SPECTRAL UNMIXING AND ANOMALY DETECTION FOR HYPERSPECTRAL IMAGES

by

Ahmed Elrewainy

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
In partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

August 2020

ABSTRACT

Least Angle Regression (LARS) solves the *basis pursuit* optimization problem, as a sparse signal recovery algorithm, for all relevant regularization parameter values simultaneously. However, despite this efficiency of LARS, it has not been applied to the spectral unmixing problem yet as large multichannel data could be very challenging in practice, due to the need for generation and storage of extremely large arrays (~ 10^{10} bytes for a relatively small spectral unmixing problem). In this thesis, we extend the standard LARS algorithm, using Kronecker products, to make it practical, i.e., without the need to construct or process very large arrays, for efficient recovery of sparse signals from large multichannel data.

This thesis also presents a new recursive Kronecker LARS (K-LARS) algorithm based on a homotopy formulation, similar to recursive methods, to update the *basis pursuit* optimization problem. Instead of completely solving a new *basis pursuit* problem that is slightly different from the previous known problem solution, we use it as the starting point to solve the new problem in a more computationally efficient, thereby faster way.

Afterward, we apply our new Kronecker LARS (K-LARS) algorithm and our new Kronecker homotopy algorithm to successfully unmix both synthetic and *AVIRIS* hyperspectral images. We compare our results to ones obtained using an earlier *basis pursuit*-based spectral unmixing algorithm, Generalized Morphological Component Analysis (GMCA), that uses a thresholding-based proximal optimization method. We show that the results are similar, albeit our results were obtained without any trial and error, or arbitrary choices, in specifying the needed regularization parameter.

The spectral unmixing problem requires a sparse endmembers constraint in an appropriate basis known as a dictionary. In this thesis, we explored and compared the use of some standard, e.g., wavelets, Discrete Cosine Transform (DCT), and custom online learned dictionaries to promote sparsity of the unknown endmembers when solving the spectral unmixing problem. We show that because of their large number of vanishing moments, a Coiflet dictionary would be optimum to sparsely represent endmembers, who are mostly smooth functions with a small number of peaks.

Finally, we present a high-spatial-resolution, i.e., using a small number of spectral pixels as background, anomaly detection algorithm that models this background using a *lasso*-penalized maximum likelihood estimation method.

ACKNOWLEDGEMENTS

I would like to thank the many people who have supported me through the preparation of this dissertation. First and foremost, I wish to express my deepest gratitude and sincerest thanks to my advisor, Dr. Sherif Sherif, who generously shared with me his vast knowledge and provided me with advice, valuable feedback, encouragement, and continuous guidance throughout this work.

I would also like to acknowledge the direction and support given by Dr. Pradeepa Yahampath and Dr. Jitendra Paliwal as members on my examining committee.

Finally, and most importantly, I extend my deepest gratitude to my parents, my wife, and my children, who have been an inspiration and a driving factor in all that I have done. Thank you for the endless encouragement, patience, and support. I could not have done this without you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AVIRIS	Airborne Visible Infra-Red Imaging Spectrometer.
AECM	Alternating Expectation-Conditional Maximization.
BIC	Bayesian Information Criterion.
BSS	Blind Source Separation.
DCT	Discrete Cosine Transform.
EM	Expectation-Maximization.
GMM	Gaussian Mixture Model.
GMCA	Generalized Morphological Component Analysis.
HYDICE	Hyperspectral Digital Imagery Collection Experiments.
HSI	Hyperspectral Imaging.
ICA	Independent Component Analysis.
ICE	Iterated Constrained Endmembers.
IEA	Iterative Error Analysis.
K-LARS	Kronecker Least Angle Regression.
LPBIC	*LASSO*-penalized Bayesian Information Criterion.
LARS	Least Angle Regression.
LMM	Linear Mixture Model.
MLE	Maximum Likelihood Estimate.
MOD	Method of Optimal Dictionary.
MV	Minimum volume-based algorithms.
MoFAs	Mixtures of Factor Analysis.
MVN	Multivariate Normal.
NIR	Near-Infrared.
NMF	Nonnegative Matrix Factorization.
OSP	Orthogonal Subspace Projection.
PPI	Pixel Purity Index.
PCA	Principal Component Analysis.
pdf	Probability density function.
ROC	Receiver Operating Curve.
SNR	Signal to Noise Ratio.
S-NMF	Sparse Nonnegative Matrix Factorization.
VCA	Vertex Component Analysis.
VNIR	Visible Near-Infrared.

LIST OF SYMBOLS

Symbol	Name
$\boldsymbol{A}$	Abundance matrix.
$\boldsymbol{B}$	Pure materials signatures library.
$\boldsymbol{C}$	Correlation matrix.
$\boldsymbol{D}$	Updated direction matrix.
$\boldsymbol{E}$	Noise matrix.
$\boldsymbol{G}$	Gram matrix.
$\boldsymbol{H}$	Coefficient matrix.
$\boldsymbol{I}$	Identity matrix.
$\boldsymbol{K}$	Mask matrix.
I	Active set.
I^c	Inactive set.
L	Number of wavelengths.
N	Number of spectral pixels.
M	Number of endmembers.
$\boldsymbol{Q}$	Sign sequence matrix.
$\boldsymbol{R}$	Residual matrix.
$\boldsymbol{S}$	Endmembers spectra matrix.
$\boldsymbol{X}$	Unknown matrix of interest.
$\boldsymbol{Y}$	Spectral pixels matrix.
$\boldsymbol{Z}$	Active set matrix.
$\boldsymbol{a}$	Abundance vector.
a	Abundance value.
$\boldsymbol{b}$	Signal measurement vector.
$\boldsymbol{c}$	Correlation vector.
$\boldsymbol{d}, \partial\boldsymbol{x}$	Updated direction vector.
$\boldsymbol{e}$	Additive noise vector.
e, ξ	Error values.
$\boldsymbol{g}$	Subdifferential function.
$\boldsymbol{h}$	Coefficient vector.
i, j, k, m, n, t	Arbitrary integer choice.
$\boldsymbol{r}$	Residual vector.
$\boldsymbol{s}$	Spectrum of endmember.
$\boldsymbol{x}$	Unknown signal of interest.
$\boldsymbol{y}$	Spectral pixel vector.
$\boldsymbol{z}$	Sign sequence vector.
$\boldsymbol{\Phi}$	Dictionary.

$\boldsymbol{\alpha}$	Coefficients of $\boldsymbol{S}$ in the dictionary $\boldsymbol{\Phi}$.
$\boldsymbol{\beta}$	Coefficients of $\boldsymbol{Y}$ in the dictionary $\boldsymbol{\Phi}$.
D	Detection statistic.
$\boldsymbol{\mu}$	Mean vector
$\boldsymbol{\Sigma}$	Covariance matrix.
$\boldsymbol{\Lambda}$	Loading matrix.
$\boldsymbol{\psi}$	Diagonal matrix.
$\mathcal{L}$	Likelihood value.
λ	Regularization parameter.
σ	Scalar correlation value.
δ	Step size value.
ϵ	Homotopy parameter.
ψ	Wavelet function ψ
φ	Scaling function.
η	Threshold value.

Chapter 1
Introduction

1.1 Motivation

Over the last few decades, hyperspectral imaging (HSI) has emerged as an important remote sensing technique due to its ability to deliver substantial quantities of spectral information, which can be exploited for image analysis techniques. Each material has a spectral signal that serves as a unique and distinctive signature, and HSI detects these signals by measuring a given object's reflectance.

Spectral imaging differs from conventional imaging in that each pixel in the spectral image becomes a vector that represents the material's reflectance or the absorption for that pixel and at contiguous spectral bands. Spectral imaging will produce a data cube—known as a spectral image cube—that is comprised of three dimensions: two spatial dimensions and one spectral dimension. Hyperspectral imaging (HSI) sensors are essentially the same as multispectral sensors; only HSI sensors acquire hundreds of spectral bands [1].

Hyperspectral imaging (HSI) has been used in a variety of civilian and military applications [2, 3]. Some civilian applications include earth observation, remote sensing, crop monitoring, food safety procedures, pharmaceutical process monitoring, air pollution detection, water contamination detection, land mine detection, urban growth monitoring, and meteorological analysis, as well as biomedical, industrial, biometric, and forensic applications. Examples of military applications include target detection and classification, automatic target recognition, and identification of camouflaged targets.

Most digital processing algorithms applied in the hyperspectral imaging field attempt to accomplish one of four tasks: spectral unmixing, anomaly detection, spectral classification, and material identification. Our research focuses on spectral unmixing techniques that can separate hyperspectral data cube's spectral pixels into a collection of its constituent materials or endmembers.

As the hyperspectral cameras typically have low spatial resolution and image acquisition usually takes place from very far distances, e.g., in the case of satellite imaging, the acquired data cubes are considered mixtures of the existing pure materials in the scene. These pure materials are called "endmembers," and they share the spectral pixels with different coefficients, which are called "abundances."

The mixed pixel spectra can be represented mathematically using either linear or nonlinear models. The linear mixture model considers each pixel as a linear superposition of the endmembers in the scene, whereas the nonlinear model considers the physical interactions between the light scattered by the endmembers in the scene. Since the nonlinear model requires more computations than the linear model, most research in spectral unmixing is done using the linear model.

The unmixing of the hyperspectral data cube is a very important process for determining the cube's present endmembers in order to analyze this cube. The unmixing problem is an ill-posed problem. If we assume that all the pixels in the hyperspectral image data cube are generated by mixtures of a small number of "endmembers," the unmixing problem must deliver both; these endmembers as well as their corresponding fractional area coverage, i.e., their abundances. As will be discussed later, this problem had been approached in different ways. However, most approaches determine the endmembers based on some prior knowledge about the scene. These approaches are referred to as supervised unmixing. If there is no prior knowledge about the scene and the present endmembers, this is called unsupervised unmixing.

An overview of approaches to linear spectral unmixing in HSI is given in [4]. These approaches could be categorized into four broad categories; geometrical approaches, statistical approaches, e.g., Bayesian techniques, spatial-spectral contextual approaches, sparse signal recovery approaches that assume the unknown endmembers are sparse in some dictionary, where they could be obtained using a sparse signal recovery algorithm.

The solution for the *basis pursuit* optimization problem can be used as an unsupervised sparse signal recovery-based spectral unmixing algorithm. This solution can be obtained using the proximal method, which is the thresholding method in this problem, as in the Generalized Morphological Component Analysis (GMCA) algorithm [5]. However, the thresholding methods solution has one particular disadvantage in that it requires a trial and error, or arbitrary choices, in specifying the regularization parameter to solve the problem. This disadvantage motivated us to find another method to solve the *basis pursuit* optimization problem.

Least Angle Regression (LARS) could solve the *basis pursuit* optimization problem efficiently [6], by simultaneously obtaining solutions corresponding to all relevant values of the regularization parameter, with a computational complexity comparable to solving a single unconstrained least-squares problem. However, despite this clear computational advantage of LARS, it has not been applied to the spectral unmixing problem before. This is likely the case because the LARS algorithm is not directly suitable for multichannel HSI data, where data vectorization [7] would be necessary before its application. Such HSI data vectorization would result in extremely large matrices (~ 10^{10} elements) and vectors that would be very challenging to store and process using a typical computer. This motivates us to use the LARS algorithm in solving the *basis pursuit* unmixing problem using Kronecker algebraic properties. We call the resulting algorithm "the Kronecker LARS" (K-LARS) algorithm.

Fast implementation of the K-LARS algorithm gave us another motivation to implement it as a dynamic algorithm called "Kronecker K-LARS homotopy." The Kronecker homotopy algorithm uses the last available estimated signal as a starting point instead of solving a new *basis pursuit* problem from scratch or starting from a zero-initial estimate at every iteration. Moreover, Kronecker K-LARS homotopy could also be used to solve the *basis pursuit* unmixing problem.

As the *basis pursuit* problem algorithm assumes that the unknown endmembers are sparse when represented in an appropriate basis, which is known as a dictionary. The choice of the dictionary for the *basis pursuit* problem is very critical for the unmixing process. This gave us the motivation to study dictionary approaches and select a suitable one for the *basis pursuit* unmixing problem. Dictionary approaches can be divided into two main categories: analytic approaches, such as wavelets and curvelets, and learning-based or custom approaches, such as Method of Optimal Dictionary (MOD) and online dictionary-learning algorithms, which will be discussed later.

Anomaly detection algorithms mainly aim to detect or distinguish between a pixel, or group of pixels, and its neighbors. Most anomaly detection algorithms use a large number of spectral pixels in the background window under test. Therefore, these algorithms will not be able to detect anomalies in complex backgrounds. This motivated us to present a novel high spatial resolution anomaly detection algorithm that requires a smaller number of spectral pixels in the window under test. The *LASSO*-penalized Bayesian Information Criterion (LPBIC) is used as a model selection technique to select a model in a model-based clustering approach.

1.2 Research Objectives

The main objectives of this thesis are:

1. To develop a fast, practical unsupervised unmixing algorithm for hyperspectral data cubes.
2. To develop optimal dictionaries used for sparse representation of spectral pixels for hyperspectral cubes unmixing process.
3. To develop a novel anomaly detection algorithm that could be used in complex scenes for hyperspectral data cubes.

1.3 Thesis contributions

1. Extended the Least Angle Regression (LARS) sparsity algorithm to handle multichannel data (K-LARS).

   a) Applied our K-LARS algorithm to achieve computationally practical unsupervised spectral unmixing of hyperspectral imaging data.

b) Developed a fast dynamically updated spectral unmixing algorithm for online hyperspectral imaging data (K-LARS dynamic updating algorithm).

c) Applied our K-LARS dynamic updating algorithm to achieve computationally practical unsupervised spectral unmixing of hyperspectral imaging data.

2. Obtained an optimal dictionary for sparse representation of spectral pixels prior to the sparsity-based unmixing process.

   a) Implementation of the online dictionary learning algorithm for the HSI sparsity-based unmixing problem.

   b) A comparison of the different dictionaries used for the sparse representation of spectral pixels in the HSI unmixing problem.

3. Developed a novel high spatial resolution anomaly detection algorithm that requires a smaller number of spectral pixels in the window under test.

1.3.1 Related publications:

1) **Ahmed Elrewainy** and Sherif S. Sherif, "Unsupervised sparsity-based unmixing of hyperspectral imaging data using an online sparse coding dictionary," *SPIE Remote Sensing and SPIE Security + Defence*, Berlin, Germany, 10 - 13 Sept., 2018.

2) **Ahmed Elrewainy** and Sherif S. Sherif, "Implementation of sparsity-based unsupervised unmixing of hyperspectral imaging data using coiflets," *11th International Conference on Electrical Engineering (ICEENG)*, Military Technical College, Cairo, Egypt, pp. 3-5, April, 2018.

3) Saccon F.A.M., **Ahmed Elrewainy**, Parcey D., Paliwal J., Sherif S.S., "Detection of Fusarium on Wheat using near-infrared hyperspectral imaging," *2016 Photonics North Conference*, Quebec City, Canada, 2016.

4) **Ahmed Elrewainy** and Sherif S. Sherif, "Kronecker least angle regression for unsupervised unmixing of hyperspectral imaging data," *Springer Signal, Image and Video Processing*, 10, (2019).

5) Ishan Wickramasingha, **Ahmed Elrewainy**, Michael Sobhy and Sherif S. Sherif, "Tensor Least Angle Regression for Sparse Representations of Multi-dimensional Signals," *Neural Computation,* Volume 32, Issue 9, September 2020, p.1697-1732.

1.4 Thesis Outline

This thesis is structured as follows:

Chapter 2 presents a background discussion of Hyperspectral Imaging (HSI) and a survey of the previous research done in the fields of HSI unmixing and anomaly detection.

Chapter 3 presents the K-LARS algorithm that adapts the LARS algorithm to multichannel data, such as those focused on spectral pixels.

Chapter 4 presents the Kronecker homotopy algorithm, which dynamically updates the K-LARS algorithm using a homotopy formulation that is very similar to recursive methods, thus enabling the algorithm to be faster than K-LARS.

Chapter 5 details the implementation of a sparsity-based unsupervised unmixing algorithm that is dependent on the two presented algorithms: the K-LARS algorithm and the Kronecker homotopy algorithm.

Chapter 6 discusses the different types of dictionaries used for the sparse representation of pixel spectra in the sparsity-based unmixing algorithm and determines which dictionary provides better performance for the *basis pursuit* unmixing algorithm.

Chapter 7 presents a high-spatial-resolution, i.e., using a small number of spectral pixels as background, anomaly detection algorithm that models this background using a *lasso*-penalized maximum likelihood estimation method.

Finally, Chapter 8 provides our concluding remarks and suggests directions for future work.

The application of the ICA as an unsupervised spectral unmixing technique for classifying sound and Fusarium-damaged wheat kernels is also presented as an appendix.

Chapter 2
Background and Literature Review

This chapter presents a background discussion of hyperspectral imaging (HSI) and a review of previous work related to the unmixing of the hyperspectral imaging data cubes and the anomaly detection algorithms used in this field. The basic idea of hyperspectral imaging is that the ratio of radiation that is reflected, absorbed, or emitted by a given material will vary with wavelength. Thus, hyperspectral imaging sensors measure the material's radiance within each pixel area at a very large number of contiguous spectral wavelength bands.

2.1 Hyperspectral imaging

The term "remote sensing" was originally used in the 1950s by Ms. Evelyn Pruitt in the U.S. Naval Research Office. This term refers to the science of measuring, identifying, and observing an object or a scene without any direct contact with it. Remote sensing aims to obtain information about a scene that is observable but cannot be directly explored. Later, remotely-sensed hyperspectral imaging (also called imaging spectroscopy) came to be defined as the science interested in the measurement, processing, and analysis of spectra acquired from a scene by an airborne or satellite sensor at great distances [1].

Modern hyperspectral sensors acquire scenes via hundreds of narrow and contiguous wavelengths of potentially different electromagnetic bands, which in turn produces a spectrum for every pixel in the scene [8]. For instance, the Airborne Visible

Infra-Red Imaging Spectrometer (*AVIRIS*), which was developed by NASA's Jet Propulsion Laboratory [9], can record the visible and near-infrared spectra (224 spectral bands ranging from 400 nanometers to 2500 nanometer) of reflected light from an area of several kilometers. Another example of the hyperspectral sensors is the 210-band Hyperspectral Digital Imagery Collection Experiments (*HYDICE*) [10], which was developed by the US Naval Research Laboratory, and NASA's HYPERION satellite sensor [11].

The data volume produced by hyperspectral instruments is a data cube that has two spatial dimensions and one spectral dimension. So, the observed spectrum is a vector comprising the values of reflected radiation at each wavelength, and each wavelength corresponds to an image (Figure 2.1). The images in Figure 2.1 show that each material in the scene has a distinct spectrum that can be used as a spectral signature or fingerprint.

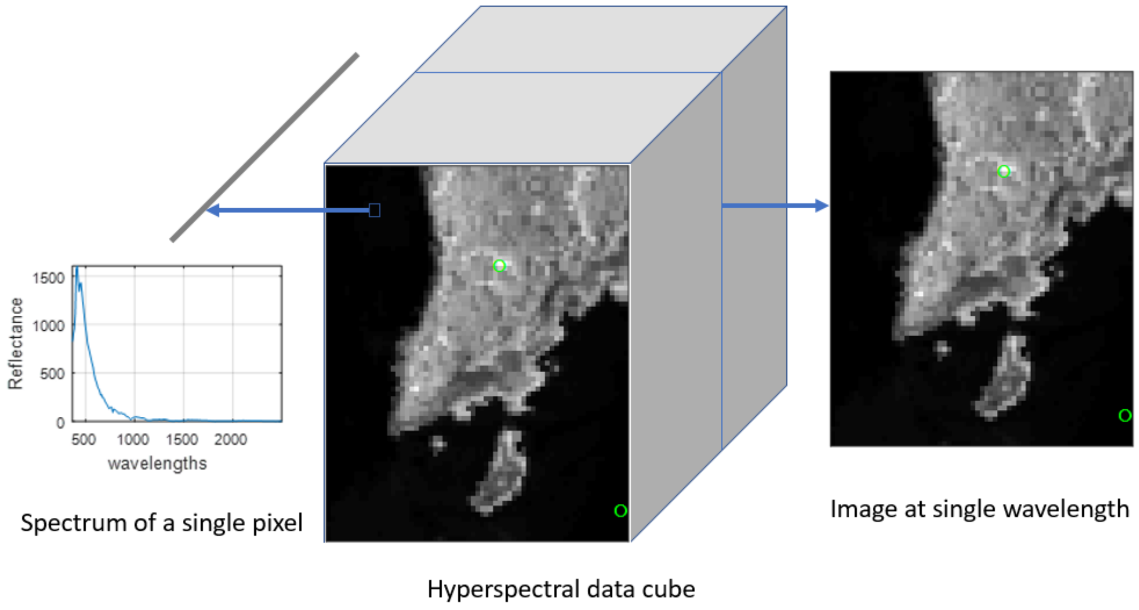


Figure 2.1. Hyperspectral data cube.

There is a variety of processing tasks that can be done via hyperspectral remote sensing. However, most of these algorithms can be categorized according to the following four tasks [12]:

- Spectral unmixing, which involves separating the spectral pixels from a hyperspectral data cube into a collection of constituent spectra.
- Anomaly detection, which uses spectral signatures to distinguish between a single-pixel or a group of pixels, and its background.
- Classification, which involves assigning a label (class) to each pixel of a hyperspectral data cube.
- Material identification, which uses a known library of spectral signatures to identify materials in a hyperspectral data cube.

Particularly, spectral unmixing has been an attractive task since the beginning of hyperspectral image and signal processing [13]. This is because hyperspectral cameras typically have low spatial resolutions and the fact that image acquisition usually takes place from far distances, as in the case of satellite imaging. These features usually produce acquired spectral pixels that are a mixture of many different materials present in the scene.

In this thesis, we focus on the spectral unmixing problem and anomaly detection, which currently are very active areas of research in hyperspectral data cube analysis.

2.2 Spectral unmixing of hyperspectral data

Spectral unmixing refers to any process that separates spectral pixels in a hyperspectral data cube into its individual pure spectral signatures, commonly referred to as *endmembers*, and a set of fractional ratios called *abundances*. In general, there will be a set of endmembers and abundances per pixel. In other words, a pixel's observed spectrum can be analyzed as a combination of the spectra of a group of pure spectral constituents, or endmembers [14]. Thus, a pixel that contains only one constituent material is called a *pure pixel*, while a pixel that contains more than one material is called a *mixed pixel*. In practice, most pixels in a given data cube are mixed.

The endmembers are assumed to represent the pure materials present in the scene, and the abundances in each pixel represent the amount of each endmember in it. For instance, in Figure 2.2, the pixel vector labeled "mixed pixel" contains a mixture of two materials: water and trees. In other words, this mixed pixel includes a small number of pure endmembers that are combined in that pixel.

In order to perform unmixing, a mathematical model must be selected to relate the endmembers and abundances to the spectra pixels. The Linear Mixture Model (LMM) is the most commonly used model [13] in spectral unmixing, and it assumes that pixel spectra can be represented as a linear combination of endmembers that are weighted by their corresponding abundances. This model is described in greater detail in the next section.

Since the linear mixture model ignores the multiple scattering effects, or secondary reflections, in the data acquisition procedure, the measured spectra can be expressed as a linear combination of the spectral signatures of the endmembers that already exist in the mixed pixel (Figure 2.3 (a)). The linear model has some practical advantages, such as ease of implementation and flexibility in different applications [12].

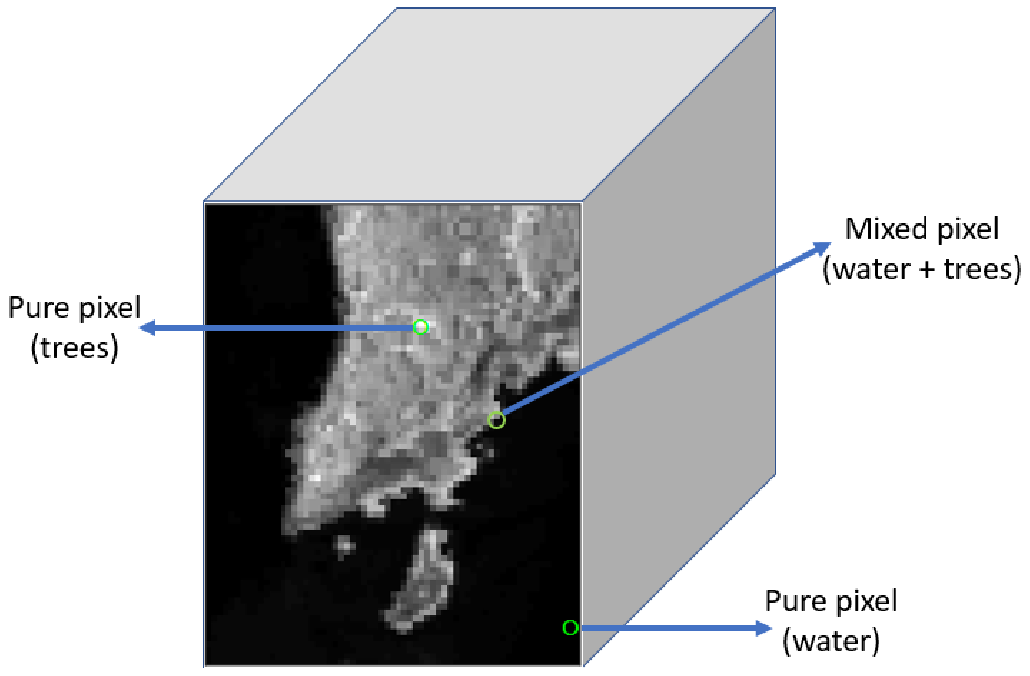


Figure 2.2. Mixed pixels in hyperspectral imaging.

Nonlinear spectral unmixing may be able to characterize mixed spectra better than linear spectral unmixing for certain endmember distributions, such as those in which the endmembers are randomly distributed throughout the scene [15]. In these cases, the mixed spectra acquired by the camera is better described if it is remembered that parts of the source radiation have been scattered multiple times before being acquired at the camera, as in Figure 2.3 (b).

Non-linear models have also been studied recently [4], but these models have a lot of mathematical computations, and their efficiency in extracting endmembers and abundances is still an open area of research [15] that is beyond the scope of this thesis.

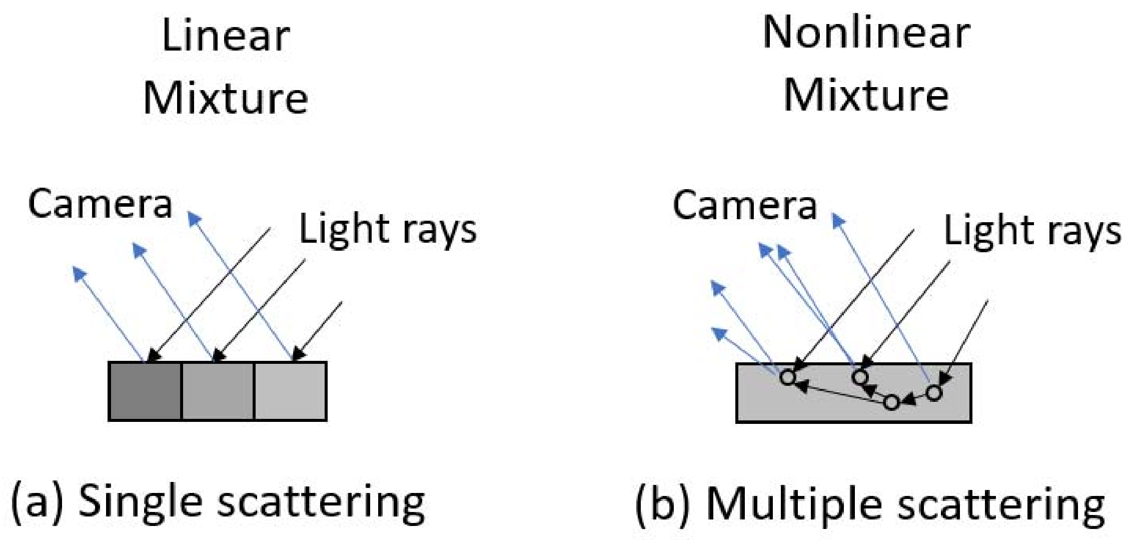


Figure 2.3. Linear and nonlinear mixture models: single and multiple scattering.

2.3 The Linear Mixture Model (LMM)

The Linear Mixture Model (LMM) represents any spectral pixel, $\boldsymbol{y}_i$, in the hyperspectral data cube as a superposition of the existing pure endmembers as follows [4]:

$$\boldsymbol{y}_i = \sum_{j=1}^{M} \boldsymbol{s}_j a_{ji} + \boldsymbol{e}_i \tag{2.1}$$

where $\boldsymbol{s}_j = [s_{j1}, \dots, s_{jL}]^T$is the spectrum of the j^{th} material (endmember) present in the scene, L is the number of spectral bands in the data cube and $a_{ji} \geq 0$ is its corresponding proportion (abundance) in the i^{th} pixel. The added vector, $\boldsymbol{e}_i$ represents an additive perturbation (noise), and M indicates the number of endmembers existing in the scene. From the physical nature of the imaging process, the abundances, a_{ji}, should be subject to two main constraints:

$$\begin{aligned} &Nonnegativity: \quad a_{ji} \geq 0 \\ &Sum - to - one: \quad \sum_{j=1}^{M} a_{ji} = 1 \end{aligned} \tag{2.2}$$

In the matrix form, (2.1) can be written as:

$$\boldsymbol{Y} = \boldsymbol{SA} + \boldsymbol{E} \tag{2.3}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_N]$ is the $L \times N$ matrix representing the observations or measurements, here, in this case, this is the spectral pixels, where N is the number of spectral pixels in the data cube, $\boldsymbol{S} = [\boldsymbol{s}_1, \dots, \boldsymbol{s}_M]$ is the $L \times M$ matrix representing the endmember (sources), and $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_N]$ is the $M \times N$ matrix representing all the abundances of the endmembers of the mixing matrix.

The LMM can be geometrically described as modeling each pixel in a convex hull of the endmembers. This is shown in Figure 2.4, which illustrates an example of a convex hull $\boldsymbol{C}$ has 3 vertices. The points in green denote spectral pixels, whereas the points in red are vertices of the convex hull, which are the endmembers. Estimating the endmembers matrix $\boldsymbol{S}$ is equivalent to identifying the vertices of this convex hull $\boldsymbol{C}$. As will be discussed in the next section, this geometrical point of view is exploited by many unmixing algorithms [4, 13].

Spectral unmixing seeks to estimate the endmember matrix $\boldsymbol{S}$ and the abundance matrix $\boldsymbol{A}$ given the spectral pixels matrix $\boldsymbol{Y}$. It could be considered a Blind Source Separation (BSS) problem [16].

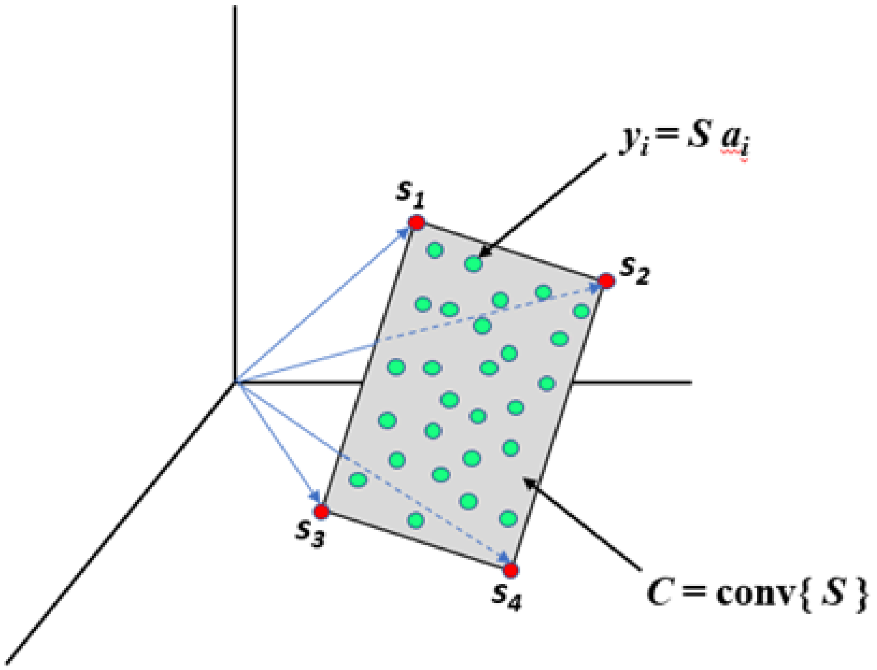


Figure 2.4. Illustration of the convex hull $\boldsymbol{C}$ of the endmembers $\boldsymbol{S}$ for $M = 4$.

2.4 Approaches to linear spectral unmixing

An overview of approaches to linear spectral unmixing in HSI is given in [4]. These approaches could be categorized into four broad categories; geometrical, statistical, spatial-spectral, and sparsity-based. First, geometrical approaches that exploit the fact that acquired spectral pixels must lie in a linear simplex formed by the available endmembers [17]. Second, statistical approaches, e.g., Bayesian techniques that use probabilistic data models, utilize prior distributions to impose model constraints, and estimate parameters of posterior probability distributions [18]. Third, spatial-spectral contextual approaches that exploit both spatial and spectral features available in HSI data through computing spatial correlations between different spectral pixels [19]. Fourth, Sparsity-based approaches, which assume that the unknown endmembers are sparse in some dictionary, where they could be obtained using a sparse signal recovery algorithm. We elect to use a sparsity-based approach in this thesis as it has recently become the approach of choice since it uses powerful but simple sparsity-based signal processing algorithms (as will be shown further). In this section, we will present a brief for each approach.

2.4.1 Geometrical Approaches

Geometrical-based approaches are divided into two main types: Pure-Pixel-based (PP) algorithms and Minimum-Volume-based (MV) algorithms.

2.4.1.1 Pure pixel-based algorithms

Pure-Pixel-based algorithms assume that at least one pure pixel per endmember exists in the hyperspectral data cube. This endmember pure pixel spectrum will be one vertex of the convex hull formed by the given spectral pixels. These algorithms are very efficient from a computational point of view, but they require pure pixels for all endmembers among the spectral pixels, which is very difficult to be achieved in practice.

Some examples of algorithms used for Pure-Pixel-based algorithms include the N-FINDR algorithm [20], the pixel purity index (PPI) algorithm [21], the vertex component analysis (VCA) algorithm [22], and the iterative error analysis (IEA) algorithm [23].

2.4.1.2 Minimum volume-based algorithms (MV)

The MV approach seeks a mixing matrix $\boldsymbol{A}$ that minimizes the volume of the conv$\{\boldsymbol{S}\}$ defined by the endmembers' columns and is subject to the constraint that conv$\{\boldsymbol{S}\}$ contains all the observed spectral pixels vectors. The constraint can be soft or hard. The pure-pixel constraint is no longer enforced, which results in a much harder nonconvex optimization problem. Figure 2.5 illustrates the concept of a minimum-sized convex hull that contains the data. The estimated endmembers spectra $\widehat{\boldsymbol{S}} \equiv [\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \hat{\boldsymbol{s}}_3, \hat{\boldsymbol{s}}_4]$ differs slightly from the true endmembers spectra because there are not enough spectral pixels to define the true convex hull.

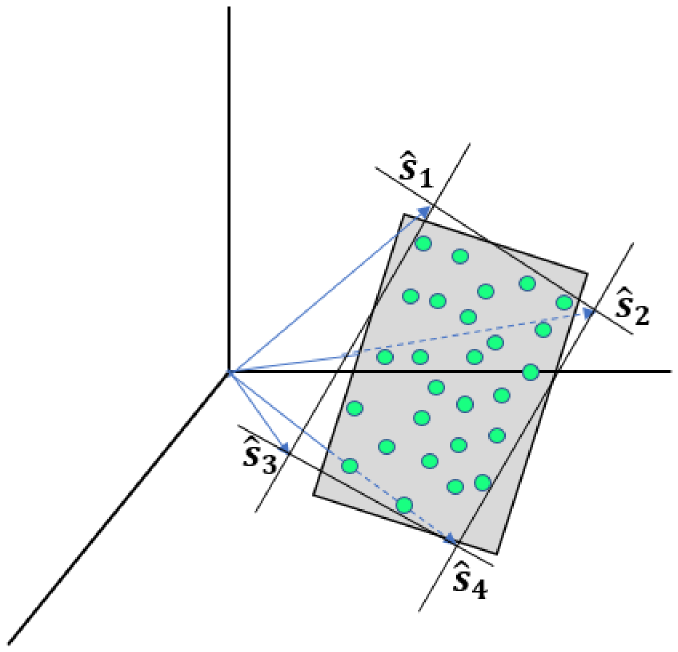


Figure 2.5. Illustration of the concept of a convex hull as the minimum volume containing the data.

The MV approach is used in two widely known methods: the Minimum Volume Constrained Nonnegative Matrix Factorization method (MVC-NMF) [24], and the Iterated Constrained Endmembers method (ICE) [25].

2.4.2 Statistical Approaches

Geometrically-based methods yield poor unmixing results when the spectral pixels are highly mixed because of the minimum number of spectral vectors in the convex hull's facets or vertices. In these cases, statistical methods have proven to be powerful alternative approaches. However, one drawback to statistical methods is that they usually entail higher computational complexity than geometrical-based approaches.

Independent Component Analysis (ICA), has been proposed as a statistical tool for blindly unmixing hyperspectral data [26]. ICA finds the endmembers' signatures by multiplying the spectral pixels data set with an unmixing matrix, which in turn minimizes the mutual information among the spectral bands. Our implementation of ICA as a statistical spectral unmixing algorithm for detecting *Fusarium* infection on wheat is described in the appendix at the end of the thesis.

Bayesian approaches can model statistical variability, regularize solutions, and impose priors that have some constraints on the solutions. The regularization of solutions is a requirement for solving ill-posed problems. In a Bayesian, the posterior probability density of the quantities to be estimated is the core of the model. Assume the unknown endmembers $\boldsymbol{S}$ and the abundance matrix $\boldsymbol{A}$ are independent in prior, the Bayesian model gives the joint posterior of $\boldsymbol{S}$ and $\boldsymbol{A}$ as [4]:

$$p_{\boldsymbol{S},\boldsymbol{A}|\boldsymbol{Y}}(\boldsymbol{S},\boldsymbol{A}|\boldsymbol{Y}) = \frac{p_{\boldsymbol{Y}|\boldsymbol{S},\boldsymbol{A}}(\boldsymbol{Y}|\boldsymbol{S},\boldsymbol{A})\, p_{\boldsymbol{S}}(\boldsymbol{S})\, p_{\boldsymbol{A}}(\boldsymbol{A})}{p_{\boldsymbol{Y}}(\boldsymbol{Y})} \tag{2.4}$$

where p stands for the probability density function (pdf), $p_{\boldsymbol{S},\boldsymbol{A}|\boldsymbol{Y}}(\boldsymbol{S},\boldsymbol{A}|\boldsymbol{Y})$, is the posterior function, $p_{\boldsymbol{Y}|\boldsymbol{S},\boldsymbol{A}}(\boldsymbol{Y}|\boldsymbol{S},\boldsymbol{A})$ is the likelihood function depending on the observation model, and the prior distributions, $p_{\boldsymbol{S}}(\boldsymbol{S})$ and $p_{\boldsymbol{A}}(\boldsymbol{A})$, are the prior knowledge of these unknown parameters.

2.4.3 Spatial-Spectral Contextual Information approaches

Most spectral unmixing approaches treat every pixel's spectrum independently from its neighbors. So, the spectral characteristics of the hyperspectral data cube alone will be taken into consideration when solving the spectral unmixing problem. Some recent studies have found that the spatial correlation between neighboring pixels provides useful information for the spectral unmixing process. The joint analysis of spatial and spectral characteristics leads to improvements in the spectral unmixing results.

There have been three primary directions in research on spatial-spectral unmixing approaches [19]. The first direction focuses on endmember extraction. The second direction focuses on the selection of endmember combinations to decompose each pixel in the whole data cube. The third direction focuses on abundance estimation, which estimates the fractional abundances of endmembers present in every mixed pixel in the data cube.

2.4.4 Sparsity-based approaches

They are called sparse signal recovery approaches. They are one of the most recently developed approaches to solving the unmixing problem. These approaches formulate the spectral unmixing problem as a sparse linear regression problem. They are used in a semi-supervised fashion or unsupervised (blind) fashion. This approach will be described in detail in the next section, as it is our main interest in this thesis due to these reasons:

1) Geometrical approaches yield poor unmixing results when the spectral pixels are highly mixed as most actual hyperspectral data cubes.
2) Statistical (Bayesian) and spatial-spectral approaches usually involve impractically high computational complexities.
3) Independent Component Analysis (ICA) is a common statistical unmixing approach. It has many limitations, e.g., assumes mutually independent endmembers, and a square mixing matrix $\boldsymbol{A}$.
4) Sparsity-based spectral unmixing approach has recently become the approach of choice because
   a) No mutual independence of endmembers is required.
   b) $\boldsymbol{A}$ is not restricted to being square.
   c) Uses the powerful but simple sparsity-based signal processing approach.

2.5 Sparsity-based spectral unmixing

Sparse signal processing has established itself as a powerful approach to representing and processing different signals [27]. Depending on the nature of the signal, one could find a suitable domain where most of its coefficients are zero. A sparse signal representation is important for signal compression, as it requires less memory for its storage. Furthermore, sparse signal representations can also result in simpler signal processing algorithms; for example, signal denoising via simple thresholding operations in a domain where the signal is assumed to be sparse [28]. In addition, the computational costs of processing a sparsely represented signal are typically less than the costs of processing its dense counterpart.

Sparse signal recovery-based spectral unmixing techniques could be implemented in a semi-supervised fashion. This is done by assuming that each acquired spectral pixel is a linear combination of a priori known endmembers. These endmembers, i.e., pure spectral signatures, would have been obtained in a lab beforehand using a spectroradiometer. Semisupervised spectral unmixing therefore aims to find the optimal subset of endmembers from a very large library that can best fit each spectral pixel [29, 30].
Also, sparse signal recovery-based spectral unmixing could be solved using an unsupervised approach, where a priori knowledge of the endmembers would not be required. This unsupervised spectral unmixing could be viewed as a blind source separation (BSS) problem [13], where the unknown sources are assumed to be sparse signals. Two common mathematical formulations of this sparse BSS problem are 1) Sparse Non-negative Matrix Factorization (S-NMF) [31-35], and 2) Generalized Morphological Component Analysis (GMCA) [36-38]. Both formulations are typically solved using a coordinate descent optimization approach [39], where one alternates between estimating endmembers while keeping their abundances constant (endmembers estimation step), and estimating abundances while keeping endmembers constant (abundances estimation step). Both steps are repeated till conversion to the sought optimal values of both endmembers and their abundances. The GMCA unmixing algorithm depends mainly on the *basis pursuit* optimization problem described in the next subsections.

2.5.1 The basis pursuit problem

Sparse signal restoration assumes that the unknown signal is sparse in an appropriate domain. Therefore, signal sparsity can be used as prior information to obtain an estimate of the signal, even if the number of available measurements is smaller than the dimension of the unknown signal. This is the basic idea of *compressed sensing* [40]. Consider the following linear observation model:

$$\boldsymbol{b} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{e} \tag{2.5}$$

where $\boldsymbol{b}$ denotes a signal measurement, $\boldsymbol{H}$ is an $\mathrm{L} \times \mathrm{nn}$ coefficient matrix, $\boldsymbol{x}$ denotes the unknown signal of interest, and $\boldsymbol{e}$ denotes added noise in the system. Sparse signal recovery can be represented as finding a minimum-cardinality solution to a constrained optimization problem. In this case, the constraints are that the solution must satisfy $\|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2 < e$; that is, the actual measurement $\boldsymbol{b}$ is $\boldsymbol{e}$-close to $\boldsymbol{H}\boldsymbol{x}$ in l_2-norm (Euclidean norm). The objective function is the cardinality of $\boldsymbol{x}$, i.e., the number of nonzeros, which is often denoted $\|\boldsymbol{x}\|_0$ and called the l_0-norm of $\boldsymbol{x}$. Mathematically, the optimization problems can be written as:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad s.t. \quad \|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2 \leq e \tag{2.6}$$

This l_0-norm problem is computationally hard, or NP-hard, due to its nonconvex combinatorial nature; therefore, we must resort to approximations. Two main approximation approaches are typically used in sparse recovery. The first approach is to address the original hard combinatorial problem via approximation methods, such as the greedy algorithms (e.g., orthogonal matching pursuit (OMP) [41]), while the second approach is to replace the intractable problem with its convex relaxation, which is easier to solve; that is, by replacing $||\boldsymbol{x}||_0$ with $||\boldsymbol{x}||_1$. In other words, one can either solve the exact problem approximately or solve an approximate problem exactly. Here, we primarily discuss the second approach, convex relaxations. The corresponding *l₁*-norm relaxation of the *l₀*-norm objective can be written as:

$$\min_{\boldsymbol{x}} \ ||\boldsymbol{x}||_1 \quad s.t. \quad ||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}||_2 \leq e \tag{2.2.7}$$

Note that, equivalently, we can impose the constraint on the square of the *l₂*-norm rather than on the *l₂*-norm itself, i.e. $||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}||_2^2 \leq \xi$, where $\xi = e^2$. Then, by using an appropriate Lagrange multiplier $\lambda(e)$, denoted simply as λ below, we can also rewrite the above problem as an unconstrained minimization:

$$\min_{\boldsymbol{x}} \frac{1}{2} ||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}||_2^2 + \lambda ||\boldsymbol{x}||_{\boldsymbol{1}} \tag{2.8}$$

where λ is a regularization parameter that quantifies the tradeoff between the squared residual term, $||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}||_2^2$, and the sparsity level term, $||\boldsymbol{x}||_1$.

For an appropriate parameter $t(e)$, denoted simply as t, the same problem can be rewritten as follows:

$$\min_{\boldsymbol{x}} \frac{1}{2} ||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}||_2^2 \quad s.t. \quad ||\boldsymbol{x}||_1 < t \tag{2.9}$$

The above *l₁*-norm regularized problem—especially in its two latter forms, (2.8) and (2.9)—is commonly known as the *lasso* or the *basis pursuit* [42] in statistical and signal processing literature, respectively. The *basis pursuit* is a convex optimization problem whose exact solution can be obtained using convex optimization methods, such as linear programming [43] or proximal methods [44].

2.5.2 The GMCA spectral unmixing algorithm

The GMCA algorithm was used in [5] to unmix hyperspectral data cubes blindly. The authors called it *hyp*GMCA algorithm. It uses a coordinate descent optimization approach to estimate the endmembers and the abundances iteratively and alternately, as will be discussed in detail in chapter 5.

For the coordinate descent optimization approach used in solving the spectral unmixing problem; in its endmember estimation step, GMCA solves an l_1-norm minimization problem, i.e., a *basis pursuit* problem [42] using proximal methods which is thresholding in this case to obtain sparse estimates of the endmembers from the given multichannel hyperspectral data. Any *basis pursuit* problem requires specifying a data-dependent regularization parameter to specify the relative importance of sparsity levels of unknown variables and the error in their estimates. Therefore, computationally inefficient trial-and-error is typically needed to find a suitable value for this parameter. GMCA, however, exploits the fact that it solves the *basis pursuit* problem only as a step in the larger coordinate descent optimization procedure. In its first iteration of coordinate descent, GMCA solves the required *basis pursuit* problem with a large initial value of this regularization parameter (obtained using trial and error), corresponding to endmembers with a high level of sparsity. In its following iterations of coordinate descent, GMCA solves the required *basis pursuit* problem with a lower value of this regularization parameter, thereby increasing details of the estimated endmembers. However, trial and error is a computationally inefficient manner to choose the initial value of the regularization parameter, and the arbitrary manner in choosing the schedule used to lower its value, are not optimal and could affect convergence and spectral unmixing results obtained using GMCA.

2.5.3 Unsupervised NMF sparse spectral unmixing

Sparsity-based spectral unmixing can be solved blindly (unsupervised) using the nonnegative matrix factorization (NMF) algorithm [45] by adding the sparsity constraints of the sources and abundances to the unmixing model in order to increase the efficiency of the unmixing process. In [31], a gradient-based sparse NMF algorithm (NMF-SMC) was presented to solve the hyperspectral unmixing problem. The authors proposed an S-measure of sparseness using the pixel spectra's higher-order norms. In [32], the authors extended the NMF method by incorporating the $l_{1/2}$ sparsity constraint ($l_{1/2}$-NMF) and proposed an iterative estimation algorithm for $l_{1/2}$-NMF that provides sparser and more accurate results than those obtained using other norms. In [33], a new constrained sparse (CS) CSNMF was proposed, wherein a second sparsity term was introduced to restrict the abundances. In [34, 35], a dual graph regularized sparse NMF (DGNMF) and a hypergraph-constrained NMF were proposed. These methods incorporated the spatial-spectral joint constraints, which in turn preserved consistency among the similar pixels in both the spatial and spectral spaces. NMF is out of scope here in this thesis.

2.5.4 Semisupervised basis pursuit spectral unmixing

The *basis pursuit* can also be used in the spectral unmixing problem in a semi-supervised fashion [30] in which the mixed pixels are expressed as the linear combinations of some pure spectral signatures obtained from a large spectral library that is known a priori.

All the used libraries are considered underdetermined systems, i.e., $L < NN$ where NN is the number of spectral signatures in the library. With this in mind, let $\boldsymbol{x} \in \mathbb{R}^{NN}$ denote the fractional abundance vector of the library $\boldsymbol{B}$ materials. We say that $\boldsymbol{x}$ is a k-sparse vector if it has, at most, k nonzero elements. With these definitions in place, we can now write our sparse recovery problem as:

$$\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \tag{2.10}$$

The task here is to estimate $\boldsymbol{x}$, which is a sparse vector that shows which pure signatures in library $\boldsymbol{B}$ are parts of the spectral pixel $\boldsymbol{y}$.

2.6 Anomaly detection

Target detection, or anomaly detection, is considered one of the most important tasks in the field of hyperspectral imaging processing research. Anomaly detection is considered a binary classification problem because it uses spectral signatures to distinguish between a single pixel, or group of pixels, and its background [46]. The anomaly detection problem is considered an unsupervised target detection problem because no prior knowledge about the pixel target or the background is available. This method focuses on differentiating unusual pixels from the background.

Most anomaly detection algorithms mainly extract statistical parameters from the background pixels spectra and use the difference between this background and the anomalies to differentiate between them [47]. Anomalies refer to unusual pixels with spectral characteristics that are significantly different from the homogenous background pixels. The background is defined as the nontarget pixels that are dominant in the scene

[48]. So, characterizing the background is a challenging key factor in anomaly detection as a more accurate model will result in better detection performance. Different background models and how that model is estimated lead to different anomaly detection algorithms.

In recent decades, a large number of anomaly detection algorithms have been proposed that employ different methods of extracting the required parameters from the background. The RX detector proposed by Reed and Yu [49] is considered the benchmark for anomaly detection algorithms. This algorithm assumes that the background is homogeneous and has a multivariate normal (MVN) model [50] that depends on a PDF function using the statistical parameters (the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ of the background). This algorithm use all the background pixels in a local region or in the entire cube to get the background statistical parameters. These parameters, such as the mean and the covariance, are subject to the presence of other anomalies in the background region [48], [52], and this can affect the differences between the anomalous pixels and the background. Furthermore, obtaining the inverse of the covariance matrix can sometimes cause unstable approximations. The subspace models [51] that depend on projecting the background in a low dimension subspace using a projection matrix $\boldsymbol{P}$ have led to several detection algorithms such as the orthogonal subspace projection (OSP) [52].

A new approach for detecting anomalies is based on a model-based clustering approach that divides the background into a few clusters. Anomaly detection based on the background mixture models performs a cluster-conditional test that is applied to pixel under test to compute a cluster-conditional Mahalanobis distance that results in selecting the *j*th mixture component as the closest one among those represented in the annulus background components.

To overcome the drawbacks of complex background modeling-based anomaly detectors, we propose a high spatial resolution anomaly detection to be used in a complex background. The *LASSO*-penalized Bayesian Information Criterion (LPBIC) is used as a model selection technique to select a model in a model-based clustering approach. This will be discussed in detail in Chapter 7.

2.7 Chapter summary

In this chapter, the concept of hyperspectral imaging and its importance was discussed. The linear mixture model was shown to be a mathematical model capable of representing the spectral pixels. Four major spectral unmixing approaches were also discussed, namely, geometrical, statistical, spatial-spectral, and sparsity-based approaches. Furthermore, sparsity-based spectral unmixing approaches that use the *basis pursuit* optimization problem in both an unsupervised and semisupervised fashion were also discussed. Finally, the sparse NMF was introduced as a viable unsupervised sparsity-based unmixing algorithm, and an overview of the selected anomaly detection algorithm was provided.

Chapter 3
Kronecker Least Angle Regression for Multichannel signals

Least Angle Regression (LARS) [53] could solve the *basis pursuit* minimization problem efficiently [6], by simultaneously obtaining solutions corresponding to all relevant values of the regularization parameter λ, with a computational complexity comparable to solving a single unconstrained least-squares problem. However, despite this clear computational advantage of LARS, it has not been applied to the spectral unmixing problem before. This is likely the case because the LARS algorithm is not directly suitable for multichannel HSI data, where data vectorization [7] would be necessary before its application. Such HSI data vectorization would result in extremely large matrices ($\sim 10^{10}$ elements) and vectors that would be very challenging to store and process using a typical computer.

We exploit the properties of Kronecker products [54] to extend the LARS algorithm to handle large multichannel data without the need to construct or process very large arrays. This extension of the LARS algorithm would make the application of LARS to HSI spectral unmixing practical. Also, this extension of LARS would be of general importance, where it could be used for practical and efficient sparse signal recovery from large multichannel data. We refer to our extended LARS method as the Kronecker LARS (K-LARS) algorithm.

3.1 Introduction

Consider the sparse source recovery problem described in the previous chapter:

$$\widehat{\boldsymbol{x}} = \min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0 \tag{3.1}$$

Since Orthogonal Matching Pursuit (OMP) solves this optimization problem by taking the largest possible steps in the direction of the least-squares solution, it is considered overly greedy. Least-angle regression (LAR) was developed as a less greedy version of OMP [55]. LAR can also be simply modified to solve the *basis pursuit* problem, which is the relaxed version of (3.1),

$$\widehat{\boldsymbol{x}} = \min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \tag{3.2}$$

for all values of λ, in a more efficient way than standard convex optimization methods. Therefore, the LAR acronym was extended by adding the letter S to it (LARS) in order to emphasize its relationship to *basis pursuit*, which is also known among statisticians as the *lasso*.

Since the development of LARS, different extensions to it have been presented in [56-60]. In this chapter, we extend LARS to handle multichannel signals practically. Therefore, in (3.2) the measurement vector $\boldsymbol{b}$, which is similar to the pixel spectrum $\boldsymbol{y}$, is replaced by a matrix $\boldsymbol{Y}$ whose columns represent multichannel measurements, or the given spectral pixels and $\boldsymbol{x}$ is replaced by a matrix $\boldsymbol{X}$ whose columns represent different components of the unknown signal, or the required endmembers. Also, the coefficient matrix $\boldsymbol{H}$ will become the abundance matrix $\boldsymbol{A}$ that results in a new *basis pursuit* problem:

$$\widehat{\boldsymbol{X}} = \min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|_2^2 + \lambda\|\boldsymbol{X}\|_1 \tag{3.3}$$

We note that the optimization problem in (3.3) cannot be decomposed into different independent problems defined by (3.2), as it has a global sparsity constraint over the complete matrix $\boldsymbol{X}$ rather than over its individual columns. Equation (3.3) could arise in different signal processing applications, such as blind source separation and tomography.

3.2 Least Angle Regression (LARS)

The LARS method was proposed in [53] as a method for solving the sparse least-squares problem defined by (3.1). In addition, the authors in [53] also explained how LARS could be modified to solve the *basis pursuit* minimization problem in a more efficient way than traditional convex optimization techniques. LARS is considered a modification of OMP that solves the optimization problem in (3.1) by including one component of the vector $\boldsymbol{x}$ at a time. At each step, OMP determines the best component to join the active components and then updates the least-squares solution. LARS uses a similar strategy as it identifies the component most correlated with the residual at each step. Rather than fit this component completely, LARS moves the coefficient of this component continuously toward its least-squares solution, thereby causing its correlation with the residual to decrease in absolute value. Once another component correlates with the residual, the process is paused. This new component then joins the active components, and their coefficients are moved together in a way that maintains equal and decreasing correlations with the residual. This process continues until all the components are included, and it ends when the full least-squares solution is obtained [61].

Solving the *basis pursuit* minimization problem using LARS allows the number of the coefficients of the solution vector $\boldsymbol{x}$ to increase along with the number of active components. The LARS method's most attractive property is that it can efficiently generate the sequence of all solutions for the optimization problem in (3.1) for all values of the regularization parameter λ [6], as shown in Figure 3.1. In the next section, the basic LARS is described, and we explain how it can be modified in order to solve the *lasso* problem.

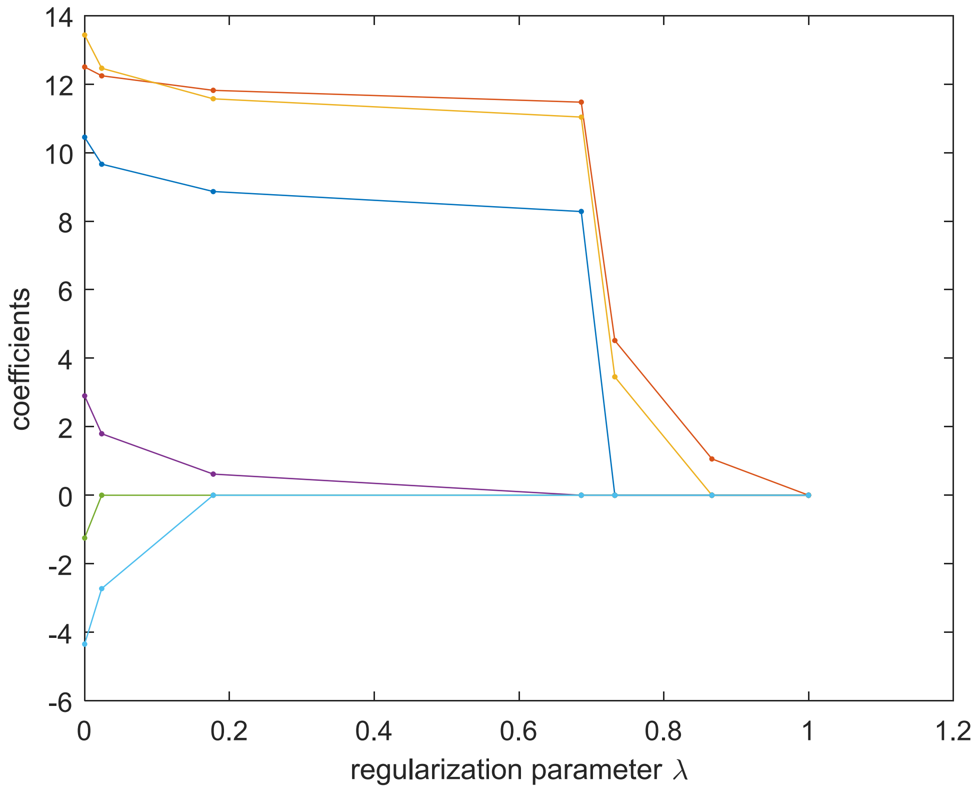


Figure 3.1. All solutions of LARS at all values of λ (regularization path). Each color represents a coefficient in the solution that changes with every iteration.

3.3 The Basic LARS Algorithm

Forward stepwise regression is a procedure that builds a model by adding one variable "predictor" at a time. At each step, it characterizes the best variable to join the active variables and then updates the least-squares solution to include all of the active variables.

LARS uses a similar strategy, and only it enters as much of a predictor as is warranted. At each step, LARS identifies the most correlated variable with the residual; however, rather than fit this variable completely, LARS moves its coefficient continuously toward its least-squares solution, thus causing its correlation with the updated residual to decrease in absolute value. Once another variable correlates with the updated residual, the process is paused. This new variable then joins the active variables, and their coefficients are moved together in a way that ensures equal and decreasing correlations with the residual. This process continues until all the variables are in the model, and it ends when the full least-squares fit is obtained [61]. The number of the LARS solution vectors' coefficients increases according to increases in the active variables. The details of the LARS algorithm are as follows:

For the optimization problem in (3.1), as a preprocessing step, we normalize the columns of the matrix $\boldsymbol{H}$ to have unit l_2-norm (i.e. $\|\boldsymbol{h}_i\|_2 = 1, i = \{1,2,\dots,nn\}$, where $\boldsymbol{h}_i$ is the i^{th} column of the matrix $\boldsymbol{H}$ and nn is the number of columns of $\boldsymbol{H}$).

LARS begins by initializing the solution $\boldsymbol{x}$ with all-zero coefficients. The residual vector $\boldsymbol{r}$, which is the remaining part of the measurement vector $\boldsymbol{b}$ after the solution vector $\boldsymbol{x}$ has been updated or $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}$, is initialized with the measurement vector $\boldsymbol{b}$. The active set I, which has the indices of the selected columns from the matrix $\boldsymbol{H}$, is initialized with an empty set.

Then, at each iteration t, LARS computes the correlation vector by multiplying columns of the matrix $\boldsymbol{H}$ by the residual vector $\boldsymbol{r}$ from the previous iteration as follows:

$$\boldsymbol{c}_t = \boldsymbol{H}^T\boldsymbol{r} \tag{3.4}$$

A new column is chosen from the matrix $\boldsymbol{H}$, which is highly correlated with the current residual $\boldsymbol{r}$, and its index is added to the active set. This is done by selecting the column from the matrix $\boldsymbol{H}$ that corresponds to the entry in the vector $\boldsymbol{c}_t$ with the largest magnitude, i.e., the largest value of the vector $\boldsymbol{c}_t$ is determined by considering the absolute value of vector, $\boldsymbol{c}_t$, entries. This can be expressed by the following equation:

$$j = \max_{1 \le j \le nn} |\boldsymbol{c}_t(j)| \tag{3.5}$$

where j is the index of the selected column.

In the first iteration, LARS chooses a column, $\boldsymbol{h}_j$, which is the most correlated column in $\boldsymbol{H}$ with the residual vector $\boldsymbol{r}$. This means that the residual vector $\boldsymbol{r}$ will have the smallest angle with the column $\boldsymbol{h}_j$ among the other columns of the matrix $\boldsymbol{H}$. Then, LARS changes the coefficient, $\boldsymbol{x}(j)$ in the solution, associated with the chosen column $\boldsymbol{h}_j$, to a nonzero value. As $\boldsymbol{x}(j)$ is added, the absolute correlation value between $\boldsymbol{h}_j$ and the current residual is decreased. LARS takes the smallest possible steps in the direction of the column $\boldsymbol{h}_j$ until another column, $\boldsymbol{h}_k$, has as much absolute correlational value with the updated residual as the column $\boldsymbol{h}_j$. Now, instead of continuing in the direction of $\boldsymbol{h}_j$, LARS moves in the direction that is equiangular with the selected columns $(\boldsymbol{h}_j, \boldsymbol{h}_k)$ until a third column, $\boldsymbol{h}_m$, has an absolute correlational value to the current residual that is equal to $\boldsymbol{h}_j$ and $\boldsymbol{h}_k$. The procedure is repeated, adding one column at a time until no remaining columns have a correlational value with the current residual; that is until all columns of $\boldsymbol{H}$ are in the active set. The name, "*least* *angle,"* is derived from a geometrical interpretation of the LARS process, which chooses the updated direction that makes the smallest and equal angle with all active columns.

At each iteration t, the *updated direction* vector $\boldsymbol{d}_t$, which should be equiangular with all the active columns $\boldsymbol{H}_I$, is calculated for the entries of the active set I using the following equation:

$$\boldsymbol{d}_t(I) = \frac{1}{\sigma_t}(\boldsymbol{H}_I^T\boldsymbol{H}_I)^{-1}\boldsymbol{H}_I^T\boldsymbol{r}_{t-1} \tag{3.6}$$

where σ_t is a scalar value representing the equal absolute correlational value between the residual vector r and the active columns $\boldsymbol{H}_I$ at each iteration t. The entries of $\boldsymbol{d}_t$ that are not in the active set are set to zero (i.e. $\boldsymbol{d}_t(I^c) = 0$).

At each iteration t, the *step size* δ_t is a scalar that is multiplied by the updated direction $\boldsymbol{d}_t$ to update the solution vector $\boldsymbol{x}$ that corresponds to the active set. LARS selects the smallest step size that will cause a column from the inactive set I^c to join the active set I at the next iteration.

$$\delta_t = \min_{i \in I^c}{}^{+}\left\{\frac{\sigma_t - \boldsymbol{c}_t(i)}{1 - \boldsymbol{h}_i{}^T\boldsymbol{v}_t}, \frac{\sigma_t + \boldsymbol{c}_t(i)}{1 + \boldsymbol{h}_i{}^T\boldsymbol{v}_t}\right\} \tag{3.7}$$

Where $\boldsymbol{v}_t = \boldsymbol{H}_I\boldsymbol{d}_t(I)$. In (3.7), only the positive components within every choice of i are considered when the minimum is taken.

After the updated direction, $\boldsymbol{d}_t$, and the step size, δ_t, are calculated, LARS updates the solution vector $\boldsymbol{x}$ using:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \delta_t\boldsymbol{d}_t \tag{3.8}$$

and computes the new residual $\boldsymbol{r}_t$ using one of these formulas:

$$\boldsymbol{r}_{t+1} = \boldsymbol{b} + \boldsymbol{H}\boldsymbol{x}_t \qquad or \qquad \boldsymbol{r}_{t+1} = \boldsymbol{r}_t + \delta_t \boldsymbol{v}_t \tag{3.9}$$

LARS repeats these steps until the active set I contains all columns of $\boldsymbol{H}$ or as long as there is still one column from the inactive set I^c that has a correlation with the current residual. When no remaining columns have a correlation with the current residual, i.e., σ approaches zero), LARS terminates and the vector $\boldsymbol{x}_t$ is returned as the final solution.

To further illustrate the LARS algorithm, consider the following example: Assume that the matrix $\boldsymbol{H}$ contains two columns $(\boldsymbol{h}_1, \boldsymbol{h}_2)$. LARS initializes the solution vector to all zero coefficients and the residual to the measurement vector $\boldsymbol{b}$. As shown in Figure 3.2, LARS begins by selecting the column $\boldsymbol{h}_1$ because it has a a greater absolute correlation with the initial residual (the vector $\boldsymbol{b}$) than with $\boldsymbol{h}_2$(i.e. $\theta_1(1) < \theta_1(2)$, where $\theta_t(i)$ is the angle between the column $\boldsymbol{h}_i$ and the current residual $\boldsymbol{r}$ at iteration t). Then, the LARS algorithm proceeds in the direction of $\boldsymbol{h}_1$ by adding the step size γ_1(the bold blue line in Figure 3.2), which is chosen such that columns $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ have the same absolute correlation with the current residual at the next iteration (i.e. $\theta_2(1) = \theta_2(2)$). When the solution vector $\boldsymbol{x}$ is updated, the solution coefficient becomes $\boldsymbol{x}_1(1) = \delta_1$.

At the second iteration, LARS adds the column $\boldsymbol{h}_2$ to the active set and proceeds in the direction that is equiangular with the columns $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$. Because there is no remaining column, LARS adds step size δ_2 (the bold red line in Figure 3.2) that leads to the vector $\boldsymbol{b}$. LARS then terminates because the residual is zero and the solution coefficients will be: $\boldsymbol{x}_2(1) = \delta_1 + \delta_2\boldsymbol{d}_2(1)$ and $\boldsymbol{x}_2(2) = \delta_2\boldsymbol{d}_2(2)$, where $\boldsymbol{d}_2$ is the updated direction at the second iteration, which is equiangular with the active columns $(\boldsymbol{h}_1, \boldsymbol{h}_2)$.

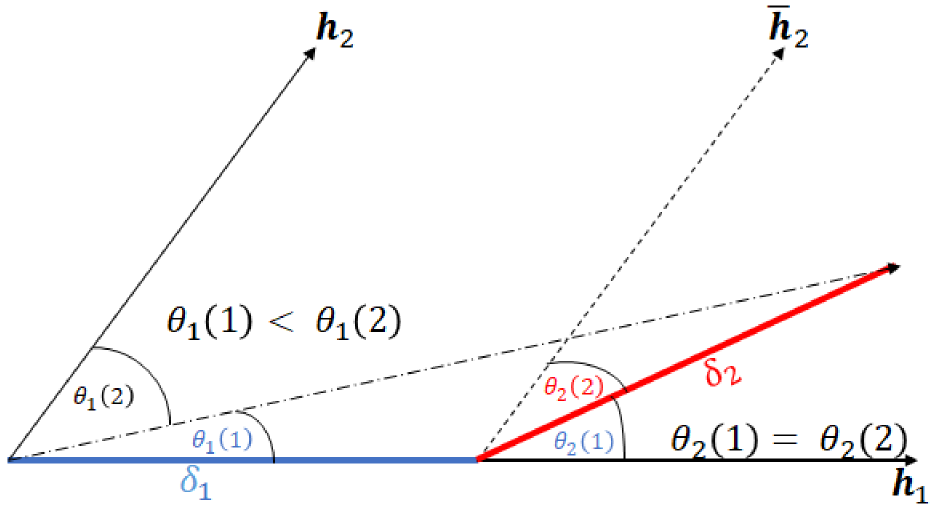


Figure 3.2. Using LARS to approximate the vector $\boldsymbol{b}$ by using the columns $(\boldsymbol{h}_1, \boldsymbol{h}_2)$.

3.3.1 Modifying LARS for solving the basis pursuit problem

LARS can be used to solve the *basis pursuit* problem if it satisfies 3 main constraints. First, the absolute correlations of the active columns $\boldsymbol{H}_I$ must be equal to σ, which is already maintained by LARS. Second, the absolute correlations of the inactive columns must be equal to or less than λ, which is also maintained by LARS. Third, the signs of the correlation and the solution vectors must be matched for the active entries I, which LARS does not enforce. Therefore, LARS should be modified to maintain the sign-matching constraint, which can be written as [53, 62]:

$$sign(\boldsymbol{x}(I)) = sign(\boldsymbol{c}(I)) \tag{3.10}$$

These signs could be different when one of the solution coefficients associated with the active set I crosses zero. Therefore, if any solution coefficient associated with the active set I equals zero during the LARS procedures, the column corresponding to this coefficient should be removed from the active set I and the updated direction $\boldsymbol{d}_t$ should be recalculated by solving (3.6). Hence, we need to determine the step size δ_t that causes one of the solution coefficients associated with the active set I to become zero after updating the solution $\boldsymbol{x}$.

The smallest positive step size that causes the i^{th} solution coefficient associated with the active set I to become zero can be found via the following minimization problem:

$$\delta_t = \min_{i \in I}{}^{+} \left\{ \frac{\boldsymbol{x}_{t-1}(i)}{\boldsymbol{d}_t(i)} \right\} \tag{3.11}$$

where only positive components within every choice of i are considered when the minimum is taken.

Therefore, at each iteration, the modified LARS for solving the *basis pursuit* problem computes two step sizes: one adds a column to the active set I as stated by (3.7), and the other drops a column from the active set I as stated by (3.11). To distinguish between these two step-size values, δ^+ is used to refer to the step size computed by (3.7), and δ^- is used to refer to the step size computed by (3.11). The modified LARS then selects the smallest step size:

$$\delta_t = \min\{\delta_t^+, \delta_t^-\} \tag{3.12}$$

where

$$\delta_t^+ = \min_{i \in I^c}{}^{+} \left\{ \frac{\sigma_t - \boldsymbol{c}_t(i)}{1 - \boldsymbol{h}_i{}^T \boldsymbol{v}_t}, \frac{\sigma_t + \boldsymbol{c}_t(i)}{1 + \boldsymbol{h}_i{}^T \boldsymbol{v}_t} \right\}$$

$$\delta_t^- = \min_{i \in I}{}^{+} \left\{ \frac{x_{t-1}(i)}{\boldsymbol{d}_t(i)} \right\} \qquad (3.13)$$

The modified LARS uses the step size computed by (3.12) instead of the step size computed by (3.7) in the basic LARS algorithm. This means that a column can be added to I if $\delta_t = \delta_t^+$, and a column can be removed from I if $\delta_t = \delta_t^-$.

The remaining steps remain the same as they are in the basic LARS algorithm stated in section 3.3. Note that the modified LARS requires more iterations than the basic LARS; this is due to, in the modified LARS, some columns are added to and dropped from the active set I, while in the basic LARS columns are always added to the active set. The modified LARS algorithm for solving the single measurement *basis pursuit* problem (3.2) is summarized in Table 3.1.

Table 3.1. The modified LARS algorithm for solving the *basis pursuit* problem in (3.1).

- Given: The measurement vector $\boldsymbol{b}$, the coefficient matrix $\boldsymbol{H}$.
- Initialization steps: The columns of the $\boldsymbol{H}$ matrix have to be normalized to unit l_2-norm. The solution vector $\boldsymbol{x}$ is initialized with all zero entries. The residual vector $\boldsymbol{r}$ is initialized with $\boldsymbol{b}$. The active set I is initialized with an empty set.
- For each iteration t:
  - The correlations vector $\boldsymbol{c}_t$ is obtained by: $\boldsymbol{c}_t = \boldsymbol{H}^T\boldsymbol{r}$
  - The updated direction vector d_t is obtained by: $\boldsymbol{d}_t(I) = \frac{1}{\sigma_t}(\boldsymbol{H}_I^T\boldsymbol{H}_I)^{-1}\boldsymbol{H}_I^T\boldsymbol{r}_{t-1}$
  - The step size δ_t is obtained by: $\delta_t = min\{\delta_t^+, \delta_t^-\}$

    $$\delta_t^+ = \min_{i\in I^c}{}^+\left\{\frac{\sigma_t-\boldsymbol{c}_t(i)}{1-\boldsymbol{h}_i{}^T\boldsymbol{v}_t}, \frac{\sigma_t+\boldsymbol{c}_t(i)}{1+\boldsymbol{h}_i{}^T\boldsymbol{v}_t}\right\} \quad \text{and} \quad \delta_t^- = \min_{i\in I}{}^+\left\{\frac{x_{t-1}(i)}{\boldsymbol{d}_t(i)}\right\}$$

    where $\boldsymbol{v}_t = \boldsymbol{H}_I\boldsymbol{d}_t(I)$.

    If $\delta_t = \delta_t^+,\ I = I \cup i$ and if $\delta_t = \delta_t^-, I = I - i$.

  - Update the solution matrix $\boldsymbol{x}$ using: $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \delta_t\boldsymbol{d}_t$
  - Update the residual matrix using: $\boldsymbol{r}_{t+1} = \boldsymbol{r}_t + \delta_t\boldsymbol{v}_t$

3.4 The Kronecker Least Angle Regression (K-LARS)

In this section, we describe an extension of the LARS algorithm that can enable the matrix *basis pursuit* optimization problem in (3.3) to be solved. In this extension, the measurement vector $\boldsymbol{b}$ in (3.2) is replaced by an $L \times N$ matrix $\boldsymbol{Y}$ containing N multichannel measurements of dimension L, and $\boldsymbol{x}$ in (3.2) is replaced by an $L \times M$ matrix $\boldsymbol{X}$ containing M components of the unknown signal of dimension L. The coefficient matrix $\boldsymbol{H}$ will tends to the $M \times N$ mixing matrix $\boldsymbol{A}$.

To solve the matrix *basis pursuit* minimization problem in (3.3), one would write the matrix linear system of equations, $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A}$, as a vector linear system of equations

$$\mathrm{vec}(\boldsymbol{Y}) = (\boldsymbol{A}^T \otimes \boldsymbol{I}_L)\mathrm{vec}(\boldsymbol{X}) \tag{3.14}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix obtained by stacking its columns on top of one another, $\otimes$ denotes Kronecker product, and $\boldsymbol{I}_L$ is an $L \times L$ identity matrix. Therefore, the matrix *basis pursuit* in (3.3), could be written in a vectorized form as

$$\min_{\boldsymbol{X}} \tfrac{1}{2}\|(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)\mathrm{vec}(\boldsymbol{X}) - \mathrm{vec}(\boldsymbol{Y})\|_2^2 + \lambda\|\mathrm{vec}(\boldsymbol{X})\|_1 \tag{3.15}$$

In the optimization problem (3.15), we note that $\mathrm{vec}(\boldsymbol{Y})$ has dimensions $(L * N) \times 1$, $\mathrm{vec}(\boldsymbol{X})$ has dimensions $(L * M) \times 1$, and $\boldsymbol{A}^T \otimes \boldsymbol{I}_L$ has dimensions $(L * N) \times (L * M)$. Therefore, even relatively small dimensions of the given measurements, e.g., the case of a small region of a hyperspectral cube has $N = 75 \times 75$ spectral pixels, $L = 512$ wavelengths, $M = 3$ endmembers, would result in $\mathrm{vec}(\boldsymbol{Y})$ with dimensions $(2.88 \times 10^6) \times 1, \mathrm{vec}(\boldsymbol{X})$ with dimensions $(1.54 \times 10^3) \times 1$, and $\boldsymbol{A}^T \otimes \boldsymbol{I}_L$ with dimensions $(2.88 \times 10^6) \times (1.54 \times 10^3)$. Such extremely large arrays (~10^9 elements) would be very challenging to process and would require very large computer storage (~10^{10} bytes).

In this section, we use the properties of Kronecker products to extend the standard LARS algorithm to solve the matrix *basis pursuit* in (3.3) without actually vectorizing it. This would allow practical and efficient recovery of sparse multicomponent signals from multichannel data, i.e., without constructing or processing very large arrays. We call this extension of standard LARS the Kronecker LARS (K-LARS) algorithm.

The following property of the Kronecker product will be used in our Kronecker LARS algorithm [54]:

Property 1: If $\boldsymbol{EE}$ and $\boldsymbol{FF}$ are matrices with vectorizations given by $\text{vec}(\boldsymbol{EE})$ and $\text{vec}(\boldsymbol{FF})$, respectively, then the inner product of these vectors can be obtained using the original matrices as follows:

$$\text{vec}(\boldsymbol{EE})^T \, \text{vec}(\boldsymbol{FF}) = \text{trace}(\boldsymbol{EE}^T\boldsymbol{FF}) \tag{3.16}$$

This property will be used to obtain the required inner products between any two large columns by treating them as matrices without actually vectorizing it.

3.4.1 Steps of the Kronecker LARS algorithm

Initial steps:

- Divide all elements of $\boldsymbol{Y}$ by $\sqrt{\text{trace}(\boldsymbol{Y}^T\boldsymbol{Y})}$. This is equivalent to normalizing $\text{vec}(\boldsymbol{Y})$ to have unit *l*2-norm i.e. $\|\text{vec}(\boldsymbol{Y})\|_2$. Subtract the mean of all elements of $\boldsymbol{Y}$ from all the elements of $\boldsymbol{Y}$. A simple post-processing step could easily offset this normalization and centering of $\boldsymbol{Y}$.
- Normalize all columns of the matrix $\boldsymbol{A}^T$ to have unit *l*2-norm. A simple post-processing step could easily offset this normalization.
- Set all elements of matrix $\boldsymbol{X}$ to zero.
- Define an initial residual matrix $\boldsymbol{R}_0$ and set it equal to $\boldsymbol{Y}$.
- Define an $L \times M$ active set matrix $\boldsymbol{Z}_0$, where the indices of its nonzero elements would specify active columns selected from $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$, and set all its elements to zero.
- Obtain correlations between all columns of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ and $\text{vec}(\boldsymbol{R}_0)$ as $\boldsymbol{C}_1 = \boldsymbol{R}_0\boldsymbol{A}^T$. Select the column that is most correlated with the residual $\boldsymbol{R}_0$, i.e.,

$$(i_1, j_1) = \max_{\substack{1\le i\le L \\ 1\le j\le M}} |\boldsymbol{C}_1(i,j)| \tag{3.17}$$

- Generate an $L \times M$ mask matrix, $\boldsymbol{K}^{i,j} = (\boldsymbol{I}_L)_{:,\boldsymbol{i}} \times (\boldsymbol{I}_M)_{j,:}$, whose elements are all zeros, except a single element of unit value at (i, j). Therefore, any column of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ specified by (i, j) would be equal to vec$(\boldsymbol{K}^{i,j}\boldsymbol{A})$.
- Update the active set matrix, $\boldsymbol{Z}_1 = \boldsymbol{Z}_0 + \boldsymbol{K}^{i_1,j_1}$, to indicate that the column of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ specified by (i_1, j_1) now belongs to the active set.

<u>For each iteration t = 1, 2,…</u>

- Obtain correlations between all columns of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ and $\boldsymbol{R}_{t-1}$ as $\boldsymbol{C}_t = \boldsymbol{R}_{t-1}\boldsymbol{A}^T$.
- Set λ_t to the maximum value of $|\boldsymbol{C}_t(i, j)|$.
- Obtain the Gram matrix $\boldsymbol{G}$ of the active columns of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ corresponding to all nonzero entries of $\boldsymbol{Z}_t$ specified by (i, j). Each element $\boldsymbol{G}(m, n)$ is given by the inner product of the m^{th} and n^{th} active columns, specified by (i_m, j_m), and (i_n, j_n), respectively, using *property 1* given above,

$$\boldsymbol{G}_t(m, n) = \mathrm{trace}(\boldsymbol{A}^T(\boldsymbol{K}^{i_m,j_m})^T\boldsymbol{K}^{i_n,j_n}\boldsymbol{A}) \tag{3.18}$$

- Obtain a vector, $\boldsymbol{q}_t$, indexed by k where (i_k, j_k) corresponds to all nonzero elements of $\boldsymbol{Z}_t$, and whose length is equal to the number of all active columns of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$. Its elements are the inner products between these active columns and vec$(\boldsymbol{R}_t)$, using *property 1* given above,

$$\boldsymbol{q}_t(k) = \mathrm{trace}(\boldsymbol{A}^T(\boldsymbol{K}^{i_k,j_k})^T\boldsymbol{R}_t) \tag{3.19}$$

- Obtain the nonzero elements of the matrix $\boldsymbol{D}_t(i, j)$ that represents the update direction of the solution $\boldsymbol{X}$, as the vector

$$\boldsymbol{u}_t = \frac{1}{\lambda_t}\boldsymbol{G}_t^{-1}\boldsymbol{q}_t \tag{3.20}$$

Using (i_k, j_k) update the matrix $\boldsymbol{D}_t(i, j)$ accordingly.

- For all (i, j) where $\boldsymbol{Z}_t(i, j) = 0$, i.e., indices of the inactive set, obtain the positive step size δ_t^+ using

$$\delta_t^+ = \min{}^+ \left\{ \frac{\lambda_t - \boldsymbol{C}_t(i, j)}{1 - \boldsymbol{V}_t(i, j)}, \frac{\lambda_t + \boldsymbol{C}_t(i, j)}{1 + \boldsymbol{V}_t(i, j)} \right\} \tag{3.21}$$

where $\boldsymbol{V}_t = \boldsymbol{D}_t \boldsymbol{A} \boldsymbol{A}^T$. Set (i_{add}, j_{add}) to the indices corresponding to δ_t^+ in (3.21).

- For all (i, j) where $\boldsymbol{Z}_t(i, j) = 1$, i.e., indices of the active set, obtain the negative step size δ_t^- using

$$\delta_t^- = \min{}^+ \left\{ \frac{\boldsymbol{X}_{t-1}(i, j)}{\boldsymbol{D}_t(i, j)} \right\} \tag{3.22}$$

Set (i_{remove}, j_{remove}) to the indices corresponding to δ_t^- in (3.22).

- Obtain the step size δ_t^* using

$$\delta_t^* = \min \{\delta_t^+, \delta_t^-\} \tag{3.23}$$

- Update both solution matrix $\boldsymbol{X}$ and residual matrix using

$$\boldsymbol{X}_t = \boldsymbol{X}_{t-1} + \delta_t^* \boldsymbol{D}_t \tag{3.24}$$

$$\boldsymbol{R}_t = \boldsymbol{R}_{t-1} - \delta_t^* \boldsymbol{D}_t \boldsymbol{A} \tag{3.25}$$

- If $\delta_t^* = \delta_t^+$, set $\boldsymbol{Z}_{t+1} = \boldsymbol{Z}_t + \boldsymbol{K}^{i_{add}, j_{add}}$, to indicate that the column of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ specified by (i_{add}, j_{add}) now belongs to the active set.

- Else if $\delta_t^* = \delta_t^-$, set $\boldsymbol{Z}_{t+1} = \boldsymbol{Z}_t - \boldsymbol{K}^{i_{remove}, j_{remove}}$, to indicate that the column of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ specified by (i_{remove}, j_{remove}) now belongs to the inactive set.

Continue iterations until all columns of $(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)$ are included in the active set, all values of the correlation matrix $\boldsymbol{C}$ become equal, λ reaches its minimum value, and all entries of the active set matrix $\boldsymbol{Z}$ are ones. This is equivalent to reaching the full least-squares solution. However, typically one would terminate the iterations when the desired sparsity level is reached or when the norm of the residual becomes within a prescribed tolerance. The K-LARS algorithm for solving the matrix *basis pursuit* problem (3.3) is summarized in Table 3.2.

Table 3.2. The K-LARS algorithm steps for solving the *basis* pursuit problem in (3.3).

- Given: The multichannel measurement matrix $\boldsymbol{Y}$, the coefficient matrix $\boldsymbol{A}$.
- Initialization steps: The columns of the $\boldsymbol{A}^T$ matrix are normalized to unit l_2-norm. The solution vector $\boldsymbol{X}$ is initialized with all zero entries. The residual vector $\boldsymbol{R}$ is initialized with $\boldsymbol{Y}$. The active set matrix $\boldsymbol{Z}$ is initialized with $\mathrm{L} \times \mathrm{M}$ zero matrix.
- For each iteration t:
  - The correlations matrix $\boldsymbol{C}_t$ is obtained using: $\boldsymbol{C}_t = \boldsymbol{R}_t \mathbf{A}^T$
  - Select the indices of the most correlated column using:

$$i, j = \max_{\substack{1 \le i \le \mathrm{L} \\ 1 \le j \le \mathrm{M}}} |\boldsymbol{C}_t(i, j)|$$

  - Set $\boldsymbol{Z}(i, j) = 1$ and obtain $\lambda_t = \max|\boldsymbol{C}_t(i, j)|$
  - Obtain the Gramian matrix element-by-element using:

$$\boldsymbol{G}_t(m, n) = \mathrm{trace}(\boldsymbol{A}^{\boldsymbol{T}}(\boldsymbol{K}^{i_m, j_m})^T \boldsymbol{K}^{i_n, j_n} \boldsymbol{A})$$

using property (3.16) where $\boldsymbol{K}^{i,j} = (\boldsymbol{I}_L)_{:,i} \times (\boldsymbol{I}_M)_{j,:}$.

  - The updated direction of the solution $\boldsymbol{D}_t$ is obtained by: $\boldsymbol{D}_t(i, j) = \frac{1}{\lambda_t} \boldsymbol{G}_t^{-1} \boldsymbol{q}_t$

where $\boldsymbol{q}$ is obtained element-by-element using: $\boldsymbol{q}_t(k) = \mathrm{trace}(\boldsymbol{A}^T (\boldsymbol{K}^{i_k, j_k})^{\boldsymbol{T}} \boldsymbol{R}_t)$

  - The step size δ_t^* is obtained by: $\delta_t^* = \min\ \{\delta_t^+, \delta_t^-\}$

$$\delta_t^+ = \min{}^+ \left\{ \frac{\lambda_t - \boldsymbol{C}_t(i, j)}{1 - \boldsymbol{V}_t(i, j)}, \frac{\lambda_t + \boldsymbol{C}_t(i, j)}{1 + \boldsymbol{V}_t(i, j)} \right\}$$

$$\delta_t^- = \min{}^+ \left\{ \frac{\boldsymbol{X}_{t-1}(i, j)}{\boldsymbol{D}_t(i, j)} \right\}$$

where $\boldsymbol{V}_t = \boldsymbol{D}_t \boldsymbol{A}\boldsymbol{A}^T$

If $\delta_t^* = \delta_t^+$, $\boldsymbol{Z}(i,j) = 1$ and if $\delta_t^* = \delta_t^-$, $\boldsymbol{Z}(i,j) = 0$.

- Update the solution matrix $\boldsymbol{X}$ using: $\boldsymbol{X}_t = \boldsymbol{X}_{t-1} + \delta_t^* \boldsymbol{D}_t$
- Update the residual matrix $\boldsymbol{R}_{t+1}$ using: $\boldsymbol{R}_{t+1} = \boldsymbol{R}_t + \delta_t^* \boldsymbol{V}_t$

3.5 Chapter summary

In this chapter, we described the Least Angle Regression (LARS), and we described how it could be extended so it can be used to solve the matrix *basis pursuit* problem. We presented the Kronecker LARS (K-LARS) algorithm as a viable alternative that can handle multichannel data. In the K-LARS algorithm, we exploited the properties of Kronecker product to process very large matrices without actually vectorizing them to avoid the need for very large computer storage.

Chapter 4
Kronecker LARS as Homotopy Algorithm

In this chapter, we discuss the Kronecker LARS algorithm's potential to be used dynamically to update the *basis pursuit* problem solutions using the last available estimated signal. Instead of solving a new *basis pursuit* problem from scratch or starting from an initial estimate of zero at every iteration, we use the last available signal estimate as the starting point in a homotopy formulation that is similar to recursive methods. We will begin by discussing the Least-Angle Regression (LARS) algorithm's potential use as a homotopy algorithm, and we will then show how *basis pursuit* problem solutions can be updated dynamically using the homotopy algorithm. We will present the homotopy K-LARS algorithm as an extension of the homotopy LARS algorithm to solve matrix *basis pursuit* problems for multichannel signals in a more computationally efficient way, thereby faster than K-LARS.

4.1 LARS homotopy algorithm

Consider the vector *basis pursuit* problem described in the previous two chapters:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \ \|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \tag{4.1}$$

The dynamic *basis pursuit* updating schemes are based on homotopy continuation [63]. In this section, we illustrate the LARS homotopy algorithm that will be used to solve problem (4.1). Briefly stated, the LARS homotopy algorithm solves (4.1) by finding the solution path for decreasing values of λ, which means that any point on the regularization path is a solution of (4.1) for that particular value of λ [53, 61-64].

The homotopy LARS algorithm starts with the max value of λ, then begins decreasing λ towards its minimum value using the sequence of processing steps described in Section 3.2. As λ changes, the solution of (4.1) follows a linear step whose size and the direction is completely determined by its solution's active set and sign. This relationship can be obtained by analyzing the optimality conditions for (4.1), as given below in (4.2) [62, 63].

The solution's active set changes as λ decreases as a result of a new column joining the active set, or an already existing column jumping out from the active set. For every homotopy step, we move from one value of λ to the next and update the active set of the solution until λ has been lowered to its minimum value.

In every homotopy step, the update direction and the step size for moving the solution to a lower value of λ can be easily obtained using certain optimality conditions, which can be derived using the subdifferential of the objective in (4.1) [62, 65, 66]. At any value of λ , the solution $\boldsymbol{x}^*$ for (4.1) must satisfy the following optimality conditions [67]:

$$\begin{aligned} \boldsymbol{H}_I{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) &= -\lambda \boldsymbol{z} \\ \|\boldsymbol{H}_{I^c}{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b})\|_\infty &\le \lambda \end{aligned} \tag{4.2}$$

where I denotes the active set or the support of $\boldsymbol{x}^*$, I^c denotes the inactive set, $\boldsymbol{z}$ denotes the sign sequence of $\boldsymbol{x}^*$ on I, and $\boldsymbol{H}_I$ denotes a matrix with columns of $\boldsymbol{H}$ at indices in the active set I. From (4.2), $\boldsymbol{x}^*$ can be obtained directly from $\boldsymbol{H}_I$ and $\boldsymbol{z}$ using:

$$\boldsymbol{x}^* = \begin{cases} \left(\boldsymbol{H}_I{}^T\boldsymbol{H}_I\right)^{-\mathbf{1}}\left(\boldsymbol{H}_I{}^T\boldsymbol{b} - \lambda\boldsymbol{z}\right), & on\ I \\ 0, & otherwise \end{cases} \tag{4.3}$$

We can denote the objective function in (4.1) as $f(\boldsymbol{x})$; although this function is convex, it's not differentiable everywhere. Each element of $\partial f(\boldsymbol{x}^*)$ is a subgradient of $f(\boldsymbol{x})$ at $\boldsymbol{x}^*$, and the subdifferential is a generalization of the gradient of $f(\boldsymbol{x})$. In fact, if $f(\boldsymbol{x})$ is convex and differentiable at a point $\boldsymbol{x}$, then:

$$\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\} \tag{4.4}$$

If $f(\boldsymbol{x})$ is convex and differentiable, then its subdifferential at $\boldsymbol{x}$ is the same as the gradient. The optimality conditions state that the subdifferential of $f(\boldsymbol{x})$ at $\boldsymbol{x}^*$ has to contain the 0-vector for $f(\boldsymbol{x})$ if it is to determine a global minimum at $\boldsymbol{x}^*$, i.e., $0 \in \partial f(\boldsymbol{x}^*)$ [63]. The subdifferential of a convex function at a point $\boldsymbol{x}$ (where it is non-differentiable) is defined as the set of all subgradients of the function at that point. A vector $\boldsymbol{g} \in \mathbb{R}^L$ is a subgradient of $f\colon \mathbb{R}^L \rightarrow R \; at \; \boldsymbol{x} \in \mathbb{R}^L$ if for all $\boldsymbol{u} \in \mathbb{R}^L$,

$$f(\boldsymbol{u}) = f(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{u} - \boldsymbol{x}) \tag{4.5}$$

which means that $f(\boldsymbol{u})$ remains below the graph of f for any $\boldsymbol{u}$ [65, 68]. We calculate the subdifferential of f as:

$$\partial f(\boldsymbol{x}) = \lambda \, \partial \|\boldsymbol{x}\|_1 + \boldsymbol{H}^T(\boldsymbol{Hx} - \boldsymbol{b}) \tag{4.6}$$

where $\partial\|\boldsymbol{x}\|_1$ denotes the subdifferential of the l_1-norm, can be obtained as follows:

$$\partial\|\boldsymbol{x}\|_1 = \left\{ \boldsymbol{g} \in \mathbb{R}^L \left| \begin{array}{ll} g_i = +1, & x_i > \boldsymbol{0} \\ g_i = -1, & x_i < \boldsymbol{0} \\ g_i \in [-1,1], & x_i = \boldsymbol{0} \end{array} \right. \right\} \tag{4.7}$$

This function shows that $\partial\|\boldsymbol{x}\|_1$ is uniquely defined for the nonzero entries in $\boldsymbol{x}$ with the sign sequence, while it can have any value in $[-1,1]$ for the zero entries in $\boldsymbol{x}$. Using the subdifferential $\boldsymbol{g} = \partial\|\boldsymbol{x}^*\|_1$ in (4.6), the optimality condition, $0 \in \partial f(\boldsymbol{x}^*)$, for a vector $\boldsymbol{x}^*$ can be described as:

$$\lambda \boldsymbol{g} + \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) = \boldsymbol{0}$$
$$\|\boldsymbol{g}\|_\infty \leq 1$$
$$\boldsymbol{g}^T\boldsymbol{x}^* = \|\boldsymbol{x}^*\|_1 \tag{4.8}$$

Thus, a vector $\boldsymbol{x}^*$ with an active set I and sign sequence, $\boldsymbol{z}$ will yield the optimality conditions described in (4.2).

The optimality conditions in (4.2) can be described as L constraints that the solution $\boldsymbol{x}^*$ needs to satisfy with equality in the active set I, and inequality elsewhere, specifically in the inactive set I^c. The violation of these constraints will only be at the different values of λ when the support changes. As we reduce λ to $\lambda - \delta$, for a small value of δ will cause the solution to move in a direction $\partial\boldsymbol{x}$. In order to maintain optimality, this process must obey:

$$\boldsymbol{H}_I{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + \delta\boldsymbol{H}_I{}^T\boldsymbol{H}\partial\boldsymbol{x} = -(\lambda - \delta)\boldsymbol{z}$$
$$\|\boldsymbol{H}_{I^c}{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + \delta\boldsymbol{H}_{I^c}{}^T\boldsymbol{H}\partial\boldsymbol{x}\|_\infty \leq (\lambda - \delta) \tag{4.9}$$

The update direction that keeps the solution optimal as λ changes can be written as:

$$\partial\boldsymbol{x} = \begin{cases} \left(\boldsymbol{H}_I{}^T\boldsymbol{H}_I\right)^{-1}\boldsymbol{z} & on\ I \\ 0 & otherwise \end{cases} \tag{4.10}$$

The solution $\boldsymbol{x}^*$ moves in direction $\partial\boldsymbol{x}$ until one of the two constraints in (4.9) is violated, which indicates that one column has to be added to the active set I, or one of the nonzero elements in $\boldsymbol{x}^*$ shrinks to zero, which indicates that a column must be removed from I. The smallest step size that will cause one of these changes in the support of the active set can be easily computed as $\delta^* = \min\left(\delta^+, \delta^-\right)$, where:

$$\delta^+ = \min_{i \in I^c}{}^+ \left\{\frac{\lambda - \boldsymbol{p}(i)}{1 - \boldsymbol{k}(i)}, \frac{\lambda + \boldsymbol{p}(i)}{1 + \boldsymbol{k}(i)}\right\}$$

$$\delta^- = \min_{i\in I}{}^{+}\left\{\frac{\boldsymbol{x}^*(i)}{\partial \boldsymbol{x}(i)}\right\} \tag{4.11}$$

where $\boldsymbol{p} = \boldsymbol{H}_{I^c}{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b})$, $\boldsymbol{k} = \delta \boldsymbol{H}_{I^c}{}^T\boldsymbol{H}\partial\boldsymbol{x}$. Note that only positive components within every choice of i are considered when the minimum is taken.

δ^+ is the smallest step size that will cause the inactive set constraint to be violated and to activate the column in the inactive set at index $i \in I^c$, thus indicating that $\boldsymbol{h}_i$ should enter the active set. δ^- is the smallest step size that will cause the active set constraint to be violated, thereby shrinking an existing element at index $i \in I$ to zero. The new critical value of λ becomes $\lambda - \delta^*$, the new signal estimate $\boldsymbol{x}^*$ becomes $\boldsymbol{x}^* + \delta^*\partial\boldsymbol{x}$, and the support of the active set I and the sign sequence $\boldsymbol{z}$ change accordingly. At every homotopy step, we compute the update direction and the step size that will cause a one-element change in the support. We repeat this procedure until λ has been lowered to its desired value.

4.2 Dynamic updating for l_1 Homotopy problem

The LARS homotopy method described above solves (4.1) by beginning with a zero vector and building the solution by reducing λ while adding or removing a column in the active set at every homotopy step. In this section, we illustrate a homotopy algorithm that dynamically updates solutions for the l_1-norm minimization problem (4.1). This homotopy procedure starts with a nonzero (warm-start) vector, which is assumed to be close to the desired solution and updates the solution in a sequence of similar homotopy steps.

This homotopy algorithm uses a known (warm-start) vector as a starting point and builds a homotopy path towards the desired solution for the l_1-norm minimization problem (4.1). Instead of solving (4.1) from scratch, the process is sped up by using some prior knowledge of the solution of (4.1). We assume that we have a sparse vector $\widehat{\boldsymbol{x}}$ that is close to the original solution of (4.1), and that has support or an active set I and a sign sequence $\boldsymbol{z}$. The homotopy algorithm can be initialized with an arbitrary vector $\widehat{\boldsymbol{x}}$ because the corresponding matrix $\boldsymbol{H}_{\hat{I}}{}^{T}\boldsymbol{H}_{\hat{I}}$ is invertible; however, the update will be quicker if $\widehat{\boldsymbol{x}}$ is closer to the final solution.

This homotopy method provides a general framework for solving an optimization program by continuously transforming it into a related problem for which the solution is either available or easy to compute. Starting from an available solution, a series of simple problems are solved along the so-called homotopy path towards the final solution of the original problem [28, 64, 69]. The progression along the homotopy path is controlled by the *homotopy parameter* ϵ, which usually varies between 0 and 1 and corresponds to the two end points of the homotopy path.

The homotopy formulation for (4.1) that uses the homotopy parameter, $\epsilon \in [0,1]$, goes as follows: we treat the given warm-start vector $\widehat{\boldsymbol{x}}$ as a starting point and solve the following optimization problem by changing ϵ from 0 to 1:

$$\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 + (1-\epsilon)\boldsymbol{u}^T\boldsymbol{x} \tag{4.12}$$

Next, we define $\boldsymbol{u}$ as:

$$\boldsymbol{u} = -\lambda\hat{\boldsymbol{z}} - \boldsymbol{A}^T(\boldsymbol{H}\widehat{\boldsymbol{x}} - \boldsymbol{b}) \tag{4.13}$$

where $\hat{\boldsymbol{z}}$ is the sign sequence of $\hat{\boldsymbol{x}}$ on $\hat{I}$ and $\hat{\boldsymbol{x}}$ is the optimal solution of (4.12) at $\epsilon = 0$. As ϵ changes from 0 to 1, the optimization problem in (4.12) gradually becomes the one in (4.1), and the solution of (4.12) follows a piece-wise linear homotopy path from $\hat{\boldsymbol{x}}$ to the solution of (4.1).

The optimality conditions for (4.12) can be derived by setting the subdifferential of its objective function to zero [65, 68]. The conditions in which a vector $\boldsymbol{x}^*$ becomes an optimal solution can be described as:

$$\begin{aligned} \lambda\boldsymbol{g} + \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon) &= \boldsymbol{0} \\ \|\boldsymbol{g}\|_\infty &\leq 1 \\ \boldsymbol{g}^T\boldsymbol{x}^* &= \|\boldsymbol{x}^*\|_1 \end{aligned} \tag{4.14}$$

where $\boldsymbol{g} = \partial\|\boldsymbol{x}^*\|_1$ denotes the subdifferential of the *l1*-norm of $\boldsymbol{x}^*$ [62, 70].This implies that, for any given value of $\epsilon \in [0,1]$, the solution $\boldsymbol{x}^*$ for (4.12) must satisfy the following optimality conditions:

$$\begin{aligned} &\boldsymbol{h}_i{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i = -\lambda z_i && for\ all\ i \in I \\ &\left|\boldsymbol{h}_i{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i\right| \leq \lambda && for\ all\ i \in I^c \end{aligned} \tag{4.15}$$

where $\boldsymbol{h}_i$ is the i^{th} column of $\boldsymbol{H}$, I is the support or the active set of $\boldsymbol{x}^*$, z_i is the i^{th} sign in the sign sequence, and u_i is the i^{th}element of $\boldsymbol{u}$.

The optimality conditions in (4.15) can be viewed as L constraints that the solution $\boldsymbol{x}^*$ needs to satisfy with equality (in terms of the magnitude and the sign) on the active set I, and inequality on the inactive set I^c(in terms of the magnitude) elsewhere. The violation of these constraints will occur at the critical values of ϵ where the support or the active set changes when either one column has to be added to the active set I, or one of the nonzero elements in $\boldsymbol{x}^*$ shrinks to zero, thus indicating that a column from I has to be removed. Note that the definition of $\boldsymbol{u}$ in (4.13) ensures that $\hat{\boldsymbol{x}}$ satisfies the optimality conditions in (4.15) at $\epsilon = 0$; hence, it is a valid initial solution. It is clear from (4.15) that, at any value for ϵ, the solution $\boldsymbol{x}^*$ is completely described by the support I and the sign sequence $\boldsymbol{z}$ (assuming that $\left(\boldsymbol{H}_I{}^T\boldsymbol{H}_I\right)^{-1}$exists).

For every homotopy step, we jump from one critical ϵ value to the next $\epsilon + \delta$ and update the support of the solution until ϵ is equal to 1. As we increase ϵ by a small value δ, the solution moves in a direction $\partial\boldsymbol{x}$. This solution has to obey the following optimality conditions:

$$\begin{aligned}
&\boldsymbol{h}_i{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i + \delta(\boldsymbol{h}_i{}^T\boldsymbol{H}\partial\boldsymbol{x} - u_i) = -\lambda z_i && for\ all\ i \in I \\
&\left|\boldsymbol{h}_i{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i + \delta(\boldsymbol{h}_i{}^T\boldsymbol{H}\partial\boldsymbol{x} - u_i)\right| \le \lambda && for\ all\ i \in I^c
\end{aligned} \tag{4.16}$$

The update direction that keeps the solution optimal as δ changes can be written as:

$$\partial\boldsymbol{x} = \begin{cases} \left(\boldsymbol{H}_I{}^T\boldsymbol{H}_I\right)^{-1}\boldsymbol{u}_I & on\ I \\ 0 & otherwise \end{cases} \tag{4.17}$$

The solution $\boldsymbol{x}^*$ moves in direction $\partial\boldsymbol{x}$ until either one of the two constraints in (4.17) is violated, which indicates that one column has to be added to the active set I, or one of the nonzero elements in $\boldsymbol{x}^*$ shrinks to zero, which indicates that we must remove a column from I. The smallest step-size that will cause one of these changes in the support of the active set can be easily computed as $\delta^* = \min(\delta^+, \delta^-)$, where:

$$\delta^+ = \min_{i\in I^c}{}^+\left\{\frac{\lambda - \boldsymbol{p}(i)}{\boldsymbol{k}(i)}, \frac{-\lambda - \boldsymbol{p}(i)}{\boldsymbol{k}(i)}\right\}$$

$$\delta^- = \min_{i\in I}{}^+\left\{\frac{-\boldsymbol{x}^*(i)}{\partial\boldsymbol{x}(i)}\right\} \tag{4.18}$$

where $\boldsymbol{p}(i) = \boldsymbol{h}_i^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i$, $\boldsymbol{k}(i) = \boldsymbol{h}_i^T\boldsymbol{H}\partial\boldsymbol{x} - u_i$, $i \in I^c$. Note that only positive components within every choice of i are considered when the minimum is taken.

δ^+ is the smallest step size that will cause the inactive set constraint to be violated and activate the column in the inactive set at index $i \in I^c$, which indicates that $\boldsymbol{h}_i$ should enter the active set and z_i should be opposite to its sign. δ^- is the smallest step size that will cause the active set constraint to be violated and shrinks an existing element at index $i \in I$ to zero. The new critical value of ϵ becomes $\epsilon + \delta^*$, the new signal estimate $\boldsymbol{x}^*$ becomes $\boldsymbol{x}^* + \delta^*\partial\boldsymbol{x}$, and the support of the active set I and the sign sequence $\boldsymbol{z}$ are updated accordingly. At every homotopy step, the update direction is computed, and the *step size* that causes one-element to be changed in the support is obtained. This procedure is repeated until $\epsilon = 1$.

The above-described homotopy algorithm for dynamically updating the l_1-norm minimization problem solution using a nonzero estimate (warm-start) is summarized in Table 4.1.

Table 4.1. The homotopy algorithm to update the solution of the l_1 problem.

➢ Given: The measurement vector $\boldsymbol{b}$, the coefficient matrix $\boldsymbol{H}$, the warm-start solution $\boldsymbol{x}^*$ and λ.

➢ Initialization steps: The columns of the matrix $\boldsymbol{H}$ must be normalized to unit l_2-norm, $\epsilon = 0$. The active set I is initialized with an empty set.

➢ Repeat

- Obtain the $\boldsymbol{u}$ vector using: $\boldsymbol{u} = -\lambda\hat{\boldsymbol{z}} - \boldsymbol{H}^T(\boldsymbol{H}\hat{\boldsymbol{x}} - \boldsymbol{b})$

  where $\hat{\boldsymbol{z}}$ is the sign sequence of $\hat{\boldsymbol{x}}$.

- Obtain the updated direction using: $\partial\boldsymbol{x} = \begin{cases} \left(\boldsymbol{H}_I{}^T\boldsymbol{H}_I\right)^{-1}\boldsymbol{u}_I & on\ I \\ 0 & otherwise \end{cases}$
- Obtain $\boldsymbol{p}$ and $\boldsymbol{k}$ for $i \in I^c$ using:

  $$\boldsymbol{p}(i) = \boldsymbol{h}_i{}^T(\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{b}) + (1-\epsilon)u_i \qquad \text{and} \qquad \boldsymbol{k}(i) = \boldsymbol{h}_i{}^T\boldsymbol{H}\partial\boldsymbol{x} - u_i$$

- Obtain the step size δ^* using: $\delta^* = \min(\delta^+, \delta^-)$

  $$\delta^+ = \min_{i \in I^c}{}^+ \left\{\frac{\lambda - \boldsymbol{p}(i)}{\boldsymbol{k}(i)}, \frac{-\lambda - \boldsymbol{p}(i)}{\boldsymbol{k}(i)}\right\} \qquad \text{and} \qquad \delta^- = \min_{i \in I}{}^+ \left\{\frac{-\boldsymbol{x}^*(i)}{\partial\boldsymbol{x}(i)}\right\}$$

  If $\delta^* = \delta^+$, then $I = I \cup i$, else if $\delta^* = \delta^-$, then $I = I - i$.

- If $\epsilon + \delta^* > 1$, then $\delta^* = 1 - \epsilon$, $\boldsymbol{x}^* = \boldsymbol{x}^* + \delta^*\partial\boldsymbol{x}$, break.

- Update the solution matrix $\boldsymbol{x}$ using: $\boldsymbol{x}^* = \boldsymbol{x}^* + \delta^*\partial\boldsymbol{x}$
- Update the homotopy parameter ϵ using: $\epsilon = \epsilon + \delta^*$

Until $\epsilon = 1$

4.3 Dynamic updating for the K-LARS homotopy algorithm

4.3.1 K-LARS Homotopy algorithm

In Section 3.4, the Kronecker Least angle regression (K-LARS) algorithm was described to solve this matrix *basis pursuit* problem (3.3) that deals with the multichannel measurements $L \times N$ matrix $\boldsymbol{Y}$, and multicomponent unknown signals $L \times M$ matrix $\boldsymbol{X}$.

The K-LARS algorithm used the properties of Kronecker products to extend the standard LARS algorithm [53] to solve the matrix *basis pursuit* in (3.3) without vectorizing it, i.e., without constructing or processing very large arrays.

K-LARS could be shown as a homotopy algorithm [64, 68] that obtains an optimal solution $\widetilde{\boldsymbol{X}}_{\boldsymbol{\lambda}}$ for every critical value of λ. In every homotopy step, the update direction $\boldsymbol{D}$ and the step size $\delta^*$for moving the solution to a lower value of λ can be easily obtained using certain optimality conditions, which can be derived using the subdifferential of the objective in [63]. At any value of λ , the solution $\boldsymbol{X}^*$ for (3.3) must satisfy the following optimality conditions [67]:

$$\begin{aligned} (\boldsymbol{X}^*\boldsymbol{A}-\boldsymbol{Y})\boldsymbol{A}^T &= -\lambda\boldsymbol{Q} \qquad for\ all\ \ \boldsymbol{Z}(i,j)=1 \\ |(\boldsymbol{X}^*\boldsymbol{A}-\boldsymbol{Y})\boldsymbol{A}^T| &\leq \lambda \qquad for\ all\ \boldsymbol{Z}(i,j)=0 \end{aligned} \tag{4.19}$$

where $\boldsymbol{Z}$ is an $L \times M$ matrix wherein the values of one correspond to the support of the active columns and the values of zero correspond to the inactive columns [71], and $\boldsymbol{Q}$ is the $L \times M$ sign sequence matrix of $\widehat{\boldsymbol{X}}$.

K-LARS starts with a very high value of λ, active set $L \times M$ matrix $\boldsymbol{Z}$ with zero entries, and a zero-matrix for $\widetilde{\boldsymbol{X}}_{\boldsymbol{\lambda}}$. As λ is decreased, the optimal solution $\widetilde{\boldsymbol{X}}_{\lambda}$ is moved in the updated direction $\boldsymbol{D}$, until one constraint of the two optimality conditions in (4.19) is violated. The optimality conditions in (4.19) can be described as $L \times M$ constraints that the solution $\boldsymbol{X}^*$ needs to satisfy with equality in the active columns, and inequality in the inactive columns. The violation of these constraints will only be at the different values of λ when the support changes wherein a value in the matrix $\boldsymbol{Z}$ changes from 0 to 1, which indicates that one column has to be added to the support. Conversely, when a value in the matrix $\boldsymbol{Z}$ changes from 1 to 0, it is an indication that one column has to be removed from the support or that one of the nonzero elements in the sources column $\boldsymbol{X}^*$ has shrunk to zero.

As we reduce λ to $\lambda - \delta$, for a small value of δ will cause the solution to move in a direction $\boldsymbol{D}$. In order to maintain optimality, this solution must obey:

$$\begin{aligned} &\left(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}\right)\boldsymbol{A}^T + \delta\boldsymbol{D}\boldsymbol{A}\boldsymbol{A}^T = -(\lambda - \delta)\boldsymbol{Q} && \text{for all } \boldsymbol{Z}(i,j) = 1 \\ &\left|\left(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}\right)\boldsymbol{A}^T + \delta\boldsymbol{D}\boldsymbol{A}\boldsymbol{A}^T\right| \le (\lambda - \delta) && \text{for all } \boldsymbol{Z}(i,j) = 0 \end{aligned} \tag{4.20}$$

The solution $\boldsymbol{X}^*$ moves in direction $\boldsymbol{D}$ until one of the two constraints in (4.22) is violated. The smallest step size δ that will cause one of these changes in the support of the active set matrix $\boldsymbol{Z}$ can be easily computed as $\delta^* = \min\,(\delta^+, \delta^-)$, where:

$$\delta^+ = \min{}^+\left\{\frac{\lambda - \boldsymbol{PP}(i,j)}{1 - \boldsymbol{KK}(i,j)}, \frac{\lambda + \boldsymbol{PP}(i,j)}{1 + \boldsymbol{KK}(i,j)}\right\}$$

$$\delta^- = \min{}^+\left\{\frac{-\boldsymbol{X}^*(i,j)}{\boldsymbol{D}(i,j)}\right\} \tag{4.21}$$

where $\boldsymbol{PP} = \left(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}\right)\boldsymbol{A}^T$, $\boldsymbol{KK} = \boldsymbol{D}\boldsymbol{A}\boldsymbol{A}^T$. Note that only positive components within every choice of (i,j) are considered when the minimum is taken.

At every step of this homotopy algorithm, the new critical value of $\boldsymbol{\lambda}$ would become $\lambda - \delta^*$, the new optimal solution $\widetilde{\boldsymbol{X}}_\lambda$ would become $\widetilde{\boldsymbol{X}}_\lambda + \delta^*\boldsymbol{D}$, and both active set matrix $\boldsymbol{Z}$ and sign sequence matrix $\boldsymbol{Q}$ would be updated accordingly. This procedure is repeated until λ has been lowered to a required minimum value [67].

4.3.2 Homotopy K-LARS dynamic updating

In this section, we describe an extension of the homotopy algorithm for updating l_1 in order to solve the optimization problem (3.3). We will call this extension "homotopy K-LARS dynamic updating." This modification will allow solutions for the matrix *basis pursuit* optimization problem (3.3) to be updated dynamically by using a warm-start, which is a near estimate, instead of starting from zero. This method is similar to the one described in Section 4.2.

Instead of solving (3.3) from scratch, we assume that we have a sparse matrix $\widehat{\boldsymbol{X}}$ that is close to the original solution of (3.3) and a sign sequence $\widehat{\boldsymbol{Z}}$. This homotopy Kronecker method continuously transforms the optimization problem (3.3) into a related problem for which the solution is either available or easy to compute. Starting from an available solution, a series of simple problems are solved along the so-called homotopy path towards the original problem's final solution [28, 64, 69]. The progression along the homotopy path is controlled by the *homotopy parameter* ϵ, which usually varies between 0 and 1 and corresponds to the two end points of the homotopy path exactly as in section 4.2.

We treat the given warm-start vector $\widehat{\boldsymbol{X}}$ as a starting point and solve the following optimization problem by changing ϵ from 0 to 1:

$$\min_{\boldsymbol{X}} \frac{1}{2} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|_2^2 + \lambda \|\boldsymbol{X}\|_1 + (1-\epsilon)\boldsymbol{U}^T\boldsymbol{X} \tag{4.22}$$

We define $\boldsymbol{U}$ as:

$$\boldsymbol{U} = -\lambda\widehat{\boldsymbol{Q}} - \boldsymbol{A}^T(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}) \tag{4.23}$$

where $\widehat{\boldsymbol{X}}$ is the optimal solution of (4.23) at $\epsilon = 0$ and $\widehat{\boldsymbol{Q}}$ is the $L \times M$ matrix representing the sign sequence of $\widehat{\boldsymbol{X}}$ and. As ϵ changes from 0 to 1, the optimization problem in (4.23) gradually becomes the one in (3.3), and the solution of (4.23) follows a piece-wise linear homotopy path from $\widehat{\boldsymbol{X}}$ towards the solution of (3.3).

The optimality conditions for (4.23) can be derived by setting the subdifferential of its objective function to zero. Similarly to section 4.2, the solution $\boldsymbol{X}^*$ for (4.23) must satisfy the following optimality conditions:

$$\begin{aligned} &\left(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}\right)\boldsymbol{A}^T + (1-\epsilon)\boldsymbol{U} = -\lambda\boldsymbol{Q} && for\ all\ \boldsymbol{Z}(i,j) = 1 \\ &\left|\left(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y}\right)\boldsymbol{A}^T + (1-\epsilon)\boldsymbol{U}\right| \leq \lambda && for\ all\ \boldsymbol{Z}(i,j) = 0 \end{aligned} \tag{4.24}$$

where $\boldsymbol{Z}$ is an $L \times M$ matrix wherein the values of one correspond to the support of the active columns and the values of zero correspond to the inactive columns (as in Section 3.4), and $\boldsymbol{Q}$ is the $L \times M$ sign sequence matrix of $\widehat{\boldsymbol{X}}$.

The violation of these constraints in (4.24) will occur at the critical values of ϵ wherein a value in the matrix $\boldsymbol{Z}$ changes from 0 to 1, which indicates that one column has to be added to the support. Conversely, when a value in the matrix $\boldsymbol{Z}$ changes from 1 to 0, it is an indication that one column has to be removed from the support or that one of the nonzero elements in the sources column $\boldsymbol{X}^*$ has shrunk to zero.

For every homotopy step, we jump from one critical value of ϵ to the next $\epsilon + \delta$ and update the support of the solution until ϵ is equal to 1. As we increase ϵ by a small value δ, the solution moves in a direction $\partial\boldsymbol{x}$. This solution must obey the following optimality conditions:

$$\begin{aligned}&\big(\hat{\boldsymbol{X}}\boldsymbol{A}-\boldsymbol{Y}\big)\boldsymbol{A}^T+(1-\epsilon)\boldsymbol{U}+\delta(\boldsymbol{\partial X}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{U})=-\lambda\boldsymbol{Z} && for\ all\ \boldsymbol{Z}(i,j)=1\\ &\left|\big(\hat{\boldsymbol{X}}\boldsymbol{A}-\boldsymbol{Y}\big)\boldsymbol{A}^T+(1-\epsilon)\boldsymbol{U}+\delta(\boldsymbol{\partial X}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{U})\right|\le\lambda && for\ all\ \boldsymbol{Z}(i,j)=0\end{aligned}\qquad(4.25)$$

The update direction $\boldsymbol{\partial X}$ is an $L\times M$ matrix that keeps the solution optimal as δ changes can be written as:

$$\boldsymbol{\partial X}=\begin{cases}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{U} & for\ \boldsymbol{Z}(i,j)=1\\ 0 & for\ \boldsymbol{Z}(i,j)=0\end{cases}\qquad(4.26)$$

The solution $\boldsymbol{X}^*$ moves in direction $\boldsymbol{\partial X}$ until one of the two constraints in (4.25) is violated. The smallest step-size that either causes one of these violations or a change in the support of the active set $\boldsymbol{Z}$ can be easily computed as $\delta^*=\min(\delta^+,\delta^-)$, where:

$$\delta^+=\min{}^+\left\{\frac{\lambda-\boldsymbol{PP}(i,j)}{\boldsymbol{KK}(i,j)},\frac{-\lambda-\boldsymbol{PP}(i,j)}{\boldsymbol{KK}(i,j)}\right\}$$

$$\delta^-=\min{}^+\left\{\frac{-\boldsymbol{X}^*(i,j)}{\boldsymbol{\partial X}(i,j)}\right\}\qquad(4.27)$$

where $\boldsymbol{PP}=\big(\hat{\boldsymbol{X}}\boldsymbol{A}-\boldsymbol{Y}\big)\boldsymbol{A}^T+(1-\epsilon)\boldsymbol{U}$, $\boldsymbol{KK}=\boldsymbol{\partial X}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{U}$. Note that only positive components within every choice of (i,j) are considered when the minimum is taken.

δ^+ is the smallest step size that will cause the inactive set constraint to be violated and allow a value in the matrix $\boldsymbol{Z}$ to change from 0 to 1, thus indicating that one column has to be added to the support and that $\boldsymbol{Q}(i,j)$ should be opposite of its sign. δ^- is the smallest step size that will cause the active set constraint to be violated and allow a value in the matrix $\boldsymbol{Z}$ to change from 1 to 0, thus indicating that one column must be removed from the support or that one of the nonzero elements in the sources columns $\boldsymbol{X}^*$has shrunk to zero. The new critical value of ϵ becomes $\epsilon+\delta^*$, the new signal estimate $\boldsymbol{X}^*$

becomes $\boldsymbol{X}^{*} + \delta^{*}\boldsymbol{\partial X}$, and the support of the active set $\boldsymbol{Z}$ and the sign sequence $\boldsymbol{Q}$ are updated accordingly. This procedure is repeated until $\epsilon = 1$.

The Kronecker homotopy algorithm for dynamically updating the solution to the *l₁*-norm minimization, matrix *basis pursuit* minimization, problem using a nonzero estimate (warm-start) is summarized in Table 4.2.

Table 4.2. The Kronecker homotopy algorithm for update the solution of the l_1 problem for a multichannel and linear system.

- Given: The measurement matrix $\boldsymbol{Y}$, the coefficient matrix $\boldsymbol{A}$, the warm-start solution $\boldsymbol{X}^{*}$, and λ.
- Initialization steps: The columns of the $\boldsymbol{A}^{T}$ matrix must be normalized to unit l_2-norm, $\epsilon = 0$. The active set $\boldsymbol{Z}$ is initialized using the $\mathrm{L} \times \mathrm{M}$ zero matrix.
- Repeat
  - The $\mathbf{U}$ matrix is obtained via $\boldsymbol{U} = -\lambda\widehat{\boldsymbol{Q}} - \boldsymbol{A}^{T}(\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y})$

    where $\widehat{\boldsymbol{Q}}$ is the sign sequence of $\widehat{\boldsymbol{X}}$.
  - The updated direction vector $\boldsymbol{\partial X}$ is obtained by:

$$\boldsymbol{\partial X} = \begin{cases} (\boldsymbol{A}^{T}\boldsymbol{A})^{-1}\boldsymbol{U} & for\ \boldsymbol{Z}(i,j) = 1 \\ 0 & for\ \boldsymbol{Z}(i,j) = 0 \end{cases}$$

  - The $\boldsymbol{p}$ and $\boldsymbol{d}$ matrices for the inactive variables are obtained using:

$$\boldsymbol{PP} = (\widehat{\boldsymbol{X}}\boldsymbol{A} - \boldsymbol{Y})\boldsymbol{A}^{T} + (1-\epsilon)\boldsymbol{U} \qquad \text{and} \qquad \boldsymbol{KK} = \boldsymbol{\partial X}\boldsymbol{A}\boldsymbol{A}^{T} - \boldsymbol{U}$$

  - The step size δ^{*} is obtained by: $\delta^{*} = \min\,(\delta^{+}, \delta^{-})$

$$\delta^{+} = \min{}^{+}\left\{\frac{\lambda - \boldsymbol{PP}(i,j)}{\boldsymbol{KK}(i,j)}, \frac{-\lambda - \boldsymbol{PP}(i,j)}{\boldsymbol{KK}(i,j)}\right\}$$

$$\delta^{-} = \min{}^{+}\left\{\frac{-\boldsymbol{X}^{*}(i,j)}{\partial \boldsymbol{X}(i,j)}\right\}$$

```
If $\delta^* = \delta^+$, then $\boldsymbol{Z}(i,j) = 1$, else if $\delta^* = \delta^-$, then $\boldsymbol{Z}(i,j) = 0$.

    • If $\epsilon + \delta^* > 1$, then $\delta^* = 1 - \epsilon$, $\boldsymbol{X}^* = \boldsymbol{X}^* + \delta^* \boldsymbol{\partial X}$, break.
    • Update the solution matrix $\boldsymbol{x}$ by:    $\boldsymbol{X}^* = \boldsymbol{X}^* + \delta^* \boldsymbol{\partial X}$
    • Update the homotopy parameter $\epsilon$ using:    $\epsilon = \epsilon + \delta^*$

Until $\epsilon = 1$
```

4.4 Chapter summary

In this chapter, we described how the Least Angle Regression (LARS) algorithm could be used as a homotopy algorithm that shows the optimality condition of the *basis pursuit* problem. We then described how the homotopy algorithm could dynamically update the solutions for the *basis pursuit* problem by using a warm-start solution instead of a zero initial solution. Finally, we presented the homotopy Kronecker algorithm as an extension of the homotopy algorithm that can handle multichannel signals to solve the matrix *basis pursuit* minimization problem in a more computationally efficient way, thereby faster, than K-LARS.

Chapter 5
Unsupervised Unmixing of Hyperspectral Images

In this chapter and according to our discussions in Chapters 3 and 4 by considering the spectral pixels, or the given measurements, as multichannel data, which, according to the Linear Mixture Model (LMM), are superpositions of the multicomponent signals, or the endmembers. To this end, we will blindly unmix the spectral pixels using the matrix *basis pursuit* optimization problem as a sparsity-based unsupervised algorithm. The matrix *basis pursuit* unmixing problem is solved using the Kronecker LARS algorithm discussed in Chapter 3 and the Kronecker homotopy algorithm discussed in Chapter 4 to successfully achieve spectral unmixing of both synthetic and *AVIRIS* hyperspectral imaging data. We also compare our results to ones obtained using an earlier *basis pursuit*-based spectral unmixing algorithm, Generalized Morphological Component Analysis (GMCA). We show that these two results are similar, albeit our results were obtained without trial and error, or arbitrary choices, in specifying the regularization parameter.

5.1 Sparsity-based spectral unmixing algorithm

For the LMM in (2.3), spectral unmixing could be formulated as a matrix least-squares based multichannel data fitting problem,

$$\min_{\boldsymbol{S},\boldsymbol{A}} \tfrac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}\boldsymbol{A}\|_2^2 \tag{5.1}$$

This problem is equivalent to decomposing the multichannel data matrix $\boldsymbol{Y}$ into the sum of rank-1 matrices. Typically, such decomposition would not be unique, so additional constraints would be needed to obtain a unique solution. In sparsity-based spectral unmixing such constraints could be added as a penalty term on the *l₁*-norm of $\boldsymbol{S}$

$$\min_{\boldsymbol{S},\boldsymbol{A}} \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}\boldsymbol{A}\|_2^2+\lambda\|\boldsymbol{S}\|_1 \tag{5.2}$$

5.1.1 Representation to promote endmember sparsity

To facilitate solving the minimization problem (5.2), we could transform it into a different domain, represented by, e.g., an $L \times L$ dictionary $\boldsymbol{\Phi}$, where the transformed endmembers $\boldsymbol{\alpha}=\boldsymbol{\Phi}^T\boldsymbol{S}$ would likely be more sparse. The optimization problem (5.2) would then become

$$\min_{\boldsymbol{\alpha},\mathbf{A}} \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\alpha}\boldsymbol{A}\|_2^2+\lambda\|\boldsymbol{\alpha}\|_1 \tag{5.3}$$

where $\boldsymbol{\beta}=\boldsymbol{\Phi}^T\boldsymbol{Y}$ is an $L \times N$ matrix whose columns represent coefficients of each transformed pixel $\boldsymbol{y}_i$. After solving the minimization problem (5.3), the original endmembers $\boldsymbol{S}$ could be restored as, $\boldsymbol{S}=\widetilde{\boldsymbol{\Phi}}\boldsymbol{\alpha}$, where $\widetilde{\boldsymbol{\Phi}}$ represents the dual of dictionary $\boldsymbol{\Phi}$ [72].

5.1.2 Solving the spectral unmixing problem

This minimization problem (5.3) is nonconvex, but it could be solved using a coordinate descent method [73]. Therefore, it could be divided into two convex problems, $\min_{\boldsymbol{\alpha}}(\cdot)|_{\boldsymbol{A}}$ and $\min_{\boldsymbol{A}}(\cdot)|_{\boldsymbol{\alpha}}$, and solved alternately until convergence to optimal values for both $\boldsymbol{\alpha}$ and $\boldsymbol{A}$.

The first of these two convex problems,

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\alpha}\boldsymbol{A}\|_2^2+\lambda\|\boldsymbol{\alpha}\|_1 \tag{5.4}$$

obtains an optimal value of $\boldsymbol{\alpha}$ by solving a matrix least-squares minimization problem with an *l₁*-norm sparsity constraint on $\boldsymbol{\alpha}$, i.e., a matrix *basis pursuit* problem [42].

The second convex problem

$$\min_{\boldsymbol{A}} \tfrac{1}{2}\|\boldsymbol{\beta} - \boldsymbol{\alpha}\boldsymbol{A}\|_2^2 \tag{5.5}$$

obtains an optimal value of $\boldsymbol{A}$ by solving a matrix least-squares minimization problem under the constraints given by (2.2) [74]. Solving this minimization problem (5.5) in its matrix form is straight forward.

5.1.3 Vectorizing of HSI multichannel data

As described in section 3.4, the matrix *basis pursuit* in (5.4), could be written in a vectorized form as

$$\min_{\alpha} \tfrac{1}{2}\|(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)\mathrm{vec}(\boldsymbol{\alpha}) - \mathrm{vec}(\boldsymbol{\beta})\|_2^2 + \lambda\|\mathrm{vec}(\boldsymbol{\alpha})\|_1 \tag{5.6}$$

5.1.4 Choice of regularization parameter $\boldsymbol{\lambda}$ value

Different solutions for the vector *basis pursuit* problem (5.6), corresponding to different values of the regularization parameter λ, could be obtained [6]. As shown by the Pareto curve in Figure 5.1, different values of $\boldsymbol{\lambda}$ quantify the trade-off between the sparsity level of the unknown endmembers, $\|\mathrm{vec}(\boldsymbol{\alpha})\|_1$, and the data fitting error resulting from using their estimates, $\frac{1}{2}\|(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)\mathrm{vec}(\boldsymbol{\alpha}) - \mathrm{vec}(\boldsymbol{\beta})\|_2^2$.

From Figure 5.1, we note that when the value of $\boldsymbol{\lambda}$ is high (low), the sparsity of the endmembers would be high (low), i.e., $\|\mathrm{vec}(\boldsymbol{\alpha})\|_1$ would be small (large), and the data fitting error, $\frac{1}{2}\|(\boldsymbol{A}^T \otimes \boldsymbol{I}_L)\mathrm{vec}(\boldsymbol{\alpha}) - \mathrm{vec}(\boldsymbol{\beta})\|_2^2$, would be high (low). Therefore, a traditional way to solve the vector *basis pursuit* problem (5.6)is to solve it many times using different values of λ, and then pick the solution that would successfully solve the spectral unmixing problem. This traditional way, however, is not practical as it could be both time and resource consuming.

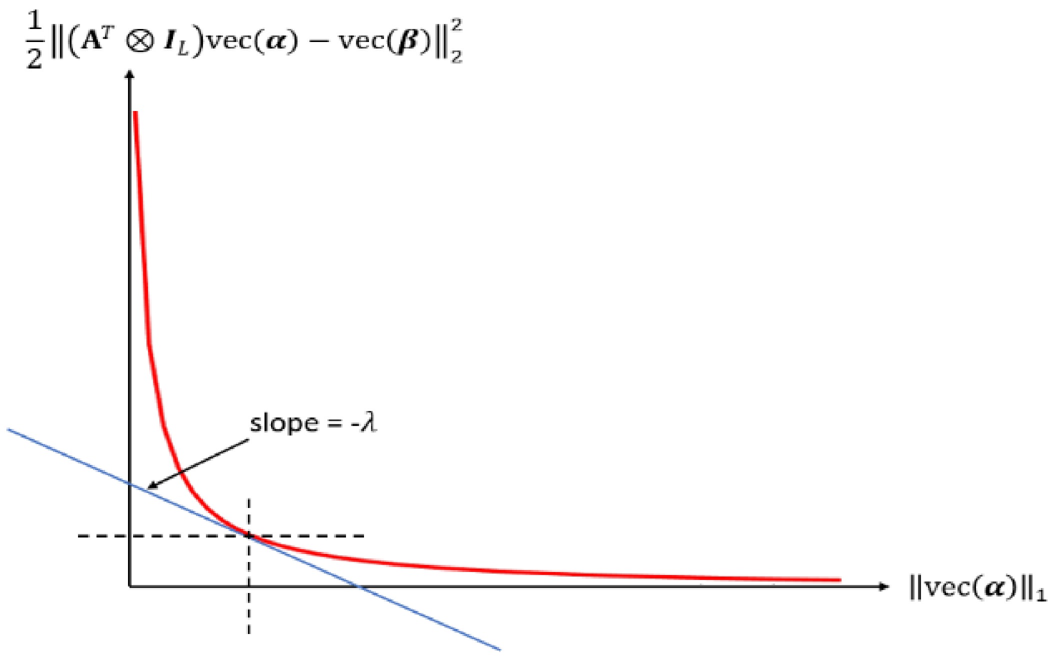


Figure 5.1. Typical Pareto curve for an arbitrary vector *basis pursuit* problem.

5.2 Spectral unmixing using the K-LARS algorithm

To demonstrate the validity of our K-LARS based spectral unmixing method, we applied it on two HSI datasets. The first is a synthetic dataset with spectral pixels that are linear mixtures of endmembers, from the *ASTER* spectral library [75], with uniformly random abundances. The second is an actual HSI dataset from *NASA's AVIRIS* website [9]. We also compare our results to ones obtained using GMCA. We show that these two results are similar, albeit our results were obtained without arbitrary choices related to specifying the regularization parameter.

5.2.1 Preprocessing of HSI multichannel data

To ensure that the given spectral pixels have a high signal to noise ratio (SNR), we estimate the noise level in every pixel using a wavelet (*Symmlet*) filter and use this noise level to obtain the SNR of every pixel. The SNR is an indication of how much of cube energy is considered as reflectances of given endmembers and not noise. This means that if the SNR is high, this is an indication of the given spectral pixels represents reflectances of materials that exist in the scene acquired, while if the SNR is low, this is an indication of the given spectral pixels are mainly noise and not reflectances. Assuming the presence of white noise in the HSI data, then a data denoising step would be necessary before solving the optimization problem (5.3). This is because white noise could not be sparsely represented in any dictionary [76]. Therefore, we denoised each raw spectral pixel using a wavelet (*Symmlet*) thresholding method with a universal threshold [77].

It's required to estimate the number of endmembers, M, present in the scene before starting to unmix the given hyperspectral data cube. This could be done using the Principal Component Analysis (PCA) [78] that gives the minimum number of vertices, $M-1$ subspace, that could be used to span a high percentage value of the given cube energy. These vertices are mainly the major components or the major eigenvectors of the given data cube that have a predetermined high percentage of all eigenvalues.

As discussed in section 5.1.1, to promote endmember sparsity, we transformed the used HSI datasets to the wavelet domain, where the transformed endmembers $\boldsymbol{\alpha}$ would likely be more sparse than in the original measurement domain. We used the *Coiflet 5* wavelet because both its scaling and wavelet functions have vanishing moments, thereby significantly promoting sparsity of piecewise smooth functions, e.g., endmember spectra, as will be discussed in Chapter 6.

5.2.2 Applying K-LARS to HSI spectral unmixing

As discussed in Section 5.1.2, the sparsity-based spectral unmixing problem, problem (5.3), could be solved by an iterative and alternate estimation of $\boldsymbol{\alpha}$, problem (5.4), and $\boldsymbol{A}$, problem (5.5). Initially, $\boldsymbol{\alpha}$ is set to zero, and $\boldsymbol{A}$ is set to random abundance values. At each iteration, we fixed $\boldsymbol{A}$ and estimated $\boldsymbol{\alpha}$ using our K-LARS algorithm, then we fixed $\boldsymbol{\alpha}$ and estimated $\boldsymbol{A}$ by solving a simple constrained linear least squares problem.

The sparsity levels of any of the estimates of $\boldsymbol{\alpha}$ are crucial to the accuracy of our final spectral unmixing results. Therefore, in our first spectral unmixing iteration, we terminated the K-LARS algorithm when the data fitting error corresponding to the sparse solution was within 0.5% of the data fitting error corresponding to the full least-squares solution. We then used this sparsity level of $\boldsymbol{\alpha}$ as the K-LARS termination condition for all subsequent spectral unmixing iterations.

5.2.3 Spectral unmixing of synthetic HSI multichannel data

We generated a synthetic HSI dataset of 3000 spectral pixels. These pixels are linear mixtures of three arbitrary chosen materials' spectra that were chosen from the *ASTER* spectral library [75], with uniformly random abundances. After generating this cube, additive Gaussian noise was added to these mixed spectral pixels to obtain a signal-to-noise ratio (SNR) of a given value (dB). In this step, the noise variance is calculated from the obtained synthetic cube power, and the required SNR value, Afterwards, noise with this calculated variance is added to the synthetic spectral pixels.

We generated two synthetic noisy HSI data cubes. The first was generated using spectra of grass, concrete, and asphalt, with $SNR = 30$ dB. And the second was generated using spectra of aluminum, copper, and steel, with $SNR = 20$ dB. Therefore, in this dataset, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 3000$, and the number of wavelengths, $L = 491$ (0.42 μm to 14 μm). Figure 5.2 (a) – (c) show the spectra of grass, concrete and asphalt, whose different linear mixtures make up different spectral pixels in this dataset.

For the 1st synthetic data cube, Figure 5.2 (d) – (f) show the estimated spectra obtained by applying the GMCA based spectral unmixing algorithm to this dataset. To obtain these GMCA based results we used trial and error to obtain both initial and final values of the regularization parameter, $\lambda_{initial} = 15$, $\lambda_{final} = 4$, and we arbitrarily chose a linear schedule for its reduction with each coordinate descent iteration. The correlation coefficients between the exact spectra and their GMCA estimated counterparts are 0.9673, 0.9942, and 0.9583, respectively.

Figure 5.2 (g) – (i) show the estimated spectra obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. The correlation coefficients between the exact spectra of grass, concrete, asphalt, and their estimated counterparts are 0.9970, 0.9993, and 0.9727, respectively.

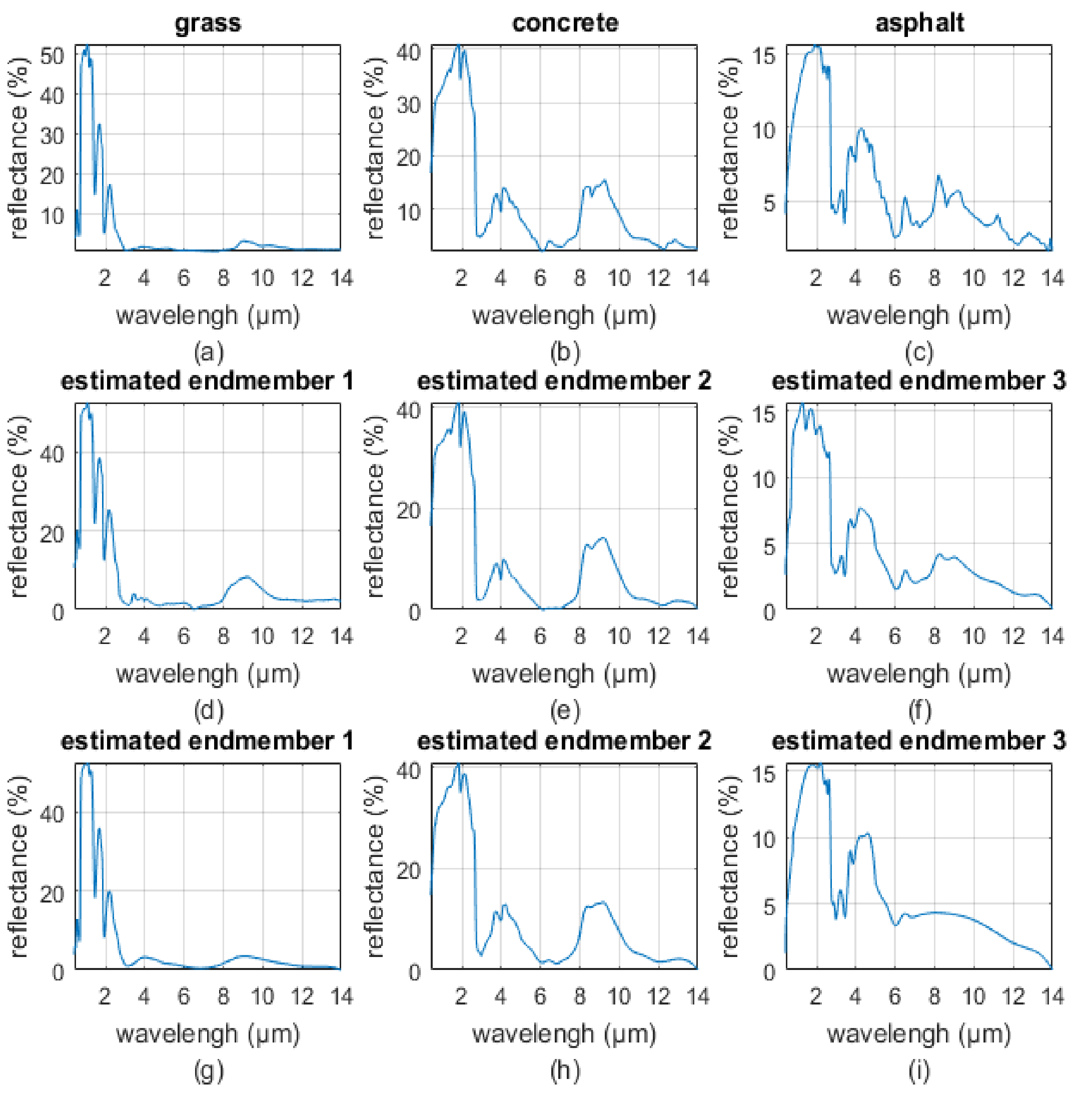


Figure 5.2. (a), (b) and (c) Exact spectra of grass, concrete, and asphalt, respectively. (d), (e) and (f) Estimated spectra obtained using the GMCA based spectral unmixing algorithm. (g), (h) and (i) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm.

For the 2nd synthetic data cube, Figure 5.3 (d) – (f) show the estimated spectra obtained by applying the GMCA based spectral unmixing algorithm to this dataset. To obtain these GMCA based results we used trial and error to obtain both initial and final values of the regularization parameter, $\lambda_{initial} = 15$, $\lambda_{final} = 4$, and we arbitrarily chose a linear schedule for its reduction with each coordinate descent iteration. The correlation coefficients between the exact spectra and their GMCA estimated counterparts are 0.9894, 0.7477, and 0.9856, respectively.

Figure 5.3 (g) – (i) show the estimated spectra obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. The correlation coefficients between the exact spectra of grass, concrete, asphalt, and their estimated counterparts are 0.9991, 0.9371, and 0.9281, respectively.

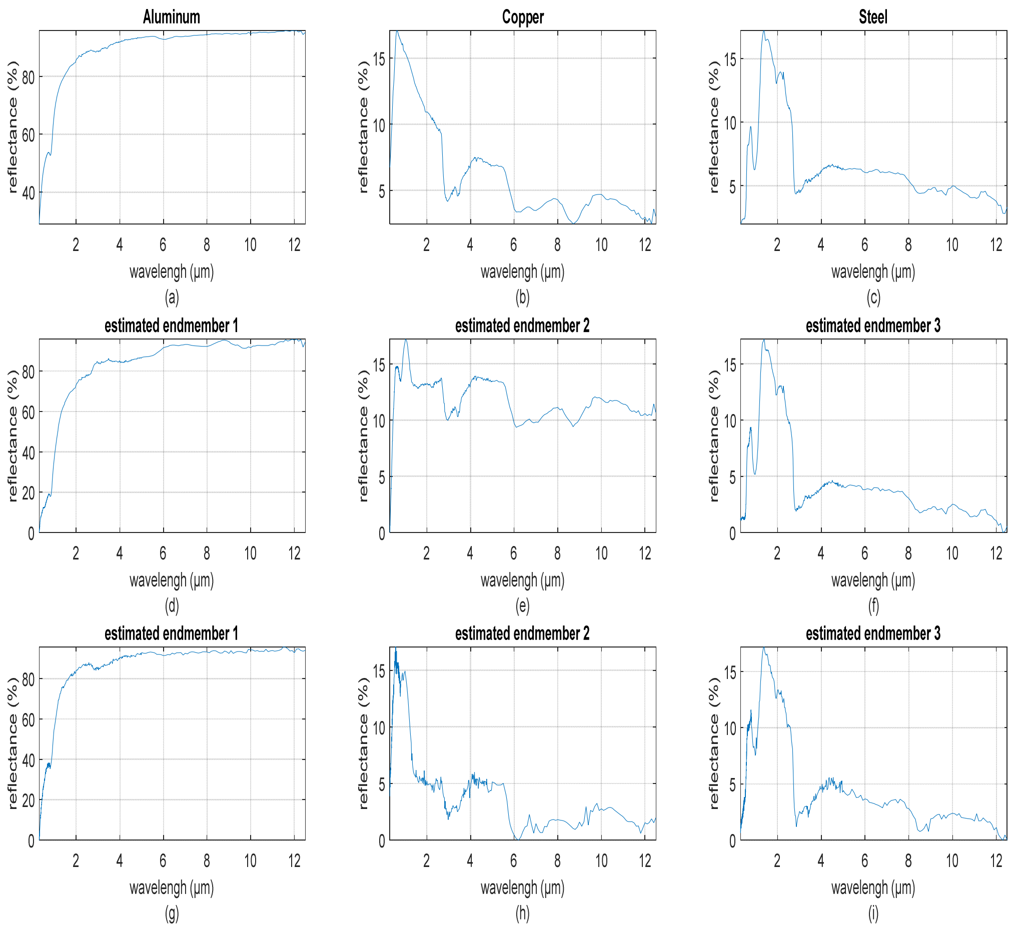


Figure 5.3. (a), (b) and (c) Exact spectra of aluminum, copper, and steel, respectively. (d), (e) and (f) Estimated spectra obtained using the GMCA based spectral unmixing algorithm. (g), (h) and (i) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm.

These quantitative results demonstrate the validity of our new K-LARS algorithm and its successful application to HSI spectral unmixing. Compared to spectral unmixing results obtained by GMCA, our results obtained by K-LARS are very close, albeit they were obtained without any trial and error, or arbitrary choices, in specifying the regularization parameter.

5.2.4 Spectral unmixing of AVIRIS HSI multichannel data

We downloaded an actual HSI data cube from *NASA's AVIRIS* website [9]. This dataset is comprised of images obtained using 224 wavelengths (0.365 μm to 2.497 μm). This cube has dimensions of (651 x 511 pixels) that shows a large island in the ocean, as shown in Figure 5.4. The scene of the hyperspectral data cube showing a large island in the ocean (single wavelength image).

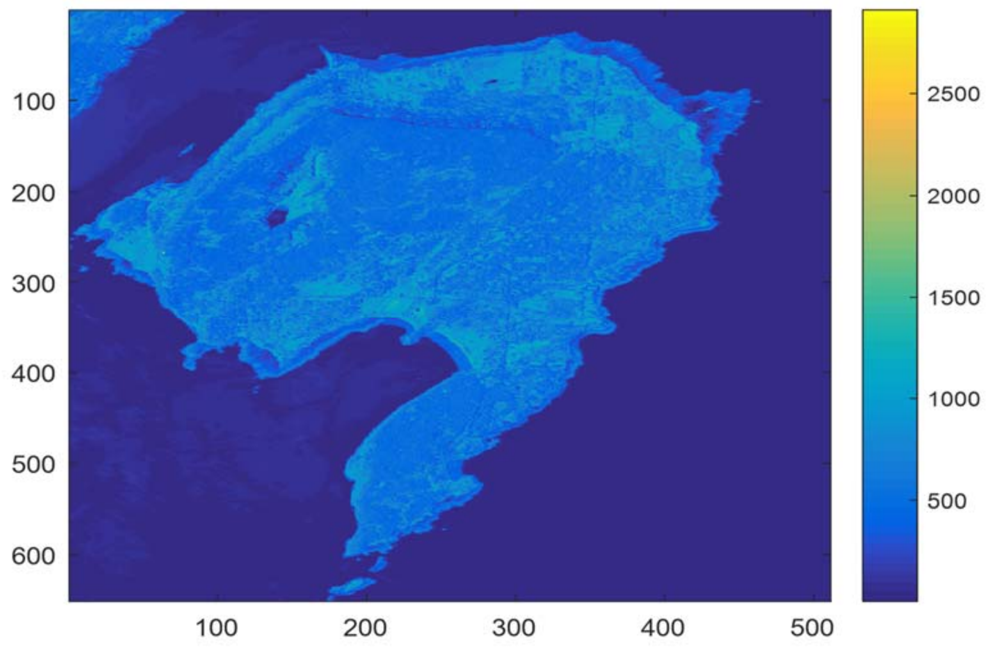


Figure 5.4. The scene of the hyperspectral data cube showing a large island in the ocean (single wavelength image).

We estimated the SNR of every spectral pixel in the cube as discussed in section 5.2.1 and obtained an average pixels SNR equal 67.47 dB, and a standard deviation of pixels SNR equal 4.56 dB which indicates extremely high SNR that means that the given spectral pixels represent reflectances of materials exists in the acquired scene or endmembers and not noise.

To determine the number of endmembers that exist in the scene, we apply the Principal Component Analysis (PCA) [78] to the given data cube and obtain 224 eigenvalues and their corresponding 224 eigenvectors. Then we keep 99% of the given data energy by selecting the major eigenvectors that have total eigenvalues larger than 0.99 of the sum of the 224 eigenvalues. Choosing 99% of the given hyperspectral cube energy is a reasonable number to justify that our results explain 99% of the data. This also means that we ignore minor endmembers that exist in the cube. We note that 99% of the cube energy could be represented using 3 major eigenvectors corresponding to 3 extremely high eigenvalues; 1.6, 0.08, and 0.0173 (all multiplied by 10^7). This means that theoretically, the number of endmembers $M = 4$, the number of spectral pixels, $N = 332661$, and the number of wavelengths, $L = 224$.

Applying the K-LARS algorithm to this hyperspectral cube gives us numerically unstable unmixing results as the abundances have extremely large and extremely low values reach 10^{15} and 10^{-15}, respectively. This means that the number of endmembers should be reduced to $M = 3$, i.e., one endmember is very rare and should be neglected.

Applying the K-LARS algorithm to the given hyperspectral cube for $M = 3$ gives us numerically stable shown in Figure 5.5.

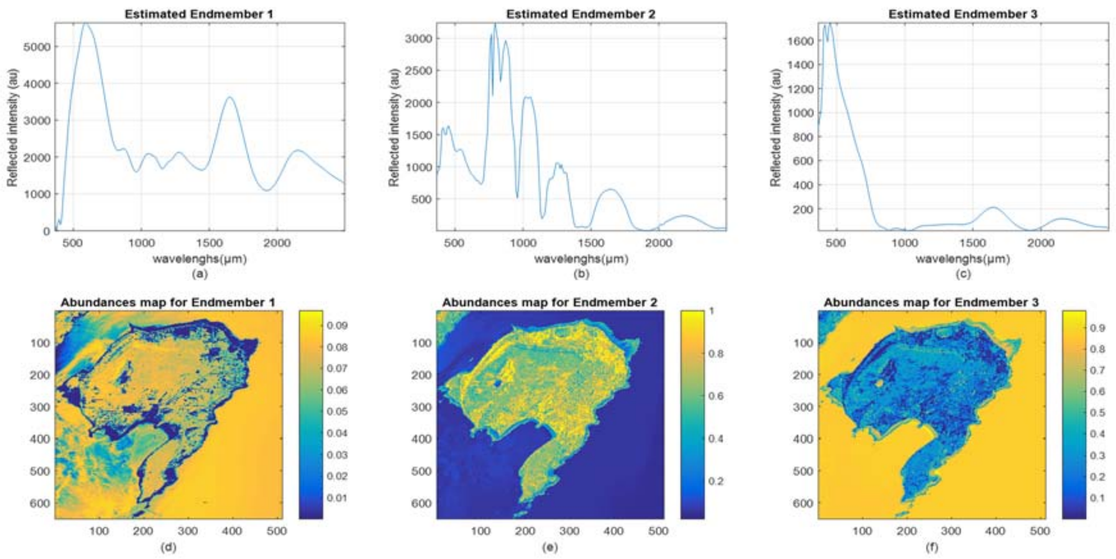


Figure 5.5. (a), (b) and (c) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm. (d), (e) and (f) The estimated abundances maps for the 3 endmembers.

Figure 5.5 (a) – (c) show the estimated spectra of these 3 endmembers obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. Figure 5.5 (d) – (f) shows the estimated abundances maps for the three endmembers individually. These estimated abundances proportions are numerically stable as all these values are under the constraints given by (2.2).

Spectral unmixing of another AVIRIS hyperspectral cube

We downloaded another actual HSI dataset called the Moffet Field cube from *NASA's AVIRIS* website [9]. This dataset is comprised of images obtained using 224 wavelengths (0.365 μm to 2.497 μm). We selected an arbitrary region (70 x 60 pixels) as our region of interest, and then applied the PCA to estimate the number of endmembers, M, as done with the previous hyperspectral cube. Keep 99% of the given data energy in PCA shows that this scene has 3 endmembers. Therefore, in the dataset corresponding to our selected region, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 4200$, and the number of wavelengths, $L = 224$.

To validate spectral unmixing results obtained by applying our K-LARS based unmixing algorithm to this dataset, we need to know its ground truth, i.e., spectra of endmembers present in our selected region. For this purpose, we used N-FINDR, a software package that uses a geometrical approach to find approximately pure spectral pixels, i.e., spectral pixels comprised of a single endmember only, in the given HSI data. The locations of these approximately pure spectral pixels in our selected region obtained by N-FINDR [20] are shown in Figure 5.6.

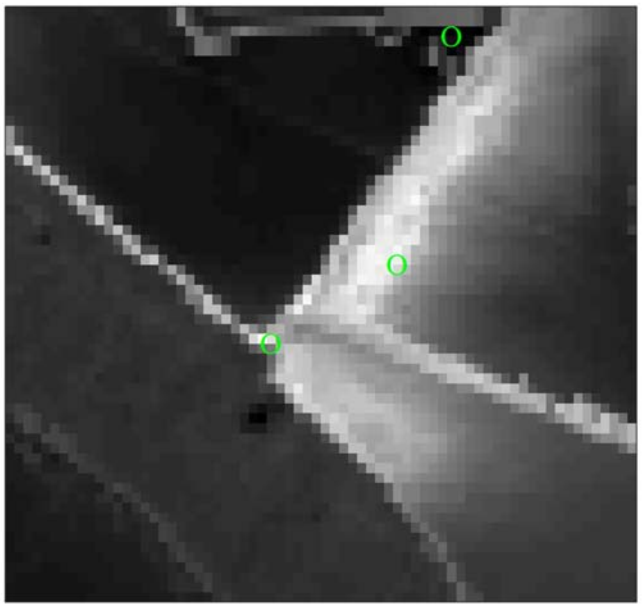


Figure 5.6. Locations in the cube scene (single wavelength image) of approximately pure spectral pixels obtained by N-FINDR.

Figure 5.7 (a) – (c) show the ground truth spectra of the 3 endmembers present in our selected region, and whose different linear mixtures make up different spectral pixels in this dataset. Figure 5.7 (d) – (f) show the estimated spectra of these 3 endmembers obtained by applying the GMCA based spectral unmixing algorithm to this dataset. To obtain these GMCA based results we used trial and error to obtain both initial and final values of the regularization parameter, $\lambda_{initial} = 15$, $\lambda_{final} = 2.8$, and we arbitrarily chose a linear schedule for its reduction with each coordinate descent iteration. The correlations between the ground truth spectra and their corresponding estimated endmember spectra are 0.9923, 0.8514, and 0.8607, respectively. Figure 5.7 (g) – (i) show the estimated spectra of these 3 endmembers obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. The correlations between the ground truth spectra and their corresponding estimated endmember spectra are 0.8358, 0.9892, and 0.8878, respectively. These quantitative results further demonstrate the validity of our new K-LARS algorithm and its successful application to HSI spectral unmixing. Compared to spectral unmixing results obtained by GMCA, our results obtained by K-LARS are very close, albeit they were obtained without any trial and error, or arbitrary choices, in specifying the regularization parameter.

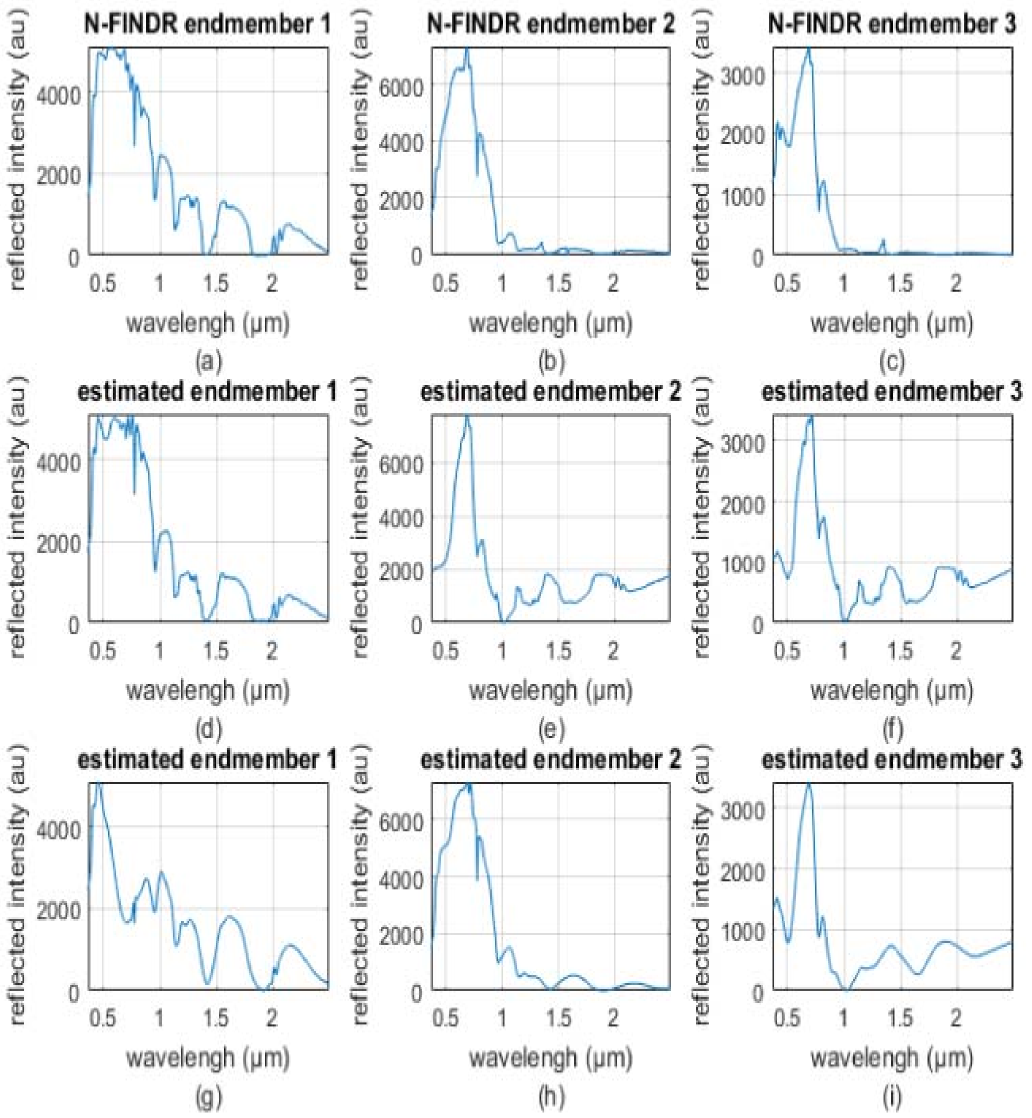
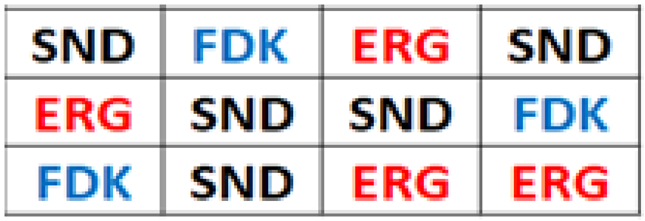


Figure 5.7. (a), (b) and (c) Ground truth spectra of 3 endmembers obtained by N-FINDR. (d), (e) and (f) Estimated spectra obtained using the GMCA based spectral unmixing algorithm. (g), (h) and (i) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm.

5.2.5 Spectral unmixing of sound and infected grains hyperspectral cube

We collect actual hyperspectral data cube in our lab. This cube was used in a project to detect sound wheat grains from infected grains. This cube is comprised of images obtained using 224 wavelengths in the VNIR band (0.4 μm to 1 μm). This data cube has dimensions of (85 x 135 pixels), and it shows grain kernels arranged as shown in Figure 5.4. The scene of the hyperspectral data cube showing a large island in the ocean (single wavelength image)., where SND means sound grains kernels, ERG means grain kernels infected with Ergot, and FDK means grain kernels infected with Fusarium.



SND	FDK	ERG	SND
ERG	SND	SND	FDK
FDK	SND	ERG	ERG

Figure 5.8. Arrangement of sound, Ergot infected, and Fusarium infected grain kernels in the imaged scene.

We applied the PCA to estimate the number of endmembers, M, as done before. Keeping 99% of the given data energy in PCA shows that this scene has 3 endmembers. Therefore, for this data cube, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 11475$, and the number of wavelengths, $L = 224$.

To validate spectral unmixing results obtained by applying our K-LARS based unmixing algorithm to this dataset, we need to know its ground truth, i.e., spectra of endmembers present in our selected region. For this purpose, we used N-FINDR as before. The locations of these approximately pure spectral pixels in our selected region obtained by N-FINDR [20] are shown in Figure 5.9.

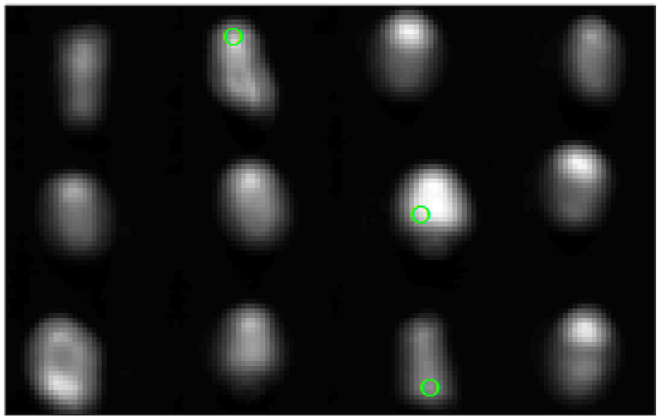


Figure 5.9. Locations in the VNIR cube scene (single wavelength image) of approximately pure spectral pixels obtained by N-FINDR for SND, ERG, and FDK grains.

Figure 5.10 (a) – (c) show the ground truth spectra of the 3 endmembers present in the scene. Figure 5.10 (d) – (f) show the estimated spectra of the 3 endmembers obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. The correlations between the ground truth spectra and their corresponding estimated endmember spectra are 0.9654, 0.9876, and 0.9314, respectively. These quantitative results further demonstrate the validity of our new K-LARS algorithm and its successful application to HSI spectral unmixing.

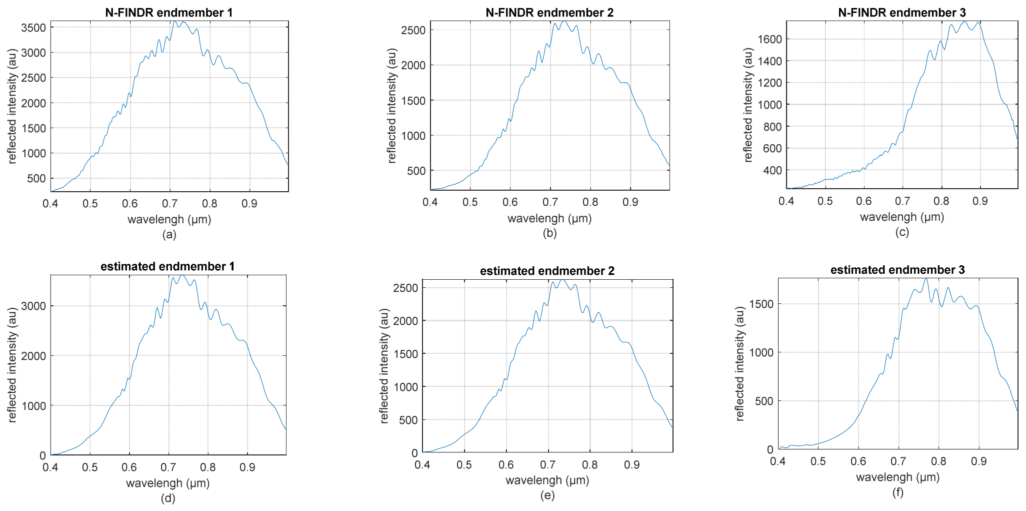


Figure 5.10. (a), (b) and (c) Ground truth spectra of 3 endmembers obtained by N-FINDR. (d), (e) and (f) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm for the VNIR cube.

Another actual hyperspectral data cube was also used in the same project to detect sound wheat grains from infected grains. This cube is comprised of images obtained using 288 wavelengths in the SWIR band (1 μm to 2.5 μm). This cube has dimensions of (75 x 110 pixels), and it shows grain kernels arranged as the previous experiment shown in Figure 5.4. The scene of the hyperspectral data cube showing a large island in the ocean (single wavelength image).

We applied the PCA to estimate the number of endmembers, M, as done before. Keeping 99% of the given data energy in PCA shows that this scene has 3 endmembers. Therefore, for this data cube, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 8250$, and the number of wavelengths, $L = 288$.

To validate spectral unmixing results obtained by applying our K-LARS based unmixing algorithm to this dataset, we need to know its ground truth, i.e., spectra of endmembers present in our selected region. For this purpose, we used N-FINDR as before. The locations of these approximately pure spectral pixels in our selected region obtained by N-FINDR [20] are shown in Figure 5.11.

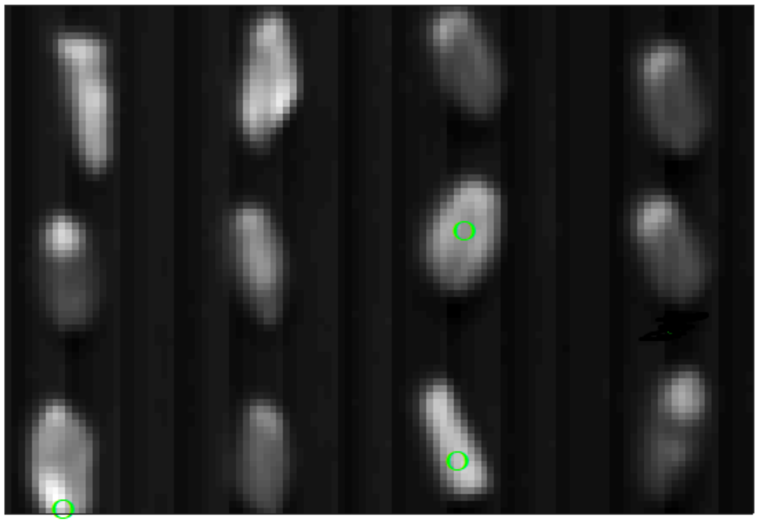


Figure 5.11. Locations in the SWIR cube scene (single wavelength image) of approximately pure spectral pixels obtained by N-FINDR for SND, ERG, and FDK grains.

Figure 5.12 (a) – (c) show the ground truth spectra of the 3 endmembers present in the scene. Figure 5.12 (d) – (f) show the estimated spectra of the 3 endmembers obtained by applying our K-LARS based spectral unmixing algorithm to this dataset. The correlations between the ground truth spectra and their corresponding estimated endmember spectra are 0.9078, 0.9846, and 0.9519, respectively. These quantitative results further demonstrate the validity of our new K-LARS algorithm and its successful application to HSI spectral unmixing.

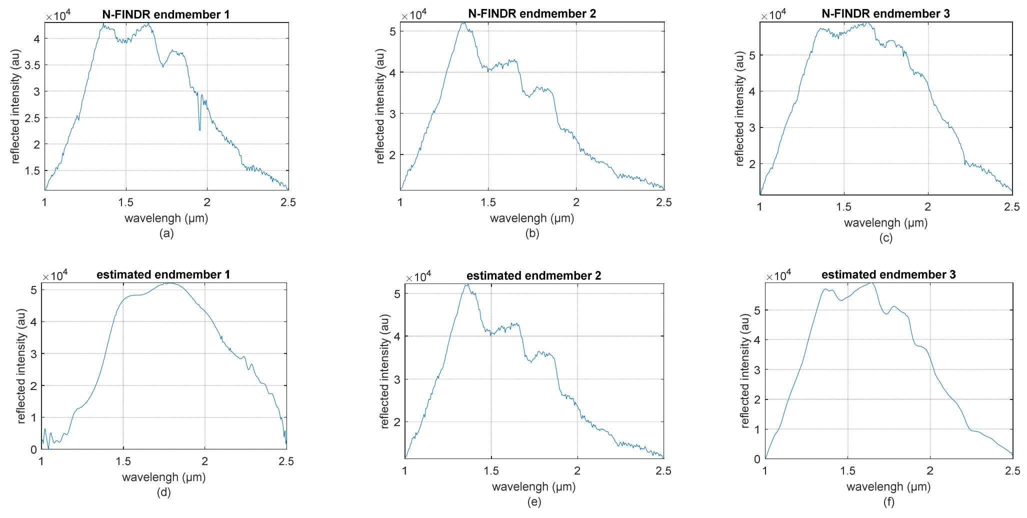


Figure 5.12. (a), (b) and (c) Ground truth spectra of 3 endmembers obtained by N-FINDR. (d), (e) and (f) Estimated spectra obtained using our K-LARS based spectral unmixing algorithm for the SWIR cube.

5.3 Spectral unmixing using Homotopy K-LARS dynamic updating algorithm

To demonstrate the validity of the Kronecker homotopy based spectral unmixing method, we applied it on the same HSI datasets given in the previous section; synthetic and actual data sets. The same preprocessing steps used in the previous section—namely, the denoising step and *Coiflet* dictionary warm-start representation—are used here as well.

The sparsity-based unmixing problem is the same as in the previous section, only this time the Kronecker homotopy algorithm described in Section 4.3 is used to estimate $\boldsymbol{\alpha}$. For each iteration, we fix $\boldsymbol{A}$ and estimate $\boldsymbol{\alpha}$ using the Kronecker homotopy algorithm using a warm-start of the unknown $\boldsymbol{\alpha}$ along the homotopy path to the final estimate of $\boldsymbol{\alpha}$; once this has been done, we then we fix $\boldsymbol{\alpha}$ and estimate $\boldsymbol{A}$ by solving the constrained linear least-squares problem.

5.3.1 Spectral unmixing of synthetic HSI multichannel data

We generated the same synthetic HSI dataset as that of the previous section has 3000 spectral pixels. In this dataset, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 3000$, and the number of wavelengths, $L = 491$ (0.42 μm to 14 μm). Figure 5.13(a) – (c) show the spectra of grass, concrete, and asphalt, whose different linear mixtures make up different spectral pixels in this dataset.

Figure 5.13 (d) – (f) show the estimated endmembers spectra obtained by applying our K-LARS dynamic updating homotopy based spectral unmixing algorithm to this dataset. The correlation coefficients between the exact spectra of grass, concrete, asphalt, and their estimated counterparts are 0.9398, 0.9986, and 0.9814, respectively. These quantitative results demonstrate the validity of our new K-LARS dynamic updating homotopy algorithm and its successful application to HSI spectral unmixing. The elapsed time of applying the K-LARS dynamic updating algorithm is 428.6 seconds compared to the static K-LARS algorithm elapsed time, which is 760 seconds. This shows that the dynamic algorithm is faster than the static one as it reduces the elapsed time to almost half, as shown in Table 5.1.

Table 5.1. Computation time of applying the K-LARS dynamic updating algorithm compared to the static K-LARS algorithm.

	K-LARS dynamic updating algorithm	Static K-LARS algorithm
Computation time [sec]	428.6	760

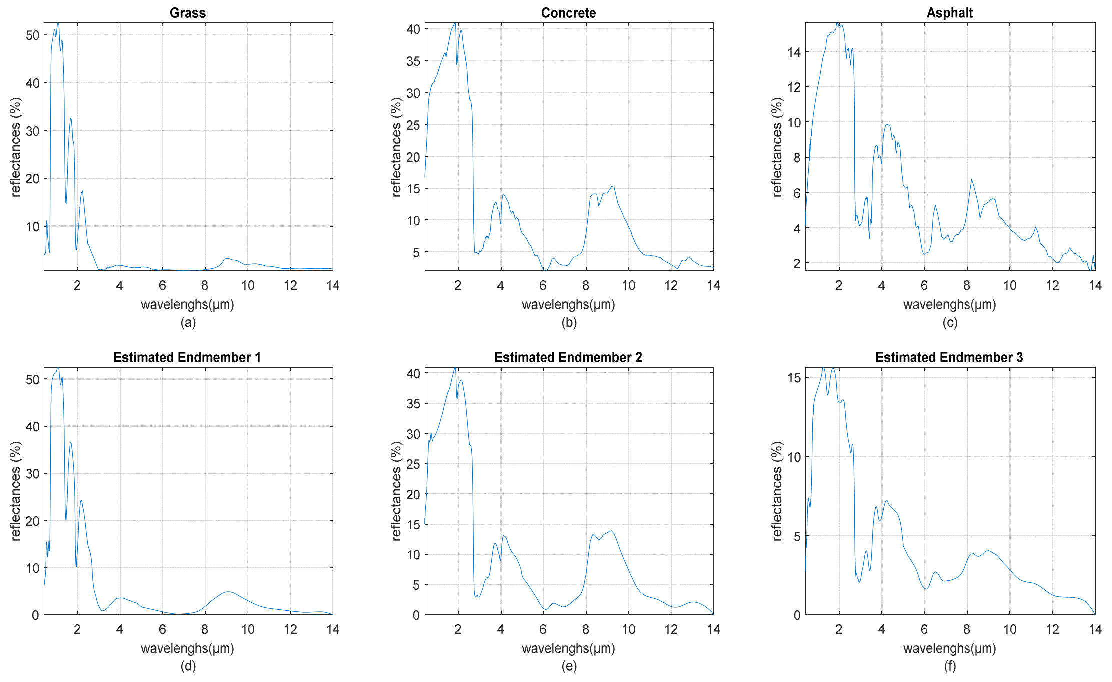


Figure 5.13. (a), (b) and (c) Exact spectra of grass, concrete, and asphalt, respectively. (d), (e) and (f) Estimated spectra obtained using our Kronecker homotopy based spectral unmixing algorithm.

5.3.2 Spectral unmixing of AVIRIS HSI multichannel data

We used the same actual HSI dataset, the Moffet Field, downloaded from *NASA's AVIRIS* website [9]. Also, we selected the same arbitrary region (70 x 60 pixels) as our region of interest. In the dataset corresponding to our selected region as estimated before, the number of endmembers, $M = 3$, the number of spectral pixels, $N = 4200$, and the number of wavelengths, $L = 224$. We also used the same ground truth endmembers obtained from N-FINDR.

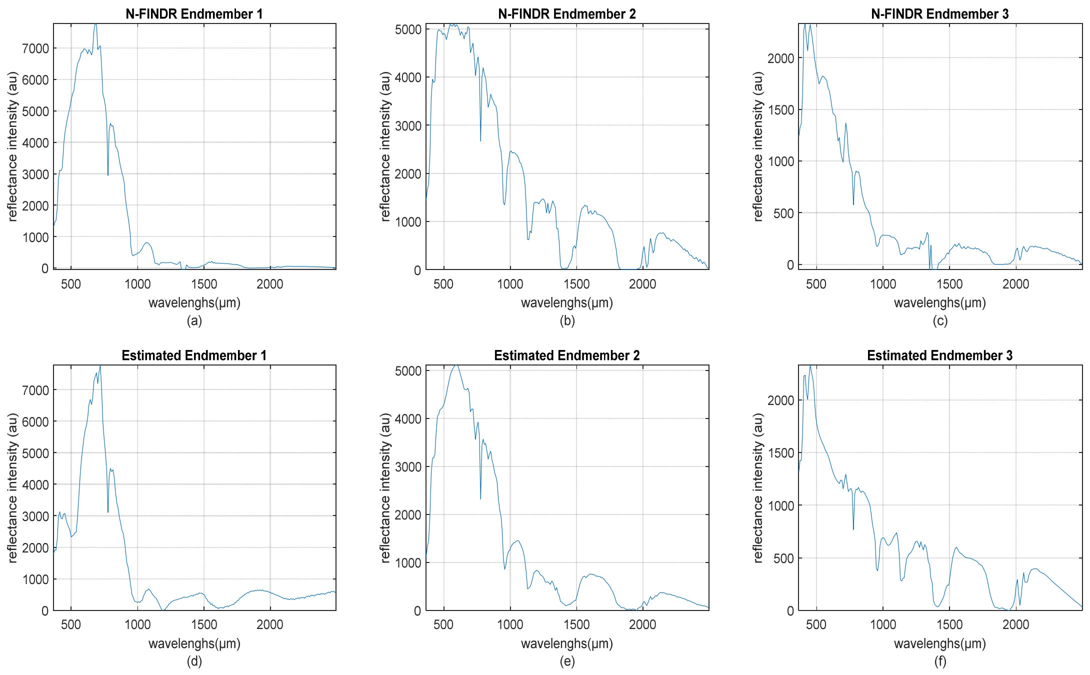


Figure 5.14. (a), (b) and (c) show the spectra of the N-FINDR results and the (d), (e), and (f) row shows our unmixing estimated endmembers spectra using the Kronecker homotopy.

Figure 5.14 (a) – (c) show the ground truth spectra of the 3 endmembers present in our selected region, and whose different linear mixtures make up different spectral pixels in this dataset. Figure 5.14 (d) – (f) show the estimated spectra of these 3 endmembers obtained by applying the Kronecker homotopy based spectral unmixing algorithm to this dataset. The correlations between the ground truth spectra and their corresponding estimated endmember spectra are 0.9545, 0.9851, and 0.9458, respectively. These quantitative results further demonstrate the validity of our new K-LARS dynamic updating homotopy algorithm and its successful application to HSI spectral unmixing. The elapsed time of applying the K-LARS dynamic updating algorithm is 198.1 seconds compared to the static K-LARS algorithm elapsed time, which is 765.5 seconds. This shows that the dynamic algorithm is faster than the static one as it reduces the elapsed time to almost a quarter, as shown in Table 5.2.

Table 5.2. Computation time of applying the K-LARS dynamic updating algorithm compared to the static K-LARS algorithm for the actual *AVIRIS* cube.

	K-LARS dynamic updating algorithm	Static K-LARS algorithm
Computation time [sec]	198.1	765.5

.

5.4 Chapter summary

In this chapter, we described a sparsity-based unsupervised hyperspectral imaging unmixing algorithm that uses iterative and alternate estimations of the endmembers and the abundances. Estimating the endmembers was done using one of two algorithms: the K-LARS algorithm described in Section 3.4, and the Kronecker homotopy algorithm described in Section 4.3. The abundances were estimated using a least-squares problem that is subject to the constraints given in (2.2). The sparsity-based unsupervised unmixing algorithm was then implemented using these two algorithms for actual *AVIRIS* data cube obtained from the *AVIRIS-NASA* website, and to a synthetic data cube constructed from selected materials from the *ASTER* spectral library. The results for each of these tests were presented.

Chapter 6
Dictionaries for Sparsity-Based Unmixing of Hyperspectral Images

Solving the *basis pursuit* optimization problem for the unmixing of hyperspectral imaging data assumes that the unknown endmembers are sparse in an appropriate dictionary. In this chapter, different types of dictionaries are used to promote the sparsity of the unknown endmembers in a sparsity-based unsupervised unmixing algorithm, which solves the *basis pursuit* optimization problem using two different methods: the proximal "thresholding" method, and our new K-LARS method. We use Wavelets as an analytic dictionary due to their suitability to the physical nature of the pixel spectra, which are mostly smooth functions with a small number of peaks. We also use an online learned custom dictionary, which should enhance the speed and simplicity of the learning algorithm. Finally, we present Coiflet as a suitable dictionary for sparse pixel spectra representation because of their large number of sparsity-enhancing vanishing moments. To demonstrate our unmixing results using different dictionaries, we use a synthetic HIS data cube generated from materials selected from the *ASTER* spectral library.

6.1 Introduction

The problem the unmixing problem, with a sparsity prior can be written as:

$$\min_{\boldsymbol{S}} \tfrac{1}{2}\|\boldsymbol{Y}-\boldsymbol{SA}\|_2^2 + \lambda\|\boldsymbol{S}\|_1 \tag{6.1}$$

This is a matrix *basis pursuit* problem [42], where λ is a regularization parameter that quantifies the tradeoff between the two terms of the equation, as described in the previous chapter. As a preprocessing step, we can transform the problem into a different domain using the dictionary $\boldsymbol{\Phi}$ to promote the sparsity of $\boldsymbol{S}$.

Selecting an appropriate dictionary is important because it allows sparser representation to be obtained and better unmixing results to be achieved. There are two main dictionary-based approaches to achieving sparse representation: the analytic dictionary approach and the learned custom dictionary approach [79]. In the analytic dictionary approach, the dictionary is generated from a family of known transforms. In the learned custom dictionaries approach, the dictionary is generated from the given measurements.

Due to the physical nature of spectra, which have a small number of peaks that correspond to their resonant absorption properties, Wavelets were chosen as an analytical dictionary for the sparse representation of the pixel spectra. In addition, the properties of the Coiflet assert it as the optimum dictionary for promoting the sparsity of the unknown endmembers in the *basis pursuit* unmixing optimization problem.

Unlike the predefined analytical dictionary, which aims to represent the sources $\boldsymbol{S}$ as sparsely as possible, we can obtain a custom dictionary from the given data $\boldsymbol{Y}$ in order to provide a better fit for the spectra [80]. Algorithms for dictionary learning are either batch-iterative procedures or online incremental-iterative procedures.

6.2 Dictionary approaches for sparse representation

A vector signal $\boldsymbol{x}$ in a finite-dimension subspace of $\mathbb{R}^{\mathrm{L}}$, $\boldsymbol{x} = [x(1), \dots, x(L)]^T$, is considered sparse if most of its entries are zeros. A k-sparse signal has only k nonzero values among its entries. A non-sparse signal can be sparsified in a known frame called a dictionary $\Phi = [\varphi_1, \dots, \varphi_T]$.

$$\boldsymbol{x} = \Phi\boldsymbol{\alpha} = \sum_{i=1}^{T} \boldsymbol{\alpha}[i]\varphi_i \tag{6.2}$$

where $\boldsymbol{\alpha}$ is the representation coefficients of $\boldsymbol{x}$ in the dictionary Φ whose columns φ_i are normalized to a unit l_2-norm. Furthermore several subdictionaries $(\Phi_\mathrm{k})_{\mathrm{k}=1,\dots,K}$ can be combined to build a larger dictionary $\boldsymbol{\Phi}$, which will be a concatenation of these K subdictionaries Φ_k such that $\boldsymbol{\Phi} = [\Phi_1, \dots, \Phi_K]$.

The signal $\boldsymbol{x}$ is sparse in $\boldsymbol{\Phi}$ if it can be well represented as a superposition of a small number of atoms found in the dictionary $\boldsymbol{\Phi}$, as shown in Figure 6.1.

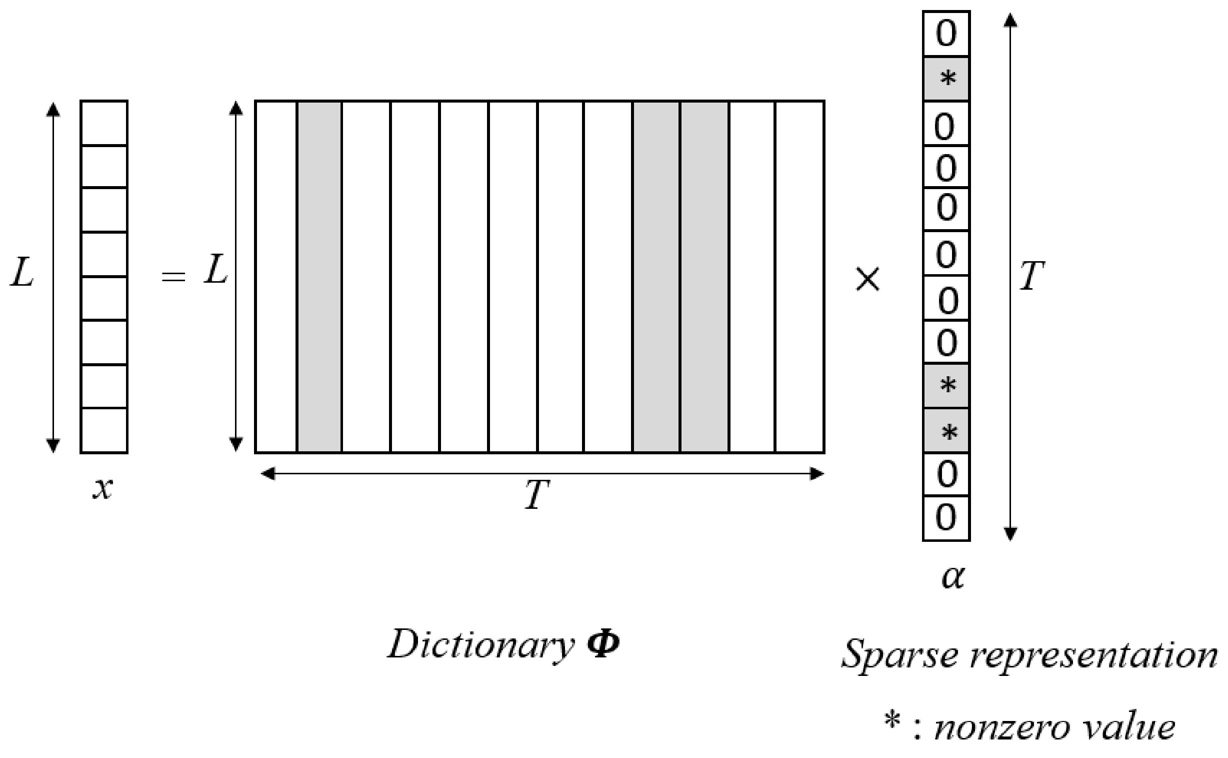


Figure 6.1. The sparse signal representation.

The choice of an appropriate dictionary is critical for obtaining sparser representation. Dictionaries for the sparse representation of signals can be divided into two broad approaches: an analytic dictionary approach and a learned custom dictionary approach.

In the analytic dictionary approach, the dictionary is generated from a known transform or a concatenation of different transforms. For example, these dictionaries could be generated using different time-frequency atoms, such as Fourier transform [81]; Discrete cosine transform (DCT) [82], which is suitable for signals that have periodic components; Wavelets [72], which are suitable for smooth signals including spikes; Curvelets [83], which are suitable for 2D smooth signals; and Contourlets [84]. A unity matrix (I) can also be used because it is suitable for spiky signals. An analytic dictionary could be a union of ortho-bases; when this is the case, it will result in a tight frame.

In contrast, the learned custom dictionary approach derives the dictionary from the spectra of the given pixels. Examples of learned custom dictionaries range from the well-known and simple Principal Component Analysis (PCA) [78] to the Method of Optimal Dictionary (MOD) [85], K-SVD [86], and online dictionary learning [87]. All these algorithms seek a direct transform into the sparse representation of signals. This approach yields dictionaries that allow for more data fitting, thus providing better performance in many applications. However, this increase in performance is also accompanied by greater computational complexity.

There are two main types of dictionary learning algorithms: the batch approach and the online approach. Batch-iterative procedures, such as the MOD algorithm, access the whole data set at each iteration in order to minimize a cost function under given constraints. This approach has one major drawback: it cannot effectively handle very large data or dynamic data changing over time.

Unlike batch approaches, online approaches [87] process the given data incrementally, one sample at a time. Online approaches outperform batch approaches in terms of speed, and the quality of the dictionary learned for both large and small data sets.

6.3 Coiflet orthogonal wavelet dictionary for spectral unmixing

The wavelet transform is an excellent transform for sparsely representing 1-D smooth signals that have a small number of irregular points. The wavelet family was identified as a potential dictionary of choice for spectral pixels that are 1-D smooth signals, whose physical nature dictates that they have a small number of peaks corresponding to resonant absorption peaks.

One important property of wavelets is the vanishing moments of a wavelet function ψ:

$$\int x^l \, \psi(x) \, dx = 0 \;\; for \; 0 \le l \le m \tag{6.3}$$

A wavelet function has m vanishing moments that can be represented as the nth order derivative of a given function θ. Thus, the resulting wavelet transform is equivalent to a multiscale differential operator [72]:

$$\psi(t) = (-1)^m \frac{d^m \theta(t)}{dt^m} \tag{6.4}$$

We note that wavelets with large numbers of vanishing moments give sparse representations for many piecewise smooth signals. However, there is a trade-off between the number of vanishing moments of ψ and the support for ψ in the time domain [72].

The support of a wavelet in the time domain and its number of vanishing moments are independent. However, if an orthogonal discrete wavelet, ψ, has m vanishing moments, then its support in the time domain will be $2m - 1$. Daubechies wavelet has minimum support in the time domain for a given number of vanishing moments, and, as such, they are considered optimal in this regard.

6.3.1 Coiflet wavelet as the optimal dictionary for sparsifying spectral pixels for basis pursuit.

Coiflet wavelet was developed by I. Daubechies at the request of R. Coifman. In addition to their wavelet function, ψ, with order n that has $2n$ vanishing moments, their scaling function, φ, also has $2n - 1$ vanishing moments. Both the wavelet function, ψ, and the scaling function, φ, have the minimum possible support in the discrete-time domain of $6n - 1$. Therefore, among all wavelets with given support in the discrete-time domain, a Coiflet wavelet would have the largest number of vanishing moments for both ψ and φ.

The use of a dictionary that is comprised of Coiflet wavelet atoms can produce sparser representations of a regular function with a small number of singularities—for example, pixel spectra with resonant absorption peaks— than any other representation using another wavelet dictionary. This will reduce the computational cost of solving the *basis pursuit* problem in the sparsity-based unmixing algorithm.

6.4 Online sparse coding algorithm

We chose the online sparse coding algorithm as an example of a dictionary learned algorithm because it is newer, simpler, and faster than other algorithms. As outlined in Table 6.1, this algorithm is based on two main stages: the sparse coding stage and the dictionary update stage. In the sparse coding stage, one single spectrum $\boldsymbol{y}_i$ is drawn at each iteration i of the algorithm, and the sparse coding for this sample is then computed using the LARS algorithm [53] described in Section 3.3 and the last update of the dictionary. The LARS algorithm has the advantage of providing all the solutions for all values of λ. To reduce processing complexity, we can stop the processing of this step at a certain sparsity level $||\alpha||_1$ instead of getting all LARS solutions:

$$\min_{\alpha} \tfrac{1}{2}||y_i - \boldsymbol{\Phi}_{i-1}\alpha||_2^2 + \lambda||\alpha||_1 \tag{6.5}$$

In the dictionary update stage, the dictionary is updated by minimizing the objective function $\widehat{f_N}(\boldsymbol{\Phi})$ using the 1^{st} order stochastic gradient descent method. This step could be done numerically with the auxiliary matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ [87].

$$\widehat{f_N}(\boldsymbol{\Phi}) = \frac{1}{N}\sum_{i=1}^{N} \tfrac{1}{2}||y_i - \boldsymbol{\Phi}\alpha_i||_2^2 + \lambda||\alpha_i||_1 \tag{6.6}$$

Table 6.1. Online dictionary learning algorithm steps.

- Given: The measurement matrix $\boldsymbol{Y}$.
- Initialization steps: The dictionary $(L \times L)$ $\boldsymbol{\Phi}_0$ is initialized randomly.
- *Sparse Coding stage:*
  - Initially, set $(L \times L)$ $\boldsymbol{U}_0$ and $(L \times L)$ $\boldsymbol{V}_0$ matrices to zero.
  - For each iteration i:
    - Choose $\boldsymbol{y}_i$ randomly from the hyperspectral data cube.
    - Solve equation (6.5) using LARS, thereby obtaining the vector α_i.
    - Update $\boldsymbol{U}$ by $\boldsymbol{U}_i = \boldsymbol{U}_{i-1} + \alpha_i {\alpha_i}^T$.
    - Update $\boldsymbol{U}$ by $\boldsymbol{V}_i = \boldsymbol{V}_{i-1} + \boldsymbol{y}_i {\alpha_i}^T$.
- *Dictionary Update stage:*
  - $\boldsymbol{\Phi}_i$ is calculated by solving (6.6) numerically using $\boldsymbol{\Phi}_{i-1}$, $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ column-wise
  - Repeat **for** loop until the difference between the current and updated dictionary elements $\boldsymbol{e}$ is below the specified value. (j^{th} column of $\boldsymbol{\Phi}_i = [\boldsymbol{\varphi}_1, \dots, \boldsymbol{\varphi}_L], \boldsymbol{U}_i = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_L]$) and $\boldsymbol{V}_t = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_L]$).

```
for $j = 1 \ to \ L$
    $\boldsymbol{q}_j = \frac{1}{\boldsymbol{U}(j,j)}(\boldsymbol{v}_j - \boldsymbol{\Phi}_{t-1}\boldsymbol{u}_j) + \boldsymbol{\varphi}_j$
    $\boldsymbol{\varphi}_j = \frac{1}{\max(\|\boldsymbol{q}_j\|_2, 1)}\boldsymbol{q}_j$
    $\boldsymbol{e}_j = \sqrt[2]{\sum_L |{\boldsymbol{\varphi}_j}^i - {\boldsymbol{\varphi}_j}^{i-1}|^2}$
End for

Checks  $\boldsymbol{e} = \frac{1}{L}\sum_{j=1}^{L} \boldsymbol{e}_j$
```

6.5 Convergence using different dictionaries

This section compares the use of 4 different dictionaries in two different sparsity-based spectral unmixing algorithms. The four dictionaries are the Coiflet dictionary, the Daubechies wavelet dictionary, the DCT dictionary, and the online learning dictionary. The two sparsity-based algorithms solve the *basis pursuit* unmixing problem. The first algorithm solves the problem using the proximal method "thresholding" [5] that has the disadvantages of requiring prior knowledge of a suitable value of λ . The second algorithm solves the *basis pursuit* problem using the K-LARS algorithm described in Section 3.4, which avoids this disadvantage by not requiring prior knowledge of suitable λ values.

Our implementation uses the synthetic cube described in Section 5.2.1; this cube was constructed from selected materials from the *ASTER* spectral library [9] and had the abundance constraint shown in (2.2). A mixed cube of 300 spectra pixels was synthesized from the spectra of three known materials: grass, concrete, and asphalt. The cube was established using random abundance matrix $\boldsymbol{A}$ and was subject to the constraints given in (2.2). Additive Gaussian noise was added to the synthetic cube spectra with an $\mathrm{SNR} = 30$ dB. In this case, the number of endmembers was $M = 3$, the number of spectral pixels was $N = 300$, and the number of wavelengths was $L = 491$.

As was with the cubes in Sections 5.2 and 5.3, the denoising step for this cube was carried out by first using the MatLab function, "ThreshWave," which was obtained from the WaveLab toolbox [88]. This function performs denoising by using the wavelet "thresholding" method [77] with the "Symmlet" wavelet function and hard thresholding.

For the wavelet transform, we used the MatLab function “FWT_PO” with a Coiflet filter that had $n = 5$ vanishing moments for the Coiflet dictionary and a Daubechies filter that had n = 6 vanishing moments for the Daubechies wavelet dictionary. This function “FWT_PO” and the filter-making function “MakeONFilter” were obtained from the WaveLab toolbox [88]. For the DCT dictionary, we used the built-in MatLab function “dct.” For the online dictionary, we implemented the algorithm described in Section 6.4.

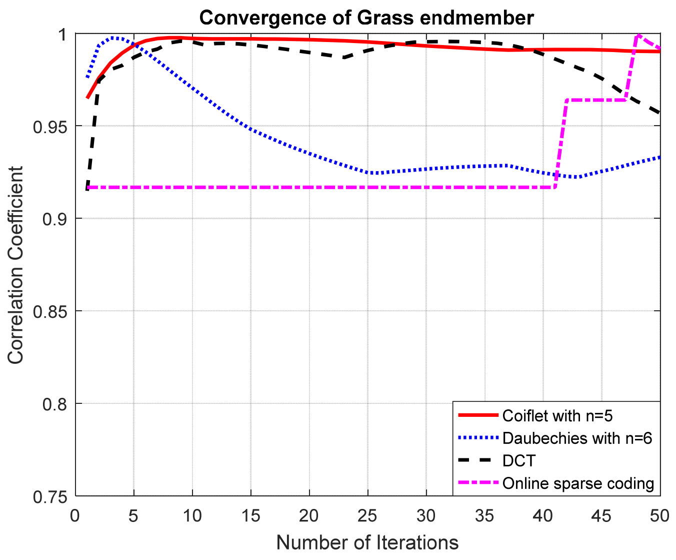
The convergence of solving the basis pursuit problem for the two unmixing algorithms was investigated to confirm that the use of the Coiflet dictionary would result in faster convergence. This convergence is obtained by calculating the correlation coefficient between the exact and estimated endmember spectrums and the number of iterations for each endmember. As the correlation coefficient increases to 1, the optimum solution becomes faster.

Four dictionaries were used in these tests: 1) a Coiflet dictionary with order $n = 5$; 2) a Daubechies wavelet dictionary with order $n = 6$; 3) a DCT dictionary; and 4) a custom online learned sparse coding dictionary (described in Section 6.4).

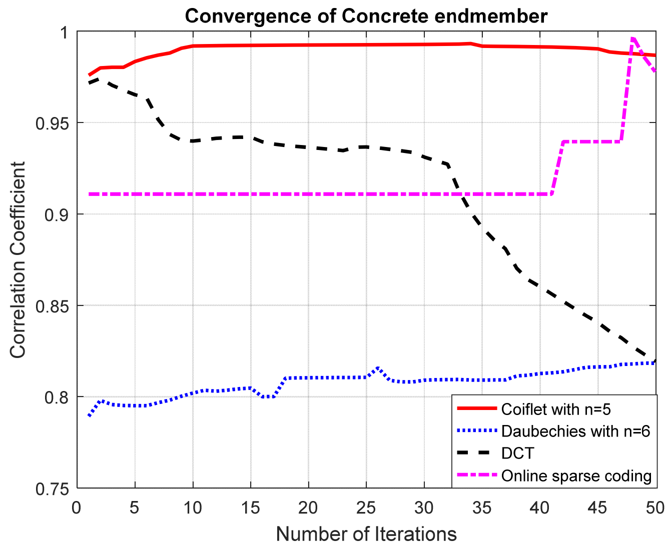
6.5.1 Proximal methods “Thresholding”

Solving the *basis pursuit* unmixing problem using proximal methods is done by iteratively and alternately estimating the sparse endmembers coefficient $\boldsymbol{\alpha}$ by using least-squares, and then applying hard thresholding, and obtaining the abundances $\boldsymbol{A}$ by performing 50 iterations of the constrained linear least-squares method. There will be faster convergence when the Coiflet dictionary is used, as it demonstrates higher correlation coefficients for all three endmembers than the other dictionaries, as shown in

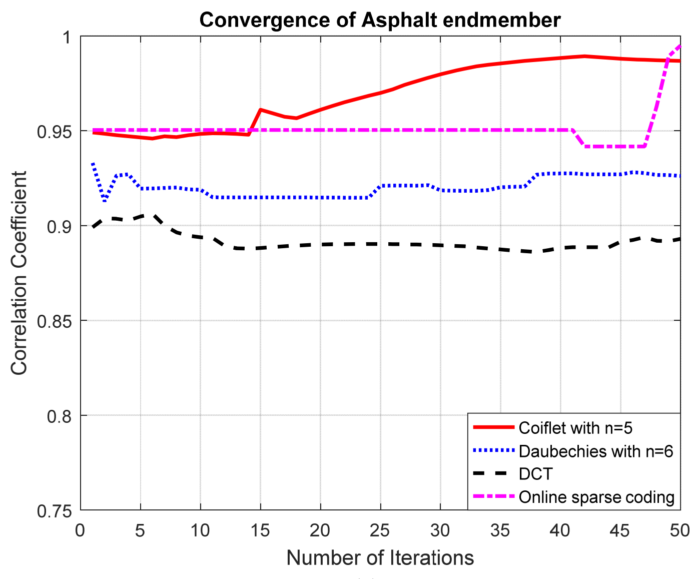
Figure 6.2. This confirms that the Coiflet dictionary is the optimum dictionary for the sparsity-based unmixing algorithm when using the proximal methods. This method has the main drawback in that the threshold value, which is indicated from λ, is not fixed for the four dictionaries and is difficult to obtain prior knowledge of.



(a)



(b)

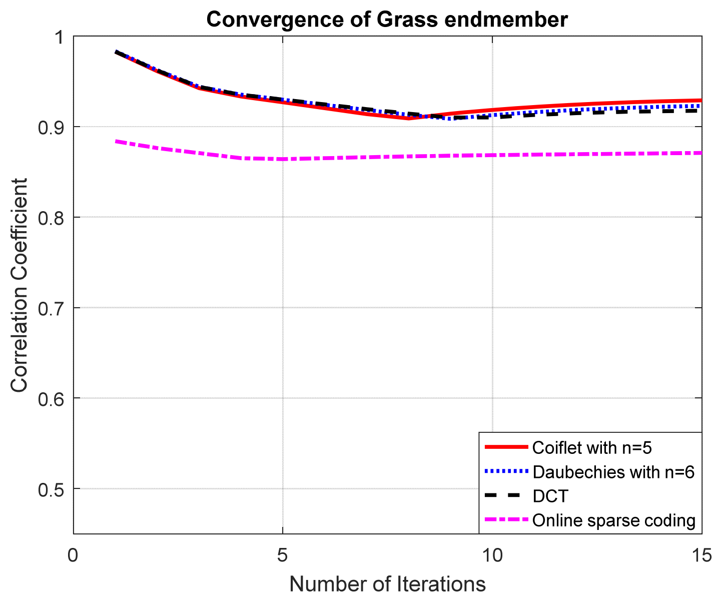


(c)

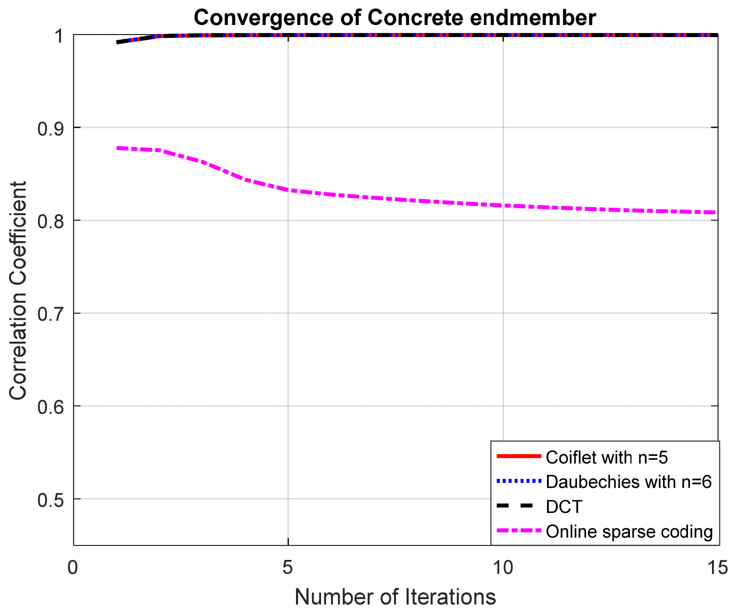
Figure 6.2. The correlation coefficient between the exact and estimated endmembers for the proximal methods using four different dictionaries: (a) for the grass endmember; (b) for the concrete endmember; and (c) for the asphalt endmember.

6.5.2 K-LARS method

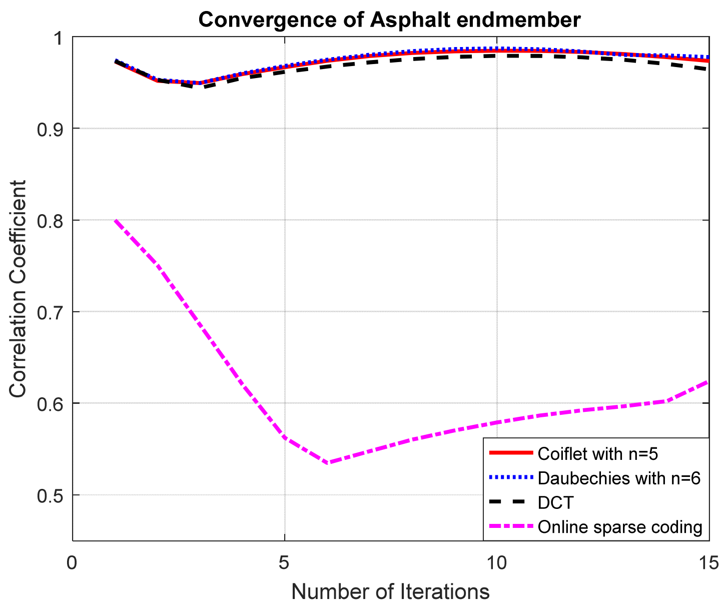
The *basis pursuit* unmixing problem can also be solved by iteratively and alternately estimating the sparse endmembers coefficient $\boldsymbol{\alpha}$ by using the K-LARS algorithm and obtaining the abundances $\boldsymbol{A}$ via 15 iterations of the constrained linear least-squares method. The tensor LARS method provides all solutions corresponding to all values of the regularization parameter λ. When using the tensor LARS method, convergence will occur faster when the Coiflet dictionary is used, as it shows higher correlation coefficients than the other dictionaries for all the three endmembers (Figure 6.3).


Convergence of Grass endmember
Correlation Coefficient
Number of Iterations
1
0.9
0.8
0.7
0.6
0.5
0
5
10
15
Coiflet with n=5
Daubechies with n=6
DCT
Online sparse coding

(a)


Convergence of Concrete endmember
Correlation Coefficient
Number of Iterations
1
0.9
0.8
0.7
0.6
0.5
0
5
10
15
Coiflet with n=5
Daubechies with n=6
DCT
Online sparse coding

(b)



(c)

Figure 6.3. The correlation coefficient between the exact and estimated endmembers for the K-LARS using four different dictionaries: (a) for the grass endmember, (b) for the Concrete endmember, and (c) for the Asphalt endmember.

Moreover, since the Coiflet dictionary enhances sparsity, its use can reduce the complexity of the K-LARS algorithm as it is controlled by its sparsity level or the number of nonzero coefficients in the estimated signals $||\boldsymbol{S}||_0$. The minimum required sparsity level is the level that produces a solution that has a fitting residual norm $||\boldsymbol{Y} - \boldsymbol{SA}||_2$ that is very close to the fitting residual norm of the full least-squares solution. In order to use the Coiflets dictionary for the K-LARS algorithm, there must be a sparsity level of 208. By contrast, the Daubechies wavelet dictionary requires a sparsity level of 242, and the DCT dictionary requires a sparsity level of 320. Moreover, the online sparse coding dictionary can only be used for the tensor LARS algorithm if there is a sparsity level of 426. These sparsity levels are obtained in order to obtain a solution that has a fitting residual norm that reaches 0.95% of the full least-squares solution's fitting residual norm.

The Coiflet dictionary's ability to provide the lowest sparsity level of the four different dictionaries, and thus requires the least amount of computational time, assets it as the optimum dictionary for use with the sparsity-based unmixing algorithm using the K-LARS algorithm.

6.6 Chapter summary

Analytic and learned custom dictionaries were used for the sparse representation of pixel spectra in a sparsity-based unsupervised unmixing algorithm for hyperspectral imaging data. The Coiflet dictionary was identified as a most suitable dictionary for the sparse representation of the hyperspectral pixel spectra due to its large number of vanishing moments. A large number of vanishing moments results in higher correlation coefficients between all the estimated and exact endmembers, as well as faster and simpler computation in solving the *basis pursuit* optimization problem for the purpose of unmixing hyperspectral imaging data cubes. In order to provide an example of a hyperspectral data, a synthetic data cube was generated from selected materials from the *ASTER* spectral library to demonstrate our results.

Chapter 7
High Spatial Resolution Anomaly Detection for Hyperspectral Images

Anomaly detection in hyperspectral images involves a binary classifier to label every spectral pixel in the hyperspectral data cube as either background or an anomaly. Popular anomaly detection algorithms typically use a relatively wide spatial window, i.e., one that includes a large number of spectral pixels, to generate an accurate statistical model of the background. However, this use of a wide spatial window results in a low spatial resolution anomaly detection. In this chapter, we present a novel high spatial resolution anomaly detection algorithm that requires a smaller number of spectral pixels in the window under test. To enable accurate modeling of complex backgrounds, e.g., near object boundaries, we use a mixture of factor analyzers (MoFAs) model that results in both *dimensionality reduction* and *Gaussian mixture modeling* in the *same* step. In this chapter, given a small number of spectral pixels comprising a small spatial window in a hyperspectral image, we estimate their MoFAs model parameters using a *lasso*-penalized maximum likelihood estimation. Also, instead of using the traditional Bayesian Information Criterion (BIC) for the model, i.e., model dimension and number of Gaussian mixture components, selection, we use the recently introduced *LASSO*-penalized Bayesian Information Criterion (LPBIC) [97]. To demonstrate the improved performance of our novel high-resolution anomaly detection algorithm, we compare its performance with an algorithm using reduced dimension spectral pixels and the traditional BIC for model selection, on *AVIRIS* hyperspectral images.

7.1 Anomaly detection basics

Anomaly detection is an important processing task performed on Hyperspectral images. Anomaly detection is a binary classification problem to distinguish between a single spectral pixel and its background [46]. In other words, the anomaly detection problem focuses on differentiating unusual pixels, called anomalies, from assumed homogenous background pixels [2].

Anomaly detection is treated as a binary hypothesis testing problem [89] carried out at every pixel in the cube. Let H_0 represent the null hypothesis that there is no anomaly at the pixel under test, and H_1 the alternative hypothesis that an anomaly is present. Let $\boldsymbol{x}$ denote the L-dimensional vector corresponding to the pixel under test, where L is the number of spectral bands. A detection statistic $D(\boldsymbol{x})$ is compared to a threshold η. If $D(\boldsymbol{x}) < \eta$, this indicates the null hypothesis H_0 or there is no anomaly and if $D(\boldsymbol{x}) > \eta$, this indicates the alternative hypothesis H_1which means that this pixel under test is an anomaly.

$$D(\boldsymbol{x}) \gtrless_{H_0}^{H_1} \eta \tag{7.1}$$

The detection statistic $D(\boldsymbol{x})$ is often obtained using N number of background pixels. These background pixels may be characterized by their 1st or 2nd order statistics, i.e., their mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$, or by their probability density function (PDF), $f_{\text{pdf}}(\boldsymbol{x})$.

Anomaly detection algorithms can be divided into global and local detection algorithms, where background pixels can include all pixels in an image, or only some pixels in the neighborhood of the pixel under test, respectively. In local anomaly detection algorithms, an annulus of pixels surrounding the pixel under test, shown in Figure 7.1, is considered as the local background. The choice of size of this annulus is important as it involves a compromise between 1) choosing a large annulus to provide an accurate estimate of its statistical model parameters; 2) choosing a small annulus to a) ensure statistical stationarity within this background window; and b) ensure anomaly detection within a small region, i.e., anomaly detection with high spatial resolution.

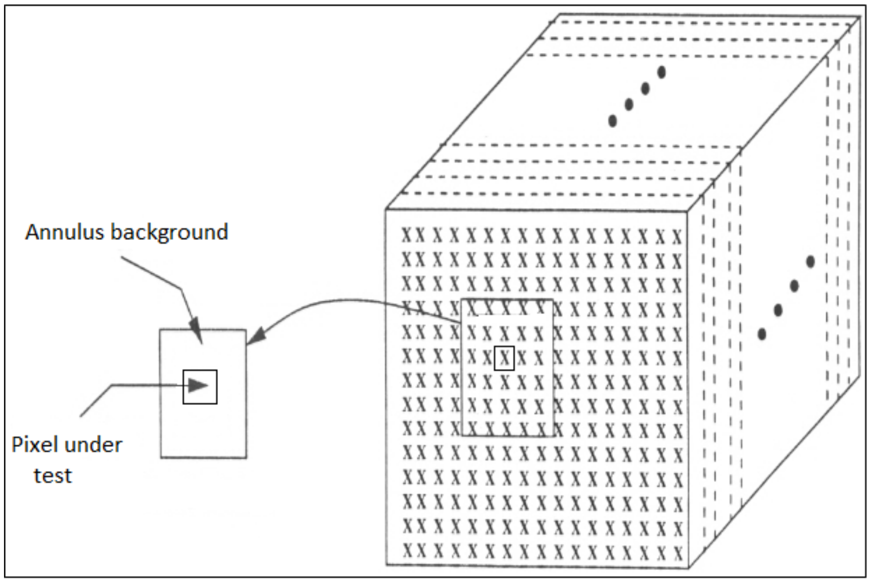


Figure 7.1. Background pixels for local anomaly detection algorithms.

These background pixels can be modeled by their probability density function (PDF), $f_{\text{pdf}}(\boldsymbol{x})$. Using this model could result in an anomaly detection algorithm that has a detection statistic $D(\boldsymbol{x}) = g(1/f_{\text{pdf}}(\boldsymbol{x}))$, where $g(.)$ is a monotonic function, e.g., a logarithmic function, resulting in a log-likelihood based detection statistic

$$D(\boldsymbol{x}) = -log(f_{\text{pdf}}(\boldsymbol{x})) \tag{7.2}$$

7.2 Background modeling

Many anomaly detection algorithms estimate parameters of the pdf of representing anomaly-free background window pixels, then using this pdf obtain the likelihood of the pixel-under-test to decide if it is an anomaly or part of the background [3]. Therefore, accurately modeling the background is an important step in anomaly detection, as obtaining a more accurate statistical model would result in better anomaly detection performance. Different background models, and how their parameters are estimated, could lead to different anomaly detection algorithms. Classical background models, such as 1) single multivariate normal (MVN) models [50] mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$; 2) subspace models [51] that depend on projecting the background pixels to a low dimension subspace using a projection matrix $\boldsymbol{P}$, have led to several anomaly detection algorithms, e.g., the RX anomaly detection algorithm [49] that use a single MVN pdf and an orthogonal subspace projection (OSP) [52].

These algorithms are simple, but they do not consider a practical, i.e., complex, background, e.g., a non-gaussian, or nonhomogeneous background where edges between different homogeneous regions, i.e., clusters of background pixels, are present in the scene [90].

7.2.1 Statistical mixture models to model complex backgrounds

A recent approach for detecting anomalies in hyperspectral images is based on a mixture model that divides the background into a small number of MVN mixture components, or clusters of spectral pixels. The pdf of all the spectral pixels in this window will be the weighted sum of the individual pdfs of these clusters of spectral pixels [91]. A Gaussian

Mixture Model (GMM) assumes the spectral pixels as a weighted sum of several multivariate gaussian distributions. Mathematically, a finite GMM with G components has a probability density function

$$f_{\text{pdf}}(\boldsymbol{x}) = \sum_{g=1}^{G} \pi_g \phi(\mathbf{x}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) \tag{7.3}$$

where $\pi_g > 0 \text{ s.t.} \sum_{g=1}^{G} \pi_g = 1$ are called mixing proportions and $\phi(\mathbf{x}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ is the g^{th} component density which has the multivariate gaussian distribution with mean $\boldsymbol{\mu}_g$ and covariance $\boldsymbol{\Sigma}_g$.

Estimating the parameters of a finite GMM, mean $\boldsymbol{\mu}_g$, covariance $\boldsymbol{\Sigma}_g$, and mixing proportions π_g, could be significantly simplified by imposing various constraints, especially upon the covariance structure, which would result in a flexible modeling paradigm. Fitting a GMM to given data, e.g., spectral pixels in a spatial window, is called *model-based clustering* [92].

Anomaly detection based on GMM modeling of the background could be implemented using two approaches; 1) using a cluster-conditional test that is applied to the pixel-under-test to compute a cluster-conditional Mahalanobis distance that results in selecting the $j^{\underline{th}}$ mixture component as the closest one among those representing the background pixels; 2) using the GMM-based estimate of $f_{\text{pdf}}(\boldsymbol{x})$ to obtain the log-likelihood in (7.2) of the pixel under test [90].

7.2.2 A mixture of factor analyzers

In addition to statistical modeling of the given data, i.e., spectral pixels, a Mixture of Factor Analyzers (MoFAs) [93] is also a data dimensionality reduction technique that maps the given data to a significantly lower-dimensional space. MoFAs assumes a latent Gaussian model structure for the given data. It has the same density as a finite GMM in (7.3) but assumes a covariance structure of a form $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g$, where $\boldsymbol{\Lambda}_g$ is the loading matrix which is $p \times q$ such that $q \ll p$ and $\boldsymbol{\psi}_g$ is a diagonal matrix. In [94], the authors developed a family of eight Gaussian mixture models for clustering by possibly imposing, three constraints; $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$, $\boldsymbol{\psi}_g = \boldsymbol{\psi}$ and $\boldsymbol{\psi}_g = \psi_g \boldsymbol{I}_p$ upon the a covariance structure $\boldsymbol{\Sigma}_g$. This MoFAs model simultaneously implements both modeling (pixel clustering), and dimensionality reduction at the same step. Compared to applying each step separately, this simultaneous approach has been shown to result in better modeling and clustering results [95].

7.2.3 Mixture model selection

We note that modeling of spectral pixels using a MoFAs model requires a choice of the number of mixture components G and the data subspace dimension q, which are commonly selected using the Bayesian Information Criterion (BIC) [96]. The BIC value depends on the likelihood value of the given data using the obtained statistical model and a penalty term that depends on the number of estimated parameters. However, BIC is not recommended for high dimensional data like spectral pixels, as the penalty term would dominate the likelihood term. This issue could be mitigated by using the recently introduced *LASSO*-penalized BIC (LPBIC) [97]. LPBIC is more suited than the traditional BIC to select the number of mixture components G and the data subspace dimension p when modeling high-dimensional data using MoFAs [95].

In this chapter, we implement two algorithms to statistically model local complex background, comprised of a different number of spectral pixel clusters. First, we model lower-dimensional spectral pixels whose dimensionality has been reduced, using Principal Component Analysis (PCA), as a MoFAs, and then use BIC for model selection. Second, we model higher dimensional spectral pixels as MoFAs using a *lasso*-penalized maximum likelihood estimation method, and then use LPBIC for model selection.

7.3 Fitting Mixtures of Factor Analyzers (MoFAs)

Factor analysis [98] is a data reduction technique that finds latent factors that demonstrate the variability in the data. It models a p-dimensional random vector $\boldsymbol{x}$ using a q-dimensional vector of latent factors $\mathbf{u}$ where $q \ll p$. The model is written as $= \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{u} + \boldsymbol{\epsilon}$, where $\boldsymbol{\Lambda}_g$ is $p \times q$ factor loading matrix, the latent factors $\mathbf{u} \sim N(\mathbf{0}, \boldsymbol{I}_q)$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\psi})$, where $\boldsymbol{\psi} = diag(\psi_1, \psi_2, \ldots, \psi_p)$. So, $\boldsymbol{x}$ has a multivariate gaussian marginal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\psi}$.

The mixture of factor analyzers model [93] was developed as an extension of the factor analysis model. It assumes a mixture of Gaussian distributions with factor analysis covariance structures. The mixture of factor analyzers model has density:

$$f_{\text{pdf}}(\boldsymbol{x}_i) = \sum_{g=1}^{G} \frac{\pi_g}{(2\pi)^{\frac{p}{2}} \left| \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g \right|^{1/2}} \times exp\left\{ -\frac{1}{2} (x_i - \boldsymbol{\mu}_g)^T \left(\Lambda_g \Lambda_g^T + \boldsymbol{\psi}_g \right)^{-1} (x_i - \boldsymbol{\mu}_g) \right\} \quad (7.4)$$

where π_g is the membership probability of a given observation $\boldsymbol{x}_i$ in group g and the density of $\boldsymbol{x}_i$ in the group, g is multivariate Gaussian with mean $\boldsymbol{\mu}_g$ and covariance $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g$. The mixture of factor analyzers model in (7.4) is extended by imposing, or not, three constraints across groups $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$, $\boldsymbol{\psi}_g = \boldsymbol{\psi}$ and $\boldsymbol{\psi}_g = \psi_g \boldsymbol{I}_p$ upon the covariance structure $\boldsymbol{\Sigma}_g$.

Fitting these models is commonly done using the Alternating Expectation-Conditional Maximization (AECM) algorithm [99]. This algorithm is the extension of the Expectation-Maximization (EM) algorithm [100] that estimates the maximum likelihoods of $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\psi}_g$ using the missing data in two main stages. For fitting MoFAs, the AECM estimates π_g and $\boldsymbol{\mu}_g$in the first stage and the missing data are the unobserved group labels **z** and at the second stage the AECM estimates $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\psi}_g$ and the missing data are the group labels **z** and the unobserved latent factors **u**.

The AECM algorithm iteratively updates the parameters until convergence to maximum likelihood estimates of $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\psi}_g$. The resulting $\hat{z}_{i_g}$values at convergence are estimates of the *a posteriori* probability of group membership for each observation and can be used to cluster observations into groups [101].

7.4 *Lasso*-penalized Bayesian Information Criterion and *Lasso*-penalized Likelihood

The likelihood of n realizations of a p-dimensional random vector $\boldsymbol{x}$ that has a G-component finite gaussian mixture model is:

$$\mathcal{L}(\boldsymbol{\vartheta}|\boldsymbol{x}) = \prod_{i=1}^{n} \sum_{g=1}^{G} \pi_g \phi(\boldsymbol{x}_i|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) \tag{7.5}$$

The Bayesian Information Criterion (BIC) [96] is by far one of the most popular methods to select the best mixture model in model-based clustering approaches. It is used to select the best model from a given family of mixture models, the number of mixture components G and the number of latent variables in the mixture of factor analyzers model [92]. It can be expressed as:

$$\mathrm{BIC} = 2\log \mathcal{L}(\widehat{\boldsymbol{\vartheta}}|\boldsymbol{x}) - \rho \log n \tag{7.6}$$

where $\widehat{\boldsymbol{\vartheta}}$ is the maximum likelihood estimate (MLE) of $\boldsymbol{\vartheta}$, ρ is the number of estimated parameters, and n is the number of observations. The model having the maximum BIC is the one to be selected from a given family of mixture models.

For high dimension given data where $p \gg n$, the penalty term, the 2nd term in (7.6), would dominate the likelihood term. Therefore, BIC is not suited to application to high-dimensional data.

Also, when the available data samples, i.e., spectral pixels, is small, compared to the number of parameters to be estimated, we could minimize the number of non-zero values in the mean vectors, $\boldsymbol{\mu}_g$, by maximizing a *lasso*-penalized log-likelihood given by:

$$\log \mathcal{L}_{pen}(\boldsymbol{\vartheta}|\boldsymbol{x}) = \log \mathcal{L}(\boldsymbol{\vartheta}|\boldsymbol{x}) - n\,\lambda_n \sum_{g=1}^{G} \pi_g \sum_{j=1}^{p} |\mu_{gj}| \tag{7.7}$$

where μ_{gj} is the jth element in $\boldsymbol{\mu}_g$ and λ_n is a tuning parameter that depends on n. In [103] and [102], the authors proved that this *lasso*-penalized likelihood would increase with each iteration of an expectation-maximization (EM) algorithm. Therefore, using the above results, the LPBIC was defined in [97] as

$$LPBIC = 2\log\mathcal{L}(\widehat{\boldsymbol{\vartheta}}|\boldsymbol{x}) - \tilde{\rho}\log n$$

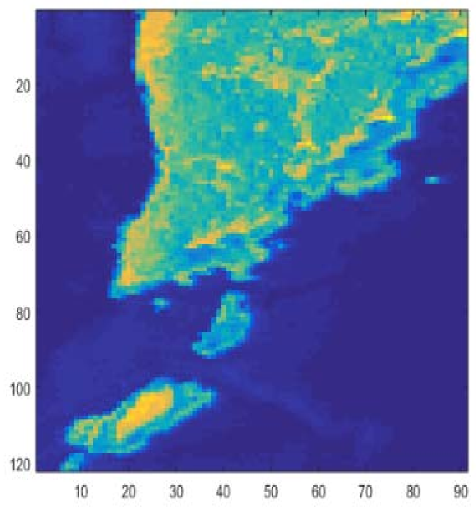
$$-2\log\mathcal{L}(\widehat{\boldsymbol{\vartheta}}|\boldsymbol{x}) - \frac{2n\,\lambda_n}{G}\sum_{g=1}^{G}\sum_{j=1}^{p_g}\left[|\hat{\mu}_{gj}| + \frac{I(\widehat{\boldsymbol{\mu}}_{\boldsymbol{g}})^{-1}{}_{jj}}{|\hat{\mu}_{gj}|} - sign(\hat{\mu}_{gj})\right] \tag{7.8}$$

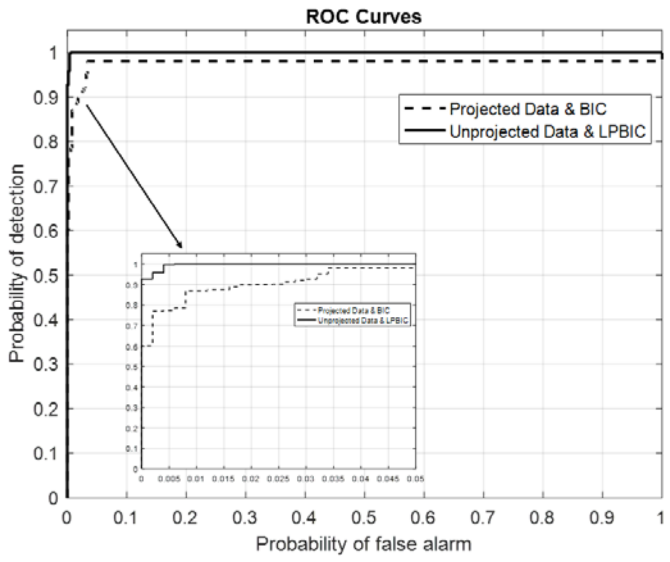
where $I(\widehat{\boldsymbol{\mu}}_{\boldsymbol{g}})$is the unit information matrix at $\widehat{\boldsymbol{\mu}}_{\boldsymbol{g}}$.

7.5 Anomaly detection experimental results

We present anomaly detection results using lower-dimensional spectral pixels whose dimensionality has been reduced, using Principal Component Analysis (PCA) [78], as a MoFAs, and then use BIC for model selection. We also present anomaly detection results from modeling higher dimensional spectral pixels as MoFAs, and using a *lasso*-penalized maximum likelihood estimation method, and finally using LPBIC for model selection.

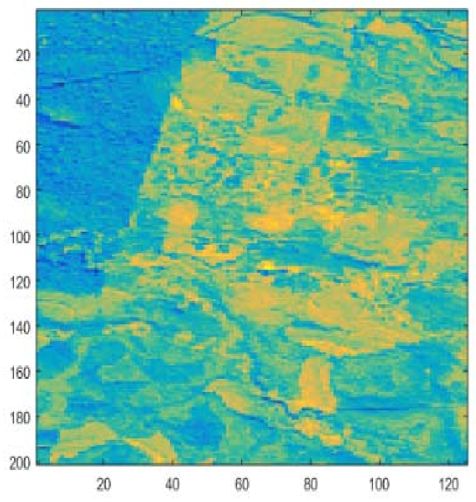
Based on the shown Receiver Operating Curves (ROCs), shown in Figure 7.2, Figure 7.3, Figure 7.4 and Figure 7.5 we note that our novel anomaly detection method results are more accurate than anomaly detection results using lower-dimensional spectral pixels whose dimensionality has been reduced, using Principal Component Analysis (PCA), and using BIC for model selection.

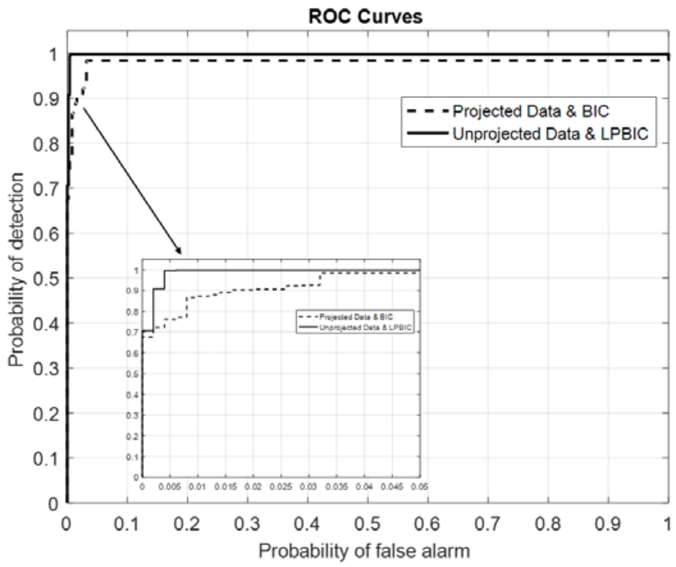




(a) (b)

Figure 7.2. (a) Hyperspectral image scene 1; (b) Receiver Operating Curve (ROC) using projected data and traditional BIC in dashed line and using unprojected data and lasso-penalized BIC in solid line.





(a) (b)

Figure 7.3. (a) Hyperspectral image scene 2; (b) Receiver Operating Curve (ROC) using projected data and traditional BIC in dashed line and using unprojected data and lasso-penalized BIC in solid line.

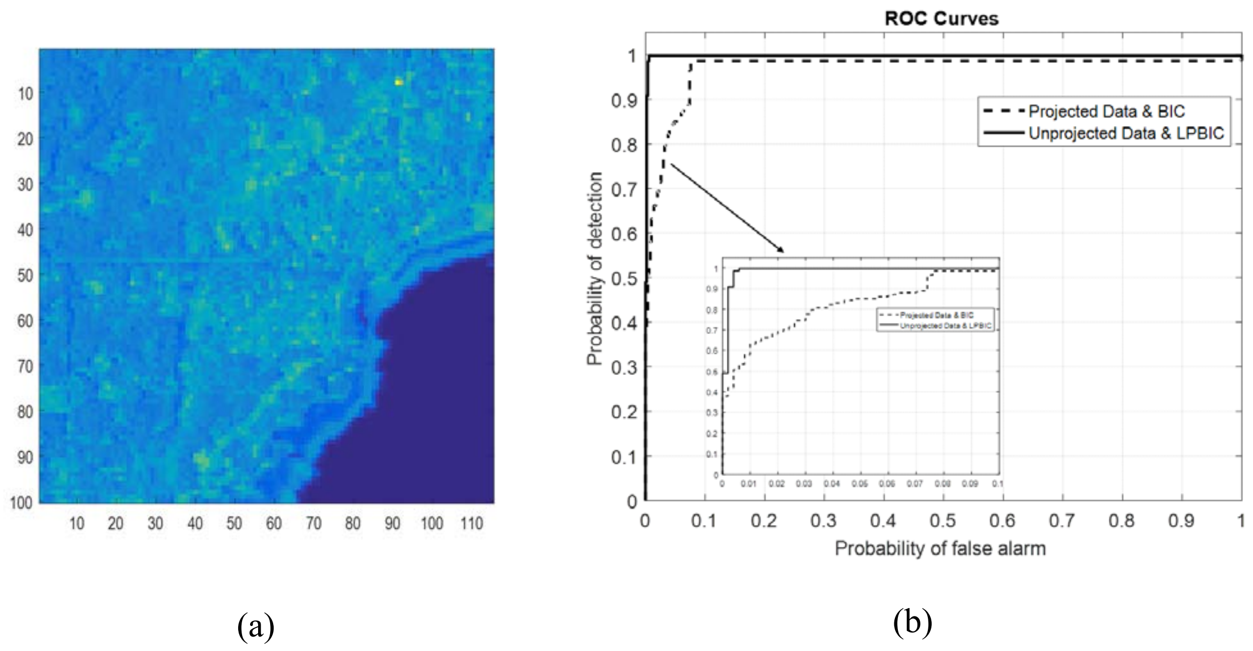


Figure 7.4. (a) Hyperspectral image scene 3; (b) Receiver Operating Curve (ROC) using projected data and traditional BIC in dashed line and using unprojected data and lasso-penalized BIC in solid line.

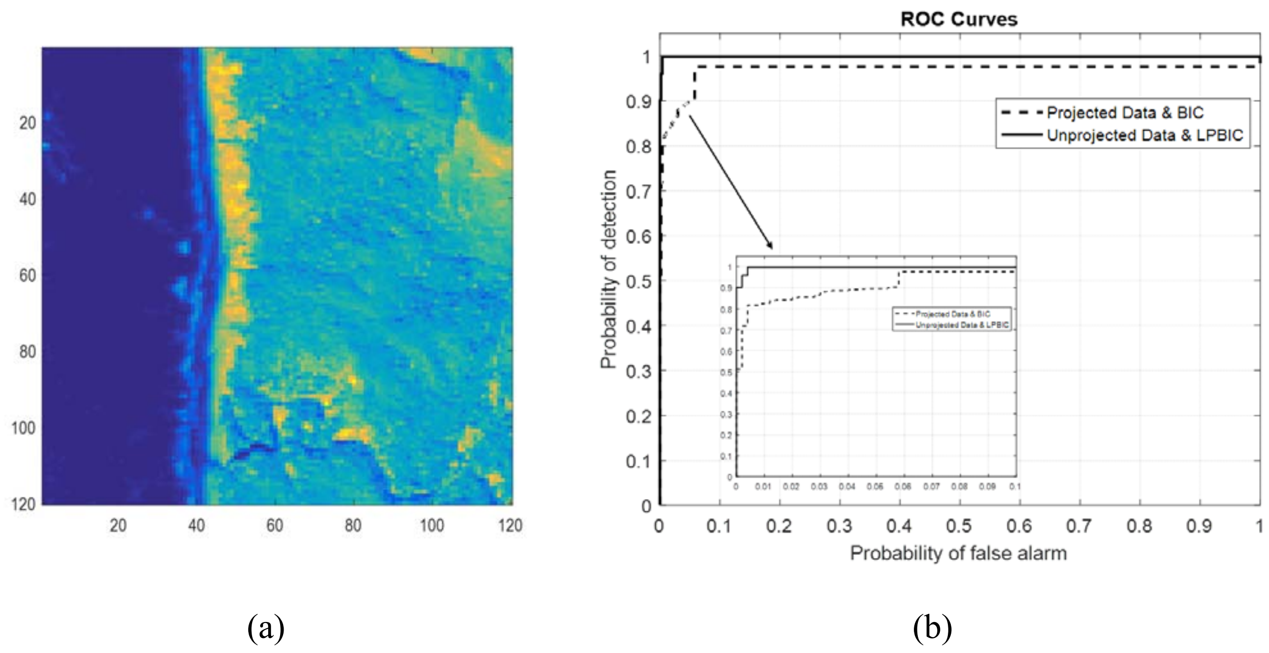


Figure 7.5. (a) Hyperspectral image scene 4; (b) Receiver Operating Curve (ROC) using projected data and traditional BIC in dashed line and using unprojected data and lasso-penalized BIC in solid line.

7.6 Chapter summary

In this chapter, we presented a novel high spatial resolution anomaly detection algorithm that requires a smaller number of spectral pixels in the window under test. To enable accurate modeling of complex backgrounds, e.g., near object boundaries, we used a mixture of factor analyzers (MoFAs) model that resulted in both *dimensionality reduction* and *Gaussian mixture modeling* in the *same* step. In this chapter, given a small number of spectral pixels comprising a small spatial window in a hyperspectral image, we estimated their MoFAs model parameters using a *lasso*-penalized maximum likelihood estimation. Also, instead of using the traditional Bayesian Information Criterion (BIC) for the model, i.e., model dimension and number of Gaussian mixture components, selection, we used the recently introduced *LASSO*-penalized Bayesian Information Criterion (LPBIC). To demonstrate the improved performance of our novel high-resolution anomaly detection algorithm, we compared its performance with an algorithm using reduced dimension spectral pixels and the traditional BIC for model selection, on *AVIRIS* hyperspectral images. Based on their resulting ROCs curves, our novel anomaly detection method is more accurate than anomaly detection using lower-dimensional spectral pixels whose dimensionality has been reduced, using Principal Component Analysis (PCA), and using BIC for model selection.

Chapter 8
Conclusion and Future Work

8.1 Conclusions

We developed the Kronecker LARS (K-LARS) algorithm to efficiently solve the matrix *basis pursuit* optimization problem for multichannel measurements and signals. This algorithm depends on the properties of Kronecker products.

We also developed the homotopy Kronecker algorithm as a dynamic version of the K-LARS algorithm that updates solutions for the matrix *basis pursuit* problem using a warm-start solution rather than a zero initial solution. This algorithm starts with an available solution of a slightly different problem and then solves a series of simple problems along the so-called homotopy path until the final solution of the original problem is reached.

We also described a sparsity-based unsupervised unmixing HSI algorithm that iteratively and alternately estimates endmembers and the abundances. Endmembers were estimated using one of these two algorithms: the K-LARS algorithm, or the Kronecker homotopy algorithm, where using the Kronecker homotopy algorithm was significantly faster than when using K-LARS. Also, the endmember abundances were estimated by solving standard constrained least-squares. These unmixing algorithms were applied to both synthetic HSI data cube and an actual *AVIRIS* HSI data cubes.

Furthermore, standard and custom learned dictionaries were compared for sparse representations of spectral pixels in a sparsity-based unsupervised unmixing algorithm.

The Coiflet dictionary was shown to be the most suitable dictionary for such a sparse representation of hyperspectral pixel spectra, as Coiflet has a large number of vanishing moments.

We also presented a novel high spatial resolution anomaly detection algorithm that requires a smaller number of spectral pixels in the window under test. Using Receiver Operating Curves (ROCs), we demonstrated that our novel anomaly detection method results are more accurate than anomaly detection results using lower-dimensional spectral pixels whose dimensionality has been reduced, using Principal Component Analysis (PCA), and using BIC for model selection.

8.2 Future work

One potential area for future work would be to do the spectral unmixing with the *Dantzing selector* [103] instead of the *basis pursuit* as an l_1-norm minimization problem. It may also be worthwhile to explore whether the homotopy path can be used to create a dynamic version of the *Dantzig selector*.

Another potentially fruitful area of inquiry would be to see if the dynamic version of the K-LARS algorithm can be applied to update acquired cube pixel spectra as though it were a video sequence. This would represent an advance that would enable the use of hyperspectral imaging in real-time applications as well as for anomaly detection.

Another possible area of future work is to incorporate spatial information of the hyperspectral imaged scene with K-LARS sparsity-based spectral unmixing.

References

[1] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, and A. Gualtieri, "Recent advances in techniques for hyperspectral image processing," *Remote Sensing of Environment,* vol. 113, pp. S110-S122, 2009.

[2] D. Manolakis, E. Truslow, M. Pieper, T. Cooley, and M. Brueggeman, "Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms," *IEEE signal processing magazine,* vol. 31, pp. 24-33, 2013.

[3] N. M. Nasrabadi, "Hyperspectral target detection: An overview of current and future challenges," *IEEE signal processing magazine,* vol. 31, pp. 34-44, 2013.

[4] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE journal of selected topics in applied earth observations and remote sensing,* vol. 5, pp. 354-379, 2012.

[5] Y. Moudden and J. Bobin, "Hyperspectral BSS using GMCA with spatio-spectral sparsity constraints," *IEEE Transactions on Image Processing,* vol. 20, pp. 872-879, 2010.

[6] I. Rish and G. Grabarnik, *Sparse modeling: theory, algorithms, and applications*: CRC press, 2014.

[7] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals using kronecker bases," *Neural computation,* vol. 25, pp. 186-220, 2013.

[8] D. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," *Lincoln laboratory journal,* vol. 14, pp. 79-116, 2003.

[9] *AVIRIS NASA Website*. Available: http://aviris.jpl.nasa.gov/alt_locator/

[10] R. Basedow, P. Silverglate, W. Rappoport, R. Rockwell, D. Rosenberg, K. Shu, R. Whittlesey, and E. Zalewski, "The *HYDICE* instrument design," in *Proceedings of the International Symposium on Spectral Sensing Research*, 1992, pp. 430-445.

[11] *Anonymous "NASA", "Hyperion Satellite Sensor"*. Available: http://edc.usgs.gov/products/satellite/eol1.html

[12] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification* vol. 1: Springer Science & Business Media, 2003.

[13] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE signal processing magazine,* vol. 19, pp. 44-57, 2002.

[14] A. Plaza, P. Martínez, R. Pérez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, pp. 650-663, 2004.

[15] K. J. Guilfoyle, M. L. Althouse, and C.-I. Chang, "A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 39, pp. 2314-2318, 2001.

[16] P. Comon, C. Jutten, and J. Herault, "Blind separation of sources, Part II: Problems statement," *Signal processing,* vol. 24, pp. 11-20, 1991.

[17] M. Parente and A. Plaza, "Survey of geometric and statistical unmixing algorithms for hyperspectral images," in *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, 2010, pp. 1-4.

[18] N. Dobigeon, S. Moussaoui, J.-Y. Tourneret, and C. Carteret, "Bayesian separation of spectral sources under non-negativity and full additivity constraints," *Signal processing,* vol. 89, pp. 2657-2669, 2009.

[19] C. Shi and L. Wang, "Incorporating spatial information in spectral unmixing: A review," *Remote Sensing of Environment,* vol. 149, pp. 70-87, 2014.

[20] M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," in *Imaging Spectrometry V*, 1999, pp. 266-275.

[21] J. W. Boardman, "Automating spectral unmixing of *AVIRIS* data using convex geometry concepts," 1993.

[22] J. M. Nascimento and J. M. Dias, "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 43, pp. 898-910, 2005.

[23] R. Neville, "Automatic endmember extraction from hyperspectral data for mineral exploration," in *International Airborne Remote Sensing Conference and Exhibition, 4 th/21 st Canadian Symposium on Remote Sensing, Ottawa, Canada*, 1999.

[24] L. Miao and H. Qi, "Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 45, pp. 765-777, 2007.

[25] M. Berman, H. Kiiveri, R. Lagerstrom, A. Ernst, R. Dunne, and J. F. Huntington, "ICE: A statistical approach to identifying endmembers in hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, pp. 2085-2095, 2004.

[26] J. D. Bayliss, J. A. Gualtieri, and R. F. Cromp, "Analyzing hyperspectral data with independent component analysis," in *26th AIPR Workshop: Exploiting New Image Sources and Sensors*, 1998, pp. 133-143.

[27] J.-L. Starck, F. Murtagh, and J. M. Fadili, *Sparse image and signal processing: wavelets, curvelets, morphological diversity*: Cambridge university press, 2010.

[28] D. L. Donoho and I. M. Johnstone, "Ideal denoising in an orthonormal basis chosen from a library of bases," *Comptes rendus de l'Académie des sciences. Série I, Mathématique,* vol. 319, pp. 1317-1322, 1994.

[29] M.-D. Iordache, A. Plaza, and J. Bioucas-Dias, "On the use of spectral libraries to perform sparse unmixing of hyperspectral data," in *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, 2010, pp. 1-4.

[30] M.-D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 49, pp. 2014-2039, 2011.

[31] Z. Yang, G. Zhou, S. Xie, S. Ding, J.-M. Yang, and J. Zhang, "Blind spectral unmixing based on sparse nonnegative matrix factorization," *IEEE Transactions on Image Processing,* vol. 20, pp. 1112-1125, 2010.

[32] Y. Qian, S. Jia, J. Zhou, and A. Robles-Kelly, "Hyperspectral unmixing via $ L_ {1/2} $ sparsity-constrained nonnegative matrix factorization," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 49, pp. 4282-4297, 2011.

[33] Z. Wu, S. Ye, J. Liu, L. Sun, and Z. Wei, "Sparse non-negative matrix factorization on GPUs for hyperspectral unmixing," *IEEE journal of selected topics in applied earth observations and remote sensing,* vol. 7, pp. 3640-3649, 2014.

[34] L. Tong, J. Zhou, X. Bai, and Y. Gao, "Dual graph regularized NMF for hyperspectral unmixing," in *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2014, pp. 1-8.

[35] W. Wang, Y. Qian, and Y. Y. Tang, "Hypergraph-regularized sparse NMF for hyperspectral unmixing," *IEEE journal of selected topics in applied earth observations and remote sensing,* vol. 9, pp. 681-694, 2016.

[36] J. Bobin, J.-L. Starck, J. Fadili, and Y. Moudden, "Sparsity and morphological diversity in blind source separation," *IEEE Transactions on Image Processing,* vol. 16, pp. 2662-2674, 2007.

[37] J. Rapin, J. Bobin, A. Larue, and J.-L. Starck, "Sparse and non-negative BSS for noisy data," *IEEE Transactions on signal processing,* vol. 61, pp. 5620-5632, 2013.

[38] C. Chenot, J. Bobin, and J. Rapin, "Robust sparse blind source separation," *IEEE Signal Processing Letters,* vol. 22, pp. 2172-2176, 2015.

[39] J. M. Bioucas-Dias and M. A. Figueiredo, "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing," in *2010 2nd*

Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2010, pp. 1-4.

[40] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*: Cambridge university press, 2012.

[41] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory,* vol. 53, pp. 4655-4666, 2007.

[42] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review,* vol. 43, pp. 129-159, 2001.

[43] E. Candes and J. Romberg, "l1-magic: Recovery of sparse signals via convex programming," *URL: www. acm. caltech. edu/l1magic/downloads/l1magic. pdf,* vol. 4, p. 14, 2005.

[44] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences,* vol. 2, pp. 183-202, 2009.

[45] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics & Data Analysis,* vol. 52, pp. 155-173, 2007.

[46] G. Shaw and D. Manolakis, "Signal processing for hyperspectral image exploitation," *IEEE signal processing magazine,* vol. 19, pp. 12-16, 2002.

[47] D. Landgrebe, "Hyperspectral image data analysis," *IEEE signal processing magazine,* vol. 19, pp. 17-28, 2002.

[48] B. Du and L. Zhang, "Random-selection-based anomaly detector for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 49, pp. 1578-1589, 2010.

[49] I. S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 38, pp. 1760-1770, 1990.

[50] S. Matteoli, M. Diani, and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images," *IEEE Aerospace and Electronic Systems Magazine,* vol. 25, pp. 5-28, 2010.

[51] D. G. Manolakis, "Taxonomy of detection algorithms for hyperspectral imaging applications," *Optical engineering,* vol. 44, p. 066403, 2005.

[52] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 32, pp. 779-785, 1994.

[53] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics,* vol. 32, pp. 407-499, 2004.

[54] D. S. Bernstein, *Matrix mathematics: theory, facts, and formulas*: Princeton university press, 2009.

[55] M. A. Hameed, *Comparative analysis of orthogonal matching pursuit and least angle regression*: Michigan State University, Electrical Engineering, 2012.

[56] T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley, "Least angle and ℓ1 penalized regression: A review," *Statistics Surveys,* vol. 2, pp. 61-93, 2008.

[57] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on signal processing,* vol. 58, pp. 5262-5276, 2010.

[58] Y. Hirose and F. Komaki, "An extension of least angle regression based on the information geometry of dually flat spaces," *Journal of computational and graphical statistics,* vol. 19, pp. 1007-1023, 2010.

[59] C. Fraley and T. Hesterberg, "Least angle regression and LASSO for large datasets," *Statistical Analysis and Data Mining: The ASA Data Science Journal,* vol. 1, pp. 251-259, 2009.

[60] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 68, pp. 49-67, 2006.

[61] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*: Chapman and Hall/CRC, 2015.

[62] D. L. Donoho and Y. Tsaig, "Fast Solution of $\ell _ {1} $-Norm Minimization Problems When the Solution May Be Sparse," *IEEE Transactions on Information Theory,* vol. 54, pp. 4789-4812, 2008.

[63] D. M. Malioutov, M. Cetin, and A. S. Willsky, "Homotopy continuation for sparse signal representation," in *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, 2005, pp. v/733-v/736 Vol. 5.

[64] M. R. Osborne, B. Presnell, and B. A. Turlach, "A new approach to variable selection in least squares problems," *IMA journal of numerical analysis,* vol. 20, pp. 389-403, 2000.

[65] S. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.

[66] J.-J. Fuchs, "On sparse representations in arbitrary redundant bases," *IEEE Transactions on Information Theory,* vol. 50, pp. 1341-1344, 2004.

[67] M. S. Asif and J. Romberg, "Dynamic Updating for l_1 Minimization," *IEEE Journal of selected topics in signal processing,* vol. 4, pp. 421-434, 2010.

[68] D. P. Bertsekas, "Nonlinear Programming. Athena Scientific Belmont," *Massachusets, USA,* 1999.

[69] R. J. Vanderbei, *Linear programming: Foundations and extensions*: Kluwer Academic Publishers, 1997.

[70] R. T. Rockafellar, "Convex Analysis Princeton University Press," *Princeton, NJ,* 1970.

[71] A. Elrewainy and S. S. Sherif, "Kronecker least angle regression for unsupervised unmixing of hyperspectral imaging data," *Signal, Image and Video Processing,* pp. 1-9, 2019.

[72] S. Mallat, *A wavelet tour of signal processing*: Elsevier, 1999.

[73] Y. Li and S. Osher, "Coordinate descent optimization for l1 minimization with application to compressed sensing; a greedy algorithm," *Inverse Problems and Imaging,* vol. 3, pp. 487-503, 2009.

[74] C. L. Lawson and R. J. Hanson, *Solving least squares problems* vol. 15: Siam, 1995.

[75] *ASTER Spectral Library*. Available: http://speclib.jpl.nasa.gov

[76] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing,* vol. 15, pp. 3736-3745, 2006.

[77] N. Dewanga and A. D. Goswam, "Image denoising using wavelet thresholding methods," *International Journal of Engineering Sciences & Management. Int. J. of Engg. Sci. & Mgmt.(IJESM),* vol. 2, pp. 271-275, 2012.

[78] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics,* vol. 2, pp. 433-459, 2010.

[79] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE,* vol. 98, pp. 1045-1057, 2010.

[80] I. Tosic and P. Frossard, "Dictionary learning: What is the right representation for my signal?," *IEEE signal processing magazine,* vol. 28, pp. 27-38, 2011.

[81] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications* vol. 31999: McGraw-Hill New York, 1986.

[82] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*: Academic press, 2014.

[83] E. Candes, L. Demanet, D. Donoho, and L. Ying, "Fast discrete curvelet transforms," *Multiscale Modeling & Simulation,* vol. 5, pp. 861-899, 2006.

[84] M. N. Do and M. Vetterli, "Contourlets: a directional multiresolution image representation," in *Proceedings. International Conference on Image Processing*, 2002, pp. I-I.

[85] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, 1999, pp. 2443-2446.

[86] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing,* vol. 54, pp. 4311-4322, 2006.

[87] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 689-696.

[88] *WaveLab 850*. Available: http://statweb.stanford.edu/~wavelab/

[89] S. M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*: Prentice-Hall, 1998.

[90] S. Matteoli, M. Diani, and J. Theiler, "An overview of background modeling for detection of targets and anomalies in hyperspectral remotely sensed imagery," *IEEE journal of selected topics in applied earth observations and remote sensing,* vol. 7, pp. 2317-2336, 2014.

[91] T. Veracini, S. Matteoli, M. Diani, and G. Corsini, "Fully unsupervised learning of Gaussian mixtures for anomaly detection in hyperspectral imagery," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, 2009, pp. 596-601.

[92] P. D. McNicholas, "On model-based clustering, classification, and discriminant analysis," *Journal of the Iranian Statistical Society,* vol. 10, pp. 181-190, 2011.

[93] Z. Ghahramani and G. Hinton, "The EM algorithm for factor analyzers," in *Technical Report CRG-TR-96-1*, ed: University of Toronto Toronto, 1997.

[94] P. D. Mcnicholas and T. B. Murphy, "Parsimonious Gaussian mixture models," *Statistics and Computing,* vol. 18, pp. 285-296, 2008.

[95] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Computational Statistics & Data Analysis,* vol. 71, pp. 52-78, 2014.

[96] G. Schwarz, "Estimating the dimension of a model Ann Stat 6: 461–464," *Find this article online,* 1978.

[97] S. Bhattacharya and P. D. McNicholas, "A LASSO-penalized BIC for mixture model selection," *Advances in Data Analysis and Classification,* vol. 8, pp. 45-61, 2014.

[98] D. Bartholomew and M. Knott, "Latent variable models and factor analysis, Kendalls, Library of Statistics, Vol. 7," *New York, NY: Edward Arnold,* 1999.

[99] X. L. Meng and D. Van Dyk, "The EM algorithm—an old folk-song sung to a fast new tune," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 59, pp. 511-567, 1997.

[100] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B (Methodological),* vol. 39, pp. 1-22, 1977.

[101] P. D. McNicholas, T. B. Murphy, A. F. McDaid, and D. Frost, "Serial and parallel implementations of model-based clustering via parsimonious Gaussian mixture models," *Computational Statistics & Data Analysis,* vol. 54, pp. 711-723, 2010.

[102] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *Journal of computational and graphical statistics,* vol. 9, pp. 1-20, 2000.

[103] E. Candes and T. Tao, "The Dantzig selector: Statistical estimation when p is much larger than n," *The Annals of statistics,* vol. 35, pp. 2313-2351, 2007.

[104] A. L. Lagopodi, A. F. Ram, G. E. Lamers, P. J. Punt, C. A. Van den Hondel, B. J. Lugtenberg, and G. V. Bloemberg, "Novel aspects of tomato root colonization and infection by Fusarium oxysporum f. sp. radicis-lycopersici revealed by confocal laser scanning microscopic analysis using the green fluorescent protein as a marker," *Molecular Plant-Microbe Interactions,* vol. 15, pp. 172-179, 2002.

[105] T. Tanaka, A. Hasegawa, Y. Matsuki, U.-S. Lee, and Y. Ueno, "Rapid and sensitive determination of zearalenone in cereals by high-performance liquid chromatography with fluorescence detection," *Journal of Chromatography A,* vol. 328, pp. 271-278, 1985.

[106] S. Delwiche, "Classification of scab–and other mold–damaged wheat kernels by near–infrared reflectance spectroscopy," *Transactions of the ASAE,* vol. 46, p. 731, 2003.

[107] J. M. Nascimento and J. M. Dias, "Does independent component analysis play a role in unmixing hyperspectral data?," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 43, pp. 175-187, 2005.

[108] J. Wang and C.-I. Chang, "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 44, pp. 1586-1600, 2006.

[109] M. A. Shahin and S. J. Symons, "Detection of Fusarium damaged kernels in Canada Western Red Spring wheat using visible/near-infrared hyperspectral imaging and principal component analysis," *Computers and Electronics in Agriculture,* vol. 75, pp. 107-112, 2011.

[110] A. Hyvärinen, J. Karhunen, and E. Oja, "ICA by maximization of nongaussianity," *Independent component analysis,* pp. 165-202, 2001.

[111] A. M. Bruckstein, M. Elad, and M. Zibulevsky, "A non-negative and sparse enough solution of an underdetermined linear system of equations is unique," *IEEE Trans. Inf. Theory,* vol. 54, pp. 4813-4820, 2008.

Appendix
Using ICA in Detecting Fusarium Infection on Wheat

Independent Component Analysis (ICA) of near-infrared hyperspectral imaging data was used to distinguish between sound samples of Canadian Western Red Spring Wheat and samples infected with *Fusarium graminearum*. These samples had moisture contents of 19%, 27%, and 35%, and the infected samples had 7 levels of infection ranging from 0 days to 56 days after manual *Fusarium* infection. Our hyperspectral imaging system acquires 256 images at different wavelengths equally spaced from 820 to 1666 nm. We demonstrate how to separate sound kernels from *Fusarium* damaged ones by applying ICA to the complete hypercube of imaging data.

1. Introduction

Fusarium is a worldwide spread fungus that commonly affects fruits and vegetables, but it is also one of the most serious fungi affecting small grains [104]. Fusarium head blight (also called scab) is a disease caused by Fusarium, which has been found in wheat, barley, and maize. Some species of Fusarium produce toxic metabolites such as nivalenol, zearalenone, and deoxynivalenol (DON) [105]. DON, also known as vomitoxin, has a toxic effect on humans and animals, causing vomit, skin damage, and loss of weight. The U.S. Food and drugs administration (FDA), for example, recommends that total levels of DON not exceed 10 ppm for cattle and chicken, 5 ppm for swine, and less than 1 ppm for humans in finished products (U.S. Department of Health and Human Services 2010). Although scab damage does not necessarily indicate the presence of DON, it is considered a physical defect that could cause grains to be downgraded during official inspection [106].

Over the last few decades, optical techniques combined with signal unmixing have been used to identify components in the scene, also known as endmembers [14]. Among those techniques, spectral imaging has some advantages and has been constantly used for grain screening. Independent Component Analysis (ICA) is a technique that is used for signal separation with neither prior knowledge of these signals nor the process that mixed them. In this chapter, ICA is used as a spectral unmixing technique for classifying sound and Fusarium damaged wheat kernels. ICA is applied to the entire hypercube data; the results for the full dataset are presented in section 4.

2. Independent component analysis (ICA)

ICA is a signal unmixing technique that could also be used for dimensionality reduction [107]. Input data are projected on independent bases, which are also called independent components (ICs). ICA assumes that a set of separate independent sources $\boldsymbol{S}$ are linearly mixed by a mixing matrix $\boldsymbol{A}$, resulting in a mixed-signal source x, i.e., $\boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{S}$. The vector $\boldsymbol{x}$ is the signal acquired by the acquisition system. The task is to find all the source signals $\boldsymbol{S}$ without any knowledge of $\boldsymbol{A}$. The unmixing procedure relies on the lack of statistical dependency between the sources measured by mutual information. An assumption is that, at most, one source in the mixture model is allowed to be a Gaussian source due to the fact that a linear mixture of Gaussian sources is still a Gaussian source. The purpose of the ICA is to find an unmixing matrix $\boldsymbol{w}$ that separates the signal source vector into a set of sources which are statistically independent. In ICA, the equation is written as:

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} \tag{1}$$

where $\boldsymbol{W}$ is the inverse of $\boldsymbol{A}$.

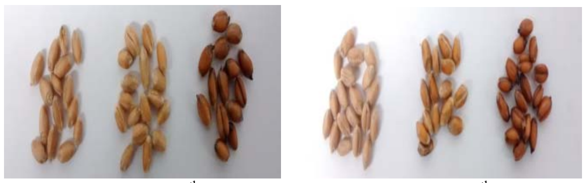
Independent components may exhibit only one endmember, but they could also be a mixture of two or more endmembers. One immediate issue with ICA is how to rank independent components in terms of significance since these ICs are generated by random initial projection vectors. Consequently, the ICs generated earlier are not necessarily more significant than those generated later. Unlike PCA, which orders principal components according to their eigenvalues, ICA does not have a criterion for prioritizing the order in which ICs are generated. So, the ICA must find a criterion higher than variance to measure the significance of each independent component. The selection of an IC, then, is based on its score as produced by the measure. Two measures, skewness and kurtosis, are of particular interest and could be used to produce such a score for each IC [108].

3. Methodology

3.1 Sample preparation

This study used Canadian Western Red Spring (CWRS) wheat grains that were sterilized using a 1% sodium hypochlorite solution before being rinsed with distilled water. The grains were then allowed to dry under a fume hood before being conditioned through the addition of distilled water in order to attain the predetermined moisture levels of 19%, 27%, and 35%. After conditioning, the samples were stored in a refrigerator at 5°C for 7 days to allow the moisture to equilibrate. For each moisture level, conditioned samples were split into two portions: a control group and Fusarium damaged kernels (FDK).

Spore suspension of *F.graminearum* was prepared by the Department of Plant Science at the University of Manitoba. An initial spore suspension count was carried out using a hemocytometer and adjusted to a concentration of 10^5 spores/ml. The artificial infection was done by misting the spore suspension on sterile conditioned wheat samples and mixed thoroughly in the fume hood. Samples were then stored in an environmental chamber at 25±1°C for the growth of *Fusarium*. Figure 1 displays infected kernels for different moisture contents on the 14th day and 56th day after infection.



(a) After the 14th day (b) After the 56th day

Figure 1. Fusarium infected kernels (a) after the 14th day and (b) after the 56th day with 19%, 27%, and 35% moisture contents for both kernels.

The experiments were carried out with grains from 0th, 14th, 21th, 28th, 35th, 42nd, and 56th days after manual infection. Sound grains were kept in the same environment as the FDK (the same controlled temperature and humidity) and were placed alongside them in a sample holder containing 9 grains – 5 control and 4 FDK, using 3 rows by 3 columns arrangement. The control samples were placed at the edges and at the center of the matrix, while FDK formed a diamond pattern at the remaining positions, as shown in Figure 2. This grain distribution was repeated for different moisture levels and for each infection range.

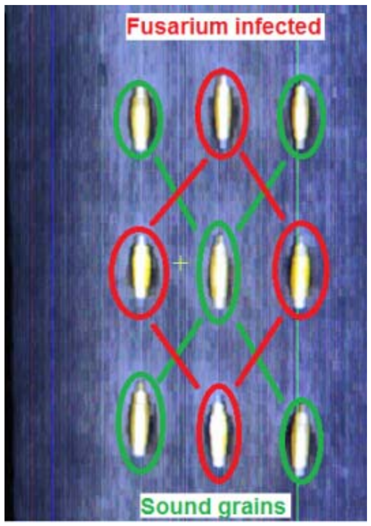


Figure 2. Configuration of infected and sound grains.

3.2 Hyperspectral data acquisition

A near-infrared (NIR) hyperspectral system composed of an InGaAs camera (Xenics Xeva 1.7-320), a spectrometer (Specim Inspector N17E) and a spectrometer lens (Specim S22.5-f/2.0) were used as shown in Fig.3. This system was able to acquire 256 wavelengths between 820 and 1666 nm, resulting in a spectral resolution of ~3.3 nm, and an approximate spatial resolution of 0.15 mm on the first spatial dimension, due to the field of view of 5 cm. A translational stage mounted under the sample holder scanned a 5 cm path in 441 steps with 0.1 mm of spatial resolution on the second spatial dimension. The output from a halogen lamp was guided to the sample through an optical fiber bundle. A warm-up period of two hours was allowed before any measurements were taken. For normalization and correction procedures, white and dark reference hypercubes were also collected. For the white reference hypercube, a Spectralon panel with 99%

reflection was used in place of the sample, while the dark reference hypercube was recorded by closing the lens cap. The reflectance was calculated by this equation [109]:

$$\boldsymbol{R} = \frac{\boldsymbol{I_{raw}} - \boldsymbol{I_{dark}}}{\boldsymbol{I_{white}} - \boldsymbol{I_{dark}}} \tag{2}$$

where I_{raw} is the image of the sample, I_{dark} is the dark current image and I_{white} is the image of the white reference.

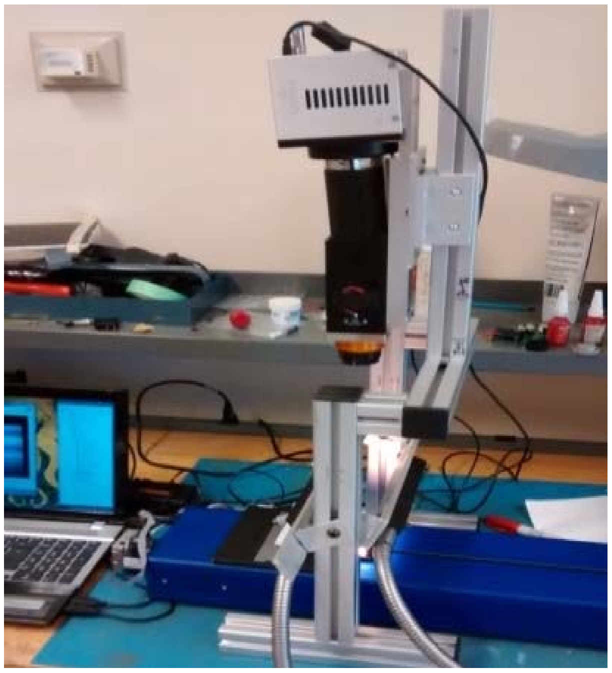


Figure 3. The hyperspectral imaging system.

3.3 Implementation of the independent component analysis algorithm

The used ICA routine utilized the Fast ICA algorithm as it has a faster convergence time than traditional gradient-based methods [110]. There are two methods of estimating the rows of $\boldsymbol{W}$ or the ICs; symmetric orthogonalization (parallel) and deflationary (sequential) orthogonalization. The symmetric orthogonalization method was used, where vectors w_i, which are the rows of $\boldsymbol{W}$, are not estimated one-by-one but in parallel. One drawback to the sequential method is that estimation errors that occur while generating the first vectors tend to be carried over to subsequent ones by orthogonalization. Hence, symmetric orthogonalization methods enable parallel computation of ICs. In symmetric orthogonalization, the data are first centered and whitened. The number of estimated independent components was chosen according to the dimension of the input; in this case, it's the number of bands which is provided to the ICA algorithm.

For every $\boldsymbol{i}$, the values $\boldsymbol{w}_i$ is initiated and updated by:

$$\boldsymbol{w}_i \leftarrow E\{\boldsymbol{z}g(\boldsymbol{w}_i^T\boldsymbol{z})\} - E\{g'(\boldsymbol{w}_i^T\boldsymbol{z})\}\boldsymbol{w}_i \qquad (3)$$

where $g(y) = tanh\ (y)$.

Orthogonalization of the matrix $\boldsymbol{W}$ was performed by:

$$\boldsymbol{W} \leftarrow (\boldsymbol{W}\boldsymbol{W}^T)^{-1/2}\boldsymbol{W} \qquad (4)$$

The iteration is repeated until convergence which means that the old and new values of $\boldsymbol{w_i}$ are the same or pointing in the same direction.

4. Results

Figure 4(a) shows a picture of sound and Fusarium damaged kernels, circled in red, displaced in the pattern described before. Although manual infection had been uniformly implemented for all grains, not all the FDK developed superficial mold. This is likely due to either a small progression of the fungi or asymptomatic infection.

Independent component images displayed several combinations when the entire hypercube was used in the ICA algorithm. While some of those images were mainly composed of noise, where no grains were displayed, others displayed only sound grains, and other groups showed only infected grains. However, the majority of independent components images displayed all the grains from the scene. This behavior is reasonable since all grains share common characteristics. One independent component is shown in Figure 4(b) displays FDK mixed with the background Using MatLab ‘jet’ colormap. Another independent component is shown in Figure 4(c) shows sound kernels as cold color pixels (green and blue) Using MatLab ‘jet’ colormap.

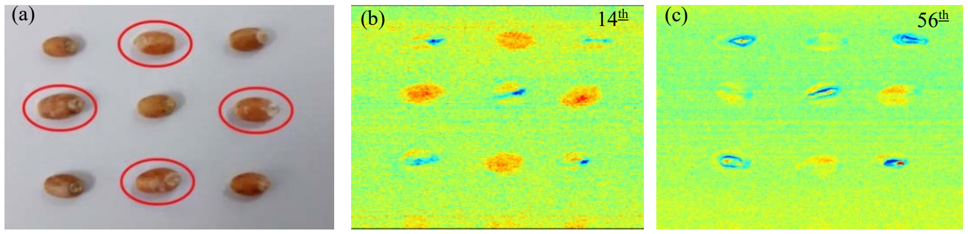


Figure 4. (a) Placement of the wheat kernels on the sample holder (infected grains are circled), (b, c) ICA component showing separation from the 14th and 56th days after infection (35% moisture).

5. Limitations of ICA

Despite its theoretical strength and elegance, ICA suffers from several limitations:

- ICA is based on the assumption of mutually independent sources, which is not the case of hyperspectral data since the sum of abundance fractions is constant, implying statistical dependence among them. This dependence compromises ICA applicability to hyperspectral data, as shown in [4].
- The ICA algorithm assumes that the mixing matrix $\boldsymbol{A}$ to be square (number of measurements equal to the number of endmembers), which is not the case of hyperspectral data in practice as the number of measurements is much greater than the number of endmembers.
- ICA is very sensitive to noise, especially when considering non-gaussian noise.

- ICA is highly dependent on statistical distributions of sources as the ICA algorithm requires information about the sources distributions, the umixing matrix (and the thus the sources) is estimated closely depends on the way the sources are modeled [111].

6. Summary

Independent component analysis as an example of statistical hyperspectral unmixing approaches was able to separate severe *Fusarium* damaged kernels from control samples. For samples with moisture contents of 27% and 35%, the infection could be identified after the 14th day. Hence, kernels with little or no visual infection were correctly identified as *Fusarium* infected samples. ICA finds a linear decomposition of the observed data by maximizing the nongaussianity of the components yielding statistically independent components based on the assumption of mutually independent sources. ICA has some limitations to be used in hyperspectral applications.