2022-12

ROBUSTIFYING SIGNAL SUBSPACE METHODS WITH GROUP SLOPE

Oh, Micah Y.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROBUSTIFYING SIGNAL SUBSPACE METHODS WITH GROUP SLOPE

by

Micah Y. Oh

December 2022

Thesis Advisor:	Robert L. Bassett
Second Reader:	Johannes O. Royset

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
ROBUSTIFYING SIGNAL SUBSPACE METHODS WITH GROUP SLOPE

5. FUNDING NUMBERS

6. AUTHOR(S) Micah Y. Oh

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Isolating and identifying specific acoustic signals in an underwater acoustic environment is challenging due to the presence of noise and potentially interfering signals. We construct an optimization problem that includes a time-sparse interference term to account for the unique form of interfering signals and use this optimization problem to detect audio samples contaminated with interference. To enforce sparsity in our interference term, we use the Group Sorted L-One Penalized Estimation norm as a sparsity-inducing penalty. Applying this estimator to more than thirty cases of simulated and real-world acoustic data demonstrates its ability in more than 90% of those cases to detect interference in acoustic data. A standard runtime of about one second for a ten-second data sample allows our contributions to be used for interference detection in real time.

14. SUBJECT TERMS
acoustic, underwater, signal, interference, slope, regression, noise, simulation, sparsity

15. NUMBER OF PAGES
79

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ROBUSTIFYING SIGNAL SUBSPACE METHODS WITH GROUP SLOPE

Micah Y. Oh
Ensign, United States Navy
BS, United States Naval Academy, 2021

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 2022

Approved by: Robert L. Bassett
Advisor

Johannes O. Royset
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Isolating and identifying specific acoustic signals in an underwater acoustic environment is challenging due to the presence of noise and potentially interfering signals. We construct an optimization problem that includes a time-sparse interference term to account for the unique form of interfering signals and use this optimization problem to detect audio samples contaminated with interference. To enforce sparsity in our interference term, we use the Group Sorted L-One Penalized Estimation norm as a sparsity-inducing penalty. Applying this estimator to more than thirty cases of simulated and real-world acoustic data demonstrates its ability in more than 90% of those cases to detect interference in acoustic data. A standard runtime of about one second for a ten-second data sample allows our contributions to be used for interference detection in real time.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

DOA	Direction of Arrival
DOD	Department of Defense
FDR	False Discovery Rate
FNR	False Negative Rate
FPR	False Positive Rate
Group SLOPE	Group Sorted L One Penalized Estimation
I/Q	In-phase and Quadrature
NPS	Naval Postgraduate School
RANSAC	Random Sample Consensus
SLOPE	Sorted L One Penalized Estimation
SOS	second-order section
USN	U.S. Navy

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Underwater acoustic detection is a critical capability for the U.S. Navy's ability to detect potential undersea threats. Interference, either from a natural or intentionally malicious source, hinders the capacity to identify underwater acoustic signals and estimate their Direction of Arrival (DOA). Our goal is to construct a method capable of detecting and removing contaminated samples, which leads to more accurate DOA estimation of the target signal using the remaining data. In this paper, we construct and test an extension of signal subspace methods that employs the Group Sorted L-One Penalized Estimation (Group SLOPE) method as a way of detecting acoustic data contaminated with interference. The method needs to demonstrate accuracy in its detection effort as well as run quickly enough that we can process data in real time, meaning the runtime does not exceed the duration of the data we input.

A key assumption for a generic DOA estimation is that noise present in the data is Gaussian. For this problem, we instead introduce an interference term separate from noise that is sparse in time. This interference term captures the signals present in the sample that are not the target signal and do not fall under the Gaussian noise assumption. In regression, a computationally tractable method for sparsity enforcement uses a penalty norm in the objective function. For this penalty norm, we use the recently developed Group SLOPE norm applied to the interference term.

Since the optimization problem minimizes over both the interference term, Δ, and a term representing the interaction of the target signal with the sensor array, A, we employ alternating minimization as the solution algorithm. The minimization in A uses a proven closed form solution while the minimization in Δ involves Group SLOPE. With a number of useful reductions, the Group SLOPE problem becomes significantly simpler than standard Group SLOPE and we solve using a calculation of the proximal operator. We implement this alternating minimization solution in a Python and Cython package.

To test the estimator we apply it to both simulated and real world data. In the simulated data, we construct three types of interference and test the estimator's performance at detecting each. We vary the parameters of the simulated data and generate twenty-nine instances

of simulated data to determine the breadth of the estimator's capabilities. Even though these cases are designed to stretch the estimator's capabilities, we see moderate to perfect detection in twenty-six of these cases and the estimator produces meaningless results in only three. In all cases where we achieve meaningful results, DOA estimation improves after removing the samples labeled as contaminated as compared to DOA estimation with the full dataset. In addition, the runtime for a ten second sample typically remains below a second with a maximum of seven seconds in the twenty-nine instances of simulated data. The runtime is short enough that employment in real time is entirely feasible. Testing on real data supplied by the NPS Physics Department from an acoustic array off the coast of Iceland also demonstrates successful interference detection and removal.

Acknowledgments

This thesis would not happened without the guidance and help of numerous people throughout the entire process. First, I must thank my parents for their support throughout this project. While I may have frequently bored them with needless descriptions of technical details, they were always and have always been available to listen and counsel through struggle and frustration. I would like to thank Dr. Robert Bassett for his instruction and for helping to direct and push me when I lacked direction. He not only taught me many of the details of the theory and implementation of our work, but also how to communicate, manage workflow, and continuously learn in a project of this nature. Working with him has been a true pleasure and an incredibly valuable experience. I am also thankful for Dr. Johannes Royset for initially introducing me to Dr. Bassett and this project and then for providing guidance as my second reader. In addition, Dr. Kai Gemba of the NPS Physics Department gave guidance in understanding the physics in underwater acoustics and supplied the data we use in this project. I would also like thank the Office of Naval Research Science of Autonomy Program for sponsoring this work and expressing interest in these types of problems. Finally, I would like to thank the members of my cohort here at NPS, in particular my fellow Ensigns, as well as any other students I have worked with during my time here. I have learned a great deal from classmates who are far more experienced than myself and relied on the support of many of my classmates throughout my time at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

A critical component of the U.S. Navy's mission across the world is sea control, which is the Navy's ability to exert dominance over certain regions (Chief of Naval Operations 2022). To exert sea control, the capability for threat detection must exist throughout the designated region. Conflict analysis frequently points to the value of first detection in contested environments. A force that detects a hostile force first has a clear advantage in any ensuing conflict. Early threat detection gives the U.S. Navy the ability to make decisions about potential threats, which leads to sea control due to the advantage gained. As a result, our Navy platforms are outfitted with complex and varied technologies built specifically for the purposes of threat detection.

The submarine is the maritime platform most equipped for stealth operations. The very nature of a subsurface threat immediately reduces the potential methods to employ for threat detection. A critical method for submarine detection, as well as detection of other maritime vessels, is the use of underwater acoustic signals. As a means of detection, underwater acoustic sensors are capable of receiving sound waves and determining the direction of arrival of the acoustic signal. The goal of this technology is to detect and pinpoint specific signals regardless of the presence of other signals and noise in the surroundings. However, in the busy acoustic environment of the ocean, identifying the target signal's direction is not a trivial task. The heart of our problem is to determine the direction of a target from a number of samples from an audio clip.

In this thesis, we construct an optimization problem that identifies sparse interference in acoustic data. We create an alternating minimization algorithm and remove the samples identified as contaminated, allowing us to conduct more accurate Direction of Arrival (DOA) estimation for a specific target signal.

1.1 Technical Development

The data format we examine is a multichannel audio clip in complex In-phase and Quadrature (I/Q) representation with one channel per hydrophone in the sensor array. A standard

approach to DOA estimation for data of this nature uses signal subspace methods to determine the target's bearing and signal strength. The probability model for a DOA estimation signal subspace method is

$$x(t) = A\, s(t) + \epsilon(t). \tag{1.1}$$

In this probability model, the $x(t) \in \mathbb{C}^m$ is a vector of received signals at a specific time, $\epsilon(t) \in \mathbb{C}^m$ is a vector of random noise terms that are assumed to be Gaussian, $A \in \mathbb{C}^{m\times d}$ is a time-invariant matrix based on the target signal's incidence angle and the sensor geometry, and $s(t) \in \mathbb{C}^m$ is a vector representing the target signal. Using the probability model in Equation (1.1) it is possible to create an estimator that enables estimation of the direction and strength of a specific target signal. This estimator performs well when the Gaussian noise assumption holds, but, as previously stated, interference is a significant issue in the underwater acoustic environment that is not addressed by the probability model represented in (1.1).

1.2 Contributions and Outline

We choose to construct our own probability model partially based on the model described in Equation (1.1). To do this we introduce a term $\delta(t) \in \mathbb{C}^m$ that is a vector representing the interference at a specific time. This change gives the probability model

$$x(t) = A\, s(t) + \delta(t) + \epsilon(t). \tag{1.2}$$

To distinguish $\delta(t)$ from our noise, $\epsilon(t)$, we assume that the interference in the acoustic data is sparse in time. This means that when we form the rows of a matrix Δ from the vectors $\delta(t)$ across some time span, the matrix should have some rows of all zeros and some rows with non-zero values representing interference. A tractable method for inducing sparsity of this nature in an optimization problem is including a penalty norm in the objective function. We choose to employ the Group Sorted L-One Penalized Estimation norm on Δ in the optimization problem we use to solve the probability model in Equation (1.2).

The resulting optimization problem minimizes over both the A matrix as well as the Δ matrix. To solve this optimization problem we create an alternating minimization algorithm that quickly computes its minimizers. While Group SLOPE typically requires a computationally

expensive solution algorithm, the structure of this specific optimization problem allows us to reduce the Group SLOPE minimization to a more computationally simple problem. The final result of our work towards this alternating minimization algorithm is a Cython/Python package that quickly and effectively solves the optimization problem.

In the remainder of this thesis we describe the background of this problem, our own efforts, and the results we generate in greater detail. In Chapter 2, we conduct a literature review that covers both other methods used for interference detection and the foundational methods for the work we do. In Chapter 3, we describe the evolution of the probability model in Equation (1.2) to an optimization problem and then detail the creation and implementation of the alternating minimization solution algorithm. In Chapter 4, we detail the results of testing the estimator using both simulated and real acoustic data. Finally in Chapter 5, we reiterate the critical pieces of this work and discuss potential areas for future research.

1.3 Notation

Before proceeding, we introduce the notation we use throughout this thesis. We denote matrices with capital letters and vectors with lowercase letters. For a specific vector x, we use x_i to denote its ith element, but use $x_{(i)}$ to denote its ith largest element so that $(x_{(1)}, ..., x_{(n)})$ arranges the elements of x in non-increasing order. We denote the Frobenius norm of a vector x as $\|x\|_F$, the L1 norm as $\|x\|_1$, and the L2 norm as $\|x\|_2$. The transpose of a vector x is x^T and the conjugate transpose is x^*. Finally, $\|x\|_{\sharp,\lambda}$ is used to denote the Group SLOPE norm with penalty vector λ. This notation enables us to communicate the method we develop to determine the direction of a target from a number of samples of an audio waveform.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Literature Review

In this chapter we discuss other methods for interference detection as well as the foundations for our own contributions. We examine the flaws in assuming Gaussian noise and estimators which do not require that assumption. We then explore Random Sample Consensus (RANSAC), which trims outliers in a dataset, as a potential method for removing the influence of contaminated data. Following this, we shift focus to the foundations of the work in this paper and discuss signal subspace methods as a way of estimating DOA of an acoustic signal. Finally, we introduce the Group SLOPE norm as a sparsity-inducing regularizer.

2.1 Robust Estimation in Signal Processing

Noise is a critical concept in nearly all methods for signal processing. Noise captures the idea that the space in which we send and receive signals is never completely empty. Whether studying electromagnetic, acoustic, or any other type of signal, noise will be present and can limit capability for signal detection (Zoubir et al. 2012). When estimating a signal, we include noise terms to account for a potentially chaotic signal space.

The standard assumption is that noise terms of this nature have a Gaussian distribution. This Gaussian assumption is frequently valid due to the central limit theorem and is helpful in simplifying problems and improving solutions. On the other hand, it is often used even when the noise may not be Gaussian (Zoubir et al. 2012). Impulse noise is an example of a type of noise that is clearly not Gaussian. Instead, impulse noise has heavier tails than a Gaussian distribution and is frequently found in electromagnetic signals (Zoubir et al. 2012). In this case, received signals have some potentially malicious contamination or interference present. This does not match the Gaussian assumption associated with noise. In fact, due to our assumed lack of knowledge about the nature of the interference, we want to be able handle interference with potentially unknown distribution or signal type, which leads to the construction of a specific interference term that relies on the sparsity assumption to distinguish itself from the noise term.

There have been other attempts to approach problems where the Gaussian noise assumption

is assumed to be invalid, otherwise known as robust estimation. These attempts use statistical methods to construct different estimators built to handle outliers in the provided data (Zoubir et al. 2012). These estimators use the concept of the breakdown point (BP) which represents the percentage of contaminated samples to indicate the presence of outliers in data (Zoubir et al. 2012). Identifying outliers and removing them from the data before conducting analysis matches the approach we will take of removing contaminated samples and using Direction of Arrival estimation. When approaching multichannel data, using estimators such as the MM-estimator, Minimum Volume Ellipsoid Estimator, or Minimum Covariance Determinant Estimator should remove the need for the Gaussian noise assumption, robustifying the signal detection process (Zoubir et al. 2012). Our efforts build on previously existing signal subspace methods to construct an estimator that can handle interfering signals contaminating a bounded proportion of samples in an audio signal.

2.2 RANSAC

RANSAC is another method applied in situations that require processing data with some amount of contaminated samples (Fischler and Bolles 1981). Robust estimation methods tackle this problem by adjusting the noise term. On the other hand, RANSAC's general idea is a bit more simple. RANSAC looks for data with significant errors in it and attempts to eliminate the problematic data through a random selection process. Even simple least squares regression models can quickly demonstrate the major detriment caused by a single extreme data point. RANSAC counteracts this by identifying these problematic outliers prior to analysis (Fischler and Bolles 1981).

RANSAC is a three step algorithm that hinges primarily on the concept that the non-contaminated data should have some common features among itself (Fischler and Bolles 1981). First, we select a random subset of the data and fit the model to that data. Using the model output, we determine which input data are within a designated error tolerance of the model output and keep these points as a new consensus set. This consensus set should ideally include only uncontaminated data points. If this consensus set is larger than some predetermined value, then we solve the model with this consensus set and use the given output. This process can continue to iterate with different initial subset selection if the first run does not yield the desired results (Fischler and Bolles 1981).

While RANSAC does effectively trim outliers from datasets, it also throws away or does not make use of a significant amount of data which is not always a feasible option. In addition, this methodology eliminates any data points that have significant differences from the consensus within the data (Fischler and Bolles 1981). While this is the intent of this method, it also is dangerous as there is potential that it removes samples that do have value from consideration. Despite this, a model of this nature would likely have some functionality for the problem as RANSAC could detect data with significant interference present. However, in situations with a moderate to high amount of contamination, RANSAC has difficulty identifying a subset of data that does not have any contaminated data and loses its effectiveness. In the model, we are able to handle datasets with moderate to high amounts of contaminated data without completely sacrificing performance.

2.3 Subspace Methods

In this thesis, we use signal subspace methods for the DOA Estimation. These methods are linear probability models built to identify certain designated characteristics of signals while accounting for the noise within the space (Paulraj et al. 1993). The most simple and generic signal subspace probability model takes the form

$$x(t) = A\, s(t) + \epsilon(t), \tag{2.1}$$

where $x(t) \in \mathbb{C}^m$ represents the received signal at time t, $s(t) \in \mathbb{C}^d$ is the actual signal we want to identify, $A \in \mathbb{C}^{m \times d}$ is a matrix of parameters based on sensor geometry applied to that signal, and $\epsilon(t) \in \mathbb{C}^m$ is the noise term (Paulraj et al. 1993). The signals in this subspace method are complex-valued since they are in the In-phase and Quadrature (I/Q) format.

A standard method for solving the probability model in Equation (2.1) is using a Maximum Likelihood Estimator (MLE) (Paulraj et al. 1993). If we assume ϵ is Gaussian and independently and identically distributed than the MLE is the optimization problem

$$\min_{A,S} \|X - AS\|_2^2. \tag{2.2}$$

Since this reduces to a least squares regression, optimization problem (2.2) has a closed-form solution which yields a DOA estimate for the target signal.

Beamforming is another method that attempts to handle a wide variety of noise in the signal space (Paulraj et al. 1993). At it core, beamforming identifies a specific azimuth angle and measures the signal power in that direction. Beamforming first discretizes the possible azimuth angles into a number of directions. The beamforming model then creates a steering vector for each direction and selects the steering vector with the maximum signal power. The azimuth angle used to construct this steering vector is the direction selected as an estimate of DOA and signals from other directions are ignored (Paulraj et al. 1993). Beamforming is a key signal processing method used frequently in problems with an interfering signal that should be ignored. It is highly effective at identifying and strengthening signals with some degree of difference in the DOA. However, this method struggles with overlapping signals, does not scale to multiple target signals as signal subspace methods does, and has potential to miss narrow peaks in signal power through its discretization process.

2.4 Group SLOPE

To induce sparsity in the interference estimate, we employ Group SLOPE. Although the overall problem has unique structure that changes how we apply Group SLOPE, in this section we will provide a foundation for the generic Group SLOPE method. Group SLOPE is a generalization of a simpler method SLOPE which, in turn, is a generalization of the more common LASSO regression method. LASSO regression takes the standard regression optimization problem and adds a penalty term on the coefficients using the L1 norm (Brzyski et al. 2019). Minimization of the L1 norm of some matrix Δ forces many Δ values towards zero, inducing sparsity in Δ. In the specific problem we solve, we will instead enforce sparsity on a designated interference term.

We employ a generalization of LASSO developed in 2013 known as the Sorted L-One Penalized Estimation (SLOPE) method (Bogdan et al. 2013). For a vector β, SLOPE uses a norm $\sum_{i=1}^{m} \lambda_i |\beta|_{(i)}$ where the λ and β values are sorted in descending order such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$ and $|\beta|_{(1)} \geq |\beta|_{(2)} \geq \ldots \geq |\beta|_{(m)}$ (Bogdan et al. 2015). The sorted vector λ means that larger β norm values are penalized more than smaller values. In Bogdan et al. (2015), SLOPE problems are solved using proximal gradient descent with the least squares piece as the smooth and convex piece and the penalty norm as the non-smooth, but still convex, piece.

In the specific problem we handle in this thesis, the data and the penalty term Δ are $m \times n$ matrices where m is the number of channels in the sensor array and n is the number of data readings we have in time. We must then enforce sparsity uniformly across different channels for each time-step. To accomplish this, we group the data by each time t. We enforce sparsity across channels so that interference detected in one channel at time t ensures detection in other channels at that same time. This method is a generalization of SLOPE known as Group SLOPE.

For Group SLOPE, we apply the L1 norm to each group of data points and sort accordingly. This means that $\|\Delta\|_{(1)} \geq \|\Delta\|_{(2)} \geq ... \geq \|\Delta\|_{(n)}$ where $\|\Delta\|_{(n)}$ is the L1 norm on the Δ values in the nth timestep. We denote the resulting Group SLOPE norm of a matrix Δ as $\|\Delta\|_{\sharp,\lambda}$ where $\sharp$ indicates use of the Group SLOPE norm and λ indicates the use of sorted parameters that influence penalty severity. (In cite, they use pox grad descent) Implementing this norm into a regression problem requires use of a proximal gradient descent algorithm to solve (Brzyski et al. 2019). For this specific problem, we are able to reduce the problem so we will only require the proximal operator and not the entire algorithm to solve.

False Discovery Rate (FDR) is a critical piece of why the Group SLOPE method is powerful (Brzyski et al. 2019). Group FDR is a ratio of the number of groups falsely detected as nonzero to the total number of groups identified as nonzero. In Group SLOPE, FDR can be controlled by different λ selection methods due to the inherent flexibility surrounding those parameters. In addition, that flexibility means that this method is a generalization of LASSO, enabling us to use Group SLOPE for situations for which LASSO can be used while taking advantage of attractive FDR guarantees.

While the Group SLOPE estimator does have a significant amount of flexibility, the problem we tackle in this thesis does require some additional functionality beyond what previously existed and is described here. With the introduction of an interference term to penalize in the subspace model, we already have a slightly different type of penalized regression. In addition, acoustic waveform data is complex-valued when in In-phase and Quadrature (I/Q) format and, as such, some of the linear algebra behind the implementation of Group SLOPE needed to be examined. Finally, with no existing Python functionality, we decided to construct the algorithm in Python based on the previous work done in R with the grpSLOPE package.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Method Development and Implementation

To construct a method for interference detection, we develop a probability model based on signal subspace methods with an additional term for interference. We then formulate a minimization optimization problem that we solve using an alternating minimization algorithm and the Group SLOPE method. Finally, we implement this algorithm in Python and use it to generate results.

3.1 Optimization Problem

A typical signal subspace methods DOA estimation equation in matrix form is

$$X = AS + \epsilon, \tag{3.1}$$

where $X \in \mathbb{C}^{m \times n}$ represents the received signal found in the data, $S \in \mathbb{C}^{d \times n}$ is the actual signal we want to identify, $A \in \mathbb{C}^{m \times d}$ is a matrix of parameters based on sensor geometry applied to that signal, and $\epsilon \in \mathbb{C}^{m}$ is the Gaussian noise term. For this problem, we require the acoustic data in the I/Q representation of a complex waveform. I/Q representation is a method of depicting a wave in two pieces: a real number known as the In-phase part and an imaginary number known as the Quadrature part. The resulting complex number represents a specific point in the wave. This datatype allows us to adequately represent a wave in a manner that supports its analysis for DOA estimation.

The solution technique for signal subspace methods are discussed in Chapter 2 and shown in Equation (2.2). However, the probability model in Equation (3.1) does not directly address an interfering signal. As such, we choose to introduce $\Delta \in \mathbb{C}^{m \times n}$ as a term to represent this potential interference and this leads to

$$X = AS + \epsilon + \Delta. \tag{3.2}$$

With the inclusion of Δ as an interference term we have the optimization problem

$$\min_{A,S,\Delta} \|X - A\,S - \Delta\|_F^2. \tag{3.3}$$

This optimization problem misses the fact that we must assume sparsity for Δ to indicate that only some of the data is contaminated and distinguish Δ from ϵ. Including a Group SLOPE sparsity penalty, denoted as $\|\Delta\|_{\sharp,\lambda}$, gives

$$\min_{A,S,\Delta} \|X - A\,S - \Delta\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{3.4}$$

Our minimization in problem (3.4) is not in a format that we can easily use the Group SLOPE functionality since $\|X - A\,S - \Delta\|_F^2$ has a different form than a standard penalized regression problem. We now require some reformulation or simplification. We first can split the minimization into each of its pieces and write an inner minimization over S as

$$\min_{A,\Delta} \min_{S} \|X - A\,S - \Delta\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{3.5}$$

Using the optimality condition in S as defined in Section 1.D of Royset and Wets (2021) gives

$$A^*(X - AS - \Delta) = 0 \Rightarrow S = (A^*A)^{-1}A^*(X - \Delta),$$

so that problem (3.4) becomes

$$\min_{A,\Delta} \|X - P_A(X - \Delta) - \Delta\|_F^2 + \|\Delta\|_{\sharp,\lambda}, \tag{3.6}$$

where $P_A = A(A^*A)^{-1}A^*$ is the orthogonal projection onto the column space of A. We can rewrite problem (3.6) as

$$\min_{A,\Delta} \|(I - P_A)(X - \Delta)\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{3.7}$$

This form of the optimization problem as presented in problem (3.7) is the version we examine throughout the remainder of this work. The objective function of optimization problem (3.7) is non-convex due to the Group SLOPE term we use to enforce sparsity which increases the difficulty in selecting a suitable algorithm to solve optimization problem (3.7).

Having two separate pieces to minimize over, suggests that an alternating minimization algorithm could be a tractable approach.

3.2 Alternating Minimization

Alternating minimization first solves optimization problem (3.7) in one variable, A, while holding the other variable, Δ, constant. We then hold A constant as the minimizer of the first step and solve optimization problem (3.7) with Δ as the variable. This process continues until some tolerance is reached between steps. For a more in-depth introduction to alternating minimization see chapter 14 of Beck (2017) or sections 1.J and 10.A in Royset and Wets (2021). While this technique does not guarantee an optimal solution for non-convex problems, we can prove convergence to a critical point as defined in Grippo and Sciandrone (2000). This initially seems improbable since the objective function of optimization problem (3.7) is not continuously differentiable so critical points do not exist. However, we are able to reformulate this minimization so it is continuously differentiable, demonstrate equivalence of the reformulation to optimization problem (3.7) and apply the theorem in Grippo and Sciandrone (2000) to prove convergence to a critical point of the reformulation. The theorem for this alternating minimization convergence is displayed in Theorem 1.

Theorem 1 *Problem* (3.7) *can be reformulated as*

$$\min_{\substack{P_A \in \mathbb{C}^{c\times c} \\ P_A^2 = P_A \\ P_A^* = P_A}} \quad \min_{\substack{c\in\mathbb{R} \\ \Delta\in\mathbb{C}^{c\times T} \\ \|\Delta\|_{\sharp,\lambda}\leq c}} \quad \|(\Delta - X)\|_F^2 - \mathrm{Tr}\left[P_A(\Delta - X)(\Delta - X)^*\right] + c,$$

which is a minimization of a continuously differentiable function over closed, nonempty, and convex sets, and therefore optimization problem (3.7) *converges to a critical point of that reformulation.*

The proof for Theorem 1 is in Appendix A.1.

3.2.1 Minimizing in A

The first step in the alternating minimization procedure is minimizing problem (3.8) over A with some initialized Δ value which gives

$$\min_{A} \|(I - P_A)(X - \Delta)\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{3.8}$$

Because the penalty does not include any A term, it suffices to minimize the optimization problem

$$\min_{A} \|(I - P_A)(X - \Delta)\|_F^2 . \tag{3.9}$$

For this minimization, we have proven a closed-form solution that we use for this step of the alternating minimization. To derive this closed-form solution we prove Theorem 2.

Theorem 2 *The closed-form solution to the minimization over the projection matrix of A is*

$$\underset{P_A}{\operatorname{argmin}} \ \|(I - P_A)(X - \Delta)\|_F^2 = \sum_{i=1}^{d} u_i u_i^*,$$

where u_i is the i-th left singular vector of $X - \Delta$. Singular vectors are sorted in descending order according to their singular values.

The proof for Theorem 2 is in Appendix A.2. This demonstrates that the closed-form solution of optimization problem (3.9) is to take $P_A = \sum_{i=1}^{d} u_i u_i^*$, where u_i is the i-th left singular vector of $X - \Delta$ assuming singular vectors are sorted in descending order according to their corresponding singular values. With a closed-form solution to the minimization in A, we can move on to the minimization in Δ.

3.2.2 Minimizing in Δ

The next step in alternating minimization is to solve optimization problem (3.7) in Δ with fixed A. Optimization problem (3.7) becomes

$$\min_{\Delta} \|(I - P_A)(X - \Delta)\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{3.10}$$

Our optimization problem now requires use of Group SLOPE. However, optimization problem (3.10) is not clearly in the format designed for Group SLOPE in Brzyski et al. (2019). While we could apply Proximal Gradient Descent to optimization problem (3.10), further reformulation actually gives a more efficient solution. This reformulation is demonstrated in Theorem 3.

Theorem 3 *Optimization problem* (3.10) *reduces to*

$$\min_{c} \sum_{i=1}^{n} (c_i - \|(I - P_A)Y_i\|)^2 + \sum_{i=1}^{n} \lambda_i c_{(i)},$$

where

$$\Delta_i = (I - P_A)\Delta_i = \frac{c_i}{\|(I - P_A)X_i\|_2}(I - P_A)X_i$$

defines the relationship between Δ_i *and* c_i.

The proof of Theorem 3 is demonstrated in Appendix A.3. With this formulation, we can determine the optimal c_i values with one evaluation of the proximal operator. We use these values in tandem with the relationship between c_i and Δ_i to return the optimal Δ_i values. After minimizing in Δ, we continue to iterate through alternating minimization as necessary to reach optimality. Upon completion of alternating minimization, we then have a Δ that indicates which data samples we identify as containing interference.

3.3 Implementation in Python

Although our work building Group SLOPE capability in Python and Cython is motivated by the existing grpSLOPE R package Brzyski et al. (2019), in the end much of the optimization problem formulation and solution technique results from taking advantage of the unique structure of optimization problem (3.7). We first expand Group SLOPE functionality to include complex valued problems. In addition, we greatly simplify the required code for optimization as compared to the grpSLOPE R package, due to the structure of optimization problem (3.7) and the reductions we are able to make. One of the key reductions is demonstrated in Appendix A.3. This reduction makes Proximal Gradient Descent unnecessary as a solution algorithm and allows us to solve the minimization problem with a single calculation of the proximal operator. For greater detail on Proximal Gradient Descent and proximal

operators see chapters 6 and 10 in Beck (2017). It should be noted that the SLOPE_prox function we construct specifically relies on pre-existing code found in the original SLOPE R package (Bogdan et al. 2015). The code that solves optimization problem (3.7) is displayed below:

```
import numpy as np
import scipy
cimport numpy as np
cimport cython

@cython.cdivision(True) #divide w/ c not python
@cython.boundscheck(False)  # Deactivate bounds checking
@cython.wraparound(False)   # Deactivate negative indexing.
cdef np.ndarray[np.float64_t, ndim=1] SLOPE_prox(np.ndarray[np.float64_t, ndim=1] y,
                np.ndarray[np.float64_t, ndim=1] lams):
'''
Returns 1D SLOPE prox.
'''

    #declare variables
    cdef:
        unsigned long i,j,k,n
        double d
        np.ndarray[np.float64_t, ndim=1] s, w, x, sign_y
        np.ndarray[np.uint_t, ndim=1] idx_i, idx_j
        np.ndarray[np.int64_t, ndim=1] argsrt

    #this is the prox_sorted_L1 function from grpSLOPE
    n = y.shape[0]
    sign_y = np.sign(y)
    y = np.abs(y)
    argsrt = np.argsort(-y) #argsort sorts in increasing order, so we negate
    y = y[argsrt]

    #allocate memory as necessary
    x = np.zeros(n, np.float64)
    s = np.zeros(n, np.float64)
    w = np.zeros(n, np.float64)
    idx_i = np.zeros(n, np.uint)
```

```
    idx_j = np.zeros(n, np.uint)

    k = 0
    for i in range(n):
        idx_i[k] = i
        idx_j[k] = i
        s[k] = y[i] - lams[i]
        w[k] = s[k]
        while k > 0 and (w[k-1] <= w[k]):
            k -= 1
            idx_j[k] = i
            s[k] += s[k+1]
            w[k] = s[k] / (i - idx_i[k] + 1)
        k += 1
    for j in range(k):
        d = w[j]
        if d < 0:
            d = 0
        for i in range(idx_i[j], idx_j[j] + 1):
            x[i] = d

    np.put(x, argsrt, x)
    x *= sign_y
    return x

@cython.cdivision(True) #divide w/ c not python
@cython.boundscheck(False)  # Deactivate bounds checking
@cython.wraparound(False)   # Deactivate negative indexing.
cpdef alternating_min(np.ndarray[np.cdouble_t, ndim=2] X, float alpha,
            np.ndarray[np.float64_t, ndim=1] lams, int max_itr=50,
            float stop_tol=1e-6):
    cdef:
        np.ndarray[np.cdouble_t, ndim=2] \\
            Delta = np.zeros((X.shape[0], X.shape[1]), complex)
        np.ndarray[np.cdouble_t, ndim=2] P_A = np.eye(X.shape[0], dtype=complex)
        np.ndarray[np.cdouble_t, ndim=2] \\
            I_minus_P_A = np.eye(X.shape[0], dtype=complex)
        np.ndarray[np.cdouble_t, ndim=2] I_minus_PA_times_X
        float I_minus_P_A_step = 1.0
```

```
        float c_step = 1.0
        np.ndarray[np.float64_t, ndim=1] c = np.zeros(X.shape[1])
        np.ndarray[np.float64_t, ndim=1] new_c, I_minus_PA_times_X_norms
        np.ndarray[np.float64_t, ndim=1] lam_vec = alpha*lams
        np.ndarray[np.cdouble_t, ndim=2] u
        int itr

    itr = 0
    for itr in range(max_itr):

        #min over A
        _, u = scipy.linalg.eigh(np.matmul((X - Delta), (X - Delta).conj().T),
                subset_by_index=(X.shape[0]-1, X.shape[0]-1),
                check_finite=False,
                overwrite_a=True)
        P_A = np.outer(u[:,0], u[:,0].conj())
        I_minus_P_A_step = np.linalg.norm(P_A + I_minus_P_A - \\
            np.eye(u.shape[0], dtype=complex)) #the algebra works here
        I_minus_P_A = np.eye(u.shape[0], dtype=complex) - P_A

        #min over Delta
        I_minus_PA_times_X = np.matmul(I_minus_P_A, X)
        I_minus_PA_times_X_norms = np.linalg.norm(I_minus_PA_times_X, axis=0)
        new_c = SLOPE_prox(I_minus_PA_times_X_norms, lam_vec)
        c_step = np.linalg.norm(c - new_c)
        c = new_c
        Delta = np.multiply(I_minus_PA_times_X, \\
            np.divide(c, I_minus_PA_times_X_norms))

        if c_step < stop_tol and I_minus_P_A_step < stop_tol:
            #Stopping tolerance reached
            break
return Delta, P_A
```

The inputs for this function are X as the data matrix, an alpha parameter that scales the vector λ, the vector λ, the maximum number of iterations allowed, and a stopping tolerance. Its primary result is an updated Δ. Beginning with the alternating_min function, we first declare all variables we will use within this function in Cython for greater efficiency.

The main portion comes within the for loop that represents the alternating minimization. First we calculate the solution to the minimization in A with the given inputs. We then determine the step-length between the previous $I - P_A$ matrix and the current solution. With this, we conduct the minimization in Δ. We calculate the necessary group norms and then input these norms into the SLOPE_prox function. This gives us a c vector which we can then use to calculate both a step-length and Δ. Once both step-lengths have reached a certain tolerance the loop ends and we return the optimal P_A as well as the optimal Δ.

Within the SLOPE_prox function we declare and allocate memory for variables and sort the norms to match the sorted vector λ (Bogdan et al. 2013). With the vector λ we calculate the proximal operator, return the data to its original order and return the calculated c values. While this code, and the SLOPE_prox function in particular, have some foundation in the R package grpSLOPE, we have greatly increased the efficiency of solving optimization problem (3.7) with a number of changes and reductions (Brzyski et al. 2019). In addition, this code represents new functionality in Python, since previous work in this area was written in R.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Method Performance Test Results

With a solution method developed, we need to demonstrate its functionality and test its performance. In this thesis, we examine the functionality and performance of the method on real-world and simulated data.

4.1 Simulated Data

Simulated data allows us to conduct a greater breadth of tests since we can easily alter parameters in the simulation to generate varied data. This is far more efficient than the necessary experimental load required for varying real-world data. In addition, simulated data allows us to know the true nature of the data since we are its creator. For our work, this means we have access to the true target signal as well as the exact samples that are contaminated. This means we can more easily evaluate the actual performance of the method. To ensure we can run the alternating minimization algorithm, we must simulate the data in the I/Q format required for input into the algorithm.

4.1.1 Construction

To imitate data collected by an underwater acoustic array, we require a few components in the simulated signal. First, we need the size of the array indicated by the number of sensors in the array and the spacing between each sensor. Next, we need to define the target signal we will see in the data with a wavelength, amplitude, Direction of Arrival, and frequency. We then define the distribution of the noise in the desired sample and build an interfering term. This interference can take the form of either additional noise or a specific directed signal interfering. In both cases, we must define the specific distribution of this interference and then determine the number of samples it will contaminate. After developing each of these terms, we generate the randomness necessary for the noise and interference terms. Finally, we piece together the signal by combining each piece for the desired number of data points. An example of the python code used for this simulated data is shown below.

```
num_channels = 50
delta = 1 #array spacing
lam = 4 #signal wavelength
theta = np.pi/4 #DOA
a = np.exp(2j*np.arange(0,num_channels)*np.pi*delta*np.sin(theta)/lam)
t = np.linspace(0,10,100000) #time
f = 300 #frequency
s = 1
in_phase = np.sin(2*np.pi*f*t)
quadrature = np.cos(2*np.pi*f*t)
eps_sigma = .1
eps = np.random.normal(0, eps_sigma, size=(a.shape[0], len(t))) + \\
    1j*np.random.normal(0, eps_sigma, size=(a.shape[0], len(t)))
prob_perturbed = .33
r_support = np.random.choice([True, False], size=len(t), \\
    replace=True, p = [prob_perturbed, 1-prob_perturbed])
r_sigma = 1
r_vals = r_support*(np.random.normal(0, r_sigma, size=(a.shape[0], \\
    len(t))) + 1j*np.random.normal(0, r_sigma, size=(a.shape[0], \\
    len(t))))
x = a[:,np.newaxis]*s + r_vals + eps
```

In this code, a represents the sensor geometry, s is the target signal strength, eps is the noise, and r_vals is the interference term. There are a number of input parameters we use to build each piece of the signal. The notation for these inputs are as follows: n is the number of channels in the data, σ_{eps} is the standard deviation of the noise normal distribution, σ_r is the standard deviation associated with the interference distribution, and p is the probability of perturbation. When we examine another type of interference, we also introduce θ_r as the azimuth angle of the interfering signal and s_r as the interfering signal strength. Finally, α is a tuning parameter we use to scale the sparsity penalty and we use run_t to denote algorithm runtime. After running the code above, we can immediately use the output x as input for the alternating minimization method as the data matrix.

In the example above, we choose a fifty sensor array with a sampling rate of 10000 Hertz and generated ten seconds of data which comes out to 100000 data points. The interference injected is additional random noise of greater strength than the standard noise term. A tenth and hundredth of a second of this data is shown for two channels in Figure 4.1.

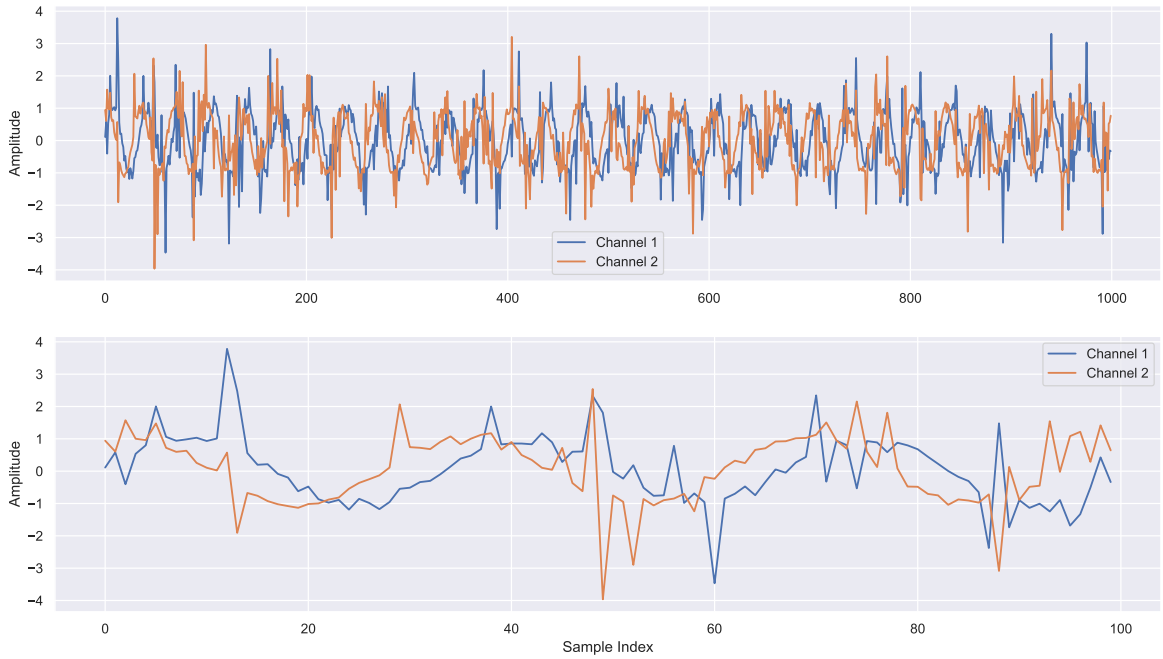

Figure 4.1. 1000 Samples and 100 Samples of Simulated Data With Random Interference: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = 1$, $p = .33$

Here we can see two clear waveforms that appear to have a dominant underlying signal. The first plot demonstrates the jaggedness of the signal throughout a tenth of a second. The second plot focuses on a specific section and more clearly shows the perturbation of the signal. While we can visually see the underlying signal, with the influence of sparse interference it is less simple for a machine to reproduce the target signal from this data. This is why we apply the method developed in Chapter 3.

With the data in hand, the other necessary input for alternating minimization is the λ vector. The Group SLOPE problem described in Chapter 2 has a method for calculating optimal values of λ based on an input of desired False Discovery Rate (Brzyski et al. 2019). The method used relies on having group selections of equal sizes resulting in identical weightings across groups. The generation method for λ is constructed around the chi distribution and the input FDR. While structure of optimization problem (3.7) does vary from standard Group SLOPE, we choose to use the same method for λ generation in the algorithm.

4.1.2 Results

We use a computer with an AMD Ryzen 7 5800X 8-Core processor running at 3.80 GHz, 32.0 GB RAM, and a Windows 11 operating system to generate the simulated data and run the solution algorithm. Solving using the data constructed above takes only 1.8 seconds for 100000 data points while producing perfect results as shown Table 4.1.

	Guessed No Interference	Guessed Interference
No Interference	67052	0
Interference	0	32948

Table 4.1. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .1$, $\sigma_r = 1$, $p = .33$, $\alpha = .2$, $run\_t = 1.73s$

After removing the contaminated samples, we conduct standard DOA estimation on the remaining data and the estimated angle of arrival is $.24999\pi$ compared to an input angle of $.25\pi$. These results appear too good to be true at first pass, but, while they do demonstrate some of the power of the estimator, they more clearly indicate that this data is not a difficult test for the estimator to handle. This could be the result of a number of factors that we will examine and vary in order to conduct some analysis on influential input parameters and the limits of the estimator. It must be noted that throughout the following tests, α is a key parameter that we hand tune to achieve the results. In the algorithm α adjusts the severity of the penalized estimation and so can encourage more or less sparsity in the results when it is changed.

Since we use a normal distribution to generate the interference the first parameter we will vary is the standard deviation, σ_r, of that normal distribution. A larger standard deviation means that the interference will be more likely to take on larger values in the simulated data. In the example above, the σ_r value that governs the distribution for the interference is one and the σ_{eps} for generic noise is a tenth of that. This large gap makes those samples interfered with far different than those with only generic noise present. Reducing the interference by decreasing its standard deviation should make it more difficult for the estimator to identify contaminated samples. Simulated data with a σ_r of .1 is shown in Figure 4.2.

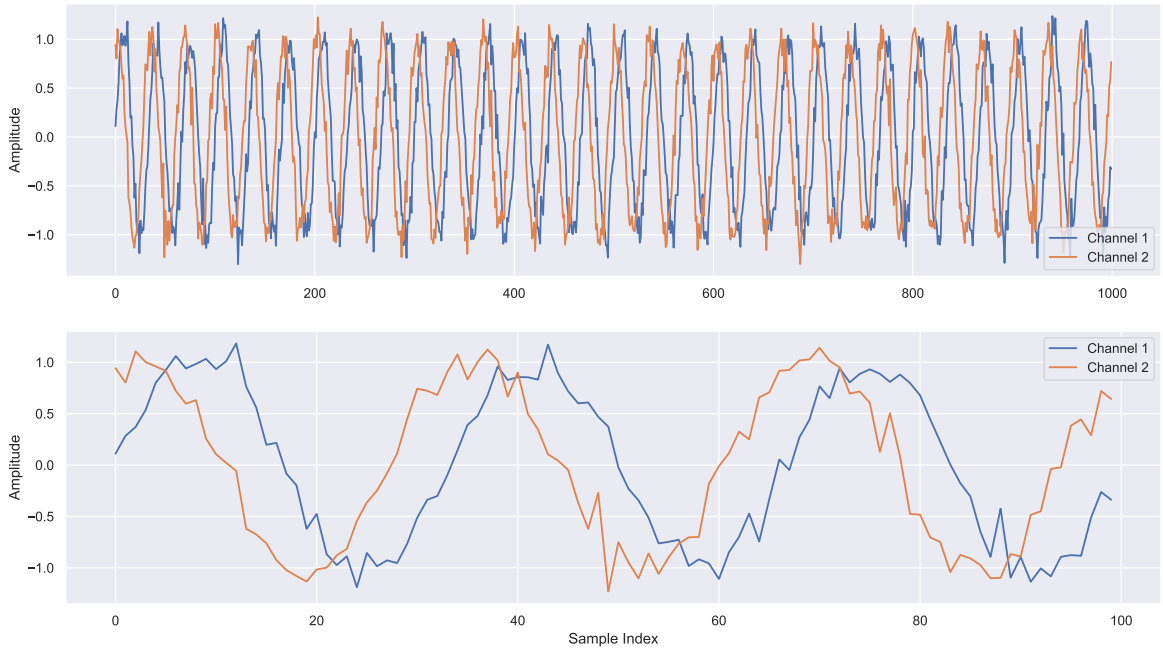

Figure 4.2. 1000 Samples and 100 Samples of Simulated Data With Random Interference: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = .1$, $p = .33$

This plot once again only shows two of the fifty channels, but clearly demonstrates the reduction in clear interference as compared to Figure 4.1. When we run the algorithm on this data, we must adjust the α value slightly to .14 to achieve the results in Table 4.2.

	Guessed No Interference	Guessed Interference
No Interference	66725	327
Interference	359	32589

Table 4.2. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .1$, $\sigma_r = .1$, $p = .33$, $\alpha = .14$, $run\_t = .72s$

With a False Positive Rate (FPR) of .0099 and a False Negative Rate (FNR) of .0054 the estimator is clearly still performing well even at detecting interference that may have little effect. This is not unique to this one instance either. Across 100 iterations of data generation and running the algorithm, we see an average FPR of .0115 and an average FNR of .0051. In addition, if we continue to reduce the standard deviation of the interference and adjust the α accordingly, we see positive results even up until $\sigma = .03$ in Table 4.3.

	Guessed No Interference	Guessed Interference
No Interference	52734	14352
Interference	18788	14126

Table 4.3. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .1$, $\sigma_r = .03$, $p = .33$, $\alpha = .124$, $run\_t = .43s$

Across 100 iterations of this scenario we have an average FPR of .5181 and an average FNR of .029. While this performance is worse than before, the interference is so small at this point that it is reasonable that the estimator struggles to detect it well. A plot of this type of data is shown in Figure 4.3.

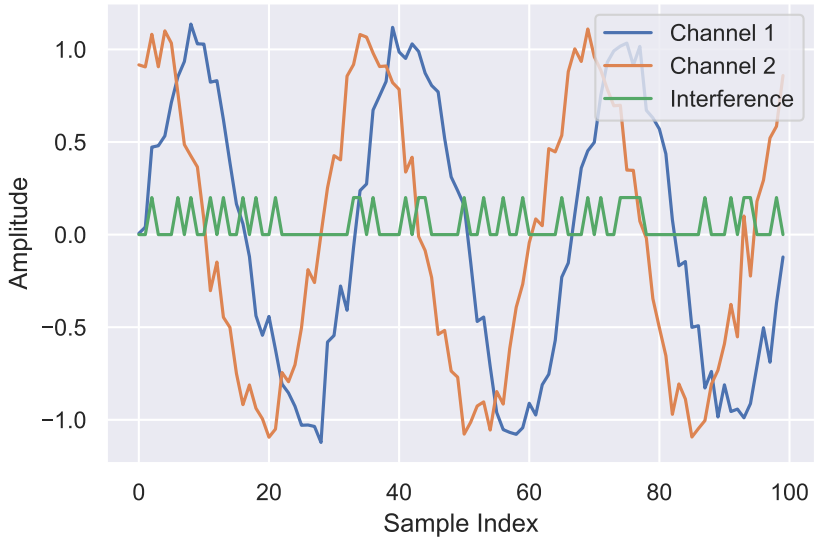

Figure 4.3. 100 Samples of Simulated Data With Random Interference Indicator: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = .03$, $p = .33$

In this specific plot, 34 out of 100 datapoints are interfered with as shown by the green plot. While some of the interference may appear visually clear, much of it is not easily apparent to the eye.

Changing the standard deviation of the noise term accomplishes something similar to changing the interference term in that greater noise will hide the interference more. However, increased noise also cloaks the target signal to a greater degree. If we change the σ_{eps} to three, which is triple that of the interference we see the data in Figure 4.4.

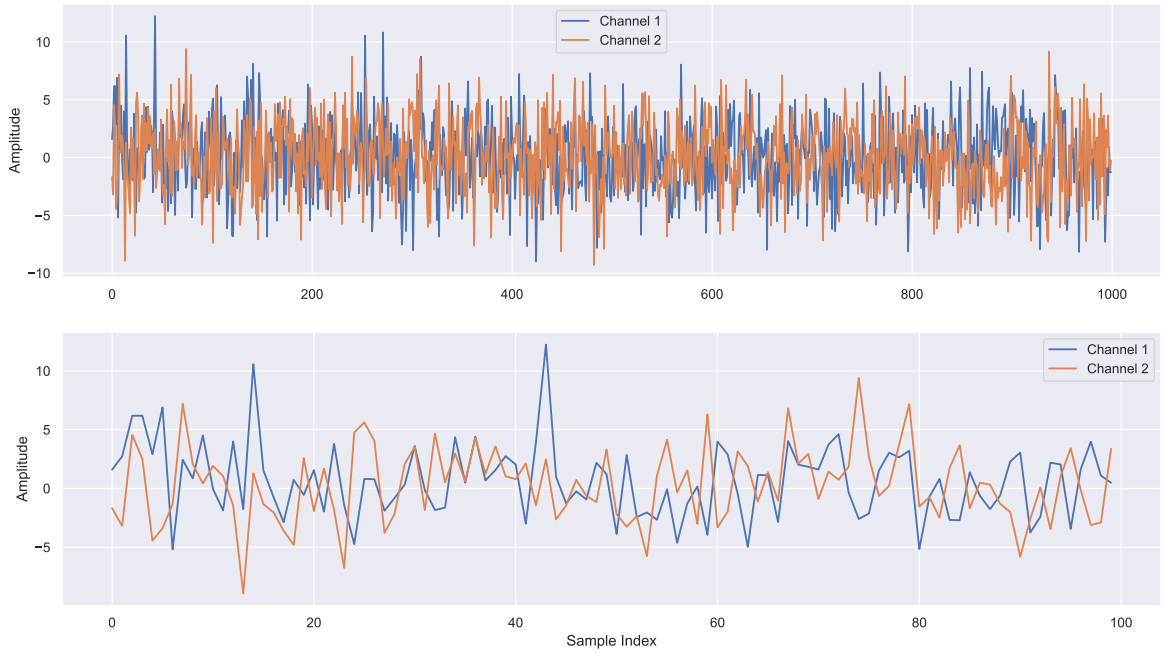

Figure 4.4. 1000 Samples and 100 Samples of Simulated Data With Random Interference: $n = 2$, $\sigma_{eps} = .3$, $\sigma_r = 1$, $p = .33$

This data has so much noise present that not only is it difficult to determine where the interference is, the target signal is also difficult to identify. Adjusting the α appropriately and running the algorithm gives the results in Table 4.4.

	Guessed No Interference	Guessed Interference
No Interference	47098	19906
Interference	14047	18949

Table 4.4. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = 3$, $\sigma_r = 1$, $p = .33$, $\alpha = 3.7$, $run\_t = .73s$

In this case 100 iterations gives a FPR of .511 and a FNR of .229. Once again these results are far from perfect, but do well for how heavily the noise dominates this data. With the removal of interference, when we estimate the direction of arrival for this case we estimate it as $.254\pi$ which is not far from the true DOA of $.25\pi$.

The final parameter we will vary before changing the nature of the interference is the number of samples that we perturb. In order to examine this effect in an adequately challenging

environment, we set the σ values to .5 for both noise and interference. Using this setup, we see that the estimator performs well under varying perturbation probabilities for this type of interference. At a .01 probability of perturbation and an α of .66 we see the results in Table 4.5.

	Guessed No Interference	Guessed Interference
No Interference	98978	36
Interference	42	944

Table 4.5. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .01$, $\alpha = .66$, $run\_t = .86s$

With a .1 probability of perturbation and the same α we obtain Table 4.6.

	Guessed No Interference	Guessed Interference
No Interference	88671	1213
Interference	62	10054

Table 4.6. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .1$, $\alpha = .66$, $run\_t = .75s$

Both of these are highly accurate results, but in the second confusion matrix it is apparent that there is a slightly higher priority on detecting all of the contaminated samples than ensuring there are no false positives. If we adjust α to .685 we see results more similar to the first confusion matrix Table 4.5, illustrated in Table 4.7.

	Guessed No Interference	Guessed Interference
No Interference	89681	203
Interference	224	9892

Table 4.7. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .1$, $\alpha = .685$, $run\_t = .56s$

If we test perturbation probabilities of .5, .9 and .99 and adjust α accordingly to .708, .708, and .685 then we see the results in Tables 4.8, 4.9, and 4.10.

	Guessed No Interference	Guessed Interference
No Interference	49654	393
Interference	412	49541

Table 4.8. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .5$, $\alpha = .708$, $run\_t = .88s$

	Guessed No Interference	Guessed Interference
No Interference	9773	237
Interference	252	89738

Table 4.9. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .9$, $\alpha = .708$, $run\_t = 1.13s$

	Guessed No Interference	Guessed Interference
No Interference	949	73
Interference	75	98903

Table 4.10. Confusion Matrix For 100000 Samples With Random Interference: $n = 50$, $\sigma_{eps} = .5$, $\sigma_r = .5$, $p = .99$, $\alpha = .685$, $run\_t = 1.16s$

If we base the results purely on the number of incorrect guesses, we see that a .5 probability of perturbation is actually the most difficult for the estimator to handle. Normally, FPR and FNR would be metrics used to assess the estimator's performance. However, we heavily influence those metrics with the selection of the α parameter. Regardless, it is apparent that the confusion matrices above all display similar levels of performance demonstrating that, for this type of noise, the estimator can handle most perturbation probabilities. We construct the entire problem around the assumption of sparse interference, but here we handle interference levels that are certainly not sparse and do well.

Added noise is not the only type of interference we include in the tests. The next interference we examine is a separate sparse signal whose strength is determined by a normal distribution. In the initial example we use an azimuth angle, θ_r, of $\frac{\pi}{2}$ and the normal distribution for signal strength is centered at zero with a standard deviation, σ_r, of 1. This results in the data shown in Figure 4.5.

Figure 4.5. 1000 Samples and 100 Samples of Simulated Data With Random Directed Interference: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$

We see a similar waveform here as in previous examples, but the interference appears quite egregious in certain data samples. However, even though some samples are clearly contaminated, it does not seem that all 33% of the interfered samples are readily apparent. Using an α of .135 we have the results displayed in 4.11.

	Guessed No Interference	Guessed Interference
No Interference	65406	1682
Interference	1947	31001

Table 4.11. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = 2.45s$

With the indicated samples removed, we estimate a DOA of $.2496\pi$, whereas with those samples present the estimation was $.285\pi$ due to the pull from the interfering signal. Once again the estimator is performing well. This scenario is quite similar to the first scenario where we achieved perfect results. However, in this case, the results are imperfect because the interference is subject to a normally distributed scaling factor. The undetected interference in the confusion matrix above should be those samples which have minimal interference.

With some initial results in hand, we will test the boundaries of the estimator's performance by varying a few of the parameters used to generate the data.

First, we examine, once again, the standard deviation associated with the distribution of the interference. If we change σ_r for the normal distribution generating the signal than we will change the strength of that signal in the data. Reducing σ_r to .1 with an α of .126 gives the results in Table 4.12.

	Guessed No Interference	Guessed Interference
No Interference	55558	11366
Interference	12009	21066

Table 4.12. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = .1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .126$, $run\_t = 1.03s$

With a FPR of .35 and a FNR of .178 the estimator runs worse than with the larger interference. This is the expected result since the smaller contamination should be harder to detect. Ideally this would mean that as we increase the σ_r value of the interference the estimator would perform better. This does hold true somewhat and we can achieve results for $\sigma_r = 2$ with $\alpha = .185$ such as those shown in Table 4.13.

	Guessed No Interference	Guessed Interference
No Interference	67049	0
Interference	2245	30706

Table 4.13. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 2$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .185$, $run\_t = 7.27s$

However, at this stage the α selection process has become quite unforgiving and volatile so small changes in the α value cause the algorithm to yield meaningless results where either all samples or no samples are identified as contaminated. In addition, as we increase the standard deviation of the interference further to just $\sigma = 3$ the best results we could achieve with a hand tuned α of .85645 are illustrated in Table 4.14.

	Guessed No Interference	Guessed Interference
No Interference	264	66780
Interference	5093	27863

Table 4.14. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 3$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .85645$, $run\_t = .73s$

which are incredibly poor. In these examples, it appears that the interfering signal, though only present in a third of the data, does begin to overpower the target signal and, for this specific case of interference, the estimator can only handle an interference term with σ_r about double the target signal strength of 1.

The next parameter we vary is the perturbation probability. When we do this with the probabilities of .01, .1, .5, .9, and .99 with α of .135 throughout we see the results in Tables 4.15, 4.16, 4.17, 4.18, and 4.19.

	Guessed No Interference	Guessed Interference
No Interference	98908	11
Interference	97	984

Table 4.15. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .01$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = .75s$

	Guessed No Interference	Guessed Interference
No Interference	89649	353
Interference	723	9278

Table 4.16. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .1$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = 1.16s$

	Guessed No Interference	Guessed Interference
No Interference	47933	2284
Interference	2650	47133

Table 4.17. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .5$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = 3.74s$

	Guessed No Interference	Guessed Interference
No Interference	7630	2168
Interference	3624	86578

Table 4.18. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .9$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = 7.23s$

	Guessed No Interference	Guessed Interference
No Interference	0	962
Interference	3187	95851

Table 4.19. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .99$, $\theta_r = \frac{\pi}{2}$, $\alpha = .135$, $run\_t = 7.26s$

The estimator does quite well throughout, but there is a noticeable decline in performance as the interference grows less sparse. When perturbation probability equals .9, the results are still decent, but at .99 the results are meaningless. This indicates that between perturbation probabilities of .9 and .99 performance rapidly decreases, eventually making the estimator useless. However, overall this demonstrates that the estimator is able to handle a good range of sparsity. Taking the .9 perturbation probability case in particular is interesting. In this case we have a target signal in all of the samples and a interference signal in 90% of samples. The interference can be both louder and quieter than the target signal depending on the normal distribution, but even though it is so prevalent in the data, the estimator can still identify with some accuracy most of the samples that have been interfered with while retaining a good number of clean data points. As a result, the DOA estimation goes from $.277\pi$ with all samples kept to $.249\pi$ with contaminated samples removed.

Our final parameter to examine is the azimuth angle of the interfering signal. We will

examine θ_r values of $\frac{3\pi}{4}, \pi$, and $\frac{5\pi}{4}$ with the understanding that the other angles around the full circle would operate similarly based on their difference from the target signal's angle of approach. An azimuth angle of $\frac{3\pi}{4}$ yields a fascinating result. First, the confusion matrix with an α of .12 appears in Rable 4.20.

	Guessed No Interference	Guessed Interference
No Interference	32832	34252
Interference	16152	16764

Table 4.20. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .33$, $\theta_r = \frac{3\pi}{4}$, $\alpha = .12$, $run\_t = .57s$

Our estimator seems incapable of detecting the presence of an interferer in this dataset. While this would appear worrisome, when we conduct DOA estimation using the full dataset we receive an estimated DOA of $.25007\pi$ coming from data shown in 4.6.

Figure 4.6. 1000 Samples and 100 Samples of Simulated Data With Random Directed Interference: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = 1$, $p = .33$, $\theta_r = \frac{3\pi}{4}$

Visually, interference seems to be present in this data, but standard DOA estimation is able to treat it as standard Gaussian noise, even though it is only present in 33% of samples, and

give an excellent estimation. This is not surprising as the interference subspace is orthogonal to the target signal subspace in this specific case.

When we move on to an azimuth angle of π with an α of .135 we see more standard results in Table 4.21.

	Guessed No Interference	Guessed Interference
No Interference	65633	1607
Interference	1898	30862

Table 4.21. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .33$, $\theta_r = \pi$, $\alpha = .135$, $run\_t = 2.19s$

These results are quite similar to those in the initial tests. On the other hand, the DOA estimation using the entire dataset is $.249\pi$ which indicates that interference from π of this type and magnitude has little impact on a target signal coming from $\frac{\pi}{4}$.

Our final azimuth angle to examine is $\frac{5\pi}{4}$ which is directly opposite of the target signal DOA. The results, with an α of .135, again are similar to previous angles as seen in Table 4.22.

	Guessed No Interference	Guessed Interference
No Interference	65782	1550
Interference	1896	30772

Table 4.22. Confusion Matrix For 100000 Samples With Random Directed Interference: $n = 50$, $\sigma_r = 1$, $p = .33$, $\theta_r = \frac{5\pi}{4}$, $\alpha = .135$, $run\_t = 2.36s$

Although promising, the DOA estimation only improves from $.245\pi$ to $.250\pi$ when discarding samples we identify as containing interference. Greater strength or higher perturbation probability may be necessary before this signal impacts DOA estimation.

There is more nuance to this interference type than random noise. The first indication of this is that the above results were more sensitive to α than those with simply noise interference. In addition, the parameters changed had larger impact and interacted with other changes in parameter values. For example, while the estimator was not able to handle large interfering

signals when we changed perturbation probability and azimuth angle, the estimator performs well when the interference strength increased.

The final type of interference we examine is a simple interfering signal with constant strength. This is similar to the previous interference, but instead of a normal distribution to scale the signal we set a constant value. This means that the construction of the interference is similar to the target signal except for the number of samples it is present in. In the first test we use a interfering signal of the same strength as the target signal in 33% of the samples shown in Figure 4.7.

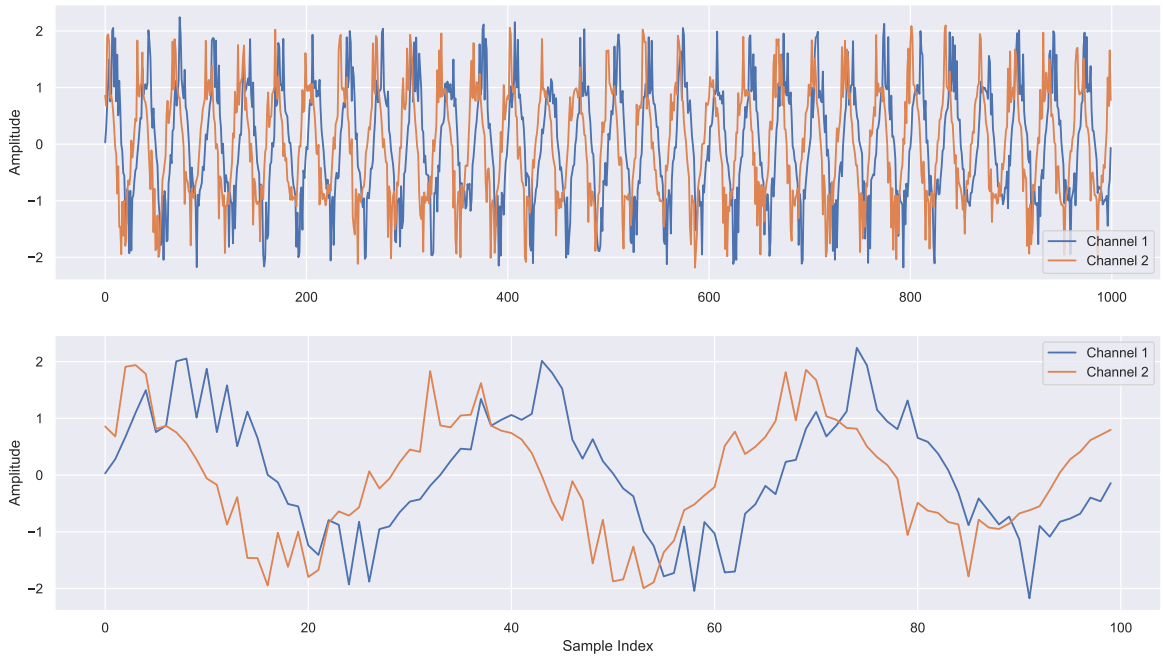

Figure 4.7. 1000 Samples and 100 Samples of Simulated Data With Directed Interference: $n = 2$, $\sigma_{eps} = .1$, $\sigma_r = 1$, $p = .33$

We can see in this plot that the interference is mostly obvious and is much more uniform than in any of the other examples. With an α of .3 we achieve the results in Table 4.23.

	Guessed No Interference	Guessed Interference
No Interference	66884	0
Interference	0	33116

Table 4.23. Confusion Matrix For 100000 Samples With Directed Interference: $n = 50$, $\sigma_{eps} = .1$, $s_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .3$, $run\_t = 4.39s$

This shifts the DOA estimation from $.291\pi$ to $.25\pi$ and, more impressively, perfectly detects all interference. This means, once again, that we must stretch the estimator to see how it handles different challenges in the data.

When we vary the azimuth angle of the interference, we see the same results as with the random strength interference. For the azimuth angle adjustment, the estimator still struggles to identify interference when the interference is perpendicular to the target signal. When we change the probability of perturbation we consistently see perfect results across all perturbation probabilities. However, it should be noted that once the perturbation probability reaches .5 we see the results in Table 4.24.

	Guessed No Interference	Guessed Interference
No Interference	0	49990
Interference	50010	0

Table 4.24. Confusion Matrix For 100000 Samples With Directed Interference: $n = 50$, $\sigma_{eps} = .1$, $s_r = 1$, $p = .5$, $\theta_r = \frac{\pi}{2}$, $\alpha = .3$, $run\_t = 2.63s$

Here we see that we are perfectly guessing incorrectly as the estimator is now identifying the contaminated data as uncontaminated and vice-versa. The next parameter we examine is the strength of the interfering signal.

With perfect detection when the signal strength is the same as the target, it is no surprise that when we increase the interference strength we continue to see perfect results. On the other hand, when we reduce the size of the interference, it does become slightly challenging. With an interference signal strength of .2, which is 20% of the target signal strength, and using an α of .147 we achieve the results in Table 4.25.

	Guessed No Interference	Guessed Interference
No Interference	65398	1548
Interference	1643	31411

Table 4.25. Confusion Matrix For 100000 Samples With Directed Interference: $n = 50$, $\sigma_{eps} = .1$, $s_r = .2$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = 1472$, $run\_t = 1.65s$

These results are still excellent, but eventually reducing interference strength does significantly hinder estimator results. An interference signal strength of .05 with an α of .123 gives the results in Table 4.26.

	Guessed No Interference	Guessed Interference
No Interference	44029	23052
Interference	17797	15122

Table 4.26. Confusion Matrix For 100000 Samples With Directed Interference: $n = 50$, $\sigma_{eps} = .1$, $s_r = .05$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .123$, $run\_t = .57s$

Here we see poor performance, but the estimator does offer slightly promising results even with such a small interfering signal. Increasing the noise present in the data by adjusting the standard deviation of the noise distribution yields similar results as decreasing the interfering signal's strength. As we increase noise, it covers the interference in the data and increases the challenge of detecting contaminated samples. With a standard deviation of 1 for noise and an α of 1.28, we have the results in 4.27.

	Guessed No Interference	Guessed Interference
No Interference	51071	15947
Interference	18835	14147

Table 4.27. Confusion Matrix For 100000 Samples With Directed Interference: $n = 50$, $\sigma_{eps} = 1$, $s_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = 1.28$, $run\_t = .87s$

Here, with greater noise present, the estimator struggles once again. The final parameter we examine the effect of is the size of the sensor array. The base case throughout this analysis has assumed that we have data from fifty sensors in a single array. In the current experiment, when we reduce this value to ten, we still see perfect results. However, with only five sensors and an α of .072, we see the reduced performance in Table 4.28.

	Guessed No Interference	Guessed Interference
No Interference	51188	15979
Interference	13064	19769

Table 4.28. Confusion Matrix For 100000 Samples With Directed Interference: $n = 5$, $\sigma_{eps} = .1$, $s_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .072$, $run\_t = .30s$

An array with only two sensors using an α of .024 gives even worse results shown in Table 4.29.

	Guessed No Interference	Guessed Interference
No Interference	48380	18768
Interference	23334	9518

Table 4.29. Confusion Matrix For 100000 Samples With Directed Interference: $n = 2$, $\sigma_{eps} = .1$, $s_r = 1$, $p = .33$, $\theta_r = \frac{\pi}{2}$, $\alpha = .024$, $run\_t = .12s$

When we reduce the number of channels in the data, we see this quick reduction in the performance. Some of the issue here could be the reduction in the size of the dataset. However, with fifty sensors and only 100 data samples, we still see perfect results, meaning reducing the dataset size has greater impact in the channel dimension than in the time dimension. Finally, when we change the number of sensors for the previous two types of interference, there is reduction in performance, however, it is a much smaller reduction than what is displayed above. Detection of a constant interfering signal seems to particularly be affected by this reduction of sensor array size.

As a final note in the simulated data results section, we must discuss two key pieces of the algorithm runs throughout this analysis. Runtime is incredibly important and a significant

goal of both ourselves and the sponsor. In order to eventually implement this estimator and algorithm in the real world and real time constraints of an autonomous system, the algorithm needs to be able to run as data is collected. Prior to some of work to reduce optimization problem (3.7), the runtime increased rapidly as the dataset size increased. This meant that an eighth of a second of data from two sensors took around an eighth of a second to run, but a full second of data from a fifty sensor array took over two minutes. These issues represented a block to future implementation and encouraged us to examine how to reduce optimization problem (3.7) for more efficient solution techniques. As shown above, we run the algorithm on a sample of ten seconds. Although the runtime is variable throughout each of the runs, its highest value is around seven seconds and frequently is less than a single second as a result of our efforts toward reduction. This brief runtime means that implementation in an autonomous system is not out of reach using the methods presented in this thesis.

The final piece to discuss is α. Across the tests above we see great changes in the α parameter to generate the results. This parameter influences how large of a penalty is applied in the Group SLOPE problem. This means that adjusting too far in either direction consistently would result in identifying all or no samples as contaminated. Careful selection of α allows us to see results where the estimator does have to discern between what is and is not contaminated. Each instance of simulated data above requires different granularity for α to achieve results. In some instances, there is a wide range of values that achieves meaningful results with varying false positive and false negative rates. However, for some more brittle instances of simulated data, a precisely tuned α is necessary to achieve any sort of results. Currently, this α selection is both a positive and negative aspect of the algorithm. On the one hand, this selection allows us to tune the results to encourage higher false positives or higher false negatives and gives the user an input to make that decision with. On the other hand, this α selection means the process is not entirely automated and for truly optimal α selection some pre-existing knowledge of the data is likely necessary. Further examination of the full effects of this parameter on the estimator could be an area for further research.

4.2 Icelandic Data

Our real world data, supplied by Professor Kay Gemba of the NPS Physics Department, comes from an acoustic sensor array placed off the coast of Iceland. The sensors on the array

have a .125 meter horizontal spacing between themselves and are placed at approximately 150 meter depth. With hundreds of sensors on the array and numerous days of data, we have an incredible amount of acoustic data from this one array. Some of the data represents a time where specific testing of the sensor occurred which enabled us, at times, to identify target signals within the acoustic samples. In addition, through both listening to the data as well as examining spectrograms, we were able to single out specific pieces of the provided data which seem to have a prevailing signal throughout the sample while another signal appears sparse in time. This scenario matches the proposed use-case for the methods in this thesis and so we apply these methods using data from three sensors on the array.

4.2.1 Data Processing

The data's original form is in .sio files and, as such, required significant transformation before we could work with it. We wrote a short script that enables us to pull a specific minute of data from the cache of files. When we visualize a minute of the data in a spectrogram we see Figure 4.8.

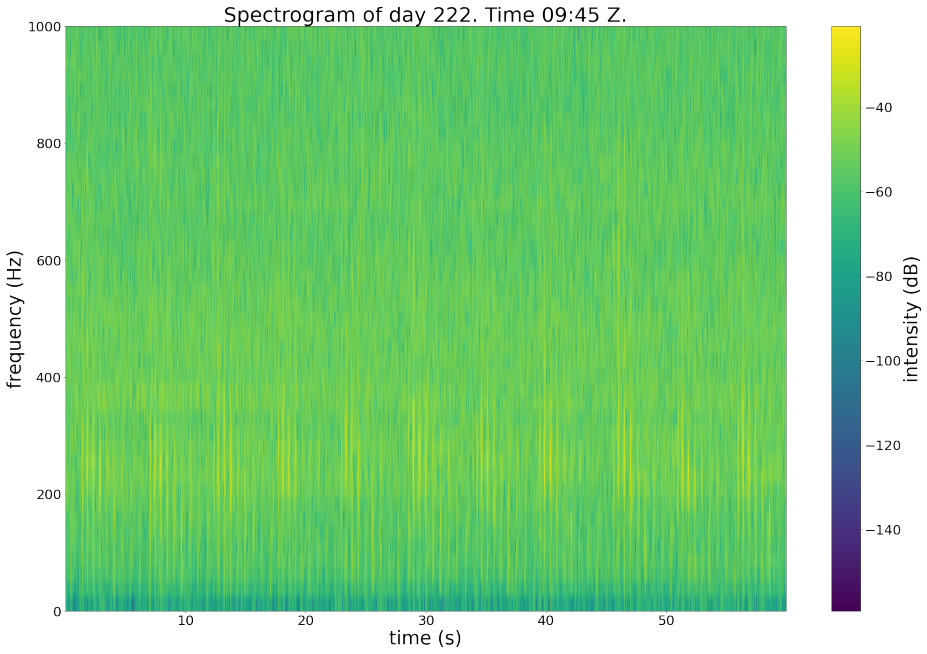

Figure 4.8. 60 Seconds of Raw Acoustic Data in Frequency Range of 0 Hz to 1000 Hz

This visualization does not immediately give a clear application to the problem. However,

it does help us to realize that most of the potentially intriguing pieces of this sound sample occur between frequencies of 200 and 400 Hertz. With this data now in complex waveform we required some final adjustments before inputting into the algorithm. After focusing the signal on specific frequencies and applying a second-order sections filter, we used the Hilbert transform, a linear operator that uses the Fourier transform to create an analytic signal in I/Q format from a real-valued input. Generating the vector λ using the same method described in Subsection 4.1.1 allowed us to then input the data into the alternating minimization algorithm.

4.2.2 Results

When we filter and focus the data the new spectrogram is Figure 4.9.

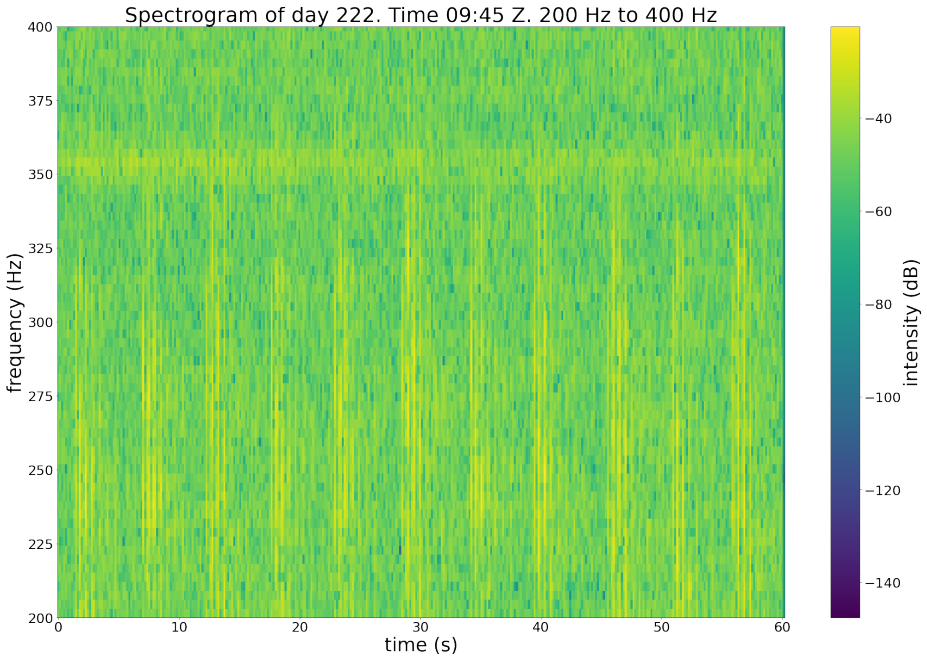

Figure 4.9. 60 Seconds of Filtered Acoustic Data in Frequency Range of 200 Hz to 400 Hz

This gives a much clearer image of the detail present in this specific minute of data. Here we see a signal that appears fairly constant with a frequency just above 350 Hertz. We also see a periodic signal that appears eleven times throughout the minute for a second or two. While this signal does not always directly cover the constant signal at 350 Hertz, the periodic interference does have some interaction with it. In addition, it does add to the

acoustic environment in a way that can not be represented by generic noise that fills the remaining space in the spectrogram. When we apply the estimator to a dataset of this nature, we want it to identify those datapoints with that periodic interference and eliminate them from the data. In this specific example we see the results shown in Figure 4.10.

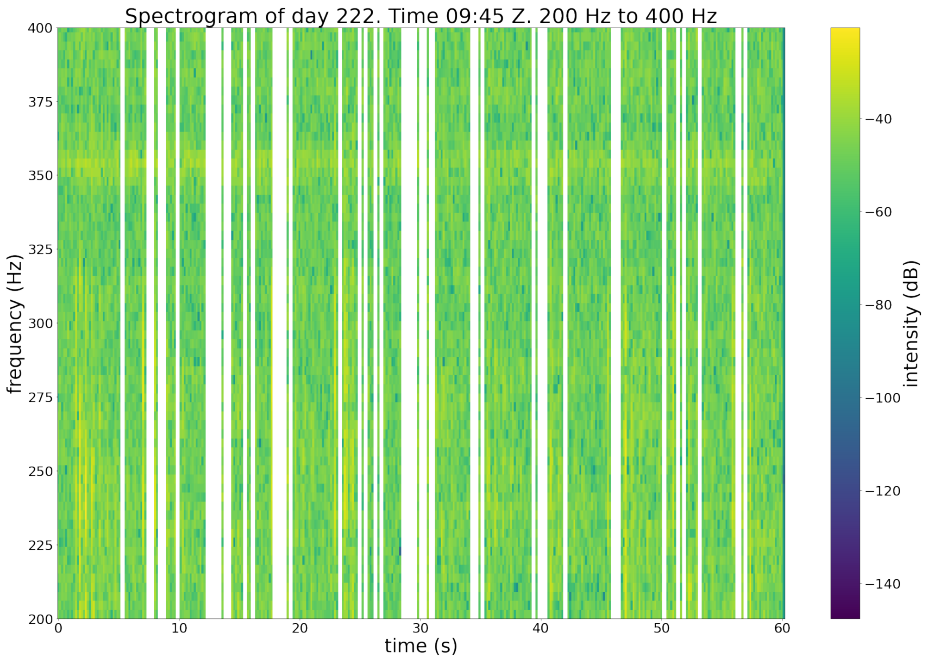

Figure 4.10. 60 Seconds of Filtered Acoustic Data With Detected Contaminated Samples Removed

In this spectrogram we have whited out the data samples that the estimator removes due to contamination. The immediate reaction may be that the white-space seems somewhat random. However, when compared to Figure 4.9, it is clear that a majority of the samples with the periodic interference are now gone. In addition, there are only a few samples that seem to have been removed without that interference present. The significant outlier is the interference that appears in the first few seconds. This likely is a result of the frequency difference between the target signal and this first instance of interference since each of the other interference comes closer to the target signal's frequency.

In general, these results indicate that this method is capable of detecting certain types of interference in real world data. In the work demonstrated above, we successfully automate a process that currently can prove difficult for machines. Due to the accurate results and minimal runtime, we have certainly accomplished that goal. In addition, the results on simulated

data demonstrate the potential breadth of application and robustness of the estimator.

CHAPTER 5:
Conclusions and Future Work

5.1 Conclusions

In this project we create a method for detecting interference in acoustic signals in order to automate the process of identifying corrupted data points. In this effort we examine a number of sub-problems. First, we construct and solve our chosen method for interference detection. Then, we choose to implement a Group SLOPE penalized regression based heavily on existing signal subspace estimation methods shown in Equation (3.2). Transforming and reducing the optimization problem associated with this regression gives us a form of a Group SLOPE problem to solve shown in optimization problem (3.7). To solve this optimization problem we employ alternating minimization for the two variables A and Δ, relying on some existing functionality while also providing some of our own solution techniques.

With an estimator and solution technique, we examine the performance of the estimator. Although we were able to prove the optimality of the solutions to optimization problem (3.7), we need to apply this estimator to data in order to examine results. The first way of doing this is to simulate data to test the estimator, which has numerous advantages for this work. Simulation enables us to generate a wide breadth of types of data for which we know the true underlying signals, allowing us to compare the results with that truth. With this simulated data we demonstrate that the estimator has a wide range of potential applications and successfully identifies the interference in the data. This successful identification and removal from the dataset consistently improves the DOA estimation, demonstrating the usefulness of the estimator. However, we also are able to test some of the limitations and boundaries of the estimator to determine where its results break down and how that might translate to the real world.

In hopes of an even clearer image of the estimator's ability to handle real-world scenarios we apply it to a dataset of acoustic signals from the Icelandic coast. Using spectrograms we give a visual depiction of the data that shows the presence of a constant target signal as well as periodic interference that is sparse in time. Applying the estimator to numerous data

selections of this type all give positive results where the estimator correctly identifies many of the contaminated data samples while ignoring the samples without interference.

The final issue that we consider throughout the entire process is the runtime of the algorithm. An algorithm built for an autonomous vehicle gathering data needs to be able keep up with data collection in real time. The initial solution methods could handle small data matrices in the required time frame, but quickly diverged as the matrix size increased either from greater sampling rate, larger sensor array, or increased complication of another parameter. After a number of attempts at improving the solution algorithm, we eventually develop enough efficiency in the methodology to handle much larger problems in the time necessary. For example, when running the algorithm on a simulated dataset with fifty sensors and 100000 data points representing ten seconds of data we rarely exceed a runtime of a second.

With this reduced runtime we can confidently say that the initial goal is met. We selected a probability model for interference detection, worked to simplify and solve it, tested it on both simulated and real data to demonstrate its use, and improved its runtime issues to encourage future deployment with autonomous vehicles.

5.2 Future Work

The main effort existing for exploration in future work is actual employment of this method in real time with some sort of autonomous vehicle. While there likely is additional tuning and development required for this goal, those issues will become more apparent when working towards implementation in the real world. Since our sponsor for this work is the Office of Naval Research, Science of Autonomy Program, we expect that this work will continue to be pursued and implemented in the real world.

Another improvement that could be worked towards in hopes of improving the estimator for real world application is more robust and engineered real world testing. Currently the estimator works well for the acoustic samples we selected in the Iceland dataset that appear to have a target signal and interference present. However, that is isolated to specific minutes in a specific dataset. Further testing using various datasets would be beneficial. Even more than this, constructing data in the real world with a known target signal and a known interferer would allow true testing of the estimator's performance in reality.

Finally, our work focuses on underwater acoustics as that is the initial problem that is of interest to both ourselves and the United States Navy. However, we represent the data input for the algorithm as a complex waveform which is not a unique medium for underwater acoustic data. Interference exists in acoustic regions outside of the subsurface region as well as in electromagnetic signals. Further research examining the use of the methodology presented in this paper for other problems such as these could prove valuable.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Proofs

For completeness, we include necessary proofs from Bassett (2022).

A.1 Proof of Theorem 1

We estimate the matrix A, the columns of which define the signal subspace, from

$$\hat{A} \in \underset{A\in\mathbb{C}^{c\times d}}{\operatorname{argmin}} \min_{\substack{S\in\mathbb{C}^{d\times T}\\ \Delta\in\mathbb{C}^{c\times T}}} \|AS+\Delta-X\|_F^2 + \|\Delta\|_{\sharp,\lambda}. \tag{A.1}$$

In the inner minimization problem, S can be computed in closed form as $(A^*A)^{-1}A^*(X-\Delta)$, where superscript * denotes conjugate transpose. A minimizer (A.2) becomes

$$\hat{A} \in \underset{A\in\mathbb{C}^{c\times d}}{\operatorname{argmin}} \min_{\Delta\in\mathbb{C}^{c\times T}} \|(I-P_A)(\Delta-X)\|_F^2 + \|\Delta\|_{\sharp,\lambda}, \tag{A.2}$$

where $P_A = A(A^*A)^{-1}A^*$ is the orthogonal projection onto the column space of A.

We proceed to solve Equation (A.2) via alternating minimization over A and Δ because the problem has structure that is amenable to this approach. The matrix group SLOPE regularizer can be shown to be a supremum of convex functions, and hence convex, so for fixed A minimizing over Δ is a convex problem. For fixed Δ, the minimizer in P_A is to take the columns of P_A as the d largest singular vectors of $\Delta - X$. Moreover, we can show that any limit point of the sequence produced by alternating minimization is a critical point of Equation (A.2). Indeed, by Grippo and Sciandrone (2000), if we can cast the problem as the minimization of a continuously differentiable function over closed, nonempty, and convex

sets, the result follows. Consider the relaxation of Equation (A.2)

$$\min_{\substack{P_A\in\mathbb{C}^{c\times c}\\ P_A^2=P_A\\ P_A^*=P_A}} \min_{\substack{c\in\mathbb{R}\\ \Delta\in\mathbb{C}^{c\times T}\\ \|\Delta\|_{\sharp,\lambda}\le c}} \|(I-P_A)(\Delta-X)\|_F^2 + c \tag{A.3}$$

$$= \min_{\substack{P_A\in\mathbb{C}^{c\times c}\\ P_A^2=P_A\\ P_A^*=P_A}} \min_{\substack{c\in\mathbb{R}\\ \Delta\in\mathbb{C}^{c\times T}\\ \|\Delta\|_{\sharp,\lambda}\le c}} \|(\Delta-X)\|_F^2 - \mathrm{Tr}\left[P_A(\Delta-X)(\Delta-X)^*\right] + c \tag{A.4}$$

$$\ge \min_{\substack{P_A\in\mathbb{C}^{c\times c}\\ 0\le P_A\le I\\ \mathrm{Tr}(P_A)=d}} \min_{\substack{c\in\mathbb{R}\\ \Delta\in\mathbb{C}^{c\times T}\\ \|\Delta\|_{\sharp,\lambda}\le c}} \|(\Delta-X)\|_F^2 - \mathrm{Tr}\left[P_A(\Delta-X)(\Delta-X)^*\right] + c. \tag{A.5}$$

The first equality above is directly from the definition of the Frobenius inner product, and the inequality follows because we have induced a relaxation on the set of projection matrices. Note that the relaxation has convex feasible sets, since P_A is restricted to the positive semidefinite cone and $\|\cdot\|_{\lambda,\sharp}$ is a convex function. Additionally, we can show that both problem (A.4) and (A.5) attain the same lower bound given by the von Neumann trace inequality, which implies that this last inequality between problems is actually an equality (Mirsky 1975). Since we have cast the problem as minimizing a continuously differentiable function over closed, convex, and nonempty sets, we conclude that limit points obtained via alternating minimization are critical points as defined in Grippo and Sciandrone (2000).

A.2 Proof of Theorem 2

Recall the following facts

1. Properties of Singular Value Decomposition: For any matrix $M \in \mathbb{C}^{m\times n}$ there exists matrices $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$ and $\Sigma \in \mathbb{R}^{m\times n}$ such that $M = U\Sigma V$, where the columns of both U and V are orthonormal, and Σ is a rectangular diagonal matrix with nonnegative real numbers on its diagonal. The SVD has a *dyadic form* $M = \sum_{i=1}^{\min\{m,n\}} u_i\sigma_i v_i^*$, where $\{u_1, ..., u_m\}$ and $\{v_1, ..., v_n\}$ are the columns of U and V, respectively.
2. Properties of Eigendecomposition: A normal matrix $M \in \mathbb{C}^{n\times n}$ has eigendecomposition $M = \sum_{i=1}^{n} \lambda_i w_i w_i^*$, where $w_1, ...w_n$ form an orthognormal basis of $\mathbb{C}^n$.
3. Properties of Orthogonal Projection Matrices: $P_A^2 = P_A = P_A^*$, and the eigenvalues

of P are either 0 or 1. The number of nonzero eigenvalues equals the rank of $\mathrm{col}(A)$.

4. Properties of Trace and Frobenious Norm: $\|A\|_F^2 = \mathrm{Tr}(A^*A) = \mathrm{Tr}(AA^*)$.

Using Property 4,

$$\min_{P_A} \|(I - P_A)(X - \Delta)\|_F^2 \tag{A.6}$$

$$= \min_{P_A} \mathrm{Tr}\left[(X - \Delta)(I - P_A)(I - P_A)^*(X - \Delta)^*\right] \tag{A.7}$$

$$= \min_{P_A} \mathrm{Tr}\left[(X - \Delta)(I - P_A)(X - \Delta)^*\right] \tag{A.8}$$

$$= \min_{P_A} \mathrm{Tr}\left[(X - \Delta)(X - \Delta)^*\right] - \mathrm{Tr}\left[(X - \Delta)P_A(X - \Delta)^*\right]. \tag{A.9}$$

Therefore it suffices to maximize $\mathrm{Tr}\left[(X - \Delta)P_A(X - \Delta)^*\right]$. Let $r = \max\{m, n\}$ and $\sum_{i=1}^{r} \sigma_i u_i v_i^*$ be the SVD of $X - \Delta$. Similarly, let $\sum_{i=1}^{k} w_i w_i^*$ be the eigendecomposition of P_A. We then have

$$\max_{w_1,\ldots w_k} \mathrm{Tr}\left[\left(\sum_{i=1}^{r} \sigma_i v_i u_i^*\right)\left(\sum_{i=1}^{k} w_i w_i^*\right)\left(\sum_{i=1}^{r} \sigma_i u_i v_i^*\right)\right] \tag{A.10}$$

$$= \max_{w_1,\ldots w_k} \mathrm{Tr}\left[\sum_{j=1}^{k}\left(\sum_{i=1}^{r} \sigma_i v_i u_i^*\right) w_j w_j^* \left(\sum_{i=1}^{r} \sigma_i u_i v_i^*\right)\right] \tag{A.11}$$

$$= \max_{w_1,\ldots w_k} \mathrm{Tr}\left[\sum_{j=1}^{k}\left(\sum_{i=1}^{r} \sigma_i v_i u_i^* w_j\right)\left(\sum_{i=1}^{r} \sigma_i w_j^* u_i v_i^*\right)\right]. \tag{A.12}$$

Applying the cyclic property of trace gives

$$\max_{w_1,\ldots w_k} \sum_{j=1}^{k}\left(\sum_{i=1}^{r} \sigma_i w_j^* u_i v_i^*\right)\left(\sum_{i=1}^{r} \sigma_i v_i u_i^* w_j\right). \tag{A.13}$$

Since $v_i \perp v_j$ for $i \neq j$ and $v_i^* v_i = 1$, we get

$$\max_{w_1,\ldots w_k} \sum_{j=1}^{k}\sum_{i=1}^{r} \sigma_i^2 \left|w_j^* u_i\right|^2. \tag{A.14}$$

Now we need to show that the optimal choice of orthonormal w_j vectors are those which

have the same span as $\{u_1, ..., u_k\}$ (i.e. the u_i with k largest σ_i^2). This is fairly intuitive but can be demonstrated as follows. By the Pythagorean Theorem (recall the u vectors are orthonormal),

$$k = \sum_{j=1}^{k} \|w_j\|_2^2 \geq \sum_{j=1}^{k} \sum_{i=1}^{r} |w_j^* u_i|^2.$$

Moreover, each $|w_j^* u_i|^2 \leq 1$ by the Cauchy-Schwartz inequality. We can thus form a relaxation of Equation (A.14) as follows by letting $\alpha_i = \sum_{j=1}^{k} |w_j^* u_i|^2$.

$$\max_{w_1,...w_k} \sum_{i=1}^{r} \sum_{j=1}^{k} \sigma_i^2 \left| w_j^* u_i \right|^2 \tag{A.15}$$

$$\leq \max_{\alpha_1,...,\alpha_r} \quad \sum_{i=1}^{r} \sigma_i^2 \alpha_i \tag{A.16}$$

$$\text{s.t.} \quad \sum_{i=1}^{r} \alpha_i \leq k \tag{A.17}$$

$$0 \leq \alpha_j \leq 1, \quad i \in \{1, ..., k\}. \tag{A.18}$$

This relaxation is a linear program, and it clearly *obtains* its maximum of $\sum_{i=1}^{k} \sigma_i^2$ (recall that σ_i are nonnegative and sorted in decreasing order). Note that the unrelaxed problem (A.14) also obtains the value $\sum_{i=1}^{k} \sigma_i^2$ by setting $w_i = u_i$ for $i \in \{1, ..., k\}$. Since the maximum value of the relaxed problem is obtained by the original problem, it must also be the maximum value of the original problem. Therefore setting $w_i = u_i$ (for $i \in \{1, ..., k\}$) is a maximizer. Recalling that $\sum_{i=1}^{k} w_i w_i^*$ is the eigendecomposition of P_A, this proves the result. □

A.3 Proof of Theorem 3

We conduct this proof to simplify the objective function in the minimization in Δ for easier computation. Consider the objective function when minimizing over Δ for fixed A,

$$\sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \|(\Delta_1, ..., \Delta_n)\|_{\lambda,\sharp}$$

$$= \sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \sum_{i=1}^{n} \lambda_i \|\Delta_{(i)}\|_2$$

$$= \sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \sum_{i=1}^{n} \lambda_i \sqrt{\|(I - P_A)\Delta_{(i)}\|_2^2 + \|P_A\Delta_{(i)}\|_2^2}.$$

A sufficient condition for minimizer is $P_A\Delta_{(i)} = 0$ for all i. This implies that the previous expression is

$$= \sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \sum_{i=1}^{n} \lambda_i \|(I - P_A)\Delta_{(i)}\|_2,$$

so the original minimization problem becomes

$$\min_{\Delta} \sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \sum_{i=1}^{n} \lambda_i \|(I - P_A)\Delta_{(i)}\|_2.$$

Introducing auxillary variable $c_i = \|(I - P_A)\Delta_i\|_2$, the minimization problem becomes

$$\min_{c,\Delta} \sum_{i=1}^{n} \|(I - P_A)(X_i - \Delta_i)\|_2^2 + \sum_{i=1}^{n} \lambda_i \|(I - P_A)\Delta_{(i)}\|_2$$

$$\text{s.t.} \quad c_i^2 = \|(I - P_A)\Delta_i\|_2^2 \quad \forall i.$$

Expanding the quadratic and neglecting constant terms, we have

$$\min_{c,\Delta} \sum_{i=1}^{n} c_i^2 - 2\Re\left[((I - P_A)X_i)^* ((I - P_A)\Delta_i)\right] + \sum_{i=1}^{n} \lambda_i c_{(i)}$$

$$\text{s.t.} \quad c_i^2 = \|(I - P_A)\Delta_i\|_2^2 \quad \forall i.$$

For fixed c, minimizing over Δ gives

$$\Delta_i \underbrace{=}_{P_A\Delta_i = 0} (I - P_A)\Delta_i = \frac{c_i}{\|(I - P_A)X_i\|_2}(I - P_A)X_i.$$

Substituting this value in gives

$$\min_{c} \quad \sum_{i=1}^{n} c_i^2 - 2c_i \|(I - P_A)X_i\| + \sum_{i=1}^{n} \lambda_i c_{(i)}.$$

Finally, completing the square by adding a constant gives

$$\min_{c} \quad \sum_{i=1}^{n} (c_i - \|(I - P_A)X_i\|)^2 + \sum_{i=1}^{n} \lambda_i c_{(i)}.$$

We have eliminated the channel dimension from the optimization problem. This minimizer is the proximal operator of $\|\cdot\|_{\lambda,\sharp}$ evaluated at

$$(\|(I - P_A)Y_1\|_2, ..., \|(I - P_A)Y_n\|_2).$$

In summary, one evaluation of grpSLOPE prox gives the optimal c values and Δ is recovered via

$$(I - P_A)\Delta_i = \frac{c_i}{\|(I - P_A)X_i\|_2}(I - P_A)X_i.$$

List of References

Bassett R (2022) private communication.

Beck A (2017) *First-Order Methods in Optimization* (Philadelphia, PA, USA: Society for Industrial and Applied Mathematics and the Matematical Optimization Society), 1st edition.

Bogdan M, Berg Evd, Sabatti C, Su W, Candès EJ (2015) SLOPE – Adaptive variable selection via convex optimization. *The Annals of Applied Statistics* 9(3):1103–1140.

Bogdan M, Berg Evd, Su W, Candès E (2013) Statistical estimation and testing via the sorted L1 norm. Cornell University Library.

Brzyski D, Gossmann A, Su W, Bogdan M (2019) Group SLOPE – Adaptive selection of groups of predictors. *Journal of the American Statistical Association* 114(525):419–433.

Chief of Naval Operations (2022) *Chief of naval operations navigation plan 2022*. Technical report, Washington, D.C.

Fischler MA, Bolles RC (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6):381–395.

Grippo L, Sciandrone M (2000) On the convergence of the block nonlinear Gauss - Seidel method under convex constraints. *Operations Research Letters* 26(3):127–136.

Mirsky L (1975) A trace inequality of John von Neumann. *Monatshefte für Mathematik* 79(4):303–306.

Paulraj A, Ottersten B, Roy R, Swindlehurst A, Xu G, Kailath T (1993) 16 Subspace methods for directions-of-arrival estimation. *Handbook of Statistics*, volume 10, 693–739 (Elsevier).

Royset JO, Wets RJ (2021) *An optimization primer* (Cham: Springer), 1st edition.

Zoubir AM, Koivunen V, Chakhchoukh Y, Muma M (2012) Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts. *IEEE Signal Processing Magazine* 29(4):61–80.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California