Preprint: Submitted to Geophysics for consideration.

A Linear Algorithm for Ambient Seismic Noise Double Beamforming Without Crosscorrelations

Eileen R. Martin

eileenrmartin@vt.edu

Department of Mathematics, Virginia Polytechnic Institute and State University,

225 Stanger St. Blacksburg, VA, 24060

(December 31, 2019)

Running head: **Linear Double Beamforming Algorithm**

ABSTRACT

Geoscientists and engineers are increasingly using denser arrays for continuous seismic monitoring, and often turning to ambient seismic noise interferometry for low-cost near-surface imaging. While ambient noise interferometry greatly reduces acquisition costs, the computational cost of pair-wise comparisons between all sensors can be prohibitively slow or expensive for applications in engineering and environmental geophysics. Double beamforming of noise correlation functions is a powerful technique to extract body waves from ambient noise, but it is typically performed via pair-wise comparisons between all sensors in two dense array patches (scaling as the product of the number of sensors in one patch with the number of sensors in the other patch). By rearranging the operations involved in the double beamforming transform, we propose a new algorithm that scales as the sum of the number of sensors in two array patches. Compared to traditional double beamforming of noise correlation functions, the new method is more scalable, easily parallelized, and does

not require raw data to be exchanged between dense array patches.

INTRODUCTION AND PRIOR WORK

Recent years have seen the rapid development of new technologies for using off-the-shelf MEMS accelerometers, low-cost and low-power nodes, and fiber optics repurposed as dense seismic arrays (Evans et al., 2014), (Manning et al., 2018), (Martin et al., 2018). These new acquisition techniques have enabled orders of magnitude denser data acquisition particularly for long-term monitoring, and are pushing the limits of seismic processing for near-surface engineering geophysics. An increasingly popular technique for near-surface imaging with these low-cost, dense systems is the use of ambient noise interferometry, which estimates signals similar to active seismic source data by performing crosscorrelations on passively recorded data responding to random seismic sources and averaging results over many windows of time (Bensen et al., 2007). In the past, ambient noise interferometry has been primarily applied at low frequencies (below 1 Hz) with sparse sensor spacing and low-frequency sampling in time. However, we wish to image smaller features in the shallow subsurface for many applications, and this requires higher frequencies which must be acquired at high-density, and several studies have investigated the use of ambient noise interferometry on dense DAS arrays already (Ajo-Franklin et al., 2015), (Dou et al., 2017), (Martin et al., 2018). Because the cost of calculating crosscorrelations between every pair of sensors grows quadratically with the number of sensors, reducing the distance between sensors by a factor of 10 (say to move from 100 meter spaced geophones in a node array to 10 meter spaced channels in a DAS array) results in a factor of 100 more computation for crosscorrelations. This means that packing sensors more densely is a particularly impactful issue in ambient noise processing.

Sometimes it is necessary to calculate noise correlation functions. When picking arrival times for tomographic imaging of the near surface, these noise correlation functions must have been calculated to determine pick times. In such instances, calculations may be sped up by performing all crosscorrelations on GPUs (Fichtner et al., 2017), and can be further sped up by performing crosscorrelations of data in a compressed matrix-factorized form if some information loss is accepted (Martin, 2019). Additionally, having a small sparse subset of pairs that have crosscorrelations calculated can be useful for pinpointing quality control issues related to local noise sources, which are of particular concern in engineering geophysics around infrastructure and in urban areas (Nakata et al., 2011), (Nakata et al., 2015), (Huot et al., 2017), (Martin et al., 2018).

There is reason to calculate noise correlation functions in some cases, but in many cases, such as surface wave dispersion analysis through multichannel analysis of surface waves (MASW) (Park et al., 2007), or the calculation of double beamforming transforms (Boué et al., 2013) the actual goal of this calculation is a transform of the noise correlation functions. In MASW, for instance, we calculate the dispersion image: the Fourier transform of a τ-p transform of each noise correlation function virtual source gather. Data from N receivers is compressed into one value per frequency and slowness pair. By reorganizing the calculation, it is possible to perform calculations that go directly from raw data to a dispersion image in $O(N)$ calculations, skipping the $O(N^2)$ explicit calculations of pair-wise noise correlation functions (Martin, 2018). These new algorithms show that for ambient noise interferometry to meet the expectation of low-cost near-surface imaging on modern, large, dense arrays, we must be willing to let go of the explicit calculation of all noise correlation functions.

Consider two array patches, array patch A with N_A sensors, and array patch B with N_B

sensors. This paper proposes rearranging the calculations in the double beamforming transform $b(t, \mathbf{u}_A, \mathbf{u}_B)$, evaluated at multiple combinations of time lags, t, and two-dimensional slownesses $\mathbf{u}$ across the two arrays, which would traditionally be done by slant-stacks of their averaged noise correlation functions following ambient noise interferometry. The rearranged calculations yield a new algorithm that:

1. is flexible for incorporating a variety of preprocessing workflows (and easily adapted for interferometry by crosscorrelation or deconvolution),
2. avoids explicitly calculating noise correlation functions,
3. scales as the sum of the number of sensors in both array patches (in serial),
4. requires just 2 rounds of communication in parallel per time window,
5. does not require any raw data to be centralized or shared between the two arrays

The last point is particularly important when seismic arrays are managed by different stakeholders, or when a site's internet connectivity or power resources are limited.

INTRODUCTION TO DOUBLE BEAMFORMING

Double beamforming has proven useful in better separating body wave energy from surface wave energy in ambient noise interferometry between patches of sensors. This has enabled higher-resolution subsurface imaging without the expense of seismic source crews (Boué et al., 2013), (Boué et al., 2014), even in urban areas where active source seismic surveys are often not an option (Nakata et al., 2015). Consider two arrays of sensors, A and B. Array A has N_A sensors and array B has N_B sensors. Let $d_n(\mathbf{x}_{A,i}, t)$ be an ambient seismic noise data trace recorded at time t in the window $t_n < t < t_{n+1}$ at the i^{th} sensor in array A

at position $\mathbf{x}_{A,i}$, perhaps with tapers near the ends and some minimal preprocessing applied as described in (Bensen et al., 2007). Denote the Fourier transform of the data in the n^{th} time window as $\hat{d}_n(\mathbf{x},\omega) = \mathcal{F}_t\left(d_n(\mathbf{x},t)\right)$. There are three types of interferometry that may be done between any receiver pair in the n^{th} time window: crosscorrelation, crosscoherence, and deconvolution. For crosscorrelation:

$$c_n(T,\mathbf{x}_{A,k},\mathbf{x}_{B,j}) = \int_{t_n}^{t_{n+1}} d(t,\mathbf{x}_{A,k})d(\mathbf{x}_{B,j},t+T)dt \tag{1}$$

so the time series $c_n(T,\mathbf{x}_{A_k},\mathbf{x}_{B_j})$ represents the noise correlation function of the n^{th} time window of the k^{th} receiver in array A with the j^{th} receiver in array B at time-lag T. We refer to the $c_n(T,\mathbf{x}_{A,k},\mathbf{x}_{B,j})$ time series as a these sensors' noise correlation function for the n^{th} time window.

Let $\mathbf{u}_A$ and $\mathbf{u}_B$ represent the apparent slowness of a ray path at array A and B, respectively. To calculate the double beamforming transform of the pair-wise average crosscorrelations (only inter-array correlation, none within either array) over N_w windows of time, we perform a $\tau - p$ transform with movout across both arrays:

$$b(t,\mathbf{u}_A,\mathbf{u}_B) \quad = \quad \sum_{k=1}^{N_A}\sum_{j=1}^{N_B}\sum_{n=1}^{N_w} \frac{c_n(t-\tau(\mathbf{x}_{A,k},\mathbf{u}_A)+\tau(\mathbf{x}_{B,j},\mathbf{u}_B),\mathbf{x}_{A,k},\mathbf{x}_{B,j})}{N_w N_A N_B} \tag{2}$$

where $\tau(\mathbf{x},\mathbf{u})$ is the time lag of a signal from some an origin in the middle of its array patch based on the relative location of the point $\mathbf{x}$ from the origin and the slowness being tested. If one were to calculate all needed pairwise-crosscorrelations $\hat{c}_n$ of noise traces with N_t samples in the time domain with N_τ time lags of interest, this would require $O(N_A N_B N_t N_\tau)$ operations per time window, followed by scanning over all $\mathbf{u}_A$ and $\mathbf{u}_B$ slant stacks of interest.

DERIVATION OF NEW ALGORITHM

Like the fast dispersion image algorithm in (Martin, 2018), we will separate factors between different sensor pairs by calculating the double beamforming transform in the frequency domain (and after calculation, performing an inverse Fourier transform to recover $b(t, \mathbf{u}_A, \mathbf{u}_B)$. Starting from equation 2 rewritten in the frequency domain, one can use fundamental properties of crosscorrelations (frequency-wise multiplication with a complex conjugate) and the Fourier transform to rewrite the frequency domain double beamforming transform between arrays A and B:

$$
\begin{aligned}
&\hat{b}(\omega, \mathbf{u}_A, \mathbf{u}_B) \\
&= \mathcal{F}_t \left(\sum_{k=1}^{N_A} \sum_{j=1}^{N_B} \sum_{n=1}^{N_w} \frac{c_n(t - \tau(\mathbf{x}_{A,k}, \mathbf{u}_A) + \tau(\mathbf{x}_{B,j}, \mathbf{u}_B), \mathbf{x}_{A,k}, \mathbf{x}_{B,j})}{N_w N_A N_B} \right) \\
&= \sum_{k=1}^{N_A} \sum_{j=1}^{N_B} \sum_{n=1}^{N_w} \frac{\hat{c}_n(\omega, \mathbf{x}_{A,k}, \mathbf{x}_{B,j})}{N_w N_A N_B} e^{2\pi i(-\tau(\mathbf{x}_{A,k}, \mathbf{u}_A) + \tau(\mathbf{x}_{B,j}, \mathbf{u}_B))\omega} \\
&= \sum_{n=1}^{N_w} \sum_{k=1}^{N_A} e^{-2\pi i \tau(\mathbf{x}_{A,k}, \mathbf{u}_A)\omega} \sum_{j=1}^{N_B} \frac{\hat{c}_n(\omega, \mathbf{x}_{A,k}, \mathbf{x}_{B,j})}{N_w N_A N_B} e^{2\pi i \tau(\mathbf{x}_{B,j}, \mathbf{u}_B)\omega} \\
&= \frac{1}{N_w} \sum_{n=1}^{N_w} \sum_{k=1}^{N_A} e^{-2\pi i \tau(\mathbf{x}_{A,k}, \mathbf{u}_A)\omega} \sum_{j=1}^{N_B} \frac{\hat{d}_n(\omega, \mathbf{x}_{A,k})}{N_A} \frac{\hat{d}_n^*(\omega, \mathbf{x}_{B,j})}{N_B} e^{2\pi i \tau(\mathbf{x}_{B,j}, \mathbf{c}_B)\omega} \\
&= \frac{1}{N_w} \sum_{n=1}^{N_w} \left(\sum_{k=1}^{N_A} \frac{\hat{d}_n(\omega, \mathbf{x}_{A,k})}{N_A} e^{-2\pi i \tau(\mathbf{x}_{A,k}, \mathbf{c}_A)\omega} \right) \left(\sum_{j=1}^{N_B} \frac{\hat{d}_n^*(\omega, \mathbf{x}_{B,j})}{N_B} e^{2\pi i \tau(\mathbf{x}_{B,j}, \mathbf{c}_B)\omega} \right) \qquad (3)
\end{aligned}
$$

The first step is to rewrite a time-lag as a frequency-domain phase-shift, then split up that phase shift into parts specific to array A and parts specific to array B. The noise correlation function c can be replaced with the (preprocessed) data d involved in the crosscorrelation, then every piece of data and information specific to array A can be factored separately from array B. In this way, it is clear that within each time step, there is one multi-dimensional array (N_ω values by the number of $\mathbf{u}$ values of interest) that need to be calculated for array A, and another for array B, then these two arrays will be multiplied element-wise with an

outer product (assuming $\mathbf{u}_A$ and $\mathbf{u}_B$ are scanned over the same range of slownesses).

PROPOSED ALGORITHM

The derivation in the previous section naturally suggests an algorithm that processes data one time window and one sensor at a time. At time window n one should calculate a tensor of phase-shifted and stacked data for array A, and a similar tensor for array B:

$$R_{n,A}(\omega, \mathbf{u}_A) = \sum_{k=1}^{N_A} \frac{\hat{d}_n(\omega, \mathbf{x}_{A,k})}{N_A} e^{-2\pi i \tau(\mathbf{x}_{A,k}, \mathbf{u}_A)\omega} \tag{4}$$

$$R^*_{n,B}(\omega, \mathbf{u}_B) = \sum_{j=1}^{N_B} \frac{\hat{d}^*_n(\omega, \mathbf{x}_{B,j})}{N_B} e^{2\pi i \tau(\mathbf{x}_{B,j}, \mathbf{u}_B)\omega} \tag{5}$$

We refer to the calculation of each of these R_n factors as the *phase 1 calculation*. Then in the *phase 2 calculation* these two R_n tensors can be multiplied as an outer product to yield the 5-dimensional double beamforming transform within the n^{th} time window for each frequency, slowness on array A, and slowness on array B: $\hat{b}_n(\omega, \mathbf{u}_A, \mathbf{u}_B)$. For each slowness pair, we calculate the inverse Fourier transform of $\hat{b}_n(\omega, \mathbf{u}_A, \mathbf{u}_B)$ then restrict to the times of interest $b_n(t, \mathbf{u}_A, \mathbf{u}_B)$. As with any ambient noise processing, we will need to average the results over many time windows to obtain a reliable estimate of the overall double beamforming transform, so we will repeat this process for each of N_w time windows, as outlined in the following algorithm:

Algorithm 1 Serial Double Beamforming Algorithm

Initialize $b(t, \mathbf{u}_A, \mathbf{u}_B)$ with zeros

for time window $n = 1, \ldots, N_w$ **do**

Phase 1 calculation on each array patch:

for each array $p \in \{A, B\}$ **do**

Initialize $R_{n,p}(\omega, \mathbf{u}_p)$ with zeros

for each sensor $k = 1, \ldots, N_p$ in array p **do**

Preprocess, take FFT, yielding $\hat{d}_n(\omega, \mathbf{x}_{p,k})$

$R_{n,p}(\omega, \mathbf{u}_p)$ += $\frac{1}{N_p}\hat{d}_n(\omega, \mathbf{x}_{p,k})e^{-2\pi i\tau(\mathbf{x}_{p,k}, \mathbf{u}_p)\omega}$ for each $(\omega, \mathbf{u}_p)$

end for

end for

Phase 2 calculation comparing the two array patches:

for each $(\mathbf{u}_A, \mathbf{u}_B)$ of interest **do**

$\beta(\omega) = R_{n,A}(\omega, \mathbf{u}_A)R^*_{n,B}(\omega, \mathbf{u}_B)$

$b(t, \mathbf{u}_A, \mathbf{u}_A)$ += $IFFT(\beta(\omega))$ restricted to times of interest

end for

end for

Scale $b(t, \mathbf{u}_A, \mathbf{u}_B)$ by $\frac{1}{N_t}$ for each $(\omega, \mathbf{u}_A, \mathbf{u}_B)$

In practice $\hat{b}(\omega, \mathbf{u}_A, \mathbf{u}_B)$ gets very large if the number of frequencies and test slownesses used is large, so calculating the product of R_A and R_B and taking the inverse Fourier transform one test-slowness pair at a time greatly reduces memory requirements compared to calculating all of $\hat{b}(\omega, \mathbf{u}_A, \mathbf{u}_B)$ prior to calculating the inverse Fourier transform.

Complexity Analysis

Here, we analyze the cost of traditional double beamforming and of the new proposed algorithm, both in serial. We will only consider the cost of analysis for a single time window where each trace has N_t time samples. We are interested in N_τ time lags in our crosscorrelations and double beamforming. Again, we will consider array patch A with N_A sensors and array patch B with N_B sensors. Denote the number of slownesses of interest for each array as N_{uA} and N_{uB} and the number of angles of interest as $N_{\theta A}$ and $N_{\theta B}$.

As a baseline, consider the traditional algorithm for double beamforming. The first step in traditional double beamforming is to calculate all needed crosscorrelations, which would cost $O(N_A N_B N_t N_\tau)$ operations if done in the time domain, and $O(N_A N_B N_t \log N_t)$ operations if done in the frequency domain. Then, the double beamforming is performed for N_τ time lags based on $N_{uA}N_{\theta A}N_{uB}N_{\theta B}$ slowness combinations, each with contributions from $N_A N_B$ noise correlation functions, leading to a total cost of $O(N_A N_B N_{uA} N_{\theta A} N_{uB} N_{\theta B} N_\tau)$ for the double beamforming transform after crosscorrelation.

Now consider the new double beamforming algorithm. First, phase 1 operates on array patch A, looping through N_A sensors and for each of $O(N_t)$ frequencies (may be truncated to nearest power of 2 for efficiency) and $N_{uA}N_{\theta A}$ slownesses calculating the phase shifted frequency-domain contribution to R_A. The cost of phase 1 for array patch A is $O(N_A N_{uA} N_{\theta A} N_t)$. Similarly, the cost of phase 1 for array patch B is $O(N_B N_{uB} N_{\theta B} N_t)$. In phase 2, the algorithm loops over $N_{uA}N_{\theta A}N_{uB}N_{\theta B}$ slowness combinations, and for each slowness combination performs $O(N_t)$ operations to multiply part of R_A and R_B then performs $O(N_t \log N_t)$ operations to calculate the inverse Fourier transform. This ultimately

leads to an $O(N_{uA}N_{\theta A}N_{uB}N_{\theta B}N_t \log N_t)$ cost for phase 2. The complexity analyses for both the traditional and new algorithms are summarized in the table below.

Workflow	Part	Complexity
Traditional	crosscorrelations	$O(N_A N_B N_t N_\tau)$
Traditional	Double Beamforming	$O(N_A N_B N_{uA} N_{\theta A} N_{uB} N_{\theta B} N_\tau)$
New	Phase 1	$O(N_A N_{uA} N_{\theta A} N_t) + O(N_B N_{uB} N_{\theta B} N_t)$
New	Phase 2	$O(N_{uA} N_{\theta A} N_{uB} N_{\theta B} N_t \log N_t)$

Note that in the new algorithm, phase 1 splits apart the cost of the number of sensors in A and the number of sensors in B, effectively turning the cost of phase 1 into $O(N_A + N_B)$ when it was previously $O(N_A N_B)$ for a given set of slownesses and time lags of interest. If all of our data acquisition systems were made more dense, this means we expect a linear growth with sensor density using the new algorithm as opposed to quadratic growth with the traditional algorithm. Phase 2 of the new algorithm is completely independent of the number of sensors, while the second part of the traditional double beamforming algorithm grows quadratically with the sensor density. As a note of caution, this order of magnitude improvement with respect to the number of sensors may not lead to runtime improvements if N_A and N_B are relatively small compared to the number of slownesses of interest. However, as we adopt increasingly dense modern sensing systems, this is rarely the case.

SCALABILITY TESTS ON FIELD DATA

Using the same subset of broadband Transportable Array (US Array) data as (Boué et al., 2014), we test this new double beamforming transform algorithm and compare to the traditional process of crosscorrelations followed by double beamforming. Specifically, vertical

data from a patch of 9 broadbands near the North/South Dakota Border and another patch of 9 broadbands in Southeastern New Mexico. Boué et al. (2014) previously demonstrated that these ambient noise data show significant surface wave and body wave energy after stacking 4 hour windows throughout 3 months of recording, and that the double beamforming transform (not just single beamforming) is needed to identify and distinguish between the body wave and surface wave energy in the ambient noise crosscorrelations.

Data were preprocessed using the same workflow as Boué et al. (2014). To review, this includes: removing broadband sensor response, bandpassing from 5 to 150 second periods, breaking the data into 4 hour windows, zeroing any windows with large transient signals (if energy is more than 1.5 times the daily average energy at the same location), whitening the remaining time series between periods of 5 and 150 seconds, removing windows with at least 10 % of zero samples (recording issues), time-domain clipping of any values more than 3.8 times each window's standard deviation.

The first four hour window of preprocessed data were fed into both the traditional method (crosscorrelations followed by double slant-stacks) and the new algorithm for cost comparison. The methods were tested in serial on a laptop with a 2.7 GHz Intel Core i7 4-core processor with 256 KB L2 cache per core, 8 MB L3 cache and 16 GB LPDDR3 memory. These are typical laptop specifications that a scientist would easily have access to in the field.

To verify the theoretical complexity of both methods with respect to the number of slownesses ($N_{uA}, N_{\theta A}, N_{uB}, N_{\theta B}$), both methods were run on the same data with 9 sensors per array patch, but with $N_{ua} = N_{\theta A} = N_{uB} = N_{\theta B}$ increasing from 2, 4, 8, then 16. These timings are plotted in Figure 2 and reported in Table 1. Note that preprocessing was not

included in any of these times: data were preprocessed and saved to disk prior to analysis because preprocessing varies significantly from one dataset to another (thus not relevant to the comparison between algorithms). crosscorrelations are performed independently of slownesses, so only one crosscorrelation timing is shown. For all $N_{ua} = N_{\theta A} = N_{uB} = N_{\theta B}$ values tested, the new algorithm (phase 1 time plus phase 2 time) performs faster than the traditional algorithm (crosscorrelation time plus double beamforming time). Traditional double beamforming and phase 2 were both predicted to take roughly 16x as long to run, and phase 1 was predicted to take 4x as long to run as every slowness dimension was doubled. The actual growth factors, indicated by the "Growth" column of the table (time for that test divided by time for previous line), show that the real data experiments generally agree with the predicted trend.

[Figure 1 about here.]

[Figure 2 about here.]

More importantly, we must verify that scalability with number of sensors matches the predicted trends, as the growing number of sensors enabled by new technologies is our primary motivator in this study. For our intitial testing, we used the same 9 sensors tested in Boué et al. (2014). To test the scalability in number of sensors, we simply made a list of sensors for each array that repeated through the same sensors multiple times to emulate the action of both algorithms on $N_A = N_B$ equal to 9, 18, 36 and 72. The new algorithm was additionally run for 144, 288, and 576. The number of slownesses was held constant. The timings are shown in Figure 4 and Table 3. For all tested sensor array sizes, the new algorithm was significantly faster than the traditional algorithm. The new algorithm took

less time to run on 576 sensors per patch than the old algorithm took to run with just 9 sensors per patch.

Each time the number of sensors per array patch is doubled, we expect the time for traditional crosscorrelations and double beamforming to increase by 4x, the time for the new algorithm phase 1 to increase by 2x, and the time for the new algorithm phase 2 to be unaffected. As shown by the "Growth" column of the table, the predicted growth factor for the crosscorrelations and the new phase 1 were reflected in the real data experiment timings. However, the traditional double beamforming after crosscorrelation only grows by an approximate factor of 2 each time the number of sensors was doubled, although it was predicted to grow by a factor of 4. This discrepancy is because the code is written such that the dependency on N_B is using vectorized Numpy calls, which can greatly reduce the cost compared to another for loop over sensors. Even with this efficiency (which is particular to this Python implementation), the new algorithm shows significant speedups over the traditional algorithm, running in seconds what took many minutes to hours to run previously.

[Figure 3 about here.]

[Figure 4 about here.]

Parallelization

The proposed serial algorithm greatly reduces the cost of double beamforming between array patches with many sensors, but as ultra-dense arrays become increasingly common, we must also consider scalability with parallel computing. The traditional beamforming algorithm

would require all-to-all communication over sensors, which requires a great deal of costly data movement. In contrast, both phases of the new proposed algorithm algorithm can be easily parallelized. The parallelization within each time window in phase 1 can happen at two levels:

- the calculation of $R_{n,A}(\omega, \mathbf{u}_A)$ is indepnedent of $R_{n,B}(\omega, \mathbf{u}_B)$ so these computations can be carried out simultaneously,
- and for each array $p \in \{A, B\}$ the calculation of $R_{n,p}(\omega, \mathbf{u}_p)$ can be parallelized over receivers within either array.

The first level of parallelism is clear, and note that even the data reads can be done completely separately, so the raw data between array A and B never need to be shared, just their resulting R_A and R_B factors. To argue for the second type of parallelism, note that many preprocessing workflows perform trace-by-trace operations (tapering, whitening, thresholding large events, etc...) as described in (Bensen et al., 2007). Thus it seems natural to assign each sensor to processes independently. After the preprocessing is over, the next step in the calculation is to apply a variety of phase shifts to the preprocessed frequency-domain data (or its complex conjugate on array B), again, a trace-by-trace operation. Finally, those results must be reduced, accumulating through addition into $R_{n,p}(\omega, \mathbf{u}_p)$. This single round of collective communication at phase 1 scales as $O(N_{up} N_{\theta p} N_t \log M)$ where M is the number of machines over which the data in the array are divided. This process of two-level parallelism in phase 1 is diagrammed in Figure 5 assuming each array has its receiver data distributed over four tasks (for illustrative purposes). Note that the only data that must be shared between array A and array B is $R_A(\omega, \mathbf{u}_A)$ and $R_B(\omega, \mathbf{u}_B)$.

[Figure 5 about here.]

Phase 2 can also be easily parallelized. The calculations in phase 2 are independent of the sensors in each array, so the parallelization must happen over tested slowness values. Assuming the data in R_A is arranged in the order (u_A, θ_A, ω), and the beamforming results in b are arranged in the order $(u_A, \theta_A, u_B, \theta_B)$ with u_A being the outermost (slowest) axis, one can break up R_A along its u_A axis (outermost) onto difference processes, so each process gets some subset of u_A values. Each process would also get a copy all of R_B, then each process would loop through its subset of u_A values, through all θ_A values, through all u_B values and through all θ_B values. At the end, each process would have a subset of b which can be easily appended along the outermost axis to make up the full b.

CONCLUSIONS

As high-density continuous seismic monitoring systems (particularly DAS, low-power nodes, and MEMS accelerometers) are being adopted increasingly for near-surface monitoring with ambient noise interferometry imaging, we can drastically improve the computational efficiency of ambient noise analysis by avoiding the explicit calculation of all pair-wise noise correlation functions whenever possible. It has been previously shown that the serial algorithm for calculating dispersion images of noise correlation functions at N sensors responding to N virtual sources could be reduced in cost from quadratic to linear by factoring the overall dispersion image into virtual source and receiver components that can be calculated separately. Using a similar technique, we have shown how to factor the double beamforming transform into two factors specific to two array patches.

This refactoring of the double breamforming transform naturally yields an algorithm that scales as the sum of the number of sensors in the arrays, much more scalable than the traditional method that scales as the product of the number of sensors in the arrays.

Further, this algorithm is easily parallelized both over sensors within either array, as well as between the two arrays, and does not require raw data to be sent between the two arrays (a major benefit when multiple stakeholders are managing arrays or when internet connectivity is limited).

By adopting this new algorithm, geoscientists and engineers can rapidly calculate the double beamforming transform between dense arrays. This processing that previously required a long wait time or centralized compute cluster can now easily be performed on a laptop, enabling more computation in the field with faster turnaround time for much larger-scale problems. By increasing the number of sensors that can be stacked for double beamforming with the same computational resources, this opens up the possibility of reprocessing existing data with a higher stack power (and thus higher sensitivity to possible body waves in the data). By enabling fast analysis with fewer computing resources required, this algorithm will further enable geoscientists and engineers to test the effect of a wider variety of preprocessing techniques, which gives valuable information about the robustness and reliability of any imaging results derived from ambient noise interferometry.

Acknowledgements

I would like to thank UT-Batelle Subcontract 4000175567 "Fast Comparative Algorithms for Sensor Array Summaries," and the Virginia Tech College of Science for funds supporting this research. I thank John Hole for useful conversations inspiring the development of this method.

Data and Materials Availability

Data from the TA network were made freely available as part of the EarthScope USArray facility, operated by Incorporated Research Institutions for Seismology (IRIS) and supported

312 by the National Science Foundation, under Cooperative Agreements EAR-1261681. Code
313 provided with this manuscript uses freely available Python packages: NumPy, SciPy, ObsPy
314 Krischer et al. (2015). Code to run all examples in this paper is publicly available under
315 the MIT License at https://github.com/eileenrmartin/doubleBeamforming.

REFERENCES

Ajo-Franklin, J., N. Lindsey, T. Daley, B. Freifeld, E. Martin, M. Robertson, C. Ulrich, and A. Wagner, 2015, A field test of distributed acoustic sensing for ambient noise recording: SEG Technical Program Expanded Abstracts.

Bensen, G., M. Ritzwoller, M. Barmin, A. Levshin, F. Lin, M. Moschetti, N. Shapiro, and Y. Yang, 2007, Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements: Geophysics J. Int., **169**, 1239–1260.

Boué, P., P. Roux, M. Campillo, and X. Briand, 2014, Phase velocity tomography of surface waves using ambient noise cross correlation and array processing: J. Geophys. Res. Solid Earth, **119**, 519–529.

Boué, P., P. Roux, M. Campillo, and B. de Cacqueray, 2013, Double beamforming processing in a seismic prospecting context: Geophysics, **78**, V101–V108.

Dou, S., N. Lindsey, A. Wagner, T. Daley, B. Freideld, M. Robertson, J. Peterson, C. Ulrich, E. Martin, and J. Ajo-Franklin, 2017, Distributed acoustic sensing for seismic monitoring of the near surface: A traffic-noise interferometry case study: Scientific Reports, **7**, article number 11620.

Evans, J., R. Allen, A. Chung, E. Cochran, R. Guy, M. Hellweg, and J. Lawrence, 2014, Performance of several low-cost accelerometers: Seismological Research Letters, **85**, 147–158.

Fichtner, A., L. Ermert, and A. Gokhberg, 2017, Seismic noise correlation on heterogeneous supercomputers: Seismological Research Letters, **88**, 1141–1145.

Huot, F., Y. Ma, R. Cieplicki, E. Martin, and B. Biondi, 2017, Automatic noise exploration in urban areas: SEG Technical Program Expanded Abstracts, 5027–5032.

Krischer, L., T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron, and J. Wasser-

mann, 2015, Obspy: a bridge for seismology into the scientific python ecosystem: Computational Science & Discovery, **8**, 014003.

Manning, T., C. Brooks, A. Ourabah, M. Popham, D. Ablyazina, V. Zhuzhel, E. Holst, and N. Goujon, 2018, The case for a nimble node, towards a new land seismic receiver system with unlimited channels: SEG Technical Program Expanded Abstracts.

Martin, E., 2018, Passive imaging and characterization of the subsurface with distributed acoustic sensing: Ph.D. Dissertation, Stanford University.

———, 2019, A scalable algorithm for cross-correlations of compressed ambient seismic noise: SEG Technical Program Expanded Abstracts, to appear, Earth ArXiv preprint at doi: 10.31223/osf.io/sx9zt.

Martin, E., F. Huot, Y. Ma, R. Cieplicki, S. Cole, M. Karrenbach, and B. Biondi, 2018, A seismic shift in scalable acquisition demands new processing: Fiber-optic seismic signal retrieval in urban areas with unsupervised learning for coherent noise removal: IEEE Signal Processing Magazine, **35**, 31–40.

Nakata, N., J. P. Chang, J. F. Lawrence, and P. Boué, 2015, Body wave extraction and tomography at long beach, california, with ambientnoise interferometry: Journal of Geophysical Research: Solid Earth, **120**, 1159–1173.

Nakata, N., R. Snieder, T. Tsuji, K. Larner, and T. Matsuoka, 2011, Shear wave imaging from traffic noise using seismic interferometry by cross-coherence: Geophysics, **76**, SA97–SA106.

Park, C. B., R. D. Miller, J. Xia, and J. Ivanov, 2007, Multichannel analysis of surface waves (masw)active and passive methods: The Leading Edge, **26**, 60–64.

LIST OF FIGURES

1 This table shows the timing results (in seconds) for each part of the traditional and new algorithms as the number of sensors per patch ($N_A = N_B$) stays constant at 9, and the number of slownesses and angles tested per patch ($N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$) grows. The growth column indicates the timing at a given size problem divided by the timing of the smaller problem size on the previous line. The traditional algorithm's crosscorrelation phase has no dependence on the number of slownesses or angles. 21

2 The increase in timing as the number of test slownesses increases is plotted for both the traditional crosscorrelation and double beamforming algorithm (split into its two parts) and the new algorithm (split into its two phases). In this test the number of sensors per array patch was held constant at 9. $N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$ in each test, and this value is doubled for each experiment. The new algorithm runs faster than the old algorithm, and has similar scalability as the number of slownesses grows. 22

3 This table shows the timing results (in seconds) for each part of the traditional and new algorithms as the number of sensors per patch ($N_A = N_B$) grows. The number of slownesses and angles tested per patch ($N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$) stays constant in these tests. The growth column indicates the timing at a given size problem divided by the timing of the smaller problem size on the previous line. The new algorithm's phase 2 is independent of the number of sensors, so timing is only reported for $N_A = N_B = 9$. 23

4 The increase in timing as the number of sensors increases for the traditional crosscorrelation and double beamforming algorithm (split into its two parts) and the new algorithm (split into its two phases) is plotted. The timings show that the new algorithm runs much faster than the traditional algorithm, and is much more scalable (lower slope), primarily driven by the high cost of traditional crosscorrelations. Both the new and old algorithm were run for $N_A = N_B$ equal to 9, 18, 36, 72, and only the new algorithm was run for 144, 288, and 576. 24

5 The proposed algorithm includes a first round of trace-by-trace operations which are embarrassingly parallel. The next step is a reduce within each array of phase-shifted data into $R_p(\omega, \mathbf{u}_p)$, $p \in \{A, B\}$, then an entry-wise outer product multiplication of $R_A(\omega, \mathbf{u}_A)R_B^*(\omega, \mathbf{u}_B)$, which is then added into the running average double beamforming transform over many time windows, $b(\omega, \mathbf{u}_A, \mathbf{u}_B)$ (which has an additional dimension not pictured here). 25

Workflow, Part	$N_A = N_B$	$N_{up} = N_{\theta p}$	Time	Growth
Traditional Crosscorrelation	9	·	4.33×10^2	·
Traditional Double Beamform	9	2	2.27×10^{-1}	·
	9	4	3.38×10^0	14.8
	9	8	5.46×10^1	16.2
	9	16	8.99×10^2	16.5
New Phase 1	9	2	2.04×10^0	·
	9	4	7.23×10^0	3.5
	9	8	2.93×10^1	4.05
	9	16	1.20×10^2	4.09
New Phase 2	9	2	2.01×10^{-1}	·
	9	4	3.21×10^0	16.0
	9	8	5.36×10^1	16.7
	9	16	7.93×10^2	14.8

Figure 1: This table shows the timing results (in seconds) for each part of the traditional and new algorithms as the number of sensors per patch ($N_A = N_B$) stays constant at 9, and the number of slownesses and angles tested per patch ($N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$) grows. The growth column indicates the timing at a given size problem divided by the timing of the smaller problem size on the previous line. The traditional algorithm's crosscorrelation phase has no dependence on the number of slownesses or angles.

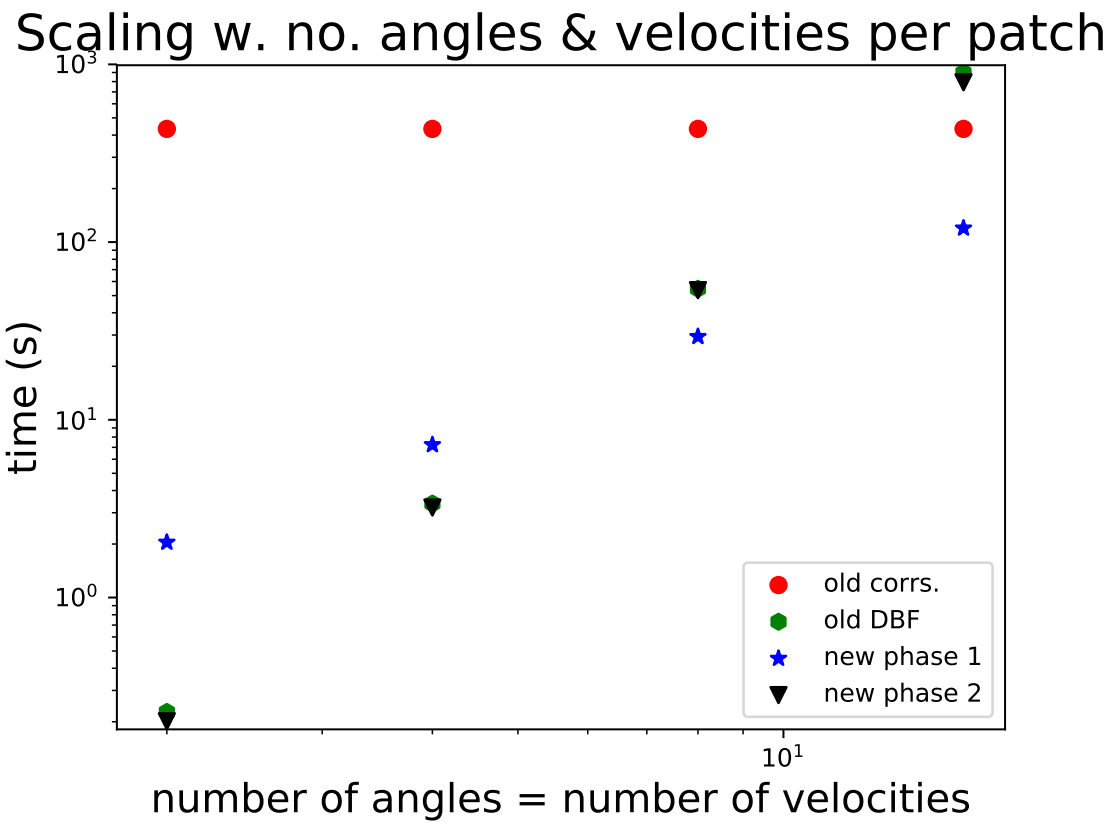

Figure 2: The increase in timing as the number of test slownesses increases is plotted for both the traditional crosscorrelation and double beamforming algorithm (split into its two parts) and the new algorithm (split into its two phases). In this test the number of sensors per array patch was held constant at 9. $N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$ in each test, and this value is doubled for each experiment. The new algorithm runs faster than the old algorithm, and has similar scalability as the number of slownesses grows.

Workflow, Part	$N_A = N_B$	$N_{up} = N_{\theta p}$	Time	Growth
Traditional Crosscorrelation	9	4	4.33×10^2	·
	18	4	1.69×10^3	3.90
	36	4	6.66×10^3	3.94
	72	4	3.03×10^4	4.55
Traditional Double Beamform	9	4	3.38×10^0	·
	18	4	7.78×10^0	2.30
	36	4	1.53×10^1	1.97
	72	4	4.25×10^1	2.78
New Phase 1	9	4	6.76×10^0	·
	18	4	1.27×10^1	1.87
	36	4	2.43×10^1	1.91
	72	4	5.70×10^1	2.34
	144	4	1.03×10^2	1.81
	288	4	2.02×10^2	1.96
	576	4	3.99×10^2	1.96
New Phase 2	9	4	3.56×10^0	·

Figure 3: This table shows the timing results (in seconds) for each part of the traditional and new algorithms as the number of sensors per patch ($N_A = N_B$) grows. The number of slownesses and angles tested per patch ($N_{uA} = N_{\theta A} = N_{uB} = N_{\theta B}$) stays constant in these tests. The growth column indicates the timing at a given size problem divided by the timing of the smaller problem size on the previous line. The new algorithm's phase 2 is independent of the number of sensors, so timing is only reported for $N_A = N_B = 9$.

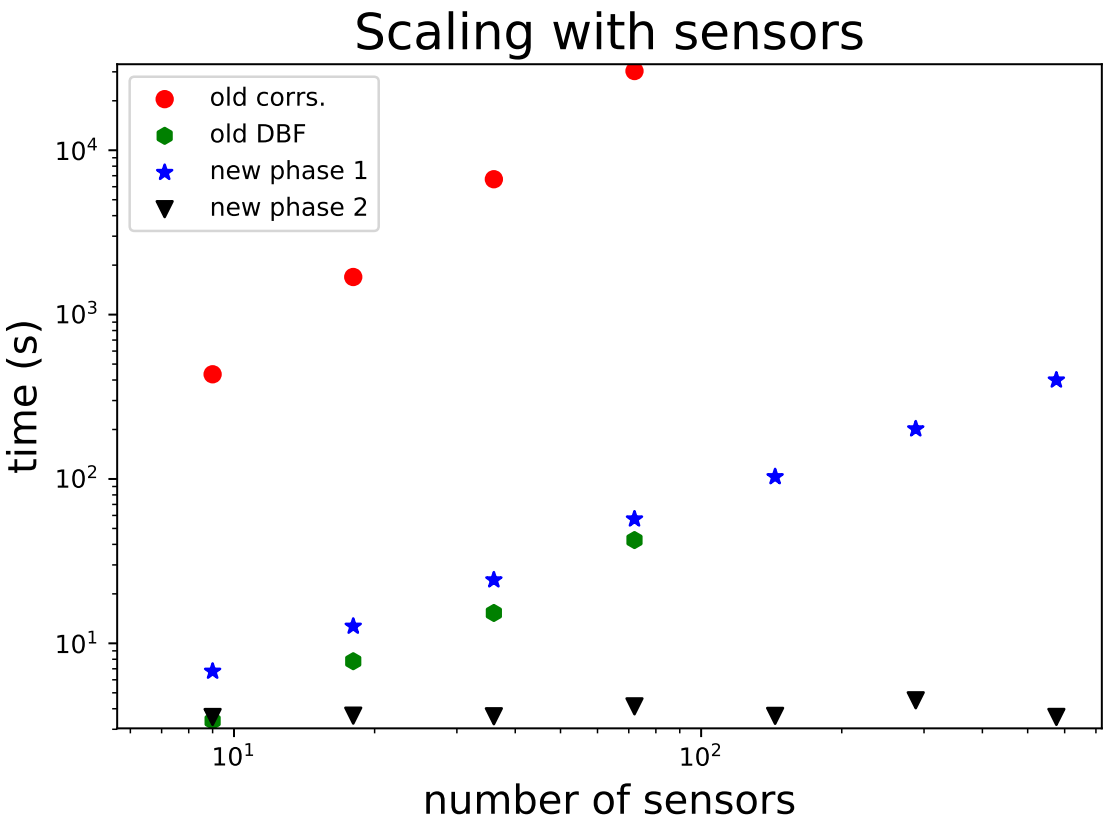

Figure 4: The increase in timing as the number of sensors increases for the traditional crosscorrelation and double beamforming algorithm (split into its two parts) and the new algorithm (split into its two phases) is plotted. The timings show that the new algorithm runs much faster than the traditional algorithm, and is much more scalable (lower slope), primarily driven by the high cost of traditional crosscorrelations. Both the new and old algorithm were run for $N_A = N_B$ equal to 9, 18, 36, 72, and only the new algorithm was run for 144, 288, and 576.

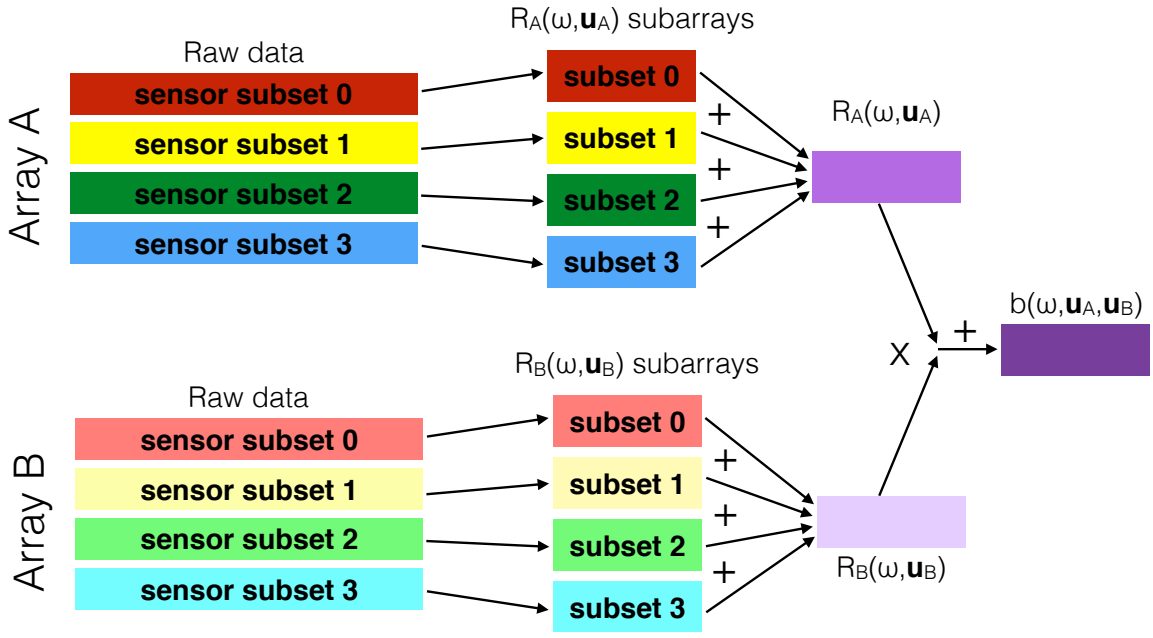

Figure 5: The proposed algorithm includes a first round of trace-by-trace operations which are embarrassingly parallel. The next step is a reduce within each array of phase-shifted data into $R_p(\omega, \mathbf{u}_p)$, $p \in \{A, B\}$, then an entry-wise outer product multiplication of $R_A(\omega, \mathbf{u}_A)R_B^*(\omega, \mathbf{u}_B)$, which is then added into the running average double beamforming transform over many time windows, $b(\omega, \mathbf{u}_A, \mathbf{u}_B)$ (which has an additional dimension not pictured here).