ELECTRONIC SEISMOLOGIST

Super-Efficient Cross-Correlation (SEC-C): A Fast Matched Filtering Code Suitable for Desktop Computers

by Nader Shakibay Senobari, Gareth J. Funning, Eamonn Keogh, Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, and Abdullah Mueen

ABSTRACT

We present a new method to accelerate the process of matched filtering (template matching) of seismic waveforms by efficient calculation of (cross-) correlation coefficients. The cross-correlation method is commonly used to analyze seismic data, for example, to detect repeating or similar seismic waveform signals, earthquake swarms, foreshocks, aftershocks, low-frequency earthquakes (LFEs), and nonvolcanic tremor. Recent growth in the density and coverage of seismic instrumentation demands fast and accurate methods to analyze the corresponding large volumes of data generated. Historically, there are two approaches used to perform matched filtering; one using the time domain and the other the frequency domain. Recent studies reveal that time domain matched filtering is memory efficient and frequency domain matched filtering is time efficient, assuming the same amount of computational resources. We show that our super-efficient cross-correlation (SEC-C) method—a frequency domain method that optimizes computations using the overlap–add method, vectorization, and fast normalization—is not only more time efficient than existing frequency domain methods when run on the same number of central processing unit (CPU) threads but also more memory efficient than time domain methods in our test cases. For example, using 30 channels of data with a sample rate of 50 Hz and 30 templates, each with durations of 8 s, SEC-C uses only 2.3 GB of memory whereas other frequency domain codes use three times more and parallelized time-domain codes use ~30% more. We have implemented a precise, fully normalized version of SEC-C that removes the mean of the data in each sliding window, and thus can be applied to raw seismic data. Another strength of the SEC-C method is that it can be used to search for repeating seismic events in a concatenated stack of individual event waveforms. In this use case, our method is more than one order of magnitude faster than conventional methods. The SEC-C method does not require specialized hardware to achieve its computation speed; instead it exploits algorithmic ideas that are both time- and memory-efficient and are thus suitable for use on off-the-shelf desktop machines.

Electronic Supplement: Additional figures and MATLAB codes (matched filter algorithm).

INTRODUCTION

Matched filtering, also known as template matching, similarity search, or "query-by-content", is a commonly used method in seismology. The matched portions of a continuous waveform data set with a template waveform can be identified by calculating normalized correlation coefficients, usually referred to by seismologists as zero-lag cross-correlation coefficients (CCCs). By choosing appropriate thresholds for these CCC values, we can detect similar or repeating patterns in those continuous data. Template matching is often used for detecting seismic events with low signal-to-noise waveforms within large volumes of continuous data. A high-detection capability along with applicability to a wide variety of seismic source types makes template matching a powerful tool for seismologists. For example, template matching can be used to detect seismic events such as foreshocks, aftershocks, icequakes, repeating earthquakes (REs), volcanic earthquakes, geothermal seismic activity, swarms, low-frequency earthquakes (LFEs), and nonvolcanic tremor to monitor nuclear explosions and to identify seismic triggering (e.g., Nadeau *et al.*, 1995; Gibbons and Ringdal, 2006; Shelly *et al.*, 2007; Peng and Zhao, 2009; Allstadt and Malone, 2014; Meng and Peng, 2014; Skoumal *et al.*, 2015; Kato *et al.*, 2016; Frank *et al.*, 2017).

In addition to the event detection applications explained above, cross-correlation analysis has become an important part of determining event locations and relocations in the last two decades (e.g., Waldhauser and Ellsworth, 2000; Hauksson and Shearer, 2005; Schaff and Waldhauser, 2005). The relative arrival time of seismic phases to seismic stations for each event in a group of nearby events is the main input information for relocation algorithms (e.g., Waldhauser and Ellsworth, 2000; Hauksson and Shearer, 2005) and is traditionally estimated by comparing the picked phase arrival times from earthquake catalogs. The picked phase arrival times usually contain errors due to station noise and uncertainty in the phase picking algorithm or human error. On the other hand, CCC methods can precisely calculate the relative time shift between individual waveforms, as the CCC between them is maximized when two waveforms are aligned. These methods can also be used to precisely detect temporal velocity changes in the Earth's crust (e.g., Poupinet *et al.*, 1984; Schaff and Beroza, 2004; Thomas *et al.*, 2012) or even the Earth's inner core (Tkalčić *et al.*, 2013). Precise information about relative phase arrival times also plays an important role in seismic tomography (e.g., Zhang and Thurber, 2003). Therefore, a fast method for performing template matching in continuous data and for pairwise cross correlations of individual waveforms could potentially lead to computation time improvements in many branches of seismology.

For most earthquake detection and (re)location applications (e.g., aftershocks, foreshocks, swarms, event relocations), choosing a waveform template can be straightforward, for example, selecting a well-recorded event within an area of interest (e.g., Shelly *et al.*, 2007; Schaff and Waldhauser, 2010; Meng and Peng, 2014). However, in cases where it is known that a specific type of event repeats over time (e.g., REs or LFEs), but cannot be identified *a priori*, seismologists have used different techniques such as array processing (e.g., Frank and Shapiro, 2014), pairwise similarity search (also known as "autocorrelation"; e.g., Brown *et al.*, 2008), or careful visualization of seismic data (e.g., Shelly *et al.*, 2009) to identify templates.

The duration of continuous data available for investigation by template matching is of the order of decades (e.g., Shelly, 2017). If, for example, we could reduce the run time of a template matching analysis from 300 to 100 s for each day of seismic data, this would add up to ~8 days of computation time savings for one decade of seismic data. In other words, given the power law increase in the volume of waveform data archived in seismic data repositories (e.g., Incorporated Research Institutions for Seismology Data Management Center [IRIS-DMC]; Hutko, *et al.*, 2017), there is a high demand for fast, precise, and user-friendly seismic analysis methods.

For a single waveform of continuous seismic data with n samples and a single template with a length of m samples, the time complexity of template matching in the time domain is $O((n-m)m)$, or approximately $O(nm)$ when $n \gg m$. In the frequency domain, the time complexity is $O(n \log n)$ (Lewis, 1995). The normalization of computed CCCs adds an additional delay to the computational run time. The relative computational time complexity between these two methods then can be determined by comparing the size of $\log(n)$ with m.

In this article, whenever we talk about time complexity comparisons between the time- and frequency-domain methods, we assume that these two methods are written in a common programming language and use the same computational resources (e.g., a single thread of a central processing unit [CPU]). For seismological applications, a template waveform usually contains several seconds of seismic data (i.e., $m \approx 10$–1000 samples) and the data set to be compared to is typically weeks, months, or years of seismic data that are usually stored as daily continuous seismic data files (i.e., $n \approx 10^6$–10^7 samples per day). This implies that the frequency domain method should be the faster method unless a template with a short length (e.g., less than a second, for one day of data with 50 Hz sample rate) is used. On the other hand, frequency domain methods, which typically involve the template being padded with zeros at least to the length of the comparison data, require much more memory (e.g., Lewis, 1995; Chamberlain *et al*, 2018). Despite not being time efficient with respect to frequency domain methods, time-domain template matching is often considered suitable for CPU and graphics processing unit (GPU) parallelization as the implementation is straightforward and memory efficient (e.g., Meng *et al.*, 2012; Mu *et al*, 2017; Beaucé *et al.*, 2018). Here we demonstrate, using several algorithmic improvements, a single CPU frequency domain method that we call super-efficient cross-correlation (SEC-C). It is equivalent in speed to modern CPU-parallelized codes running on more than 10 processor cores, and only a few times slower than GPU parallelized codes. Such a code, running on a regular desktop computer with a few processor cores, can be a powerful tool for template matching that in some cases (e.g., long duration of templates, 100 Hz sample rates) is as powerful as GPU parallelized codes without requiring extensive memory or additional hardware.

SEC-C is fast, memory efficient, and, for a minimal increase in computation time, can be precise to machine precision. A fully normalized version of the algorithm removes the mean of the data for each sliding window internally and therefore can be used for template matching of raw seismic data (Ⓔ Fig. S1, available in the electronic supplement to this article). Several of the parallelized methods mentioned above require prior operations on the data, or specific conditions or assumptions (e.g., removing the mean from the data, low variability on the amplitude, or using single-precision floating points; Beaucé *et al.*, 2018; Chamberlain *et al.*, 2018; Mu *et al.*, 2017; respectively). Chamberlain *et al.* (2018) reported that using single-precision floating points for normalization calculations can introduce errors in the CCC results of up to 20% for a large earthquake within low-amplitude noise. Beaucé *et al.* (2018) tested the removal of the mean from the data from an M_w 7.8 earthquake and showed that the CCC error is 1.2%. SEC-C is capable of outputting both the CCC sum as well as the individual CCC for each channel without introducing extra run time. The latter case is useful when the moveout of *P*-wave arrivals is not precisely known and when the stations are far from each other (i.e., the moveout is very large), meaning that matched filtering at individual stations is a better option.

Our aim in this study is to calculate fast, memory efficient, and precise CCCs for template matching applications in seismology. SEC-C employs a combination of several speed-up techniques such as the fast Fourier transform (FFT), the overlap–add method (Rabiner and Gold, 1975), vectorization tricks, and a fast normalization method inspired by Mueen's algorithm for similarity search (MASS; Mueen *et al.*, 2015). This method can be coded in any array programming computer language (e.g., MATLAB, see Data and Resources; Fortran 90, R, the NumPy extension to Python). It does not require any special libraries except for the FFT. The SEC-C MATLAB code is provided in the Ⓔ electronic supplement. The MATLAB code along with a Python version is also available in a GitHub repository (see Data and Resources).

THE ALGORITHM

The Traditional Time-Domain Sliding Window Cross-Correlation Method

Assume that we have a seismic template waveform X with a length of m samples and a continuous time series Y with a length of n samples. A traditional brute force way of performing template matching is to calculate the CCC of X with a subwindow of Y that has the same length (i.e., of m samples) and repeat this procedure with a sliding subwindow shifted by one sample or some small interval (e.g., 0.02 s; Shelly *et al.*, 2009). Assume that Y^i is the ith subwindow of Y, then the CCC for any subwindow is defined as below:

$$\mathrm{CCC}_i = \frac{(X-\bar{X})\cdot(Y^i-\overline{Y^i})}{\sqrt{((X-\bar{X})\cdot(X-\bar{X}))((Y^i-\overline{Y^i})\cdot(Y^i-\overline{Y^i}))}}. \quad (1)$$

in which the bar symbol above X and Y^i refers to the mean values for each and the dots indicate scalar (dot) products. For example, $\bar{X}$ refers to a vector with a length of m in which each component is the mean of X. We assume that the local mean is already removed from the data and templates can thus be reduced to the equation below:

$$\mathrm{CCC}_i = \frac{X\cdot Y^i}{\sqrt{(X\cdot X)(Y^i\cdot Y^i)}}. \quad (2)$$

This procedure typically requires looping of this calculation over many subwindows of Y. For most real seismological applications, this needs to be repeated for multiple stations with multiple components and several templates. This becomes time consuming for a long continuous waveform and with the sample rates required for most seismic applications (e.g., usually greater than 20 Hz). For example, calculating CCCs for one day of continuous waveform with a 100 Hz sample rate and sliding for a 0.02 s interval for 10 stations, three components and for 20 templates, requires the evaluation of equation (1) 24(hrs) × 60(min) × 60(s) × 50 (CC evaluations per second) × 10 (stations) × 3 (components) × 20 (templates) = ~2.6×10^9 times. For one year of data, the number of calculations increases to $\sim10^{12}$. The time complexity of equation (1) has a linear relationship with template length (m), and as m increases the total computational time increases proportionately. To tackle the run time problem of computing many nested loops, recent time-domain-based methods have focused on parallelization using either CPU clusters or GPU architecture that can compute this calculation using hundreds to thousands of threads simultaneously (e.g., Meng *et al.*, 2012; Mu *et al.*, 2017; Beaucé *et al.*, 2018). However, performing a real-world case of template matching using a regular desktop machine in the time domain is still a challenge and out of reach.

The alternative way of performing template matching is to use the frequency domain to calculate the numerator of equation (2) without looping over sliding windows. Here, we give a brief introduction to frequency domain template matching, using the CCC metric.

The Traditional Frequency Domain Cross-Correlation Method

We first define two vectors with the same length, extended to the next highest power of two:

1. X' = reverse X and append $(n+l-m)$ zeros to the end,
2. Y' = append Y with l zeros at the end.

   Here, l is the number of zeros that needs to be added to the Y to make the length of Y a power of two. In the past, the FFT algorithm performed optimally when the length of the data was a power of two. New FFT libraries, however, can calculate the FFT efficiently if the prime factors are small (e.g., the Fastest Fourier Transform in the West (FFTW); Frigo and Johnson, 2005). Therefore, depending on the FFT libraries and n, l can be chosen to be 0 or any number that can make $n+l$ a power of two. Then, the convolution of X' and Y' would produce all the possible numerators of equation (1) (Lewis, 1995):

$$(X' * Y')_i = X\cdot Y^i. \quad (3)$$

   As mentioned above, subscript i is referred to here as the ith subwindow. We call this vector the sliding dot product of X and Y, sdp(X, Y). We can calculate this sliding dot product using the FFT method as below (Lewis, 1995; Smith, 1997):

3. $\mathrm{sdp}(XY) := (X' * Y') = FFT^{-1}(FFT(X')\cdot FFT(Y'))$
   The three procedures above allow us to calculate the numerator of equation (1).

Algorithms for calculating the denominator of equation (2) (i.e., the normalization part) may vary from method to method. As the time complexity of the FFT is $O(n\log n)$, if we assume a normalization with a linear time complexity, then the overall time complexity of the frequency domain method is $O(n\log n)$. If we compare this to the time domain time complexity (i.e., $O(nm)$), when m is greater than $\log n$, the frequency domain approach becomes a better choice of method. For a single day of seismic data, depending on the sample rate (e.g., from 20 to 100 Hz), $\log n$ varies between ~14 and 16. This means that the frequency domain approach is more efficient if $m > 16$—corresponding to a template length of 0.32 s when a sample rate of 50 Hz is assumed. The exact template length m at which the frequency domain becomes more time efficient depends on the hardware and the FFT libraries (Smith, 1997). For most seismic applications, however, template lengths of more than several seconds of data are required, implying that methods that make use of the frequency domain are more time efficient. (Note as we mentioned above our assumption is that both methods use the same amount of CPU resources, e.g., one CPU thread.)

On the other hand, frequency domain methods are not typically memory efficient. During the procedure (1), the template length increases at least to the length of the data (i.e., n if l is assumed to be zero). Our tests (see the SEC-C Memory Efficiency section) show that these types of frequency domain methods (e.g., EQcorrscan; Chamberlain *et al.*, 2018) can almost exceed the memory of a desktop computer with 16 GB of RAM in some use cases (e.g., using 40 templates for 10 stations with three components with a sample rate of 50 Hz and template length of 8 s). Even if a test case includes a small number of channels of data (not 30 channels as above), having a memory efficient method allows the user to perform matching for more templates at the same time and therefore at a reduced run time overall. The frequency method memory limitations can be circumvented using algorithmic improvements, however, below we describe how our "super efficient" algorithm is efficient in terms of both computation time and memory.

The SEC-C Algorithm

Here, we use several methodological tricks to reduce both the run time and memory usage when calculating CCCs for the multistation and multitemplate case of matched filtering of seismic data using a frequency domain-based method. First, to reduce the run time and memory overhead required, we use a "block convolution" procedure (also called "sectioned convolution"; e.g., Rabiner and Gold, 1975) using the "overlap–add" method (e.g., Rabiner and Gold, 1975) to calculate the sliding dot product using the FFT method (i.e., procedure 3). This method is used in signal processing techniques to perform the convolution of a long signal with a finite impulse response filter (e.g., Rabiner and Gold, 1975; Smith, 1997). The main idea is to divide a long signal into small pieces and then perform the FFT convolution for each piece. To ensure accurate calculation of the sliding dot product at the border of two neighboring pieces there should be an overlap of $m-1$ samples between neighboring pieces. If we assume that the length of each piece is k and if we ignore the recomputation in areas of overlap, the time complexity for a single trace of the data then becomes $O((n/k)(k \log k)) = O(n \log k)$, as we need to compute (3) for n/k pieces. Here, k becomes a tunable parameter that should be carefully chosen for optimal performance. If $k \ll n$ (e.g., comparable in size with m), performing many repetitions (loops) will slow down the process. If k is large and comparable in size with n then calculating the FFT for each piece will be time consuming. The optimal value for k can be determined by trial and error, and depends mainly on hardware aspects (e.g., CPU cache size and clock speed). Using a trial and error procedure that we performed using two different desktop machines, we recommend assigning a power of two for k (e.g., 2^{12} for one day of 20 Hz data or 2^{13} for one day of 50–100 Hz data) for efficient performance. However, in all cases we advise running some test cases to find values of k to optimize the run time.

One other feature of the overlap–add method is that the template is not required to be padded by zeros to the length of the data, which for large n can require hundreds of MB of memory space. Using the shorter waveform pieces of the overlap–add method reduces this requirement to padding until the length of k (equivalent to tens of KB of memory usage if $k = 2^{13}$). This can result in a large memory savings when multiple stations and templates are used. The k value can also be chosen to be very small to minimize memory usage, although this will come at the expense of increased run times.

The second trick to speed up the template matching is to use vectorized calculations of sliding dot product for all overlap–add pieces instead of looping over these pieces. For example, MATLAB has options for vectorized FFT, dot product, and inverse FFT and therefore the whole procedure of (3) can be vectorized. For the case of multitemplate matched filtering, the FFTs of the templates can also be calculated in a vectorized basis as well.

A third optimization trick is that we apply a very efficient normalization (i.e., denominator of equation 1) inspired by the MASS algorithm (Mueen *et al.*, 2015), which we describe below. $X \cdot X$ is a constant and can be precalculated. For calculating $Y^i \cdot Y^i$, we use the following procedures:

1. Calculate the cumulative sum of Y squared and prepend a zero to it as below:

$$C_{k+1} = \begin{cases} \sum_{j=1}^{k} Y_j Y_j, & (1 \le k \le n) \\ 0, & (k = 0) \end{cases}$$

2. Then, $Y^i \cdot Y^i$ can be calculated as below:

$$Y^i \cdot Y^i = C_{i+m} - C_i.$$

These will give us the denominator of equation (2) with the time complexity of $O(n)$. Recent versions of MATLAB (2017a and later) include a built-in function, *movsum*, that performs this procedure with a similar time complexity and run time. *movsum* returns the sliding m-points sums of a vector. We use this function for simplicity in the current version of SEC-C. For other programming languages and older versions of MATLAB, the procedure explained above can be used.

The output of the algorithm we describe above is the sliding CCC for the template X and the continuous waveform Y. The computation has the time complexity of $O(n \log k)$. We then loop over the stations, components, and templates to calculate the various CCCs required in the multichannel and multitemplate cases. Along with these required loops, some of the operations, such as zero padding, reversing, and calculating FFTs of templates, are performed in a vectorized basis. SEC-C can output either the CCC for each channel individually for each template or produce weighted CCC sums of all channels. For the second option, weightings should be provided by the user. For more details of the algorithm, we refer to the MATLAB code provided in the Ⓔ electronic supplement.

SEC-C is a single CPU code that is optimized for seismic data sets with lengths of approximately one day that can handle hundreds of channels of data and templates in an efficient run time. If faster run times are needed, the user can simply parallelize the problem by running SEC-C in parallel for each different day of data on each single CPU core of a multicore desktop machine. A toy example of running SEC-C using the MATLAB parallelized for-loop, *parfor*, is provided in the SEC-C GitHub repository (see Data and Resources).

The Fully Normalized Version of the SEC-C Algorithm

The algorithm explained above is based on equation (2) and includes the assumption that the local mean is removed from the data and templates. This can be acceptable for most cases of seismological applications, but in some cases, for example, when there are sudden large fluctuations in the data, such as instrument spikes or large nearby earthquakes, this assumption can be problematic. Most current approaches based on the assumption that the mean and any spikes are removed from data (e.g., Beaucé *et al.*, 2018; Chamberlain *et al.*, 2018). Beauce *et al.* (2018) indicate that this assumption can affect the results of CCC calculations for the 2016 M_w 7.8 Kaikōura earthquake by as much as 1.2%; however, this is an extreme case where there are large deviations from the mean in the data. Our experiment on Mount St. Helens seismicity (i.e., a more normal test case) shows the differences between CCCs calculated by equation (1), and the SEC-C method with the zero-mean assumption (equation 2), for a single channel of data are of the order of 10^{-4} (Fig. 1e).

Because SEC-C is a versatile algorithm, we can make some simple changes that calculate CCCs based on equation (1) without the need for simplifying assumptions. Here, we briefly discuss this implementation.

First, the mean of the templates can be precalculated and removed. Equation (1) in this case becomes:

$$\mathrm{CCC}_i = \frac{X\cdot(Y^i-\overline{Y^i})}{\sqrt{(X\cdot X)((Y^i-\overline{Y^i})\cdot(Y^i-\overline{Y^i}))}} = \frac{(X\cdot Y^i)-(X\cdot\overline{Y^i})}{\sqrt{(X\cdot X)(Y^i\cdot Y^i-(2Y^i\cdot\overline{Y^i}-\overline{Y^i}\cdot\overline{Y^i}))}}. \qquad (4)$$

There are two extra terms in this equation with respect to equation (2), $X\cdot\bar{Y}^i$ in the numerator and $(2Y^i\cdot\bar{Y}^i-\bar{Y}^i\cdot\bar{Y}^i)$ in the denominator. Before calculating these two terms, we define S^i as a local sum of Y:

$$S^i=\sum_{j=i}^{i+m}Y_j. \qquad (5)$$

The term $X\cdot\bar{Y}^i$ vanishes as the mean of X is removed. In other words

$$X\cdot\bar{Y}^i=\mathrm{mean}(Y^i)(\mathrm{sum}(X))=((S^i)/\mathrm{m})(\mathrm{sum}(X)) = ((S^i))(\mathrm{sum}(X)/\mathrm{m})=(S^i)(\mathrm{mean}(X))=0.$$

The extra term in the denominator of equation (4) can be calculated as below:

$$2Y^i\cdot\bar{Y}^i-\bar{Y}^i\cdot\bar{Y}^i=2(S^i)(S^i/\mathrm{m})-\mathrm{m}(S^i/\mathrm{m})(S^i/\mathrm{m})=(S^i)^2/\mathrm{m}.$$

The mean of Y^i is simply $S^i/\mathrm{m}\cdot S^i$ can be calculated with a similar algorithm as that described above for $Y^i\cdot Y^i$, or by using the *movsum* function in MATLAB, with a linear time complexity.

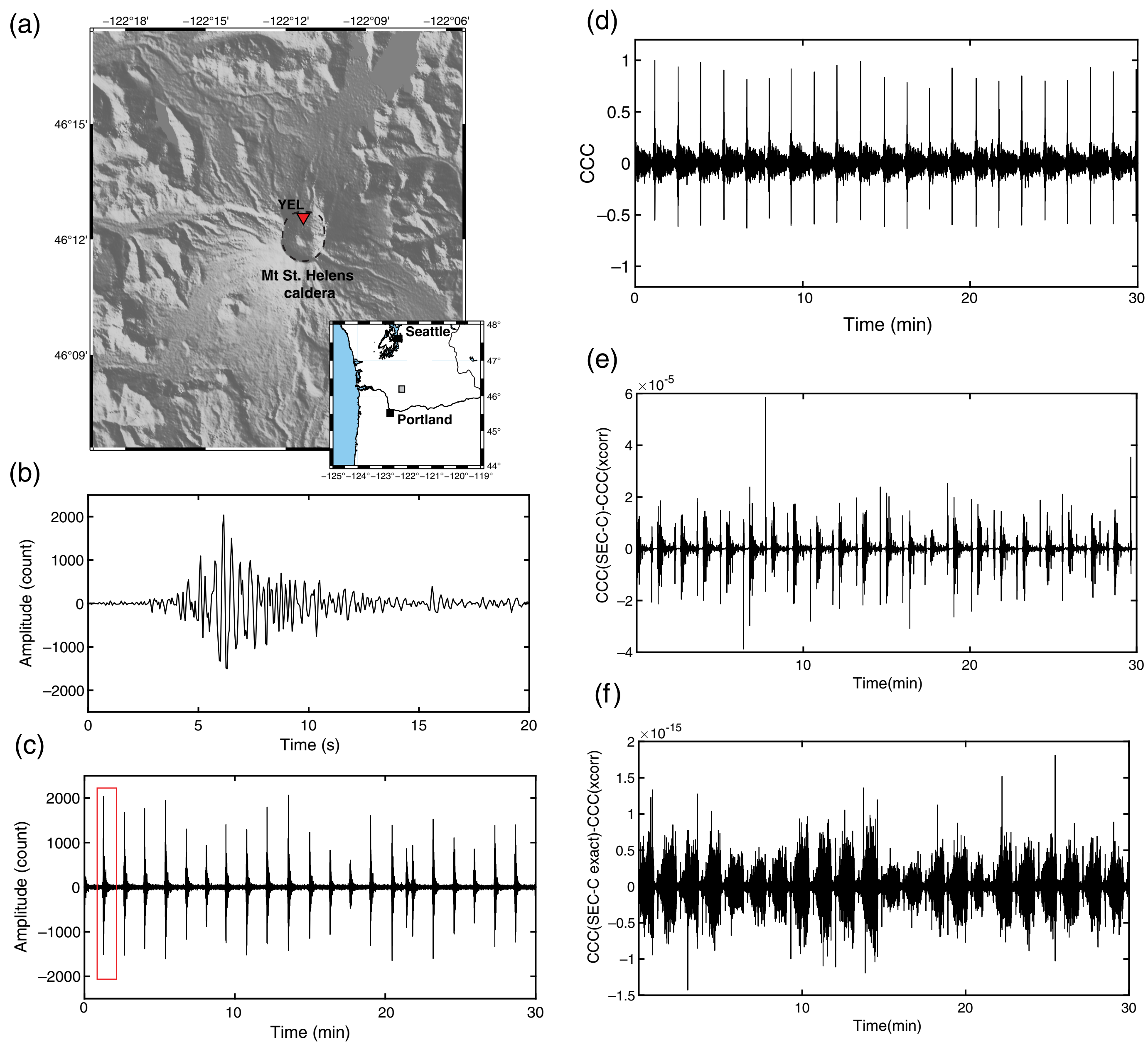
After applying these modifications, the run time of SEC-C increases by less than 1.1% of that of the regular SEC-C algorithm and in our heaviest test case the run time is almost the same (Fig. 2a). This is because the cost of computing the mean of the data is borne only once; once the mean is computed, as many templates as desired can be run at no additional cost. Because this fully normalized version of SEC-C removes the sliding mean from the data, it can be used for raw data (e.g., Ⓔ Fig. S1).

COMPARISONS WITH OTHER APPROACHES

We compare SEC-C with other contemporary methods in terms of accuracy, speed, and memory efficiency, the main characteristics of any matched filter method. For the accuracy test, we compare SEC-C with *xcorr*, a built-in MATLAB function for calculating CCCs. To test SEC-C in terms of speed and memory usage, we compare it with two current, recently published methods: EQcorrscan (Chamberlain *et al.*, 2018), a frequency domain-based matched filter method and fast matched filter (FMF; Beaucé *et al.*, 2018), a time domain-based method.

SEC-C Accuracy and Precision, and the Impact of the Zero-Mean Assumption

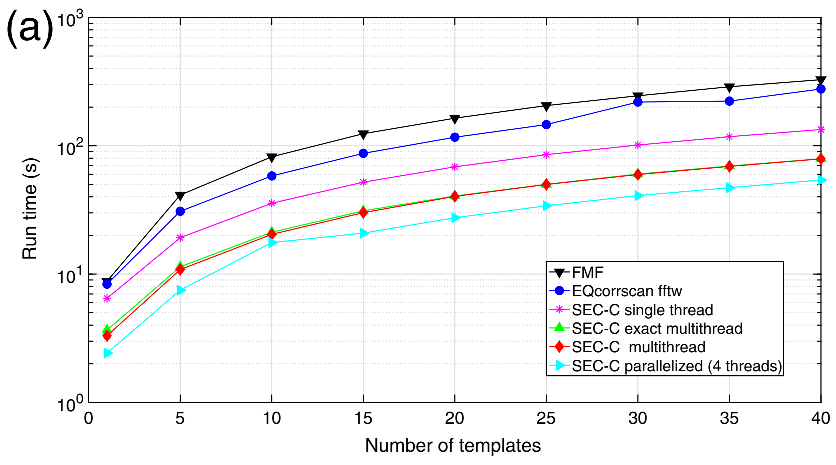
To test the accuracy and precision of this method, we applied the SEC-C algorithm to the seismicity at Mount St. Helens volcano. We select a template waveform from repeating volcanic earthquake swarms that occurred on 3 December 2005, recorded in the vertical channel of the seismic station YEL (Fig. 1a). The high-seismicity rates on this day are related to the dome building eruption in 2004–2005 at Mount St. Helens. So-called "drumbeat" earthquakes, repeating events that occur at regular, short intervals, occurred every 30–300 s during this eruptive episode (Iverson *et al.*, 2006; Fig. 1b). We calculate the CCC between our template and a 24-hr-long continuous waveform (the whole of 3 December) using both SEC-C and a sliding *xcorr* function over each window calculated with zero lag (Fig. 1c,d). We removed the sliding mean of the data for each sliding window prior to calling *xcorr* for that window. This is a brute force and therefore very slow method, but it is effective as a reference method for calculating CCCs precisely and accurately. A comparison between the CCC values output by this precise, traditional method, and the SEC-C method on this identical data set show that the differences of the results are on the order of 10^{-4} and 10^{-15} for the regular and fully normalized versions of SEC-C, respectively (Fig. 1e), which implies that the fully normalized version of our method is precise to machine precision and can reproduce the results of traditional methods on the order of machine precision. SEC-C uses double precision for its calculations, and this, along with the option of removing the mean for each sliding window, underpins its capacity for accurate and precise CCC computations.

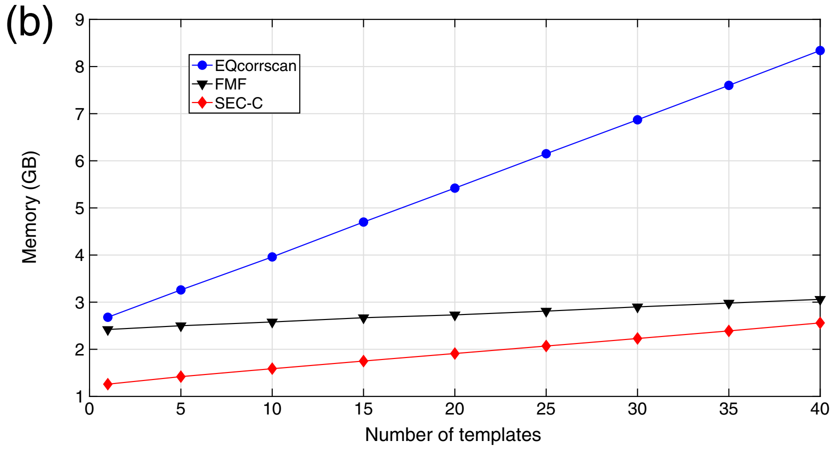


▲ **Figure 1.** (a) A topographic map of the Mount St. Helens volcano area. Inverted triangle shows the location of seismic station YEL. The dashed circle delimits the caldera, the source of drumbeat seismicity. (b) A "drumbeat" earthquake template waveform recorded on the vertical-component channel of station YEL. (c) 30 min of seismic data recorded at the same station on 3 December 2004. Box indicates the template event shown in (b). (d) Cross-correlation coefficient (CCC) function calculated using the traditional sliding window method using the *xcorr* function in MATLAB (with the mean of the sliding window removed) for one day of data, note that in this and subsequent plots we show only a 30 min subset of this CCC function. (e) Difference of CCC calculated with the regular super-efficient cross-correlation (SEC-C) method (sliding window mean not removed) with the CCC from (d). (f) Same as (e) but for the fully normalized version of SEC-C in which the mean is removed from each sliding window. The amplitude of (f) shows that the differences between CCC results are approximately on the order of machine precision (i.e., double precision), indicating the precision of the fully normalized version of the SEC-C method. The color version of this figure is available only in the electronic edition.

SEC-C Speed

We compare SEC-C with two contemporary codes: one that computes CCCs in the frequency domain, EQcorrscan v. 0.2.7, and one that uses the time domain, FMF, in terms of run time and memory usage. Both the EQcorrscan and FMF methods use routines compiled in C for calculating CCCs, accompanied with multithreaded routines and OpenMP (Dagum and Menon, 1998) loops for parallelization. Both packages have a wrapper for use in Python; FMF also has a wrapper in MATLAB. We use the fastest version of the correlation backend of





▲ **Figure 2.** (a) Matched filter run time comparison between SEC-C, the fully normalized version of SEC-C, EQcorrscan, and fast matched filter (FMF), performed on a desktop machine with a quad-core (Intel i7-4790) processor. SEC-C run time is reported for three different computation strategies: SEC-C single thread (all computations run on a single central processing unit [CPU] thread), SEC-C multithread (a single instance of SEC-C, but with some MATLAB functions using multithreading in the background) and a parallelized case of SEC-C, in which four single thread instances are run on a quarter of the data set each at the same time. The test case data set includes one day of data for 10 stations, each with three components, with a 50 Hz sample rate and a template length of 8 s. The run time is plotted on a log scale versus the number of templates on a linear scale. We consider only the CCC sum procedure for the comparison and therefore run time does not include the loading of data or preprocessing, such as median absolute deviation (MAD) calculation or detection. The comparison shows that SEC-C would be the best choice to run the matched filter procedure on a desktop computer as it is 2–6 times faster than other contemporary methods, depending on the computational resources used. The speed of the fully normalized version of SEC-C is almost equal to that of regular SEC-C for higher numbers of templates, indicating that in such cases, fully normalized SEC-C would be the better choice as it does not significantly increase the run time. (b) Matched filter peak memory usage comparison between SEC-C, EQcorrscan, and FMF for the same test case in (a). Memory overhead is measured by monitoring the memory usage during each run using the *htop* command. For all of the test cases, SEC-C has the lowest memory usage, using approximately 2–3 times less memory than EQcorrscan, and 20%–30% less memory than FMF. The trend of memory usage with increasing numbers of templates suggests that for large numbers of templates (> 70), FMF will be more memory efficient than SEC-C, as expected for a time-domain code. The color version of this figure is available only in the electronic edition.

EQcorrscan that uses the FFTW library (Frigo and Johnson, 2005) for the Fourier transform procedure. We use synthetic data for the comparison test generated by test codes accompanying both software packages. We also use the same synthetic data generated by the FMF test code to test SEC-C. Our tests are run on a desktop machine with an Intel Core i7-4790k CPU processor that includes four cores (eight threads) and 16 GB of memory. This is the intended platform (i.e., a desktop computer) for the current version of the SEC-C method. In contrast, EQcorrscan can take advantage of CPU clusters with large memory capacity, and FMF is designed to take advantage of GPU hardware where available. Therefore, the comparisons stated below do not reflect the capabilities of these methods for their intended cases, rather they show performance of the methods when there are limitations in computation power, memory, or both.

From now on, we demonstrate the matched filter test cases with a vector with six numbers indicating the number of days of seismic data, number of stations, number of components, sample rate in Hz, template length in seconds, and number of templates, respectively. In this case, our test case vector was (1, 10, 3, 50, 8, x). We tried different values of x by varying the number of templates from 1 to 40. We cannot test a greater number of templates as the EQcorrscan code exceeds the available memory on our test machine with more than 40 templates.

To make a clearer comparison between the speed of SEC-C and the speeds of the other codes, we run SEC-C using three different strategies: (1) forcing MATLAB to use only one CPU thread for the computations (hereafter referred to as the single thread case of SEC-C); (2) allowing MATLAB to use multithreading (i.e., the use of multiple CPU threads) for some built-in functions that are optimized for it (the regular case of SEC-C); and (3) running multiple single thread instances of SEC-C independently and simultaneously in parallel (the parallelized case of SEC-C). For strategy (3), if the number of stations is sufficiently small, one day of data can be run per CPU thread; if not, and if memory limitations become an issue, each day of data can be divided by the number of threads into equally sized smaller subsets, with an appropriate overlap that takes moveout into account.

Figure 2 demonstrates the results for our speed test. We find that SEC-C's performance is on average ~2, 4, and 6 times faster than FMF for the three SEC-C strategies explained above, respectively. Because FMF makes use of multithreading to gain computation speed, a second test on a machine with an Intel Core i5 processor (four cores, four threads) was ~8 times slower than the regular case of SEC-C for the same test vector (Ⓔ Fig. S2). Overall, assuming limited computational resources such as a desktop computer, the strength of the SEC-C algorithm with respect to FMF is when the template length is large (e.g., > 5 s), the number of stations and components is also large (e.g., > 30 channels), and when higher sample rates (e.g., > 50 Hz) are needed. If the use case involves templates with short lengths (e.g., a few seconds) and uses data with lower sample rates (e.g., 20 Hz), then FMF becomes more effective with respect to SEC-C. Also, if the data do not involve higher frequency content, the step feature in FMF, which calculates CCCs at regular sample steps, rather than for every sample, can be used to speed up the computation. However, this comes at the expense of potentially degraded matching performance and/or lower peak CCC values, especially when the step size is bigger than the shortest period used. As mentioned above, SEC-C can be effectively parallelized by running multiple instances on different CPU threads for enhanced performance, whereas FMF makes use of all CPU resources for a single run.

Although the memory efficiency tweaks that we have made in SEC-C trade-off with computation speed, our run test mentioned above shows that SEC-C runs approximately twice as fast with respect to EQcorrscan when both are using a single thread (Fig. 2a). Memory issues with EQcorrscan did not allow us to perform our test case using the parallelized version of EQcorrscan; however, our tests with fewer templates (less than 5) show that the run time of the parallelized version of EQcorrscan is slightly longer compared with the parallelized version of SEC-C (i.e., strategy 3 mentioned above). Overall, as the number of processes increases (e.g., increasing sample rate, number of stations, number of templates), the speed of SEC-C with respect to EQcorrscan increases.

Here, we give two more examples that show the relative efficiency of SEC-C with respect to other contemporary codes in terms of speed. Beaucé *et al.* (2018) reported the run time for a matched filter with the test vector (1, 12, 3, 50, 8, 20) while running the test using 24 CPU cores for EQcorrscan and FMF with one sample step (i.e., on all samples). EQcorrscan finished this test in 15.8 s, compared with 55.5 s for FMF. We run the same test using SEC-C on a single CPU thread on the desktop PC mentioned above, completing it in 88 s. This indicates that the single thread case of SEC-C has run times on the order of EQcorrscan and FMF running on multiple threads. The run time of the regular case of SEC-C (i.e., strategy 2 from above) is 49 s for this test. In another study, Mu *et al.* (2017) reported a test case vector of (1, 1, 1, 100, 2.56, 18) that finished in 2.97 s using the most CPU efficient and parallelized version of their matched filter code (the C2 method) running on 18 processor cores. SEC-C can complete this test case, again using one CPU thread on the same desktop machine mentioned above, in 5.45 s. Both FMF and the method of Mu *et al.* (2017) are GPU optimized and their GPU implementations can run much faster than their reported runtimes for CPU clusters. However, the examples and tests above highlight the efficiency of SEC-C when the computational resources are limited (e.g., few CPU cores and no available GPU, such as may be available on a desktop computer or laptop).

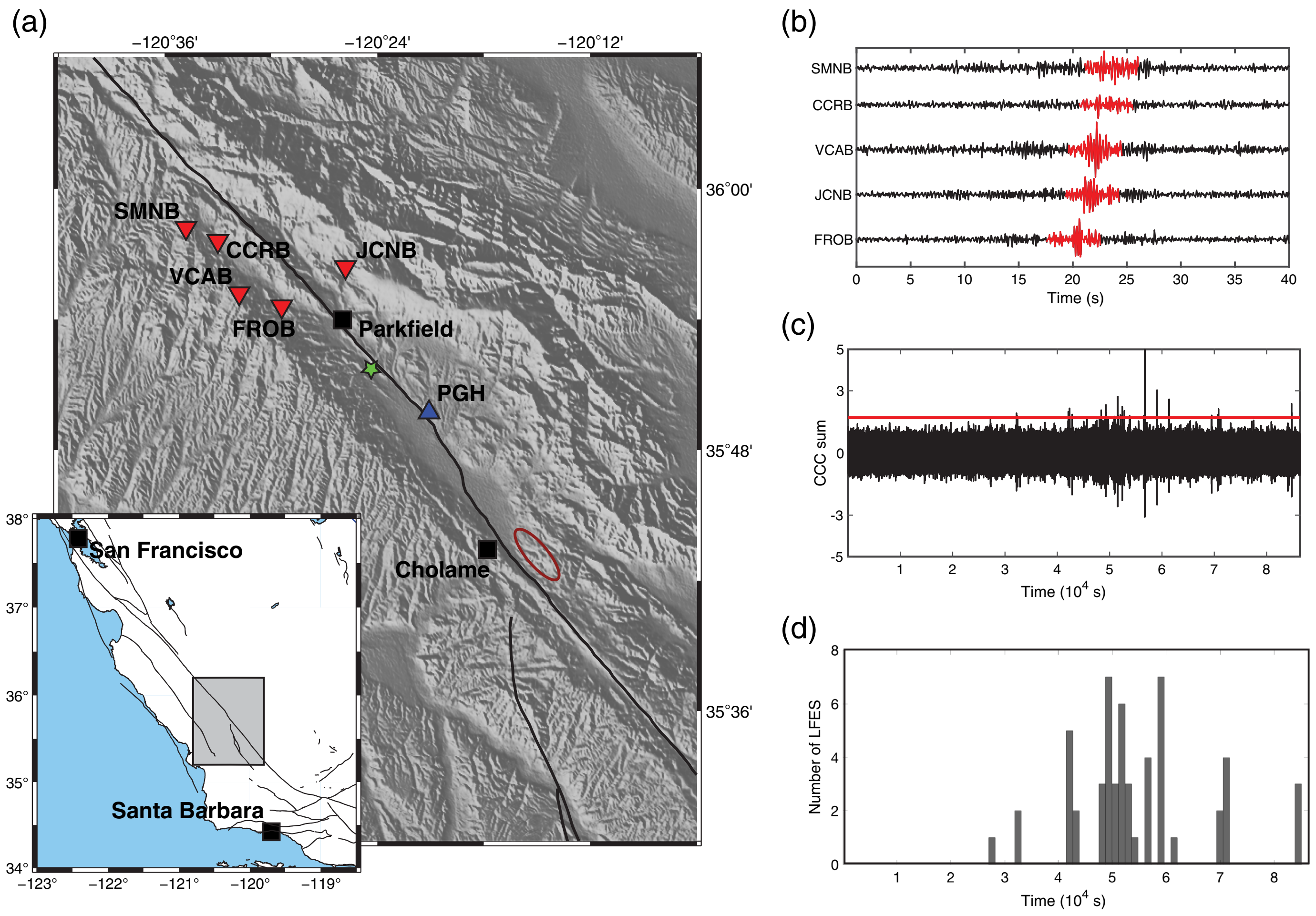
SEC-C Memory Efficiency

The strength of the SEC-C method compared with time-domain CPU codes is the run time speed. However, compared with other frequency domain methods (e.g., EQcorrscan), its strength is its memory efficiency. Here, we test the memory usage of SEC-C with respect to FMF and EQcorrscan while running a test with the case vector of (1, 10, 3, 50, 8, 30). We monitor the memory usage of these three methods using the *htop* (see Data and Resources) command. We compare the peak of memory usage before and during the runs. We find that the peak memory usage of EQcorrscan was 6.9 GB, compared with 2.95 GB for FMF and only 2.31 GB for SEC-C (Fig. 2b). The memory usage corresponding to the input and output data in this test case adds up to ~2 GB. This shows that our memory-based implementation made SEC-C even more efficient than time-domain methods (e.g., in this case FMF uses ~0.64 GB more memory than SEC-C). Peak memory usage estimated in this way for a range of numbers of templates (between 1 and 40) is shown in Figure 2b. For all of the test cases, SEC-C has the lowest memory usage, using approximately 2–3 times less memory than EQcorrscan, and 20–30% less memory than FMF. The trend of memory usage with increasing numbers of templates suggests that for large numbers of templates (> 70), FMF will be more memory efficient than SEC-C, as expected for a time-domain code.

One more example that demonstrates the strength of the SEC-C method with respect to the other methods is when applying a matched filter to a large array of seismic stations, for example, with a test vector of (1, 60, 3, 50, 10, 10). Using a regular desktop or even a small cluster, it is not possible to achieve this efficiently using a time-domain method (e.g., FMF). To avoid memory problems when using a regular frequency domain method (e.g., EQcorrscan), the user must divide the data into smaller subsets with smaller n and loop over those subsets. The additional disk read and write operations when loading data subsets and saving the results could potentially be more time consuming compared with a case where the matched filtering can be completed in one process. SEC-C can complete the example above with the test case mentioned above in approximately two minutes on our test machine. SEC-C can perform matched filtering of up to a test case of (1, 110, 3, 50, 10, 1) without a memory problem and in a similarly efficient time (i.e., less than a minute) using the same desktop machine.

EXAMPLE APPLICATIONS OF THE SEC-C ALGORITHM

SEC-C is a versatile method that can be used for speeding up detection of any similar seismic events, for example, REs, LFEs,



▲ **Figure 3.** (a) A map of the San Andreas fault area near Parkfield, California. Inverted triangles are the locations of Parkfield high-resolution seismic network (HRSN) stations that are used in this study to search for low-frequency earthquakes (LFEs), regular triangle is station PGH from the Northern California Seismic Network (NCSN) used to search for repeating earthquakes (REs), the star is the location of a family of REs (see Fig. 3c), and the ellipse shows the approximate locations of the LFEs detected by Shelly *et al.* (2009). (b) Waveforms from HRSN seismic stations showing our LFE template (indicated by dashed lines). The waveforms are arranged from top to bottom based on their stations' approximate distance to the source (i.e., most to least distant, respectively). (c) Sum of the CCC functions from the five HRSN stations calculated using the template shown in (b) and the SEC-C method, for waveforms from 6 October 2007 (UTC). The horizontal line is the detection threshold we use, eight times the MAD, based on Shelly *et al.* (2007). (d) A histogram of LFEs detected using the SEC-C method and our template. Although we used a different method, our results (i.e., detection times and number of detections) broadly agree with those of Shelly *et al.* (2009; their fig. 2). The color version of this figure is available only in the electronic edition.

triggered earthquakes, swarms of nonvolcanic or volcanic earthquakes, foreshocks, and aftershock sequences. We show two examples of these applications.

1. Detection of LFEs: Here, we present an example to show how this method can help us in the rapid detection of LFEs. We searched for LFEs in waveform data from a tremor burst that occurred on 6 October 2007 on the San Andreas fault near Parkfield, California (Fig. 3a), in which many LFEs were detected by template matching (Shelly *et al.*, 2009). We select an LFE template waveform for each station (Fig. 3b) by stacking matrix profiles (a measure of waveform self-similarity) from 24 hrs of data spanning the tremor burst from three borehole stations of the high-resolution seismic network (HRSN) in the Parkfield area. (For full details of this procedure and of the matrix profile method, see Zhu *et al.*, 2016, 2018.)

We then use the SEC-C method to calculate CCC functions for five HRSN stations and sum these CCC functions, aligning them by accounting for the differential arrival time (i.e., moveouts) for the template at each station (i.e., using the *S*-wave envelope peak). We then use the threshold of eight times of median absolute deviation (MAD, e.g., Shelly *et al.*, 2007, 2009). Figure 3c shows the sum of CCC functions for five stations and the threshold. In general, the temporal pattern of detected event origin times (Fig. 3d) is consistent with the results of Shelly *et al.* (2009; Fig. 2a); any differences in detail can be attributed to the different network configurations used, and the recursive matched filter process used in the earlier study.

In this example, the test case vector is (1, 5, 1, 20, 5, 1) and the regular case of SEC-C can complete it in 0.21 s. For the 50 and 100 Hz cases, run time increases to 0.54 and 1.38 s, respectively. Assuming that this computation time would scale linearly with the number of days of data searched, performing the same procedure for one year of continuous data would take between ~77 and 504 s depending on the sample rate. To test this hypothesis, we use the parallelized case of SEC-C on 365 days of data, with each of eight CPU threads on our test machine running a single thread instance of SEC-C on one day of data at a time, simultaneously. The run time for this case, including loading data, running SEC-C, and saving the output for the 20 and 100 Hz cases took 81 and 423 s, respectively, using our desktop test machine. This shows how this method could greatly expedite searches for repeating seismic events in continuous waveform data if suitable event templates are known.

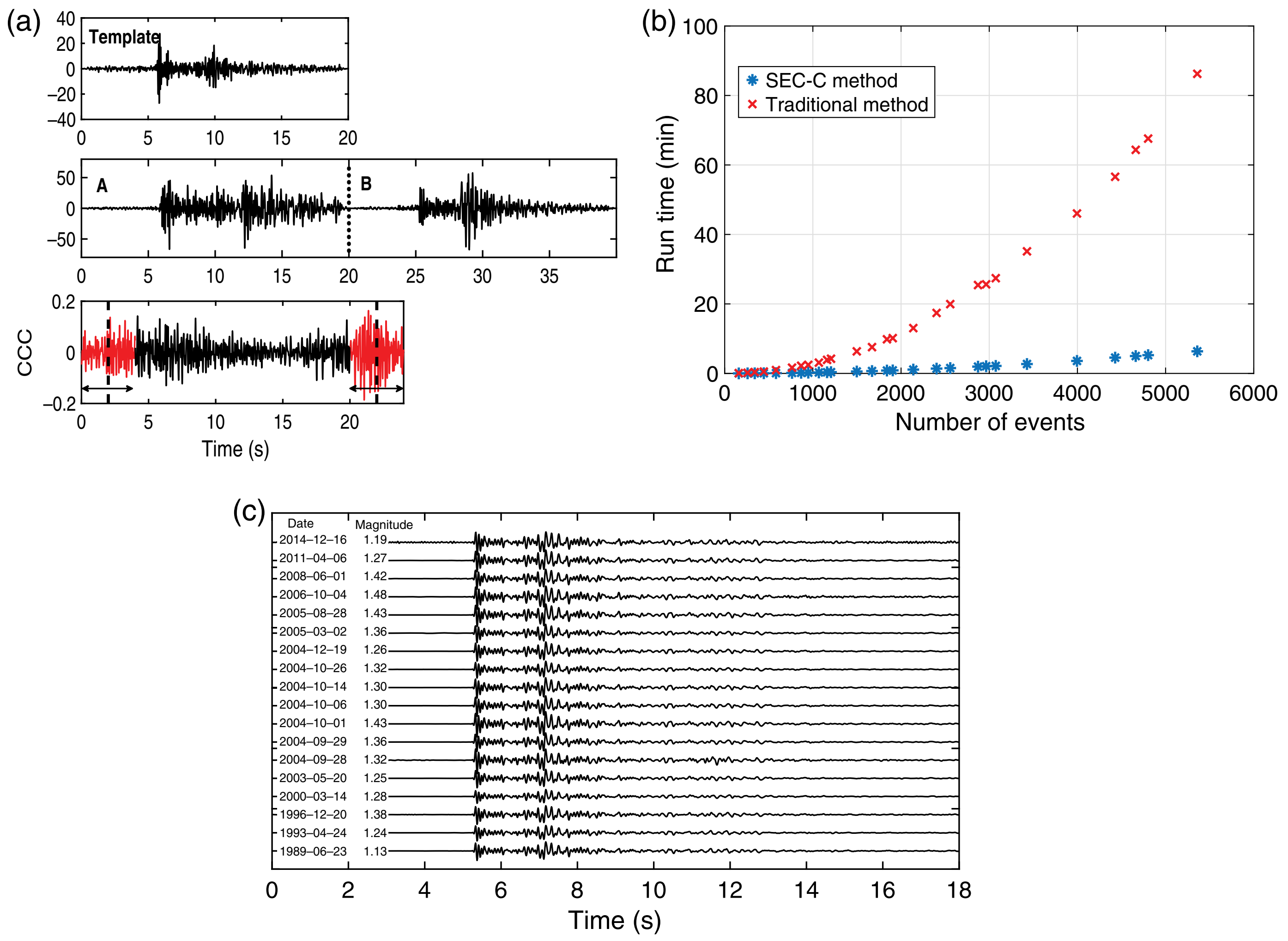
2. Detection of REs from individual detected catalog events: Along with the acceleration of template matching in continuous seismic data, one main strength of the SEC-C method is that it can be applied to template matching among individual waveforms from previously detected events. Here, we show one example: searching for REs in central California near Parkfield. For this purpose, we compared the performance of SEC-C with that of the *xcorr* function, as the latter in this case is an efficient way of calculating CCC functions (i.e., CCC as a function of lag time) for the individual event waveforms. The maximum lag (in terms of number of samples) that the CCC function needs to be calculated over depends on the errors in the seismogram phase information (e.g., *P* arrival pick), which are typically of the order of 1–2 s, multiplied by the sample rate. In this section, we use a brute force traditional method using the MATLAB cross-correlation routine *xcorr*, in which individual waveforms are compared with each other one-by-one, via two nested loops, as a comparison to the SEC-C method.

   Our run time tests show that the SEC-C method can accelerate the search for REs by up to a factor of 15.5 faster than the traditional method, depending on the number of individual candidate events we start with.

   To make use of the SEC-C method to search for REs in a set of individual event waveforms, we must first concatenate these event waveforms together to form one continuous waveform. The SEC-C method can then be used to compute cross correlations between this continuous waveform and a template waveform as described above. Although SEC-C is fast at computing CCCs, as we demonstrate above, many of the CCCs calculated in this case are not necessary for the detection process. The unnecessary CCC calculations result from our concatenated waveform effectively having a large number of artificial waveforms (or waveform chimeras) composed of parts of pairs of neighboring waveforms. For example, if we have two event waveforms, A and B, and concatenate them, the SEC-C method would calculate the CCC between the template and a waveform window containing the second half of event A and the first half of event B (Fig. 4a). The resulting calculated value would be a scientifically meaningless quantity, and in the traditional method we would not compute it. A great many of the CCCs computed using SEC-C in this setup would be of this unnecessary type. Because we would not expend computing resources to compute these meaningless cross correlations under the traditional method, preferring instead to search for a small range of plausible time shifts within the target waveform, the differential in computation time between the two methods is greatly reduced for this application compared with scanning a continuous waveform, but we still obtain faster run times using SEC-C, as we document below.

Our experiments in searching for REs near Parkfield (Figs. 3a and 4a,b) show that the search for REs using the SEC-C algorithm is more than one order of magnitude faster than the traditional method. We use for this demonstration triggered event data from the Northern California Seismic Network station PGH (Fig. 3a) that has historically high signal-to-noise ratios and also a long period of operation (1987–present). We retrieve event waveforms from this station, targeting events whose catalog locations are within a small area in Parkfield where the occurrence of REs is expected (e.g., Lengliné and Marsan, 2009; Nadeau, 2014). In total, we perform 14,399,661 pairwise CCC calculations for 5366 waveforms that are band-pass filtered between 1 and 15 Hz, with 100 Hz sample rate and with 10 s duration. We find 284 candidate RE families, each having more than three events in a family with CCCs greater than or equal to 0.95 between their pairs. The family with the largest number of repeats has 49 events in total. Figure 3b shows an example of a RE family with 18 recurrences since 1987 detected by SEC-C. The first five sequences of this family reoccurred regularly before the 2004 M_w 6.0 Parkfield earthquake with a recurrence interval of 3.5 ± 0.3 yrs. The sequences triggered by the 2004 event had recurrence intervals that were shortened to hours in its immediate aftermath and eventually, following a typical Omori–Utsu law, recovered to their original recurrence intervals from before the 2004 mainshock in a period of ~7 yrs.

The new method improves the computation time for searching for REs from around one-and-a-half hours under the traditional approach to less than seven minutes using SEC-C on our desktop test machine. To compare the run times between the two methods, we run multiple tests on each using a series of differently sized random subsets of these waveforms. On average, we find the SEC-C method is 12.1 times faster than the traditional, looped CCC method (see Fig. 4a). The speed-up factor stays above ~11 for all the subset sizes we test. We have started to apply the SEC-C code to large scale seismic applications, such as mining a large seismic data set (i.e., including 40,000+ events, 300+ stations, 600,000+ event waveforms) to search for REs in northern California (Funning *et al.*, 2017). Although a discussion of the results of that work is beyond the scope of this study, we found that the entire



▲ **Figure 4.** (a) An example shows how we use the SEC-C method to calculate CCCs for individual event waveforms. From top to bottom: a template, concatenated waveforms A and B, and CCC between the template and the concatenated waveform. Portions of the CCC function indicated by double-headed arrows are the scientifically useful calculated CCCs and the remainder the unnecessary CCCs calculated in this process. Dashed lines indicate the CCC when the template is aligned with A and B based on *P*-arrival phase information. The majority of CCC calculations are unnecessary (more than 83%). (b) Computational time comparison between SEC-C method and the traditional method of searching for REs in different data sets containing different numbers of events. Both of these methods show computation time proportional to the square of the number of events, *n* (i.e., $O(n^2)$). This comparison shows that the SEC-C method is 10.8–15.5 times faster for data sets ranging from hundreds to thousands of events and has a mean improvement of 12.1 times faster in general. (c) One example of a RE family detected by the SEC-C method using waveform data from seismic station PGH (see Fig. 3a for locations of the RE family and PGH). The color version of this figure is available only in the electronic edition.

process, including data downloading and preprocessing, computation of the waveform comparisons and clustering of the results, could be completed in one week using the same desktop machine.

CONCLUSIONS

We use a combination of different algorithm improvements such as FFT convolution, the overlap–add method, vectorization, and fast normalization to produce an accurate sliding CCC algorithm with zero-lag that is inexpensive to compute for large seismic data sets. This method, which we call SEC-C, is usable for many time series applications that require efficient computation of cross correlations, including various seismological applications such as detecting REs, foreshocks, aftershocks, LFEs, etc. SEC-C is a seismic cross-correlation package that can leverage a regular desktop machine and make it a powerful tool that can handle demanding matched filter projects. The MATLAB code is available in the Ⓔ electronic supplement and it is also available, along with a Python version, from our GitHub repository. An example of performing template matching that includes retrieving, prepossessing, performing template matching using SEC-C, and postprocessing results is also included in the GitHub repository, for new users with low-computational resources. We test this method on several different seismic data sets at a range of sample rates and compare it with other CPU-based contemporary methods. Our tests reveal that SEC-C is not only accurate to machine precision (i.e., double precision), but also it is the most efficient in terms of speed and memory usage. SEC-C can efficiently calculate the CCC sum and can also save the individual CCCs for each channel without introducing extra cost in terms of speed. Despite calculating many unnecessary CCCs, searching for repeating seismic events in a set of individual event waveforms using the SEC-C method shows a speed improvement of more than one order of magnitude on average for sets of hundreds to thousands of waveforms with respect to regular pairwise CCC calculations. This will reduce the run time required for performing pairwise cross correlation of several thousands of events from hours to minutes using a regular desktop machine

Our development of the SEC-C method is part of an ongoing effort for speeding up seismic cross-correlation analysis. We plan to continue our time and memory optimization for SEC-C in future through, for example, producing versions in lower level programming languages (e.g., C++) and exploring the possibility of parallelization, both for CPUs and GPUs.

DATA AND RESOURCES

We retrieved the seismic data for stations near Mount St. Helens and Parkfield from the Incorporated Research Institutions for Seismology Data Management Center (IRIS-DMC) using the IRISFETCH MATLAB software that can be downloaded from http://ds.iris.edu/ds/nodes/dmc/software/downloads/irisFetch.m (last accessed July 2018). We managed the seismic data (e.g., filtering, merging, visualizing, etc.) using the MATLAB signal processing toolbox and Seismic Analysis Code (SAC, https://ds.iris.edu/files/sac-manual/, last accessed March 2018). Some figures were made using the Generic Mapping Tools (GMT, http://gmt.soest.hawaii.edu/, last accessed March 2018; Wessel *et al.*, 2013; last accessed March 2018). We used EQcorrscan v. 0.2.7 (https://eqcorrscan.readthedocs.io/en/latest/; last accessed July 2018) and fast matched filter (FMF; https://github.com/beridel/fast_matched_filter; last accessed July 2018) for our speed and memory comparison tests. GitHub repository is available at https://github.com/Naderss/SEC_C (last accessed September 2018). htop command is available at https://hisham.hm/htop/ (last accessed October 2018). MATLAB is available at www.mathworks.com/products/matlab (last accessed March 2018).

ACKNOWLEDGMENTS

This study was supported by U.S. Geological Survey (USGS) Award G16AP00034. N. S. S. acknowledges additional support from a National Aeronautics and Space Administration (NASA) Earth and Space Science Fellowship, Award NNX15AM66H. The authors would like to thank Editor-in-Chief Zhigang Peng, William Frank, and one anonymous reviewer for their helpful comments and useful criticisms that have led us to improve both our algorithm and our article.

REFERENCES

Allstadt, K., and S. D. Malone (2014). Swarms of repeating stick-slip icequakes triggered by snow loading at Mount Rainier volcano, *J. Geophys. Res.* **119,** no. 5, 1180–1203.

Beaucé, E., W. B. Frank, and A. Romanenko (2018). Fast matched filter (FMF): An efficient seismic matched-filter search for both CPU and GPU architectures, *Seismol. Res. Lett.* **89,** no. 1, 165–172.

Brown, J. R., G. C. Beroza, and D. R. Shelly (2008). An autocorrelation method to detect low frequency earthquakes within tremor, *Geophys. Res. Lett.* **35,** no. 16, doi: 10.1029/2008GL034560.

Chamberlain, C. J., C. J. Hopp, C. M. Boese, E. Warren-Smith, D. Chambers, S. X. Chu, K. Michailos, and J. Townend (2018). EQcorrscan: Repeating and near-repeating earthquake detection and analysis in python, *Seismol. Res. Lett.* **89,** no. 1, 173–181.

Dagum, L., and R. Menon (1998). OpenMP: An industry-standard API for shared-memory programming, *IEEE Comput. Sci. Eng.* **5,** no. 1, 46–55, doi: 10.1109/99.660313.

Frank, W. B., and N. M. Shapiro (2014). Automatic detection of low-frequency earthquakes (LFEs) based on a beamformed network response, *Geophys. J. Int.* **197,** no. 2, 1215–1223.

Frank, W. B., P. Poli, and H. Perfettini (2017). Mapping the rheology of the Central Chile subduction zone with aftershocks, *Geophys. Res. Lett.* **44,** doi: 10.1002/2016GL072288.

Frigo, M., and S. G. Johnson (2005). The design and implementation of FFTW3, *Proc. IEEE* **93,** no. 2, 216–231.

Funning, G., N. Shakibay Senobari, and J. L. Swiatlowski (2017). Distribution of creep in the northern San Francisco Bay Area illuminated by repeating earthquakes and InSAR, Abstract T21A-0542 presented at *2017 Fall Meeting, AGU,* New Orleans, Louisiana, 11–15 December.

Gibbons, S. J., and F. Ringdal (2006). The detection of low magnitude seismic events using array-based waveform correlation, *Geophys. J. Int.* **165,** no. 1, 149–166.

Hauksson, E., and P. Shearer (2005). Southern California hypocenter relocation with waveform cross-correlation, part 1: Results using the double-difference method, *Bull. Seismol. Soc. Am.* **95,** no. 3, 896–903.

Hutko, A. R., M. Bahavar, C. Trabant, R. T. Weekly, M. Van Fossen, and T. Ahern (2017). Data products at the IRIS-DMC: Growth and usage, *Seismol. Res. Lett.* **88,** no. 3, 892–903.

Iverson, R. M., D. Dzurisin, C. A. Gardner, T. M. Gerlach, R. G. LaHusen, M. Lisowski, S. D. Malone, J. A. Messerich, S. C. Moran, and J. S. Pallister (2006). Dynamics of seismogenic volcanic extrusion at Mount St Helens in 2004-05, *Nature* **444,** no. 7118, 439.

Kato, A., J. I. Fukuda, S. Nakagawa, and K. Obara (2016). Foreshock migration preceding the 2016 M_w 7.0 Kumamoto earthquake, Japan, *Geophys. Res. Lett.* **43,** no. 17, 8945–8953.

Lengliné, O., and D. Marsan (2009). Inferring the coseismic and postseismic stress changes caused by the 2004 $M_w = 6$ Parkfield earthquake from variations of recurrence times of microearthquakes, *J. Geophys. Res.* **114,** no. B10303, doi: 10.1029/2008JB006118.

Lewis, J. P. (1995). Fast Template Matching, in *Vision Interface 95, Canadian Image Processing and Pattern Recognition Society*, Quebec City, Canada, 15–19 May, 120–123.

Meng, X., and Z. Peng (2014). Seismicity rate changes in the Salton Sea geothermal field and the San Jacinto fault zone after the 2010 M_w 7.2 El Mayor-Cucapah earthquake, *Geophys. J. Int.* **197,** no. 3, 1750–1762.

Meng, X., X. Yu, Z. Peng, and B. Hong (2012). Detecting earthquakes around Salton Sea following the 2010 M_w 7. 2 El Mayor-Cucapah earthquake using GPU parallel computing, *Proc. Comput. Sci.* **9,** 937–946.

Mu, D., E. J. Lee, and P. Chen (2017). Rapid earthquake detection through GPU-Based template matching, *Comput. Geosci.* **109,** 305–314.

Mueen, A., K. Viswanathan, C. K. Gupta, and E. Keogh (2015). *The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance*, available at http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html (last accessed March 2018).

Nadeau, R. M. (2014) Parkfield borehole network (HRSN): Activities: (SOH using similar and repeating events, and efforts in support of SAFOD), *Annual Rept. 2013–2014*, Berkeley Seismological Laboratory, Berkeley, California.

Nadeau, R. M., W. Foxall, and T. V. McEvilly (1995). Clustering and periodic recurrence of microearthquakes on the San Andreas fault at Parkfield, California, *Science* **267,** 503–507.

Peng, Z., and P. Zhao (2009). Migration of early aftershocks following the 2004 Parkfield earthquake, *Nature Geosci.* **2,** no. 12, 877.

Poupinet, G., W. L. Ellsworth, and J. Frechet (1984). Monitoring velocity variations in the crust using earthquake doublets: An application to the Calaveras fault, California, *J. Geophys. Res.* **89,** no. B7, 5719–5731.

Rabiner, L. R., and B. Gold (1975). *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 777 pp.

Schaff, D. P., and G. C. Beroza (2004). Coseismic and postseismic velocity changes measured by repeating earthquakes, *J. Geophys. Res.* **109,** no. B10, doi: 10.1029/2004JB003011.

Schaff, D. P., and F. Waldhauser (2005). Waveform cross-correlation-based differential travel-time measurements at the Northern California Seismic Network, *Bull. Seismol. Soc. Am.* **95,** no. 6, 2446–2461.

Schaff, D. P., and F. Waldhauser (2010). One magnitude unit reduction in detection threshold by cross correlation applied to Parkfield (California) and China seismicity, *Bull. Seismol. Soc. Am.* **100,** no. 6, 3224–3238.

Shelly, D. R. (2017). A 15-year catalog of more than 1 million low-frequency earthquakes: Tracking tremor and slip along the deep San Andreas Fault, *J. Geophys. Res.* **122,** no. 5, 3739–3753, doi: 10.1002/2017JB014047.

Shelly, D. R., G. C. Beroza, and S. Ide (2007). Non-volcanic tremor and low-frequency earthquake swarms, *Nature* **446,** no. 7133, 305.

Shelly, D. R., W. L. Ellsworth, T. Ryberg, C. Haberland, G. S. Fuis, J. Murphy, R. M. Nadeau, and R. Bürgmann (2009). Precise location of San Andreas fault tremors near Cholame, California using seismometer clusters: Slip on the deep extension of the fault?, *Geophys. Res. Lett.* **36,** no. 1, doi: 10.1029/2008GL036367.

Skoumal, R. J., M. R. Brudzinski, and B. S. Currie (2015). Distinguishing induced seismicity from natural seismicity in Ohio: Demonstrating the utility of waveform template matching, *J. Geophys. Res.* **120,** no. 9, 6284–6296.

Smith, S. W. (1997). *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Pub., San Diego, California, 626 pp.

Thomas, A. M., R. Bürgmann, D. R. Shelly, N. M. Beeler, and M. L. Rudolph (2012). Tidal triggering of low frequency earthquakes near Parkfield, California: Implications for fault mechanics within the brittle-ductile transition, *J. Geophys. Res.* **117,** no. B5, doi: 10.1029/2011JB009036.

Tkalčić, H., M. Young, T. Bodin, S. Ngo, and M. Sambridge (2013). The shuffling rotation of the Earth's inner core revealed by earthquake doublets, *Nature Geosci.* **6,** no. 6, 497.

Waldhauser, F., and W. L. Ellsworth (2000). A double-difference earthquake location algorithm: Method and application to the northern Hayward fault, California, *Bull. Seismol. Soc. Am.* **90,** no. 6, 1353–1368.

Wessel, P., W. H. Smith, R. Scharroo, J. Luis, and F. Wobbe (2013). Generic mapping tools: Improved version released, *Eos Trans. AGU* **94,** no. 45, 409–410.

Zhang, H., and C. H. Thurber (2003). Double-difference tomography: The method and its application to the Hayward fault, California, *Bull. Seismol. Soc. Am.* **93,** no. 5, 1875–1889.

Zhu, Y., Z. Zimmerman, N. S. Senobari, C. C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh (2016). Matrix profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins, *Data Mining (ICDM), 2016 IEEE 16th International Conference*, 739–748.

Zhu, Y., Z. Zimmerman, N. S. Senobari, C. C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh (2018). Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins, *Knowl. Inf. Sys.* **54,** 1–34, doi: 10.1007/s10115-017-1138-x.

Nader Shakibay Senobari
Gareth J. Funning
Eamonn Keogh
Yan Zhu
Chin-Chia Michael Yeh
Zachary Zimmerman
University of California, Riverside
900 University Avenue
Riverside, California 92521 U.S.A.
nshak006@ucr.edu

Abdullah Mueen
University of New Mexico
Albuquerque, New Mexico 87131 U.S.A.

Published Online 7 November 2018