This volume is the property of the University of Oklahoma, but the literary rights of the author are a separate property and must be respected. Passages must not be copied or closely paraphrased without the previous written consent of the author. If the reader obtains any assistance from this volume, he or she must give proper credit in his own work.

I grant the University of Oklahoma Libraries permission to make a copy of my thesis/dissertation upon the request of individuals or libraries. This permission is granted with the understanding that a copy will be provided for research purposes only, and that requestors will be informed of these restrictions.

NAME [redacted]

DATE 8/24/15

A library which borrows this thesis/dissertation for use by its patrons is expected to secure the signature of each user.

This thesis/dissertation by JOSHUA REESE SING YUN HARDISTY has been used by the following persons, whose signatures attest their acceptance of the above restrictions.

NAME AND ADDRESS DATE

Revised 10/11/2012

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/326556

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

# INTERFEROMETRIC ANALYSIS OF AN EXPLORATION SEISMIC SURVEY IN NORTHERN TEXAS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

JOSHUA REESE SING YUN HARDISTY
Norman, Oklahoma
2015

THESIS
AR
OP.3

# INTERFEROMETRIC ANALYSIS OF AN EXPLORATION SEISMIC SURVEY IN NORTHERN TEXAS

A THESIS APPROVED FOR THE
CONOCOPHILLIPS SCHOOL OF GEOLOGY AND GEOPHYSICS

BY [redacted]

Dr. Kurt Marfurt, Chair

[redacted]

Dr. Jamie Rich

[redacted]

Dr. Xiaowei Chen

Geology

© Copyright by JOSHUA REESE SING YUN HARDISTY 2015
All Rights Reserved.

I dedicate this to my Grandma

## Acknowledgements

I would like to thank the countless people who have helped me on my graduate school journey.

I would like to thank Dr. Jamie Rich for introducing me to the geophysics world and advising me on my thesis.

I would like to thank Dr. Marfurt and Dr. Chen for doing me the honor of serving on my committee.

And I would like to thank Mike Burnett and the Tamecat LLC team for providing me with my data.

A special thanks goes to Thang Ha and Brad Wallet for motivation and software support.

Table of Contents

**List of Tables** ..... vii

**List of Figures** ..... viii

**Abstract** ..... xiii

**Introduction** ..... 1

Motivation ..... 4

**Chapter 1: Geologic Background** ..... 6

Data properties ..... 6

Geologic Background ..... 6

Figures ..... 9

**Chapter 2: Passive Source Interferometry** ..... 14

Workflow ..... 17

Data conditioning ..... 17

*Reference trace selection* ..... 17

*Load data and crop* ..... 18

*Resample and detrend* ..... 19

*Band-pass filter* ..... 20

*Normalization* ..... 20

*Whitening* ..... 24

Surface wave computation: applying interferometric equations ..... 24

*Correlation* ..... 25

*Power spectral normalization* ..... 27

*Determining interval length, **T*** ..... 28

*Symmetric-component correlation* ..... 30

Results ..... 30

Figures ..... 33

**Chapter 3: Data Management** ..... 48

Computation analysis ..... 48

*Surface wave computation: correlation* ..... 50

*Data conditioning: normalization* ..... 53

Hardware analysis ..... 55

Figures ..... 60

**Chapter 4: Conclusion** ..... 64

**Chapter 5: Further Work** ..... 66

**References** ..... 67

**Appendix: code** ..... 73

Main program ..... 73

Time Sort ..... 81

Normalization length-to-1 ..... 83

Normalization running-absolute-mean ..... 83

Whiten ..... 84

Cross-correlation ..... 85

Cross-coherence ..... 86

## List of Tables

Table 1. Names given to different stages of computational complexity in Big O analysis ..... 50

Table 2. A comparison of inputs into an interferometric algorithm and the corresponding computational costs. The computational cost is based on hardware specific to this study..... 65

## List of Figures

Figure 1. The extent of the Fort-Worth Basin – Bend Arch total petroleum system in north central Texas. Major structural features are shown. The type of hydrocarbon produced out of each well is plotted (Pollastro,2007).....9

Figure 2. Major geological structures and the extent of hydrocarbon maturation (Montgomery et al., 2007).....10

Figure 3. Paleozoic section showing the depositional topography of the Bend Arch (Galloway et al., 1973).....11

Figure 4. Generalized cross-section of the Fort Worth Basin – Bend Arch petroleum system. The Fort Worth Basin is shown in the middle with the structurally high Bend Arch on the left (Pollastro et al., 2003).....12

Figure 5. Generalized stratigraphic section of the Fort-Worth Basin – Bend Arch petroleum system with annotated source rocks, seal rocks, and producing formations (Pollastro et al., 2003).....13

Figure 6. Thought experiment proposed by Curtis et al. (2006) to describe interferometry.....33

Figure 7. The source on the left is detected by receivers (triangles) A and B at different times, $T_A$ and $T_B$. The correlated trace on the right is the difference in travel time between receiver A and receiver B mirrored around T=0. (Schuster et al., 2004).....33

Figure 8. The workflow to compute an interferogram. The section including "Set up," "Data conditioning," and "Computation" computes the interferogram. The

section titled "Data extraction" is the analysis of the interferogram.....34

Figure 9. Cross-coherence of all traces along line 34 with trace 60 corresponding to the blue star. Using baseline parameters of 4 ms sampling interval, 30 minute interval length, pre-correlation whitening, and bandpass filtering from 0 – 60 Hz (a) without and (b) with interpretation of group velocity of coherent events. (c) Spectrum of a representative correlated trace. Each trace is scaled by the maximum value in each trace.....35

Figure 10. Map view of the survey. Average spacing between receiver lines is 660 ft with a receiver group interval of 165 ft. Line 34 is indicated with an arrow. Trace 55 used in Figure 9 is indicated by a star.....36

Figure 11. Cross-line 60 filtered from 1 – 30 Hz. The surface wave has hyperbolic move-out since the cross-line is offset from the virtual source on cross-line 55. Trace separation is 660ft. The blue line indicates picked surface wave arrivals in the causal portion of the interferogram.....37

Figure 12. Transfer rate over a USB 3.0 cable from RAIDBank5 external hard drive. The values are in number of values transferred per second.....38

Figure 13. Representative 20s data windows at a given receiver. Noise bursts compromise the signal. No filtering, normalization, or whitening was performed.....39

Figure 14. Comparison of the effects of normalization for a zoomed section between 18 and 21 s of windows highlighted in Figure 13.....39

Figure 15. Results of cross-coherence using alternative normalization methods (a) no

normalization, (b) sign bit, (c) 0.5s running window absolute mean, and (d) 0.5s automatic gain control (AGC) based on RMS average prior to correlation. All traces are filtered from 1 – 60 Hz. Arrows indicate a surface wave arrival that can be identified in the running absolute mean and absolute value AGC normalization, but not in the un-normalized or sign bit images.....40

Figure 16. Spectrum of a representative trace (900,000 samples with $\Delta t = 1\ ms$): (a) unfiltered original data, (b) after suppression of spikes due to an unknown noise source, (c) after application of a (0-0-60-120 Hz) band pass filter, and (d) after spectral balancing using Equation 5 and a value of $\varepsilon = 0.03$.....41

Figure 17. Cross-correlation of all traces in line 0 with trace 60 line 0. The inputs are a 4 ms sampling interval, 30 minute interval length, pre-correlation whitening, running absolute mean normalization, and filtering from 0 – 60 Hz. The corresponding frequency spectrum is shown. Each trace is normalized by the maximum value of each trace.....42

Figure 18. Cross-coherence results for interval lengths of (a) 1, (b) 5, (c) 30, and (d) 120 minutes. Data have been filtered from 0 – 60 Hz. Each trace has been normalized by the maximum value in each respective trace except for the interferogram with an interval length of 1 minute, which is scaled by a single value.....43

Figure 19. (a) The symmetric correlation of Figure 9 where causal and anti-causal components are averaged to improve the signal to noise ratio. Each trace is scaled by a single value. (b) The spectrum of the ground roll (highlighted in green) and (c) ambient noise (highlighted in red).....44

Figure 20. Stack of correlated gathers corresponding to lines 28 – 39 forming a super gather that smears the geology but suppresses random noise (a) without and (b) with interpretation. The arrows in (a) point to events with velocities shown in (b).....45

Figure 21. Time slices through the 2-4 Hz filtered component of the data at $t = 0.04$, 0.30, 0.80, and 1.20 s. The surface wave from the virtual source, green star, expands with time and reaches the edges of the survey at about $t = 1.20$ s. Seismic data have been linearly interpolated in the E-W direction to match the resolution in the N-S direction. Data above 4 Hz are spatially aliased in the E-W direction and give poor images.....46

Figure 22. Line A-A' filtered between 2-4 Hz as seen in Figure 21 at 0.30 seconds. The 0.30 s line is marked and shows that the surface wave can be seen for about 30 traces. This corresponds to the length of the surface wave in Figure 21 at 0.30 seconds. The yellow lines indicate the start of the surface waves, generally.....47

Figure 23. Time required to normalize an increasing amount of traces of the same length.....60

Figure 24. The same data as in Figure 23, but each type of normalization time has been normalized by the maximum value of each set of data.....60

Figure 25. Time required to correlate an increasing amount of traces of all the same length. Cross-coherence increases at a faster rate than cross correlation.....61

Figure 26. Time required to compute running absolute mean normalization on 600 traces of 2000 values using a GPU and CPU. The GPU's slope increases at 3

times the rate of the CPU.....61

Figure 27. Figure 26 with data normalized by the cost of the hardware. The GPU's slope increases at 6 times the rate of the CPU.....61

Figure 28. Time required to correlate an increasing amount of traces of all the same length using a CPU and a GPU. The black dashed line is the division of the GPU time by the CPU time. The GPU tends to be 7 times as fast.....61

Figure 29. Figure 28 normalized by the cost of the hardware. GPU tends to be 3 times as fast.....62

## Abstract

The extraction of Earth responses from seismic data without an active source has received more attention in the past decade than ever before. This growth in popularity is primarily due to the increased availability of computing capabilities required to process such data. Interferometry is the most common method of processing passive ambient data. Different methods of interferometric computation are compared in this study and a workflow for the interferogram with the most clarity is presented. Methods of normalization include running absolute mean, sign bit, and an automatic gain control (AGC) based on root mean squared (RMS) average. Interval lengths from 1 minute to 120 minutes are compared, and the differences between cross-correlation and cross-coherence are examined. The final workflow uses running absolute mean normalization, cross-coherence, and a 30 minute interval length. Interferometry often deals with large amounts of data, greater than 17 terabytes in this case. Additionally, Central Processing Units (CPUs) and Graphical Processing Units (GPUs) are both used on each step of the workflow to find the most efficient hardware for each process. I analyzed the time cost associated with steps in interferometric computation and found CPUs operate faster on complicated normalizations and GPUs operate faster on simple normalizations and correlations. The workflow does not change based on these findings.

## Introduction

Modern exploration geophysics requires more efficient and effective ways to extract meaningful data from measurements. The main tool exploration geophysicists use today are elastic vibrations in the Earth. The sources of these vibrations fall into two distinct categories: active sources and passive sources. Active sources include any vibrations that are intentionally created such as dynamite buried underground and large vibroseis trucks. Passive sources include vibrations that are not intentionally created such as earthquakes, highway vibrations, or anything else that might unintentionally cause the Earth to move. Geophones along with accelerometers are used to record the Earth's seismic response. A few decades ago, data collected from geophones were incredibly limited because computer storage had not reached the level needed to hold large seismic datasets. The amount of storage required to house today's seismic data sets can be more than 20 Tb for a single survey. Today's data storage is adequate for these surveys, but simple processing computations have suddenly become monumental challenges. The problem has spread to other areas like processing speed and data transfer rates. These computational steps must be controlled in a way to minimize the time to extract useful information. Interferometry makes use of a computationally inexpensive and mathematical simple operator: cross-correlation. I use interferometry to test data management methods on a 17 Tb passive seismic data set.

Interferometry, or ambient noise cross-correlation, is an effective and increasingly popular way in exploration geophysics to extract surface waves. The extraction of surface waves has applications including removal of surface waves from

active exploration seismic data and near surface shear wave velocity estimates. Seismic interferometry uses correlation and stacking on passive seismic data to approximate the Green's function, or lag time, between two receivers. Interferometry assumes that the signal received from the passive seismic data consist of Earth responses that are indistinguishable from real noise. By cross-correlating and stacking all traces, or data received from a set position on the Earth, with a single trace over a large amount of time (days), the real noise is suppressed and meaningful signals that originally appeared as noise are enhanced. The resulting "interferogram," or image produced from interferometry, looks as if one receiver had been a source recorded at all other receivers. The receiver that looks like a source is called a "virtual source." Deconvolution is used in other types of interferometry, but the source is meant to be suppressed in those applications (Vasconcelos and Snieder, 2008). This study, however, does not want to suppress the source. Once the interferogram is created, it can be analyzed to discover new information. Lin et al. (2013) used interferograms to create a near surface velocity model. Until recently, only surface waves have been found, but Nakata et al. (2015) and Lin et al. (2013) found and extracted body waves from an interferogram. Grechka et al. (2012) uses geometry and data already present in microseismic surveys to find anisotropic parameters. My goal will be to produce the clearest possible interferogram and find the computational cost of creating that interferogram. All applications of interferometry, such as creating near surface shear velocity estimates and estimating anisotropic parameters, rely on the user's ability to distinguish surface waves from noise. Producing the interferogram with the most distinct waves will make the applications of interferometry even more accurate. I hypothesize the interferogram with

the most distinguishable surface waves will become apparent by correlating with cross-coherence, normalizing with running-absolute-mean normalization, and using a longer interval length. A graphics processing unit (GPU) will be most useful in computing correlation and a central processing unit (CPU) will be most effective on running-absolute-mean normalization, but both will be limited due to hardware constraints.

Chapter 1 is a review of the regional geology of the Fort Worth Basin and Bend Arch areas.

Chapter 2 describes passive interferometry and how useful information can be derived from oscillations provided by unknown sources. This chapter is divided into two sections: data conditioning and application of the interferometric equations.

Chapter 3 analyzes computational methods and how to deal with a large dataset. This chapter is split into three sections: computational analysis of data conditioning, computational analysis of the application of interferometric equations, and hardware analysis. The computational analysis section focuses on the elements that are the most computationally rigorous – normalization and correlation for data conditioning and the application of interferometric equations, respectively.

Chapter 4 concludes the study and reviews the outcomes of Chapter 2 and Chapter 3.

Chapter 5 gives suggestions of work that could be done in the future related to interferometry and data management.

## Motivation

The goal of this study is to identify the best process of interferometric computation to improve the separation of signal from noise in an interferogram. All applications of interferometry require the identification and extraction of signals from an interferogram. All of the extraction techniques depend on the ratio of the amplitude of the signal to the amplitude of the noise. Consequently, accurately identifying surface and body waves from interferograms contributes significantly to the extraction of information useful to exploration geophysics such as near surface shear velocity models, reflection imaging, and anisotropic parameter estimation. I will make the applications of interferometry more accurate by creating the workflow to achieve the most distinct waves. Some studies that analyze the applications of interferometry are presented here.

Lin et al. (2013) applied interferometry to a data set from Long Beach, CA, constructing a near surface shear wave velocity model up to 600 m. Lin et al. (2013) started by using interferometry on the dense receiver array to record higher frequency, 0.5 – 4 Hz, Rayleigh waves. Eikonal tomography was then used to compute phase velocity. Next, the surface wave phase velocity was inverted to find shear velocity for that area. The spacing between the receivers was small enough to find the higher mode surface waves or body waves.

Lin et al. (2012) and Nakata et al. (2015) used interferometry to distinguish surface wave and body wave signals in frequencies less than 15 Hz. The waves are identified by their velocities and dominate amplitudes in lower frequency bands. Both studies follow a workflow that is similar to the one used by Lin et al. (2013) to compute

the interferogram. The primary difference is that Nakata et al. (2015) used a slightly different version of correlation called "cross-coherence." While both studies see clear body waves, Nakata et al. (2015) uses correlation, selection filters, and noise suppression filters to further isolate body waves and creates a near surface p-wave velocity estimate.

Grechka et al. (2012) shows how interferometry can have a large impact on microseismic surveys using geophones placed inside of a well as opposed to on the Earth's surface. As microseismic data becomes more common, more geophones are used to record signals from hydraulic fracturing. Microseismic events are located from these signals and the majority of the seismic signal is not used. Grechka et al. (2012) shows how with a little computing power, this noise can be turned into something useful. Grechka et al. (2012) was able to directly measure shear velocity in the horizontal and vertical part of a well and demonstrates how interferometric methods could gather vertical-seismic-profile-like data. The workflow that Grechka et al. (2012) uses is almost identical to the workflow used in this study. Unfortunately, this study does not have any downhole seismic data, but the **Data Management** section of this study touches on some of the computational difficulties.

All of these interferometric applications rely on how well surface and body waves can be extracted, and almost all of the methods rely on a signal to noise ratio (SNR) greater than 10 (Halliday et al., 2010). With this study, I will compare different methods of interferometry and the variables that go into those methods to obtain the interferogram with the most clarity.

# Chapter 1: Geologic Background

## Data properties

For this study, seismic data were provided by Tamecat, LLC and acquired by NodalSeismic, LLC. NodalSeismic, LLC was acquiring seismic during a vibroseis exploration survey on the Bend Arch in North Texas and let their geophones acquire additional seismic data for varying lengths. The geophones gathered data for approximately 10 days while the active source exploration seismic survey was being completed. The survey was a mobile survey capturing the vertical component of the Earth's responses and contains more than 5,000 trace locations. The sampling interval is 1 millisecond (ms). The survey's data size is approximately 17 terabytes (TB).

## Geologic Background

The survey used in this study is located in North Texas on the Bend Arch at the northwest section of the Barnett Shale in the same petroleum system as the Fort Worth Basin as seen in Figure 1. The area is adjacent to the Fort Worth Basin, where oil and gas have been found since the Civil War and has been producing since the early 1900s (Pollastro, 2007). This area, formed by the Ouachita Thrust Belt, is home to the Barnett Shale source rock. Until 1998, conventional wells were producing oil and gas from Ordovician and Permian age formations, but as of 2000, however, the

Mississippian Barnett Shale has become the largest producer of gas in the area due to unconventional drilling and production methods (Pollastro et al., 2007).

The oval shaped Fort Worth Basin elongates north and south and is one of many basins formed by the Ouachita Thrust Belt (Walper, 1982). The Fort Worth Basin's northern end is characterized by the Red River and Muenster arches while its west boundary includes the Bend and Concho arch (Pollastro, 2007). Figure 2 shows the general location of the oil and gas reservoirs relative to major structural formations. The seismic survey in this study was conducted to the west of the Bend arch. Pollastro et al. (2007) also finds common structures in the Fort Worth Basin include major and minor faults, local folds, fractures, and thrust-fold structures.

The uppermost part of the Bend Arch is dominated by inter-bedded limestone and shale. The Flippen Limestone is at the surface in this area as seen in Figure 3. The limestone is interbedded with layers of shale and thins to the east. Additionally, beds of black shale and dark limestone become more common towards the east of the Bend Arch (Galloway et al., 1973). Agnich (1949) finds that the seismic velocities for limestone deposits in west central Texas can range from 8,000 to 16,000 feet per second, or 2.4 to 4.9 kilometers per second. These velocities are consistent with the velocities found later in this study.

Prior to the discovery of oil and gas in the area, the history of the Bend Arch and Fort Worth Basin area was key to determining if the timing was correct for hydrocarbon maturity. This plays an important role in determining economic areas for field development.

This Precambrian interval is overlain by a section called the Ellenburger that extends from the Cambrian to Mississippian times. Precambrian granite and diorite is found beneath the sedimentary section of the Fort Worth Basin and Bend Arch. This section has not produced hydrocarbons (Pollastro et al., 2003). Carbonates were then deposited on top of the Precambrian rock over an area extending across the modern state of Texas. Sea level dropped towards the end of the deposition of the Ellenburger which caused the development of karst features throughout the basin (Pollastro et al., 2003).

The Silurian and Devonian rocks were eroded away and the Barnett Shale was deposited during the Mississippian age. Structurally, the Bend Arch and Fort Worth Basin were formed during the late Mississippian to early Pennsylvanian periods when the Ouachita structural belt thrust onto the North American margin (Pollastro et al., 2003). The Bend Arch was a regional structural high as seen in Figure 4. This minor uplift created erosional surfaces. Clastic rocks with origins from the Ouachita thrust sheets began depositing during this Pennsylvanian stratigraphic section. Rocks consist of mostly sandstones and conglomerates from the Middle and Late Pennsylvanian Age with limestone beds becoming less frequent. Most conventional oil and gas were found in the Pennsylvanian age rock, but oil and gas has also been found from Wolfcampian age sandstones on the Bend Arch (Pollastro et al., 2003). The generalized stratigraphic section in Figure 5 shows the producing areas. The converging plates during this time not only caused the Fort Worth basin to form, but other similar basins including the Black Warrior, Arkoma, Val Verde, and Marfa Basins (Pollastro et al., 2003).

## Figures

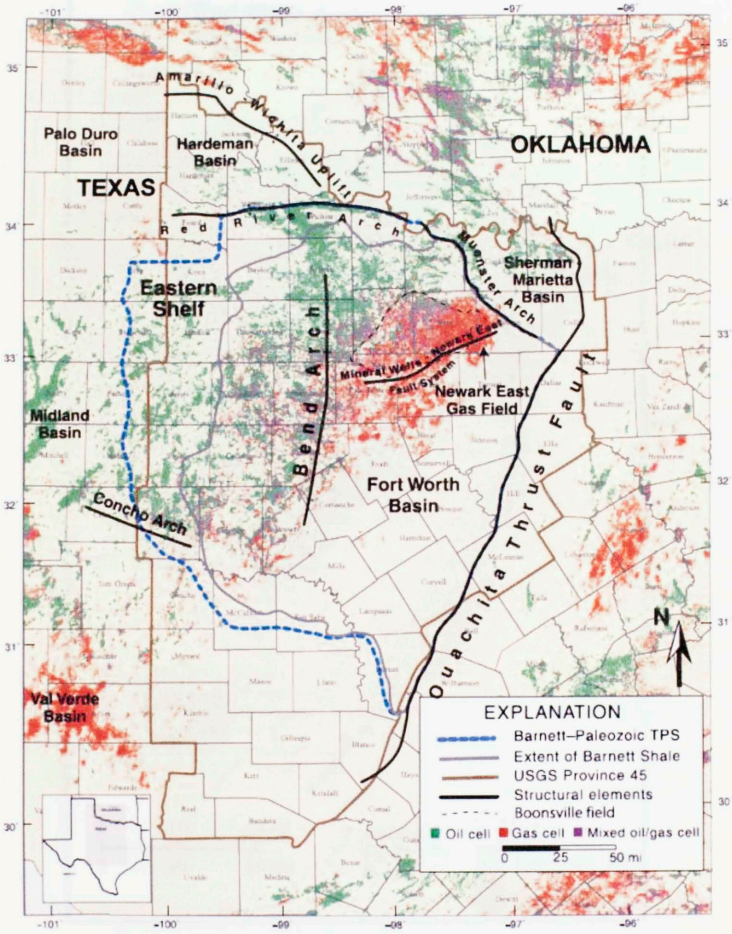



Figure 1. The extent of the Fort-Worth Basin – Bend Arch total petroleum system in north central Texas. Major structural features are shown. The type of hydrocarbon produced out of each well is plotted (Pollastro, 2007).

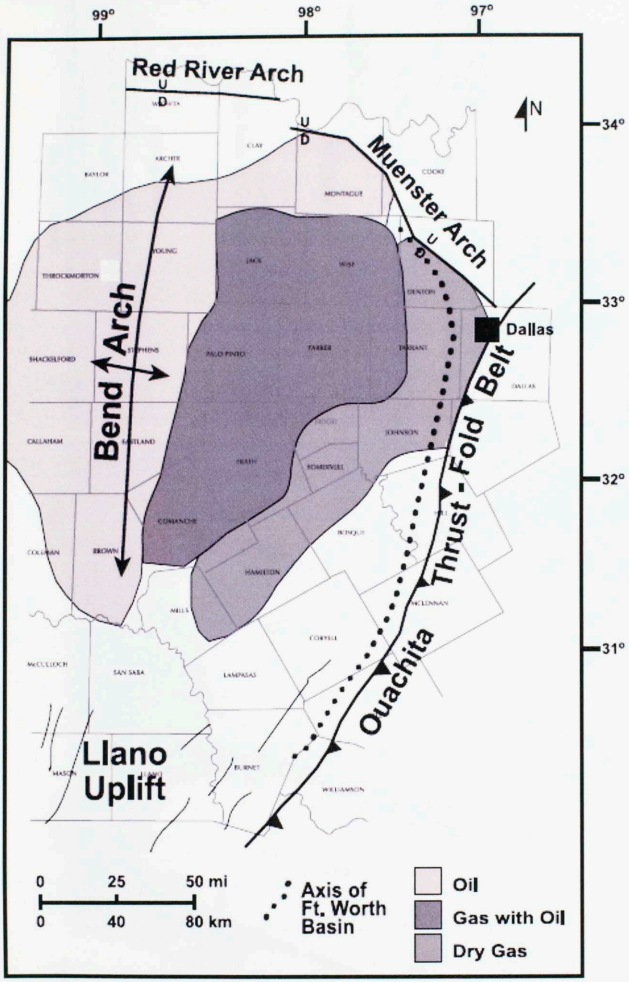



Figure 2. Major geological structures and the extent of hydrocarbon maturation (Montgomery et al., 2007).

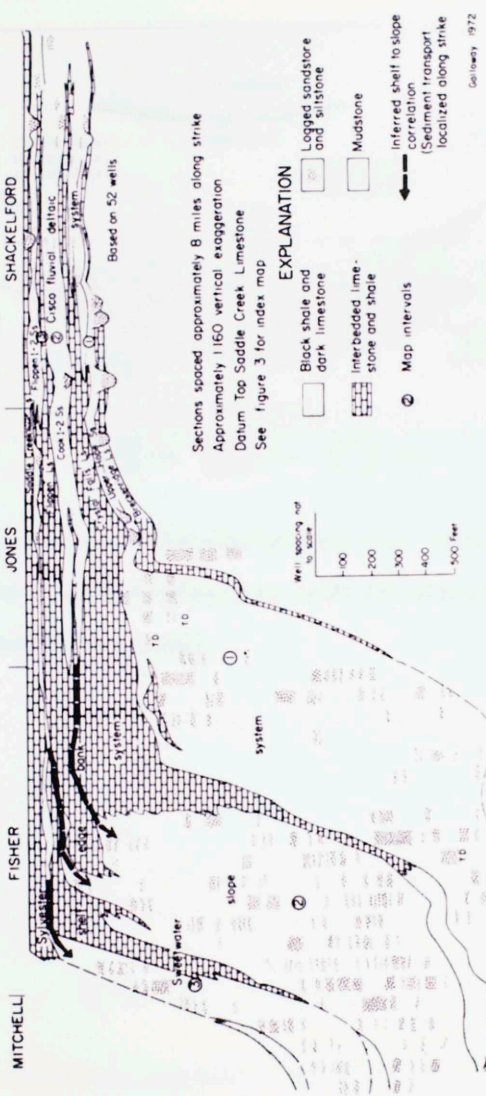



Figure 3. Paleozoic section showing the depositional topography of the Bend Arch (Galloway et al., 1973).

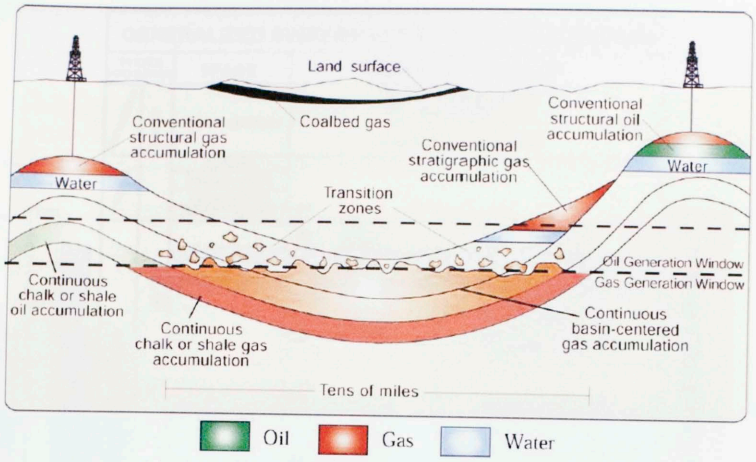



Figure 4. Generalized cross-section of the Fort Worth Basin – Bend Arch petroleum system. The Fort Worth Basin is shown in the middle with the structurally high Bend Arch on the left (Pollastro et al. 2003).

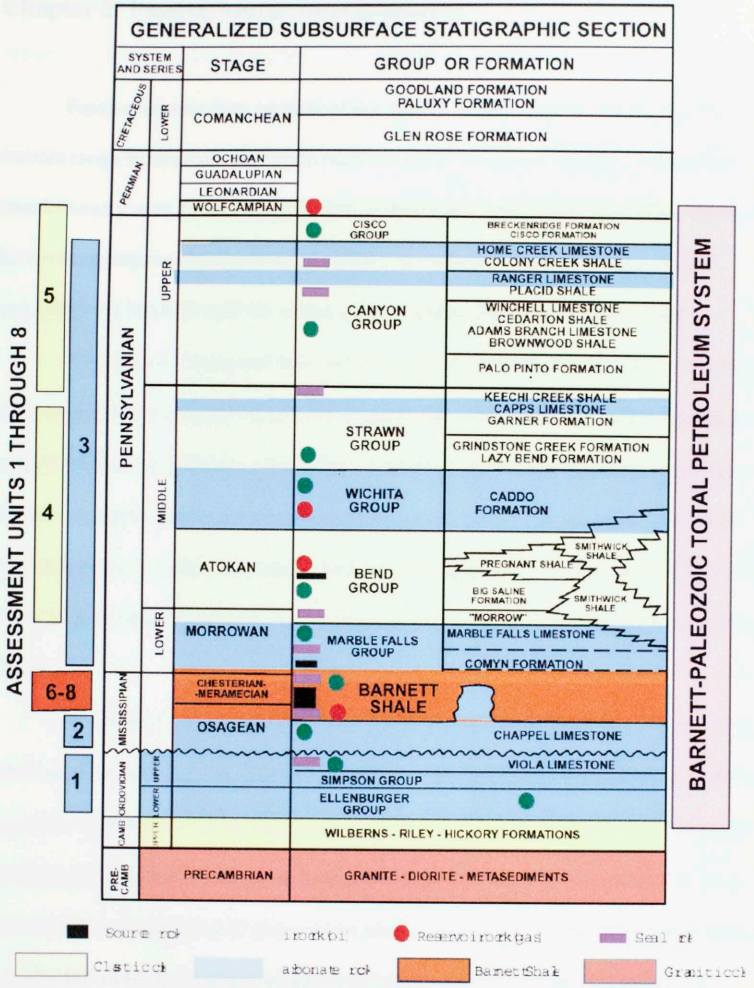



Figure 5. Generalized stratigraphic section of the Fort-Worth Basin – Bend Arch petroleum system with annotated source rocks, seal rocks, and producing formations (Pollastro et al., 2003).

# Chapter 2: Passive Source Interferometry

Passive seismic data consists of the natural vibrations in the Earth. Passive sources range from small amplitude highway noise to large amplitude earthquakes. Interferometry uses passive seismic data to derive the Earth's true responses. By using the method proposed by Curtis et al. (2006), the passive seismic data are cross correlated and stacked until the actual noise is suppressed and a signal is seen.

"Correlation" measures how well two signals relate to one another. "Stacking" is the process of averaging signals occurring in the same location. Cross correlation, a type of correlation, provides a trace that correlates higher when equivalent signals are present. A signal may be identified when traces with low SNR are cross-correlated even though the signal is not seen in either of the input traces. When many traces with a low SNR are stacked at the same location, the identification of a signal is even more likely.

This idea of cross correlating and stacking seems simple enough, but not very intuitive. Consider the "simple thought experiment" proposed by Curtis et al. (2006) in Figure 6. Imagine if two receivers and a reflector are placed randomly and vertically between two impulsive sources as in the left image in Figure 6. The center-left image shows each receiver for each source individually. Source 1 R1 shows a large amplitude initially followed by a second, smaller amplitude. These represent the direct and reflected waves respectively. Source 1 R2 similarly shows a large amplitude followed by a smaller amplitude for the same reasons. The two amplitudes are closer together because R2 is closer to the reflector. Source 2 shows small amplitudes for R1 and R2

because it is showing the transmitted wave through the reflector. The center-right image shows the result of the cross-correlation between the traces for each source. The two traces are then stacked, as shown in the right image of Figure 7, to produce what is called an "interferogram." The stacked image is mirrored on a value of zero which represents "zero" time. Zero time represents the middle of the cross-correlation result. Below zero time is positive time and, by convention, is referred to as the "causal" portion of the cross-correlation. Above zero time is negative time and is referred to as the "acausal" portion of the cross-correlation. The causal portion represents the comparison of a signal travelling from R1 to R2 while the acausal portion represents the signal travelling in the opposite direction.

For two signals, an interferogram shows how well the two signals relate. If the interferogram is analyzed from the perspective of R1, the interferogram shows an image that you would see if R1 was the source and R2 was a receiver. By performing cross-correlation and stacking of multiple receivers with one specific receiver, one is able to create an image that looks as if the chosen receiver is a source. This is shown in Figure 7. The receiver that is a source is called a "virtual source."

A reflector and two receivers in the middle of a medium surrounded by evenly distributed sources will never occur naturally. Claerbout (1968) shows that the reflection coefficients can be retrieved if that situation occurred naturally. The situation where many receivers are "sort of" surrounded by non-uniform sources on top of, or in, a multilayered medium is much more likely. The signals need to be conditioned to fit the assumptions made in Figure 6.

The first assumption is that the receiver must be completely and evenly surrounded by the signal that it is measuring. This is difficult to do because often signals are discrete, vary in amplitude, and include the vibroseis sweep used in this study for conventional exploration geophysics. The vibroseis sweep does not have an equal amplitude in all areas and has a distinct direction of wave propagation. An artificial way to create an evenly distributed signal that seems random would be to normalize the signal in a way that made each signal indistinguishable from each other. The signal would still be there, but the normalization would lessen the effect of it. Another method would be to just remove the part of the trace that contained the known signal. Many methods have been attempted to imitate an evenly distributed signal. These methods will be discussed in **Normalization**.

The second major assumption is that the signals that are received are produced partly by direct arrivals and reflections in the Earth. In other words, the signals are coming from waves bouncing off reflectors and not from noise from instrument sources and random signals. The stacking process should minimize the role of the instrument sources and random signals, but there needs to be a satisfactory amount of real Earth sources. For this study, this means that there needs to be enough traces to be stacked and a long enough trace such that the number of sources occurring in each trace is maximized. These methods will be discussed in **Determining interval length, *T***.

Even with these constraints, Lin et al. (2013), Snieder (2004), Bensen et al. (2007), Halliday et al. (2008), Nakata et al. (2015), Grechka and Zhao (2013), and others have shown that the opportunities for ambient noise cross-correlation are growing every day. This section will describe the theory behind interferometry and

how data can be conditioned to represent a uniform and even signal to extract surface waves. The study generally follows the workflow set by Bensen et al. (2007), Lin et al. (2013), and Nakata et al. (2015).

## Workflow

The workflow created for the study is shown in Figure 8. This workflow was created from an analysis of workflows developed by Bensen et al. (2007), Sneider (2004), Nakata et al. (2011), and Halliday et al. (2008). Before the interferometric equation can be applied the data must be conditioned to fit the assumptions made in the previous sections. The whole process can be broken down into two parts: data conditioning and computation. In data conditioning, the processes include: reference trace selection, band-pass filtering, normalization, and spectral whitening. Computation is just the application of the inteferometric equations which is dominated by correlation.

## Data conditioning

### *Reference trace selection*

A "reference trace" in this study refers to the trace that is cross-correlated with all other traces so that the location of the "reference trace" becomes the location of the virtual source. This can be done for all trace locations. As shown by Lin et al. (2013),

a study that uses the maximum number of trace locations has the smallest amount of error. Usually studies are limited by computing power. Optimization of processes for computing will be discussed in **Data Management**. Figure 9 shows line 34 in line with the reference trace at trace 60. All wiggle plots were created using SeisLab 3.0, a MATLAB Toolbox available on MATLAB CENTRAL's File Exchange library. The blue cones enclose the surface waves. This reference trace was chosen because it is the middle of the survey shown in Figure 10 by the blue star. Figure 11 shows the cross-line at line 55.

*Load data and crop*

A secondary goal of this study was to find the most efficient way of computing the interferogram. In most exploration case, the data loading time expense is small. In interferometry it can be quite large.

In this study, each geophone held the Earth's response at a location in a singular direction for about 10 days. These geophones were spaced approximately 165 ft in the north-south direction and generally 660 ft in the east-west direction. There are exceptions to this. Figure 10 shows the actual spacing of the survey. The files holding the signals were each a little over 3 gigabytes (GB). For reference, each file was greater in size than the stacked seismic volume collected from the vibroseis sweep by a factor of 2. The total size of the passive seismic volume is 17.3 TB. The data was stored on an external hard drive, 20 TB RAIDBank5 by MircroNet, and was transferred by a cable with a USB 3.0 port. The transfer speed varied but averaged around 60

megabytes (MB) per second which is about 4 times as fast as a USB 2.0 connection. Figure 12 shows the length of time required to transfer one ten day trace to MATLAB. The time is measured for 150 ten day traces.

Additionally, the signals that are to be cross-correlated need to occur at the same moment in time. The signals do not start at the same time in their raw form, so they must be cropped. This specific data set had a few more issues than surveys procured for academic purposes because its main purpose was oil and gas exploration. Because of this, the survey was done with a mobile vibroseis sweep. A common way to complete a vibroseis sweep is to have a group of geophones that move across the area being surveyed. In this survey, the center geophones were kept static while groups were moved around the center. The traces needed to be sorted in to groups that occurred at the same time. After this is done, the signals can be cut into interval length sections.

### *Resample and detrend*

The effect of resampling will be discussed in the **Data Management** section, but it is generally used for computational efficiency. The goal of a study determines the extent of resampling. For example, increasing the sampling interval from 1 ms to 5 ms may not significantly affect a study that looks to extract surface waves because surface waves exist at frequencies lower than 50 Hz, and the Nyquist frequency is 500 Hz and 100 Hz for 1 ms and 5 ms, respectively. Nakata et al. (2015) increases his sampling interval from 2 ms to 30 ms because they are analyzing waves below 15 Hz. Schuster et al. (2004) has found that a wave needs to have propagated at least 2 – 3 periods before

the wave can be identified. Consequently, the sampling interval needs to be small enough to accommodate at least 2 – 3 periods of the wave the study is trying to observe.

The standard sampling interval for this study is increased from 1 ms to 4 ms. The sampling intervals that were tested were 1 ms, 4 ms, and 20 ms with Nyquist frequencies of 500 Hz, 125 Hz, and 25 Hz respectively. The total number of data points used in interferometric computation decreases with increasing sampling interval.

### *Band-pass filter*

Band-pass filtering is an important step because it helps isolate the surface waves. Surface waves are identified in interferometry because they have a high SNR and occur in a well-defined frequency range. According to Halliday et al. (2010) surface waves are dominant in the 0 – 30 Hz frequency band. The study done by Halliday et al. (2010) used active seismic data which has a higher frequency range than passive source seismic data. This higher frequency range allows for high mode surface waves to be observed.

In this study, a 1-30 Hz band-pass filter (0-1-20-30 Hz Ormsby filter) was applied to the passive seismic data and tested against broadband data. This choice of frequency range was made to fully cover the possible surface wave modes coming from the unknown sources. All images are broadband unless specified otherwise.

### *Normalization*

Time domain normalization is the most important step in creating a signal that replicates noise. Figure 13 shows the raw seismic windows for ten 20s intervals that are representative of the data at any receiver. Figure 14 shows the effect of different normalization methods on a noise burst in the second window of Figure 13. The assumption that the receiver must be evenly surrounded by equal sources must be satisfied. If this is not satisfied, then large amplitude events will dominate the result. This causes the interferogram to have large, non-physical spurious events that make the surface waves difficult to identify, as seen in Figure 15. Surfaces are more easily identified in 15c and 15d as shown by the highlighted area. The negative effect of normalization is that the amplitude data from the extracted surface waves is lost completely. There are many different approaches to normalization and Bensen et al. (2007) describes a few that he has found to be affective for interferometry.

Bensen et al. (2007) considers five methods to normalize a signal: one-bit, clipping based on RMS amplitude, event detection and removal, running absolute mean, and "water-level" normalization. These methods cover the whole spectrum of severity of normalization. One-bit normalization is the most extreme out of the five because it destroys any amplitude data that might have existed. Running absolute mean normalization gives the user the option to vary the data in a large or small way. Running absolute mean normalization becomes one-bit normalization in its strongest case and doesn't change the data in its weakest case. The one-bit and running absolute mean normalization seem to provide the most meaningful data (Bensen et al., 2007).

One-bit, also referred to as sign bit, normalization assigns a 1 to positive amplitudes and a -1 to negative amplitudes as seen in Equation 1,

$$v_j = \begin{cases} 1 \ if \ u_j > 0 \\ 0 \ if \ u_j = 0, \\ -1 \ if \ u_j < 0 \end{cases} \quad (1)$$

where $u_j$ represents a specific value in a trace and $v_j$ represents the weighted value with index $j$. Changing the amplitude in such a radical way keeps only a sliver of amplitude information, but satisfies the first assumption discussed at the top of page 16. One-bit normalization turns the uneven signal into a signal with even amplitudes throughout the signal. The one-bit normalization method also takes the least amount of computational power. Other methods require the computation of a value to weight the original signal by while one-bit normalization just has to determine the sign of the value. One-bit normalization is the normalization of choice for quick looks at data and data sets with large amounts of data.

The running absolute mean method computes an average of the absolute value of the trace in a certain window and weights each point that the window is centered on. The weight is shown by Bensen et al. (2007) in Equation 2,

$$u_{j\,running\ absolute\ mean} = \frac{1}{2N+1} \sum_{n=-N}^{+N} |u_{j+n}|, \quad (2)$$

where $N$ is the window size. The window is important because it determines how much amplitude information is kept. A window of length 1 ($N = 0$) will give the one-bit normalization answer while an infinite window will give an unaltered trace. A window that is equivalent to half of the maximum period being studied is suggested by Bensen et al. (2007). In this study, the normalization window is 0.5s centered on the value being normalized. Running absolute mean normalization is the most computationally intensive.

The last normalization is an automatic gain control (AGC) based on RMS average. This study will term this type of normalization "length-to-1." This process weights all values in a set window so that the length of the values within the window is equal to one as seen in Equation 3

$$u_{j_{length-to-1}} = [\frac{1}{2N+1}(\sum_{n=-N}^{+N} u^2{}_{j+n})^{\frac{1}{2}}]. \quad (3)$$

The window used in this study is also 0.5s. Like running absolute mean normalization, length-to-1 normalization will become one-bit normalization with a window of 1 (Bensen et al., 2007). This method takes up more time than one-bit normalization but produces a clearer image. Length-to-1 normalization is more time efficient than running absolute mean normalization and produces a similar image.

Figure 14 shows comparisons of length-to-1, one-bit, and running absolute mean normalization. For running absolute mean and length-to-1 normalization, the data are dived by the weights as seen in Equation 4

$$v_j = \frac{u_j}{u_{j_{normalization}}} \quad (4)$$

where $v_j$ is the weighted data. Figure 15 shows the interferogram for one inline for different normalization methods. The problem with one-bit normalization is that it would require a massive amount of correlated traces to stack to produce a clear image. There are two methods of increasing the amount of correlated traces: decreasing interval length and longer recorded signals. Decreasing interval length leads to problems that will be discussed in **Determining interval length, *T*** and the recorded signal length is set for this study.

### *Whitening*

Whitening is a step that is used to further enforce the assumption that data must be evenly surrounded by homogenous sources for the interferometric equation to apply. Bensen et al. (2007) refers to this step as whitening, but other studies might describe it as energy normalization. "Whitening" refers to the background frequency spectrum of the Earth. The Earth's background frequency is assumed to be without any distinct peaks. Therefore, if a signal is recorded and has distinct frequencies, then it must not be completely from the Earth. Whitening is done to make the frequency bandwidth of the data have an even amplitude distribution. This study follows Bensen's et al. (2007) to multiply by the inverse of the smoothed frequency spectrum of a trace. The whitening step is also often "built-in" to the interferometric equations as will be discussed in **Correlation**.

The spectrum of a trace shown in Figure 16 is representative of the spectrum of the entire data set. Noticeable spikes in the spectrum occur at every 25 Hz interval. Manual examination of the time domain data did not reveal any periodic spikes or tape ecryption errors. Therefore, I hypothesize that there is an ambient periodic source with multiple harmonics.

### Surface wave computation: applying interferometric equations

Application of the interferometric equation is the next step in extracting the Green's function. After the cross-correlations are completed for one 10-day trace, the

correlations are stacked. The correlations consist of a two-sided time series representing positive and negative correlation lag times both beginning at the middle value of the correlogram. After stacking, the signal representing the Rayleigh wave emerges.

The signals received from the sources in interferometry are not always transmitted directly from the original source. The recorded signal is often a combination of signals from multiple that have been scattered by anything that would reflect the wave. A collection of multiple scattered waves is called a "coda wave." Additionally, it is likely that these coda waves are scattered multiple times before being recorded and have a small amplitude as a result. The goal of applying interferometric equations is to constructively add the information contained in each of these waves over a large amount of coda waves.

### *Correlation*

Consider two receivers that are located in a medium and separated by a distance, $R$. In the medium, there are also sources that emit a signal, $S_n(t)$, where $n$ is the index of each source. These sources include coda waves (Snieder, 2006). We assume that the velocity of the source signal does not change with time. The signal recorded by the first receiver is labeled $u(x_A, t)$ and the signal recorded by the second receiver is $u(x_B, t)$. The location of each receiver is represented by $\boldsymbol{x_i}$. Receiver at location $A$, $x_A$, is used in this example, and the source location is represented by $x_B$. $|S_n(\omega)|^2$ is the average power spectrum in the frequency domain. The time derivative of the cross-correlation

of $u(x_A, t)$ with $u(x_B, t)$ is equal to the causal and acausal parts of the Green's function between the two receivers (Snieder, 2004):

$$G_{AB}(x_A, x_B, T) + G_{AB}(x_A, x_B, -T) = \frac{d}{dt}[D_{AB}(x_A, x_B, T), C_{AB}(x_A, x_B, T)] \quad (5)$$

$G_{AB}(x_A, x_B, T)$ corresponds to the causal Green's function between receivers $A$ and $B$ for a time interval of length $T$, or interval length. The right-hand side of Equation 5 is the time derivative of the result of the cross-correlation and stack, $D_{AB}(t)$

$$D_{AB}(x_A, x_B, T) = \frac{1}{|S(\omega)|^2}\sum_m u(x_B, t_m) * u(x_A, t_m) \quad (6)$$

The length of the full recording is $t$. The recording is then split into equally sized sections to be correlated of length $T$. The number of correlated sections is $m$, and $t_m$ is a specific interval of the recording, $t$, corresponding to index $m$. If the recording time starts at $t = 0$, then $t_m$ is the time interval from 0 to $T$ seconds. The record length for any index $m$ is represented by $t_m = (m-1)T \; to \; mT$ seconds within length $t$.

Nakata et al. (2011) proposed "power normalized cross-correlation" called "cross-coherence"

$$C_{AB}(x_A, x_B, \omega) = \sum_m \frac{U(x_B, \omega_m)U^*(x_A, \omega_m)}{|U(x_B, \omega_m)||U(x_A, \omega_m)| + \varepsilon < |U(x_B, \omega_m)||U(x_A, \omega_m)| >}, \quad (7)$$

where $C_{AB}$ is the cross-coherence between receivers at $x_A$ and $x_B$ in the frequency domain; $\omega_m$ is the frequency of a specific interval length corresponding to $t_m$; $U$ is the Fourier transform of signal $u$; $U^*$ is the complex conjugate of $U$; $< \cdots >$ denotes the ensemble average; and $\varepsilon$is a regularization parameter. Nakata et al. (2011, 2013) finds that $\varepsilon = 0.01$ is the "smallest value needed to overcome the potential instability of $[C_{AB}]$ introduced by division" in Equation 7.

Figure 9 and Figure 17 show cross-coherence and cross-correlation respectively for the same line of traces. Nakata et al. (2011, 2015) both find cross-coherence to show the clearest surface waves. This study finds that for this area, cross-coherence and cross-correlation are very similar and indistinguishable.

The seismic data for a ten day recording is split up into sections of a pre-determined length. This length is called the "interval length." Each of these interval lengths of data are then cross-correlated with data from another recording and stacked $m$ times. The interval length can be different for each study, but it determines how much data the recording represents. A longer interval length results in an estimate of the Green's function that contains more frequency information and a more similar number of natural sources per interval length. A shorter interval length results in an estimate that has a larger amount of stacking, $m$, but more variability in the amount of sources per interval length. Equations 6 and 7 consist of normal terms and cross-terms (Snieder, 2006). The normal terms are real arrivals and reflections measured by the signal while the cross-terms are not real and can result in a noisy image. Snieder (2004) finds that for an average over the number of sources is taken, the ratio of cross-terms to normal terms decreases by a factor $\sqrt{\frac{2}{nT}}$, where $n$ is the number of natural sources and $T$ is the interval length.

### *Power spectral normalization*

The difference between cross-correlation and cross-coherence is one of spectral normalization. Whitening consists of dividing by the square root of the average of a

power spectrum in the frequency domain (Oppenheim and Verghese, 2010). This is the term, $|S(\omega)|^2$, in Equation 6 and the denominator term, $|U(x_B, \omega_m)||U(x_A, \omega_m)|$, in Equation 7. In exploration seismic, this can be thought of as the difference between "post-stack" and "pre-stack" spectral whitening. For two receivers, any given length of time will contain different sources. Dividing by the source power spectrum normalizes the variability in source distribution when the sources are unknown. Dividing by the average power spectrum of your virtual source can approximate the division by the real source average power spectrum in cross-correlation, $|S(\omega)|^2$. Snieder (2004) shows that if an interval length is used such that the number of sources per trace is approximately equal, then the normalized average power spectrum is 1. According to Snieder et al. (2009), many studies do not divide by the average power spectrum because the average power spectrum is not known. While removal of this division can lead to ringing, this study does not divide by the average power spectrum for cross-correlation.

### *Determining interval length, T*

As the interval length, $T$, increases, the signal to noise ratio also increases. The SNR of the surface waves is important because it determines how well the surface waves can be identified. Figure 18 illustrates how increasing the interval length affects the signal to noise ratio. The standard interferogram is shown in Figure 9 and has an interval length of 30 minutes.

Figure 18 shows that the interferogram is clearer as the interval length is increased. For an interval length of 1 minute, surface waves can be identified visually

from trace -30 to trace 15. For an interval length of 5, 30, and 120, surface waves can be identified from trace -50 to trace 30. The clarity of the interferogram increases slightly from an interval length of 5 minutes to an interval length of 120 minutes. The blue arrow in Figure 18 indicates an area on the 120 minute interval length interferogram where a surface wave can be identified, but identification on the 5 minute interval length interferogram is questionable. The increase in clarity from 30 minutes to 120 minutes is nonexistent. I find that the 120 minute interval length interferogram is not significantly clearer than the 30 minute interferogram because there are not enough stacked traces for a stable result. A larger interval length generally produces a clearer image, but there must be enough traces to stack to adequately reduce random noise. The number of traces from an interval length of 30 minutes to 120 minutes is reduced by a factor of 4 for a set total time.

I chose an interval length of 30 minutes, half of the interval length chosen in the Long Beach study done by Lin et al. (2012) and equivalent to the interval chosen by Nakata et al. (2015). This length of time is long enough for the number of sources in each length to be approximately equal. Each recorded signal is split into blocks of 1,800,000 values before it is cross-correlated.

$$number\ of\ values = \frac{1000\ samples}{second} \times \frac{60\ seconds}{minute} \times \frac{30\ minutes}{interval\ length}$$

$$= 1{,}800{,}000\ values$$

The large interval length of 30 minutes was chosen to reduce the number of cross-terms in the surface wave data (Snieder, 2004). These cross-terms are non-physical quantities that cause the interferogram to look noisy. They can contaminate the result and reduce

the SNR. To further reduce the contamination caused by an uneven source distribution, the symmetric-component correlation can be calculated.

### *Symmetric-component correlation*

Once stacked, the causal and acausal components are averaged to retrieve the "symmetric-component" cross-correlations (Lin et al., 2008). This is a method of suppressing the error due to a non-even source distribution (Lin et al., 2008). Analysis can be done on the raw interferogram before stacking to reveal the direction that the waves were propagating. Different sides of the cross-correlation result represent different directions of wave propagation.

### Results

Ultimately, the best result is determined by the image that gives clear surface waves for the longest distance from the virtual source. I have found that using a smaller sampling interval, a larger interval length, running absolute mean normalization, and cross-coherence produces the result that has the most distinguishable surface waves. Specifically in this study I used, a 4 ms sampling interval, 30 minute interval length, running absolute mean normalization with a 0.5 s window of normalization, and cross-coherence. Figure 20 shows different surface wave modes that have appeared using the previously stated constraints. Figure 21 shows a 2 – 4 Hz surface wave propagating through a volume at different times. Figure 22 shows a north to south line intersecting the volume in Figure 21 at a location offset from the virtual source.

I created Figure 20 finding interferometric gathers, or cross-section of seismic traces, and summing them. The traces are summed based on the distance of the trace from the virtual source in the gather. Figure 20 can't be used for geologic interpretation applications, but it does present a clear image of the surface and body waves. In addition to the strong surface wave amplitudes shown by the blue arrow, there are other wave modes present. Again, an interferogram is mirrored across $t = 0\ s$. At first glance, one sees the three distinct modes exhibiting different velocities. There is also a high velocity wave that is too fast to be a surface wave shown by the green arrow in Figure 20. I interpret this to be a "diving" body wave as described by Nakata et al. (2015). The body wave could not be seen in a single gather like in Figure 9, but it is visible when gathers are stacked. The diving wave is evidence that body waves are present in the interferogram.

Interferograms can also be analyzed to determine the direction of the majority of the sources. If the source distribution were even from all directions, then the interferogram would look symmetric. In Figure 9, this is not the case. The surface waves in the lower left quadrant have stronger amplitudes than the other quadrants. The amplitudes in the upper right quadrant are also stronger than the amplitude in the lower right quadrant. The difference in amplitude occurs because most of the sources are coming in from the direction corresponding to the left: south. When traces are correlated, the secondary trace is cross-correlated with the reference trace. Additionally, the causal portion, or bottom portion, of the interferogram represents the wave heading from the secondary trace to the reference trace for the left side of the interferogram. The greater number of sources from the south causes the amplitudes to

be large in the lower left portion of the interferogram. When the secondary trace is on the right-hand side of the reference trace, the reference trace is more southerly than the secondary trace. One then correlates the secondary trace with reference trace and the stronger amplitudes appear in the acausal, or upper, portion of the interferogram. Evidence of body waves is also seen due to the stronger amplitudes in the upper right and lower left quadrants. I therefore conclude that there are more sources coming from the south. The origin of these sources is most likely from the highway that cuts through a corner of the survey and the machinery from a small industrial area that is also on the southern boundary of the survey.

## Figures

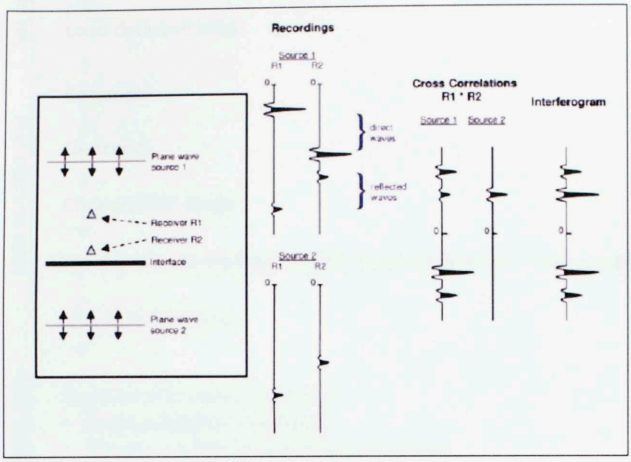



Figure 6. Thought experiment proposed by Curtis et al. (2006) to describe interferometry.

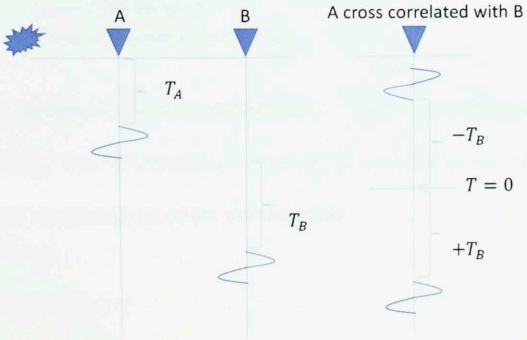



Figure 7. The source on the left is detected by receivers (triangles) A and B at different times, $\boldsymbol{T_A}$ and $\boldsymbol{T_B}$. The correlated trace on the right is the difference in travel time between receiver A and receiver B mirrored around T=0. (Schuster et al., 2004)

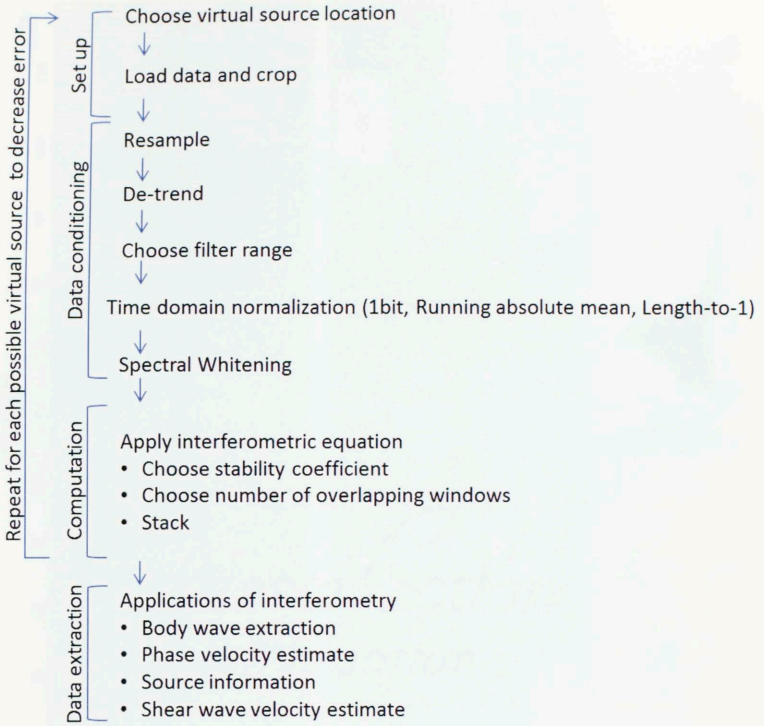



Figure 8. The workflow to perform interferometry. The section including "Set up," "Data conditioning," and "Computation" computes the interferogram. The section titled "Data extraction" is the analysis of the interferogram.

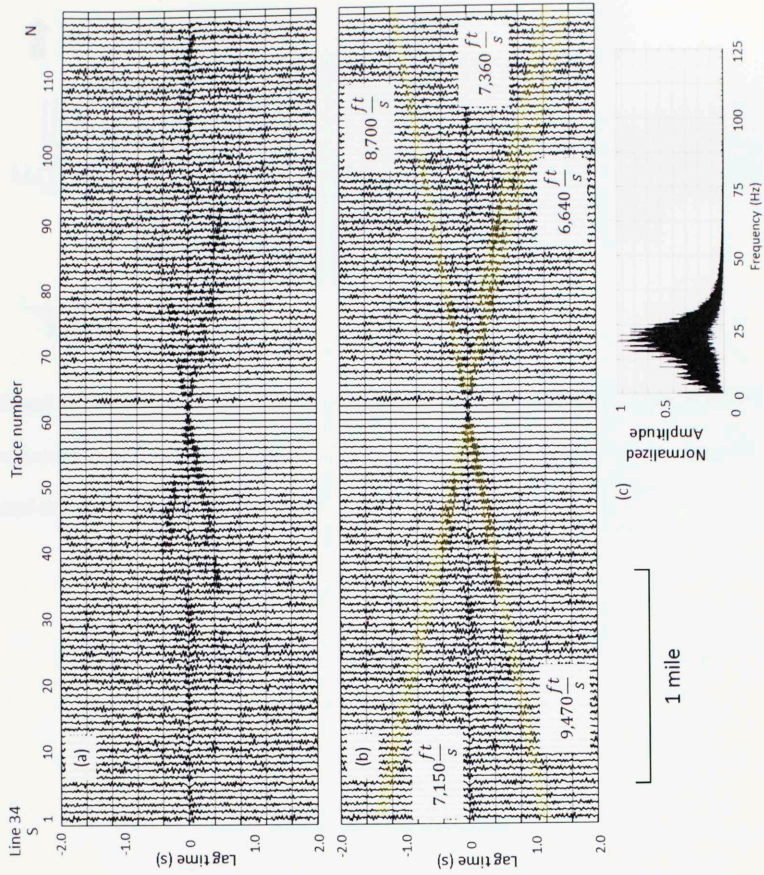



Figure 9. Cross-coherence of all traces along line 34 with trace 60 corresponding to the blue star. Using baseline parameters of 4 ms sampling interval, 30 minute interval length, pre-correlation whitening, and bandpass filtering from 0 – 60 Hz (a) without and (b) with interpretation of group velocity of coherent events. (c) Spectrum of a representative correlated trace. Each trace is scaled by the maximum value in each trace.

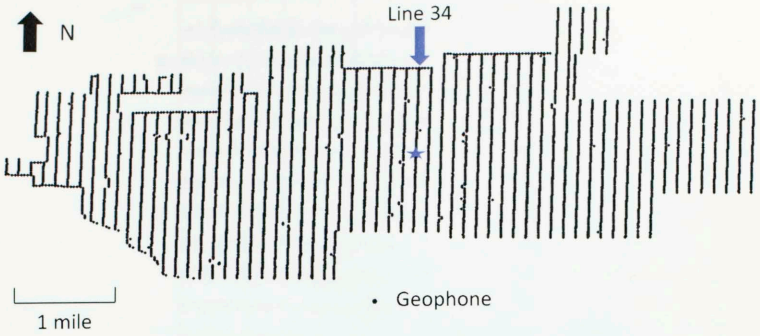



Figure 10. Map view of the survey. Average spacing between receiver lines is 660 ft with a receiver group interval of 165 ft. Line 34 is indicated with an arrow. Trace 55 used in Figure 9 is indicated by a star.

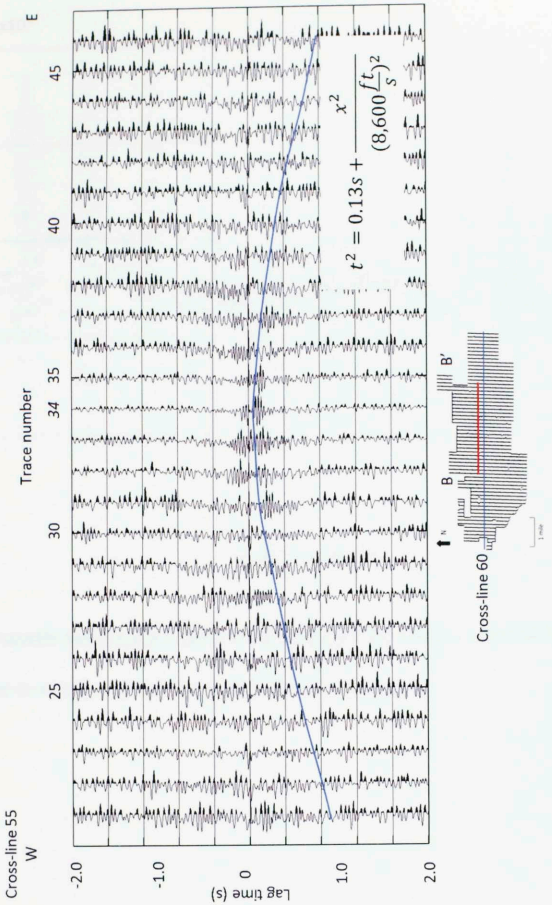



Figure 11. Cross-line 60 filtered from 1 – 30 Hz. The surface wave has hyperbolic move-out since the cross-line is offset from the virtual source on cross-line 55. Trace separation is 660ft. The blue line indicates picked surface wave arrivals in the causal portion of the interferogram. The equation describes the move-out associated with the picked points.

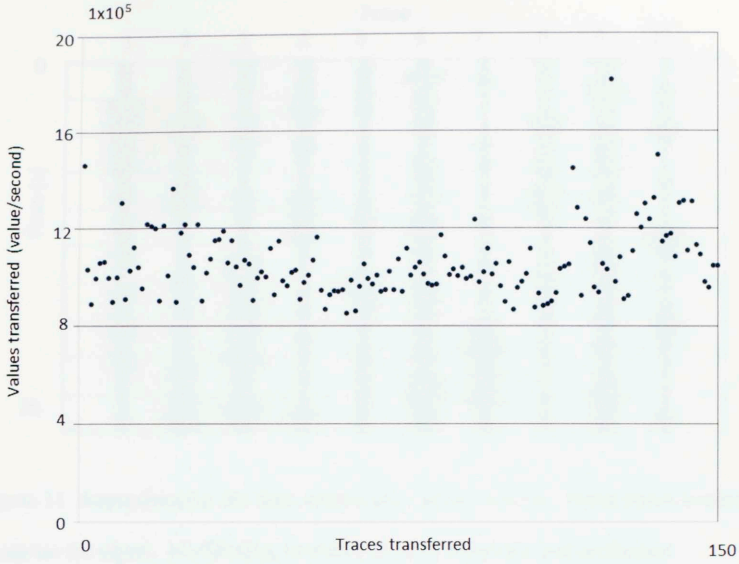



Figure 12. Transfer rate over a USB 3.0 cable from RAIDBank5 external hard drive. The values are in number of values transferred per second.

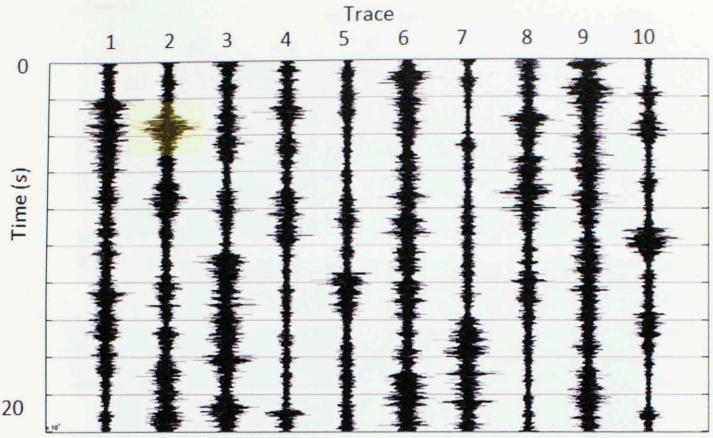



Figure 13. Representative 20s data windows at a given receiver. Noise bursts (yellow) comprise the signal. No filtering, normalization, or whitening was performed.

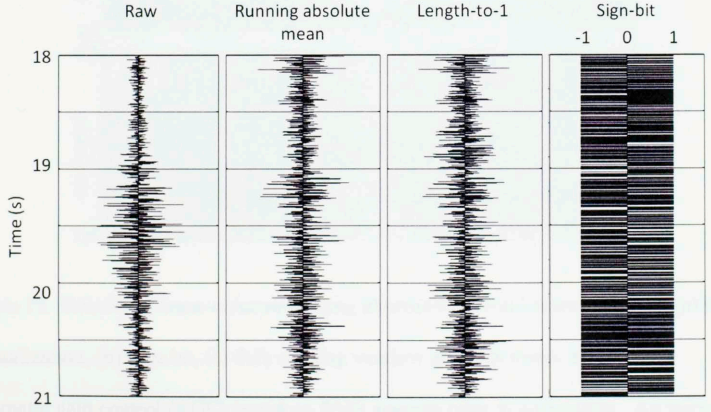



Figure 14. Comparison of the effects of normalization for a zoomed section between 18 and 21 s of windows highlighted in Figure 13.

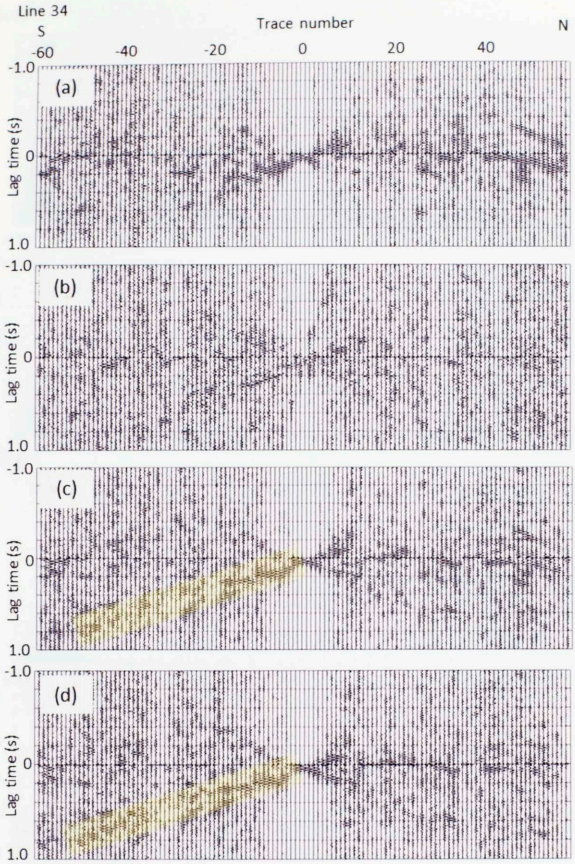



Figure 15. Results of cross-coherence using alternative normalization methods (a) no normalization, (b) sign bit, (c) 0.5s running window absolute mean, and (d) 0.5s automatic gain control (AGC) based on RMS average prior to correlation. All traces are filtered from 1 – 60 Hz. Highlighted areas indicate a surface wave arrival that can be identified in the running absolute mean and absolute value AGC normalization, but not in the un-normalized or sign bit images.

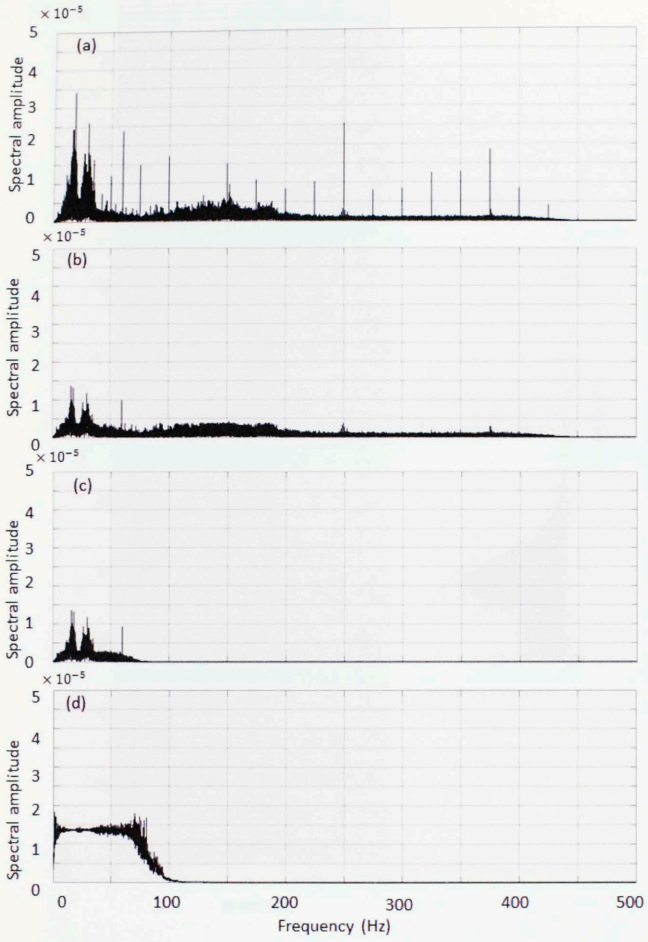



Figure 16. Spectrum of a representative trace (900,000 samples with $\Delta t = 1\ ms$): (a) unfiltered original data, (b) after suppression of spikes due to an unknown noise source, (c) after application of a (0-0-60-120 Hz) band pass filter, and (d) after spectral balancing using Equation 7 and a value of $\varepsilon = 0.03$.

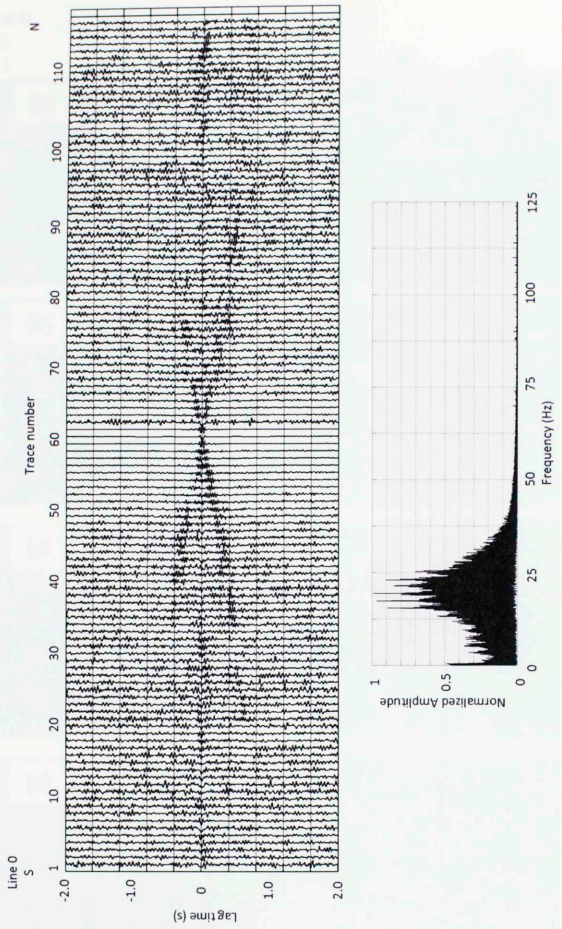



Figure 17. Cross-correlation of all traces in line 0 with trace 60 line 0. The inputs are a 4 ms sampling interval, 30 minute interval length, pre-correlation whitening, running absolute mean normalization, and filtering from 0 – 60 Hz. The corresponding frequency spectrum is shown. Each trace is normalized by the maximum value of each trace.

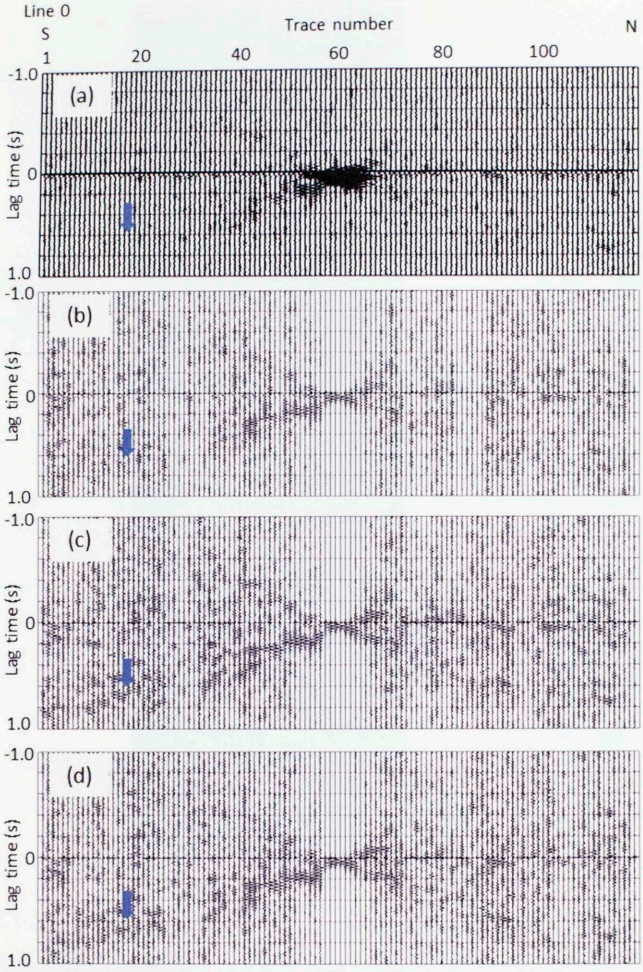



Figure 18. Cross-coherence results for interval lengths of (a) 1, (b) 5, (c) 30, and (d) 120 minutes. Data have been filtered from 0 – 60 Hz. Each trace has been normalized by the maximum value in each respective trace except for the interferogram with an interval length of 1 minute, which is scaled by a single value.

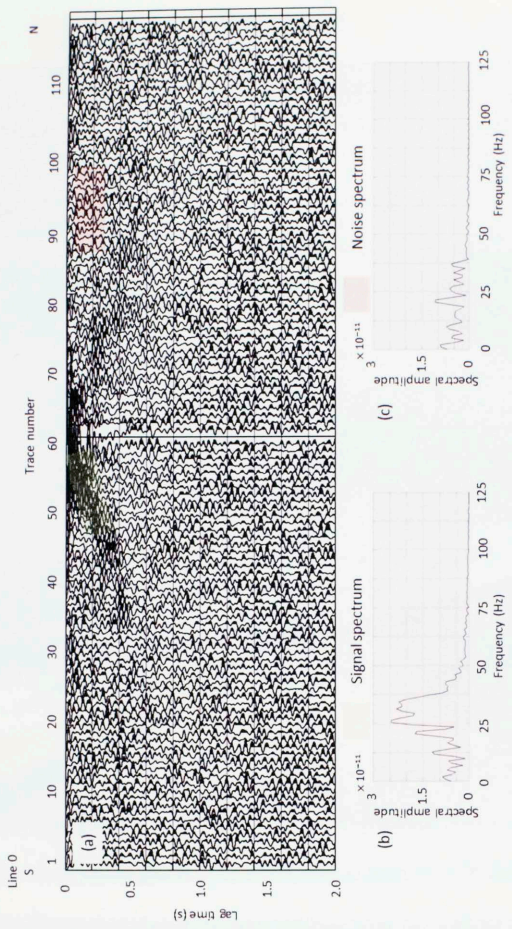



Figure 19. (a) The symmetric correlation of Figure 9 where causal and anti-causal components are averaged to improve the signal to noise ratio. Each trace is scaled by a single value. (b) The spectrum of the ground roll (highlighted in green) and (c) ambient noise (highlighted in red).

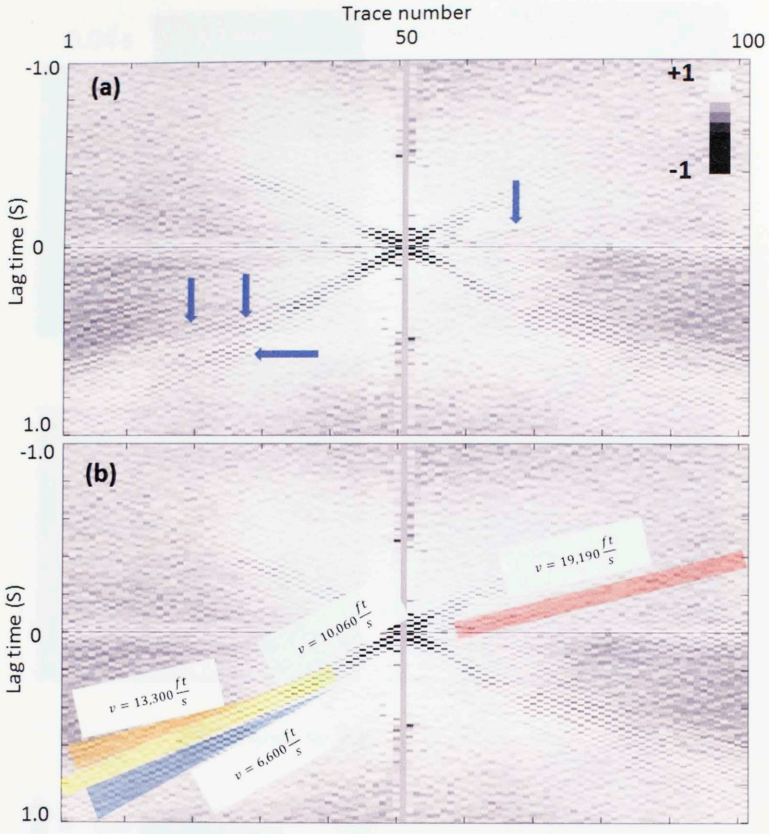



Figure 20. Stack of correlated gathers corresponding to lines 28 – 39 forming a super gather that smears the geology but suppresses random noise (a) without and (b) with interpretation. The arrows in (a) point to events with velocities shown in (b).

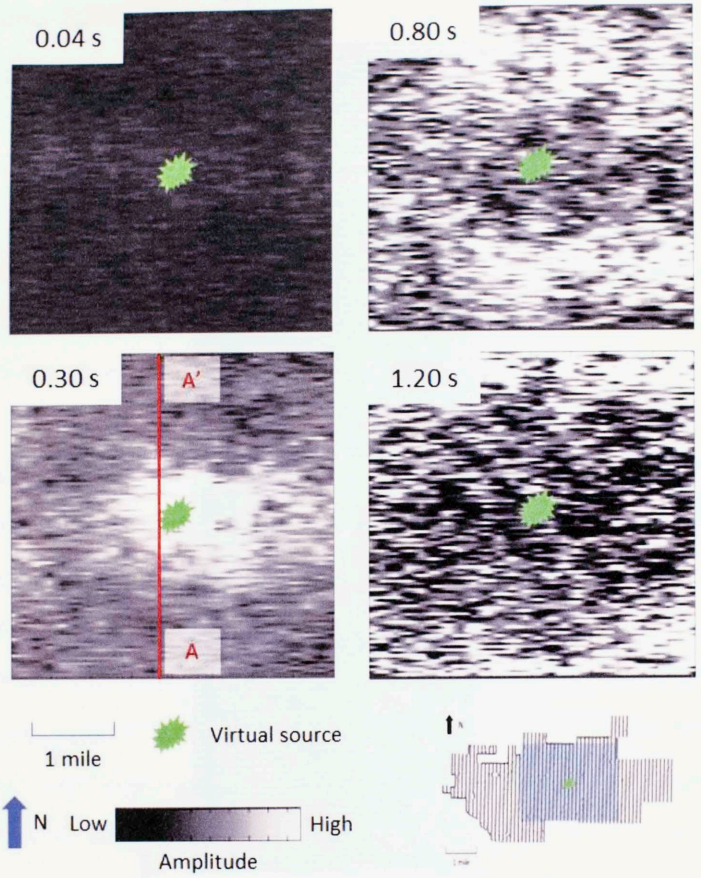



Figure 21. Time slices through the 2-4 Hz filtered component of the data at $t = 0.04$, 0.30, 0.80, and 1.20 s. The surface wave from the virtual source, green star, expands with time and reaches the edges of the survey at about $t = 1.20$ s. Seismic data have been linearly interpolated in the E-W direction to match the resolution in the N-S direction. Data above 4 Hz are spatially aliased in the E-W direction and give poor images.

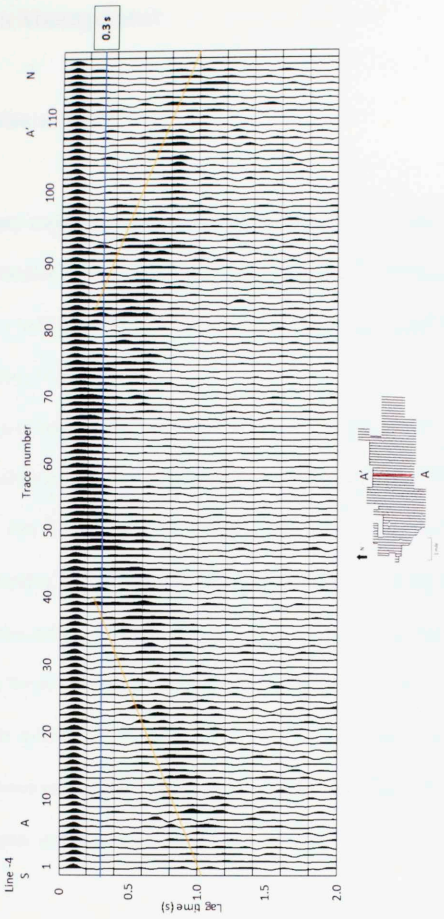



Figure 22. Line A-A' filtered between 2-4 Hz as seen in Figure 21 at 0.30 seconds. The 0.30 s line is marked and shows that the surface wave can be seen for about 30 traces. This corresponds to the length of the surface wave in Figure 21 at 0.30 seconds. The yellow lines indicate the start of the surface waves, generally.

# Chapter 3: Data Management

## Computation analysis

Interferometry turns supposed noise into information. Computing a full interferometric estimation seems to require incredible computing power for a large data set. In this section, I will describe just how much computing power is needed. This will be done using Big O analysis.

The O in Big O notation is not a zero but a capital letter. The symbol is also called Landau's symbol after a German theoretical mathematician, Edmund Landau (Lundqvist, 2003). The O is a reference to the order of the complexity of the algorithm being studied (Lundqvist, 2003). Big O analysis describes the complexity of a computation as it approaches infinity. A decade ago, the 17.3 TBs, or 17,300,000,000,000 bytes, of data that this study examines would have been a term used interchangeably with infinity, but modern surveys would just say that the data set is "large." Big O analysis counts how many times a program "touches," or operates on, this data. For example, given random integers, $n$ and $m$,

$$n + m = x \tag{8}$$

results in a Big O count of 1. If $N$ was a vector of 100 terms and $m$ was an integer,

$$Nm = X \tag{9}$$

results in a Big O count of $N$ where $X$ represents the respective terms of the product of each value in $N$ with $m$. $N$ happens to be 100 in this case. The Big O count is 100 because each of the 100 terms in N is multiplied by the integer, $m$. Luckily, Big O

analysis is characterized by a notation that does not require each "touch" to be documented. Big O analysis seeks to pinpoint the operation that affects the data set the most. It does not take into account each detail. Parker Phinney, founder of Interview Cake, describes Big O math as "awesome, not-boring kind of math where you get to wave your hands through the details and just focus on what's *basically* happening." Big O notation determines what the biggest factor in a computation is, "basically." The Big O notation to describe

$$f(n) = n + 1 \tag{10}$$

is $O(n)$, or

$$f(n) = O(n) \tag{11}$$

where $n$ represents the number of values in a set of data. The 1 is dropped because as $n$ becomes arbitrarily large, the value 1 becomes inconsequential. For

$$f(n) = n^3 + n + 1 \tag{12}$$

the Big O notation is $O(n^3)$, or

$$f(n) = O(n^3). \tag{13}$$

The $n$ and 1 are dropped for the same reasons. As $n$ becomes arbitrarily large, $n^3$ develops so much more quickly than $n$ or 1 that the values become computationally inconsequential. Lundqvist (2003) has listed some of the orders of Big O notation in Table 1 in order of complexity. Notice that Big O analysis measures complexity in relative terms. Again Parker Phinney, personal tutor to many engineers in coding, describes this as "... [expressing] the runtime in terms of – brace yourself – how quickly [complexity] grows relative to the input, as the input gets arbitrarily large."

Until now, researchers have only used Big O analysis to describe the efficiency of computation time. Big O analysis can also be used to characterize other efficiencies including memory usage, disk usage, and network usage.

| Notation | Name |
|---|---|
| $O(1)$ | Constant |
| $O(\log(n))$ | Logarithmic |
| $O((\log(n))^c)$ | Polylogarithmic |
| $O(n)$ | Linear |
| $O(n^2)$ | Quadratic |
| $O(n^c)$ | Polynomial |
| $O(c^n)$ | Exponential |

Table 1. Names given to different stages of computational complexity in Big O analysis

### *Surface wave computation: correlation*

Interferometry is mainly based on how well correlations can be computed. This study looks at two methods of correlation: cross-correlation and cross-coherence. These correlations are computationally similar. The cross-correlation Equation 6 simplifies to Equation 14 for one interval length.

$$D_{AB}(x_A, x_B, t_m) = \frac{1}{|S(\omega)|^2} u(x_B, t_m) * u(x_A, t_m) \quad (14)$$

The length of $u(x_B, t_m)$ is one interval length, $t$. In the discrete time domain cross-correlation is defined by

$$(f * g)(t) \equiv \sum_{\tau=0}^{t_m} f(\tau) g(\tau + t) \tag{15}$$

for $f(t)$ and $g(t)$ are interval length vectors in the time domain and $g(\tau + t)$ is an interval length vector that is shifted over $f$ for all time represented by $f$. Each point in $f$ is multiplied by all points in $g$ and then summed. An analysis of the operations leads to an equation of

$$n \times n + n = n^2 + n = \mathrm{O}(n^2) \tag{16}$$

for $n$ representing the number of each discrete value in one interval length. The cross-correlation equation in the frequency domain is defined by

$$(f * g)(t) \equiv \mathcal{F}^{-1}[F^*(\omega) G(\omega)]. \tag{17}$$

The cross-correlation in the time domain is the product of the complex conjugate of one function with the other in the frequency domain. Each function is Fourier transformed, multiplied together, and the result is Fourier transformed back to the time domain. Analysis of the operations for one interval length leads to an equation of

$$n + n + n = 3n = \mathrm{O}(n) \tag{18}$$

for $n$ representing interval length. The equation for cross-correlation in the frequency domain is often used because its number of computations is $\mathrm{O}(n)$ rather than its time counterpart $\mathrm{O}(n^2)$. The operations analysis equation for Equation 6 is

$$m + m + m + m + k = 4m + 2n + 1 = \mathrm{O}(m), \tag{19}$$

where $k = n + n + 1$ and $m = rn$

where $r$ is the number of interval length traces, the fourth $m$ term is the multiplication of $\frac{1}{|S(\omega)|^2}$, and $k$ is the average RMS amplitude of all the points in the virtual source.

Again, the equation for cross-coherence is

$$C_{AB}(x_A, x_B, t) = \sum_m \frac{U(x_B,\omega_m)U^*(x_A,\omega_m)}{|U(x_B,\omega_m)||U(x_A,\omega_m)|+\epsilon<|U(x_B,\omega_m)||U(x_A,\omega_m)|>}, \quad (5)$$

and the operations equation is

$$m + m + m + m + k + k + 1 = 4m + 4n + 3 = \mathrm{O}(m). \quad (20)$$

The fourth $m$ term in this instance refers to the division by the denominator and $k$ refers to the average RMS amplitude of each of the terms in the denominator. An extra 1 is added for the scalar multiplication of $\epsilon$. Both correlation methods vary linearly, $\mathrm{O}(n)$, with the number of input values.

The cost of computation time varies linearly with the number of values as a result of the equations described previously. This is expected because the two equations differ by only a sum in the denominator of cross-coherence. The cost of computation dramatically decreases when correlation is computed in the frequency domain. Figure 25 shows the increase in time with increased input values for cross-coherence. The increase in time is due to the difference in Equation 19 and Equation 20. This computational difference is the division by the denominator in Equation 7.

The computational efficiency of correlation is necessary to compute an interferogram in a reasonable amount of time. The correlation is computed in the frequency domain because correlation increase in time is quadratic and the correlation increase in frequency is linear, as shown in Equation 16 and Equation 18. The difference in correlation methods by Big O standards is negligible as shown by

Equation 19 and Equation 20. The quality gained from using cross-coherence is worth the computational time that is used.

*Data conditioning: normalization*

Normalizing a data set in the time domain with a large amount of values can also be computationally intensive. In this study I test 3 different types of normalization: one-bit, running-absolute-mean, and length-to-1.

One-bit normalization is the least computationally intensive because it only requires a few steps to compute. The first step is to determine whether a value is positive, negative, or zero. This can take a maximum of three "touches" per value by asking the following questions. Is the value positive? Is the value negative? Is the value zero? The second step would be to replace the value with its corresponding one-bit value, 1, -1, or 0. This represents one interaction. The largest computation equation is

$$m + m + m + m = 4m = \mathrm{O}(m) \tag{21}$$

where $m$ is the number of input values ($m = rn$). One-bit normalization also varies linearly with the number of input values. Figure 23 and 24 show the increase in time with the number of input values for one-bit normalization. As expected, normalization time increases linearly with an increasing number of input values.

Running-absolute-mean is the most computationally intensive out of all the normalizations. Running-absolute-mean normalization divides each value by the local RMS amplitude. The "local" RMS amplitude refers to the RMS amplitude in a user defined window. The operations equation is given by

$$m + w(m + m) = (2w + 1)m = O(m) \quad (22)$$

Where the first $m$ is the division of all values by the local RMS amplitude, $w$ is the length of the local window, and $(m + m)$ represents the RMS amplitude of all values. Even though this is an $O(m)$ type of equation, $w$ can be large depending on how much amplitude information the user wants to retain. Figure 23 and Figure 24 show the increase in time with the number of input values for running-absolute-mean normalization. This increase is also linear. However, the effects of $w$ can clearly be seen in Figure 23. The running-absolute-mean normalization time is orders of magnitude greater than one-bit and length-to-1 normalization. I would suggest using length-to-1 normalization because it provides a similar quality image but does not require such a long normalization period.

Length-to-1 normalization is less computationally intensive than running-absolute-mean normalization but takes more time to compute than one-bit normalization. Length-to-1 normalization weights the values in a user defined window such that the square root of the square of each value in that window is equal to 1. Each value in the window is weighted accordingly. The operations equation is given by

$$m + (m + m) = 3m = O(m) \quad (23)$$

Where the first $m$ represents the division by the weight and $(m + m)$ represents the computation of the RMS amplitude of all values. Again, linearity can be seen in Figure 23 and Figure 24 and efficiency compared to running-absolute-mean normalization can be clearly seen in Figure 24.

Normalization can be as computationally intensive as correlation, so great care must be taken when setting up the interferometric workflow. Running-absolute-mean

normalization may give the best result, but it is marginally better than the length-to-1 method. It also has a computational cost that one might not be willing to pay depending on his or her available resources. One-bit normalization is the fastest method, but produces the worst results. In the future, it may be worthwhile to try one-bit normalization on a data set that has a total length of greater than 10 days, the length of this survey, with an interval length of 30 minutes, the interval length in this survey. I prefer one-bit normalization because it completely takes care of the assumption of an evenly distributed signal, but I think there is not enough data to produce an image similar to the images produced by length-to-1 or running absolute mean normalization. Increasing the total length of the data to over 10 days might allow enough data for one-bit normalization to be effective.

## Hardware analysis

Another factor in the discussion of data management is the tools that are used for computation. There are two popular tools that are used for computation: computer processing units (CPU) and graphics processing units (GPU). CPUs are faster and have more varied applications than GPUs. GPUs are slower but have orders of magnitudes more processors. GPU processors aren't designed for varied applications.

GPUs tend to work well when computing data that does not rely on previously computed data. For example, if 10,000,000,000 different numbers were multiplied by 2, a GPU should be able to perform it quicker than a CPU. A disadvantage of using a GPU is that data needs to be transferred to and from a GPU, so that time needs to be

taken into consideration. A GPU works well for simple calculations, but the transfer speed to the GPU needs to be taken into account.

CPUs tend to work well for data that builds on previously calculated data. This is because a CPU's clockspeed, or speed of a single processor, is much faster than a GPU. For example, a CPU would be used over a GPU if one had a 1e10 array of numbers, wanted to start at the first number, and then add or subtract based on the average up to that point, then a CPU would be used. Generally, a more complicated process would be better suited for a GPU.

Cross-correlation and running-absolute-mean normalization was run on a GPU and CPU. Cross-correlation was chosen because the algorithm is built in a way that favors the GPU. Running-absolute-mean's algorithm is built in a way that would benefit a CPU. MATLAB's built in GPU function was used to transfer data to the GPU and MATLAB's parallel computing functions were used to access all computing cores on the CPUs. A NVIDIA Tesla C2075 workstation card was the GPU used for computation. The CPU used for computation was 2 quad-core Intel Xeon E5-2643 processors. The Tesla GPU costs $ 1399.99 (NVIDIA by Amazon.com) and a single Xeon processors costs $1249.95 (newegg.com). Figure 26 and Figure 28 are the raw result of normalization and correlation, respectively. Figure 25 and Figure 29 are divided by the cost of each piece of hardware to normalize the data. Since the CPU and GPU may be of different quality, the normalization of the data is an attempt to reduce the difference in quality.

The comparison of the CPU and GPU for normalization in Figure 27 shows a slightly unexpected result. Normalization favors CPUs because its algorithm iterates

depending on the previous results. However, the CPU is much slower than the GPU, but the rate of increase in computational time is much smaller. The time required for computation for the GPU grows at 6 times the rate of the CPU. This means that for a smaller data set, the GPU would be optimized for computation, but for a larger data set, the CPU would be the best choice. The situation where the GPU is faster is almost not worth discussing because the time difference between the CPU and GPU for a small data set is still small. With current technology, the CPU is a better choice when computing an algorithm that is more complicated.

The comparison of the CPU and GPU in Figure 29 is expected. Correlation favors the GPU because the algorithm doesn't depend on any computations before or after it. As the amount of data increases, the GPU is faster than the CPU by approximately a consistent factor of 3. For simple calculations like the fast Fourier transforms used in correlation, the GPU would be ideal.

The computational hardware is not the only hardware involved in time management. Memory management was a main contributor to data management but was not studied in this analysis because of resource constraints. Additionally, MATLAB was chosen as the platform to execute the computations, and MATLAB does not manage its memory as well as a compiled programming language might, such as Fortran. Even though this study does not cover memory management, here is another thought experiment. I present a few facts:

- The external hard drive that stored the 17 TB of passive seismic was transferred to my computer via a USB 3.0 connecting cable and ports (each 3 GB file transfer takes 30 second on average)

- MATLAB could only compute interferograms of 2 passive seismic traces at a time with my 32 GB of RAM (the reference trace and one other trace)
- A full interferometric analysis would require the correlation of all of the passive seismic traces to each other
- There are a little over 5000 traces

The number of inter-receiver correlations is $5000 \times (5000 - 1) \cong 25{,}000{,}000$. For this study, that would require 2.5e7 transfers between the external hard drive and the computer because MATLAB could only hold 2 passive seismic traces at a time. That is 23.8 years at 30 seconds per transfer, and 23.8 years is too long for any study. A larger amount of RAM would significantly reduce this time. The factor increase of RAM would decrease the computation by more than that factor squared and would be related to the number of passive seismic traces held. For example, if RAM is increased by a factor of 4 for this study I am able to hold 8 traces at once in MATLAB. I am able to compute inter-receiver correlations between all traces, or 28 correlations. This translates to a saving of 7 transfers of data per trace. Another method of time reduction would be to use a different computational platform and a faster transfer speed. The issue of transfer speed may be solved for by the addition of solid state drives (SSD). I like to think that essentially infinite computing power is available due to the proliferation of computer processors, but when computing power grows, other variables like RAM need to grow with it.

There are many factors to consider when managing a large dataset. GPUs and CPUs have their own areas of computational superiority. CPUs were faster for specific types of normalization and GPUs were faster for correlation. Lastly, variables like data

transfer speed may not be something one immediately thinks about when deciding the feasibility of a computation. Mapping out the flow of data is crucial to minimizing these variable computational "hang-ups."

## Figures

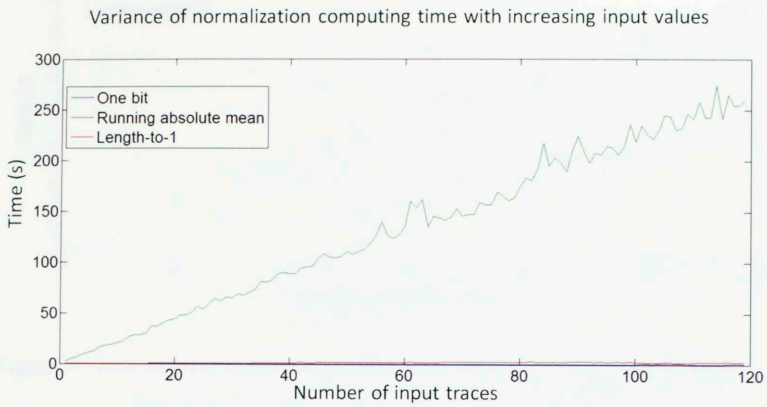



Figure 23. Time required to normalize an increasing amount of traces of the same length.

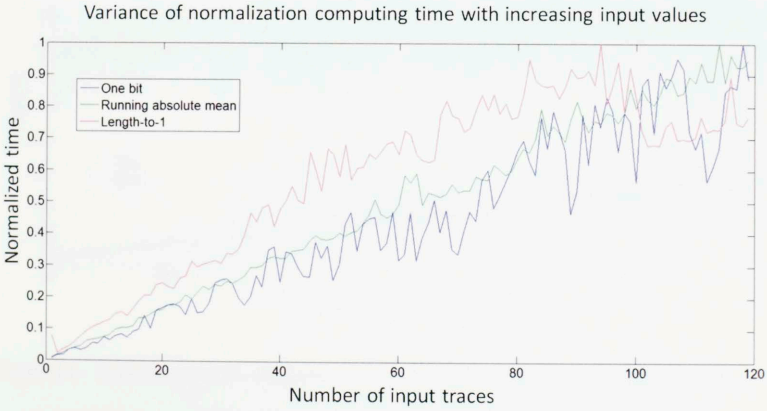



Figure 24. The same data as in Figure 23, but each type of normalization time has been normalized by the maximum value of each set of data.

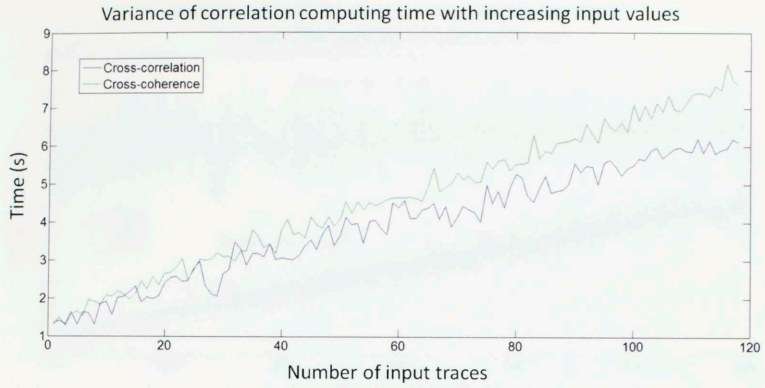



Figure 25. Time required to correlate an increasing amount of traces of all the same length. Cross-coherence increases at a faster rate than cross-correlation.

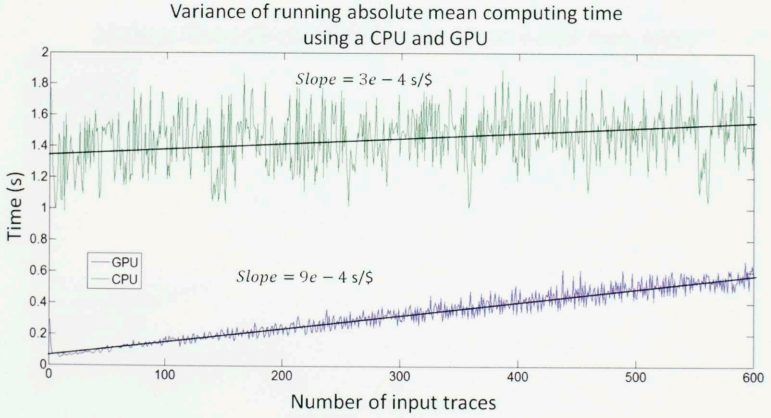



Figure 26. Time required to compute running absolute mean normalization on 600 traces of 2000 values using a GPU and CPU. The GPU's slope increases at 3 times the rate of the CPU.

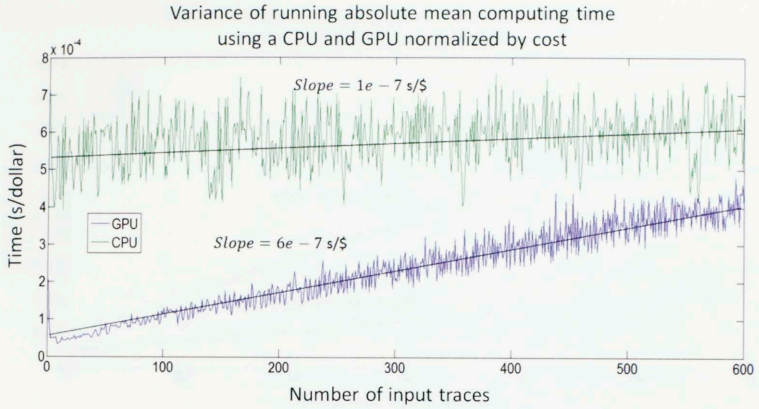



Figure 27. Figure 26 with data normalized by the cost of the hardware. The GPU's slope increases at 6 times the rate of the CPU.

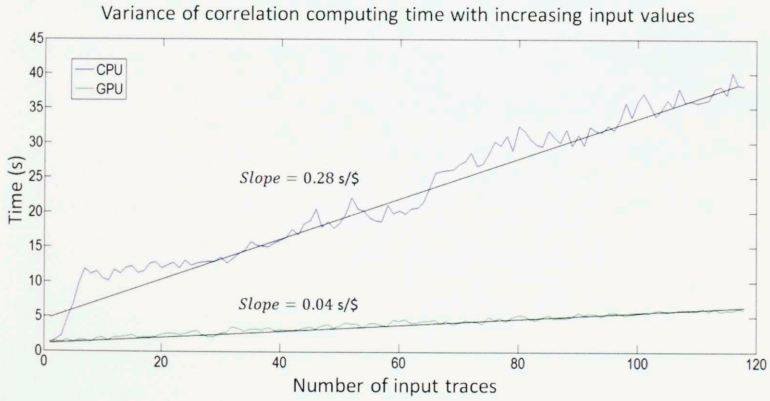



Figure 28. Time required to correlate an increasing amount of traces of all the same length using a CPU and a GPU. The black dashed line is the division of the GPU time by the CPU time. The GPU tends to be 7 times as fast.

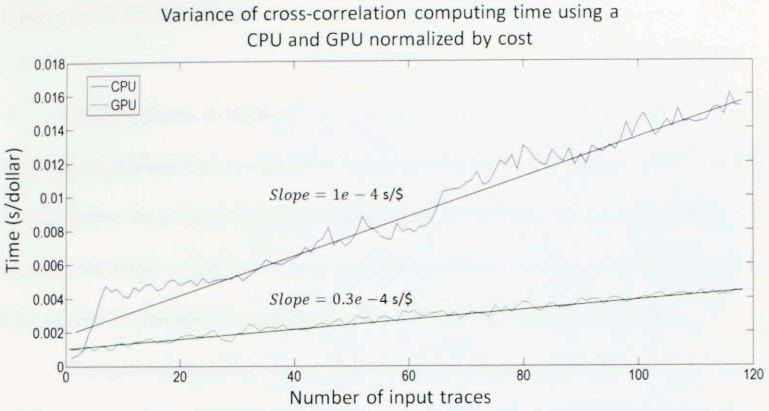



Figure 29. Figure 28 normalized by the cost of the hardware. GPU tends to be 3 times as fast.

# Chapter 4: Conclusion

Interferometry, or ambient noise cross-correlation, is an effective and increasingly popular way in exploration geophysics to estimate surface waves. Seismic interferometry uses cross-correlation and stacking of ambient "noise" seismic data to approximate Green's function, or lag time, between two receivers. Interferometry has been around for decades but progress in the field has only recently exploded.

The goal of this thesis was to find the cost of obtaining the clearest interferogram. There are two sub-goals in this problem: obtain the clearest interferogram and determine the cost of doing so. I hypothesized that this would come from using cross-coherence, having a generally longer interval length, and normalizing with running absolute mean normalization. The processes that make the most difference are the choice of interval length and normalization method. An interval length of 5 minutes will provide an adequate result, but an interval length of 30 minutes will provide the best result. Running absolute mean normalization and length-to-1 normalization provide similar results that are both clearer than interferograms produced from one-bit normalization. This study did not find a significant difference between cross-coherence and cross-correlation, but cross-coherence is preferred because of the addition of power spectral normalization with only a slight increase in computational cost.

Data management is always a key factor in determining interferometric inputs. Correlation in the frequency domain is ideal because computation time increases linearly with the number of time samples. In fact, the computation time of all processes

increases linearly with the number of time samples. Even though the increase is linear, interferometry still requires massive amounts of computational power. The hardware choices made when finding an interferometric result is almost as important as the algorithm. GPUs perform more efficiently than CPUs for correlation, and CPUs perform faster when more complex algorithms are involved, such as running absolute mean normalization. Data storage and transfer can also influence the speed of the interferometric computation just as much as the choice of computational processor. All of these variables must be considered when performing interferometry. In the future, we may find that interferometry's usefulness is limited in exploration geophysics, but it has come a long way and I believe it has a long way to go.

| | Sampling interval (ms) | Interval Length (min) | Number of virtual sources | Normalization | Correlation algorithm | Computation time (s) per trace |
|---|---|---|---|---|---|---|
| **Adequate** | 4 | 30 | 20 | Length-to-1 | Cross-coherence | 220 |
| **Best** | 1 | 30 | All (~5000) | Running-absolute-mean | Cross-coherence | 920 |

Table 2. A comparison of the adequate and best choices of inputs into the interferometric workflow to obtain the most distinct surface waves.

# Chapter 5: Further Work

There a few steps that would improve the results from interferometry. Some of these steps are related to the workflow and some are hardware improvements.

I have done the primary work to improve the workflow, but there are still a few things that can be done to improve it. The first and most obvious next step would be to continue the workflow and compute all inter-receiver signals. Secondly, I suggest building a normalization algorithm that detects the frequency range of anomalous events and builds weights using the running-absolute-mean normalization method. Then combine these weights with the weights from other events, and apply the combination of these weights to the raw data. Currently, the method of normalization lacks the adaptive flexibility associated with a variety of events. Building this new normalization algorithm allows for normalization of events and doesn't affect other data.

Improvements in hardware can easily solve many time management problems, but the latest hardware is not always available. My first suggestion is to compute interferograms on a computer with much more computing power. This will require a computer with a greater number of processing cores and CUDA cores. My second suggestion is to find a faster data transfer method than a USB 3.0 port and cable. A solution to this may be the addition of an SSD and the use of a different programming platform.

## References

Agnich, F. J., 1949, Geophysical exploration for limestone reefs: Geophysics, **14**, 486 500, doi: 10.1190/1.1437554.

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: Geophysical Journal International, **169**, 1239–1260, doi: 10.1111/j.1365-246X.2007.03374.x.

Claerbout, J. F., 1968, Synthesis of a layered medium from its acoustic transmission response: Geophysics, **33**, 264–269, doi: 10.1190/1.1439927.

Draganov, D., X. Campman, J. Thorbecke, A. Verdel, and K. Wapenaar, 2009, Reflection images from ambient seismic noise: Geophysics, **74**, A63–A67, doi: 10.1190/1.3193529.

Curtis, A., P. Gerstoft, H. Sato, R. Snieder, and K. Wapenaar, 2006, Seismic interferometry—turning noise into signal: The Leading Edge, **25**, 1082-1092. doi: 10.1190/1.2349814.

Galloway, W. E., and L. F. Brown Jr., 1973, Depositional systems and shelf-slope relations on cratonic basin margin, uppermost Pennsylvanian of north-central

Texas: AAPG Bulletin, **57**, 1185-1218.

Grechka, V. and Y. Zhao, 2013, Microseismic Interferometry: SEG Technical Program Expanded Abstracts 2013, 2034-2039. doi: 10.1190/segam2013-0017.1.

Halliday D. and A. Curtis, 2008, Seismic interferometry, surface waves, and source distribution: Geophysics Journal International, **175**, 1067-1087. doi: 10.1111/j.1365-246X.2008.03918.x.

Halliday D., A. Curtis, and E. Kragh, 2008, Seismic surface waves in a suburban environment: Active and passive interferometric methods: The Leading Edge, **27**, 210-218. doi: 10.1190/1.2840369

Halliday D., A. Curtis, P. Vermeer, C. Strobbia, A. Glushchenko, D. van Manen, and J. Robertsson, 2010, Interferometric ground-roll removal: Attenuation of scattered surface waves in single sensor data: Geophysics, **75**, SA15-SA25. doi: 10.1190/1.3360948

Lin, F., D. Li, R. W. Clayton, and D. Hollis, 2012, Interferometry with a dense 3D dataset: SEG Technical Program Expanded Abstracts 2012, 1-6. doi: 10.1190/segam2012-0519.1.

Lin, F., D. Li, R. W. Clayton, and D. Hollis, 2013, High-resolution 3D shallow crustal

structure in Long Beach, California: Application of ambient noise tomography on a dense seismic array: Geophysics, **78**, Q45-Q56. doi: 10.1190/geo2012-0453.

Lin, F., M. H. Ritzwoller, and R. Snieder, 2009, Eikonal tomography: Surface wave tomography by phase front tracking across a regional broad-band seismic array: Geophysical Journal International, **177**, 1091–1110. doi: 10.1111/j.1365 246X.2009.04105.x.

Lundqvist, K, 2003, Big O notation: Massachusetts Institute of Technology Introduction to Computers and Programming, Class notes.

Montgomery, S. L., 2005, Mississippian Barnett Shale, Fort Worth Basin, North Central Texas: Gas-shale play with multi-trillion cubic foot potential: AAPG Bulletin, **89**, 155–175, doi: 10.1306/09170404042.

Nakata, N., J. Change, J. Lawrence, and P. Boue′, 2015, Body-wave extraction and tomography at Long Beach, CA, with ambient-noise interferometry: Journal of Geophysical Research. doi: 10.1029.

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. doi:10.1190/GEO2010-0188.1.

Oppenheim, A. V., and G. C. Verghese, 2010, Signals, Systems, and Inference – Class Notes for 6.011: Introduction to Communication, Control and Signal Processing Spring 2010, 183 – 194.

Pollastro, R. M., 2007, Total petroleum system assessment of undiscovered resources in the giant Barnett Shale continuous (unconventional) gas accumulation, Fort Worth Basin, Texas: AAPG Bulletin, **91**, 551–578. doi:10.1306/06200606007.

Pollastro, R. M., D. M. Jarvie, R. J. Hill, C. W. Adams, 2007, Geologic framework of the Mississippian Barnett Shale, Barnett Paleozoic total petroleum system, Bend arch–Fort Worth Basin, Texas: AAPG Bulletin, **91**, 405-436. doi: 10.1306/10300606008.

Pollastro, R. M., R. J. Hill, D. M. Jarvie, and M. E. Henry, 2003, Assessing undiscovered resources of the Barnett-Paleozoic total petroleum system, Bend arch–Fort Worth Basin province, Texas: Transactions of the Southwest Section, AAPG Annual Convention, Fort Worth, Texas: AAPG Datapages, 18.

Ruigrok, E., X. Campman, D. Draganov, and K. Wapenaar, 2010, High-resolution lithospheric imaging with seismic interferometry: Geophys. J. Int., **183**, 339 – 357, doi:10.1111/j.1365-246X.2010.04724.x.

Schuster, G. T., J. Yu, J. Sheng, and J. Rickett, 2004, Interferometric/daylight seismic imaging: Geophysical Journal International, **157**, 838–852, doi: 10.1111/gji.2004.157.issue-2.

Snieder, R., 2004, Extracting the Green's function from the correlation of coda waves: A derivation based on stationary phase: Phys. Rev. E, 046610.

Snieder, R., 2006, The theory of coda wave interferometry: Pure and Applied Geophysics, **163**, 455-473. doi: 10.1007/s00024-055-0026-6.

Snieder, R., M. Miyazawa, E. Slob, I. Vasconcelos, and K. Wapenaar, 2009, A comparison of strategies for seismic interferometry: Surveys in Geophysics, **30**, 503–523.

Vasconcelos, I., and R. Snieder, 2008a, Interferometry by deconvolution: Part 1 Theory for acoustic waves and numerical examples: Geophysics, **73**, S115–S128, doi: 10.1190/1.2904554.

Walper, J. L., 1982, Plate tectonic evolution of the Fort Worth Basin, in Martin, C.A., ed., Petroleum geology of the Fort Worth basin and Bend arch area: Dallas Geological Society, 237-251.

Wapenaar, K., D. Draganov, R. Snieder, X. Campman, and A. Verdel, 2010, Tutorial on

seismic interferometry: Part 1 — Basic principles and applications: Geophysics, **75**, 75A195-75A209. doi: 10.1190/1.345744

Wapenaar, K. and J. Fokkema, 2006, Green's function representations for seismic interferometry, Geophysics, **71**, SI33–SI44. doi: 10.1190/1.2213955.

Wapenaar, K., E. Slob, R.Snieder, and A. Curtis, 2010, Tutorial on seismic interferometry: Part 2 — Underlying theory and new advances: Geophysics, **75**, 75A211-75A227. doi: 10.1190/1.3463440

# Appendix: code

## Main program

```
% Cross correlates traces with header information in
'header_array.xls'
% by Joshua Hardisty

% Assumptions
% 1. Reference trace is predetermined
tic;

number_of_cores = 6;
matlabpool('open', number_of_cores);

disp('"processors are go"');
disp('The power of this many cores has been granted to you...');
disp(number_of_cores);
toc;


clear all;

%% Load all headers in 'header_array.xls' to find file locations

[header_numbers,header_text]=xlsread('header_array.xls');
load('Cheb2_1-30_80dB_order10.mat');

%% Retrieve reference trace


% Set receiver line(s)


receiver_point_column = 9;
receiver_line_column = 8;

% Choose reference traces

inline_rows = [1157];
xline_rows = [5118];
reference_row_location_vector = find_reference_rows(header_numbers,
inline_rows, xline_rows);

all_reference_trace = length(reference_row_location_vector);
length_from_reference_receiver_point = 59;
number_of_inlines = 3;
% overlapping_windows = 4;
overlapping_windows_vector = [4];
```

```
for multi_counter = 1:length(overlapping_windows_vector)

    overlapping_windows = overlapping_windows_vector(multi_counter);

% Length of trace

minute_length = 60000;  % 60000 ms is one minute
interval_length_minutes = 30;
interval_length = minute_length * interval_length_minutes;

% Time vector for length of trace in ms

old_sample_rate = 0.001;
new_sample_multiplier = 4;
new_sample_rate = old_sample_rate * new_sample_multiplier;
interval_length_time =
new_sample_rate:new_sample_rate:interval_length/1000;

% norm_length = minute_length*30/(10000 * new_sample_multiplier);
norm_length = 125;

% Whitening parameters

aec_length = 0.5;
flow = 0/new_sample_multiplier;
fhigh = 80/new_sample_multiplier;
nfilt = 100;

% Set frequency range

phase = 1;
f1 = [4]/new_sample_multiplier;
f2 = [5]/new_sample_multiplier;
f3 = [9]/new_sample_multiplier;
f4 = [12]/new_sample_multiplier;

lp1 = 20;
lp2 = 30;
amp1 = 1;
amp2 = 60;
sample_rate = 1/new_sample_rate;

freq_ranges = length(f1);

for reference_trace_counter = 1:all_reference_trace

    number_of_files_column = 7;
      if
      header_numbers(reference_row_location_vector(reference_trace_cou
      nter),number_of_files_column) < 3000
        continue;
    end

    % Retrieve reference_trace
```

```
    reference_receiver_point =
header_numbers(reference_row_location_vector(reference_trace_counter),
receiver_point_column);
    reference_receiver_line =
header_numbers(reference_row_location_vector(reference_trace_counter),
receiver_line_column);

    number_of_inlines_counter = number_of_inlines:1:number_of_inlines;

    receiver_vector = zeros(size(number_of_inlines_counter));
    for receiver_length_counter = 1:length(number_of_inlines_counter)
        receiver_vector(receiver_length_counter) =
      reference_receiver_line +
      (number_of_inlines_counter(receiver_length_counter)*4);
    end

    % Uses to compute against specific inlines

    receiver_vector = [1161,1157,1153];

    all_receivers = length(receiver_vector);

    reference_row_location =
      reference_row_location_vector(reference_trace_counter); % the
row location in 'header_numbers'

    [reference_trace, reference_trace_header,reference_records] =
      retrieve_reference_trace(reference_row_location,header_numbers,
      eader_text,'E');

    disp('Mission complete: import of reference trace successful');
    toc;
    %% Determine traces to cross correlate

    % 'for' loop used to process multiple lines in one job

    for receiver_counter = 1:all_receivers
        for freq_counter = 1:freq_ranges
            % Go through receiver lines one at a time
            receiver_line = receiver_vector(receiver_counter); % cross
correlate traces on receiver line

            % Find all receivers in the receiver line and keep their
coordinates in
            % "header_numbers" and "header_text"


            row =
            find(header_numbers(:,receiver_line_column)==receiver_lin
            );

            % Choose what traces to start and end on
```

```
receiver_points = 5118;

trace_locations =
zeros(length_from_reference_receiver_point*2 + 1,9);

loop_number = 1;
average_number_of_values = 0;

% Pre allocate memory for
correlated_traces_xcohere_notwhite = zeros(((2 *
interval_length)./new_sample_multiplier) - 1,
length_from_reference_receiver_point*2 + 1);
correlated_traces_xcohere_notwhite =
single(correlated_traces_xcohere_notwhite);
avg_power_spec = zeros(((2 *
interval_length)./new_sample_multiplier) - 1, 1);
avg_power_spec = single(avg_power_spec);
timing = zeros(length(receiver_points),6);


% Retrieve one trace, data condition, cross correlate, and
save
for counter = 1:length(receiver_points)
    %% Retrieve secondary trace
    retrieve_tic = tic;
    % Trace held in multiple files check

    receiver_point_row =
      find(header_numbers(row,receiver_point_column) ==
      receiver_points(counter));
    if isempty(receiver_point_row)
       continue;
    end

    row_location = row(receiver_point_row(1));


    [secondary_trace, secondary_trace_header,
      secondary_records, receiver_point] =
      retrieve_secondary_trace(row_location,reference_row
      location,header_numbers,header_text,'E');

    disp('Secondary trace loaded.');
    timing(counter,1) = toc(retrieve_tic);

    %% Time Sort
    secondary_trace_header =
      cell2mat(secondary_trace_header(:,2));

    [time_1, time_2] = time_sort(reference_trace_header,
      secondary_trace_header, reference_records,
      secondary_records);

    % Determine start and end times of correlated trace
```

```
% Start and end times are the times at which the
  traces overlap

if (time_1 == 0) || (time_2 == 0)
    disp('Error with determining start and end
        times');
    continue;
end


disp('Time sort complete.');


%% Crop reference trace and secondary trace, but keep
  original reference trace
crop_tic = tic;
reference_trace = single(reference_trace);
secondary_trace = single(secondary_trace);
[reference_crop, secondary_crop, error] =
  data_crop(reference_trace, reference_trace_header,
  secondary_trace, secondary_trace_header, time_1,
  time_2, reference_records, secondary_records,
  interval_length);
reference_crop_traces = length(reference_crop(1,:));
secondary_crop_traces = length(secondary_crop(1,:));

if error ~= 0
    disp('Error with determining crop');
    return;
end

clear secondary_trace;

disp('Crop check.');
timing(counter,2) = toc(crop_tic);

%%  Resample data

reference_crop =
  reduce_samples(reference_crop,new_sample_multiplier
  ;
secondary_crop =
  reduce_samples(secondary_crop,new_sample_multiplier
  ;

disp('Resample check.');


%% Detrend

reference_crop = detrend(reference_crop);
secondary_crop = detrend(secondary_crop);
```

```
disp('Detrend check.');

%% Ormsby filter secondary trace

        reference_crop = filter(Hd,reference_crop);
        filter_tic = tic;
        secondary_crop = filter(Hd,secondary_crop);
        disp('secondary trace filtered');
        timing(counter,3) = toc(filter_tic);

%   disp('No filter.');

%% Normalize cropped reference and secondary traces
norm_tic = tic;
%   % Length to 1
%   reference_crop =
    norml_pc(reference_crop,norm_length);
%   secondary_crop =
    norml_pc(secondary_crop,norm_length);

%     % 1bit
%     reference_crop = sign(reference_crop);
%     secondary_crop = sign(secondary_crop);

% Running absolute mean normalization

reference_crop =
  running_abs_mean_normalization_edit(reference_crop,
  orm_length);
secondary_crop =
  running_abs_mean_normalization_edit(secondary_crop,
  orm_length);

disp('Normalize check.');
toc(norm_tic);
timing(counter,5) = toc(norm_tic);

%   disp('No normalization.');

%% Whiten traces
white_tic = tic;

%     reference_crop = single(reference_crop);
%     secondary_crop = single(secondary_crop);
```

```
% % %           %%% reference_crop =
                %%%     whiten_traces(reference_crop,new_sample_rate);
% % %           %%% secondary_crop =
                %%%     whiten_traces(secondary_crop,new_sample_rate);
                reference_crop = whiten_traces_short(reference_crop);
                secondary_crop = whiten_traces_short(secondary_crop);

                timing(counter,6) = toc(white_tic);

                disp('Whitening check.')
                toc(white_tic);

%                disp('No whitening.');


                %% Cross correlate traces
                xcohere_tic = tic;
                lag_time_xcohere = single(zeros(((2 *
                  interval_length)./new_sample_multiplier) - 1,1));
                reference_crop = single(reference_crop);
                secondary_crop = single(secondary_crop);

                for xcorr_counter = 1:overlapping_windows
                    reference_hold =
                        single(overlap_time_windows(reference_crop,
                        overlapping_windows, xcorr_counter));
                    secondary_hold =
                        single(overlap_time_windows(secondary_crop,
                        overlapping_windows, xcorr_counter));
                    lag_time_xcohere_hold =
                        cross_cohere_fft_gpu_L2mean(secondary_hold,
                        reference_hold);
                    lag_time_xcohere = lag_time_xcohere +
                        single(lag_time_xcohere_hold);
                    clear reference_hold;
                    clear secondary_hold;
                end


                clear reference_crop;
                clear secondary_crop;

                disp('cross corr check');
                timing(counter,4) = toc(xcohere_tic);
                %% Save cross correlated trace and location of trace,
receiver line known


                correlated_traces_xcohere_notwhite(:,loop_number)=
                  lag_time_xcohere;

                disp_text = sprintf('Reference Trace Location: inline
                  %d xline %d', reference_receiver_line,
                  reference_receiver_point);
```

```
disp(disp_text);
disp_text = sprintf('Current secondary line: %d inline
  %d', receiver_line);
disp(disp_text);
disp_text = sprintf('                                Trace
  %d out of %d', counter, length(receiver_points));
disp(disp_text);
number_of_computed_values = (time_2-time_1)/1000;
disp_text = sprintf('                                Number
  of values computed: %d', number_of_computed_values);
disp(disp_text);
average_number_of_values = average_number_of_values +
  number_of_computed_values/length(receiver_points);

trace_locations(loop_number,1) = receiver_line;
trace_locations(loop_number,2) = receiver_point;
actual_x_column = 13;
actual_y_column = 14;
trace_locations(loop_number,3) =
  header_numbers(row_location,actual_x_column); %
  actual x
trace_locations(loop_number,4) =
  header_numbers(row_location,actual_y_column); %
  actual y
trace_locations(loop_number,5) =
  reference_crop_traces;
trace_locations(loop_number,6) =
  average_number_of_values;
trace_locations(loop_number,7) =
  abs(header_numbers(reference_row_location,actual_x_
  olumn) -
  header_numbers(reference_row_location,actual_x_colu
  n)); % relative x
trace_locations(loop_number,8) =
  abs(header_numbers(reference_row_location,actual_y_
  olumn) -
  header_numbers(reference_row_location,actual_y_colu
  n)); % relative y
trace_locations(loop_number,9) =
  ((trace_locations(loop_number,7)^2) +
  (trace_locations(loop_number,8)^2))^(.5); % relative
  distance


loop_number= loop_number + 1;

%% Save cross correlated trace and location of trace,
  receiver line known

% create file name

save_file = sprintf('%d
  %s_%d_%d_%d',new_sample_multiplier ,'samp_Hdfilt1
  30order10_RAM_wind4_white_xcohere_line_single',
  reference_receiver_line,
  reference_receiver_point,receiver_line);
```

```
                save_info = sprintf('%d-%s_%d_%d_%d',
                  new_sample_multiplier,'samp_Hdfilt1
                  30order10_RAM_wind4_white_xcohere_end_info_single',r
                  eference_receiver_line, reference_receiver_point,
                  receiver_line);

                save(save_file,
                  'correlated_traces_xcohere_notwhite','trace_locatio
                  s','avg_power_spec','timing');

                save(save_info, 'loop_number', 'counter',
                  'average_number_of_values', 'f1','f2','f3','f4');

                disp('save check');

                toc;
            end
        end
    end
end
end
% matlabpool('close');
disp('Bomb-diggity.')
% matlabpool('close');
```

## Time Sort

```
% Function to sort data
% by Joshua Hardisty

 function [start_time, end_time] = time_sort(reference_trace_header,
secondary_trace_header, reference_records, secondary_records)

% Suggestion: 'header_array.xls' to determine sections of the traces
that occur
% at the same time


header_1 = reference_trace_header;
header_2 = secondary_trace_header;

header_1_start = header_1(6,1);
header_1_end = header_1_start +
(header_1(1,1)*reference_records*1000);
header_2_start = header_2(6,1);
```

```
header_2_end = header_2_start +
(header_2(1,1)*secondary_records*1000);

%% Check to make sure that the numbers make sense

if header_1_start > header_1_end
    start_time = 0; % Error: start time in header 1 is greater than
      end time
    end_time = 0; % Error: start time in header 1 is greater than end
      time
    return;
end

if header_2_start > header_2_end
    start_time = 0; % Error: start time in header 2 is greater than
      end time
    end_time = 0; % Error: start time in header 2 is greater than end
      time
    return;
end

%% Find start time


if header_2_end < header_1_start
    % secondary trace occurs before reference trace
    start_time = 0;
elseif header_2_start <= header_1_start
    % secondary trace starts before reference trace
    % reference trace is limiting the start time
    start_time = header_1_start;
else % (header_2_start => header_1_start)
    % secondary trace starts during reference trace
    % secondary trace is limiting the start time
    start_time = header_2_start;
end

%% Find end time

if header_2_start > header_1_end
    % secondary trace occurs after reference trace
    end_time = 0;
elseif header_2_end >= header_1_end
    % secondary trace ends after reference trace
    % reference trace is limiting end time
    end_time = header_1_end;
else % header_2_end < header_1_end
    % secondary trace ends before reference trace
    % secondary trace is limiting end time
    end_time = header_2_end;
end
```

## Normalization length-to-1

```
function [ norm_matrix ] = norm1( matrix, norm_length )
% function so that the square root of all inputs squared is 1


norm_matrix = zeros(size(matrix));
steps = fix(length(matrix(:,1))/norm_length);


for norm_counter = 1:length(matrix(1,:))
    for norm_counter_hold = 1:steps
        start_norm = 1 + ((norm_counter_hold - 1)*norm_length);
        end_norm = norm_counter_hold*norm_length;
        matrix_hold =
      gpuArray(matrix(start_norm:end_norm,norm_counter));
        norm_matrix(start_norm:end_norm,norm_counter) =
      gather(normc(matrix_hold));
    end
    if steps ~= length(matrix(:,1))/norm_length
        start_norm = 1 + (steps * norm_length);
        end_norm = length(matrix(1,:));
        matrix_hold =
            gpuArray(matrix(start_norm:end_norm,norm_counter));
        norm_matrix(start_norm:end_norm,norm_counter) =
      gather(normc(matrix_hold));
    end

end
```

## Normalization running-absolute-mean

```
% Function performs running-absolute-mean normalization
% by Joshua Hardisty

function [normalized_trace] =
running_abs_mean_normalization(non_normal_trace, time_window)

running_window = time_window;
abs_trace = abs(non_normal_trace);
number_of_traces = length(abs_trace(1,:));
length_of_trace = length(abs_trace(:,1));
upper_limit = length_of_trace - running_window;
weights = zeros(length_of_trace, number_of_traces);
```

```
running_window_times2_plus1 = 2*running_window + 1;


parfor counter = 1:number_of_traces
    single_trace = abs_trace(:,counter);
    initial_sum = sum(single_trace(1:running_window));
    for sub_counter = 1:length_of_trace
        if sub_counter <= (running_window + 1)
            initial_sum = initial_sum + single_trace(sub_counter +
            running_window);
            window_total = (initial_sum)/(running_window +
            sub_counter);
        elseif sub_counter >= upper_limit + 1
            initial_sum = initial_sum - single_trace(sub_counter -
            running_window - 1);
            window_total = (initial_sum)/(running_window +
            length_of_trace - sub_counter + 1);
        else
            initial_sum = initial_sum + single_trace(sub_counter +
            running_window) - single_trace(sub_counter -
            running_window - 1);
            window_total = (initial_sum)/running_window_times2_plus1;
        end
        weights(sub_counter,counter) = window_total;
    end
end

normalized_trace = non_normal_trace./weights;
```

## Whiten

```
function [matrix] = whiten_traces_short(matrix)

matrix = single(matrix);

matrix = fft(matrix);
keep_phase = angle(matrix);
matrix = matrix./(abs(matrix)+max(mean(abs(matrix))));
% matrix = matrix./abs(matrix);
matrix = abs(matrix).*exp(1i*keep_phase);
matrix = abs(ifft(matrix));
```

## Cross-correlation

```
% Function to cross correlate two (n,1) vectors
% by Joshua Hardisty

% Uses FFT method
% Assumptions:
% 1. matrix_1 and matrix_2 have the same dimensions


function [time_lag_fft] = cross_corr_fft_pc(matrix_1,matrix_2)

side_width = length(matrix_1(1,:));
down_width = length(matrix_1(:,1));
xcorr_length = (down_width*2)-1;
sum_xcorr_fft = zeros(xcorr_length,1);
sections = 4;


for counter = 1:sections
    fftstart = 1 + (side_width*(counter-1)/sections);
    fftend = counter*side_width/sections;
    current_xcorr_fft =
      ifft(fft(matrix_1(:,fftstart:fftend),xcorr_length).*conj(fft(ma
      rix_2(:,fftstart:fftend),xcorr_length)));
    current_xcorr_fftshft=fftshift(current_xcorr_fft);
    sum_xcorr_fft = sum_xcorr_fft +
sum(current_xcorr_fftshft,2)/side_width;
end

time_lag_fft = sum_xcorr_fft;
```

## Cross-coherence

```
% Function use cross coherence on two (n,1) vectors
% by Joshua Hardisty

% Uses FFT method
% Assumptions:
% 1. matrix_1 and matrix_2 have the same dimensions


function [time_lag_total] =
cross_cohere_fft_gpu_L2mean(matrix_1,matrix_2)

% Reset gpuDevice
reset(gpuDevice);

max_gpu_count = 5e6;
side_width = length(matrix_1(1,:));
down_width = length(matrix_1(:,1));
xcorr_length = (down_width*2)-1;
max_lines = fix(max_gpu_count/down_width);
sections = fix(side_width/max_lines);



if sections == 0

    matrix_1_gpu = gpuArray(matrix_1);
    matrix_2_gpu = gpuArray(matrix_2);
    % NEW ***************
    fft_matrix_1 = fft(matrix_1_gpu,xcorr_length);
    fft_matrix_2 = fft(matrix_2_gpu,xcorr_length);
    fft_matrix_1_scale = sqrt(sum(abs(fft_matrix_1).^2,2));
    fft_matrix_2_scale = sqrt(sum(abs(fft_matrix_2).^2,2));
    fft_matrix_scale = fft_matrix_1_scale.*fft_matrix_2_scale;
    fft_matrix_1_norm = sqrt(sum(abs(fft_matrix_1).^2));
    fft_matrix_2_norm = sqrt(sum(abs(fft_matrix_2).^2));
    fft_matrix_norm = (fft_matrix_1_norm.*fft_matrix_2_norm);
    matrix_white = .01*(abs(fft_matrix_1)+abs(fft_matrix_2));
    fft_matrix_norm = bsxfun(@plus,matrix_white,fft_matrix_norm);
    current_xcorr_fft = fft_matrix_1.*conj(fft_matrix_2);
    keep_angle = angle(current_xcorr_fft);
    current_xcorr_fft = abs(current_xcorr_fft)./abs(fft_matrix_norm);

    current_xcorr_fft = current_xcorr_fft.*exp(1i*keep_angle);
    current_xcorr_fft = ifft(current_xcorr_fft);
    % NEW ***************
    current_xcorr_fftshft=fftshift(current_xcorr_fft);
    time_lag_fft = sum(current_xcorr_fftshft,2)/side_width;
    time_lag_fft = gather(time_lag_fft);
    time_lag_total = abs(time_lag_fft);
```

```
else


    time_lag_total = zeros(xcorr_length,1);
    increment = fix(side_width/sections);

    % Cross correlate beginning of matrix
    for counter = 1:sections
        fftstart = 1 + increment*(counter-1);
        fftend = counter * increment;
        matrix_1_hold = matrix_1(:,fftstart:fftend);
        matrix_2_hold = matrix_2(:,fftstart:fftend);
        matrix_1_gpu = gpuArray(matrix_1_hold);
        matrix_2_gpu = gpuArray(matrix_2_hold);
        % NEW ***************
        fft_matrix_1 = fft(matrix_1_gpu,xcorr_length);
        fft_matrix_2 = fft(matrix_2_gpu,xcorr_length);
        fft_matrix_1_norm = sqrt(sum(abs(fft_matrix_1).^2));
        fft_matrix_2_norm = sqrt(sum(abs(fft_matrix_2).^2));
        fft_matrix_1_scale = sqrt(sum(abs(fft_matrix_1).^2,2));
        fft_matrix_2_scale = sqrt(sum(abs(fft_matrix_2).^2,2));
        fft_matrix_scale = fft_matrix_1_scale.*fft_matrix_2_scale;
        fft_matrix_norm = (fft_matrix_1_norm.*fft_matrix_2_norm);
        matrix_white = .01*(abs(fft_matrix_1)+abs(fft_matrix_2));
        fft_matrix_norm = bsxfun(@plus,matrix_white,fft_matrix_norm);
        current_xcorr_fft = fft_matrix_1.*conj(fft_matrix_2);
        keep_angle = angle(current_xcorr_fft);
        current_xcorr_fft =
abs(current_xcorr_fft)./abs(fft_matrix_norm);

        current_xcorr_fft = current_xcorr_fft.*exp(1i*keep_angle);
        current_xcorr_fft = ifft(current_xcorr_fft);
        % NEW ***************
        current_xcorr_fftshft=fftshift(current_xcorr_fft);
        time_lag_fft = sum(current_xcorr_fftshft,2)/side_width;
        time_lag_fft = gather(time_lag_fft);
        time_lag_total = time_lag_total + time_lag_fft;
    end

        % Cross correlate end of matrix that doesn't go into
            'sections' evenly
        fftstart = (sections * increment) + 1;
        fftend = side_width;
        matrix_1_hold = matrix_1(:,fftstart:fftend);
        matrix_2_hold = matrix_2(:,fftstart:fftend);
        matrix_1_gpu = gpuArray(matrix_1_hold);
        matrix_2_gpu = gpuArray(matrix_2_hold);
        % NEW ***************
        fft_matrix_1 = fft(matrix_1_gpu,xcorr_length);
        fft_matrix_2 = fft(matrix_2_gpu,xcorr_length);
        fft_matrix_1_norm = sqrt(sum(abs(fft_matrix_1).^2));
        fft_matrix_2_norm = sqrt(sum(abs(fft_matrix_2).^2));
        fft_matrix_1_scale = sqrt(sum(abs(fft_matrix_1).^2,2));
        fft_matrix_2_scale = sqrt(sum(abs(fft_matrix_2).^2,2));
        fft_matrix_scale = fft_matrix_1_scale.*fft_matrix_2_scale;
```

```
        fft_matrix_norm = (fft_matrix_1_norm.*fft_matrix_2_norm);
        matrix_white = .01*(abs(fft_matrix_1)+abs(fft_matrix_2));
        fft_matrix_norm = bsxfun(@plus,matrix_white,fft_matrix_norm);
        current_xcorr_fft = fft_matrix_1.*conj(fft_matrix_2);
        keep_angle = angle(current_xcorr_fft);
        current_xcorr_fft =
abs(current_xcorr_fft)./abs(fft_matrix_norm);

        current_xcorr_fft = current_xcorr_fft.*exp(1i*keep_angle);
        current_xcorr_fft = ifft(current_xcorr_fft);
        % NEW ***************
        current_xcorr_fftshft=fftshift(current_xcorr_fft);
        time_lag_fft = sum(current_xcorr_fftshft,2)/side_width;
        time_lag_fft = gather(time_lag_fft);
        time_lag_total = time_lag_total + time_lag_fft;

end
time_lag_total =abs(time_lag_total);
```