferometry for estimating nonlinear behavior in seismic velocity, Science, **295**, 2253-2255.

Waldhauser, F., and W. L. Ellsworth, 2000, A double-difference earthquake location algorithm: Method and application to the northern Hayward fault, California: Bulletin of the Seismology Socioty of America, **90**, 1353-1368.

Zhao, Y., A. Curtis, B. Baptie, 2017, Locating micro-seismic sources with a single seismometer channel using coda wave interferometry: Geophysics, **82**, A19-A24.

Appendix 1 Descriptions of codes and variables for clustering

Crosscorrelation

- `sort_cr.m`:

`sort_cr.m` selects events and station channels using the SAC files according to the given criteria. For each station channel, correlation coefficient of all available event pairs are computed and the maximum correlation coefficient of each pair is found. The maximum correlation coefficient is then averaged over all channels that have recorded the given event pair. The average maximum correlation coefficients are then sorted in a descending order and sorted in `max_cr.xls` or `max_cr.csv` depending on the availability of Excel.

The following inputs are required:

- `data_location`: directory storing the .SAC files
- `MIN_channel`: minimum number of recording channels for an event to be considered
- `MIN_event_per_channel`: minimum number of events a channel records for the channel to be used
- `xcorr_range`: part of waveform (in seconds) to compute crosscorrelation
- `max_time_lag`: expected largest time-lag (in seconds) of waveforms from each other.

The following files are generated:

- `events_channels_selected.txt`: a list of the selected station channels and the selected events
- `max_cr.xls` (or `.csv`): average maximum crosscorrelation coefficients in descending order
- `file_indices.xls` (or `.csv`): the indices of the SAC files of the selected events and station channels.

Whether the generated files are in `.xls` or `.csv` depends on the availability of Excel.

Identify clusters

- `clustering.m`:

`clustering.m` classifies the selected events into clusters according to waveform similarity, which is reflected by maximum crosscorrelation coefficient.

The following inputs are required:

- `data_location`: directory storing the SAC file
- `MIN_corr`: threshold of maximum crosscorrelation coefficient to be considered for clustering
- MIN_E_per_CLUSTER: the minimum number of event for a cluster
- max_cr_pair: the maximum crosscorrelation coefficients used for clustering, can be read from file `max_cr.xls` (or `.csv`)
- `file_indices`: indices of the SAC files of the selected events and station channels, can be read from file `file_indices.xls` (or `.csv`).

The following files are required (generated by `sort_cr.m`):

- `max_cr.xls` (or `.csv`)
- `file_indices.xls` (or `.csv`).

The following files are generated:

- `files_clusteri_channelj.txt`: a list of SAC file names of waveforms of events in cluster i recorded by channel j.

Appendix 2 Descriptions of codes and variables for estimating inter-source separations

Pick first arrival of waveforms

- `first_arrival_pick.m`

`first_arrival_pick.m` follows the clustering method used by Ottemoller et al. (2017). It allows users to interactively pick the first arrival of the waveforms listed in a .txt file in three simple steps:

1) On an exemple waveform of events in the cluster recorded by the same station channel (listed in the given .txt file), select a rough (initial) range containing the early arrivals and with a small period before the first arrival.

2) The script then allows the user to pick first arrival of each waveform within the initial searching range selected in setp 1. For each waveform, two plots are generated. The first plot shows the part of waveform inside the initial range, and allows the user to select a narrower range that contains the first arrive and only a few subsequent arrivals of the waveform.

3) For each event, the second plot shows the part of waveform inside the narrow range selected in step 2, which should be narrow enough for the user to location the center of the first arrival with a high accuracy. Pick the first arrival at its center.

If in step 2, the part of waveform selected is found inconvenient for selecting the narrow range and picking the first arrival, end the process by closing any plotted figure and rerun the script.

The following inputs are required:

- `data_location`: directory storing the SAC files

- `txtfilename`: name of the .txt file that lists SAC filenames of waveforms for first arrival picking

The following files are required (generated by `clustering.m`):

- `files_clusteri_channelj.txt`

The following files are generated:

- `first_arrivals_clusteri_channelj.xls` (or `.csv`): sample indices of first arrival of waveforms listed in `files_clusteri_channelj.txt`

Determine parameters for estimating source separations

- `separation_parameters.m`

`separtion_parameters.m` finds the combination of [number, length, start] of time-windows for running `CWI_sep.m`. A 3D std-matrix is generated. Each element is the average standard deviation of inter-source separations over all available pairs in the given cluster recorded by the given station channel(s).

The following inputs are required:

- `coda_start`: start of the searching range of windowing (in seconds)
- `coda_end`: end of the searching range of windowing (in seconds)
- `dl_coda_start`: increment of the start of windowing
- `min_num_win`: minimum number of time-windows allowed
- `min_l_win`: minimum length of time-windows allowed
- `max_num_win_limit`: maximum number of time-windows allowed, regardless the number of time-windows contained within the searching range

- `dl_win`: increment of the length of time-windows to be searched
- `plt_flag`: plot the slice of the 3D std-matrix containing the lowest value?
  - '`y`' for plotting
  - '`n`' for not plotting
- `save_flag`: save the workspace?
  - '`y`' for saving
  - '`n`' for not saving

The following inputs are required to call `CWI_seps.m`, for details see code description for `CWI_seps.m`: `veloc`, `source_type`, `search_range`, and `num_interpo_point`.

The following files are required (generated by `first_arrival_pick.m`):

- `first_arrivals_clusteri_channelj.xls` (or `.csv`)

The following scripts/functions are required:

- `std_maxtrix_3d.m`
- `std_matrix_element.m`
- `CWI_sep.m`

The following files are generated on request (`save_flag` set to '`y`'):

- `seps_param_clusteri_channelj.mat`: stores the workspace after running `separtion_parameters.m` for cluster *i* channel *j*.

- `CWI_sep.m`

`CWI_sep.m` is one of the core functions in this package, called by multiple scripts and functions. It applies coda wave interferometry (CWI) to estimate the separation between a pair of

sources with similar mechanisms for isotropic sources in a 2D or 3D acoustic medium, or double-couple sources in an elastic medium, assuming both sources are on the same fault plane.

The following inputs are required:

- `S1`, `S2`: waveforms of the event pair
- `time`: time series of waveforms
- `dt`: length of interval between two samples in seconds
- veloc: seismic velocity in the source region

  - for acoustic media veloc is a scaler;

  - for elastic media, veloc is a 1x2 vector consisting of [P-wave-velocity S-wave-velocity]
- `source_type`: ype of source mechanism and propagating medium

  - '`2d`' for isotropic sources in 2D an acoustic medium

  - '`3d`' for isotropic sources in 3D an acoustic medium

  - '`doublecouple`' for double-couple sources in an elastic medium
- `win_length`: length of each time-window in coda
- `win_start`: starting time of windowing
- `win_end`: ending time of windowing
- `search_range`: number of data to search on each side of the point where time-shift is zero when searching for the maximum of the correlation coefficient (`Rmax`) for each time-window
- `num_interpo_point`: number of data to interpolate between adjacent samples when looking for `Rmax` for each time-window.

The following output are generated:

- `source_sep`: an array containing estimated source separation from all time-windows
- `sep_mean`: the mean of the estimated source separations
- `sep_std`: the standard deviation of the estimated source separations.

- `std_maxtrix_3d.m`

`std_maxtrix_3d.m` is called by `separation_parameters.m` generates a three-dimensional matrix. Each element is the average standard deviation of inter-source separations over all available pairs in the given cluster recorded by the given station channel(s).

The inputs required are provided in `separation_parameters.m`.

The following scripts/functions are required:

- `std_matrix_element.m`
- `CWI_sep.m`

The following output are given:

`STD_matrix_3D`: a three-dimensional matrix. Each element is the average standard deviation of inter-source separations over all available pairs in the given cluster recorded by the given station channel(s), calculated with one possible combination of [number, length, start] of time-windows

- dimension 1: possible number of time-windows
- dimension 2: possible length of time-windows
- dimension 3: possible starting time of windowing

- `std_matrix_element.m`

`std_matrix_element.m` is called by `std_maxtrix_3d.m`. It calculates elements of `STD_matrix_3D`, with given [number, length, starting time] of time-windows.

The inputs required are provided in `std_maxtrix_3d.m`.

The following scripts/functions are required:

- `CWI_sep.m`

The following output are generated:

- `STD_mean`: an element of `STD_matrix_3D`, for the given parameter combination.

Estimate inter-source separations

- `separations.m`

`separations.m` estimates the inter-source separations of all available event pairs in the given cluster recorded by the given station channel. It generates a `.xls` (or `.csv`) files containing the means and standard deviations of all event pairs from the cluster. For event pairs whose waveforms do not all exist, the results are marked with `-1`.

The following inputs are required:

- `first_arrival_ind`: sample index of the first arrival for all given waveforms
- `txtfilename`: name of the .txt file listing SAC filenames, for which the waveform first arrival to be picked
- `data_location`: directory storing the SAC files.

The following inputs are required to call `CWI_seps.m`, for details see code description for `CWI_seps.m`: `velocity`, `source_type`, `win_length`, `win_start`, `win_end`, `search_range`, and `num_interpo_point`.

The following files are required:

- `files_clusteri_channelj.txt` (generated by `clustering.m`)
- `first_arrivals_clusteri_channelj.xls` (or `.csv`) (generated by `first_arrival_pick.m`).

The following scripts/functions are required:

- `CWI_seps.m`

The following files are generated:

- `seps_clusteri_channelj.xls` (or .csv): stores the means and standard deviations of source separation estimates of all time-windows for each available waveform pairs. The mean and standard deviation is marked with "`-1`" for pairs where at least one waveform does not exist.
    - column 1: means of source separation estimates
    - column 2: standard deviations of source separation estimates.

Appendix 3 Descriptions of codes and variables for location

- **`Source_Location.m`**

`Source_Location.m` estimates the relative location of a cluster of events, using inter-source separation data (their mean and standard deviations) estimated with CWI. The location is solved as a minimization problem, where the most probable set of event locations are found where the objective function L attains its minimum.

The following inputs are required:

- `initial_loc`: an N x 3 array storing the initial locations are minimization
  - column 1 - x-coordinates of the initial event locations
  - column 2 - y-coordinates of the initial event locations
  - column 3 - z-coordinates of the initial event locations
- `MU_n`: an $N(N-1)/2 \times K$ array, each column storing the means of the CWI separation estimates for all available event pairs, with N being the number of events, K being the number of channels
- `SIGMA_n`: an $N(N-1)/2 \times K$ array, each column storing the standard deviations of the CWI separation estimates for all available event pairs
- `discarded_pairs`: indices of event pairs to be discarded

  an $0 \times K$ array - no pair to be discarded, with K being the number of channels

  an $n \times 1$ array - single channel case with n pair(s) to be discarded

  an $n \times K$ array - K channle case, with n being the largest number of pairs to be discarded for any individual channel
- `max_iteration`: maximum number of iterations allowed

- `L_tolerance`: a trivial constant, the threshold for the iterations to terminate when the reduction in the value of L is smaller than.

The following outputs are generated:

- optm_loc: an $N\times3$ array, storing the location results found by optimization
  - column 1 - x-coordinates of the final event locations
  - column 2 - y-coordinates of the final event locations
  - column 3 - z-coordinates of the final event locations
- Func_L_add: an $m\times1$ array, storing the value of L of after each iteration, with m being the number of iterations.

The following scripts/functions are required:

- `line_search.m`
- `gradient.m`
- `ln_joint_likelihood.m`
- `likelihood_individual.m`.

Parameters of the empirical relations (equation 5) between the true separation and the mean and standard deviation of the CWI estimates are given inside function `Source_Location.m`. Here we use the ones derived by Robinson et al. (2011), however, users can change them to those derived from their own experiments.

- **`ln_joint_likelihood.m`**

`ln_joint_likelihood.m` estimates the negative logarithm of the joint likelihood function L for a cluster of events. For each station channel, all available event pairs are considered, except for those instructed to be discarded in the input of the function.

The following inputs are required:

- `X (Y,Z)`: x (y, z)-axis coordinates of the event locations in the cluster
- `domin_wavelength`: dominate wavelength of the propagating waves
- `a_mu1`: parameters of the empirical relation between true separation and the mean of the CWI estimates
- `a_sigma1,c`: parameters of the empirical relation between true separation and the standard deviation of the CWI estimates
- `MU_n`: an $N(N-1)/2\times K$ array, each column stores the means of the CWI separation estimates for all available event pairs, with N being the number of events, K being the number of channels
- SIGMA_n: an $N(N-1)/2\times K$ array, each column stores the standard deviations of the CWI separation estimates for all available event pairs
- `discarded_pairs`: indices of event pairs to be discarded

  an $0\times K$ array - no pair to be discarded, with K being the number of channels

  an $n\times 1$ array - single channel case with n pair(s) to be discarded

  an $n\times K$ array - K channle case, with n being the largest number of pairs to be discarded for any individual channel.

The following outputs are generated:

- `Ln_prob_joint`: negative logarithm of the joint likelihood function for data from single or multiple station channels.

The following scripts/functions are required:

- `likelihood_individual.m`

- **likelihood_individual.m**

likelihood_individual.m estimates the likelihood function for an individual pair of events.

The following inputs are required:

- x (y, z): x (y, z)-axis coordinates of the location of the two events
- domin_wavelength: dominate wavelength of the propagating waves
- a_mu1: parameters of the empirical relation between true separation and the mean of the CWI estimates
- a_sigma1,c: parameters of the empirical relation between true separation and the standard deviation of the CWI estimates
- mu_n: the mean of the CWI separation estimates
- sigma_n: the standard deviation of the CWI separation estimates.

The following outputs are generated:

- prob_indiviudal: the likelihood function of an individual pair of events.

- **gradient.m**

gradient.m computes the gradient of *L* at a given set of event locations

The following inputs are required:

- Xc (Yc,Zc): x (y, z)-axis coordinates of the current event locations
- domin_wavelength: dominate wavelength of the propagating waves
- a_mu1: parameters of the empirical relation between true separation and the mean of the CWI estimates

- `a_sigma1,c`: parameters of the empirical relation between true separation and the standard deviation of the CWI estimates
- `MU_n`: an $N(N-1)/2\times K$ array, each column stores the means of the CWI separation estimates for all available event pairs, with N being the number of events, K being the number of channels
- SIGMA_n: an $N(N-1)/2\times K$ array, each column stores the standard deviations of the CWI separation estimates for all available event pairs
- `discarded_pairs`: indices of event pairs to be discarded

  an $0\times K$ array - no pair to be discarded, with K being the number of channels

  an $n\times 1$ array - single channel case with n pair(s) to be discarded

  an $n\times K$ array - K channle case, with n being the largest number of pairs to be discarded for any individual channel.

The following outputs are generated:

`gradient`: gradient of L at the given point (`Xc, Yc, Zc`). An $N\times 3$ matrix, with the 1st, 2nd, and 3rd column storing the partial derivatives in terms of x, y and z-coordinates of the event locations.

The following scripts/functions are required:

- `ln_joint_likelihood.m`
- `likelihood_individual.m`.

- **`line_search.m`**

`line_search.m` conducts searches for the minimum of an objective function L in a given direction in three steps:

1) along the given direction, evaluate L in a large range with increments of `delta_lambda_coarse`, and find the (coarse) minimum among these values of L.

2) define a smaller range around the minimum found, evaluate L in the smaller range with increments of `delta_lamba_dense`, and find a new (dense) minimum among these values of L.

3) adjust the dense minimum by fitting a parabola using the minimum point found and two of its adjacent points, and take the vertex of the parabola as the final minimum of L along the given direction.

The following inputs are required:

- `Xc (Yc,Zc)`: x (y, z)-axis coordinates of the current event locations
- `dc`: current searching direction
- `domin_wavelength`: dominate wavelength of the propagating waves
- `min_lambda_coarse`: lower bound of the large (coarse) searching range
- `max_lambda_coarse`: upper bound of the large (coarse) searching range
- `delta_lambda_coarse`: increment of the large (coarse) searching range
- `delta_lambda_dense`: increment of the small (dense) searching range
- `a_mu1`: parameters of the empirical relation between true separation and the mean of the CWI estimates
- `a_sigma1,c`: parameters of the empirical relation between true separation and the standard deviation of the CWI estimates
- `MU_n`: an $N(N-1)/2 \times K$ array, each column stores the means of the CWI separation estimates for all available event pairs, with N being the number of events, K being the number of channels

- SIGMA_n: an $N(N-1)/2 \times K$ array, each column stores the standard deviations of the CWI separation estimates for all available event pairs
- `discarded_pairs`: indices of event pairs to be discarded

  an $0 \times K$ array - no pair to be discarded, with K being the number of channels

  an $n \times 1$ array - single channel case with n pair(s) to be discarded

  an $n \times K$ array - K channle case, with n being the largest number of pairs to be discarded for any individual channel.

The following outputs are generated:

lambda_step: step length the current iteration should take to attain the minimum in the given searching direction

fvalue_new: value of L at the minimum found in the given searching direction.

The following scripts/functions are needed:

- `ln_joint_likelihood.m`
- `likelihood_individual.m`.

- **`initialize_locations.m`**

`initialize_locations.m` generates a set of initial locations for the minimization problem solved by `Source_Location.m`. It first creates a set of locations for the given event number, uniformly distributing in the given range. It then reorders the created events to ensure the least square residuals of inter-source separations (residuals being the difference in separation of each event pair between that calculated from the created initial locations and that estimated with CWI from seismic data).

The following inputs are required:

- `x_range (y_range,z_range)`: an 1×2 array, storing the range for x (y, z) - coordinates
- `num_event`: number of events in the given cluster
- `MU_n`: an $N(N-1)/2\times1$ array, storing the means of the CWI separation estimates for all available event pairs, averaged over all channels if data from multiple channels are used, with N being the number of events.

The following outputs are generated:

- initial_position: an $N\times3$ array, storing the locations to be used as initialization of the minimization problem
  - column 1 - x-coordinates
  - column 2 - y-coordinates
  - column 3 - z-coordinates

- **`rotate_cluster.m`**

`rotate_cluster.m` rotates a cluster of points about its center in two orthogonal directions in the 3D space by given angles.

The following inputs are required:

- `P0`: an $n\times3$ array, storing the original point coordinates, where n is the number of events.
- `alpha`: angel by which the cluster rotates anticlockwise when observed from above about a line passing the cluster center and parallel to z-axis.
- `theta`: angel by which the cluster rotates anticlockwise about a line passing the cluster center and parallel to x-axis.

The following outputs are generated:

- `P_rotate:` an $n\times3$ array, storing the rotated point coordinates,where n is the number of events.

- **`coordinate_modify.m`**

`coordinate_modify.m` adjusts the coordinate system of event cluster to remove the ambiguity, following four steps (Robinson et al., 2013):

1) Move the cluster so that event 1 is at the origin, i.e. $e_1 = (0,0,0)$;

2) Rotate the cluster so that event 2 lies on positive x-axis, i.e. $e_2 = (x2, 0,0)$, where $x2 > 0$;

3) Rotate the cluster so that event 3 lies in x-y plane, and the y-coordinate is positive, i.e. $e_3 = (x3, y3,0)$, where $y3 > 0$;

4) Make sure z-coordinate of event 4 is positive, i.e. $e_4 = (x4, y4, z4)$, where $z4 > 0$.

The following inputs are required:

- `X (Y,Z):` $N\times1$ array, storing the x (y,z)-coordinate of event locations to be adjusted
- `num_event`: number of events in the cluster

The following outputs are generated:

- `X4 (Y4,Z4)`: $N\times1$ array, storing the x (y,z)-coordinate of the adjusted event locations