ELECTRONIC SEISMOLOGIST

A Graphics Processing Unit Implementation for Time–Frequency Phase-Weighted Stacking

by Xiangfang Zeng and Clifford H. Thurber

ABSTRACT

Stacking is an efficient approach to increase signal-to-noise ratio, which is a key issue in seismic data processing. The time–frequency domain- phase weighted stack (tf-PWS) that uses coherency of instantaneous phase as a weighting function can significantly improve the stacked signal quality of many datasets. A graphics processing unit implementation was developed to reduce the heavy computational cost of tf-PWS. Synthetic tests suggest the speed-up factor is up to 20. Our real-data test shows that the convergence of noise cross-correlation functions can be substantially improved by tf-PWS without a computational cost increase.

INTRODUCTION

Signal quality is a key issue in seismic data, especially regarding the detection of weak signals. Stacking is one of the most powerful tools for improving signal-to-noise ratio. The uncorrelated, zero-mean noise component that varies among traces will be reduced after stacking. For example, signal-generated noise is difficult to suppress by ordinary (linear) stacking. Several nonlinear stacking methods have been proposed and applied to the detection of weak signals, including the *N*th-root stack (Richards and Wicks, 1990) and dual bootstrap stack (DBS; Korenaga, 2013). Schimmel and Paulssen (1997) proposed using the coherency of the instantaneous phase, which is less dependent on amplitude, as a weighting function to improve the stacked signal quality. They termed their method as "phase-weighted stack" (PWS). DBS works well in a low signal-to-noise situation, which is very useful to detect the arrival of a weak signal, but it may slightly distort the waveform. Performances of the *N*th-root stack and PWS are very similar in low signal-to-noise situations, but PWS introduces smaller waveform distortion. With the *S* transform (Stockwell *et al.*, 1996), PWS can be extended to the time–frequency domain (tf-PWS), allowing for frequency-dependent resolution of the signal (Schimmel and Gallart, 2007). The tf-PWS method has been applied to the determination of noise cross-correlation functions and mantle body-wave phases, and its effectiveness has been confirmed with regional and global data (Baig *et al.*, 2009; Schimmel *et al.*, 2011). Its application to stacking low-frequency earthquake data also shows substantial improvement (Thurber *et al.*, 2014).

Although the advantages of tf-PWS have been demonstrated in previous studies, its application is somewhat limited by the high computational cost. Stockwell (2007) introduced a new discrete orthonormal *S* transform (DOST) that has significantly reduced the computational cost for the *S* transform, but DOST uses a boxcar function in the frequency domain that causes some negative effects, such as Gibb's ringing. In this article, we present a graphics processing unit (GPU) implementation to accelerate tf-PWS computations. An alternative, efficient CPU code based on the fastest Fourier transform in the west (FFTW) library (see Data and Resources; Frigo and Johnson, 2005) is also described.

GPU IMPLEMENTATION

The tf-PWS method utilizes coherence of the instantaneous phase to down weight incoherent noise. The instantaneous phase $\theta(t)$ can be obtained from the complex trace $S(t)$, which consists of the input signal $s(t)$ and its Hilbert transform $H[s(t)]$ (Schimmel and Paulsen, 1997):

$$S(t) = s(t) + iH[s(t)] = A(t)\mathrm{e}^{i\theta(t)}. \tag{1}$$

In the time–frequency domain, Schimmel and Gallart (2007) show that the *S*-transform decomposition of the signal is analytic, so the instantaneous phase $\theta(\tau,f)$ at time τ and frequency f can be defined as

$$\theta(\tau,f) = \tilde{S}(\tau,f)\mathrm{e}^{i2\pi f\tau}/|\tilde{S}(\tau,f)|, \tag{2}$$

in which $\tilde{S}(\tau,f)$ is the S transform of $s(t)$:

$$\tilde{S}(\tau,f) = \int_{-\infty}^{+\infty} s(t)w(\tau - t,f)\mathrm{e}^{-i2\pi f t}\mathrm{d}t. \tag{3}$$

The *S* transform can be represented as the inverse Fourier transform of a signal that has been weighted by a Gaussian window function $w(\tau - t,f)$:

$$w(\tau - t,f) = \exp\left[\frac{-f^2(\tau - t)^2}{2k^2}\right]|f|/k\sqrt{2\pi}. \tag{4}$$

It can also be represented as a convolution of the signal's spectrum with a phase shift and the Gaussian window

doi: 10.1785/0220150192

function. The Gaussian window function is controlled by three parameters: its center point is at time τ and frequency f controls the width with a resolution parameter k (equation 4).

For an n-point trace, the S transform requires $n \times n$-point Fourier transforms, the most time-consuming part of the computations. The coherency weight of an M-trace stack is defined by a sum of instantaneous phases, given by

$$c_{ps}(\tau,f) = \left|\frac{1}{M}\sum_{j=1}^{M} \tilde{S}_j(\tau,f)\mathrm{e}^{i2\pi f\tau}/|\tilde{S}_j(\tau,f)|\right|^{\gamma}. \qquad (5)$$

Consequently, the tf-PWS stack is obtained by the inverse S transform of the weighted S transform of the linear stack:

$$S_{\mathrm{tf-pws}}(t) = \mathrm{inv}\ [c_{ps}(\tau,f)\tilde{S}_{\mathrm{linear}}(\tau,f)]. \qquad (6)$$

To stack M n-point traces, tf-PWS has to calculate $M \times n$-point Fourier transforms to obtain spectra of all traces then calculate $M \times n$ times to finish the S transforms. The linear stack also requires another $(M+1) \times n$-point Fourier transforms. Finally one n-point Fourier transform is used to finish the inverse transform. There are $[(M+1)n + M + 2]\times$ n-point Fourier transforms in total. Such a heavy computation is suitable for the use of a GPU for acceleration.

A GPU consists of hundreds or thousands of cores, whereas a typical workstation may have (at most) tens of cores. The theoretical floating-point operation per second for a GPU is about an order of magnitude higher than for a typical CPU. The compute unified device architecture (CUDA) platform provides one means for running parallel computations on a personal computer. Recently, the GPU acceleration with CUDA has been applied in wavefield simulation (Komatitsch *et al.*, 2010), earthquake detection (Meng *et al.*, 2012), and travel-time tomography (Liao *et al.*, 2014). CUDA supports running independent jobs on multiple threads. The threads are organized into a "block-and-grid" structure. For example, the GeForce GT 650M graphic card with compute capability 3.0 (see Data and Resources) provides 1024 threads per block, and it can simultaneously run 4096 threads. In previous studies, the GPU implementation gained a speed-up factor on the order of 10 or more. The GPU has limitations, however, because of its unusual programming model (Owens *et al.*, 2007). Therefore, Nvidia provides GPU-accelerated libraries, such as *cuBLAS* and *cuFFT*.

The tf-PWS employs many Fourier transforms to calculate instantaneous phase. The cuFFT provides an FFTW interface and as high as hundreds of giga floating-point operations per second capability. We use cuFFT to compute all Fourier transforms, and the other operations on matrices are run on several CUDA kernel functions.

The tf-PWS procedure can be divided into three parts. The first part computes the Fourier transforms of all traces and obtains the spectrum of every trace. To improve the calculation efficiency, zeros are padded to the raw data to extend the traces to a power of two in length. Memory is allocated with *fftw_malloc* to align memory in our FFTW-based code. The data transfer between host (CPU) and device (GPU) is not only limited by bus width but also by memory allocation. The memory allocated by the host is pageable by default, which means it can be swapped out from memory. The device has to copy host data to a temporary page-locked host array, and then transfer it to device memory. Instead, we allocate page-locked memory using *cudaMallocHost* to avoid the cost of transfer within the host. The data transfer speed of page-locked memory is up to three times faster than pageable memory (Harris, 2012).

After the spectrum of each trace $[\tilde{s}(\omega)]$ is obtained, $\tilde{S}(\tau,f)$ can be obtained by the inverse Fourier transform of $\tilde{s}(f)$ with the phase shift and Gaussian windowing function:

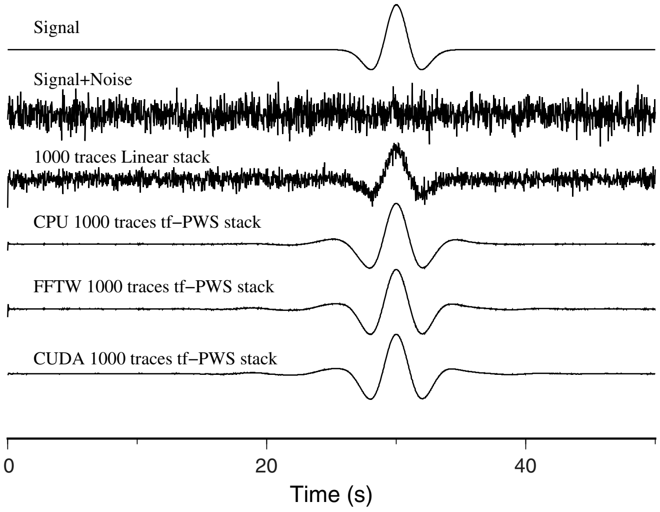
$$s(\tau,f) = \int_{-\infty}^{+\infty} \tilde{s}(f+a)\mathrm{e}^{-\pi a^2/f^2}\mathrm{e}^{i2\pi a\tau}\mathrm{d}a. \qquad (7)$$

A matrix is set up to represent the Gaussian window function. The spectra with phase shifts are also stored in a matrix. A device kernel function finishes the $n \times n$ matrix multiplication. The *cuFFT* routine is utilized to compute the n inverse Fourier transforms. Because larger-size batched Fourier transforms may require impractical memory size in the device, it also can be divided into several batches. Then the coherency weight (equation 5) is updated using a device function. Thus, the second part is done completely within the device, and there is no data transfer between host and device.

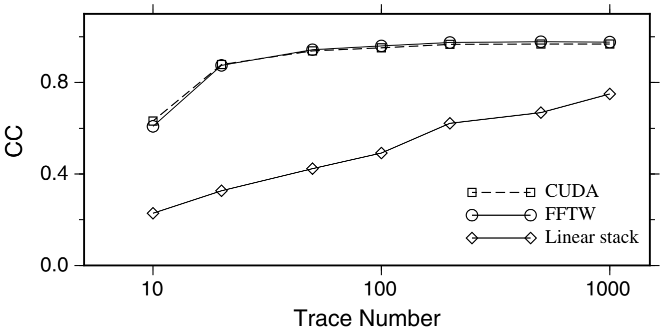
Once the coherency weight has been obtained, the last part needed is to finish the S transform of the linear stack with the same scheme as the second part. The inverse S transform is the inverse Fourier transform of the summation over all frequencies of the weighted $\tilde{S}_{\mathrm{linear}}(\tau,f)$.

VALIDATION AND PERFORMANCE

A synthetic low signal-to-noise ratio data example is used to validate our codes. The signal is a Ricker wavelet with a center frequency of 0.2 Hz and a peak value of 1 (Fig. 1). Gaussian random noise with a standard deviation of 2 is added to the signal. One thousand synthetic traces are assembled, with each trace consisting of 2001 points. The test environment is a 2.7 GHz Intel Core i7 CPU with 8GB 1600 MHz DDR3 memory. The graphic card is Nvidia GeForce GT650M with 1024 MB memory. As Figure 1 shows, the tf-PWS stack nearly recovers the original signal, whereas the signal quality of the linear stack is significantly poorer. A straightforward, naïve code based on the fast Fourier transform is employed as a reference (CPU in Fig. 1). The FFTW-based code and the CUDA-based code provide almost exactly the same results (Fig. 1). We also present convergence curves with different numbers of traces. The quality of the stack is measured with the cross-correlation coefficient between stacks and the original signal. We find that tf-PWS rapidly converges with only about 100 traces, whereas the linear stack needs many more traces to reach stable quality (Fig. 2). The computation time of the FFTW-based code substantially increases with a larger amount data, whereas



▲ **Figure 1.** The signal, one trace of test data, and stacked traces with the different codes used in the validation test.

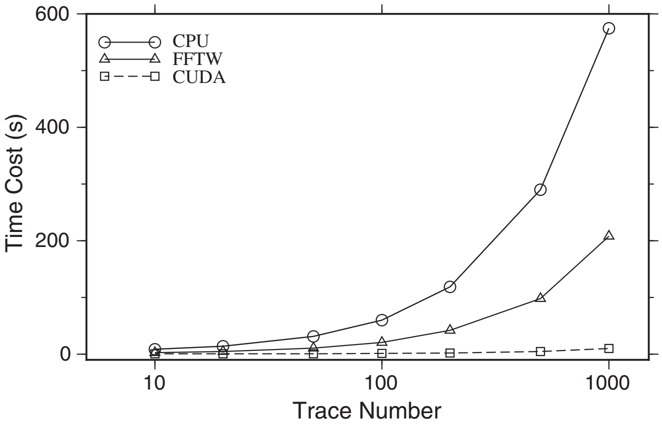


▲ **Figure 2.** Cross-correlation coefficients (CC) between original signal and stack versus number of traces used in stacking for the different codes.

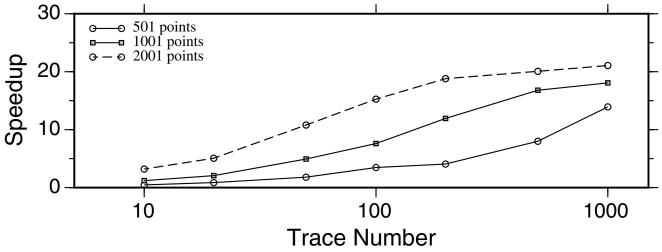
the CUDA-based code shows only a linear increase with data amount. The speed-up factor reaches about 20 with 1000 traces (Fig. 3).

The length of trace also affects the speed-up factor. A longer trace requires a larger-size Fourier transform in the spectrum calculation and *S* transform parts. The time cost, including data transfer, is proportional to the data size, but the percentage in total time cost will decrease with larger data size. We tested three datasets, with the number of points of each trace ranging from 501 to 2001. The time cost of the FFTW-based code is employed as a reference. The speed-up factor increases with the number of traces, and the longer trace case increases faster than the shorter one (Fig. 4). The speed-up factor difference between longer and shorter traces trends toward equality with an increasing number of traces.

Finally, a real ambient-noise cross-correlation function dataset is used to show the potential of our code. We apply the code to two stations in California, SMM and PKD, which are equipped with broadband seismometers. Following Bensen *et al.* (2007), the raw vertical component data are divided into 1-hr-long records. The teleseismic and local earthquake signals are reduced by two band-pass filters, and the spectra are whitened between 0.1 and 2 Hz. About three months of data are used. As Figure 5 shows, the Rayleigh wave signal emerges between −70 and −20 s. Figure 6 shows the *S* transform of one noise cross-correlation function (NCF) and two stacked traces. Both linear and tf-PWS stacks have significantly improved signal quality. The tf-PWS reaches similar quality with only about 50 traces, whereas the linear stack requires 1000 traces. This means it is possible to extract a reliable empirical Green's function with limited duration data and potentially better monitor short-term temporal change in the medium.



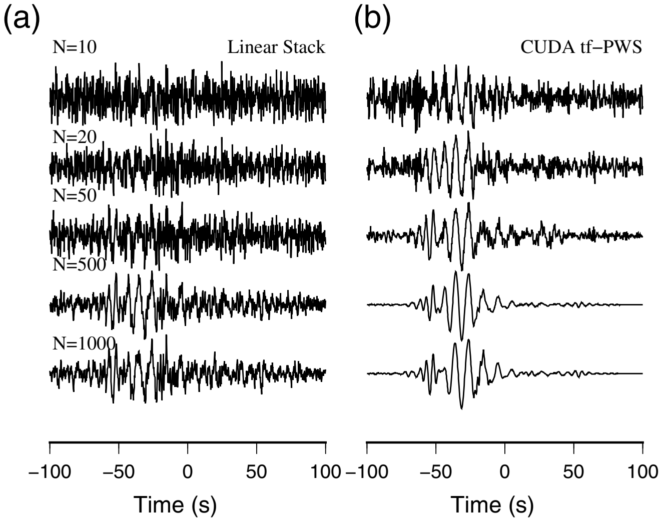
▲ **Figure 3.** Computation time cost of the different time–frequency domain-phase weighted stack (tf-PWS) codes.



▲ **Figure 4.** Speed-up factor versus number of traces using in the stack.

SUMMARY

The tf-PWS method was implemented using the FFTW library and the CUDA platform. Both implementations show significant improvement in computational efficiency. The GPU-based code gains a speed-up factor of up to about 20 in our synthetic data test. Our real-data test shows that the convergence of noise cross-correlation functions can be substantially improved, and/or the length of data required can be substantially reduced, with a modest additional computational cost.

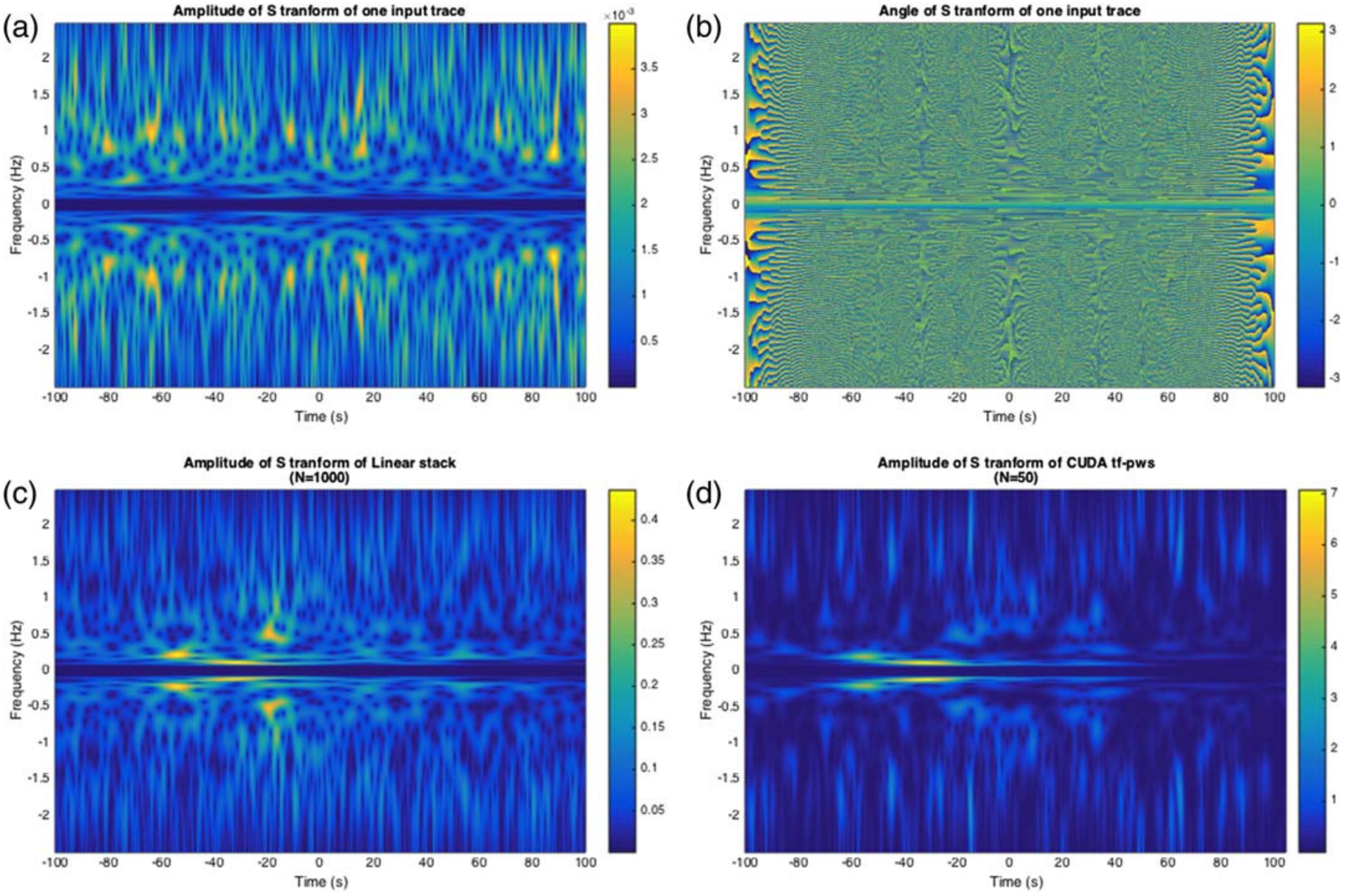


▲ **Figure 5.** Stacked traces using different amounts of data comparing (a) linear stack and (b) graphics processing unit (GPU) tf-PWS.

For the code and examples for this article, see Data and Resources.

DATA AND RESOURCES

Seismograms used in this study were accessed through the Northern California Earthquake Data Center (NCEDC), doi: 10.7932/NCEDC. Some plots were made using Generic Mapping Tools v.4.5.9 (www.soest.hawaii.edu/gmt; Wessel and Smith, 1991). The fastest Fourier transform in the west (FFTW) library is available from http://www.fftw3.org, and the GeForce GT 650M graphic card with compute capability 3.0 is available at https://developer.nvidia.com/cuda-gpus. The code and examples in this article are available at https://github.com/uwzxf/gpu_tf_pws. The CUDA C programming guide is available at http://docs.nvidia.com/cuda/cuda-c-programming-guide. All the above websites are last accessed on May 2015.



▲ **Figure 6.** (a) Amplitude of the *S* transform of one noise cross-correlation function. (b) Phase angle of the *S* transform of one noise cross-correlation function. (c) Amplitude of *S* transform of linear stacked traces ($N = 1000$). (d) Amplitude of the *S* transform of tf-PWS stacked traces ($N = 50$). The color version of this figure is available only in the electronic edition.

ACKNOWLEDGMENTS

Research was supported in part by the U.S. Geological Survey (USGS), Department of the Interior, under USGS Award Number G14AP00056 to the University of Wisconsin–Madison.This research was also supported in part by National Science Foundation Grant Numbers EAR-1114228, EAR-1246975, and EAR-1411779. X. Z. was also supported in part by National Basic Research Program of China (973 Program), Number 2014CB845900. We appreciate the constructive reviews from Adam Baig, an anonymous reviewer, and Editor-in-Chief Zhigang Peng. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government.

REFERENCES

Baig, A. M., M. Campillo, and F. Brenguier (2009). Denoising seismic noise cross correlations, *J. Geophys. Res.* **114,** no. B08310, doi: 10.1029/2008JB006085.

Bensen, G. D., M. H. Ritzwoller, M. P. Barmin, A. L. Levshin, F.-C. Lin, M. P. Moschetti, N. M. Shapiro, and Y. Yang (2007). Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements, *Geophys. J. Int.* **169,** no. 3, 1239–1260.

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

Harris, M. (2012). How to optimize data transfers in CUDA C/C++, http://deveblogs.nvidia.com/parallelforall/how-optimize-data-transfer-cuda-cc (last accessed May 2015).

Komatitsch, D., G. Erlebacher, D. Göddeke, and D. Michéa (2010). High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster, *J. Comput. Phys.* **229,** no. 20, 7692–7714.

Korenaga, J. (2013). Stacking with dual bootstrap resampling, *Geophys. J. Int.* **195,** no. 3, 2023–2036, doi: 10.1093/gji/ggt373.

Liao, P.-C., C.-C. Lii, Y.-C. Lai, P.-Y. Chang, H. Zhang, and C. Thurber (2014). A graphics processing unit implementation and optimization for parallel double-difference seismic tomography, *Bull. Seismol. Soc. Am.* **104,** no. 2, 953–961, doi: 10.1785/0120130236.

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, *Procedia Comput. Sci.* **9,** 937–946.

Owens, J. D., D. P. Luebke, N. K. Govindaraju, M. J. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell (2007). A survey of general-purpose computation on graphics hardware, *Comput. Graph Forum* **26,** no. 1, 80–113, doi: 10.1111/j.1467-8659.2007.01012.x.

Richards, M. A., and C. W. Wicks (1990). S–P conversions from the transition zone beneath Tonga and the nature of the 670 km discontinuity, *Geophys. J. Int.* **101,** 1–35.

Schimmel, M., and J. Gallart (2007). Frequency-dependent phase coherence for noise suppression in seismic array data, *J. Geophys. Res.* **112,** no. B04303, doi: 10.1029/2006JB004680.

Schimmel, M., and H. Paulssen (1997). Noise reduction and detection of weak, coherent signals through phase-weighted stacks, *Geophys. J. Int.* **130,** 497–505, doi: 10.1111/j.1365-246X.1997.tb05664.x.

Schimmel, M., E. Stutzmann, and J. Gallart (2011). Using instantaneous phase coherence for signal extraction from ambient noise data at a local to a global scale, *Geophys. J. Int.* **184,** 494–506, doi: 10.1111/j.1365-246X.2010.04861.x.

Stockwell, R. G. (2007). A basis for efficient representation of the S-transform, *Digit. Signal Process.* **17,** 371–393.

Stockwell, R. G., L. Mansinha, and R. P. Lowe (1996). Localization of the complex spectrum: The S transform, *IEEE Trans. Signal Process.* **44,** no. 4, 998–1001.

Thurber, C., X. Zeng, A. Thomas, and P. Audet (2014). Phased-weighted stacking applied to low-frequency earthquakes, *Bull. Seismol. Soc. Am.* **104,** no. 5, 2567, doi: 10.1785/0120140077.

Wessel, P., and W. H. Smith (1991). Free software helps map and display data, *Eos Trans. AGU* **72,** no. 41, 441–446.

Xiangfang Zeng
Clifford H. Thurber
Department of Geoscience
University of Wisconsin–Madison
1215 West Dayton Street
Madison, Wisconsin 53706 U.S.A.
zengxf@geology.wisc.edu
clifft@geology.wisc.edu

Published Online 13 January 2016