Improving Separation of Signals from Multiple Physical Quantities Detected by Sensor Arrays

Sarah E. Morgan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mathematics

Eileen R. Martin, Chair
Serkan Gugercin
Gary R. Pickrell

May 5, 2022
Blacksburg, Virginia

Keywords: Applied Math, Signal Processing, Time Series Analysis

Improving Separation of Signals from Multiple Physical Quantities Detected by Sensor Arrays

Sarah E. Morgan

(ABSTRACT)

Modern array sensing systems, such as distributed fiber optic sensing, are used in many applications which may record a mixture of responses to multiple physical quantities. In these applications, it may be helpful to be able to separate this mixture of responses into the signals resulting from the individual sources. This is similar to the cocktail party problem posed with Independent Component Analysis (ICA), in which we use gradient ascent and fixed point iteration optimization algorithms to achieve this separation. We then seek to apply the problem setup from ICA to mixed signals resulting from a sensor array with the goal of maintaining coherence throughout resulting spatial arrays. We propose a new post-processing technique after separation to pair up the signals from different types of physical quantities based on the Symmetric Reverse Cuthill-McKee (SRCM) and Symmetric Approximate Minimum Degree (SAMD) permutations of the coherence matrix.

Improving Separation of Signals from Multiple Physical Quantities Detected by Sensor Arrays

Sarah E. Morgan

(GENERAL AUDIENCE ABSTRACT)

Some modern sensing systems are able to collect data resulting from different types of sources, such as vibrations and electromagnetic waves, at the same time. This means we have signals resulting from a mixture of sources. An example of one such modern sensing system is distributed fiber optic sensors used in geoscience applications, such as seismology and subsurface imaging, to measure strain along the fiber optic cable. In many applications, it may be helpful to obtain the signals from each of these sources separately, instead of having a mixture of these sources. We propose the use of optimization algorithms, in particular two algorithms arising from Independent Component Analysis (ICA), which seek to maximize a function in order to separate these signals. We then explore changes required to the algorithms for scenarios in which we have multiple sensors spaced some distance away from each other which record signals from two different sources. We also present a method of determining which separated signals correspond to which sensors after performing signal separation.

Acknowledgments

First I would like to thank my advisor, Dr. Eileen Martin, for providing valuable mentorship and guidance throughout the entire research process.

I would also like to acknowledge the grant providing the funding for this work: DE-FE0031786 and thank some of the collaborators involved in this project; Dr. Daniel Homa, Zachary Hileman, Dr. Anbo Wang, and Sentek Instrument LLC.

Next I would like to thank Dr. Gary Pickrell and Dr. Serkan Gugercin for their guidance as committee members.

Lastly, I would like to thank all my friends and family who have supported me throughout my years in school. A special thanks to my fiancé Jacob who made my last two years of school some I'll never forget.

Contents

List of Figures

List of Tables

List of Abbreviations

DAS Distributed Acoustic Sensing

FBG Fiber Bragg Grating

ICA Independent Component Analysis

NMF Nonnegative Matrix Factorization

SAMD Symmetric Approximate Minimum Degree

SRCM Symmetric Reverse Cuthill-McKee

Chapter 1

Introduction

Signal separation is a topic studied in many fields today with the emergence of large scale data sets that contain noisy signals or signals resulting from multiple sources [4, 14, 15, 16]. Signal separation seeks to recover a set of source signals from a set of mixed signals. This may involve separating a signal from noise or separating multiple simultaneous source signals. In some instances we may know something about the source signals themselves, such as their frequency range. In these cases we may be able to separate the signals by performing a fairly simple process such as performing a Fourier transform on the mixed signal, masking the frequency components outside the expected range, and performing the inverse Fourier transform to recover one of the individual sources. In other cases, we might not have much information about the source signals or the process in which they were mixed. This leads to what is known as blind source separation. Blind signal separation seeks to recover the original component signals from a set of mixed signals with little to no information about the signals themselves. Topics involving signal separation range from audio signal separation [14] to recovering signals from brain activity in medical imaging [4].

Audio source separation has been widely studied. Two examples of techniques include non-negative matrix factorization (NMF) and neural network based audio source separation [14]. With NMF, the given signal is represented by a matrix where the columns are the data samples and the rows are the features. This matrix is then factored into two matrices $\mathbf{V} \approx \mathbf{WH}$ where $\mathbf{W}$ acts as a dictionary of recurring patterns and $\mathbf{H}$ contains the decomposition

coefficients that approximate every column for the dictionary matrix. NMF can be used for audio spectral analysis if we transform the time domain data into the frequency domain using a short-time Fourier transform. Then after applying the NMF we get the unmixed spectra elements in the dictionary matrix $\mathbf{W}$ and the mixing proportions of each time-frame in $\mathbf{H}$. Neural network based techniques extend this NMF approach by learning source-specific dictionaries during training time and then decompose the data using these learned dictionaries as templates. These techniques can effectively identify specific features of the data, such as noise, but may have difficulty separating two signals in their entirety [14].

Signal separation is also applicable to medical imaging with the electroencephalogram (EEG) test [4]. This test detects electrical activity in the brain using electrodes attached to the scalp of the patient. With this kind of high sensitivity test, there are many sources for noise in the data such as eye movements, heart beat, and muscle movements. Signal separation techniques seek to separate this noise from the electrical activity in the brain itself. Techniques used for EEG data include independent component analysis (ICA) and the approximate joint diagonalization of Fourier co-spectral matrices [4].

In this thesis, we study the multidimensional signal separation problem arising from a new type of distributed fiber-optic sensing that is sensitive to both acoustic and magnetic fields [7]. We seek to develop an algorithm which can be used to separate signals resulting from simultaneous acoustic and magnetic sources as we evaluate the performance of this new fiber-optic sensing system.

This work introduces the several numerical optimization algorithms for signal separation arising from the ICA model [9]. The optimization algorithms include both gradient ascent and fixed point iteration methods for the choice of two different objective functions. The accuracy and the robustness of these algorithms is analyzed through the comparison of separation results for synthetic signals, signals produced from laboratory experiments, and signals

resulting from simulated pressure waves. We then introduce a new multi-channel setup for the ICA model as well as a post-processing method for maintaining coherence throughout resulting spatial arrays, which each contain array responses to one of the separated sources.

1.1 Acoustic and Magnetic Sensing Fiber

Distributed Acoustic Sensing (DAS) provides the measurement of a fiber's strain through optical interferometry. Laser pulses are applied to the fiber as we measure the optical phase changes in Rayleigh backscattered light as seen in Figure 1.1.

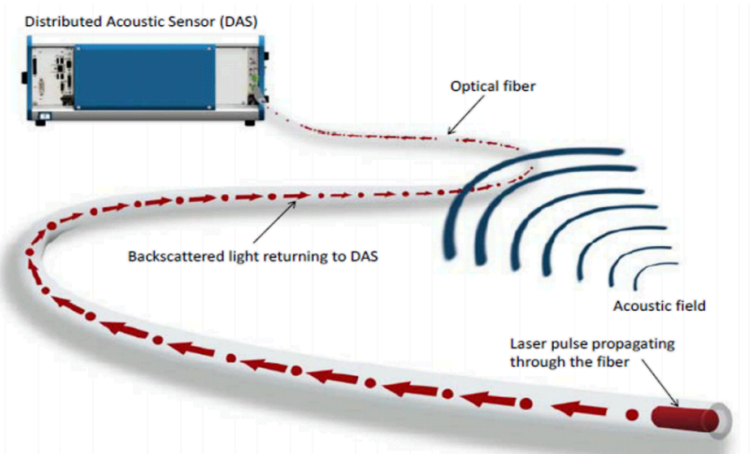

Figure 1.1: Simple model of DAS [6]

This provides an array of strain sensors at meter-scale intervals which can monitor areas often present in applications in the oil and gas industry where it may not be possible to use traditional sensors [11].

Unconventional oil and gas resources have led to extraction methods other than the traditional vertical well extraction, which cause complex subsurface situations. These include directional wellbore drilling, hydraulic fracturing, and large-scale pipelines [3, 12, 23]. Conventional fiber-optic cables measure only acoustic sources. With these new complex sit-

uations, it would be beneficial to have a fiber-optic cable capable of measuring multiple subsurface parameters. A new optical fiber and sensing system is currently in developmental stages which will be capable of distributed and real-time measurement of multiple parameters; specifically acoustic and magnetic. The multi-material fiber opting sensing design is proposed with a single mode core and two nickel rods with a silica cladding as seen in Figure 1.2 [7].

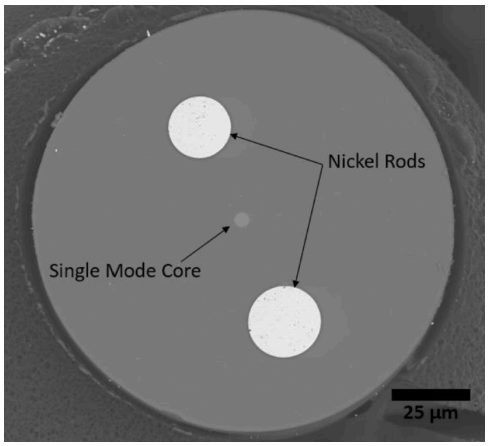

Figure 1.2: Distributed magnetic sensing fiber [7]

Fiber Bragg Gratings (FBG) are inscribed in the core of the optical fiber which reflect particular wavelengths of light and transmit all others. A FBG-based interferometer is then used to measure changes in the field strength when subjected to mechanical strain or a magnetic field. These resulting signals are represented in terms of strain as a function of time. An example of DAS array data can be seen in Figure 1.3.

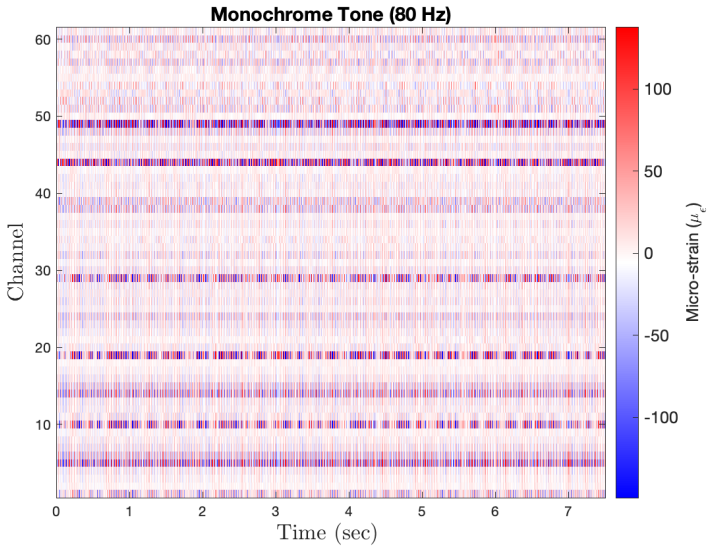

Figure 1.3: Example of DAS array using a 300 meter acoustic fiber-optic cable created by Sentek Instrument LLC [13]. Depicts fiber strain response from monochromatic tone of 80 Hz across 60 channels spaced 5 meters apart with a sampling frequency of 62500 Hz.

In testing and assessing the results of this new sensing system, we want to be able to separate the signals from the distinct acoustic and magnetic sources. This will allow us to evaluate the performance of the new sensing system as well as provide algorithms that would allow for real time separation of signals outside of the laboratory environment.

1.2 Thesis Organization and Contributions

This thesis presents methods to improve the separation of signals from multiple physical quantities detected by sensor arrays. Chapter 2 introduces Independent Component Analysis (ICA) with assumptions and numerical algorithms from existing literature as well as additional proofs and derivations not found in previous literature. In Chapter 3, we study the signal separation of several test cases. We present a way to determine if the signal separation was successful and compare the numerical algorithms introduced in Chapter 2. Chapter

4 then introduces a new post processing technique to be used after ICA signal separation for sensor arrays. For this post-processing we present the formulation of a coherence matrix and prove the structure of the matrix. A final conclusion and suggestions for future work can be found in Chapter 5. The original contributions in this thesis are:

- Full derivations of the kurtosis gradient ascent and fixed point iterations (Sections 2.3.1, 2.4.2);
- Pre-processing technique involving cross-correlations for signal separation of non-instantaneous mixtures (Section 3.3);
- Method of comparing signal separation results for kurtosis and negentropy fixed point iteration algorithms across several test cases (Section 3.1);
- Array coherence post-processing technique involving the calculation of a coherence matrix and sparse matrix operations (Chapter 4);
- Theoretical properties of the coherence matrix that lead to the use of sparse matrix operations (Section 4.2.3).

1.3 Available Software

All software used in this thesis was developed in Matlab by the author unless otherwise specified. The code used to reproduce each figure in this paper is publicly available at https://doi.org/10.5281/zenodo.6533203. Descriptions of each file can be found in the repository's README file.

Chapter 2

Independent Component Analysis

A technique often used for blind source separation is Independent Component Analysis (ICA) [9]. ICA poses the problem of signal separation by introducing a linear mixing model, which then separates mixed signals into separate individual signals utilizing optimization techniques. The most common introductory explanation of ICA is the cocktail party problem. Suppose two people are in a room talking simultaneously and there are two sensors, such as microphones, in different locations recording them as seen in Figure 2.1. We can treat each of these signals as a function of time t. In this case our source signals, $s_1(t), s_2(t)$, would be the individual audio signals from the two people talking and the mixed signals, $x_1(t), x_2(t)$, would be the audio signals picked up by the microphones. Then we can represent the mixed signals as a linear combination of the source signals

$$x_1(t) = \alpha_{11} s_1(t) + \alpha_{12} s_2(t) \tag{2.1}$$

$$x_2(t) = \alpha_{21} s_1(t) + \alpha_{22} s_2(t) \tag{2.2}$$

where $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}$ are our "mixing parameters" which depend on the distances of the speakers from the microphones. The cocktail party problem seeks to recover estimations of the original signals of the two people speaking, $\tilde{s_1}(t), \tilde{s_2}(t)$ from only the mixed recorded signals $x_1(t)$ and $x_2(t)$.

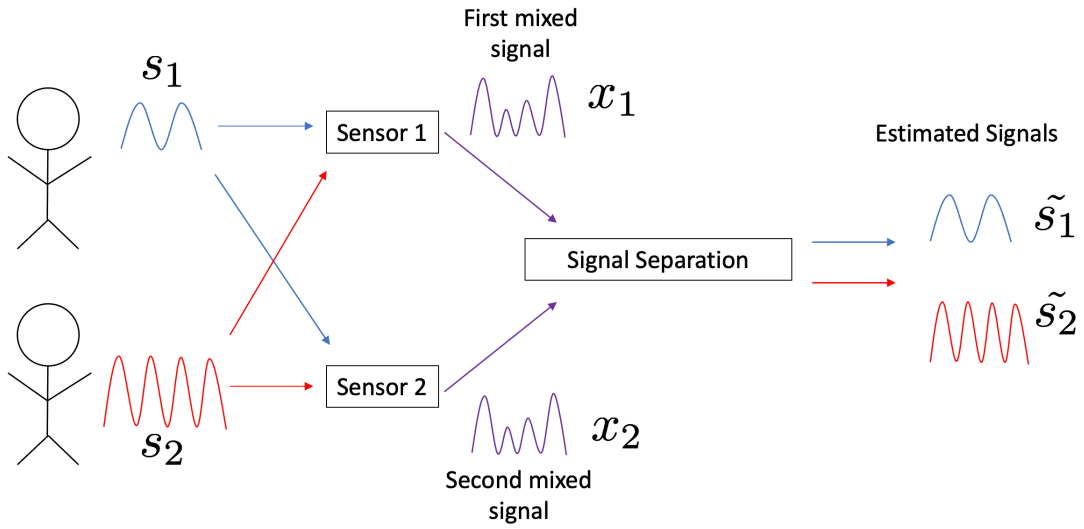

Figure 2.1: Cocktail party problem

After making some further assumptions about the individual component signals we seek to recover we can use statistical properties and optimization techniques to define an algorithm which can recover the source signals with some limitations as discussed further in the subsequent sections.

2.1 Overview of Problem

2.1.1 Notation

First we will clarify the notation used to denote matrices, vectors, and scalars.

- Matrix: bold face, $\mathbf{A}$
- Column vector: arrow overhead lowercase letter, $\vec{x}$
- Row vector: transpose of column vector, $\vec{x}^\top$

- scalar: lowercase Greek letter, α

Then, for example, we have

$$\mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix}, \quad \vec{a_1} = \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \end{bmatrix}, \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} \vec{x_1}^\top \\ \vec{x_2}^\top \end{bmatrix} = \begin{bmatrix} \chi_{11} & \chi_{12} & \chi_{13} & \cdots \\ \chi_{21} & \chi_{22} & \chi_{23} & \cdots \end{bmatrix}.$$

2.1.2 General Problem Statement

Suppose we have n independent source signals, $s_1(t), s_2(t), \ldots, s_n(t)$, and n mixed components $x_1(t), x_2(t), \ldots, x_n(t)$. First we treat each mixture $x_i(t)$ and each individual component $s_i(t)$ as a random variable instead of a time signal. This means that we are assuming an instantaneous mixture of signals so $\mathbf{A}$ is the same for every time sample. Then we have that

$$\vec{x_j}^\top = \alpha_{j1}\vec{s_1}^\top + \alpha_{j2}\vec{s_2}^\top + \cdots + a_{jn}\vec{s_n}^\top, \quad \forall j \tag{2.3}$$

where α_{jn} is a scalar, $\vec{x_j}^\top \in (1 \times k)$, and $\vec{s_n}^\top \in (1 \times k)$ for k time samples. We can also rewrite this in matrix vector notation

$$\mathbf{X} = \mathbf{AS} = \begin{bmatrix} \sum_{i=1}^n \vec{a_i}^\top s_{i1} & \ldots & \sum_{i=1}^n \vec{a_i}^\top s_{ik} \end{bmatrix} \tag{2.4}$$

where $\mathbf{X} \in \mathbb{R}_{n\times k}$, $\mathbf{A} \in \mathbb{R}_{n\times n}$, $\mathbf{S} \in \mathbb{R}_{n\times k}$ and

$$\mathbf{X} = \begin{bmatrix} \vec{x_1}^\top \\ \vec{x_2}^\top \\ \vdots \\ \vec{x_n}^\top \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nn} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s_1}^\top \\ \vec{s_2}^\top \\ \vdots \\ \vec{s_n}^\top \end{bmatrix}$$

such that $\vec{a_i}$ denotes a column of the matrix $\mathbf{A}$ and s_{ik} is the kth time sample of $\vec{s_i}$. We call the matrix $\mathbf{A}$ the "mixing matrix" because its entries determine how the individual independent components from $\mathbf{S}$ combine to form $\mathbf{X}$ [9].

2.1.3 Aim of ICA Optimization

In order to separate the mixed signals $\mathbf{X}$ into individual signals $\mathbf{S}$ we must solve Equation (2.4) for the inverse of the matrix $\mathbf{A}$. We will utilize optimization techniques such as gradient ascent and fixed point iterations to do so. With gradient ascent and fixed point iteration optimization methods, we must have an objective function which we wish to maximize. We cannot use $\mathbf{A}^{-1}$ directly as our objective function because it is a matrix of scalars and we need a variable to differentiate over for gradient ascent and fixed point iterations. Instead we will define a function which depends on both $\mathbf{A}^{-1}$ and $\mathbf{X}$. As explained further in Section 2.2.4, we will choose an objective function, which is a function of $\mathbf{A}^{-1}$ and $\mathbf{X}$, and minimizes how Gaussian the separated signals $\mathbf{A}^{-1}\mathbf{X}$ are.

2.2 Assumptions

One of the main assumptions of the ICA model is that the individual components we are trying to estimate are independent [9]. To simplify the algorithm, it also is assumed that the individual and mixed components have a zero mean and a unit variance. These assumptions allow for the explicit derivation of gradient ascent and fixed point iterations in the optimization algorithms. Also note that we observe $\mathbf{X}$, but both $\mathbf{A}$ and $\mathbf{S}$ are assumed to be unknown. If we knew our mixing matrix $\mathbf{A}$, we could directly recover the individual components as $\mathbf{S} = \mathbf{A}^{-1}\mathbf{X}$.

2.2.1 Independence

Suppose we have random variables $y_1, y_2, \ldots, y_n$. These are said to be **independent** if

$$p(y_1, y_2, \ldots, y_n) = p(y_1)p(y_2)\ldots p(y_n), \tag{2.5}$$

where $p(y_1, y_2, \ldots y_n)$ is their joint probability density function and $p(y_1), p(y_2), \ldots p(y_n)$ are marginal probability density functions [9]. Independence is important to recall because it allows for the simplification of calculations within the derivations involved for the optimization algorithms. ICA essentially finds the "most independent" features of the mixed input signals. Assuming independence also allows for significant simplification when we begin calculating expected values of the signals.

2.2.2 Zero Mean

If the observed mixed signals do not have a zero mean, we can center the components through some pre-processing without changing the mixing matrix $\mathbf{A}$. Suppose we have our observed mixed signals $\mathbf{X}_{obs}$. Then we can center the signals by subtracting their sample mean and obtain

$$\mathbf{x} = \mathbf{X}_{obs} - E\{\mathbf{X}_{obs}.\} \tag{2.6}$$

Notice the independent components now also have a zero mean because

$$E\{\mathbf{S}\} = \mathbf{A}^{-1}E\{\mathbf{X}\} = \mathbf{0}. \tag{2.7}$$

After performing our ICA algorithm and finding $\mathbf{S}$, we can reconstruct our components with the subtracted mean added back in by doing

$$\mathbf{S} = \mathbf{S} + \mathbf{A}^{-1} E\{\mathbf{X}_{obs}\}, \tag{2.8}$$

see [9] for more details.

2.2.3 Unit Variance

To obtain variables with a unit variance, we can perform what is known as *whitening*. This means that the components are uncorrelated, a consequence of independence, and they have a unit variance. A common pre-processing technique uses the eigenvalue decomposition of the covariance matrix to obtain whitened data [9]. Lets consider the covariance matrix

$$E\{\mathbf{X}\mathbf{X}^T\} = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{2.9}$$

and the whitening matrix

$$\mathbf{V} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T, \tag{2.10}$$

where $\mathbf{E}$ is an orthogonal matrix of columns of eigenvectors and $\mathbf{D}$ is a diagonal matrix of eigenvalues. Following along with [9], our new mixing matrix becomes

$$\tilde{\mathbf{A}} = \mathbf{V}\mathbf{A} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{A} \tag{2.11}$$

and our whitened data is

$$\mathbf{X}_w = \mathbf{VAS} = \tilde{\mathbf{A}}\mathbf{S}. \tag{2.12}$$

We can see that this new data is whitened because

$$\begin{aligned} E\{\mathbf{X}_w\mathbf{X}_w^T\} &= \mathbf{V}E\{\mathbf{X}\mathbf{X}^T\}\mathbf{V}^T \\ &= (\mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T)(\mathbf{E}\mathbf{D}\mathbf{E}^T)(\mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T) \\ &= \mathbf{E}\mathbf{D}^{-1/2}(\mathbf{E}^T\mathbf{E})\mathbf{D}(\mathbf{E}^T\mathbf{E})\mathbf{D}^{-1/2}\mathbf{E}^T \\ &= \mathbf{E}\mathbf{D}^{-1/2}\mathbf{D}\mathbf{D}^{-1/2}\mathbf{E}^T \qquad \text{(since } \mathbf{E} \text{ is orthogonal)} \\ &= \mathbf{E}\mathbf{E}^T \\ &= \mathbf{I}. \end{aligned} \tag{2.13}$$

Also notice

$$\begin{aligned} E\{\mathbf{X}_w\mathbf{X}_w^T\} &= \tilde{\mathbf{A}}E\{\mathbf{S}\mathbf{S}^T\}\tilde{\mathbf{A}}^T \\ &= \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \quad \text{(since } \mathbf{S} \text{ is made up of independent components)} \\ &= \mathbf{I} \quad \text{(from Equation 2.13)} \end{aligned}$$

Thus our new mixing matrix $\tilde{\mathbf{A}}$ is orthogonal. A visual interpretation of this concept is shown in Figure 2.2.

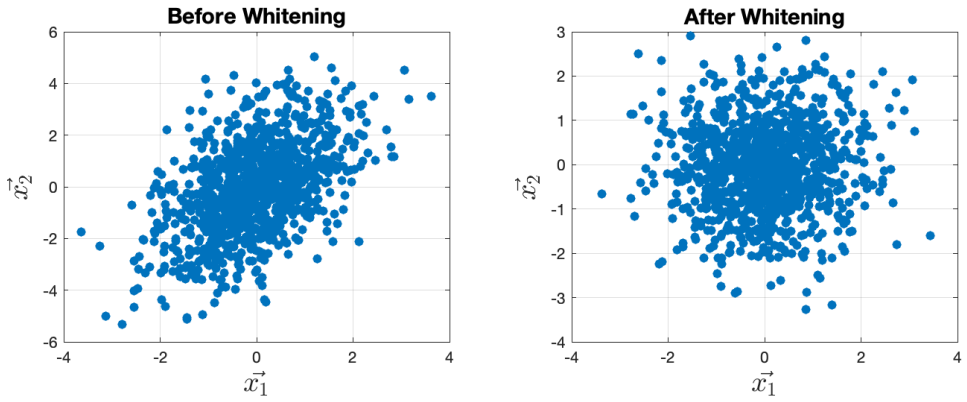

Figure 2.2: Figure comparing distributions of Gaussian random variables before and after whitening transformation. (created by whitening.m)

Notice the whitening transformation spheres the data around the origin.

2.2.4 Non-Gaussian

Another assumption of the ICA model is that the individual independent components $\vec{s_1}, \vec{s_2}, \ldots \vec{s_n}$ must be non-Gaussian. Recall from Section 2.2.3 that we have determined the mixing matrix $\mathbf{A}$ is orthogonal after whitening. Below we will prove that the orthogonal transformation of Gaussian variables has the same distribution as the variables themselves.

Proposition 1. *Suppose we have an orthogonal matrix* $\mathbf{A}$ *and Gaussian variables* $\vec{s_1}^\top, \vec{s_2}^\top, \ldots, \vec{s_n}^\top$, *which each have a mean* $\mu = 0$ *and variance* $\sigma = 1$, *and are independent. Then each of the rows of* $\mathbf{AS}$ *are Gaussian.*

Proof. We will show the $\mathbb{R}^2$ case, but the proof could be extended. Suppose $\mathbf{A} \in \mathbb{R}^2$ is orthogonal and

$$\mathbf{S} = \begin{bmatrix} \vec{s_1}^\top \\ \vec{s_2}^\top \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2n} \end{bmatrix} \tag{2.14}$$

such that $\vec{s_1}^\top, \vec{s_2}^\top$ are Gaussian for every time sample. Notice we are only looking at the first time sample $\begin{bmatrix}\sigma_{11}\\ \sigma_{21}\end{bmatrix}$ so we will treat $\begin{bmatrix}\vec{s_1}^\top\\ \vec{s_2}^\top\end{bmatrix} = \begin{bmatrix}\sigma_{11}\\ \sigma_{21}\end{bmatrix}$ to simplify the notation for the calculations. Since we assumed that $\vec{s_1}^\top, \vec{s_2}^\top$ are independent, have mean $\mu = 0$, and variance $\sigma = 1$ we have

$$p(\sigma_{11}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma_{11}^2}{2}}, \quad p(\sigma_{21}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma_{21}^2}{2}} \tag{2.15}$$

and

$$p(\sigma_{11}\sigma_{21}) = p(\sigma_{11})p(\sigma_{21}) = \frac{1}{2\pi} e^{-\frac{(\sigma_{11}^2+\sigma_{21}^2)}{2}}. \tag{2.16}$$

Then $\mathbf{X} = \mathbf{AS} = \begin{bmatrix}\alpha_{11}\vec{s_1}^\top + \alpha_{12}\vec{s_2}^\top\\ \alpha_{21}\vec{s_1}^\top + \alpha_{22}\vec{s_2}^\top\end{bmatrix} = \begin{bmatrix}\alpha_{11}\sigma_{11} + \alpha_{12}\sigma_{21}\\ \alpha_{21}\sigma_{11} + \alpha_{22}\sigma_{21}\end{bmatrix}$ and $p(\vec{x_1}\vec{x_2}) = \frac{1}{2\pi} e^{-\frac{\vec{x_1}^2+\vec{x_2}^2}{2}}$ where

$$\begin{aligned}
\vec{x_1}^2 + \vec{x_2}^2 &= (\alpha_{11}\sigma_{11} + \alpha_{12}\sigma_{21})^2 + (\alpha_{21}\sigma_{11} + \alpha_{22}\sigma_{21})^2 \\
&= \alpha_{11}^2\sigma_{11}^2 + 2\alpha_{11}\sigma_{11}\alpha_{12}\sigma_{21} + \alpha_{12}^2\sigma_{21}^2 + \alpha_{21}^2\sigma_{11}^2 + 2\alpha_{21}\sigma_{11}\alpha_{22}\sigma_{21} + \alpha_{22}^2\sigma_{21} \\
&= (\alpha_{11}^2 + \alpha_{21}^2)\sigma_{11}^2 + (\alpha_{12}^2 + a_{22}^2)\sigma_{21}^2 + 2(\alpha_{11}\alpha_{12} + \alpha_{21}\alpha_{22})\sigma_{11}\sigma_{21} \\
&= \sigma_{11}^2 + \sigma_{21}^2 \qquad \text{since } \mathbf{A} \text{ is orthogonal} \\
&\implies p(\vec{s_1}\vec{s_2}) = p(\vec{x_1}\vec{x_2})
\end{aligned} \tag{2.17}$$

Thus we have shown that the orthogonal transformation of Gaussian independent variables has the same distribution as the Gaussian variables themselves. □

Under the assumptions of the ICA model this means that our mixed signals are the result of an orthogonal transformation of the original source signals.

Now we will use the Central Limit Theorem to argue how to estimate the individual independent components given that they are non-Gaussian.

Theorem 2.1 (Central Limit Theorem [20])**.** *Let $x_1, x_2, \ldots, x_n$ be independent and identi-*

cally distributed random variables with $E(X_i) = \mu_i$ *and* $Var(X_i) = \sigma^2 < \infty$. *Define*

$$U_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} = \frac{\frac{1}{n}\sum_{i=1}^{n} X_i}{\sigma/\sqrt{n}}. \tag{2.18}$$

Then the probability density of U_n *converges to the standard normal distribution function as* $n \to \infty$. *So we have that*

$$\lim_{n\to\infty} P(U_n \leq u) = \int_{-\infty}^{u} \frac{1}{2\pi} e^{-t^2/2} dt \quad \forall u. \tag{2.19}$$

Suppose $\mathbf{X} = \mathbf{AS}$ is made up of n by 1 vectors of mixed independent components and suppose $\mathbf{A}^{-1}$ exists and $\mathbf{W} = \mathbf{A}^{-1}$. Then $\mathbf{S} = \mathbf{WX}$. If we want to estimate one of the individual components, consider $\vec{y}^{\top} = \vec{w_j}^{\top}\mathbf{X}$, a linear combination of the mixed components, where $\vec{w_j}^{\top}$ is one of the rows of $\mathbf{W}$. Then

$$\vec{y}^{\top} = \vec{w_j}^{\top}\mathbf{X} = \vec{w_j}^{\top}\mathbf{AS}. \tag{2.20}$$

It follows that $\vec{y}^{\top}$ is a linear combination of the individual components $\mathbf{S}$ with weights $\vec{w}^{\top}\mathbf{A}$. From Theorem 2.1 we know that the sum of independent random variables converges to a Gaussian distribution as n goes to infinity. So the distribution of the entries of $\vec{y}^{\top}$ is closer to a Gaussian distribution than any individual component $\vec{s_i}^{\top}$, and the Gaussianity is minimized (i.e., the distribution is furthest from Gaussian) when $\vec{y}^{\top} = \vec{s_i}^{\top}$. In this case $\vec{w_j}^{\top}\mathbf{A}$ has one non-zero component. Thus, to recover one of the individual components we need to maximize the non-Gaussianity of $\vec{y}^{\top} = \vec{w_j}^{\top}\mathbf{X}$ [9].

We can illustrate why this concept of non-Gaussian variables is important with an example. Suppose we have the orthogonal mixing matrix $\mathbf{A} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and we consider the joint

distribution of independent components $\vec{s_1}$ and $\vec{s_2}$ generated from a Gaussian distribution. As described in Proposition 1, we can see in Figure 2.3 that the distribution of $\mathbf{X}$ resulting from the orthogonal transformation $\mathbf{A}$ has the same distribution as $\mathbf{S}$.

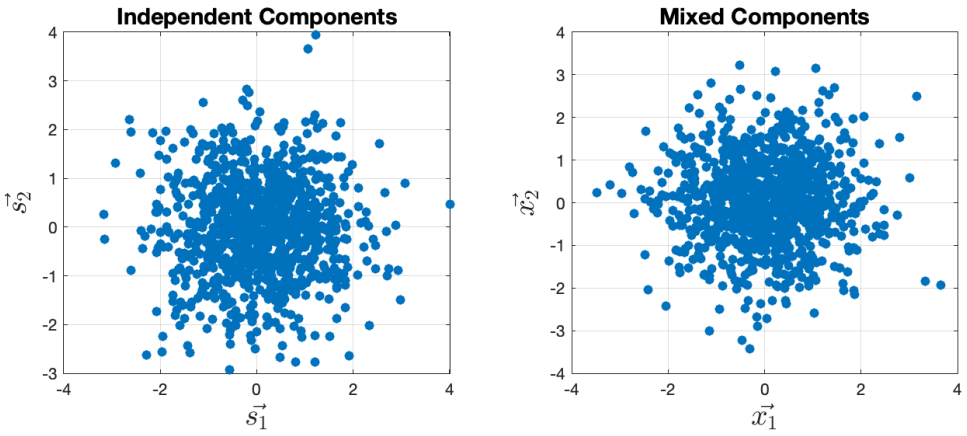

Figure 2.3: Joint distributions of Gaussian random variables before and after linear transformation by mixing matrix $\mathbf{A}$. (created by joint_distributions.m)

The joint distribution of Gaussian random variables after a linear transformation by $\mathbf{A}$ does not reveal any new information about the structure of the mixing matrix $\mathbf{A}$ so we cannot estimate the independent components $\mathbf{S}$.

Now consider the joint distribution of independent components $\vec{s_1}$ and $\vec{s_2}$ generated from a uniform distribution.

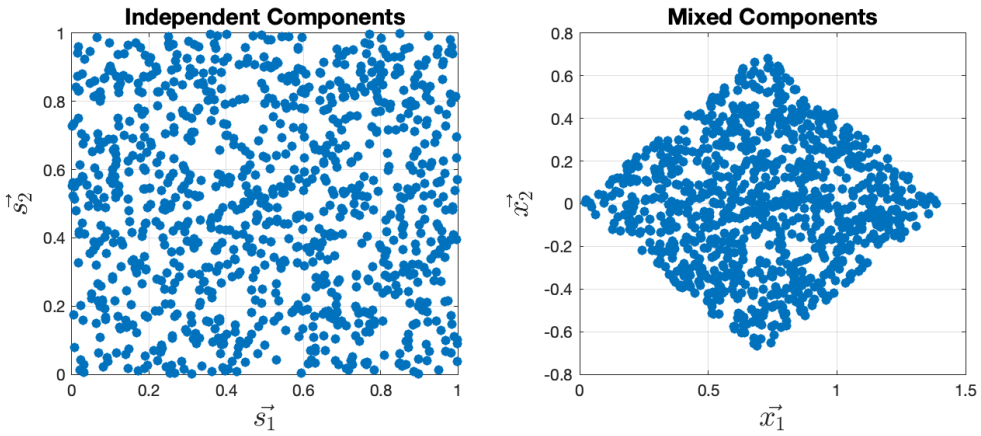

Figure 2.4: Joint distributions of non-Gaussian random variables before and after linear transformation by mixing matrix $\mathbf{A}$. (created by joint_distributions.m)

Notice that the joint distribution of $\vec{s_1}, \vec{s_2}$ is uniform on a square while the joint distribution of $\vec{x_1}, \vec{x_2}$ is uniform on a parallelogram. Thus we can determine the matrix $\mathbf{A}$ by estimating the edges of the parallelogram which are the directions of the columns of $\mathbf{A}$. This demonstrates why we need the independent components to be non-Gaussian in order to separate the mixed components.

2.2.5 Limitations

One limitation of using the ICA model is that we cannot determine the amplitudes of the independent components since $\mathbf{S}$ and $\mathbf{A}$ are unknown. To see this suppose one of our individual components are multiplied by scalars $\lambda_1, \ldots, \lambda_n$ such that

$$\tilde{\mathbf{S}} = \begin{bmatrix} \lambda_1 \vec{s_1}^\top \\ \lambda_2 \vec{s_2}^\top \\ \cdots \\ \lambda_n \vec{s_n}^\top \end{bmatrix}. \tag{2.21}$$

Then from our mixed signal $\mathbf{X}$ we would estimate $\tilde{\mathbf{A}}$ to be

$$\tilde{\mathbf{A}} = \begin{bmatrix} \frac{1}{\lambda_1}\alpha_{11} & \frac{1}{\lambda_2}\alpha_{12} & \dots & \frac{1}{\lambda_n}\alpha_{1n} \\ \frac{1}{\lambda_1}\alpha_{21} & \frac{1}{\lambda_2}\alpha_{22} & \dots & \frac{1}{\lambda_n}\alpha_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{1}{\lambda_1}\alpha_{n1} & \frac{1}{\lambda_2}\alpha_{n2} & \dots & \frac{1}{\lambda_n}\alpha_{nn} \end{bmatrix} \tag{2.22}$$

and ultimately get $\mathbf{X} = \tilde{\mathbf{A}}\tilde{\mathbf{S}} = \mathbf{AS}$. Since we will not be able to determine the amplitude of the signal, we can further simplify the problem by assuming the individual components have a unit variance, so $E\{\vec{s_i}^{\top^2}\} = 1$. Notice even with this distinction, we cannot determine the sign of the amplitude. We will show an example of this occurring in a later section.

Another limitation of the ICA model is that we cannot determine the order in which we estimate the individual components. Suppose we have a permutation matrix $\mathbf{P}$ such that our independent components become $\mathbf{PS}$. Then we would have a new mixing matrix $\mathbf{AP}^{-1}$ such that $\mathbf{X} = \mathbf{AP}^{-1}\mathbf{PS}$. This means that if we are able to successfully recover the independent components and we want to perform further analysis that is repeatable across runs, we may need to do some post-processing to match up similar separated signals. [9]

2.3 Measures of non-Gaussianity

We consider two measures of non-Gaussianity; kurtosis and negative entropy.

2.3.1 Kurtosis

As explained in the previous section we want to maximize how non-Gaussian our random variables are. One measure of non-Gaussianity is kurtosis. Kurtosis is defined as

$$kurt(y) = E[y^4] - 3(E[y^2])^2. \tag{2.23}$$

If our data follows the necessary assumptions of ICA, we have that the variance is 1 so $E[y^2] = 1$ and

$$kurt(y) = E[y^4] - 3. \tag{2.24}$$

Suppose y is Gaussian, then

$$kurt(y) = E[y^4] - 3 = 3(E[y^2])^2 - 3 = 3((E[y^2])^2 - 1) = 3(1 - 1) = 0.$$

So for non-Gaussian y we want kurtosis to be nonzero. Suppose we pose this as an optimization problem to maximize $|kurt(\vec{w}^\top \mathbf{X})|$. Then we can use a basic gradient ascent algorithm as a simple method to attempt finding the maximum by deriving $\nabla_{\vec{w}} kurt(\vec{w}^\top \mathbf{X})$. We will first prove an intermediate step needed for the derivation.

Proposition 2. *Suppose* $\vec{w} = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix}$ *and* $\mathbf{X} = \begin{bmatrix} \chi_{11} & \chi_{12} & \cdots & \chi_{1k} \\ \chi_{21} & \chi_{22} & \cdots & \chi_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ \chi_{n1} & \chi_{n2} & \cdots & \chi_{nk} \end{bmatrix}$ *where* $E[\chi_{ij}^2] = 1$ *and* $E[\chi_{ij}\chi_{sj}] = 0$ *for* $i \neq s$. *Then* $E[(\vec{w}^\top \mathbf{X})\mathbf{X}] = \vec{w}$.

Proof. Let us consider the jth column of $\mathbf{X}$, $[\chi_{1j}\ \chi_{2j} \dots\ \chi_{nj}]^\top$ (i.e., one time sample) to

simplify the notation of the calculations.

$$
\begin{aligned}
E[(\vec{w}^\top \mathbf{X})\mathbf{X}] &= E\left(\left[\textstyle\sum_{i=1}^{n} \omega_i \chi_{ij}\right]\right)\mathbf{X} \\
&= E\begin{bmatrix} \sum_{i=1}^{n} \omega_i \chi_{ij}\chi_{1j} \\ \cdots \\ \sum_{i=1}^{n} \omega_i \chi_{ij}\chi_{nj} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{n} \omega_i E[\chi_{ij}\chi_{1j}] \\ \cdots \\ \sum_{i=1}^{n} \omega_i E[\chi_{ij}\chi_{nj}] \end{bmatrix} \\
&= \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} \quad \text{Since } E[\chi_{ij}^2] = 1 \text{ and } E[\chi_{ij}\chi_{sj}] = 0 \text{ for } i \neq s \\
&= \vec{w} \qquad (2.25)
\end{aligned}
$$

□

Now we can move on to the derivation of the gradient. Notice $\vec{w}^\top \in \mathbb{R}_{1\times k}$ and $\vec{w}^\top \mathbf{X} \in \mathbb{R}_{1\times n}$. Then

$$
\begin{aligned}
\frac{\partial kurt(\vec{w}^\top \mathbf{X})}{\partial \vec{w}} &= \frac{\partial}{\partial \vec{w}}\left[E((\vec{w}^\top \mathbf{X})^4) - 3(E((\vec{w}^\top \mathbf{X})^2))^2\right] \\
&= \frac{\partial}{\partial \vec{w}}\left[E((\vec{w}^\top \mathbf{X})^4)\right] - \frac{\partial}{\partial \vec{w}}\left[3(E((\vec{w}^\top \mathbf{X})^2))^2\right] \\
&= 4E((\vec{w}^\top \mathbf{X})^3\mathbf{X}) - [3 \cdot 2E(\vec{w}^\top \mathbf{X})^2 \cdot 2E(\vec{w}^\top \mathbf{X})\mathbf{X}] \qquad \text{(using the chain rule)} \\
&= 4E((\vec{w}^\top \mathbf{X})^3\mathbf{X}) - 12E((\vec{w}^\top \mathbf{X})^2)E\left((\vec{w}^\top \mathbf{X})\mathbf{X}\right) \\
&= 4\left[E((\vec{w}^\top \mathbf{X})^3\mathbf{X}) - 3\vec{w}E((\vec{w}^\top \mathbf{X})^2)\right] \qquad \text{(using Equation 2.25)} \\
&= 4\left[E((\vec{w}^\top \mathbf{X})^3\mathbf{X}) - 3\vec{w}\right] \qquad \text{(since } E((\vec{w}^\top \mathbf{X})^2) = 1) \qquad (2.26)
\end{aligned}
$$

Then

$$\nabla_{\vec{w}}|kurt(\vec{w}^\top \mathbf{X})|= \frac{\partial|kurt(\vec{w}^\top \mathbf{X})|}{\partial \vec{w}} = 4\text{sign}(kurt(\vec{w}^\top \mathbf{X}))\left[E((\vec{w}^\top \mathbf{X})^3 \mathbf{X}) - 3\vec{w}\right] \tag{2.27}$$

where we add in the sign$(kurt(\vec{w}^\top \mathbf{X}))$ term to account for the absolute value.

We have two immediate approaches for optimization algorithms resulting from the gradient. We will further discuss a gradient ascent algorithm and a fixed point iteration in Section 2.4.

2.3.2 Negative Entropy

Following [8], we define the differential entropy of a random vector $\vec{y}$ as

$$H(\vec{y}) = -\int f(\vec{y}) \log f(\vec{y}) dy, \tag{2.28}$$

where f is the probability density function. From this we get the negative entropy ("negentropy"), which is given by

$$J(\vec{y}) = H(\vec{y}_{gauss}) - H(\vec{y}), \tag{2.29}$$

where $\vec{y}_{Gauss}$ is a Gaussian random variable with the same covariance matrix as $\vec{y}$ [8]. We can then have the following approximation of negentropy

$$J(\vec{y}) \approx [E[G(\vec{y})] - E[G(\vec{\nu})]]^2 \tag{2.30}$$

where G is a non-quadratic function and $\vec{\nu}$ is a Gaussian random variable with a zero mean and unit variance. According to [8], a common choice of G is

$$G(\vec{y}) = \log\cosh(\vec{y}) \quad \text{and} \quad g = G' = \tanh(\vec{y}).$$

Then for our optimization problem we want to maximize $E[G(\vec{w}^\top \mathbf{X})]$. Using the same setup as in Equation (2.25) we have

$$\begin{aligned}
\frac{\partial E[G(\vec{w}^\top \mathbf{X})]}{\partial \vec{w}} &= \frac{\partial}{\partial \vec{w}} E\left(\begin{bmatrix} \sum_{i=1}^n \omega_i \chi_{i1} & \cdots & \sum_{i=1}^n \omega_i \chi_{ik} \end{bmatrix}\right) \\
&= \frac{\partial}{\partial \vec{w}} \begin{bmatrix} \vec{w}^\top E \begin{bmatrix} \chi_{1i} \\ \vdots \\ \chi_{n1} \end{bmatrix} \dots \vec{w}^\top \begin{bmatrix} \chi_{1k} \\ \vdots \\ \chi_{nk} \end{bmatrix} \end{bmatrix} \\
&= E[g(\vec{w}^\top \mathbf{X})\mathbf{X}]. \qquad (2.31)
\end{aligned}$$

We can again describe a gradient ascent and fixed point iteration from this calculation as outlined further in Section 2.4.

2.4 Numerical Algorithms

We will now introduce the numerical algorithms implemented in Matlab for the gradient ascent and fixed point iterations of ICA using kurtosis and negentropy for our objective function. For these algorithms we are updating the rows of $\mathbf{W}$, which is an estimate for $\mathbf{A}^{-1}$, at each iteration. We solve for each row of $\mathbf{W}$ separately. So we will repeat the process n times if we have n mixed signals and n source signals. When we are solving for any row of $\mathbf{W}$ other than the first one, we must add in a step to decorrelate the rows of $\mathbf{W}$ using the

Gram-Schmidt process since we want to estimate components which are independent from each other.

Theorem 2.2 (Gram-Schmidt [1]). *Let* $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ *be a set of linearly independent vectors in* W*. Then there exists an orthogonal set of set of vectors* $\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$ *in* W *such that* $\text{span}\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\} = \text{span}\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$*, for any index* q *in the range* $1 \leq q \leq n$ *where*

$$\vec{y}_n = \vec{x_n} - \sum_{i=1}^{n-1} (\vec{x}_n^{\top}) \vec{y}_i \vec{y}_i. \tag{2.32}$$

The gradient ascent algorithms in Section 2.4.1 follow directly from the calculations of the gradients of kurtosis and negentropy. The fixed point iteration algorithms in Section 2.4.2 are derived using Lagrange multipliers.

2.4.1 Gradient Ascent Algorithms

The first algorithm we will present, Algorithm 1, is a standard gradient ascent algorithm to find the directions in which the absolute value of kurtosis is maximized. This will give us the rows of our $\mathbf{W}$ matrix. We also add in the step to decorrelate the rows of $\mathbf{W}$ at each iteration in line 6.

Algorithm 1 Kurtosis Gradient Ascent Algorithm

1: **for** $i = 1 : n$ **do**
2: Choose initial random weight vector $\vec{w}_i$ on unit circle
3: **while** $j <$ maxIter **do**
4: $\vec{w}_i^+ = \vec{w}_i + \alpha \nabla_{\vec{w}_i} |kurt(\vec{w}_i^\top \mathbf{X})| w_i$
5: **if** $i > 1$ **then**
6: $\vec{w}_i^+ = \vec{w}_i - \sum_{k=1}^{i-1} \vec{w}_i^\top \vec{w}_k \vec{w}_k$
7: **end if**
8: $\vec{w}_i{}^+ = \frac{\vec{w}_i{}^+}{\|\vec{w}_i{}^+\|}$
9: **if** $|(\vec{w}_i \cdot \vec{w}_i{}^+) - 1| <$ tol **then**
10: break
11: **end if**
12: **end while**
13: **end for**

Notice line 6 means we use Gram-Schmidt at every iteration except when solving for the first row of $\mathbf{W}$. Also line 9 provides our convergence criterion where we continue this iterative process until the dot product between the $\vec{w}$ from our previous and current iterations is 1, meaning the vectors point in the same direction along the unit circle.

Next we have a gradient ascent algorithm using negentropy as our objective function as shown in Algorithm 2. Notice the only line that changes is the step for updating the weight vector on line 4.

Algorithm 2 Negentropy Gradient Ascent Algorithm

for $i = 1 : n$ **do**
  Choose initial random weight vector $\vec{w}_i$ on unit circle
  while $j <$ maxIter **do**
    $\vec{w}_i^+ = \vec{w}_i + \alpha \nabla_{\vec{w}_i} E[G(\vec{w}^\top \mathbf{X})] w_i$
    if $i > 1$ **then**
      $\vec{w}_i^+ = \vec{w}_i - \sum_{k=1}^{i-1} \vec{w}_i^\top \vec{w}_k \vec{w}_k$
    end if
    $\vec{w}_i{}^+ = \frac{\vec{w}_i{}^+}{\|\vec{w}_i{}^+\|}$
    if $|(\vec{w}_i \cdot \vec{w}_i{}^+) - 1| <$ tol **then**
      break
    end if
  end while
end for

In testing these algorithms, we want to be able to evaluate the performance of the separation of signals. It is beneficial in this testing and evaluation phase to decrease the overall number of unknowns involved so the signal separation is not negatively affected by the miscalculation of a parameter in the optimization process. With the gradient ascent algorithms, we must tune the step size α using a line search method. This parameter determines the speed at which our algorithm explores the space of possible solutions, and can even lead to convergence to different local optimal points. In calculating α we also have a risk of overstepping the optimal solution. This means that depending on our specific problem, we may not converge at all or we may converge to the wrong solution. Using a method that does not involve this step size parameter α may avoid scenarios where the signals are not separated due to overstepping the optimal solution.

2.4.2 Fixed Point Iteration Algorithms

Consider the general form of constrained optimization problems using the Karush-Kuhn-Tucker Conditions [10]

$$\begin{aligned} \max f(\mathbf{x}) \quad \text{s.t.} \quad & g(\mathbf{x}) = 0, \\ & h(\mathbf{x}) \leq 0. \end{aligned} \tag{2.33}$$

We can then arrange our optimization problems using kurtosis and negentropy as objective functions in this form.

Kurtosis Fixed Point Iteration

Considering our optimization problem in the context of Equation (2.33), we have

$$\begin{aligned} f(\mathbf{X}) &= |\text{kurt}(\vec{w}^\top \mathbf{X})| \\ g(\mathbf{X}) &= E[(\vec{w}^\top \mathbf{X})^2] - 1 \end{aligned}$$

and our optimization problem becomes

$$\max_{\vec{w}} |\text{kurt}(\vec{w}^\top \mathbf{X})| \quad \text{s.t.} \quad E[(\vec{w}^\top \mathbf{X})^2] - 1 = \|\vec{w}\|_2^2 - 1 = 0. \tag{2.34}$$

Notice we do not have any inequality constraints so there is no $h(\mathbf{x})$ function in this case. Using Lagrange multipliers it follows that at the optimal solution, we have

$$\begin{aligned}
&\nabla_{\vec{w}}|\text{kurt}(\vec{w}^\top \mathbf{X})| = \lambda \nabla_{\vec{w}} \left(\|\vec{w}\|_2^2 - 1\right) \\
&4\text{sign}(\text{kurt}(\vec{w}^\top \mathbf{X})) \left(E[(\vec{w}^\top \mathbf{X})^3 \mathbf{X}] - 3\vec{w}\right) = 2\lambda \vec{w} \\
&\implies \vec{w} = \frac{2}{\lambda \text{sign}(\text{kurt}(\vec{w}^\top \mathbf{X}))} \left(E[(\vec{w}^\top \mathbf{X})^3 \mathbf{X}] - 3\vec{w}\right)
\end{aligned} \tag{2.35}$$

Since we normalize $\vec{w}$ at each iteration, we can drop the constants and get the fixed point iteration

$$\vec{w} = \text{sign}(\text{kurt}(\vec{w}^\top \mathbf{X})) \left(E[(\vec{w}^\top \mathbf{X})^3 \mathbf{X}] - 3\vec{w}\right) \tag{2.36}$$

This leads to a fixed point iteration where we have an explicit formula to update $\vec{w}$ at each iteration. Then we obtain Algorithm 3 in a similar manner to the gradient ascent algorithms where only line 4 of the algorithm has changed.

Algorithm 3 Kurtosis Fixed Point Iteration Algorithm

for $i = 1 : n$ **do**
  Choose initial random weight vector $\vec{w}_i$
  while $j <$ maxIter **do**
    $\vec{w}_i^+ = \text{sign}(\text{kurt}(\vec{w}^\top \mathbf{X}))E[(\vec{w}_i{}^\top \mathbf{X})^3 \mathbf{X}] - 3\vec{w}_i$
    if $i > 1$ **then**
      $\vec{w}_i^+ = \vec{w}_i - \sum_{k=1}^{i-1} \vec{w}_i^\top \vec{w}_k \vec{w}_k$
    end if
    $\vec{w}_i{}^+ = \frac{\vec{w}_i{}^+}{\|\vec{w}_i{}^+\|}$
    if $|(\vec{w}_i \cdot \vec{w}_i{}^+) - 1| <$ tol **then**
      break
    end if
  end while
end for

Negentropy Fixed Point Iteration

Applying the same method as in Section 2.4.2 for the maximization of negentropy, we get the following constrained optimization problem:

$$\max_{\vec{w}} E[G(\vec{w}^\top \mathbf{X})] \quad \text{s.t.} \quad E[(\vec{w}^\top \mathbf{X})^2] - 1 = \|\vec{w}\|_2^2 - 1 = 0. \tag{2.37}$$

Similarly to the kurtosis fixed point iteration, we use Lagrange multipliers to find the optimal solution:

$$\begin{aligned} &\nabla_{\vec{w}} E[G(\vec{w}^\top \mathbf{X})\mathbf{X}] = \lambda \nabla_{\vec{w}} (\|\vec{w}\|2^2 - 1) \\ &\implies E[g(\vec{w}^\top \mathbf{X})\mathbf{X}] - 2\lambda \vec{w} = 0 \end{aligned} \tag{2.38}$$

Following the derivation in [8], suppose we utilize the Newton iteration method to find the $\vec{w}$ which satisfies Equation (2.38).

$$\begin{aligned} \frac{\partial}{\partial \vec{w}} \left(E[g(\vec{w}^\top \mathbf{x})\mathbf{X}] - 2\lambda \vec{w} \right) &= E[g'(\vec{w}^\top \mathbf{X})\mathbf{X}\mathbf{X}^\top] - 2\lambda \mathbf{I}] \\ &\approx E[g'(\vec{w}^\top \mathbf{X})] E[\mathbf{X}\mathbf{X}^\top] - \beta \mathbf{I}] \quad \text{since our data is whitened} \\ &= E[g'(\vec{w}^\top \mathbf{X})]\mathbf{I} - 2\lambda \mathbf{I} \end{aligned} \tag{2.39}$$

Then from Newton's method,

$$\vec{w}^+ = \vec{w} - \frac{E[g'(\vec{w}^\top \mathbf{X})\mathbf{X}] - 2\lambda \vec{w}}{E[g'(\vec{w}^\top \mathbf{X})] - 2\lambda}. \tag{2.40}$$

As described in [8], we can provide a further simplification by multiplying both sides of this equation by $(2\lambda - E[g'(\vec{w}^\top \mathbf{X})])$, resulting in

$$\vec{w}^+ = E[g(\vec{w}^\top \mathbf{X})\mathbf{X}] - E[g'(\vec{w}^\top \mathbf{X})]\vec{w}. \tag{2.41}$$

We use this formula as our update step for $\vec{w}$ in Algorithm 4.

Algorithm 4 Negentropy Fixed Point Iteration Algorithm

for $i = 1 : n$ **do**
  Choose initial random weight vector $\vec{w}_i$
  while $j <$ maxIter **do**
    $\vec{w}_i^+ = E[g(\vec{w}_i{}^\top \mathbf{X})\mathbf{X}] - E[g'(\vec{w}_i{}^\top \mathbf{X})]\vec{w}_i$
    if $i > 1$ **then**
      $\vec{w}_i^+ = \vec{w}_i - \sum_{k=1}^{i-1} \vec{w}_i^\top \vec{w}_k \vec{w}_k$
    end if
    $\vec{w}^+ = \frac{\vec{w}^+}{\|\vec{w}^+\|}$
    if $|(\vec{w} \cdot \vec{w}^+) - 1| <$ tol **then**
      break
    end if
  end while
end for

One benefit of these fixed point iteration algorithms, Algorithm 3 and Algorithm 4, over the gradient ascent algorithms, Algorithm 1 and Algorithm 2, is that there are fewer unknowns in the optimization algorithm since we do not have the step size parameter αr. We should also keep in mind that the fixed point iterations are not guaranteed to converge.

Chapter 3

Numerical Analysis and Comparison of Algorithms

In this chapter we will compare the fixed point iterations for both kurtosis and negentropy as objective functions for our optimization algorithm. We will compare these algorithms for synthetically mixed signals, signals resulting from simulated pressure waves, and signals resulting from multi-source laboratory experiments. In Section 3.2 we will study the performance of these algorithms on the separation of instantaneous mixtures while in Section 3.3 we will study the performance of these algorithms for non-instantaneous mixtures. We then compare the performance of these algorithms on data obtained from laboratory testing with the acoustic and magnetic sensing fiber [7] in Section 3.4.

The general ICA algorithm described in the following section is an instantaneous linear mixing model, meaning the source signals are assumed to mix in exactly the same way at every time sample. This means that our matrix $\mathbf{A}$ stays constant for the entire time series. While this problem setup may need some modifications for real world applications, we can use it to evaluate different optimization algorithms for simple test cases of synthetic data.

3.1 Method of Comparing Algorithms

Suppose we have two source signals which are sine waves with different frequencies, one that is 100 Hz and the other that is 40 Hz. We will consider a 3 second duration of the signal with a sampling frequency of $F_s = 666.6$ Hz. So we have

$$s_{true1}(t) = \sin(2\pi(100t)), \quad s_{true2} = \sin(2\pi(40t)).$$

In Figure 3.1 we show these two sine wave source signals as well as two mixed signals generated by multiplying the source signals by a 2 by 2 mixing matrix.

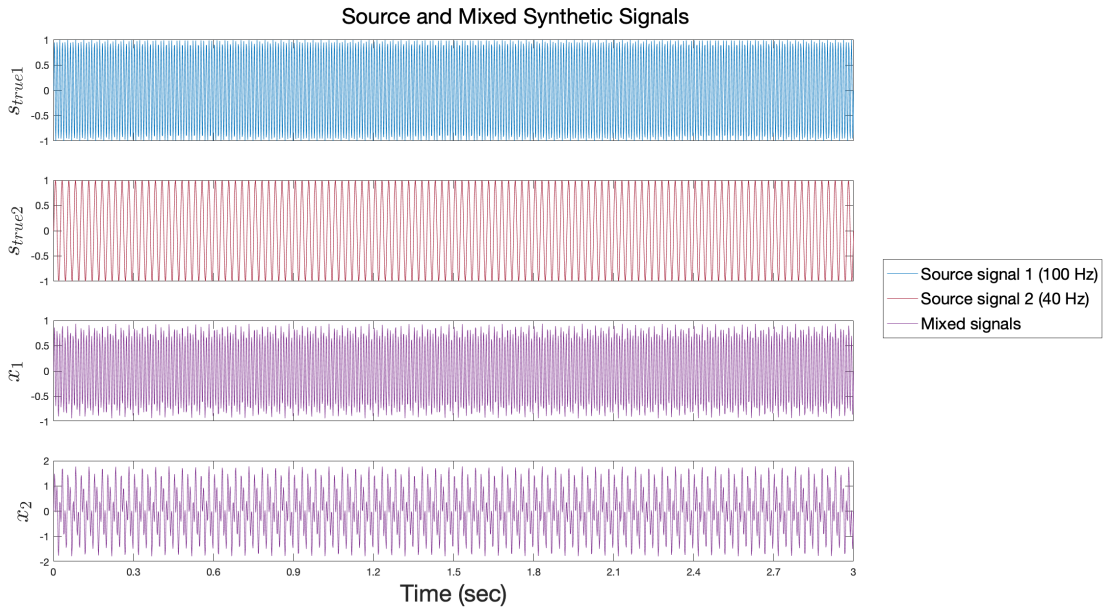

Figure 3.1: Figure of source signals, sine waves with frequencies of 100 Hz and 40 Hz, and mixed signals generated by multiplying the source signals by a mixing matrix. (created by sine_nonoise.m)

In this example, our signal separation goal is to take the two mixed signals and estimate the original sine wave source signals. This simple single-frequency problem allows us to easily

and automatically evaluate the performance of the fixed point iterations using the following process:

1. Create the two source signals

2. Mix the two source signals with a 2×2 random mixing matrix (add noise ϵ after mixing if applicable) resulting in

$$x_1(t) = \alpha_{11} s_{true1}(t) + \alpha_{12} s_{true2}(t) + \epsilon, \quad x_2(t) = \alpha_{21} s_{true1}(t) + \alpha_{22} s_{true2}(t) + \epsilon$$

3. Perform signal separation with kurtosis and negentropy fixed point iterations resulting in $s_1(t), s_2(t)$ the estimated independent components.

4. Calculate the Fast Fourier Transform of each estimated independent component

$$s_1(\omega) = \int_{-\infty}^{\infty} s_1(t) e^{-i\omega t} dt, \quad s_2(\omega) = \int_{-\infty}^{\infty} s_2(t) e^{-i\omega t} dt$$

5. Find the frequencies with the top two peak amplitudes for each estimated source

   $\hat{s_1}(\omega_1)$: amplitude of highest peak frequency for estimated source 1

   $\hat{s_1}(\omega_2)$: amplitude of 2nd highest peak frequency for estimated source 1

6. Reorder the recovered estimated signals based on the frequencies of the top two peak

amplitudes.

$\hat{s_1}(\omega_1^{(1)})$: amplitude of highest peak frequency for estimated source corresponding to 1st original source (100 Hz)

$\hat{s_1}(\omega_2^{(1)})$: amplitude of 2nd highest peak frequency for estimated source corresponding to 2nd original source (40 Hz)

Recall that we are considering the mixture of two single frequency sources and the order of the estimated signals after separation is unknown. We can find the corresponding source signal for each estimated signal by finding which frequency has the highest amplitude after applying a Fourier transform to the estimated signals.

7. Calculate the relative magnitude of the peak frequency amplitudes for each estimated signal

$$\left|\frac{\hat{s_1}(\omega_2^{(1)})}{\hat{s_1}(\omega_1^{(1)})}\right|, \quad \left|\frac{\hat{s_2}(\omega_2^{(2)})}{\hat{s_2}(\omega_1^{(2)})}\right|$$

3.1.1 Optimization Landscape

We will first consider the optimization landscape of the proposed problem. Recall that in order to estimate the individual source signals $\vec{s_1}$ and $\vec{s_2}$, we define an optimization algorithm that maximizes the non-Gaussianity of the estimated individual signals. Our two choices for objective functions are kurtosis (2.23) and negentropy (2.30).

In the case of kurtosis as our cost function, we want to maximize the absolute value of kurtosis in order minimize how close our estimated signals are to Gaussian. As we go through the Kurtosis Fixed Point Iteration (Algorithm 3) for this example, we calculate two weight vectors $\vec{w_1}^\top$ and $\vec{w_2}^\top$. The optimization landscape can then be described as a function of the

angle of $\vec{w}^\top$ in radians (described in polar coordinates) and $|\text{kurt}(\vec{w}^\top \mathbf{X})|$. Notice also our weight vectors are normalized so they are projected onto the unit circle. Figure 3.2 depicts this optimization landscape with the convergence of each weight vector $\vec{w}^\top$ to a maximum of $|\text{kurt}(\vec{w}^\top \mathbf{X})|$.

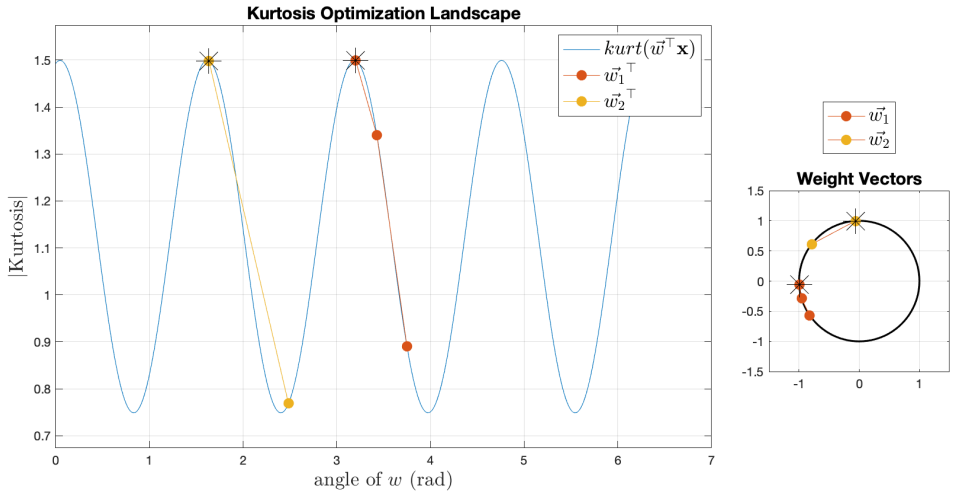

Figure 3.2: Figure of kurtosis optimization landscape and convergence of weight vectors for example with the mixing of two sine waves with frequencies 100 Hz and 40 Hz. (created by sine_nonoise.m)

In the case of negentropy as our cost function, we want to maximize the expected value of $G(\vec{w}^\top \mathbf{X})$ in order minimize how Gaussian our estimated signals are using the Negentropy Fixed Point Iteration Algorithm (Algorithm 4). The optimization landscape be described similarly to the kurtosis optimization landscape as a function of the angle of $\vec{w}^\top$ in radians and $E[G(\vec{w}^\top \mathbf{X})]$ with the weight vectors projected onto the unit circle. Figure 3.3 depicts this optimization landscape with the convergence of each weight vector $\vec{w}^\top$ to a maximum of $E[G(\vec{w}^\top \mathbf{X})]$.

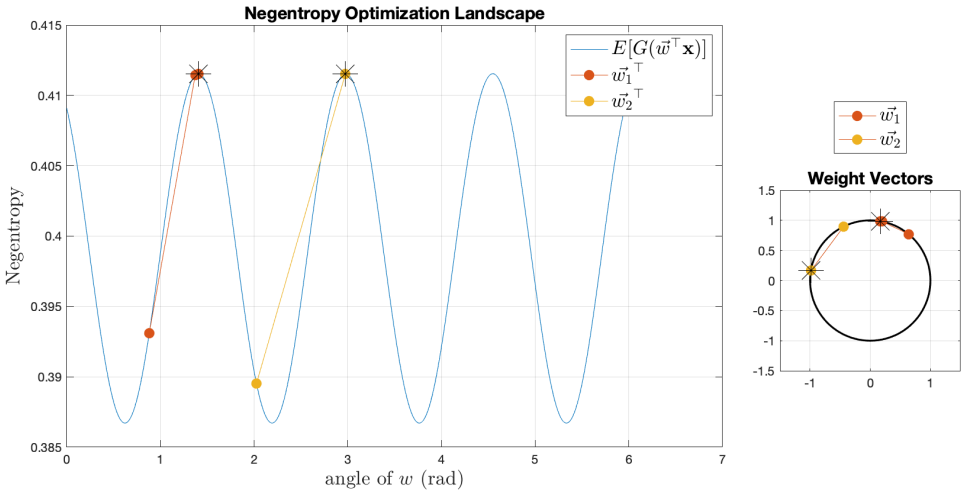

Figure 3.3: Figure of negentropy optimization landscape and convergence of weight vectors for example with the mixing of two sine waves with frequencies 100 Hz and 40 Hz. (created by sine_nonoise.m)

Depending on the initial guess of $\vec{w}$, we may get different separation results, so we need to run the algorithms many times to understand the distribution of performance. Now that we have looked at the optimization landscape and the convergence of both the Kurtosis and Negentropy Fixed Point Iteration Algorithms (Algorithms 3, 4), we will consider the performance of the two fixed point iterations over many runs of the algorithms as described in Section 3.1.

3.2 Instantaneous Mixtures

We will now study the example described in Section 3.1 for synthetic instantaneous mixtures with and without noise being added after mixing.

3.2.1 Instantaneous Mixture Without Noise

Suppose we have the example described in Section 3.1 with no noise added after mixing the two sources. Following the 7-step procedure for testing and evaluation the two single frequency sources we do the mixing and separation process 1000 times to evaluate the overall performance of the algorithms.

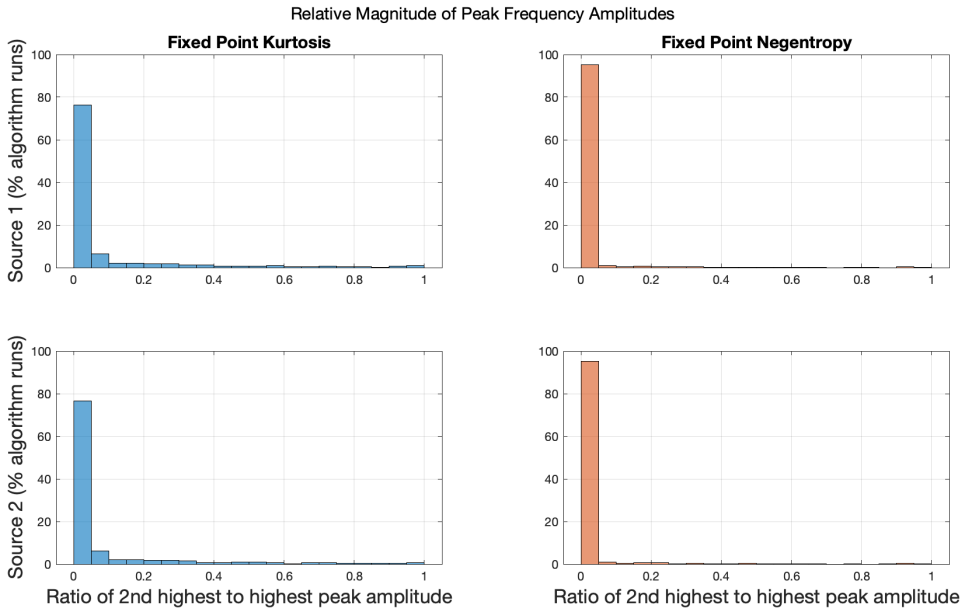

Figure 3.4: Relative magnitude of peak frequency amplitudes after signal separation by kurtosis and negative entropy fixed point iteration algorithms. Synthetic data produced by mixing two sine waves (100 Hz and 40 Hz) with no noise added. (created by sine_nonoise.m)

In order for the separation to be successful we want the ratio of the 2nd highest to highest peak amplitude to be as small as possible, i.e., close to 0. If the ratios were 1-1 this would mean that the relative magnitudes of the two peak frequency amplitudes are the same and the sources are still heavily mixed. We will define a ratio of 0.1 as successful separation. From Figure 3.4 we can see that the kurtosis fixed point iteration results in successful separation of the signals 82.8% of the time while the negentropy fixed point iteration does so 96.2% of

the time. This suggests that the negentropy algorithm may be a better starting point for us to improve upon.

3.2.2 Instantaneous Mixture With Noise

Now suppose we have the same original signals, but this time we add noise in after mixing them together. The standard deviation of noise distribution considered is 1% of the amplitude of our signal and is taken from a normal distribution.

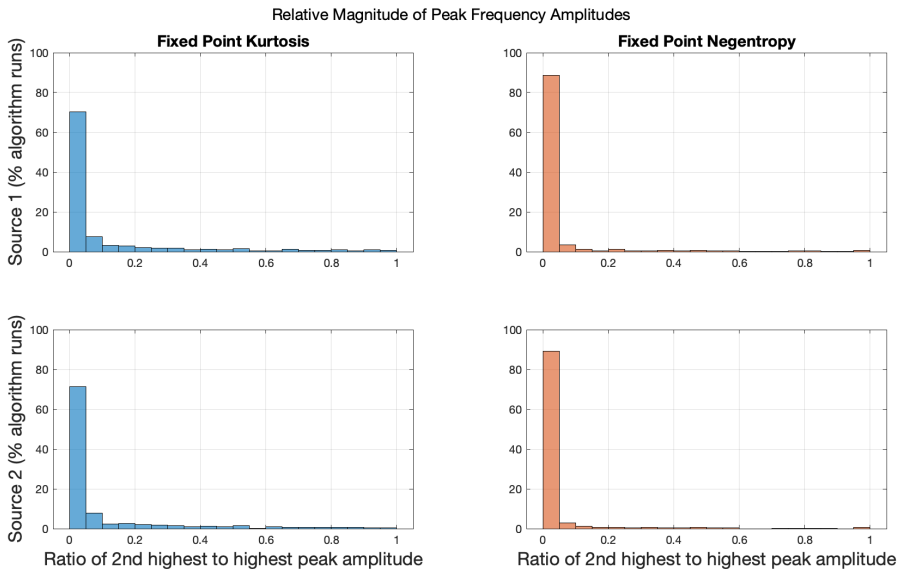

Figure 3.5: Relative magnitude of peak frequency amplitudes after signal separation by kurtosis and negative entropy fixed point iteration algorithms. Synthetic data produced by mixing two sine waves (100 Hz and 40 Hz) with noise added from a normal distribution with a standard deviation that is 1% of the amplitude of the given signal. (created by sine_noise.m)

Notice in Figure 3.5 the overall ratios of the 2nd highest to highest peak amplitudes decreases in comparison to the ratios in Figure 3.4, but this is to be expected since noise was added after the signals were mixed together. We still have 91.8% of the algorithm runs for the negentropy fixed point iteration with ratios below 0.1.

3.3 Non-instantaneous Mixtures

The general ICA problem setup assumes we have instantaneous mixtures meaning **A** stays the same over time and the signals mix in exactly the same way at every time sample. This is unrealistic in real applications because we will have some lag in arrival time depending on the location of our sensors compared to the location of the sources or source signals. In the case of the acoustic and magnetic sensing fiber we have additional limitations because we are measuring different physical quantities with different speeds of propagation. We can use k-Wave [19], a Matlab toolbox, to demonstrate a simple example of this arrival time lag problem and some potential solutions. Consider the problem setup where we have two point pressure sources and two point sensors in a homogeneous medium as shown in Figure 3.6.

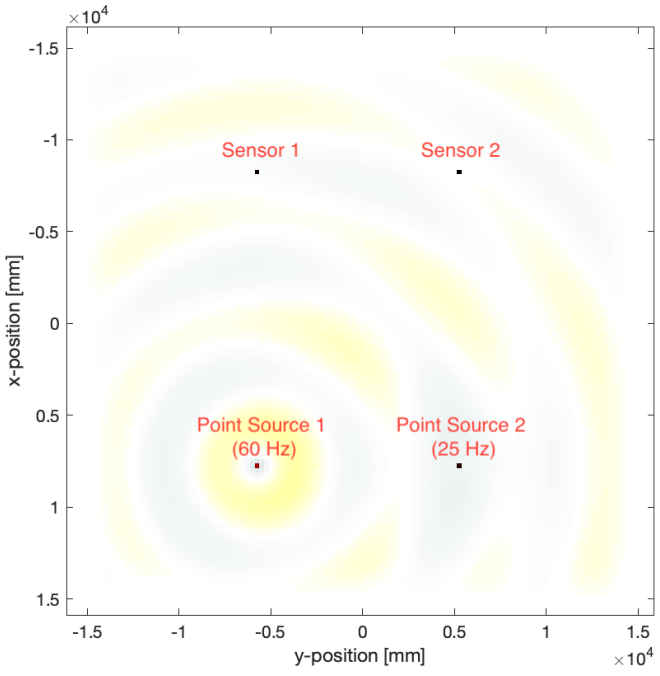

Figure 3.6: Figure of k-Wave example setup; two point sensors and two point sources (60 Hz and 25 Hz). (created by k_wave_sim.m)

Here the point source pressure inputs' frequencies are set to 60 Hz and 25 Hz with amplitudes

of 1. Our true source signals at the point sources are

$$s_{true1}(t) = A_1 \sin(2\pi f_1 t), \quad s_{true2} = A_2 \sin(2\pi f_2 t)). \tag{3.1}$$

The input signals and resulting signals detected by the sensors are as shown below in Figure 3.7.

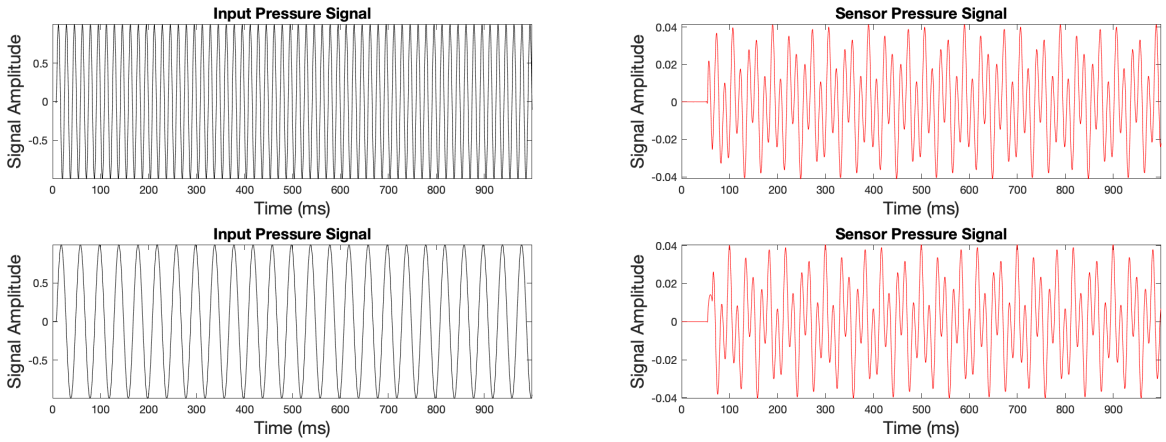

Figure 3.7: Plots of input pressure signals with frequencies set to 60 Hz and 25 Hz and the sensor pressure signal output generated using the kWave Matlab toolbox. (created by k_wave_sim_timelag.m)

We will first try the negentropy fixed point iteration without accounting for any difference in arrival time of the signals to each sensor.

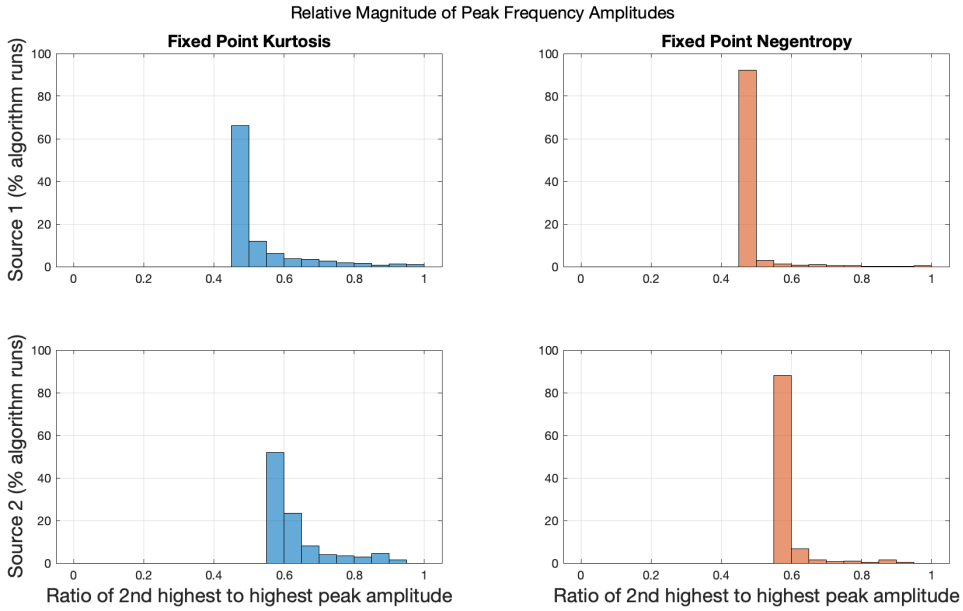

Figure 3.8: Relative magnitude of peak frequency amplitudes after signal separation by kurtosis and negative entropy fixed point iteration algorithms. Synthetic data generated by kWave Matlab toolbox with sine wave input pressure signals (60 Hz and 25 Hz). (created by k_wave_sim_no_timelag.m)

We consider the algorithm to have successfully separated the signals when the ratio is closer to 0. However, in Figure 3.8 we see that the ratios for all algorithms are above 0.45. As expected, these signals are not separated correctly because the ICA algorithm assumes instantaneous mixing, which is not the case in this simulation.

There are several approaches to accounting for this difference in arrival time. These approaches include calculating the cross correlation between two signals and using known properties of the medium and the source signals to directly calculate the time lag.

3.3.1 Background on Cross-Correlations

Consider two continuous functions $f(t)$ and $g(t)$. The cross-correlation between these two functions is defined as

$$(f * g)(\tau) = \int_{-\infty}^{\infty} \overline{f(t)} g(t+\tau) dt = \int_{-\infty}^{\infty} \overline{f(t-\tau)} g(t) dt \tag{3.2}$$

where τ is the time lag [21]. This allows us to calculate the correlation between shifted versions of the two functions. The value of the lag τ with the highest correlation represents the best fit between the two continuous functions. In the case of our continuous functions being shifted versions of signals, this lag will represent the difference in arrival time between the two signals. Also note that the correlation between two signals at any given time is between -1 and 1.

Consider two continuous impulsive signals s_1, s_2 comprised of shifted versions of the Ricker wavelet as shown in Figure 3.9.

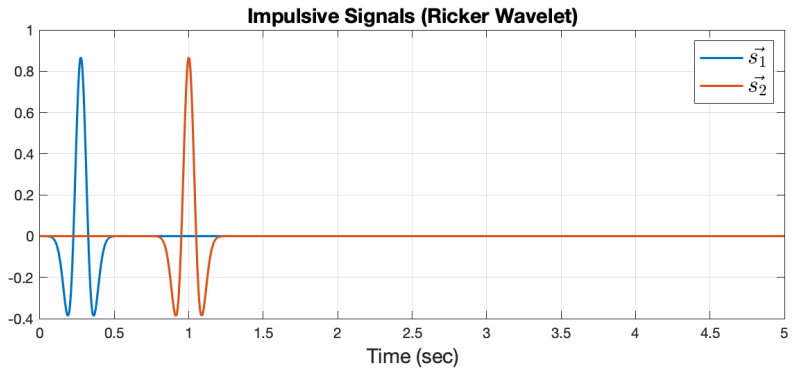

Figure 3.9: Figure of two impulsive signals (Ricker wavelets) to demonstrate the concept of calculating the cross correlation to compute the difference in arrival time of the signals. (created my cross_corr_ex.m)

We can calculate the cross-correlation $(s_1 * s_2)(\tau)$, which has a maximum value of 1 at $\tau = -0.7250$ seconds as shown in Figure 3.10.

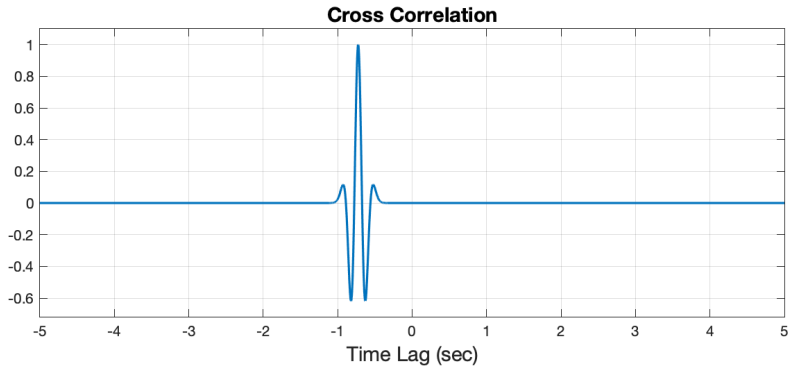

Figure 3.10: Figure of cross correlation between two signals from Figure 3.9. The peak value is a negative time lag meaning the first signal arrives before the second signal. (created by cross_corr_ex.m)

The maximum correlation value being 1 means that our two signals are exact copies other than the fact that they are shifted in time. In real data examples we do not expect the maximum correlation to be 1 because there may be some noise that prevents the signals from being exact copies even if they originate from the same source. The lag value of $\tau = -0.7250$ seconds means s_1 arrives 0.7250 seconds before s_2. Once we have obtained this time lag value we can shift the signals accordingly to line up in time. This can be treated as a pre-processing step for ICA to realign the signals in time before performing separation.

3.3.2 Signal Separation with Pre-processing for Time Lags

In testing scenarios we may be able to calculate this time lag directly given information such as the speed of propagation in the given medium and the distance between the sources and the sensors. For instance, we can find the difference in arrival time for each point source to each sensor by dividing the distance by the speed of propagation in the medium. We can then shift one of our mixed signals to make the arrival times for one of the sources line up. The results doing so for the given k-Wave simulation are shown in Figure 3.11.

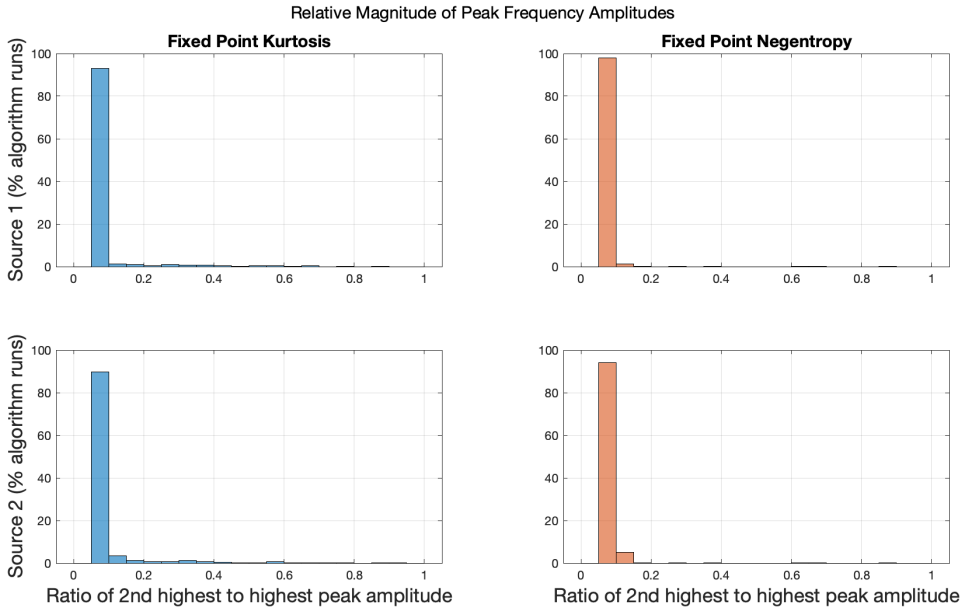

Figure 3.11: Relative magnitude of peak frequency amplitudes after signal separation by kurtosis and negative entropy fixed point iteration algorithms and a pre-processing step to account for difference in arrival time of the signals. Synthetic data generated by kWave Matlab toolbox with sine wave input pressure signals (60 Hz and 25 Hz). (created by k_wave_sim_timelag.m)

Notice the improvement since we have shifted the signals to line up as if the mixture was instantaneous. Before accounting for the difference in arrival time of the signals, ratios for all algorithms were above 0.45. After accounting for the difference in arrival time, 98% of our algorithm runs have ratios at 0.1 or below and the remaining algorithm runs have ratios at or below 0.2.

3.4 Acoustic and Magnetic Sensing Fiber Example

We will now evaluate the performance of the kurtosis and negentropy fixed point iteration algorithms with data obtained from laboratory testing with the acoustic and magnetic sens-

ing fiber. In Figure 3.12 we see the array setup of the fiber optic cable with channel spacing of 2 meters where channels 42-46 measure only acoustic sources and channel 47 measures both the acoustic and magnetic sources.

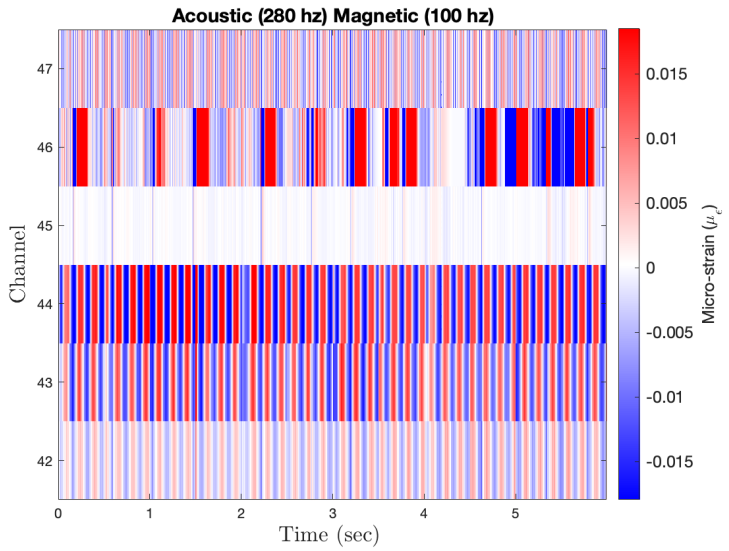

Figure 3.12: DAS array of acoustic and magnetic sensing fiber in a laboratory environment with a magnetic field generated from an air core solenoid; an alternating current source set at 100 Hz and an acoustic source at 280 Hz. The channels are spaced 2 meters apart; channels 42-46 measure only acoustic sources, channel 47 measures both acoustic and magnetic sources.

Recall from the basic ICA model in Section 2.1.2, if we want to recover 2 source signals, we need 2 observed mixed signals. Since this new fiber is still in developmental stages, we currently only have one 2-meter channel, channel 47, along the fiber which can measure both acoustic and magnetic sources.

To demonstrate the signal separation algorithm, we will collect the data for the acoustic and magnetic sources separately and then synthetically mix them together. In Figure 3.13 we observe the Fourier transforms of the individually recorded acoustic and magnetic sources before they have been mixed together. Here $s_1(t)$ is the magnetic source and $s_2(t)$ is the acoustic source.

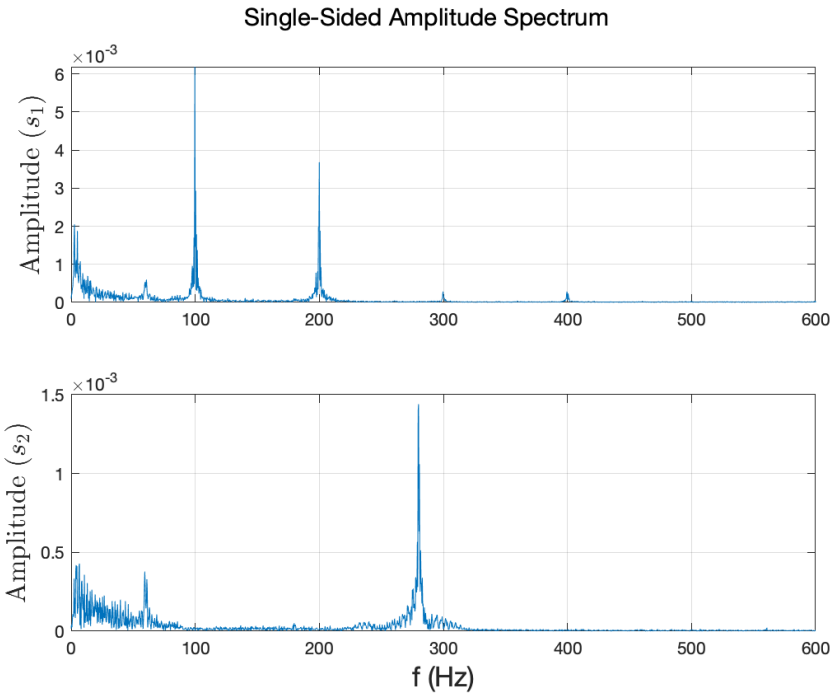

Figure 3.13: Fourier transforms of the individually recorded magnetic (100 Hz) and acoustic (280 Hz) sources. The data for both sources is taken from channel 47 of the acoustic and magnetic sensing fiber which has the capability of measuring acoustic and magnetic sources.

As with the previous test cases, we will perform this mixing and subsequent separation process 1000 times to evaluate the kurtosis and negentropy fixed point iteration algorithms. For this real data example, even though our sources are set to single frequencies, the fiber response to the sources is not only at this single frequency. For example, with the magnetic source, we see frequency responses at multiples of 100 Hz, mostly 100 Hz and 200 Hz. Also, with both the acoustic and the magnetic sources we have some frequency response at 60 Hz likely due to background noise and fiber response to the testing environment. For our 7-step procedure for testing and evaluation, we will compare the magnitudes of the highest peak frequencies for all multiples of 100 Hz and 280 Hz.

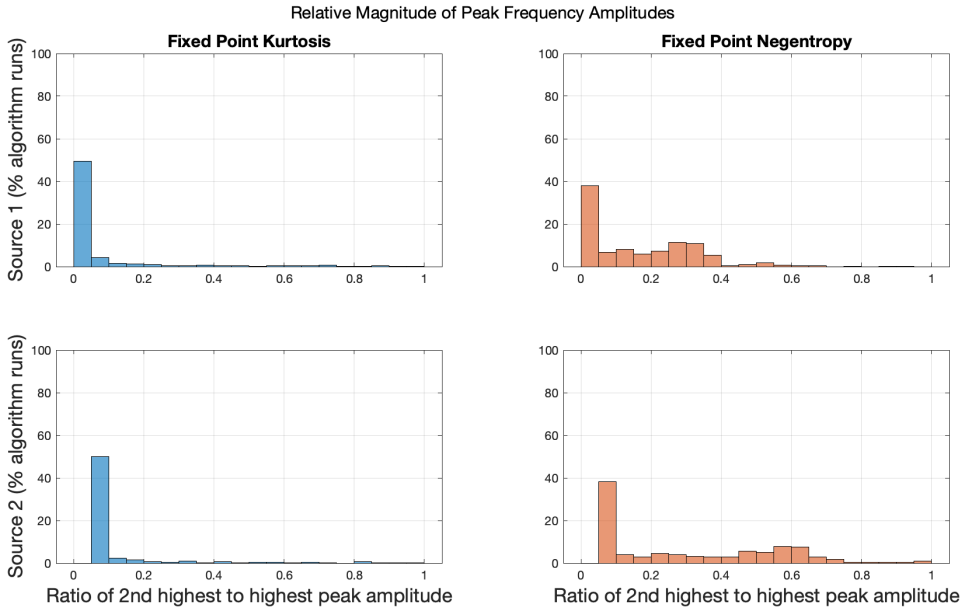

Figure 3.14: Relative magnitude of peak frequency amplitudes after signal separation by kurtosis and negentropy fixed point iteration algorithms. Data is from laboratory testing with acoustic and magnetic sensing fiber which as been normalized and then synthetically mixed together before separation.

Notice in Figure 3.14, the kurtosis fixed point iteration achieves successful separation for just over 50% of the algorithm runs for both sources, while the negentropy fixed point iteration achieves successful separation for around 40% of the algorithm runs. One reason we may expect to see a lower rate of successful separations versus in the other examples, is that the source signals are more complex than single frequency signals. We have frequency responses to background noise as well as frequency responses at the harmonics of the original source frequencies as shown in Figure 3.13. Further development and studies of the fiber such as understanding this frequency response at 60 Hz may be helpful in improving the separation of the signals. For example, additional pre-processing to remove background noise may improve the results of the signal separation.

3.5 Review of Test Cases

In this chapter we compared the results of the fixed point kurtosis and negentropy algorithms for the separation of single frequency sources with five different test cases. The test cases being

1. Two sine waves (no noise)
2. Two sine waves (noise added after synthetic mixing)
3. k-Wave simulation (no preprocessing for timelag)
4. k-Wave simulation with preprocessing for timelag
5. Acoustic and magnetic sensing fiber data

Test case 1 involved two single frequency sources that were synthetically mixed together. Test 2 used the same two single frequency sources, but with noise added after synthetically mixing them together. Recall that we defined an algorithm run as achieving a successful separation if the ratio of the second highest to highest peak amplitude of the resulting signal after separation is less than 0.1. For both cases, the negentropy fixed point iteration performed better with more algorithm runs achieving a successful separation. Additionally, test case 1 has about 4% more successful algorithm runs than test case 2. This is expected since test case 1 is the idealized scenario with no noise added after the sources are mixed together.

Test cases 3 and 4 apply the signal separation algorithms to more realistic simulated data where there is a time delay between when the signals are generated at the source and when they reach the sensors. In test case 3 we achieve no successful separations since we have

not accounted for this difference in arrival of the signals. In test case 4, we added a pre-processing step to shift the signals to align in time which produces successful separations for 90% or more of algorithm runs for both fixed point iterations. As seen with test cases 1 and 2, the negentropy fixed point iteration outperforms the kurtosis fixed point iteration.

Finally, with test case 5 we evaluate the performance of these fixed point iterations on real data obtained in a laboratory setting. With this scenario the kurtosis fixed point iteration slightly outperforms the negentropy in contrast to the previous four test cases and only 50% of the algorithm runs achieve successful separations.

Tables demonstrating the results across all described test cases can be found in Appendix A.

Chapter 4

Array Coherence Post-processing

In this chapter we will introduce a post-processing technique which can be implemented after signal separation to maintain array coherence. Coherence is a metric which can be used compare two signals and determine a relationship between them. Recall from Section 2.2.5 we cannot determine the order of the individual signals after separation. For the simple example of having two mixed signals $\vec{x_1}$ and $\vec{x_2}$ we may estimate the signals in the opposite order after signal separation where $\vec{s_1}$ corresponds to $\vec{x_2}$ and $\vec{s_2}$ corresponds to $\vec{x_1}$. For this scenario we may be able to directly compare the separated signals to the original signals, but this process becomes much more computationally expensive the larger this problem becomes.

In the case where we have spatially separated sensors receiving these mixed signals, such as with DAS, we want to ensure that we have maintained coherence across the entire array. This means we want to ensure that we match up the separated signals correctly at each location and that the separated signals at adjacent locations corresponding to the same source signal are similar up to some time lag. For the specific application we are considering with source signals being acoustic and magnetic, we must also consider the different speeds of propagation of these waves from the point sources to the sensors.

4.1 Problem Setup

Suppose we have an array setup with 2 point sources and 8 point sensors as shown below in Figure 4.1. Notice our mixed signals at the sensors are denoted by purple dots since since they are a combination of the two sources, one blue and one red. Also, recall from the assumptions of ICA that we cannot determine the order of the signals after separation. So we may get a separated signal that corresponds to the second source in the first position.

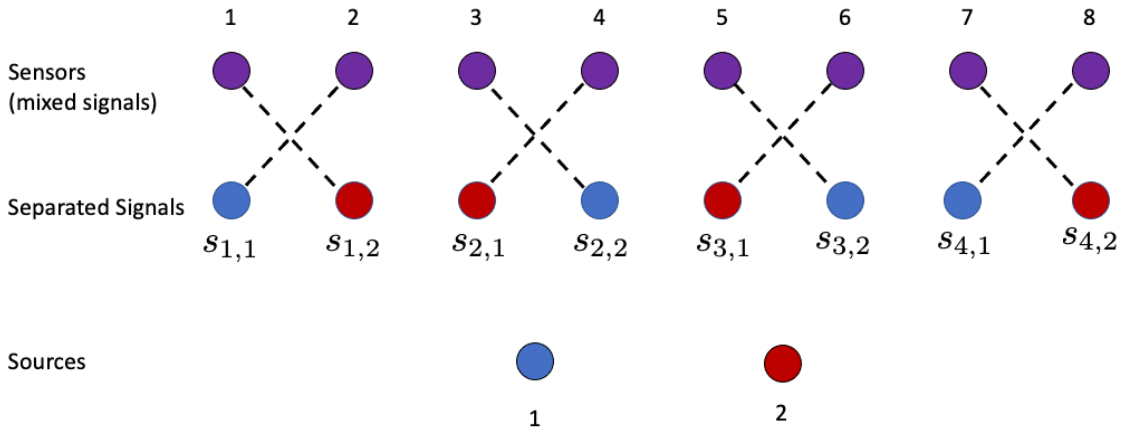

Figure 4.1: Figure showing the physical setup of the sensors, sources, and separated signals for the signal separation array case.

In the context of the ICA problem, we can view this as 4 pairs of mixed signals which we seek to separate into 4 pairs of individual signals. Thus we are actually solving this optimization problem 4 times. For each pair we have the setup $\mathbf{X} = \mathbf{AS}$ where $\mathbf{S}$ and $\mathbf{s}$ are the following:

- Sensor pair 1: $\mathbf{X} = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s}_{1,1} \\ \vec{s}_{1,2} \end{bmatrix}$
- Sensor pair 2: $\mathbf{X} = \begin{bmatrix} \vec{x}_3 \\ \vec{x}_4 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s}_{2,1} \\ \vec{s}_{2,2} \end{bmatrix}$

- Sensor pair 3: $\mathbf{X} = \begin{bmatrix} \vec{x}_5 \\ \vec{x}_6 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s}_{3,1} \\ \vec{s}_{3,2} \end{bmatrix}$
- Sensor pair 4: $\mathbf{X} = \begin{bmatrix} \vec{x}_7 \\ \vec{x}_8 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s}_{4,1} \\ \vec{s}_{4,2} \end{bmatrix}$

Keeping the assumption that there are two sources, this setup can be extended past 4 sensor pairs where we would have

- Sensor pair j: $\mathbf{X} = \begin{bmatrix} \vec{x}_{2j-1} \\ \vec{x}_{2j} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \vec{s}_{j,1} \\ \vec{s}_{j,2} \end{bmatrix}$

4.2 Magnitude Squared Coherence

Magnitude squared coherence [17] is a real-valued function defined as

$$C_{s_i s_j}(f) = \frac{|P_{s_i s_j}(f)|^2}{P_{s_i s_i}(f) P_{s_j s_j}(f)} \tag{4.1}$$

where $P_{s_i s_i}(f)$ is the power spectral density of a signal s_i at frequency f and $P_{s_i s_j}(f)$ is the cross spectral density of two signals s_i and s_j at frequency f. We will use this metric to compare signals against each other, evaluate the performance of the separation algorithm, and efficiently and automatically pair up separated signals corresponding to the same source.

4.2.1 Power Spectral Density

The power spectral density of a signal describes the frequency components that make up the signal in terms of their power. In other words it reveals which frequency components are

most dominant for the signal. The power spectral density [18] is defined as

$$P_{s_i s_i}(f) = \int_{-\infty}^{\infty} R_{s_i s_i}(\tau) e^{-i2\pi f\tau} d\tau \tag{4.2}$$

where

$$R_{s_i s_i}(\tau) = (s_i * s_i)(\tau) = \int_{-\infty}^{\infty} \overline{s_i(t)} s_i(t+\tau) d\tau \tag{4.3}$$

is the auto-correlation function of $s_i(t)$.

We can also find an alternative representation of the power spectral density utilizing the Cross-Correlation Theorem.

Theorem 4.1 (Cross-Correlation Theorem [22])**.** *Given functions $s_i(t)$ and $s_j(t)$ let $R_{s_i s_j}$ denote the cross-correlation. Then*

$$R_{s_i s_j} = \int_{-\infty}^{\infty} \overline{s_i}(t) s_j(t+\tau) d\tau = \mathcal{F}^{-1}\left[\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)\right], \tag{4.4}$$

where $\mathcal{F}(s_i)$ denotes the Fourier transform of $s_i(t)$ and $\mathcal{F}^{-1}$ denotes the inverse Fourier transform.

This means that our power spectral density from Equation (4.2) becomes

$$P_{s_i s_i}(f) = \overline{\mathcal{F}(s_i)}\mathcal{F}(s_i). \tag{4.5}$$

4.2.2 Cross Spectral Density

The cross spectral density [18] of two signals is defined as

$$P_{s_i s_j}(f) = \int_{-\infty}^{\infty} R_{s_i s_j}(\tau) e^{-i2\pi f\tau} d\tau \tag{4.6}$$

where

$$R_{s_i s_j}(\tau) = (s_i * s_j)(\tau) = \int_{-\infty}^{\infty} \overline{s_i(t)} s_j(t+\tau) d\tau \tag{4.7}$$

is the cross-correlation function of $s_i(t)$ and $s_j(t)$. We can again use Theorem 4.1 to rewrite the cross spectral density as

$$P_{s_i s_j}(f) = \overline{\mathcal{F}(s_i)} \mathcal{F}(s_j). \tag{4.8}$$

4.2.3 Coherence Matrix Properties

Recalling the definition of magnitude squared coherence in Equation (4.1) and using Theorem 4.1, we can rewrite the magnitude squared coherence as

$$C_{s_i s_j}(f) = \frac{|\overline{\mathcal{F}(s_i)} \mathcal{F}(s_j)|^2}{\left(\overline{\mathcal{F}(s_i)} \mathcal{F}(s_i)\right) \left(\overline{\mathcal{F}(s_j)} \mathcal{F}(s_j)\right)} \tag{4.9}$$

We will use Welch's method for estimating coherence where the signal is split into N segments which may be overlapping [17]. Then the magnitude squared coherence can be written as

$$C_{s_i s_j}(f) = \frac{|\sum_{n=1}^{N} \overline{\mathcal{F}(s_i)_n} \mathcal{F}(s_j)_n|^2}{\left(\sum_{n=1}^{N} \overline{\mathcal{F}(s_i)_n} \mathcal{F}(s_i)_n\right) \left(\sum_{n=1}^{N} \overline{\mathcal{F}(s_j)_n} \mathcal{F}(s_j)_n\right)} \tag{4.10}$$

where $\mathcal{F}(s_i)_n$ denotes the Fourier transform of s_i over time interval n. Notice if we have one time interval, the magnitude squared coherence between two signals will always be 1. Breaking the signals into N segments creates cross products in the numerator that will not immediately cancel with the spectral densities in the denominator, allowing for values other than 1.

Now using our setup from Section 4.1 we can define a coherence matrix which calculates the magnitude squared coherence between every pair of estimated signals after separation as

shown in table 4.1.

	$s_{1,1}$	$s_{1,2}$	$s_{2,1}$	$s_{2,2}$	$s_{3,1}$	$s_{3,2}$	$s_{4,1}$	$s_{4,2}$
$s_{1,1}$	$C_{s_{1,1}s_{1,1}}$	$C_{s_{1,1}s_{1,2}}$	$C_{s_{1,1}s_{2,1}}$	$C_{s_{1,1}s_{2,2}}$	$C_{s_{1,1}s_{3,1}}$	$C_{s_{1,1}s_{3,2}}$	$C_{s_{1,1}s_{4,1}}$	$C_{s_{1,1}s_{4,2}}$
$s_{1,2}$	$C_{s_{1,2}s_{1,1}}$	$C_{s_{1,2}s_{1,2}}$	$C_{s_{1,2}s_{2,1}}$	$C_{s_{1,2}s_{2,2}}$	$C_{s_{1,2}s_{3,1}}$	$C_{s_{1,2}s_{3,2}}$	$C_{s_{1,2}s_{4,1}}$	$C_{s_{1,2}s_{4,2}}$
$s_{2,1}$	$C_{s_{2,1}s_{1,1}}$	$C_{s_{2,1}s_{1,2}}$	$C_{s_{2,1}s_{2,1}}$	$C_{s_{2,1}s_{2,2}}$	$C_{s_{2,1}s_{3,1}}$	$C_{s_{2,1}s_{3,2}}$	$C_{s_{2,1}s_{4,1}}$	$C_{s_{2,1}s_{4,2}}$
$s_{2,2}$	$C_{s_{2,2}s_{1,1}}$	$C_{s_{2,2}s_{1,2}}$	$C_{s_{2,2}s_{2,1}}$	$C_{s_{2,2}s_{2,2}}$	$C_{s_{2,2}s_{3,1}}$	$C_{s_{2,2}s_{3,2}}$	$C_{s_{2,2}s_{4,1}}$	$C_{s_{2,2}s_{4,2}}$
$s_{3,1}$	$C_{s_{3,1}s_{1,1}}$	$C_{s_{3,1}s_{1,2}}$	$C_{s_{3,1}s_{2,1}}$	$C_{s_{3,1}s_{2,2}}$	$C_{s_{3,1}s_{3,1}}$	$C_{s_{3,1}s_{3,2}}$	$C_{s_{3,1}s_{4,1}}$	$C_{s_{3,1}s_{4,2}}$
$s_{3,2}$	$C_{s_{3,2}s_{1,1}}$	$C_{s_{3,2}s_{1,2}}$	$C_{s_{3,2}s_{2,1}}$	$C_{s_{3,2}s_{2,2}}$	$C_{s_{3,2}s_{3,1}}$	$C_{s_{3,2}s_{3,2}}$	$C_{s_{3,2}s_{4,1}}$	$C_{s_{3,2}s_{4,2}}$
$s_{4,1}$	$C_{s_{4,1}s_{1,1}}$	$C_{s_{4,1}s_{1,2}}$	$C_{s_{4,1}s_{2,1}}$	$C_{s_{4,1}s_{2,2}}$	$C_{s_{4,1}s_{3,1}}$	$C_{s_{4,1}s_{3,2}}$	$C_{s_{4,1}s_{4,1}}$	$C_{s_{4,1}s_{4,2}}$
$s_{4,2}$	$C_{s_{4,2}s_{1,1}}$	$C_{s_{4,2}s_{1,2}}$	$C_{s_{4,2}s_{2,1}}$	$C_{s_{4,2}s_{2,2}}$	$C_{s_{4,2}s_{3,1}}$	$C_{s_{4,2}s_{3,2}}$	$C_{s_{4,2}s_{4,1}}$	$C_{s_{4,2}s_{4,1}}$

Table 4.1: Table showing coherence matrix structure.

Next we will explore some properties of the magnitude squared coherence which will inform the structure of our coherence matrix.

Proposition 3. *Given a signal s_i, the magnitude squared coherence $C_{s_i s_i}(f) = 1$. Thus the elements along the main diagonal of the coherence matrix are all 1.*

Proof. Assume we have a signal s_i. Then the magnitude squared coherence between this

signal and itself is

$$
\begin{aligned}
C_{s_i s_i} &= \frac{|P_{s_i s_i}|^2}{P_{s_i s_i} P_{s_i s_i}} \\
&= \frac{|\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)|^2}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(from Equation 4.8)} \\
&= \frac{\left(\overline{\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)}\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(since } |a|^2 = \overline{a}a\text{)} \\
&= \frac{\left(\mathcal{F}(s_i)\overline{\mathcal{F}(s_i)}\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(since } \overline{\overline{a}b} = a\bar{b}\text{)} \\
&= \frac{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(since } \overline{a}b = b\overline{a}\text{)} \\
&= 1
\end{aligned}
$$

□

Proposition 4. *Given signals s_i and s_j, $C_{s_i s_j}(f) = C_{s_j s_i}(f)$. Thus the coherence matrix is symmetric.*

Proof. Assume we have signals s_i and s_j. Then the magnitude squared coherence between

s_i and s_j is

$$
\begin{aligned}
C_{s_i s_j} &= \frac{|P_{s_i s_j}|^2}{P_{s_i s_i} P_{s_j s_j}} \\
&= \frac{|\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)|^2}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(from Equation 4.8)} \\
&= \frac{\left(\overline{\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)}\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(since } |a|^2 = \overline{a}a\text{)} \\
&= \frac{\left(\mathcal{F}(s_i)\overline{\mathcal{F}(s_j)}\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(since } \overline{\overline{a}b} = a\overline{b}\text{)} \\
&= \frac{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)\right)\left(\mathcal{F}(s_i)\overline{\mathcal{F}(s_j)}\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(commuting the products on the numerator)} \\
&= \frac{\left(\mathcal{F}(s_j)\overline{\mathcal{F}(s_i)}\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(since } \overline{a}b = b\overline{a}\text{)} \\
&= \frac{\left(\mathcal{F}(s_j)\overline{\mathcal{F}(s_i)}\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(commuting the products on the denominator)} \\
&= \frac{\left(\overline{\overline{\mathcal{F}(s_j)}\mathcal{F}(s_i)}\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_i)\right)}{\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(since } \overline{\overline{a}b} = a\overline{b}\text{)} \\
&= \frac{|\overline{\mathcal{F}(s_j)}\mathcal{F}(s_i)|^2}{\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)} && \text{(since } |a|^2 = \overline{a}a\text{)} \\
&= \frac{|P_{s_j s_i}|^2}{P_{s_j s_j} P_{s_i s_i}} = C_{s_j s_i} && \text{(from Equation 4.8)}
\end{aligned}
$$

□

Proposition 5. *The amplitudes of two signals has no effect on their coherence.*

Proof. Suppose we have two signals $s_i(t)$ and $s_j(t)$ with amplitudes A_i and A_j. Define $\tilde{s}_i(t)$

and $\tilde{s}_j(t)$ such that

$$s_i(t) = A_i \tilde{s}_i(t) \tag{4.11}$$

$$s_j(t) = A_j \tilde{s}_j(t) \tag{4.12}$$

Then

$$\begin{aligned}
C_{s_i s_j} &= \frac{|P_{s_i s_j}|^2}{P_{s_i s_i} P_{s_j s_j}} \\
&= \frac{|\overline{\mathcal{F}(s_i)}\mathcal{F}(s_j)|^2}{\left(\overline{\mathcal{F}(s_i)}\mathcal{F}(s_i)\right)\left(\overline{\mathcal{F}(s_j)}\mathcal{F}(s_j)\right)} && \text{(from Equation 4.8)} \\
&= \frac{|\overline{\mathcal{F}(A_i\tilde{s}_i)}\mathcal{F}(A_j\tilde{s}_j)|^2}{\left(\overline{\mathcal{F}(A_i\tilde{s}_i)}\mathcal{F}(A_j\tilde{s}_i)\right)\left(\overline{\mathcal{F}(A_j\tilde{s}_j)}\mathcal{F}(A_j\tilde{s}_j)\right)} \\
&= \frac{|\overline{A_i\mathcal{F}(\tilde{s}_i)}A_j\mathcal{F}(\tilde{s}_j)|^2}{\left(\overline{A_i\mathcal{F}(\tilde{s}_i)}A_i\mathcal{F}(\tilde{s}_i)\right)\left(\overline{A_j\mathcal{F}(\tilde{s}_j)}A_j\mathcal{F}(\tilde{s}_j)\right)} \\
&= \frac{(A_iA_j)^2|\overline{\mathcal{F}(\tilde{s}_i)}\mathcal{F}(\tilde{s}_j)|^2}{A_i^2A_j^2\left(\overline{\mathcal{F}(\tilde{s}_i)}\mathcal{F}(\tilde{s}_i)\right)\left(\overline{\mathcal{F}(\tilde{s}_j)}\mathcal{F}(\tilde{s}_j)\right)} \\
&= \frac{|P_{\tilde{s}_i\tilde{s}_j}|^2}{P_{\tilde{s}_i\tilde{s}_i}P_{\tilde{s}_j\tilde{s}_j}} = C_{\tilde{s}_i\tilde{s}_j}
\end{aligned}$$

□

Using properties 3 and 4 we can rewrite the coherence matrix denoted by M_C.

$$M_C = \begin{bmatrix} 1 & * & * & * & * & * & * & * \\ C_{s_{1,2}s_{1,1}} & 1 & * & * & * & * & * & * \\ C_{s_{2,1}s_{1,1}} & C_{s_{2,1}s_{1,2}} & 1 & * & * & * & * & * \\ C_{s_{2,2}s_{1,1}} & C_{s_{2,2}s_{1,2}} & C_{s_{2,2}s_{2,1}} & 1 & * & * & * & * \\ C_{s_{3,1}s_{1,1}} & C_{s_{3,1}s_{1,2}} & C_{s_{3,1}s_{2,1}} & C_{s_{3,1}s_{2,2}} & 1 & * & * & * \\ C_{s_{3,2}s_{1,1}} & C_{s_{3,2}s_{1,2}} & C_{s_{3,2}s_{2,1}} & C_{s_{3,2}s_{2,2}} & C_{s_{3,2}s_{3,1}} & 1 & * & * \\ C_{s_{4,1}s_{1,1}} & C_{s_{4,1}s_{1,2}} & C_{s_{4,1}s_{2,1}} & C_{s_{4,1}s_{2,2}} & C_{s_{4,1}s_{3,1}} & C_{s_{4,1}s_{3,2}} & 1 & * \\ C_{s_{4,2}s_{1,1}} & C_{s_{4,2}s_{1,2}} & C_{s_{4,2}s_{2,1}} & C_{s_{4,2}s_{2,2}} & C_{s_{4,2}s_{3,1}} & C_{s_{4,2}s_{3,2}} & C_{s_{4,2}s_{4,1}} & 1 \end{bmatrix} \tag{4.13}$$

Notice the $*$ denotes the reflection of the values in the lower triangular portion of the matrix since it is symmetric.

4.2.4 Coherence Matrix Structure

Consider the scenario where we have two source signals $s_{true1}(t)$ and $s_{true2}(t)$ and 4 pairs of separated signals, 8 in total. Suppose we have an ideal separation where out of our 8 separated signals, we have 4 signals which are exactly $s_{true1}(t)$ and 4 signals which are exactly $s_{true2}(t)$

$$\begin{bmatrix} \vec{s}_{1,1} \\ \vec{s}_{1,2} \end{bmatrix} = \begin{bmatrix} \vec{s}_{2,2} \\ \vec{s}_{2,1} \end{bmatrix} = \begin{bmatrix} \vec{s}_{3,2} \\ \vec{s}_{3,1} \end{bmatrix} = \begin{bmatrix} \vec{s}_{4,1} \\ \vec{s}_{4,2} \end{bmatrix} = \begin{bmatrix} \vec{s}_{true1} \\ \vec{s}_{true2} \end{bmatrix}$$

as depicted in Figure 4.1. Suppose also that $C_{s_{true1}s_{true2}} = 0$ for the sake of simplifying this example.

	$s_{1,1}$	$s_{1,2}$	$s_{2,1}$	$s_{2,2}$	$s_{3,1}$	$s_{3,2}$	$s_{4,1}$	$s_{4,2}$
$s_{1,1}$	1	0	0	1	0	1	1	0
$s_{1,2}$	0	1	1	0	1	0	0	1
$s_{2,1}$	0	1	1	0	1	0	0	1
$s_{2,2}$	1	0	0	1	0	1	1	0
$s_{3,1}$	0	1	1	0	1	0	0	1
$s_{3,2}$	1	0	0	1	0	1	1	0
$s_{4,1}$	1	0	0	1	0	1	1	0
$s_{4,2}$	0	1	1	0	1	0	0	1

$\longrightarrow$

	$s_{1,1}$	$s_{2,2}$	$s_{3,2}$	$s_{4,1}$	$s_{1,2}$	$s_{2,1}$	$s_{3,1}$	$s_{4,2}$
$s_{1,1}$	1	1	1	1	0	0	0	0
$s_{2,2}$	1	1	1	1	0	0	0	0
$s_{3,2}$	1	1	1	1	0	0	0	0
$s_{4,1}$	1	1	1	1	0	0	0	0
$s_{1,2}$	0	0	0	0	1	1	1	1
$s_{2,1}$	0	0	0	0	1	1	1	1
$s_{3,1}$	0	0	0	0	1	1	1	1
$s_{4,2}$	0	0	0	0	1	1	1	1

Table 4.2: Coherence matrix and reordered coherence matrix for simple example.

Then our coherence matrix is as shown on the left side of Table 4.2. If we can reorder this coherence matrix as shown on the right side of Table 4.2 we will have all of the separated signals corresponding to the same source signal paired together. To reorder this matrix as shown, we may want to find some permutation vector p where M_C is the original matrix and $M_C(p, p)$ is the reordered matrix.

4.3 Sparse Matrix Operations

Once we calculate the coherence matrix we want to be able to group the separated signals together to determine which source signal they correspond to and to distinguish between the signals that have been separated and those that are still mixed.

In many matrix reordering algorithms, it is more useful to begin with a sparse matrix. With our coherence matrix, we expect to see signals corresponding to the same source to have a value closer to 1, while signals corresponding to different sources will have coherence values closer to 0. We may also have some coherence values in the middle where the signal separation was not entirely successful and the signals are still mixed. To obtain a sparse

matrix we will threshold the coherence matrix so only some percentage of the values remain non-zero. Then we are able to explore more matrix reordering techniques.

4.3.1 Symmetric Reverse Cuthill-McKee

The Symmetric Reverse Cuthill-McKee (SRCM) [5] algorithm seeks to permute a symmetric matrix such that the nonzero elements are closer to the diagonal. This algorithm is often used as a pre-ordering technique for Cholesky factorization since it produces a matrix with a more narrow bandwidth [5].

Given an $n \times n$ symmetric matrix, this algorithm returns an n-tuple of reordered vertices. In order to do this, we will treat the matrix as the adjacency matrix of a graph. For example, given

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

we treat each row as a vertex of a graph. Then we see that rows 1 and 3 are adjacent since the 3rd column has a nonzero value in row 1 and similarly rows 2 and 4 are adjacent. Also note that the degree of a vertex (or row in this case) is the number of vertices adjacent to the given vertex. For example, the degree of vertex (row) 1 is 1. Then the SRCM algorithm [5] is as follows:

Algorithm 5 Symmetric Reverse Cuthill-McKee Algorithm

1: Let p be the vertex (row) with the lowest degree.
2: **for** $i = 1 : n$ **do**
3: Find all vertices adjacent to the ith component of p and let $A_i = \text{Adj}(p_i) \setminus p$
4: Sort A_i ascending by minimum predecessor and degree of the vertex
5: Let $p = \{p, p_i\}$
6: Reverse the order of the vertices in p
7: **end for**

We will utilize the Matlab implementation `symrcm`.

4.3.2 Symmetric Approximate Minimum Degree Permutation

The symmetric approximate minimum degree permutation (SAMD) [2] seeks to permute a symmetric positive definite matrix **A** such that the permuted matrix has a sparser Cholesky factor than A as described in [2]. This is also used as a pre-ordering technique for Cholesky factorization as it produces a with large blocks of zeros. We will utilize the Matlab implementation `symamd`.

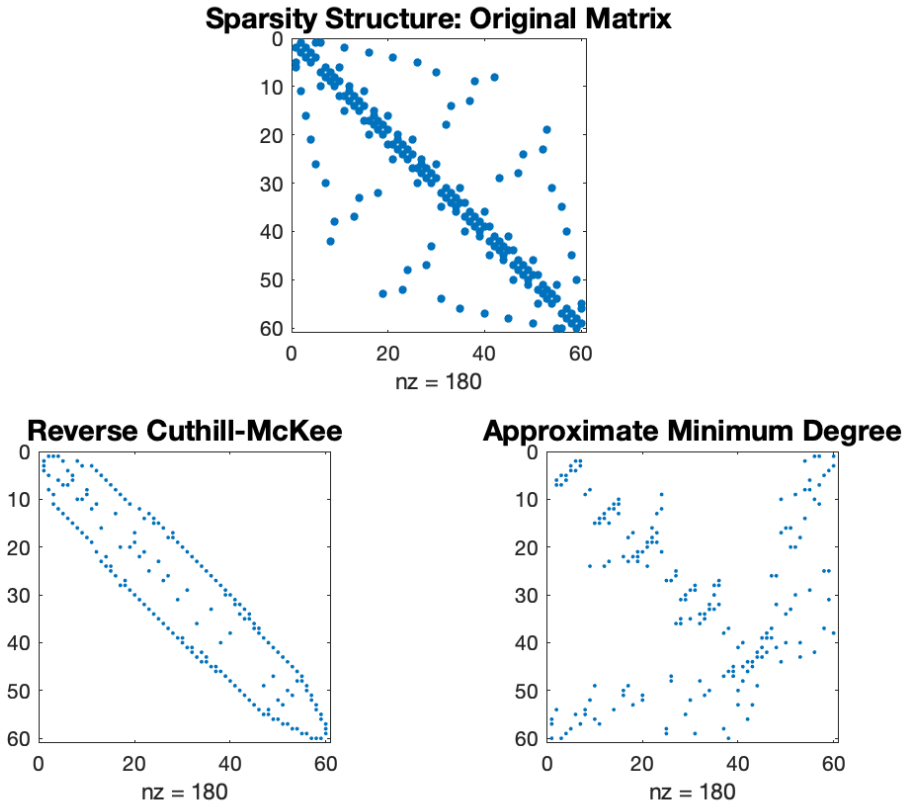

Figure 4.2: Figure showing sparsity structure for a given matrix and the permuted matrices after applying the SRCM and the SAMD permutations. (created by matrix_reordering.m)

4.4 Test Cases

In this section we will explore the technique of calculating a coherence matrix and performing some permutations on the matrix for several synthetic examples involving a square and hat function. We we will first look at an example assuming ideal separation and then will consider cases involving time lags and unsuccessful separations.

4.4.1 Synthetic Example: Ideal Separation

Suppose we consider two functions, a square wave and a hat wave defined on the interval $[a, b]$ as follows:

$$s_1(t) = \begin{cases} A_1 & t \in [a, b] \\ 0 & t \in (-\infty, a) \cup (b, \infty) \end{cases} \tag{4.14}$$

$$s_2(t) = \begin{cases} \frac{2A_2}{b-a}t + \frac{2A_2 a}{a-b} & t \in [a, \frac{a+b}{2}] \\ \frac{2A_2}{a-b}t + \frac{2A_2 b}{b-a} & t \in [\frac{a+b}{2}, b] \end{cases}. \tag{4.15}$$

with a sampling frequency of $f_s = 50000$. If we have $A_1 = A_2 = 1$ and $[a, b] = [.5, 1.5]$, we have the functions as shown in Figure 4.3.

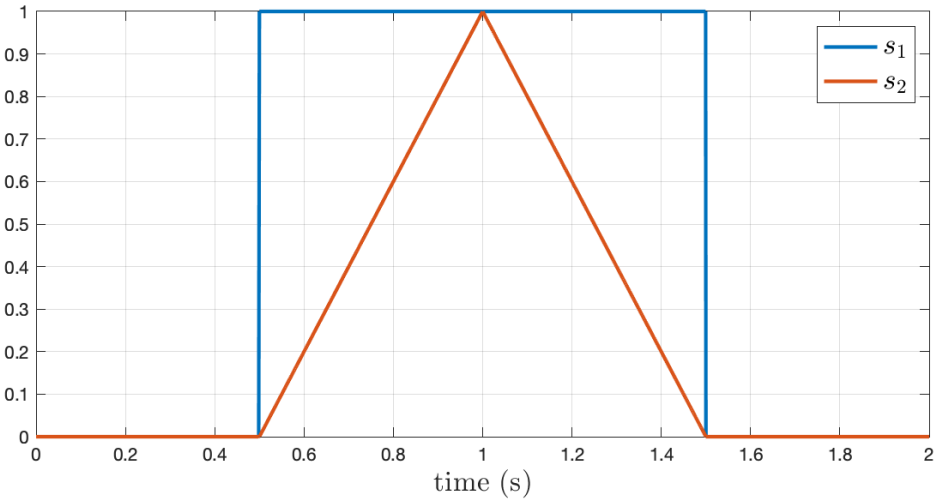

Figure 4.3: Figure of square and hat waves on the interval $[.5, 1.5]$ with amplitudes of 1. (created by coherence_square_hat.m)

Suppose we have the signal separation problem setup as described in Section 4.1 where four pairs of separated signals are exactly s_1 and s_2 from Equations (4.14) and (4.15), but not

necessarily in that order for each pair as shown below

$$\begin{bmatrix} \vec{s}_{1,1} \\ \vec{s}_{1,2} \end{bmatrix} = \begin{bmatrix} \vec{s}_{2,2} \\ \vec{s}_{2,1} \end{bmatrix} = \begin{bmatrix} \vec{s}_{3,2} \\ \vec{s}_{3,1} \end{bmatrix} = \begin{bmatrix} \vec{s}_{4,1} \\ \vec{s}_{4,2} \end{bmatrix} = \begin{bmatrix} \vec{s}_{true1} \\ \vec{s}_{true2} \end{bmatrix}.$$

Then we have that rows 1, 4, 6, 7 of the coherence matrix correspond to s_1 and rows 2, 3, 5, 8 of the coherence matrix correspond to s_2. We can calculate the coherence matrix as defined in Equation (4.13) by evaluating

```
mscohere(s1,s2,window,noverlap,f,Fs)
```

with the inputs `window = 10000`, `noverlap = 5000`, `f=0:5:100`, `Fs = 50000` to ensure we have window lengths with enough support for low frequencies. Then our resulting coherence matrix evaluated at $f = 10$ is approximately

$$M_C = \begin{bmatrix} 1 & .0131 & 0.131 & 1 & .0131 & 1 & 1 & .0131 \\ .0131 & 1 & 1 & .0131 & 1 & .0131 & .0131 & 1 \\ .0131 & 1 & 1 & .0131 & 1 & .0131 & .0131 & 1 \\ 1 & .0131 & 0.131 & 1 & .0131 & 1 & 1 & .0131 \\ .0131 & 1 & 1 & .0131 & 1 & .0131 & .0131 & 1 \\ 1 & .0131 & 0.131 & 1 & .0131 & 1 & 1 & .0131 \\ 1 & .0131 & 1 & .0131 & 1 & .0131 & 1 & .0131 \\ .0131 & 1 & .0131 & 1 & .0131 & 1 & .0131 & 1 \end{bmatrix}. \tag{4.16}$$

After obtaining this matrix we will perform thresholding so we can apply sparse matrix reordering techniques. Here we will apply a threshold so all values below the 40th percentile of the values in the matrix are zeroed out. We can then apply the SRCM and SAMD permutations as described in Sections 4.3.1 and 4.3.2 to the thresholded matrix and we

obtain

$$M_C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.17}$$

Then we can apply these permutation vectors to the original matrix before thresholding, and get

$$M_C = \begin{bmatrix} 1 & 1 & 1 & 1 & .0131 & .0131 & .0131 & .0131 \\ 1 & 1 & 1 & 1 & .0131 & .0131 & .0131 & .0131 \\ 1 & 1 & 1 & 1 & .0131 & .0131 & .0131 & .0131 \\ 1 & 1 & 1 & 1 & .0131 & .0131 & .0131 & .0131 \\ .0131 & .0131 & .0131 & .0131 & 1 & 1 & 1 & 1 \\ .0131 & .0131 & .0131 & .0131 & 1 & 1 & 1 & 1 \\ .0131 & .0131 & .0131 & .0131 & 1 & 1 & 1 & 1 \\ .0131 & .0131 & .0131 & .0131 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.18}$$

where our permutation vector p for the SRCM is $p = [7\ 6\ 1\ 4\ 8\ 5\ 2\ 3]$ and for the SAMD is $p = [1\ 4\ 6\ 7\ 2\ 3\ 5\ 8]$. This means the signals in rows 1,4,6,7 and rows 2,3,5,8 correspond to each other as expected. Notice in this case the SRCM and SAMD permutations produce the same reordered matrix, but with slightly different permutation vectors.

4.4.2 Synthetic Example: Time lags

Suppose now we consider the sample problem as in Section 4.4.1, but we now have time lags from our original source square and hat wave signals to our sensors. We will denote the distance from mixed signal pair i to source j by d_{ij} measured in meters. Considering the physical setup of the problem as shown in Figure 4.1 suppose we have

$$d_{1,1} = 15, \quad d_{2,1} = 10, \quad d_{3,1} = 15, \quad d_{4,1} = 20$$
$$d_{1,2} = 20, \quad d_{2,2} = 15, \quad d_{3,2} = 10, \quad d_{4,2} = 15$$

Now suppose our first source s_1, the square wave, is an acoustic source with speed of propagation being 343 m/s and our second source s_2, the hat wave, is an electromagnetic source with speed of propagation being $3 * 10^8$ m/s. Then our separated signal would be some time shifted version of the original signals according to the distance of the sensors from the sources as shown in Figure 4.4.

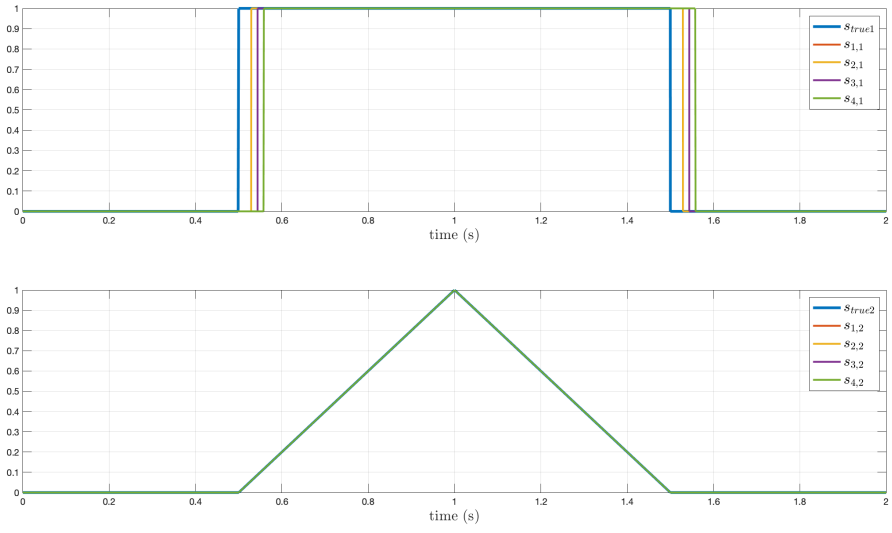

Figure 4.4: Figure showing time shifted versions of the square and hat waves with different speeds of propagation. (created by coherence_square_hat_shift.m)

Notice for the hat wave, since the speed of electromagnetic propagation is so fast, we observe too small a time shift to easily visualize over these short distance of 20 meters or less.

Suppose we performed signal separation and obtained these shifted versions of the original signals as our result. Then using the same method as in Section 4.4.1, we can calculate the coherence matrix as shown in Figure 4.5.

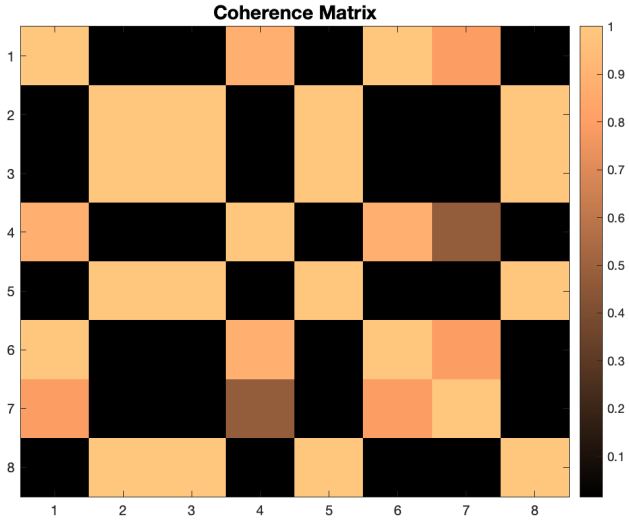

Figure 4.5: Coherence matrix for square and hat wave functions where signal separation resulted in shifted versions of original signals. The color bar denotes the coherence value. (created by coherence_square_hat_shift.m)

In this figure, the color bar denotes the value of the coherence. Notice the coherence along the diagonal is 1 as expected since this is the coherence of a signal with itself. If we apply the matrix permutations as discussed in Sections 4.3.1 and 4.3.2, we expect this matrix to be grouped into blocks of lighter colors, with coherence closer to 1, in the top left and bottom right corners, and darker colors, with coherence closer to 0 elsewhere.

The resulting coherence matrix after matrix reordering with the SRCM permutation is

$$M_C = \begin{bmatrix} 1 & .7957 & .7957 & .4693 & .0176 & .0176 & .0176 & 0.0176 \\ .7957 & 1 & 1 & .8842 & .0178 & .0178 & .0176 & .0178 \\ .7957 & 1 & 1 & .8842 & .0178 & .0178 & .0176 & .0178 \\ .4693 & .8842 & .8842 & 1 & .0153 & .0153 & .0153 & .0153 \\ .0176 & .0178 & .0178 & .0153 & 1 & 1 & 1 & 1 \\ .0176 & .0178 & .0178 & .0153 & 1 & 1 & 1 & 1 \\ .0176 & .0178 & .0178 & .0153 & 1 & 1 & 1 & 1 \\ .0176 & 1 & .0178 & .0153 & 1 & 1 & 1 & 1 \end{bmatrix}$$

where the permutation vector is $p = [7\ 6\ 1\ 4\ 8\ 5\ 2\ 3]$. If we instead use the SAMD permutation, we have the coherence matrix

$$M_C = \begin{bmatrix} 1 & .8842 & 1 & .4693 & .7957 & ..0178 & ..0178 & 0.0178 \\ .8842 & 1 & .8842 & .4693 & .0153 & .0153 & .0153 & .0153 \\ 1 & .8842 & 1 & .7957 & .0178 & .0178 & .0178 & .0178 \\ .7957 & .4693 & .7957 & 1 & .0176 & .0176 & .0176 & .0176 \\ .0178 & .0153 & .0178 & .0176 & 1 & 1 & 1 & 1 \\ .0178 & .0153 & .0178 & .0176 & 1 & 1 & 1 & 1 \\ .0178 & .0153 & .0178 & .0176 & 1 & 1 & 1 & 1 \\ .0178 & .0153 & .0178 & .0176 & 1 & 1 & 1 & 1 \end{bmatrix}$$

where the permutation vector is $p = [1\ 4\ 6\ 7\ 2\ 3\ 5\ 8]$. We can visualize these matrices with colored pixels for each coherence value as shown in Figure 4.6.

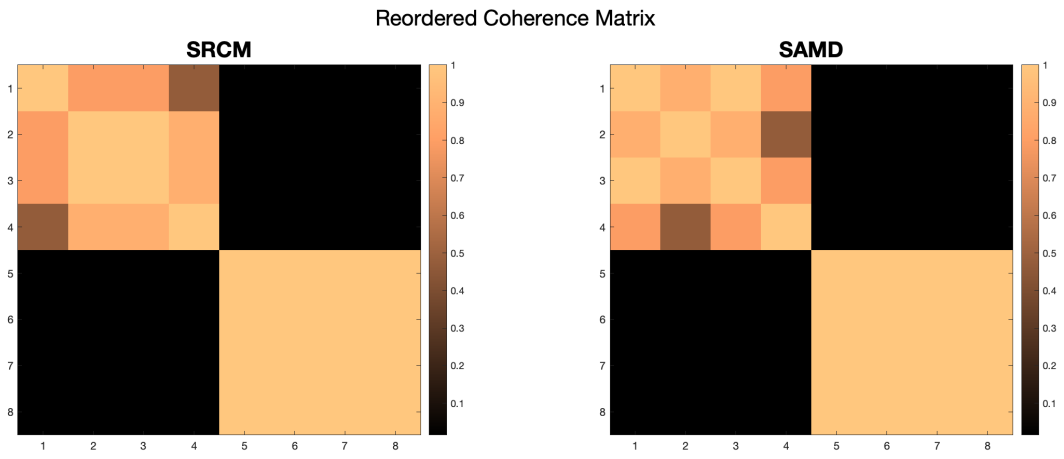

Figure 4.6: Reordered coherence matrices for shifted versions of square and hat wave functions using SRCM and SAMD. We see blocks of high coherence values at the top left and bottom right corners forming groupings of signals corresponding to the two sources. (created by coherence_square_hat_shift.m)

Even with the shifted signals we still obtain the correct pairings of signals together for both reordering methods since rows 1,4,6,7 correspond to the square wave and rows 2,3,5,8 correspond to the hat wave. Notice though that we do have different orderings of signals within the groupings for the two different matrix reordering techniques.

4.4.3 Synthetic Example: Non-Ideal Separation

For this example, we will suppose we have not successfully separated all of our signals during the signal separation algorithm. Let s_1 be the square wave in Equation (4.14) and s_2 be the hat wave in Equation (4.15). Suppose our resulting signals are as follows where the signals are shifted in time according to Section 4.4.2:

- Sensor pair 1: $s_{1,1} = s_1 - s_2, \quad s_{1,2} = s_1$
- Sensor pair 2: $s_{2,1} = s_2, \quad s_{2,2} = s_1$

- Sensor pair 3: $s_{3,1} = s_2, \quad s_{3,2} = \text{random noise}$
- Sensor pair 4: $s_{4,1} = s_1\ , \quad s_{4,2} = s_2$

Notice for the first sensor pair, we recover the first source signal and a mixed signal. Also for the third sensor pair, we recover the second source signal as well as some random noise. This is a scenario that could occur with real data as ICA can be used to separate signals from noise. For our coherence matrix and reordering, the permutation vector will be calculated with the positions of the rows being

$$\begin{bmatrix} s_{1,1} & s_{1,2} & s_{2,1} & s_{2,2} & s_{3,1} & s_{3,2} & s_{4,1} & s_{4,2} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

We obtain the coherence matrix as shown in Figure 4.7.

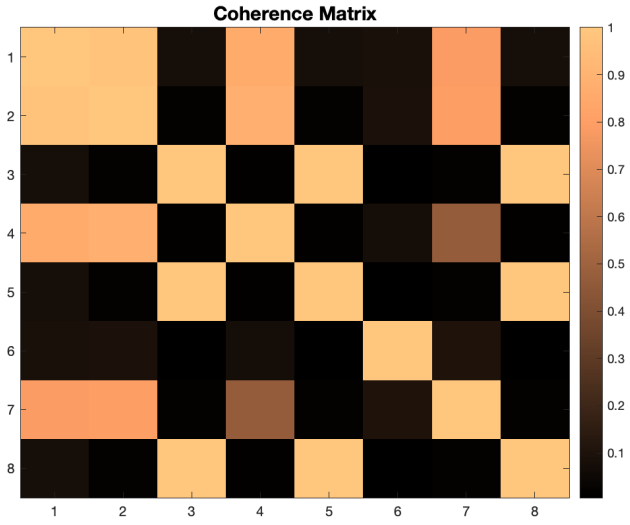

Figure 4.7: Coherence matrix for square and hat wave functions where signal separation is not entirely ideal. The resulting signals after separation are returned in a different order and may still be mixed signals. (created by coherence_square_hat_shift_nonideal.m)

After matrix permutations, we should expect to see the first source being positions 2,4,7 and the second source being positions 3,5,8 while positions 1,6 correspond to signals that are

still mixed. We perform the SRCM and SAMD permutations and obtain the new coherence matrices as shown in Figure 4.8. The permutation vector for SRCM is $p = [8\ 5\ 3\ 6\ 4\ 2\ 1\ 7]$ and the permutation vector for SAMD is $p = [3\ 5\ 8\ 6\ 4\ 7\ 1\ 2]$.

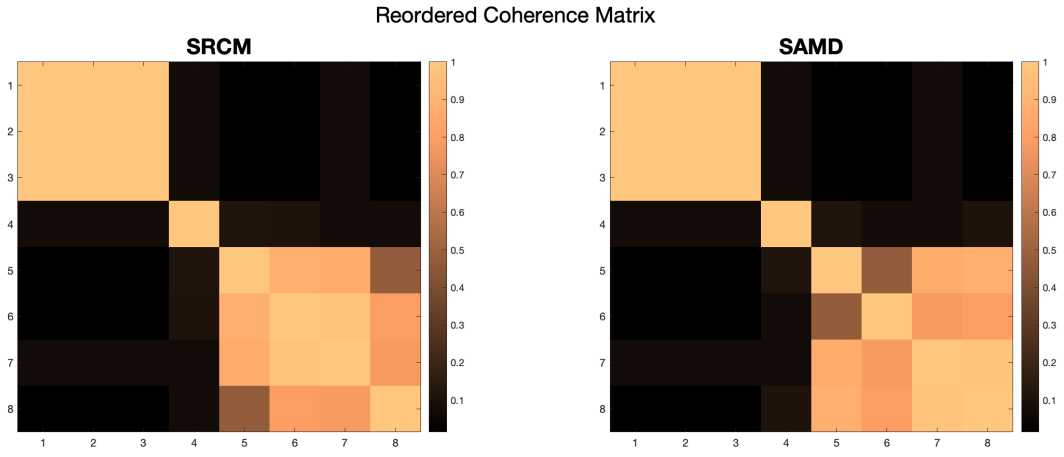

Figure 4.8: Reordered coherence matrices for square and hat wave functions using SRCM and SAMD permutations where signal separation is not entirely ideal. We see blocks of high coherence values at the top left and bottom right corners forming groupings of signals corresponding to the two sources. (created by coherence_square_hat_shift_nonideal.m)

Looking at the reordered coherence matrices, we see that the top left corner has coherence values close to 1 suggesting that the signals in positions 8,5,3 of the original coherence matrix correspond to the same source. This is as we expected because the signals in positions 8,5,3 are shifted versions of the first source signals. Similarly the lower right corner has four sources with higher coherence values. These are the signals corresponding to positions 1,7,2,4. The signals in positions 7,2,4 correspond to the second source signal, while the signal in position 1 is a mixed signal.

4.4.4 Acoustic Sensing Fiber Example

Now that we have looked at several examples calculating the array coherence for synthetic data, we will try this method with some real data from an acoustic sensing fiber. This acoustic sensing fiber has 60 channels of data, each spaced 5 meters apart and was deployed in a trench outdoors and then covered in sand. For this example, the two sources of vibration considered are an air conditioning unit outside of a nearby building and a bass guitar. The data for these two sources was collected separately. To create a coherence matrix we choose four channels along the fiber as our response points, channels 20, 25, 30, and 35. Each channel is spaced 5 meters apart along the fiber. Then our four signal pairs used in the coherence matrix are as follows:

- Sensor pair 1: $s_{1,1}$: guitar at channel 20, $s_{1,2}$: AC at channel 20
- Sensor pair 2: $s_{2,1}$: guitar at channel 25, $s_{2,2}$: AC at channel 25
- Sensor pair 3: $s_{3,1}$: guitar at channel 30, $s_{3,2}$: AC at channel 30
- Sensor pair 4: $s_{4,1}$: guitar at channel 35, $s_{4,2}$: AC at channel 35

Then we obtain the coherence matrix as shown in Figure 4.9 with $f = 80$.

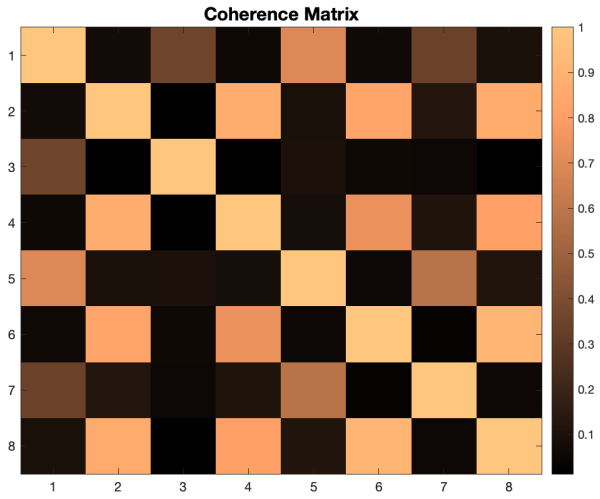

Figure 4.9: Coherence matrix for data obtained from acoustic sensing fiber buried in a trench outside. The two sources are an air conditioning unit outside of a nearby building and a bass guitar.

We can then apply the SRCM and SAMD permutations to try and recover the groupings of signals that correspond to the same sources. We expect rows 1,3,5,7 of the coherence matrix to be grouped together as they are all channels of data from the bass guitar and rows 2,4,6,8 to be grouped together since they are channels of data from the air conditioning unit.

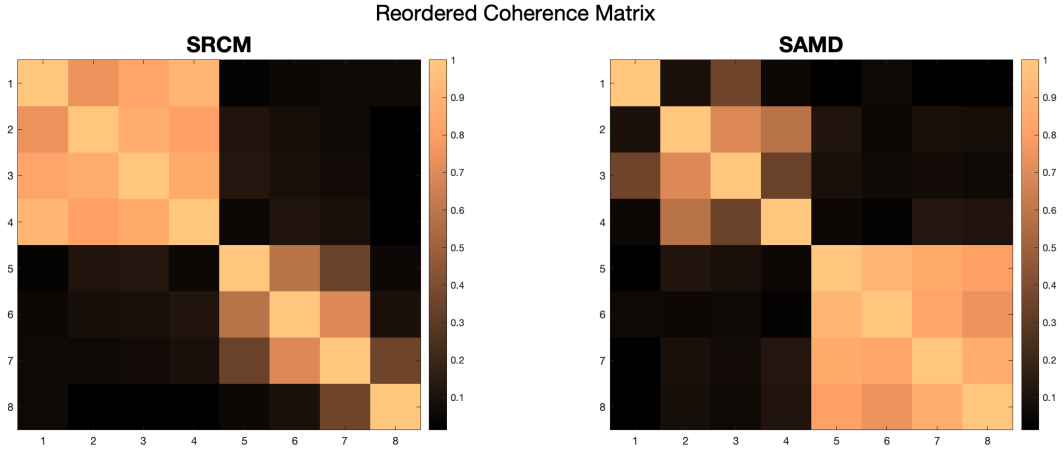

Figure 4.10: Reordered coherence matrices for data obtained from acoustic sensing fiber using the SRCM and SAMD permutations.

In Figure 4.10 we obtain the coherence matrix with permutation vector $p = [6\ 4\ 3\ 8\ 7\ 5\ 1\ 3]$ for the SRCM permutation and permutation vector $p = [3\ 5\ 1\ 7\ 8\ 6\ 2\ 4]$ for the SAMD permutation. Both methods have successfully grouped together the corresponding signals, but have done so with different permutation vectors and even with some channels in the guitar group having lower coherence.

Chapter 5

Conclusions and Future Work

In this thesis we have presented a method of improving the separation of signals from multiple physical quantities detected by sensor arrays. We first studied existing applications and methods of signal separation. Then we focused on the ICA problem setup and existing numerical algorithms, namely the gradient ascent and fixed point iteration optimization algorithms with kurtosis and negentropy as objective functions.

In Chapter 3 we presented a method of comparing signal separation results from numerical optimization problems with kurtosis and negentropy objective functions for single frequency source signals. We focused on the fixed point iterations over the gradient ascent algorithms since the fixed point iteration does not require the estimation of a step size. This step size parameter can further affect the results if not estimated correctly for the specific problem context. We found in all cases with synthetic data that the negentropy fixed point iteration algorithm resulted in successful signal separations in more algorithm runs than the kurtosis fixed point iteration. All the test cases explored in this chapter were for single frequency sources since this thesis was developed with the laboratory testing of the acoustic and magnetic sensing fiber in mind. Using single frequency sources also allowed us to evaluate the performance of the signal separation over many algorithm runs. In the future, the optimization algorithms presented might be used to separate signals of more complicated wave-forms that are not necessarily the result of single frequency sources. As seen with the data from the acoustic and magnetic sensing fiber [7], the separation was not as successful when the fiber

response from single frequency sources also had responses at harmonics of those frequencies. Additionally, modifications to the fixed point iterations themselves should be made to account for time lags present with real data instead of pre-processing the mixtures to transform them into instantaneous mixtures.

Chapter 4 presented a way of applying the ICA model to an array of sensors. With this problem setup we developed a method of maintaining coherence throughout resulting spatial arrays and pairing together signals which correspond to each other. We were able to do so by calculating the magnitude squared coherence between each pair of sensors and subsequently applying sparse matrix permutations to reorder the columns and rows of the matrix. In the limited examples shown, we find that this method successfully pairs together the signals corresponding to the same source. Further work should be done to develop an automated and adaptive method to determine at what frequency this method should be carried out to achieve reliable results. Also, the advantages and drawbacks of each matrix reordering technique, SRCM and SAMD, could be better characterized for further examples and through theoretical analyses.

Bibliography

[1] Kaber Allaire. *Numerical Linear Algebra*. Springer, 2008.

[2] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4): 886–905, 1996. doi: 10.1137/S0895479894278952. URL https://doi.org/10.1137/S0895479894278952.

[3] P.A.. A. Amundsen, S.. Ding, B.K.. K. Datta, T.. Torkildsen, and A.. Saasen. Magnetic Shielding During MWD Azimuth Measurements and Wellbore Positioning. *SPE Drilling & Completion*, 25(02):219–222, 07 2009. ISSN 1064-6671. doi: 10.2118/113206-PA. URL https://doi.org/10.2118/113206-PA.

[4] Marco Congedo, Cedric Gouy-Pailler, and Christian Jutten. On the blind source separation of human electroencephalogram by approximate joint diagonalization of second order statistics. *Clinical Neurophysiology*, 119(12):2677–2686, December 2008. doi: 10.1016/j.clinph.2008.09.007. URL https://hal.archives-ouvertes.fr/hal-00343628.

[5] E. Cuthill and J. McMkee. Reducing the bandwidth of sparse symmetric matrices. *ACM 69': Proceedings of the 1969 24th National Conference*, pages 157–172, 08 1969. doi: 0.1145/800195.805928.

[6] Daniel Finfer, Veronique Mahue, Sergey Shatalin, Teja Parker, and Mahmoud Farhadiroushan. Borehole flow monitoring using a non-intrusive passive distributed acoustic sensing (das). *Proceedings - SPE Annual Technical Conference and Exhibition*, 5:3597–3605, 10 2014. doi: 10.2118/170844-MS.

[7] Zachary Hileman, Daniel Homa, Lingmei Ma, Bo Dong, Eileen Martin, Gary Pickrell, and Anbo Wang. Development of a multimaterial optical fiber for fully distributed magnetic sensing applications. *IEEE Sensors Letters*, 6(1):1–4, 2022. doi: 10.1109/LSENS.2021.3137640.

[8] A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999. doi: 10.1109/72.761722.

[9] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(00)00026-5. URL https://www.sciencedirect.com/science/article/pii/S0893608000000265.

[10] Kenneth Lange. *Optimization*. Springer, second edition edition, 2013.

[11] Nathaniel J. Lindsey and Eileen R. Martin. Fiber-optic seismology. *Annual Review of Earth and Planetary Sciences*, 49(1):309–336, 2021. doi: 10.1146/annurev-earth-072420-065213. URL https://doi.org/10.1146/annurev-earth-072420-065213.

[12] Huan Liu, Roman Shor, and Simon Park. Intelligent Wellbore Path Estimation Using Multiple Integrated MEMS Sensors. Day 2 Wed, March 06, 2019, 03 2019. doi: 10.2118/194127-MS. URL https://doi.org/10.2118/194127-MS. D021S010R003.

[13] Sentek Instrument LLC. Distributed acoustic sensing. http://sentekinstrument.com/das/.

[14] Shoji Makino. *Audio Source Separation*. Springer, 2018.

[15] Ali Mansour, Nabih Benchekroun, and Cédric Gervaise. Blind separation of underwater acoustic signals. 3889, 03 2006. doi: 10.1007/11679363_23.

[16] Wang Naik. *Blind Source Separation Advances in Theory, Algorithms and Applications*. Springer, 2014.

[17] William D. Penny. Signal processing course. https://www.fil.ion.ucl.ac.uk/~wpenny/course/course.html.

[18] Dennis W. Ricker. *Echo Signal Processing*. Springer, 2003.

[19] Cox Treeby. Bk-wave: Matlab toolbox for simulation and reconstruction of photoacoustic wave fields. *Journal of Biomedical Optics*, 15, 03 2010.

[20] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. Brooks/Cole Cengage Learning, seventh edition edition, 2008.

[21] Eric W. Weisstein. Cross-correlation. https://mathworld.wolfram.com/Cross-Correlation.html, .

[22] Eric W. Weisstein. Cross-correlation theorem. https://mathworld.wolfram.com/Cross-CorrelationTheorem.htmll, .

[23] Liming Zhang, Ji Qi, Lixin Li, Kai Zhang, Zhixue Sun, Yongfei Yang, and Qin Luo. A forward modeling method based on electromagnetic theory to measure the parameters of hydraulic fracture. *Fuel*, 251:466–473, 09 2019. doi: 10.1016/j.fuel.2019.04.075.

Appendices

Appendix A

Comparison of Fixed Point Iterations

Ratio	Test 1	Test 2	Test 3	Test 4	Test 5
0-0.1	828	778	0	930	537
0.1-0.2	41	60	0	20	25
0.2-0.3	35	38	0	13	13
0.3-0.4	25	28	0	15	10
0.4-0.5	15	23	662	5	9
0.5-0.6	17	17	179	8	5
0.6-0.7	10	15	79	5	9
0.7-0.8	9	12	48	2	8
0.8-0.9	5	13	17	2	5
0.9-1	15	16	15	0	3

Table A.1: Table for evaluation of kurtosis fixed point iteration for the first source of each of the five test cases. The ratios in the left column are the ratio of the 2nd highest to highest peak amplitude of the resulting signal after separation. The values in the table are percentages of algorithm runs (1000 total algorithm runs for each test case).

Ratio	Test 1	Test 2	Test 3	Test 4	Test 5
0-0.1	830	792	0	898	501
0.1-0.2	41	48	0	47	38
0.2-0.3	36	39	0	14	13
0.3-0.4	23	26	0	18	11
0.4-0.5	17	20	0	6	8
0.5-0.6	18	17	513	8	7
0.6-0.7	8	16	317	4	5
0.7-0.8	9	12	88	3	2
0.8-0.9	7	14	68	1	8
0.9-1	11	7	14	1	3

Table A.2: Table for evaluation of kurtosis fixed point iteration for the second source of each of the five test cases. The ratios in the left column are the ratio of the 2nd highest to highest peak amplitude of the resulting signal after separation. The values in the table are percentages of algorithm runs (1000 total algorithm runs for each test case).

Ratio	Test 1	Test 2	Test 3	Test 4	Test 5
0-0.1	962	918	0	980	446
0.1-0.2	10	16	0	13	140
0.2-0.3	7	15	0	2	188
0.3-0.4	4	11	0	2	161
0.4-0.5	3	9	937	0	14
0.5-0.6	4	10	39	0	24
0.6-0.7	2	3	11	2	8
0.7-0.8	1	5	7	0	2
0.8-0.9	1	5	5	1	1
0.9-1	6	8	1	0	1

Table A.3: Table for evaluation of negentropy fixed point iteration for the first source of each of the five test cases. The ratios in the left column are the ratio of the 2nd highest to highest peak amplitude of the resulting signal after separation. The values in the table are percentages of algorithm runs (1000 total algorithm runs for each test case).

Ratio	Test 1	Test 2	Test 3	Test 4	Test 5
0-0.1	962	919	0	941	382
0.1-0.2	10	20	0	52	71
0.2-0.3	7	13	0	2	84
0.3-0.4	4	11	0	2	61
0.4-0.5	4	10	0	0	86
0.5-0.6	3	9	897	0	129
0.6-0.7	2	0	77	2	103
0.7-0.8	2	3	13	0	21
0.8-0.9	1	2	8	1	8
0.9-1	5	7	5	0	14

Table A.4: Table for evaluation of negentropy fixed point iteration for the second source of each of the five test cases. The ratios in the left column are the ratio of the 2nd highest to highest peak amplitude of the resulting signal after separation. The values in the table are percentages of algorithm runs (1000 total algorithm runs for each test case).