ELECTRONIC SEISMOLOGIST

Seismic Noise Correlation on Heterogeneous Supercomputers

by Andreas Fichtner, Laura Ermert, and Alexey Gokhberg

ABSTRACT

We present a high-performance tool for the computation of ambient seismic noise correlations on central processing unit (CPU) and graphic processing unit (GPU) clusters. This is intended to address emerging challenges in noise correlation studies with increasingly large data volumes. We propose a parallelization scheme and strategies to efficiently harness modern supercomputing resources, and we demonstrate that the use of GPUs can accelerate the computation of noise correlations by one order of magnitude or more compared with a homogeneous implementation on CPUs. In addition to reducing wall-clock time, our tool enables on-the-fly computations of large noise correlation datasets, thereby eliminating the need for mass storage to archive results.

INTRODUCTION

The ambient seismic field contains valuable information on Earth structure and noise generation mechanisms that can be extracted from interstation cross correlations (e.g., Sabra *et al.*, 2005; Shapiro *et al.*, 2005; Stehly *et al.*, 2006, 2009; Yang and Ritzwoller, 2008; Saygin and Kennett, 2012; Ye and Ritzwoller, 2015; Bowden *et al.*, 2016; Ermert *et al.*, 2016; Fichtner *et al.*, 2017). To obtain useful noise correlations, various processing steps are needed to accelerate convergence and suppress earthquakes and other impulsive signals. Typical processing includes the averaging of causal and acausal correlation branches, spectral whitening, time-domain running averages, frequency-domain normalization (e.g., Bensen *et al.*, 2007; Groos *et al.*, 2012), 1-bit normalization (e.g., Cupillard *et al.*, 2011; Hanasoge and Branicki, 2013), phase-weighted stacking (Schimmel and Paulssen, 1997), directional balancing (Curtis and Halliday, 2010), and various selection and suppression filters (Nakata *et al.*, 2015). The wealth of suggested processing schemes already indicates that general rules for noise processing do not exist. Each dataset has its own characteristics and requires significant trial and error to determine a suitable sequence of processing steps.

The volume of seismic data to which noise correlation can be applied is large, and it grows exponentially. For instance, the data archive of Incorporated Research Institutions for Seismology currently holds several hundred terabytes of waveform data, gaining around 25% per year (see Data and Resources). Rapidly growing data volumes combined with increasingly demanding processing techniques are challenging developments for ambient noise correlation studies, where computational intensity scales to the square of the available waveform data. The computation of a large noise correlation dataset and the trial-and-error experimentation needed to find a good processing setup become increasingly difficult.

Emerging applications in time-dependent seismology that monitor reservoirs, fault zones or volcanoes further increase the computational requirements (e.g., Brenguier *et al.*, 2008; Obermann *et al.*, 2013, 2015; de Ridder *et al.*, 2014; Mordret *et al.*, 2014; Delaney *et al.*, 2017). Large noise correlation datasets will need to be computed frequently—for example, every few hours or days—to achieve the desired temporal resolution.

To address these challenges, we developed the ambient noise correlation toolbox *Mirmex*, which implements common processing methods and operates on large central processing unit (CPU) and graphic processing unit (GPU) clusters. *Mirmex* complements noise correlation tools developed earlier using the high-level programming language Python; for example, *Whisper* (Briand *et al.*, 2013) and *MSNoise* (Lecocq *et al.*, 2014). Although Python eases development and is therefore suitable for working on new seismological concepts, a compiled language is clearly more suitable to harness the computational resources of current heterogeneous supercomputers.

The description and illustration of *Mirmex* is the objective of this article, which is organized as follows: we start with a review of software engineering difficulties on heterogeneous supercomputers, and of the algorithms and workflows typically needed in noise correlation studies. This will be followed by a detailed description of the homogeneous (CPU only) and heterogeneous (CPU + GPU) implementations of *Mirmex*, including the parallelization scheme. Finally, we illustrate the capabilities of *Mirmex* using an example large waveform dataset that was recorded globally at 188 stations during one year with a sampling rate of 1 Hz.

doi: 10.1785/0220170043

PROGRAMMING REQUIREMENTS AND CONCEPTS

Heterogeneous supercomputing systems introduced in recent years provide numerous computing nodes interconnected via high throughput networks. Every node contains processing elements of different architectures. Typically, several sequential processor cores are combined with one or a few GPUs serving as accelerators to form one node. Heterogeneous supercomputers provide the opportunity for manifold application performance enhancement and are more energy-efficient. However, their hardware is considerably more complex than that of homogeneous systems and an appropriate engineering approach is crucial to efficiently utilize the potential of this hardware. (Our definition of heterogeneous excludes Beowulf systems of different CPU processor type and architecture.)

The noise correlation application uses a wide range of common signal processing methods. These include several infinite impulse response filter designs, amplitude and instantaneous phase correlation, computing the analytic signal, and discrete Fourier transform. Furthermore, various processing methods specific to seismology, such as rotation of seismic traces, are used. Efficient implementation of all these methods on the heterogeneous massively parallel systems represents several challenges. In particular, it requires a careful distribution of work between the sequential processors and accelerators. Furthermore, because the application is designed to process very large volumes of data, special attention has to be paid to the efficient use of the available memory and networking hardware resources to reduce intensity of data input and output.

We address these challenges by the detailed study and engineering of parallelization schemes at two principal levels: (1) coarse-grained, aimed at work distribution between the parallel computing nodes and (2) fine-grained, exploiting the inherent data-level parallelism of signal processing algorithms on the GPU accelerators.

ALGORITHMS AND WORKFLOW

Mirmex is designed to perform a series of tasks that translate raw seismic waveform data into interstation cross correlations that may then be used for further analyses. The processing workflow includes two principal stages: preprocessing and correlation.

In the preprocessing stage, *Mirmex* converts raw observed seismograms into a format suitable for cross correlation. Preprocessing includes operations such as band-pass filtering, downsampling, and the removal of linear trends and the instrument response. Processing schemes that require communication between different components or stations are technically possible, but currently not implemented. The preprocesses data are then stored on the hard disk, which usually does not constitute a limitation because the downsampling operation considerably reduces the data volume in most applications.

The correlation stage starts with building a set of correlation time windows for each preprocessed trace. Traces are split into windows of user-defined length and the data in each individual window undergo further processing. At this stage, various noise processing schemes can be applied to suppress large-amplitude earthquake signals in an automatic fashion. These include 1-bit normalization, clipping above a given amplitude threshold, and various others.

Finally, *Mirmex* determines possible correlation pairs and computes classical correlations or phase correlations (Schimmel *et al.*, 2011) for each pair. Correlations are computed separately for individual time windows; results are then averaged using either linear stacks or phase-weighted stacks (Schimmel and Paulssen, 1997). The length of the stacks can be adjusted by the user.

These tasks have been implemented for homogeneous and for heterogeneous systems, as described in the following sections.

HOMOGENEOUS IMPLEMENTATION

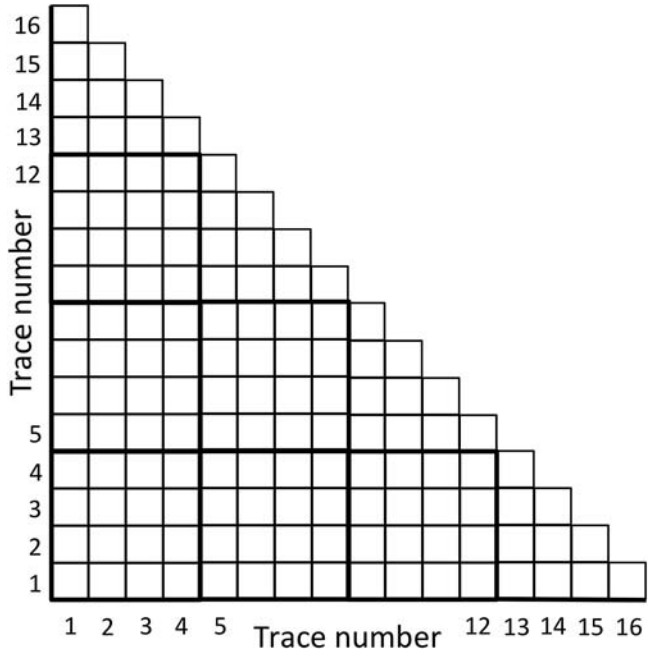
We started with the purely homogeneous implementation of the algorithm intended to run on a CPU cluster. A reference prototype was coded in Python, primarily for use in ambient noise source imaging (Ermert *et al.*, 2016). This is based on open-source packages including *ObsPy* (Beyreuther *et al.*, 2010; Megies *et al.*, 2011; Krischer *et al.*, 2015). In the next step, the code has been fully re-engineered as a C++ library. For the assessment of new ideas and data processing schemes, the Python research code continues to serve as a valuable testing platform. Based on this, modifications can then be made in the C++ production code.

The parallelization of the correlation procedure at coarse-grained level aims at the balanced distribution of work among the available computing nodes. For this purpose, we partition the entire set of seismic traces into a group of square-shaped tiles, as illustrated in Figure 1. These tiles represent units of work, which are scheduled between the computing nodes. Scheduling is static; that is, each node determines which tiles belong to it and then processes them one by one. Seismic traces are loaded once per tile and remain resident in memory during the tile processing.

Testing of the homogeneous implementation revealed that the vast majority of computing time is consumed by just two signal processing methods: correlation itself and the Hilbert transform needed to compute phase correlations and phase-weighted stacks (Schimmel and Paulssen, 1997; Schimmel *et al.*, 2011). In contrast, reading the seismic data takes a modest amount of time and does not require any further optimization.

HETEROGENEOUS IMPLEMENTATION

The homogeneous implementation enabled us to identify the Hilbert transform and the cross correlation itself as obvious candidates for implementation on GPU. The Hilbert transform is implemented as a convolution in the frequency domain, and its computation is dominated by the forward and inverse fast Fourier transform (FFT). We implemented it on GPU in a simple and straightforward way using the NVIDIA



▲ **Figure 1.** Schematic illustration of the correlation parallelization for 16 stations with one trace each. Trace pairs are grouped into tiles of either 16 or 10 pairs—marked by thicker lines—leading to 10 tiles that are distributed onto 10 compute cores.

cuFFT library. The implementation of correlations was more complex. The generic N-point discrete, classical correlation for two evenly sampled traces u_1 and u_2,

$$c_{12}(i) = \sum_{k=0}^{N-i} u_1(k)u_2(k+i) \qquad (1)$$

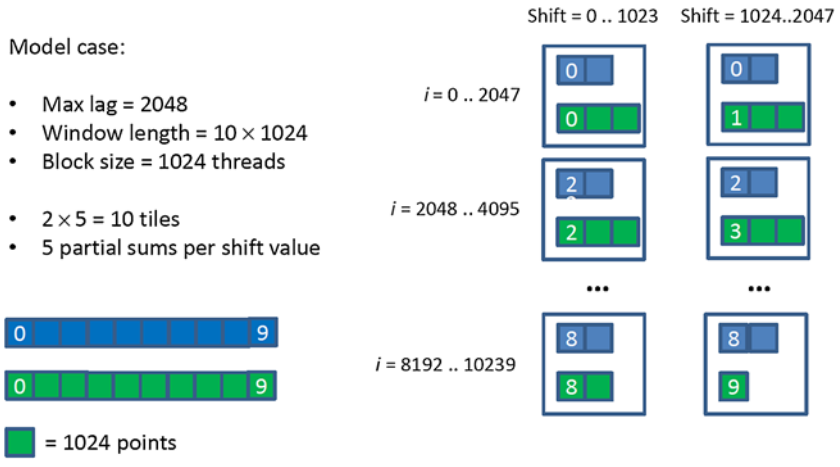
reveals a great degree of data-level parallelism that can be exploited with GPU. GPU accelerators are specialized in parallel processing of large numbers of independent threads, executing a similar computation pattern. The sum in equation (1) represents a very suitable candidate for a single thread of work. With this approach, each thread will compute a correlation value for a single shift i. However, it is easy to see that in the course of computations the same data elements are repeatedly accessed. These memory accesses represent a bottleneck that can substantially decrease the GPU performance.

We address this problem using the GPU-shared memory. This is a special kind of very fast memory, which has, however, a rather limited size. We partition the entire data domain into a set of tiles in such a way that each tile can stay resident in the shared memory during the respective computations. Each tile is assigned to an individual GPU thread block for processing.

In our case, the problem domain can be represented as a rectangular matrix with rows corresponding to time samples and columns corresponding to shift values. A tile will therefore correspond to an interval of points and interval of shift values. A respective thread block will compute partial correlation sums corresponding to these intervals. To obtain full correlation values, these sums will be added at the second pass. Partitioning parameters depend on the architecture of the GPU device used.

A schematic illustration of our GPU parallelization approach is shown in Figure 2. It shows a fine-grained parallelization scheme for a simplified model of computations with the window length of 10,240 samples, maximum correlation lag of 2048, and CUDA thread block size of 1024 threads. CUDA threads in a single block process 1024 time shifts and compute partial sums for 2048 (out of 10,240) samples for each shift value. Therefore, the processing of 10,240 samples for 2048 time shifts requires five partial sums for each sample as well as two thread blocks for each partial sum, and all shifts must yield a total of ten tiles.

For phase correlations (Schimmel *et al.*, 2011), the same approach can be used because its computation involves a sum very similar to equation (1) for the classical correlation.

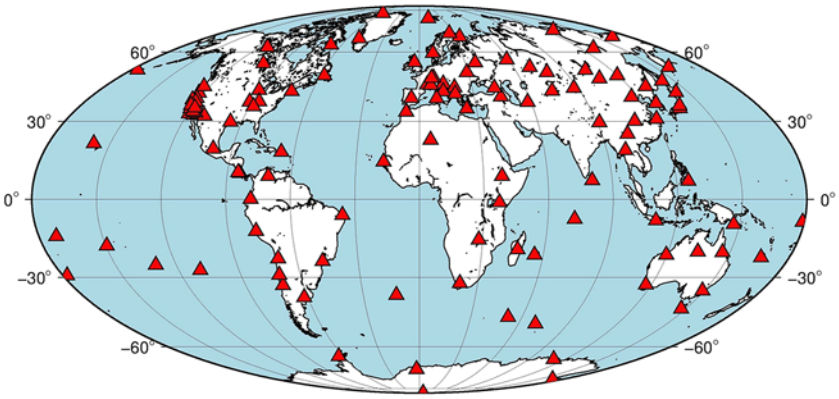


▲ **Figure 2.** Schematic illustration of the correlation parallelization on graphic processing unit for the simplified model case of partitioning for two windows (differently colored) containing 10 times 1024 points each and 1024 correlation shifts. The color version of this figure is available only in the electronic edition.

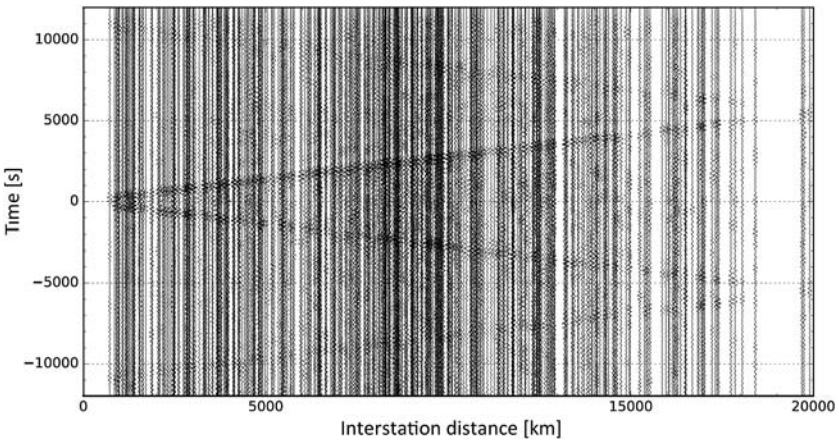
EXAMPLE

We illustrate the performance of *Mirmex* using a waveform dataset that has been recorded continuously for one year, from 1 January 2014 to 1 January 2015, on the LHZ channel of 188 globally distributed stations shown in Figure 3. Excluding autocorrelations, we thus consider a total of $188 \times 187/2 = 17,578$ vertical–vertical station pairs. Using a sampling rate of 1 Hz, we organize the one-year-long recordings of each station into time windows with $2^{15} = 32,768$ samples and an overlap of 6000 s (~1.7 hrs). The maximum correlation lag is ±12,000 s (~3.3 hrs).

To process data and compute correlations, we used the massively parallel heterogeneous high-performance computing system *Piz Daint*,



▲ **Figure 3.** Geographic distribution of the broadband stations that provided data for the performance test. The color version of this figure is available only in the electronic edition.



▲ **Figure 4.** Selection of 420 classical interstation correlations computed during the performance tests. The period range is 3–5 mHz.

a supercomputer of the Cray XC 30 family operated by the Swiss National Supercomputing Center. We performed our tests prior to a recent upgrade, when it featured 5272 computing nodes, each equipped with an 8-core 64-bit Intel Sandy Bridge CPU (Intel Xeon E5-2670), an NVIDIA Tesla K20x GPU accelerator with 6 GB GDDR5 memory, and 32 GB of host memory. The nodes were connected by the proprietary Aries interconnect from Cray.

Using a single CPU of *Piz Daint*, the wall-clock time per station pair is 803.6 s for classical correlation and 6359.5 s for phase correlation. Nearly all of this time, more than 99%, are needed to compute Fourier transforms and sums in the correlations themselves. The time needed to read and preprocess the traces is practically negligible. Additionally, exploiting the GPU accelerator available on each *Piz Daint* node reduces these wall-clock times to 3.39 and 17.82 s, respectively. The wall-clock times are thus reduced by factors of 237 for the classical and 357 for the phase correlation.

The significance of these wall-clock time reductions becomes apparent when the complete dataset with 17,578 station pairs is considered. On one CPU, 163 days would be needed for classical correlation, and 1293 days for phase correlation. This illustrates, for instance, that our phase correlation dataset is not computable on one CPU within the time limits of a typical research project. With the help of the GPU accelerator, wall-clock times are translated to tractable time scales of 0.68 days (16 hrs) and 3.62 days (87 hrs), respectively. Further reduction in wall-clock time can be achieved using multiple nodes. Because the parallel noise correlation does not involve any significant internode communication, scaling is perfect by design. Thus, using for instance 32 nodes allows us to compute the complete phase correlation dataset in less than 3 hrs. This allows us to efficiently test the impact of different processing schemes and their impact on inferences of Earth structure or noise sources. A selection of 420 classical interstation correlations filtered in the 3–5 mHz frequency range is shown in Figure 4.

CONCLUSIONS

Using a heterogeneous supercomputing platform, we achieved a speedup by an order of magnitude or higher compared, on a socket to socket basis, to the purely homogeneous system. Our solution enables the cross correlation of a large number of seismic stations, and at high resolution, thus making practical use of large volumes of accumulated seismic data. Furthermore, the availability of *Mirmex* permits the computation of large correlation datasets on-the-fly, thus eliminating the need for large mass storage to archive noise correlation results. This is of particular relevance when multiple combinations of processing parameters need to be independently applied to the same large set of source data.

Mirmex includes a modular library of software components that can be reused for a wide class of seismology applications, thus reducing the programming effort required for their implementation on massively parallel heterogeneous computing platforms.

As directions for future research, we consider (1) the automated classification of correlation results using machine learning, (2) the development of a platform for monitoring the seasonal migration of noise sources, and (3) the application of *Mirmex* to full waveform inversion projects.

DATA AND RESOURCES

Seismograms used in this study were obtained from the Incorporated Research Institutions for Seismology (IRIS) Data Management Center at www.iris.edu (last accessed August

2016) and at http://ds.iris.edu/data/distribution/ (last accessed May 2017). *Mirmex* is open source and can be obtained freely from the authors upon request.

ACKNOWLEDGMENTS

Our developments were supported by the Swiss National Supercomputing Centre (CSCS) through projects ch1 and d26, and by the Swiss National Science Foundation (SNF) under Grant 200021_149143. We thank Daniel Bowden and the editors for constructive comments that helped us to improve the article. *Mirmex* is freely available on the software webpages of the ETH Computational Seismology Group at http://www.cos.ethz.ch/software.html (last accessed May 2017).

REFERENCES

Bensen, G. D., M. H. Ritzwoller, M. P. Barmin, A. L. Levshin, F. 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,** 1239–1260.

Beyreuther, M., R. Barsch, L. Krischer, and J. Wassermann (2010). ObsPy: A Python toolbox for seismology, *Seismol. Res. Lett.* **81,** 47–58.

Bowden, D. C., M. D. Kohler, V. C. Tsai, and D. S. Weeraratne (2016). Offshore Southern California lithosphereic velocity structure from noise cross-correlation functions, *J. Geophys. Res.* **121,** 3415–3427, doi: 10.1002/2016JB012919.

Brenguier, F., M. Campillo, C. Haziioannou, N. M. Shapiro, R. M. Nadeau, and E. Larose (2008). Postseismic relaxation along the San Andreas fault at Parkfield from continuous seismological observations, *Science* **321,** 1478–1481.

Briand, X., M. Campillo, F. Brenguier, P. Boué, P. Poli, P. Roux, and T. Takeda (2013). Processing of terabytes of data for seismic noise analysis with the Python codes of the Whisper Suite, *AGU Fall Meeting Abstracts*, IN51B–1544.

Cupillard, P., L. Stehly, and B. Romanowicz (2011). The one-bit noise correlation: A theory based on the concepts of coherent and incoherent noise, *Geophys. J. Int.* **184,** 1397–1414.

Curtis, A., and D. Halliday (2010). Directional balancing for seismic and general wavefield interferometry, *Geophysics* **75,** no. 1, doi: 10.1190/1.3298736.

de Ridder, S. A. L., B. L. Biondi, and R. G. Clapp (2014). Time-lapse seismic noise correlation tomography at Valhall, *Geophys. Res. Lett.* **41,** 6116–6122.

Delaney, E., L. Ermert, K. Sager, A. Kritski, S. Bussat, and A. Fichtner (2017). Passive seismic monitoring with non-stationary noise sources, *Geophysics* doi: 10.1190/geo2016-0330.1.

Ermert, L., A. Villasenor, and A. Fichtner (2016). Cross-correlation imaging of ambient noise sources, *Geophys. J. Int.* **204,** 347–364.

Fichtner, A., L. Stehly, L. Ermert, and C. Boehm (2017). Generalized interferometry—I. Theory for inter-station correlations, *Geophys. J. Int.* **208,** 603–638.

Groos, J. C., S. Bussat, and J. R. R. Ritter (2012). Performance of different processing schemes in seismic noise cross-correlations, *Geophys. J. Int.* **188,** 498–512.

Hanasoge, S. M., and M. Branicki (2013). Interpreting cross-correlations of one-bit filtered noise, *Geophys. J. Int.* **195,** 1811–1830.

Krischer, L., T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron, and J. Wassermann (2015). ObsPy: A bridge for seismology into the scientific Python ecosystem, *Comput. Sci. Discov.* **8,** no. 1, doi: 10.1088/1749-4699/8/1/014003.

Lecocq, T., C. Caudron, and F. Brenguier (2014). MSNoise, a Python package for monitoring seismic velocity changes using ambient seismic noise, *Seismol. Res. Lett.* **85,** 715–726.

Megies, T., M. Beyreuther, R. Barsch, L. Krischer, and J. Wassermann (2011). ObsPy—What can it do for data centers and observatories? *Ann. Geophys.* **54,** 47–58.

Mordret, A., N. Shapiro, and S. Singh (2014). Seismic noise-based time-lapse monitoring of the Valhall overburden, *Geophys. Res. Lett.* **41,** 4945–4952.

Nakata, N., J. P. Chang, J. F. Lawrence, and P. Boué (2015). Body wave extraction and tomography at Long Beach, California, with ambient-noise interferometry, *J. Geophys. Res.* **120,** 1159–1173.

Obermann, A., T. Kraft, E. Larose, and S. Wiemer (2015). Potential of ambient seismic noise techniques to monitor the St. Gallen geothermal site (Switzerland), *J. Geophys. Res.* **120,** 4301–4316, doi: 10.1002/2014JB011817.

Obermann, A., T. Planes, E. Larose, and M. Campillo (2013). Imaging preeruptive and coeruptive structural and mechanical changes of a volcano with ambient seismic noise, *J. Geophys. Res.* **118,** 1–10.

Sabra, K. G., P. Gerstoft, P. Roux, and W. A. Kuperman (2005). Surface wave tomography from microseisms in Southern California, *Geophys. Res. Lett.* **32,** L14311, doi: 10.1029/2005GL023155.

Saygin, E., and B. L. N. Kennett (2012). Crustal structure of Australia from ambient seismic noise tomography, *J. Geophys. Res.* **117,** no. B01304, doi: 10.1029/2011JB008403.

Schimmel, M., and H. Paulssen (1997). Noise reduction and detection of weak, coherent signals through phase-weighted stacks, *Geophys. J. Int.* **130,** 497–505.

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.

Shapiro, N. M., M. Campillo, L. Stehly, and M. Ritzwoller (2005). High resolution surface wave tomography from ambient seismic noise, *Science* **307,** 1615–1618.

Stehly, L., M. Campillo, and N. M. Shapiro (2006). A study of the seismic noise from its long-range correlation properties, *J. Geophys. Res.* **111,** no. B10306, doi: 10.1029/2005JB004237.

Stehly, L., B. Fry, M. Campillo, N. M. Shapiro, J. Guilbert, L. Boschi, and D. Giardini (2009). Tomography of the Alpine region from observations of seismic ambient noise, *Geophys. J. Int.* **178,** 338–350.

Yang, Y., and M. H. Ritzwoller (2008). Characteristics of ambient seismic noise as a source for surface wave tomography, *Geochem. Geophys. Geosys.* **9,** Q02008, doi: 10.1029/2007GC001814.

Ye, T., and M. H. Ritzwoller (2015). Directionality of ambient noise on the Juan de Fuca plate: Implications for source locations of the primary and secondary microseisms, *Geophys. J. Int.* **201,** 429–443.

Andreas Fichtner
Laura Ermert
Alexey Gokhberg
Department of Earth Sciences
ETH Zurich
Sonneggstrasse 5
8092 Zurich, Switzerland
andreas.fichtner@erdw.ethz.ch

Published Online 31 May 2017