Hyperspectral Super-resolution Using Matrix Factorization

CHAN, Chun Hei

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Electronic Engineering

The Chinese University of Hong Kong

September 2017

Abstract

Hyperspectral imaging systems have recently drawn significant interest in areas such as computer vision, signal processing, and remote sensing because of their ability to provide rich information in both spectral and spatial domain of a scene. These imaging systems acquire spectral images from far infrared to visible range with many spectral bands, resulting in images that have a lot more spectral bands than that of the conventional multispectral images. Due to hardware limitation, spectral cameras with high resolution in both spectral and spatial domain are uncommon, especially in the application of remote satellite imagery. The sensors are usually called hyperspectral camera and multispectral camera.

Hyperspectral super-resolution is a problem of fusing a hyperspectral image and a multispectral image of the same scene to form a full spectral image having the high spectral resolution of the former and high spatial resolution of the latter. In the existing methods, some of them can be classified as matrix factorization-based methods in which the desired full spectral image can be expressed as the product of an endmember matrix and an abundance matrix. The objective of this family of methods is to estimate the endmember-abundance matrix pair solely from the hyperspectral and multispectral observations. Conventionally, matrix factorization requires alternating optimization with respect to each factor matrix individually. This makes the hyperspectral super-resolution problem an iterative procedure of solving subproblems with respect to the endmember matrix and abundance matrix alternatingly.

In this thesis, some matrix factorization-based formulations of the hyperspectral super-resolution problem are investigated. In each of the investigation there are two major focus. The first focus is on the effect of exactness of solving the subproblems during alternating optimization against the fusion and unmixing performance, measured by both reconstruction accuracy of full spectral images and the unmixing accuracy of the endmember-abundnace matrix pair. The second focus is on the fusion and unmixing performance using a variety of first order optimization methods for solving the subproblems.

The contribution of this thesis is that a framework of alternating optimization on matrix factorization is developed. Simulation results show that by inexactly solving the subproblems, the hyperspectral super-resolution problem can still be solved to comparable accuracy but with much less computation time than the state of the art.

摘要

近年來，由於超光譜圖 (hyperspectral image) 蘊含豐富的光頻與空間訊息，超光譜圖成像系統 (hyperspectral imaging system)在計算機視訊、訊號處理與遙距感應 (remote sensing) 等領域備受關註。由於超光譜圖成像系統拍攝的圖像包含了許多由遠紅外光頻域至可見光頻域的光譜帶 (spectral band)，超光譜圖 (hyperspectral image) 相較於傳統的多光譜圖 (multispectral image) 擁有更多光譜帶。因硬件所限，在遙感衛星圖成像系統中，在光頻域和空間域中同時具有高解像度的光頻相機並不普及。所以，傳統的光頻相機不是擁有高光譜－低空間解像度，就是擁有低光譜－高空間解像度。

超光譜超解像度問題 (hyperspectral super-resolution) 是一個融合同一場景所觀測之超光譜圖和多光譜圖的問題，目的為整合兩圖的訊息以構成一幅具有前者之高光譜解像度和後者之高空間解像度的超解像度圖 (super-resolution image)。現有的一種做法是基於矩陣因子分解 (matrix factorization) 來進行超光譜圖的融合。此做法假設超解像度圖可表達為端元矩陣 (endmember matrix) 與豐度矩陣 (abundance matrix) 的積。所以，這類超光譜超解像度的問題，便轉化為從某一場景之超光譜圖和多光譜圖中估算端元矩陣－豐度矩陣對 (endmember and abundance matrix pair) 的問題。一般來說，矩陣因子分解問題需依靠對於每個因子矩陣的交替優化 (alternating optimization) 來解決。因此，通過解決端元矩陣和豐度矩陣的交替優化問題，超光譜超解像度問題可得到解決。

對於超光譜超解像度問題，本論文探討了一些基於矩陣因子分解而建模 (matrix factorization-based formulation) 的方法，並著重探討它們在交替優化算法中，解決子問題的精確性，與超解像度圖的融合誤差和其端元與豐度的分離誤差的關係。此外，我們也著重探討解決子問題時使用不同的一階優化算法，對該融合和分離性能的影響。

本論文對於矩陣因子分解問題提出了一個交替優化的算法框架。模擬結果表明，非精確地解決子問題的做法在解決超光譜超解像度問題中，仍可得出比擬現有算法精確性的結果，但本論文提出的算法框架下的算法所耗的運算時間卻是遠少於現有算法。

Acknowledgements

First of all, I would like to greatly thank my supervisor, Prof. Wing-Kin Ma, for his guidance and advice in my research throughout the past few years in which I have received great amount of freedom, patience, tolerance and opportunities in learning and exploring signal processing and remote sensing.

I would also like to thank all my friends in the Digital Signal Processing and Speech Technology Laboratory (DSPST Lab) including Dr. Jiaxian Pan, Prof. Xiaoxiao Wu, Prof. Xiao Fu, Mr. Hoi To Wai, Dr. Yun Liu, Dr. Ran Cai, Dr. Wai Man Ng, Mr. Shing Yu, Dr. Yu Ting Yeung, Dr. Wang Kong Lam, Ms. Lufei Gao, Mr. Chun Hoy Wong, Mr. Man Wai Un, Mr. Ruiyuan Wu, Mr. Mingjie Shao, Mr. Siyuan Feng, Ms. Ying Qin, Ms. Qiong Wu, Mr. Yatao Liu, Mr. Yuzhong Wu, Ms. Man Ling Sung, Mr. Shuiyang Mao, Mr. Xiongwei Wu and Mr. King Hang Matthew Ma for their encouragement and helpful discussions in technical materials, research-related subjects and computation facilities issues. Especially, I must thank Mr. Hoi To Wai, Prof. Xiao Fu and Mr. Ruiyuan Wu for all of the accademic interaction and support in many aspects. Their kind support have been helping me a lot in overcoming the difficulties in my research. Besides, I would like to thank Dr. Yu Ting Yeung for guiding me using Linux as the core scientific computation platform which is vital to processing and maintaining all those massive experiments in the research.

Lastly, I would like to thank our laboratory technician, Mr. Arthur Luk, for his unlimited support to my studies and the establishment of many high-speed computation servers which are the keystone of the entire thesis research.

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**
1.1 Overview of the Hyperspectral Super-resolution (HSR) 2
1.2 Overview of Convex Optimization 4
1.2.1 Convex Sets 4
1.2.2 Convex Functions 4
1.2.3 Convex Optimization Problems 5
1.3 Significance of Convex Optimization 5
1.4 Coordinate Descent Algorithms 6
1.5 Contributions 7
1.6 Organization 7

2 The HSR Problem and The State-of-the-art Algorithms **8**
2.1 Signal Model of HS and MS Sensors 8
2.1.1 Spatial Degradation Model in HS Sensors 9
2.1.2 Spectral Degradation Model in MS Sensors 10
2.2 Low-Rank Representation of HSIs: the Linear Mixing Model (LMM) 12
2.3 The HSR Problem 13
2.4 The State-of-the-art HSR Formulations and Approaches 14
2.4.1 HSR Formulation and Approach by Yokoya *et al.* 14
2.4.2 HSR Formulation and Approach by Wei *et al.* 17
2.4.3 General Review on the State of the Arts 21

3 HSR with First-Order Gradient Methods (FOGMs) **22**
3.1 HSR Problem as Convex Constrained QPs 22
3.2 Projection onto a Convex Set 24
3.3 Solving QPs with BCD using FOGMs 25
3.3.1 Gradient Projection Method (GP) 25
3.3.2 Proximal Gradient Method (PG) 31
3.3.3 Fast Proximal Gradient Method (FISTA) 34
3.3.4 Frank-Wolfe Method (FW) 35
3.4 Proposed Algorithm Framework for HSR:
ALternating Gradient-based Optimization (ALGO) 37

3.4.1 ALGO using GP . . . 39
3.4.2 ALGO using BBGP . . . 40
3.4.3 ALGO using PG . . . 41
3.4.4 ALGO using FISTA . . . 44
3.4.5 ALGO using FW . . . 45
3.4.6 ALGO using Hybrid BCD Method . . . 46

4 Simulations 48
4.1 Experimental Details . . . 48
4.1.1 Wald's Protocol . . . 49
4.1.2 Spatial Degradation . . . 49
4.1.3 Spectral Degradation . . . 49
4.1.4 Synthetic Data Generation . . . 50
4.1.5 Semi-real Data Generation . . . 50
4.1.6 Algorithm Initialization . . . 52
4.1.7 Stopping Criteria of Algorithms . . . 53
4.1.8 Algorithm Implementation . . . 53
4.1.9 HU and HSR Quality Measures . . . 54
4.2 Experiment 1: Benchmark HSR via Exact BCD and Inexact BCD . . . 56
4.3 Experiment 2: Benchmark HSR via ALGO using FOGMs . . . 57
4.4 Experiment 3: Benchmark HSR via ALGO and FUMI . . . 59
4.4.1 Results on Synthetic Data . . . 59
4.4.2 Results on Semi-real Cuprite Dataset . . . 60
4.4.3 Results on Semi-real Chikusei Dataset . . . 64
4.5 Experiment 4: Benchmark HSR via ALGO and CNMF . . . 67
4.5.1 Results on Semi-real Chikusei Dataset . . . 67

5 Conclusion and Discussion 69

A MATLAB Code of ALGO for Solving Wei et al.'s HSR Problem 71
A.1 ALGO using GP . . . 71
A.2 ALGO using BBGP . . . 73
A.3 ALGO using PG . . . 75
A.4 ALGO using FISTA . . . 76
A.5 ALGO using FW . . . 78
A.6 ALGO using Hybrid BCD . . . 80

B Full Synthetic Simulations Results 82
B.1 Performance of ALGO Against the Exactness in solving Subproblems . . . 83
B.2 Performance of ALGO Against the Choice of FOGMs . . . 98
B.3 Algorithm Speed of ALGO-FISTA and ALGO-FW Against Model Order . . . 115
B.4 Algorithm Speed of ALGO-FISTA and ALGO-FW Against Pixel Number . . . 117

Bibliography 118

List of Algorithms

1 Coordinate Descent Algorithm 6
2 Lee-Seung Algorithm for Nonnegative Matrix Factorization (NMF) 15
3 Coupled Nonnegative Matrix Factorization (CNMF) 17
4 Alternating Direction Method of Multipliers (ADMM) 18
5 Fusion based on Unmixing for Multiband Images (FUMI) 20
6 Gradient Projection (GP) Method (case a) 30
7 Barzilai-Borwein Gradient Projection (BBGP) Method (case c) 30
8 Proximal Gradient (PG) Method 33
9 Fast Proximal Gradient Method (FISTA) 34
10 Frank-Wolfe Method (FW) 37
11 Algorithm Framework of HSR via *Exact BCD* by FOGM 37
12 ALGO (Algorithm Framework of HSR via *Inexact BCD* by FOGM) 38
13 ALGO using GP 39
14 ALGO using BBGP 40
15 ALGO using PG 43
16 ALGO using FISTA 44
17 ALGO using FW 45
18 ALGO using Hybrid BCD 47
19 Successive Projection Algorithm (SPA) 52

Chapter 1

Introduction

A spectral image contains rich information in spectral domain by capturing the electromagnetic (EM) reflectance at a number of wavelengt in each pixel. An ordinary color camera, for example, captures the EM reflectance at three broad spectral bands roughly at the wavelength corresponding to red ($0.6\,\mu$m-$0.68\,\mu$m), green ($0.53\,\mu$m-$0.57\,\mu$m) and blue ($0.42\,\mu$m-$0.48\,\mu$m) in the visible range. A spectral camera, rather than only receiving signals at few spectral bands, captures over a hundred spectral bands spanning the visible range and the infrared region.

In remote sensing and Earth science applications, a popular hyperspectral sensor named "Airborne Visible/InfraRed Imaging Spectrometer" (AVIRIS) captures hyperspectral images (HSIs) at 224 spectral bands covering wavelengths between $0.4\,\mu$m and $2.5\,\mu$m at norminally 10 nm intervals. Usually the AVIRIS sensor is placed at 20 km altitude inspecting the Earth surface at about 20 m ground sampling distance (GSD), which is defined as the actual ground distance between neighbouring pixels in the HSI [1–3]. Suppose the spatial dimension of HSIs taken by a AVIRIS sensor has $m \times n$ pixels and the spectral dimension contains λ bands. The resultant HSI is an image cube with $m \times n \times \lambda$ values, as illustrated in Figure 1.1.

Hyperspectral unmixing and classification belong to a class of quantitative analysis techniques that retrieve useful information from the pixels of HSIs. These analysis have great potential in applications where the image scenes tend to be large and complex, such as remote sensing, Earth science, industrial resource exploration, aircraft-based systems, agricultural science and wide area monitoring, or in small scale laboratory-based studies like food quality analysis, chemical spectroscopy, biomedical science and forensic science [4–8], to mention a few.

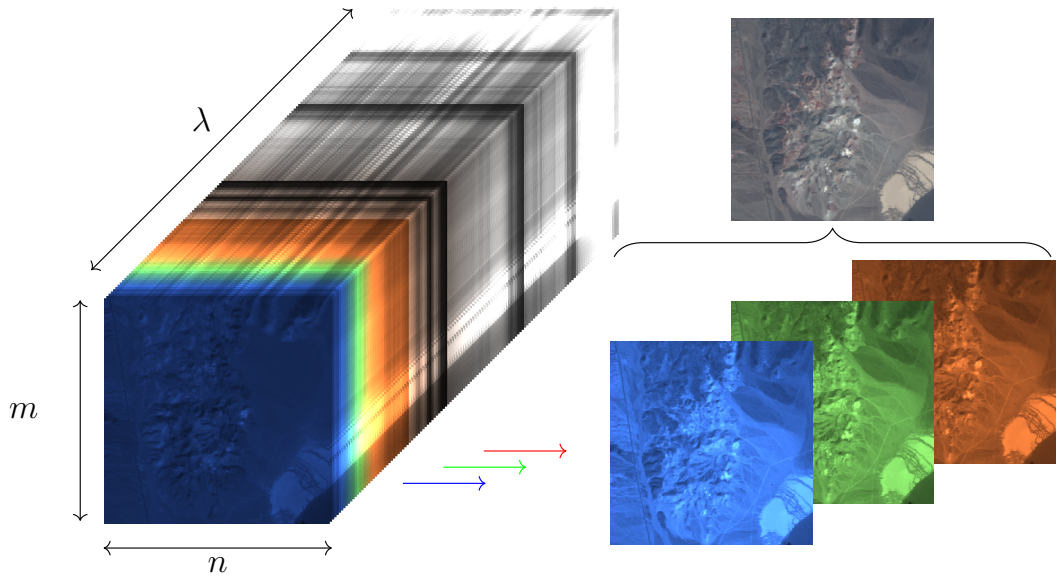


Figure 1.1: HSI over an Earth surface and its conversion to ordinary color image.

Usually an enhancement in the quality of an HSI in terms of spatial and spectral resolution is beneficial to all hyperspectral imagery applications. In this thesis, the enhancement of HSIs in spatial and spectral quailty, called hyperspectral super-resolution (HSR), for remote sensing purpose is discussed.

1.1 Overview of the Hyperspectral Super-resolution (HSR)

In geoscience and Earth science research, a number of industries have launched satellites for Earth observation missions. These projects include Sentinel, Landsat, RapidEye, QuickBird, EnMAP, AVIRIS, HYPXIM, CHRIS, HISUI, Hyperion, DESIS, PRISMA, SHALOM, etc. The resolution characteristics of the aforementioned wide spectrum of remote sensing instruments are summarized in Figure 1.2 [9] in which more desirable sensors characteristics are marked in the purple region; the hyperspectral (HS) and multispectral (MS) sensors are marked by the blue dots and red dots, respectively. From the figure we see that an ideal hyperspectral sensor should have small GSD and large amount of spectral bands within a desired wavelength range. However, in practice, hyperspectral sensors with such high spatial and spectral resolution are not common due to hardward limitation, i.e., we are inevitably required to have a trade-off between the two resolutions.

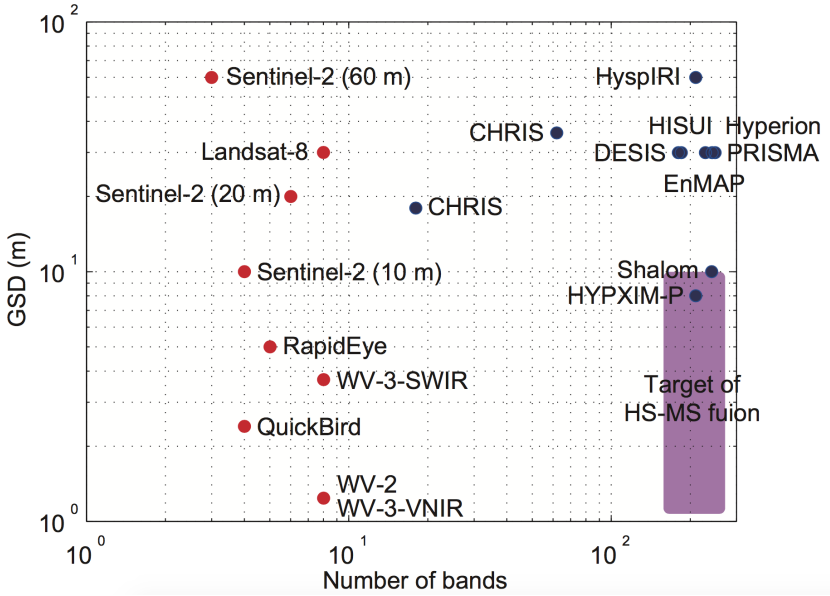


Figure 1.2: Resolutions of the latest remote satellite sensors [9].

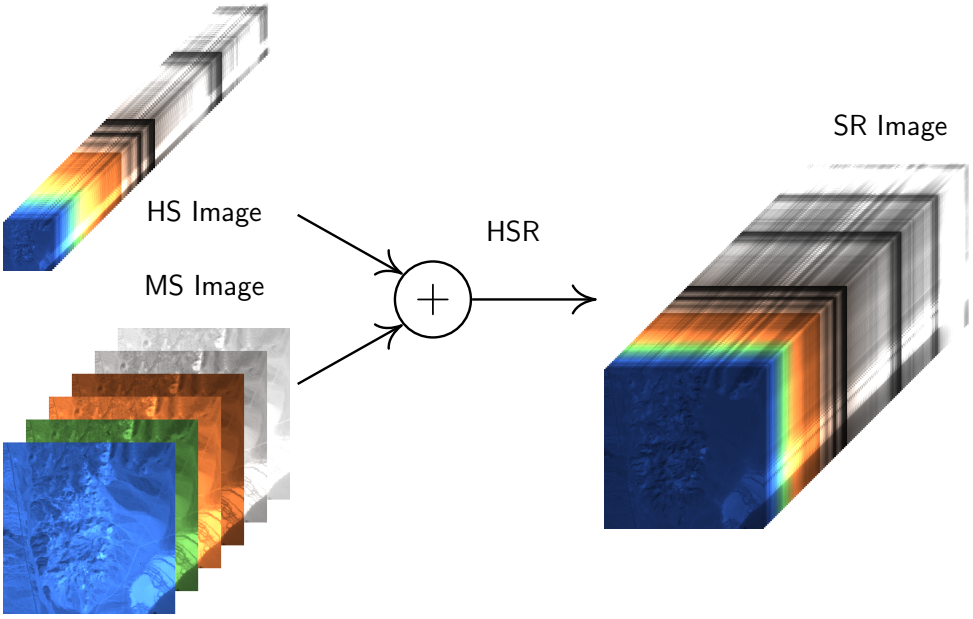


Figure 1.3: Hyperspectral super-resolution from low resolution HS and MS images.

In this thesis, for ease of presentation, we define images with high spectral resolution and low spatial resolution as hyperspectral (HS) images; images with low spectral resolution and high spatial resolution as multispectral (MS) images; images with high resolution in both spatial and spectral domain as super-resolution (SR) images. HSR estimates the SR image of a scene from its HS and MS observations such that the combined image at most possesses the spectral resolution of the former and spatial resolution of the latter (Figure 1.3). Many

approaches have been studied including Component Substitution (CS) [10–12], Multiresolution Analysis (MRA) [13, 14], Bayesian approach [15–17] and spectral unmixing [18–23]. The competitiveness of these classes of methods have been summarized and reviewed in [9, 24]. In this thesis, we focus on HSR by spectral unmixing.

1.2 Overview of Convex Optimization

Some basic concepts of convex optimization are reviewed in this section. First of all, a few definitions on convex sets and convex functions are given before further discussion on convex optimization problems.

1.2.1 Convex Sets

A set $\mathcal{S} \subseteq \mathbb{R}^N$ is convex if the line segment drawn from any two points in $\mathcal{S}$ lies in $\mathcal{S}$, i.e., $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$,

$$\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y} \in \mathcal{S}, \quad \forall\, \alpha \in [0,1]. \tag{1.1}$$

Some examples of convex sets are given as

- $\mathbb{R}^n_+$ (nonnegative orthant);
- $\{\boldsymbol{x} \in \mathbb{R}^2_+ \mid x_1 - 3x_2 \leq 0, -2x_1 + 3x_2 \leq 0\}$ (a convex cone);
- $\{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_2 \leq 1\}$ (unit ℓ_2-norm ball);
- $\{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_1 \leq 1\}$ (unit ℓ_1-norm ball);
- $\{\boldsymbol{x} \in \mathbb{R}^n \mid |x_i| \leq 1\}$ (a box).

1.2.2 Convex Functions

A function $f : \mathcal{S} \to \mathbb{R}$ is said to be convex if $\mathcal{S}$ is a convex set and f satisfies

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}), \quad \forall\, \boldsymbol{x}\ ,\ \boldsymbol{y} \in \mathbb{R}^N\ ,\ \alpha \in [0,1]. \tag{1.2}$$

If $f(\boldsymbol{x})$ is differentiable in $\boldsymbol{x}$ within $\mathcal{S}$, then the condition in (1.2) is equivalent to

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}), \tag{1.3}$$

where $\nabla f(\boldsymbol{x})$ denotes the gradient of f at point $\boldsymbol{x}$. (1.3) is called the first order condition in the literature.

1.2.3 Convex Optimization Problems

Consider the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{S}, \end{aligned} \tag{1.4}$$

where $\boldsymbol{x}$ is the decision variable, f is the objective function in $\boldsymbol{x}$ and $\mathcal{S}$ is the feasible set of $\boldsymbol{x}$. If function f and feasible set $\mathcal{S}$ of (1.4) are both convex, then (1.4) is called a convex optimization problem. A convex optimization problem has the following properties:

1) Any locally optimal points are globally optimal;

2) If f is differentiable everywhere in $\mathcal{S}$, then the first-order condition holds;

3) If $\boldsymbol{x}^* \in \mathcal{S}$ is a minimizer of problem (1.4), then

$$\nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{y} - \boldsymbol{x}^*) \geq 0 \quad \forall\, \boldsymbol{y} \in \mathcal{S}. \tag{1.5}$$

1.3 Significance of Convex Optimization

If a real world problem can be formulated as a convex optimization problem, then in general this problem can be solved exactly, efficiently and easily by methods like first order gradient method (FOGM), interior point method and many more. History has shown that convex optimization has been benefitting many practical situations including automatic control systems, signal estimation and detection, communications, network design and data modelling while there are abundant reliable computer software that help solve their problems.

1.4 Coordinate Descent Algorithms

A coordinate descent algorithm solves optimization problems by successively performing approximate minimization along coordinate directions (or coordinate hyperplanes) [25]. It is an iterative method in which each iterate is obtained by fixing most components of the decision variable vector $\boldsymbol{x}$ at their values from the current iteration, and approximately minimizing the objective with respect to the remaining components. Typically, this approach lowers the dimensions of the subproblems and thus motivates people solving the subproblems and updating the variables in an easier and more efficient way compared with handling the full problem and the full variable vector.

Specifically, consider the decision variable $\boldsymbol{x} \in \mathcal{X}$ that is decomposable into N blocks as $\boldsymbol{x} := (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$; $\mathcal{X} := \mathcal{X}_1 \times \cdots \mathcal{X}_N$ as the feasible set; $f : \mathcal{X} \to \mathbb{R}$ as the objective function that can be expressed as $f(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$; the function that depends on $\boldsymbol{x}_i$ as $f_i(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_N)$. A general optimization problem of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{X} \end{aligned} \tag{1.6}$$

can be solved to a stationary point by a coordinate descent algorithm described below.

Algorithm 1 Coordinate Descent Algorithm

Input: $\boldsymbol{x}^{(0)} \in \mathcal{X}$.

1: **for** $k = 1, 2, \cdots$ until stopping criteria holds **do**

2: $\quad \boldsymbol{x}^{(k,0)} \leftarrow \boldsymbol{x}^{(k)}$.

3: $\quad$ **for** $i = 1, \cdots, N$ **do**

4: $\quad\quad$ Update:

$$\boldsymbol{x}_i^{(k,i)} \leftarrow \begin{cases} \min\limits_{\boldsymbol{x}_j \in \mathcal{X}_i} f_i(\boldsymbol{x}_1^{(k,i)}, \cdots, \boldsymbol{x}_{j-1}^{(k,i)}, \boldsymbol{x}_j, \boldsymbol{x}_{j+1}^{(k,i-1)}, \cdots, \boldsymbol{x}_N^{(k,i-1)}) & j = i, \\ x_j^{(k,i-1)} & j \neq i. \end{cases} \tag{1.7}$$

5: $\quad$ **end for**

6: $\quad \boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k,i)}$.

7: **end for**

Output: $\boldsymbol{x}^{(k+1)}$.

At the kth iteration, the above CD algorithm separates the full problem $\min\limits_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ into N subproblems which approximate f along each coordinate direction at the current state $x^{(k)}$. The algorithm then exactly solves each subproblem in a cyclic sense. Note that the variable

update is not necessarily ordered in a cyclic manner. It could be a randomized sequence between 1 and N or an order following the coordinate index whose gradient component has the maximal absolute value, as illustrated in [26].

CD algorithm can be easily extended to a Block-CD (BCD) algorithm by each time considering the decision variable vector $\boldsymbol{x}$ along multiple coordinate directions (or coordinate hyperplanes). Examples of BCD benefitting a wide range of applications can be found in [27–34].

1.5 Contributions

In this thesis we investigate the efficiency and effectiveness of BCD using FOGMs to tackle the HSR problem. Specifically, the non-convex HSR problem is separable into two block convex subproblems that can be individually solved by FOGMs. We study the effect of exactness of solving the subproblems towards the full HSR performance in terms of accuracy and runtime. We find that with inexact solving of the subproblems (i.e., inexact BCD), the HSR can still be solved to a stationary point while the accuracy can be preserved and, more importantly, the runtime is largely reduced compared to state-of-the-art algorithms.

1.6 Organization

The organization of this thesis is as follows. Chapter 2 describes the system model and the HSR problem. Chapter 3 describes how FOGMs may solve a quadratic program in detail. After that the HSR problem is shown to be a two block quadratic program that can be handled by BCD algorithms. Chapter 4 presents synthetic and semi-real simulations to compare the performance by BCD algorithms using FOGMs and that by state-of-the-art algorithms. Chapter 5 summarizes this thesis.

Chapter 2

The HSR Problem and The State-of-the-art Algorithms

This chapter describes the HSR problem in detail. The organization of this chapter is as follows. First the HS and MS sensor characteristics are discussed, followed by the signal model that simulates their characteristics are developed. Then we described the commonly used signal model for hyperspectral images named "Linear Mixing Model" (LMM). Finally the HSR problem statement is presented.

2.1 Signal Model of HS and MS Sensors

Let $\boldsymbol{\mathcal{Y}} \in \mathbb{R}_{+}^{\ell_1 \times \ell_2 \times M}$ be a nonnegative tensor representing the pixel values of an SR image with $L = \ell_1 \cdot \ell_2$ pixels and with M spectral bands. The tensor $\boldsymbol{\mathcal{Y}}$ can be lexicographically unfolded to a nonnegative matrix $\boldsymbol{Y} \in \mathbb{R}_{+}^{M \times L}$. The column-wise view and row-wise view of $\boldsymbol{Y}$ are shown in (2.1) and (2.2), respectively, such that $\{\boldsymbol{y}_i\}_{i=1}^{L} \in \mathbb{R}_{+}^{M}$ represents the spectral reflectance signature at the ith pixel and $\{\boldsymbol{y}^j\}_{j=1}^{M} \in \mathbb{R}_{+}^{L}$ represents the vectorized reflectance map of the SR image at the jth spectral band. Figure 2.1 illustrates the process of unfolding a spectral image from a tensor to a matrix.

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_L \end{bmatrix} \tag{2.1}$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}^1 & \boldsymbol{y}^2 & \cdots & \boldsymbol{y}^M \end{bmatrix}^{\mathsf{T}} \tag{2.2}$$

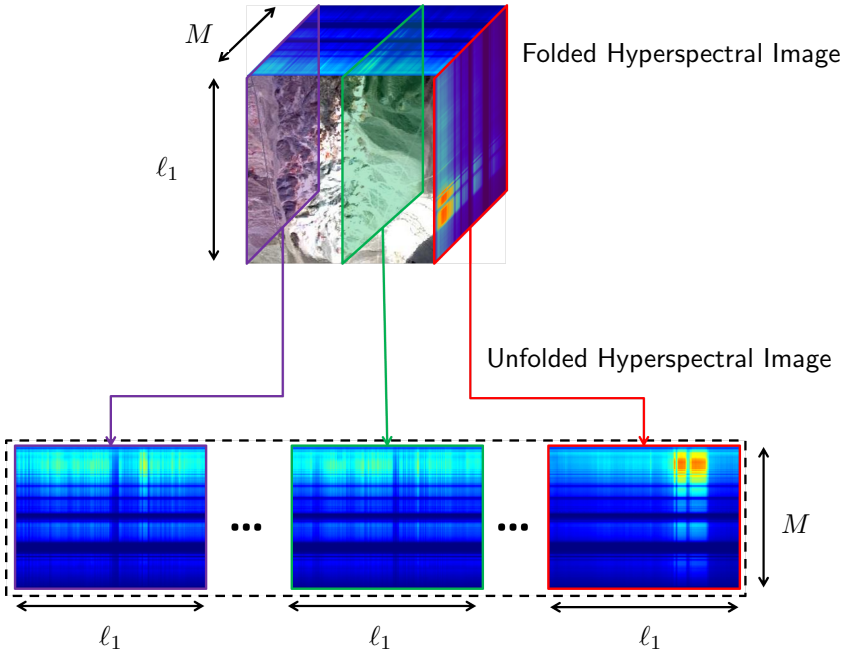


Figure 2.1: Unfolding a hyperspectral image from a tensor to a matrix.

In the signal model an SR image is partially observed by an HS sensor and by an MS sensor. Let $\boldsymbol{Y}_{\mathsf{H}} \in \mathbb{R}_{+}^{M \times L_{\mathsf{H}}}$ and $\boldsymbol{Y}_{\mathsf{M}} \in \mathbb{R}_{+}^{M_{\mathsf{M}} \times L}$ represent the vectorized observed HS and MS images, respectively, where L_{H} is the number of pixels of the HS image and M_{M} is the number of spectral bands of the MS image. As introduced in Section 1.1, we know that HS images have higher spectral resolution and MS images have higher spatial resolution so that the following inequality hold:

$$M > M_{\mathsf{M}}, \tag{2.3}$$

$$L_{\mathsf{H}} < L. \tag{2.4}$$

2.1.1 Spatial Degradation Model in HS Sensors

An HS sensor degrades the spatial contents of SR images. The relation between $\boldsymbol{Y}$ and $\boldsymbol{Y}_{\mathsf{H}}$ can be modelled as

$$\boldsymbol{Y}_{\mathsf{H}} = \boldsymbol{Y}\boldsymbol{G} + \boldsymbol{V}_{\mathsf{H}}, \tag{2.5}$$

where $\boldsymbol{G} \in \mathbb{R}^{L \times L_{\mathsf{H}}}$ is a matrix representing the spatial degradation effect that may be composed of geometric wrapping, translation, blurring and subsampling operations and $\boldsymbol{V}_{\mathsf{H}}$

(a) (b) (c)

Figure 2.2: Effect of spatial degradation by HS sensors: (a) High spatial resolution image with 400×400 pixels; (b) Image of (a) blurred by a 25×25 Gaussian filter of variance $\sigma = 3$; (c) Image of (b) subsampled on every 8 pixels vertically and horizontally.

represents the noise in the HS sensor. Note that $\boldsymbol{G}$ is a matrix multiplied to the vectorized reflectance maps of each band $\{(\boldsymbol{y}^j)^\top\}_{j=1}^N$. In this thesis the spatial responses considered include blurring and subsampling operations. Therefore, $\boldsymbol{G}$ can be expressed as

$$\boldsymbol{G} = \boldsymbol{B}\boldsymbol{D}, \tag{2.6}$$

where the multiplication with $\boldsymbol{B} \in \mathbb{R}^{L \times L}$ and $\boldsymbol{D} \in \mathbb{R}^{L \times L_{\mathsf{H}}}$ represents two different stages of spatial responses. The first stage is a spatial low-pass filtering where the ℓth column of $\boldsymbol{B}$ is a vectorized point spread function applied onto the ℓth pixel of $\boldsymbol{Y}$. Typically $\boldsymbol{B} = [\boldsymbol{b}_1 \cdots \boldsymbol{b}_L]$ is a convolution matrix of a two dimensional normalized low-pass filter kernel. The second stage is a spatial subsampling process with $\boldsymbol{D} \in \{0,1\}^{L \times L_{\mathsf{H}}}$ being a subsampling matrix that draws the reflectance maps $\{(\boldsymbol{y}^j)^\top\}_{j=1}^M$ vertically and horizontally on every d pixels, where $d = \sqrt{L/L_{\mathsf{H}}}$ is the downsampling factor. Figure 2.2 illustrates a standard high spatial resolution color image (a), passing through a spatial low-pass filter (b), and then further passing through a subsampling response (c).

2.1.2 Spectral Degradation Model in MS Sensors

An MS sensor undersamples the SR images in spectral domain by receiving signals at fewer but wider spectral range, resulting in MS images having only few broad band images. The matrices $\boldsymbol{Y}$ and $\boldsymbol{Y}_{\mathsf{M}}$ can be related by

$$\boldsymbol{Y}_{\mathsf{M}} = \boldsymbol{F}\boldsymbol{Y} + \boldsymbol{V}_{\mathsf{M}}, \tag{2.7}$$

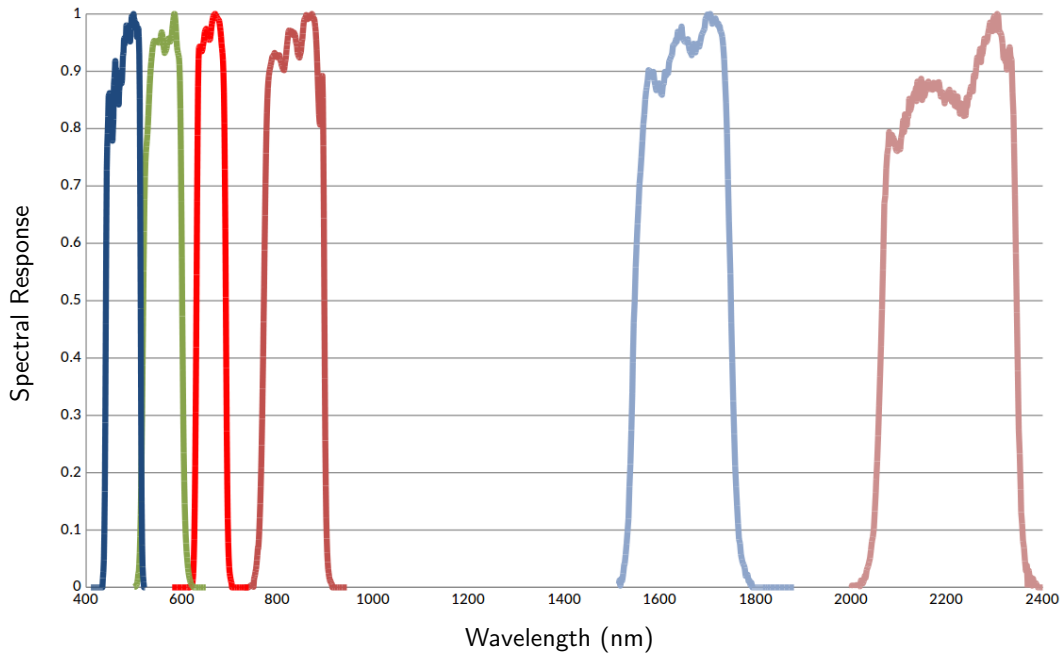


Figure 2.3: Spectral response of the operational TM sensor in LANDSAT 7 Project [35–37].

where $\boldsymbol{F} \in \mathbb{R}_{+}^{M_{\mathsf{M}} \times M}$ represents the spectral degradation matrix of the MS sensor system; $\boldsymbol{V}_{\mathsf{M}}$ is the noise in the MS sensor.

To better illustrate the spectral response of HS and MS sensors, we take an airborne hyperspectral sensor named "AVIRIS" and a satellite multispectral sensor named "Thematic Mapper (TM) Sensor" as examples. They have been serving as the core instruments of NASA's Earth monitoring projects for many years since mid 80s. The AVIRIS sensor, as introduced in Chapter 1, observes the Earth surface at 224 spectral bands covering wavelengths between $0.4\,\mu$m and $2.5\,\mu$m at norminally $10\,$nm intervals. The TM sensor, on the other hand, observes the Earth surface at fewer spectral bands with a wider interval. Among them, six of which overlap with the spectral range of the AVIRIS sensor ($0.4\,\mu$m-$2.5\,\mu$m). The spectral response of the TM sensor is shown in Figure 2.3 [35–37]. The comparison of spectral characteristics between AVIRIS and TM sensors are shown in table 2.1. We can see that a single spectral band of the TM sensor covers multiple number of AVIRIS spectral bands.

Wavelength (μm)	TM Band	AVIRIS Band	Spectral Band Placement
$0.45 - 0.52$	1	$10 - 16$	Visible (VIS-B)
$0.52 - 0.60$	2	$17 - 24$	Visible (VIS-G)
$0.63 - 0.69$	3	$28 - 35$	Visible (VIS-R)
$0.76 - 0.90$	4	$44 - 57$	Near Infrared (NIR)
$1.55 - 1.75$	5	$127 - 146$	Short Wave Infrared (SWIR-1)
$10.4 - 12.5$	6	$-$	Thermal Infrared (TIR)
$2.08 - 2.35$	7	$180 - 208$	Short Wave Infrared (SWIR-2)

Table 2.1: Spectral characteristics of TM sensor and AVIRIS sensor [36–40].

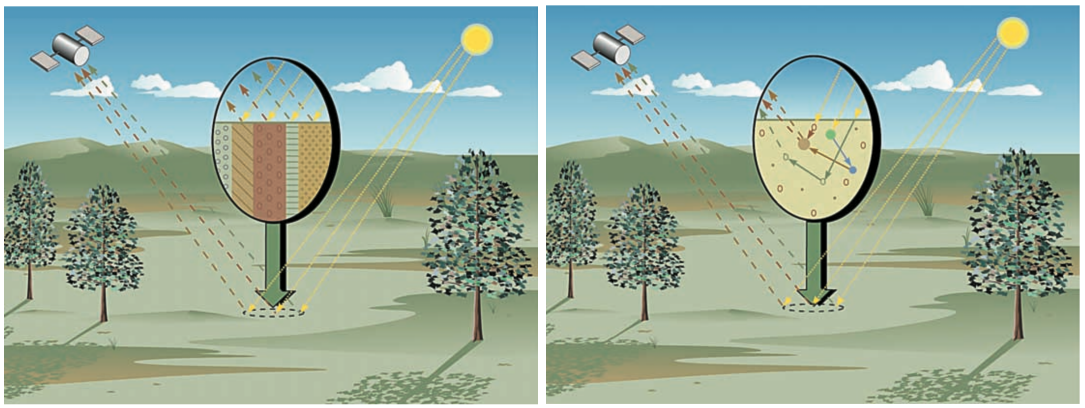
2.2 Low-Rank Representation of HSIs: the Linear Mixing Model (LMM)

In the literature, the geoscience and remote sensing community have long believed HSIs to be high dimensional data with low-rank characteristics [41–59]. From the low-rank model of HSIs, hyperspectral imagery techniques have been successfully devised based on the Linear Mixing Model (LMM) (2.8) which assumes that the spectral signatures in the HSIs are linearly mixed [18, 41–50, 53–64]. Specifically, a HSI matrix can be expressed as

$$\begin{aligned} \boldsymbol{Y} &= \begin{bmatrix} \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_L \end{bmatrix} && \text{(2.8a)} \\ &= \begin{bmatrix} \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_1 & \cdots & \boldsymbol{s}_L \end{bmatrix} && \text{(2.8b)} \\ &= \begin{bmatrix} \sum_{n=1}^{N} \boldsymbol{a}_n s_{n,1} & \cdots & \sum_{n=1}^{N} \boldsymbol{a}_n s_{n,L} \end{bmatrix} && \text{(2.8c)} \\ &= \boldsymbol{A}\boldsymbol{S}, && \text{(2.8d)} \end{aligned}$$

where N, also called the "model order", is the total number of material in $\boldsymbol{Y}$; $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is the endmember matrix that is tall and contains those N distinct material spectra $\{\boldsymbol{a}_i\}_{i=1}^{N}$ in the HSI; $\{\boldsymbol{s}_j\}_{j=1}^{L}$ is the N-dimensional abundance vector at the jth pixel such that the fat abundance matrix $\boldsymbol{S} \in \mathbb{R}^{N \times L}$ tells the contribution of the distinct material spectra in the image. Usually N is assumed to be small, i.e., $N \ll \min\{M, L\}$.

Despite the popularity of the LMM model, it is not always true when the acquired HSI exhibits linearity. As discussed in [65], LMM is more likely to be valid if the endmembers in a pixel appear in spatially segregated patterns (Figure 2.4(a)), while is less likely to be valid if light is multiply scattered before reaching the sensor (Figure 2.4(b)). The Earth surface condition in Figure 2.4(a) can never be met because Earth surface materials are never naturally arranged in discrete, segregated patches. The debate on whether linear or nonlinear mixing processes



(a) (b)

Figure 2.4: Scenarios when LMM is (a) more likely to hold; (b) less likely to hold [65].

dominate the real world scenarios is an unresolved issue. Nevertheless, LMM is widely regarded as an acceptable model and has been demonstrated to be valid in a number of laboratory-based studies [66–69].

2.3 The HSR Problem

By using the spatial and spectral degradation model in (2.6) and (2.7) we may formulate the HSR problem as

$$\begin{aligned} \min_{\boldsymbol{Y}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{Y}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{Y}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{Y} \in \mathbb{R}_{+}^{M\times L}. \end{aligned} \tag{2.9}$$

Although problem (2.9) is convex, it is an ill-posed (underdetermined) problem because the objective is to estimate $\boldsymbol{Y}$ from $\boldsymbol{Y}_{\mathsf{H}}$ and $\boldsymbol{Y}_{\mathsf{M}}$, that says the problem is to estimate many (ML) variables from few $(ML_{\mathsf{H}} + M_{\mathsf{M}}L)$ observed coefficients (directly from $\boldsymbol{Y}_{\mathsf{H}}$ and $\boldsymbol{Y}_{\mathsf{M}}$ to $\boldsymbol{Y}$ in Figure 2.5) and thus the problem has nonunique optimal solutions. Further, problem (2.9) does not guarantee a low-rank estimate of HSI. Therefore, by employing the low-rank LMM model in (2.8), problem (2.9) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in \mathcal{A}, \quad \boldsymbol{S} \in \mathcal{S}, \end{aligned} \tag{2.10}$$

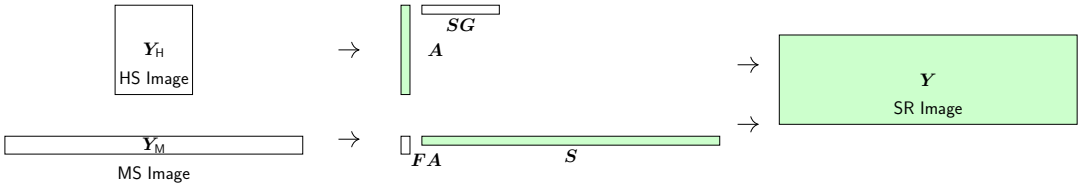


Figure 2.5: Estimate SR image from HS and MS image pair.

where the Frobenous norm $\|\cdot\|_F^2$ computes the sum-square error, $\mathcal{A} \subseteq \mathbb{R}^{M\times N}$ and $\mathcal{S} \subseteq \mathbb{R}^{N\times L}$ are the feasible sets of $\boldsymbol{A}$ and $\boldsymbol{S}$, respectively. Since the rank of the SR image is bounded by model order N, the estimation problem (2.10) is better than (2.9) in a sense that the desired SR image $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{A}}\hat{\boldsymbol{S}}$ has low-rank characteristics. However, even if $\mathcal{A}$ and $\mathcal{S}$ are convex sets, introducing the low-rank model to the HSR problem sacrifices its convexity so that there is no guarantee on finding a global optimal solution in problem (2.10).

2.4 The State-of-the-art HSR Formulations and Approaches

2.4.1 HSR Formulation and Approach by Yokoya et al.

In Yokoya *et al.* 's work, called Coupled Nonnegative Matrix Factorization (CNMF) [18], the HSR problem is formulated as

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{ASG}\|_F^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{FAS}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in \mathbb{R}_+^{M\times N}, \quad \boldsymbol{S} \in \mathbb{R}_+^{N\times L}, \end{aligned} \tag{2.11}$$

which has the same formulation as (2.10) by setting $\mathcal{A} = \mathbb{R}_+^{M\times N}$ and $\mathcal{S} = \mathbb{R}_+^{N\times L}$. In their work problem (2.11) is separated into two independent subproblems such that they can be individually solved by nonnegative matrix factorization (NMF) via multiplicative updates (sometimes called Lee-Seung Algorithm) proposed by Lee and Seung in 1999 [70, 71], i.e. solve the HSR problem by alternatingly solving the two NMF subproblems via Lee-Seung Algorithm. In the following part of this section we first review the NMF problem and the Lee-Seung Algorithm. Then we present the CNMF algorithm that solves problem (2.11) by alternating optimization (AO).

2.4.1.1 Nonnegative Matrix Factorization (NMF)

Let $\boldsymbol{V} \in \mathbb{R}_+^{m\times\ell}$ be a rank-n matrix whom we want to decompose as $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$, where $\boldsymbol{W} \in \mathbb{R}_+^{m\times n}$ and $\boldsymbol{H} \in \mathbb{R}_+^{n\times\ell}$. The NMF problem can be stated as

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{H}} \quad & \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{W} \in \mathbb{R}_+^{m\times n}, \quad \boldsymbol{H} \in \mathbb{R}_+^{n\times\ell}. \end{aligned} \tag{2.12}$$

Although NMF problem is nonconvex, it is convex with respect to (w.r.t.) $\boldsymbol{W}$ and $\boldsymbol{H}$ individually. Usually people attack the NMF problem by AO to find a local optimal solution by repeating

$$\boldsymbol{H}^{(k+1)} \leftarrow \arg\min_{\boldsymbol{H}\geq\boldsymbol{0}} \|\boldsymbol{V} - \boldsymbol{W}^{(k)}\boldsymbol{H}\|_{\mathsf{F}}^2, \tag{2.13}$$

$$\boldsymbol{W}^{(k+1)} \leftarrow \arg\min_{\boldsymbol{W}\geq\boldsymbol{0}} \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}^{(k+1)}\|_{\mathsf{F}}^2, \tag{2.14}$$

for some given $\boldsymbol{W}^{(0)}$ and $\boldsymbol{H}^{(0)}$. The AO algorithm proposed by Lee and Seung is shown in Algorithm 2, which is a multiplicative update (MU) type update scheme that indirectly solves (2.13) and (2.14). In the algorithm description, "$*$" and "$/$" represent elementwise multiplication and division, respectively.

Algorithm 2 Lee-Seung Algorithm for Nonnegative Matrix Factorization (NMF)

Input: $\boldsymbol{V}$, $\boldsymbol{W}^{(0)}$, $\boldsymbol{H}^{(0)}$.

for $k = 0, 1, 2, \cdots$ until stopping criteria reaches **do**
  `// H subproblem`
  $\boldsymbol{H}^{(k,0)} \leftarrow \boldsymbol{H}^{(k)}$.
  for $\ell = 1, 2, \cdots$ **do**
    $\boldsymbol{H}^{(k,\ell)} \leftarrow \boldsymbol{H}^{(k,\ell-1)} * (\boldsymbol{W}^{(k)\top}\boldsymbol{V})/(\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}\boldsymbol{H}^{(k,\ell-1)})$.
  end for
  $\boldsymbol{H}^{(k+1)} \leftarrow \boldsymbol{H}^{(k,\ell)}$.
  `// W subproblem`
  $\boldsymbol{W}^{(k,0)} \leftarrow \boldsymbol{W}^{(k)}$.
  for $\ell = 1, 2, \cdots$ **do**
    $\boldsymbol{W}^{(k,\ell)} \leftarrow \boldsymbol{W}^{(k,\ell-1)} * (\boldsymbol{V}\boldsymbol{H}^{(k+1)\top})/(\boldsymbol{W}^{(k,\ell-1)}\boldsymbol{H}^{(k+1)}\boldsymbol{H}^{(k+1)\top})$.
  end for
  $\boldsymbol{W}^{(k+1)} \leftarrow \boldsymbol{W}^{(k,\ell)}$.
end for

Output: $\boldsymbol{W}^{(k+1)}$,$\boldsymbol{H}^{(k+1)}$.

2.4.1.2 Coupled Nonnegative Matrix Factorization (CNMF)

Let

$$\hat{\boldsymbol{W}},\ \hat{\boldsymbol{H}} \leftarrow \texttt{LeeSeung}(\boldsymbol{V}, \boldsymbol{W}', \boldsymbol{H}')$$

denote the process of solving the NMF problem (2.12) by Algorithm 2 using $\boldsymbol{W}'$ and $\boldsymbol{H}'$ as the algorithm input. Yokoya *et al.* tackle the HSR problem by applying the Lee-Seung Algorithm to alternatingly factorize $\boldsymbol{Y}_{\mathsf{H}}$ and $\boldsymbol{Y}_{\mathsf{M}}$ to obtain the desired endmember and abundance matrix pair. Rather than directly solve problem (2.11), CNMF starts from initial guess on matrix $\boldsymbol{A}^{(0)}$ and $\boldsymbol{S}^{(0)}$. Then, with $\boldsymbol{E}^{(0)} = \boldsymbol{F}\boldsymbol{A}^{(0)}$ and $\boldsymbol{R}^{(0)} = \boldsymbol{S}^{(0)}\boldsymbol{G}$, CNMF solves the HSR problem by repeating (2.15) and (2.16) below:

$$\boldsymbol{A}^{(k+1)}, \boldsymbol{R}' \leftarrow \texttt{LeeSeung}(\boldsymbol{Y}_{\mathsf{H}}, \boldsymbol{A}^{(k)}, \boldsymbol{R}^{(k)}) \tag{2.15a}$$

$$\boldsymbol{E}^{(k+1)} \leftarrow \boldsymbol{F}\boldsymbol{A}^{(k+1)}, \tag{2.15b}$$

$$\boldsymbol{E}', \boldsymbol{S}^{(k+1)} \leftarrow \texttt{LeeSeung}(\boldsymbol{Y}_{\mathsf{M}}, \boldsymbol{E}^{(k+1)}, \boldsymbol{S}^{(k)}) \tag{2.16a}$$

$$\boldsymbol{R}^{(k+1)} \leftarrow \boldsymbol{S}^{(k+1)}\boldsymbol{G}. \tag{2.16b}$$

$\boldsymbol{A}^{(K+1)}$ and $\boldsymbol{S}^{(K+1)}$ are the desired endmember and abundance matrix pair, respectively; $\boldsymbol{E}^{(K+1)}$ and $\boldsymbol{R}^{(K+1)}$ are the byproduct matrix pair. The CNMF algorithm is shown in Algorithm 3. The recovered SR image by CNMF can be obtained by $\hat{\boldsymbol{Y}} = \boldsymbol{A}^{(k+1)}\boldsymbol{S}^{(k+1)}$.

Despite its efficiency and ease of implementation, we remark that the algorithm is an indirect approach to solve (2.11) because MS recovery error does not take part in the endmember estimation, and the same goes for the HS recovery error in abundance estimation. An online available program code of CNMF can be found from Yokoya's homepage [1].

[1] Matlab implementation can be found in http://naotoyokoya.com/Download.html. Note that in their implementation they attempt to enforce some further constraints on the variables by using tricks suggested in [47]. However in this thesis we reimplement CNMF to exactly follow the algorithm description in Yokoya's paper.

Algorithm 3 Coupled Nonnegative Matrix Factorization (CNMF)

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{A}^{(0)}$, $\boldsymbol{S}^{(0)}$.

1: **for** $k = 0, 1, 2, \cdots$ until stopping criteria reaches **do**

`// Hyperspectral Subproblem`

2: $\boldsymbol{R}^{(k)} \leftarrow \boldsymbol{S}^{(k)}\boldsymbol{G}$, $\boldsymbol{R}^{(k,0)} \leftarrow \boldsymbol{R}^{(k)}$, $\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$.

3: **for** $\ell = 1, 2, \cdots$ **do**

4: $\boldsymbol{R}^{(k,\ell)} \leftarrow \boldsymbol{R}^{(k,\ell-1)} * (\boldsymbol{A}^{(k,\ell-1)\top}\boldsymbol{Y}_{\mathsf{H}})/(\boldsymbol{A}^{(k,\ell-1)\top}\boldsymbol{A}^{(k,\ell-1)}\boldsymbol{R}^{(k,\ell-1)})$.

5: $\boldsymbol{A}^{(k,\ell)} \leftarrow \boldsymbol{A}^{(k,\ell-1)} * (\boldsymbol{Y}_{\mathsf{H}}\boldsymbol{R}^{(k,\ell)\top})/(\boldsymbol{A}^{(k,\ell-1)}\boldsymbol{R}^{(k,\ell)}\boldsymbol{R}^{(k,\ell)\top})$.

6: **end for**

7: $\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,\ell)}$.

`// Multispectral Subproblem`

8: $\boldsymbol{E}^{(k)} \leftarrow \boldsymbol{F}\boldsymbol{A}^{(k+1)}$, $\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{E}^{(k,0)} \leftarrow \boldsymbol{E}^{(k)}$.

9: **for** $\ell = 1, 2, \cdots$ **do**

10: $\boldsymbol{S}^{(k,\ell)} \leftarrow \boldsymbol{S}^{(k,\ell-1)} * (\boldsymbol{E}^{(k,\ell-1)\top}\boldsymbol{Y}_{\mathsf{M}})/(\boldsymbol{E}^{(k,\ell-1)\top}\boldsymbol{E}^{(k,\ell-1)}\boldsymbol{S}^{(k,\ell-1)})$.

11: $\boldsymbol{E}^{(k,\ell)} \leftarrow \boldsymbol{E}^{(k,\ell-1)} * (\boldsymbol{Y}_{\mathsf{M}}\boldsymbol{S}^{(k,\ell)\top})/(\boldsymbol{E}^{(k,\ell-1)}\boldsymbol{S}^{(k,\ell)}\boldsymbol{S}^{(k,\ell)\top})$.

12: **end for**

13: $\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,\ell)}$.

14: **end for**

Output: $\boldsymbol{A}^{(k+1)}, \boldsymbol{S}^{(k+1)}$.

2.4.2 HSR Formulation and Approach by Wei et al.

In Wei *et al.* 's work, called "Fusion based on Unmixing for Multiband Images (FUMI)" [48], the HSR problem is formulated as

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in [0,1]^{M\times N}, \quad \boldsymbol{S} \in \triangle^{N\times L}, \end{aligned} \tag{2.17}$$

where

$$\triangle^{N\times L} = \{\boldsymbol{X} \in \mathbb{R}_+^{N\times L} \mid \mathbf{1}_N^\top \boldsymbol{x}_i = 1 \ , \ \forall\, i\} \tag{2.18}$$

denotes a set of matrix that is nonnegative and columnwise sum-to-one. Similar to CNMF, Wei *et al.* apply AO to the HSR problem by splitting it into two. Their formulation differs

from CNMF by having more constraints on the decision variables, i.e., in addition to non-negative constraints on the variables, the endmember matrix is further bounded from above by $\mathbf{1}$ and the abundance matrix is forced to be on a unit simplex. Their approach to solving the HSR problem is also different from CNMF such that both HS and MS recovery error are considered in both subproblems. Due to the extra constraints, they solve the subproblems by Alternating Direction Method of Multipliers (ADMM) [72], an algorithm that efficiently handles convex constrained quadratic program (QP). In the following part of this section, we first review the ADMM algorithm and solving QP by ADMM. Then the algorithm details of FUMI are presented.

2.4.2.1 Alternating Direction Method of Multipliers (ADMM)

ADMM, popularized by Boyd *et al.* in 2010 [72], is an algorithm that solves problems with the following structure:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n,\boldsymbol{z}\in\mathbb{R}^m} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{P}\boldsymbol{x} + \boldsymbol{Q}\boldsymbol{z} = \boldsymbol{r}, \end{aligned} \tag{2.19}$$

where $\boldsymbol{P} \in \mathbb{R}^{k\times n}$, $\boldsymbol{Q} \in \mathbb{R}^{k\times m}$ and $\boldsymbol{r} \in \mathbb{R}^k$; f and g are convex functions. By defining the Augmented Lagrangian of problem (2.19) as

$$L_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{y}^\top(\boldsymbol{P}\boldsymbol{x} + \boldsymbol{Q}\boldsymbol{z} - \boldsymbol{r}) + \frac{\mu}{2}\|\boldsymbol{P}\boldsymbol{x} + \boldsymbol{Q}\boldsymbol{z} - \boldsymbol{r}\|_2^2, \tag{2.20}$$

where $\boldsymbol{y} \in \mathbb{R}^p$ is called the dual variable; $\mu > 0$ is a positive constant, the ADMM algorithm that solves problem (2.19) is presented in Algorithm 4. Its general convergence analysis can be found in [72–74].

Algorithm 4 Alternating Direction Method of Multipliers (ADMM)

Input: $\boldsymbol{x}^{(0)}$, $\boldsymbol{z}^{(0)}$, $\boldsymbol{y}^{(0)}$.

1: **for** $i = 1, 2, \cdots$ until stopping criteria reaches **do**
2: $\quad \boldsymbol{x}^{(i)} \leftarrow \arg\min_{\boldsymbol{x}} L_\mu(\boldsymbol{x}, \boldsymbol{z}^{(i-1)}, \boldsymbol{y}^{(i-1)})$.
3: $\quad \boldsymbol{z}^{(i)} \leftarrow \arg\min_{\boldsymbol{z}} L_\mu(\boldsymbol{x}^{(i)}, \boldsymbol{z}, \boldsymbol{y}^{(i-1)})$.
4: $\quad \boldsymbol{y}^{(i)} \leftarrow \boldsymbol{y}^{(i-1)} + \mu(\boldsymbol{P}\boldsymbol{x}^{(i)} + \boldsymbol{Q}\boldsymbol{z}^{(i)} - \boldsymbol{r})$.
5: **end for**

Output: $\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}$.

QP is a special case of problem (2.19). A standard QP can be written as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{2.21}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex feasible set of $\boldsymbol{x}$. If we let

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{I}, \\ \boldsymbol{Q} &= -\boldsymbol{I}, \\ \boldsymbol{r} &= \boldsymbol{0}, \\ f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}, \\ g(\boldsymbol{z}) &= \mathcal{I}_{\mathcal{X}}(\boldsymbol{z}) \\ &= \begin{cases} 0, & \boldsymbol{z} \in \mathcal{X} \\ +\infty, & \boldsymbol{z} \notin \mathcal{X}, \end{cases}, \end{aligned}$$

where $\mathcal{I}_{\mathcal{X}}(\cdot)$ is an indicator function w.r.t. convex set $\mathcal{X}$, then problem (2.19) becomes a convex constrained QP.

2.4.2.2 Fusion based on Unmixing for Multiband Images (FUMI)

The problem formulation by Wei *et al.* is splitted into $\boldsymbol{S}$ subproblem and $\boldsymbol{A}$ subproblem. The subproblems are listed below.

$$\begin{aligned} \min_{\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{S} \in \triangle^{N\times L}, \end{aligned} \tag{2.22}$$

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in [0,1]^{M\times N}. \end{aligned} \tag{2.23}$$

And the Augmented Lagrangian of $\boldsymbol{S}$ and $\boldsymbol{A}$ subproblems of (2.22) become

$$\begin{aligned} \mathcal{L}_S(\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}) &= \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ &\quad + \mathcal{I}_{\triangle^{N\times L}}(\boldsymbol{\Sigma}) + \langle \boldsymbol{S} - \boldsymbol{\Sigma}, \boldsymbol{\Psi}\rangle + \frac{\mu}{2}\|\boldsymbol{S} - \boldsymbol{\Sigma}\|_{\mathsf{F}}^2 \end{aligned} \tag{2.24}$$

and

$$\begin{aligned}\mathcal{L}_A(\boldsymbol{A},\boldsymbol{S},\boldsymbol{\Lambda},\boldsymbol{\Phi}) &= \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}}-\boldsymbol{A}\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2+\frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}}-\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}\|_{\mathsf{F}}^2 \\ &\quad +\mathcal{I}_{[0,1]^{M\times N}}(\boldsymbol{\Lambda})+\langle\boldsymbol{A}-\boldsymbol{\Lambda},\boldsymbol{\Phi}\rangle+\frac{\mu}{2}\|\boldsymbol{A}-\boldsymbol{\Lambda}\|_{\mathsf{F}}^2, \end{aligned} \tag{2.25}$$

where $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ are the split variables; $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ are the dual variable of $\boldsymbol{S}$ and $\boldsymbol{A}$ subproblems, respectively, and $\mu > 0$. FUMI is therefore presented as in Algorithm 5 [2].

Algorithm 5 Fusion based on Unmixing for Multiband Images (FUMI)

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{A}^{(0)}$, $\boldsymbol{S}^{(0)}$

1: $\boldsymbol{\Sigma}^{(0)} \leftarrow \boldsymbol{S}^{(0)}$, $\boldsymbol{\Psi}^{(0)} \leftarrow \boldsymbol{0}$, $\boldsymbol{\Lambda}^{(0)} \leftarrow \boldsymbol{A}^{(0)}$, $\boldsymbol{\Phi}^{(0)} \leftarrow \boldsymbol{0}$.

2: **for** $k = 1, 2, \cdots$ until stopping criteria reaches **do**

// `S-Subproblem`

3: $\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{\Sigma}^{(k,0)} \leftarrow \boldsymbol{\Sigma}^{(k)}$, $\boldsymbol{\Psi}^{(k,0)} \leftarrow \boldsymbol{\Psi}^{(k)}$.

4: **for** $\ell = 1, 2, \cdots$ **do**

5: $\boldsymbol{S}^{(k,\ell)} \leftarrow \arg\min_{\boldsymbol{S}} \mathcal{L}_S\left(\boldsymbol{A}^{(k)}, \boldsymbol{S}, \boldsymbol{\Sigma}^{(k,\ell-1)}, \boldsymbol{\Psi}^{(k,\ell-1)}\right)$.

6: $\boldsymbol{\Sigma}^{(k,\ell)} \leftarrow \arg\min_{\boldsymbol{\Sigma}} \mathcal{L}_S\left(\boldsymbol{A}^{(k)}, \boldsymbol{S}^{(k,\ell)}, \boldsymbol{\Sigma}, \boldsymbol{\Psi}^{(k,\ell-1)}\right)$.

7: $\boldsymbol{\Psi}^{(k,\ell)} \leftarrow \boldsymbol{\Psi}^{(k,\ell-1)} - \left(\boldsymbol{S}^{(k,\ell)} - \boldsymbol{\Sigma}^{(k,\ell)}\right)$.

8: **end for**

9: $\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,\ell)}$, $\boldsymbol{\Sigma}^{(k+1)} \leftarrow \boldsymbol{\Sigma}^{(k,\ell)}$, $\boldsymbol{\Psi}^{(k+1)} \leftarrow \boldsymbol{\Psi}^{(k,\ell)}$.

// `A-Subproblem`

10: $\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{\Lambda}^{(k,0)} \leftarrow \boldsymbol{\Lambda}^{(k)}$, $\boldsymbol{\Phi}^{(k,0)} \leftarrow \boldsymbol{\Phi}^{(k)}$.

11: **for** $\ell = 1, 2, \cdots, L$ **do**

12: $\boldsymbol{A}^{(k,\ell)} \leftarrow \arg\min_{\boldsymbol{A}} \mathcal{L}_A\left(\boldsymbol{A}, \boldsymbol{S}^{(k+1)}, \boldsymbol{\Lambda}^{(k,\ell-1)}, \boldsymbol{\Phi}^{(k,\ell-1)}\right)$.

13: $\boldsymbol{\Lambda}^{(k,\ell)} \leftarrow \arg\min_{\boldsymbol{\Lambda}} \mathcal{L}_A\left(\boldsymbol{A}^{(k,\ell)}, \boldsymbol{S}^{(k+1)}, \boldsymbol{\Lambda}, \boldsymbol{\Phi}^{(k,\ell-1)}\right)$.

14: $\boldsymbol{\Phi}^{(k,\ell)} \leftarrow \boldsymbol{\Phi}^{(k,\ell-1)} - \left(\boldsymbol{A}^{(k,\ell)} - \boldsymbol{\Lambda}^{(k,\ell)}\right)$.

15: **end for**

16: $\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,\ell)}$, $\boldsymbol{\Lambda}^{(k+1)} \leftarrow \boldsymbol{\Lambda}^{(k,\ell)}$, $\boldsymbol{\Phi}^{(k+1)} \leftarrow \boldsymbol{\Phi}^{(k,\ell)}$.

17: **end for**

Output: $\boldsymbol{A}^{(k+1)}$, $\boldsymbol{S}^{(k+1)}$

[2] The Matlab implementation of FUMI can be found in author's GitHub page https://github.com/qw245/FUMI. Note that in their implementation the algorithm does not solve the subproblems properly because the convergence of ADMM is not checked. Besides they perform subspace projection before solving the subproblems in order to reduce computation cost. In this thesis, however, we modify FUMI to exactly follow the algorithm description in Wei's paper.

Similar to CNMF, FUMI recovers the estimated SR image by $\hat{\boldsymbol{Y}} = \boldsymbol{A}^{(k+1)}\boldsymbol{S}^{(k+1)}$.

2.4.3 General Review on the State of the Arts

The discussed state-of-the-art HSR problem formulations can be decomposed to convex subproblems. To solve them the CNMF and FUMI algorithms use the famous Lee-Seung Algorithm and ADMM, respectively. CNMF uses Lee-Seung Algorithm because this algorithm handles NMF-type problems efficiently. On the contrary, however, CNMF is a less flexible algorithm because the factor matrix pair constraints are limited to be nonnegative orthant and cannot be easily extended to other constraints. FUMI uses ADMM to solve the convex subproblems. A possible drawback is that ADMM may cost too much runtime if the spectral images size grows to very large scale.

Chapter 3

HSR with First-Order Gradient Methods (FOGMs)

In this chapter we discuss an inexact BCD approach to tackle the HSR problem, namely ALternating Gradient-based Optimization (ALGO) using FOGMs. The organization of this chapter is as follows. We first express the subproblems of the HSR problem as convex constrained QP. Then we give a review on a few kind of FOGMs that solve this QP. After that a traditional algorithm framework that solves the HSR problem by exact BCD is presented, followed by our proposed algorithm framework ALGO. We also show the specific algorithm implementation of ALGO using each of the discussed FOGM at the end of this chapter.

3.1 HSR Problem as Convex Constrained QPs

The general HSR problem is re-stated below.

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{ASG}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{FAS}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in \mathcal{A}, \quad \boldsymbol{S} \in \mathcal{S}. \end{aligned} \tag{$*$}$$

Problem $(*)$ is separately convex in $\boldsymbol{S}$ and in $\boldsymbol{A}$. The $\boldsymbol{S}$ and $\boldsymbol{A}$ subproblem of $(*)$ can be expressed as

$$\begin{aligned} \min_{\boldsymbol{S}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{ASG}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{FAS}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{S} \in \mathcal{S}, \end{aligned} \tag{$\mathcal{S}$}$$

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{ASG}\|_{\mathsf{F}}^2 + \frac{1}{2}\|\boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{FAS}\|_{\mathsf{F}}^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in \mathcal{A}, \end{aligned} \tag{$\mathcal{A}$}$$

respectively. We see that the objective function of $(\mathcal{S})$ and $(\mathcal{A})$ can be expressed in least squares form using the Kronecker product equality

$$\mathtt{vec}(\boldsymbol{PXQ}) = \left[\boldsymbol{Q}^\top \otimes \boldsymbol{P}\right] \mathtt{vec}(\boldsymbol{X}), \tag{3.1}$$

where $\boldsymbol{P} \in \mathbb{R}^{m\times n}$; $\boldsymbol{X} \in \mathbb{R}^{n\times p}$; $\boldsymbol{Q} \in \mathbb{R}^{p\times q}$; $\otimes$ denotes the Kronecker product and $\mathtt{vec}(\cdot)$ is an operation that lexicographically concatenate all columns of the argument matrix to form a single column. i.e., $\boldsymbol{x} = \mathtt{vec}(\boldsymbol{X}) \in \mathbb{R}^{np}$. Let $\boldsymbol{s} = \mathtt{vec}(\boldsymbol{S})$, $\boldsymbol{a} = \mathtt{vec}(\boldsymbol{A})$, $\boldsymbol{y}_{\mathsf{H}} = \mathtt{vec}(\boldsymbol{Y}_{\mathsf{H}})$, $\boldsymbol{y}_{\mathsf{M}} = \mathtt{vec}(\boldsymbol{Y}_{\mathsf{M}})$. Then $(\mathcal{S})$ and $(\mathcal{A})$ are equivalent to

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{H}_S\boldsymbol{s}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{S} \in \mathcal{S}, \quad \boldsymbol{s} = \mathtt{vec}(\boldsymbol{S}) \end{aligned} \tag{$\mathcal{s}$}$$

and

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{H}_A\boldsymbol{a}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{A} \in \mathcal{A}, \quad \boldsymbol{a} = \mathtt{vec}(\boldsymbol{A}), \end{aligned} \tag{$\mathcal{a}$}$$

respectively, in which

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_{\mathsf{H}} \\ \boldsymbol{y}_{\mathsf{M}} \end{bmatrix}; \quad \boldsymbol{H}_S = \begin{bmatrix} \boldsymbol{G}^\top \otimes \boldsymbol{A} \\ \boldsymbol{I}_L \otimes \boldsymbol{FA} \end{bmatrix}; \quad \boldsymbol{H}_A = \begin{bmatrix} (\boldsymbol{SG})^\top \otimes \boldsymbol{I}_M \\ \boldsymbol{S}^\top \otimes \boldsymbol{A} \end{bmatrix}.$$

In the remaining chapters, we focus our discussion of FOGMs on the QPs $(\mathcal{s})$ and $(\mathcal{a})$. To see they are QPs we let $f_s(\boldsymbol{s})$ and $f_a(\boldsymbol{a})$ be their objective functions and observe that

$$f_s(\boldsymbol{s}) = \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{H}_S^\top \boldsymbol{H}_S \boldsymbol{s} - \boldsymbol{y}^\top \boldsymbol{H}_S \boldsymbol{s} + \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{y} \tag{3.2}$$

and

$$f_a(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{H}_A^\top \boldsymbol{H}_A \boldsymbol{a} - \boldsymbol{y}^\top \boldsymbol{H}_A \boldsymbol{a} + \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{y} \tag{3.3}$$

both attain a quadratic form. Also, either using Yokoya *et al.* 's formulation or Wei *et al.* 's formulation, the constraints of $(\mathfrak{s})$ and $(\mathfrak{a})$ are all convex. Specifically, we may consider:

- Yokoya *et al.* 's formulation:
  $\mathcal{S} = \mathbb{R}_+^{N\times L}$ and $\mathcal{A} = \mathbb{R}_+^{M\times N}$ so that the constraints on $\boldsymbol{s}$ and $\boldsymbol{a}$ are $\boldsymbol{s} \in \mathbb{R}_+^{NL}$ and $\boldsymbol{a} \in \mathbb{R}_+^{MN}$, respectively, which are convex.

- Wei *et al.* 's formulation:
  $\mathcal{S} = \triangle^{N\times L}$ and $\mathcal{A} \in [0,1]^{M\times N}$. The feasible set of $\boldsymbol{s} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_L)$ can be viewed as $\triangle^N \times \cdots \times \triangle^N$ and that of $\boldsymbol{a}$ as $\{\boldsymbol{x} \in \mathbb{R}^{MN} \mid \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1}\}$, which are also convex.

For convenience, our discussion on FOGMs towards the HSR problem in later sections focuses onto the following problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{QP}$$

where $\mathcal{X}$ is a convex set and $\boldsymbol{R} \succeq \boldsymbol{0}$ is a matrix that is symmetric and positive semidefinite (PSD). As a preliminary but essential concept, we review the projection onto the aforementioned convex sets before the discussion of FOGMs.

3.2 Projection onto a Convex Set

Let $\boldsymbol{y} \in \mathbb{R}^n$ be an arbitrary point and $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex subset of $\mathbb{R}^n$. The problem of projecting $\boldsymbol{y}$ onto $\mathcal{X}$ can be stated as

$$\hat{\boldsymbol{y}} = \Pi_{\mathcal{X}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2, \tag{3.4}$$

where Π denotes a projection operator. The projection above can be easily solved if

a) $\mathcal{X} = \mathbb{R}_+^n$:
   $\hat{\boldsymbol{y}} = \Pi_{\mathcal{X}}(\boldsymbol{y}) \quad \Leftrightarrow \quad \hat{y}_i = \max\{0, y_i\}$;

b) $\mathcal{X} = [0,1]^n$:
   $\hat{\boldsymbol{y}} = \Pi_{\mathcal{X}}(\boldsymbol{y}) \quad \Leftrightarrow \quad \hat{y}_i = \min\{1, \max\{0, y_i\}\}$;

c) $\mathcal{X} = \triangle^n$:
   $\Pi_{\mathcal{X}}(\boldsymbol{y})$ can be computed using the algorithm detailed in [75].

3.3 Solving QPs with BCD using FOGMs

Let $f(\boldsymbol{x})$ be the objective function of problem (**QP**). Since it is convex and the gradient $\nabla f(\boldsymbol{x})$ exists everywhere in $\mathcal{X}$, gradient methods can solve for a global optimal solution if we allow the algorithm to run for many iterations. FOGMs refer to a class of gradient methods that only requires the first-order information of f, i.e., $\nabla f(\boldsymbol{x})$. A general mechanism of FOGMs is described below. Usually an arbitrary point $\boldsymbol{x}^{(0)} \in \mathcal{X}$ is first given. Then, at the kth iteration, when we take a step to the next point $\boldsymbol{x}^{(k+1)}$, we choose the descent direction in which $f(\boldsymbol{x})$ decreases quickly, which could be the direction opposite to $\nabla f(\boldsymbol{x}^{(k)})$ or any directions that guarantee a decrement of $f(\boldsymbol{x})$. A sequence of points $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots$ are recursively generated by repeating

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \gamma \boldsymbol{d}^{(k)}, \quad \text{for } k = 0, 1, 2, \cdots, \tag{3.5}$$

where γ is a step-size controlling the movement and $\boldsymbol{d}^{(k)}$ is the descent direction at the kth iteration. The stopping criteria is usually measured by checking whenever the objective value converges.

Popular FOGMs include Gradient Projection Method (GP), Barzilai-Borwein Gradient Projection Method (BBGP) [76], which is a special case of GP, Proximal Gradient Method (PG), Fast Proximal Gradient Method (also known as Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)) [77, 78] and Frank-Wolfe Method (FW) [79]. In the remaining subsections we review them one by one.

3.3.1 Gradient Projection Method (GP)

The mechanism of GP is as follows. Given a starting point $\boldsymbol{x}^{(0)}$, GP recursively computes

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)} \left(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)} \right), \tag{3.6}$$

where

$$\boldsymbol{z}^{(k)} = \Pi_{\mathcal{X}} \left(\boldsymbol{x}^{(k)} - \beta^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \right). \tag{3.7}$$

Here $\alpha^{(k)} \in [0,1]$ is the stepsize and $\beta^{(k)} \in [\beta_{\min}, \beta_{\max}]$ is the positive scalar in the kth iteration. Usually $\left(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}\right)$ in (3.6) is referred as the descent direction or the feasible direction, where $\boldsymbol{z}^{(k)}$ is obtained by moving $\boldsymbol{x}^{(k)}$ along the steepest descent direction, i.e., take

$$\boldsymbol{d} = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)}), \tag{3.8}$$

with an amount controlled by $\beta^{(k)}$, followed by a projection onto the convex set $\mathcal{X}$ so that $\boldsymbol{z}^{(k)}$ must be in the feasible set $\mathcal{X}$. In each iteration the decision variable $\boldsymbol{x}^{(k)}$ is updated along a feasible direction by using stepsize $\alpha^{(k)}$.

To guarantee the convergence of GP, $\alpha^{(k)}$ and $\beta^{(k)}$ should be chosen carefully. There exists serveral methods choosing them as shown below.

a) In search of $\alpha^{(k)}$ with fixed $\beta^{(k)}$:
   Set $\beta^{(k)} = \beta$ for all k, two popular strategies are

   – exact line search: find $\alpha^{(k)}$ by solving

$$\alpha^{(k)} = \arg \min_{0 \le \alpha \le 1} f(\boldsymbol{x}^{(k)} + \alpha(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)})); \tag{3.9}$$

   – Armijo rule: choose positive scalars σ and γ in the interval $(0,1)$ and the initial stepsize α. Then find for the smallest natural number i such that the inequality

$$f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^{(k)} + \gamma^i \alpha(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)})) \ge \sigma \gamma^i \alpha \nabla f(\boldsymbol{x}^{(k)})^\top (\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}) \tag{3.10}$$

   holds. Once we have i, we set $\alpha^{(k)} = \tau^i \alpha$. In some classical literature, σ is preferred to be close to 0, i.e., $\sigma \in [10^{-5}, 10^{-1}]$, while β the fixed constant is chosen in $[0.1, 0.5]$. We can always take $\alpha = 1$ if we do not have a better choice [80].

b) In search of $\beta^{(k)}$ with fixed $\alpha^{(k)}$:
   Set $\alpha^{(k)} = \alpha$ for all k and use Armijo rule to choose $\beta^{(k)}$. To distinguish from (3.10) that searches along the feasible direction, the search here is commonly referred as searching by Armijo rule along the projection arc.

c) In search of both $\alpha^{(k)}$ and $\beta^{(k)}$:
   This is a more sophisticated way with an objective that is to improve the convergence

rate. In this case $\alpha^{(k)}$ is found by exact line search or Armijo rule. On the other hand, $\beta^{(k)}$ can be generalized to a symmetric and positive definite (PD) matrix similar to approach in Newton and quasi-Newton methods [81, 82].

d) Fixed $\alpha^{(k)}$ and $\beta^{(k)}$:
It is possible to fix both parameters for all k and guarantee the convergence. As will be discussed later that, under some conditions, Proximal Gradient Method (PG) is of this favor.

In this section we consider only case a) and c).

3.3.1.1 In search of $\alpha^{(k)}$ with fixed $\beta^{(k)}$ (case a)

In general problem (3.9) is hard and case-dependent. As a compromise, the standard and easy Armijo rule line search is commonly used, which involves recursive operations but attains only suboptimal stepsize. However when the problem at hand is a QP, there admits a closed form solution to (3.9). Specifically, the solution of $\alpha^{(k)}$ can be derived as follows. Let $f(\boldsymbol{x})$ be the objective of problem (**QP**) and $g(\alpha) = f(\boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x}))$. Then we have

$$\begin{aligned} g(\alpha) &= f(\boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x})) \\ &= \frac{1}{2}\left[\boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x})\right]^\top \boldsymbol{R}\left[\boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x})\right] + \boldsymbol{q}^\top\left[\boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x})\right] \\ &= \frac{1}{2}\alpha^2(\boldsymbol{z} - \boldsymbol{x})^\top \boldsymbol{R}(\boldsymbol{z} - \boldsymbol{x}) + \alpha(\boldsymbol{z} - \boldsymbol{x})^\top(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{q}) + f(\boldsymbol{x}), \qquad (3.11) \\ \frac{\mathrm{d}g(\alpha)}{\mathrm{d}\alpha} &= \alpha(\boldsymbol{z} - \boldsymbol{x})^\top \boldsymbol{R}(\boldsymbol{z} - \boldsymbol{x}) + (\boldsymbol{z} - \boldsymbol{x})^\top(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{q}). \qquad (3.12) \end{aligned}$$

Solving $\frac{\mathrm{d}g(\alpha)}{\mathrm{d}\alpha} = 0$ gives

$$\alpha^* = \frac{(\boldsymbol{x} - \boldsymbol{z})^\top(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{q})}{(\boldsymbol{x} - \boldsymbol{z})^\top \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{z})}. \qquad (3.13)$$

Therefore the parameter $\alpha^{(k)}$ of (3.9) can be computed by

$$\alpha^{(k)} = \min\left\{\max\left\{\frac{(\boldsymbol{x}^{(k)} - \boldsymbol{z}^{(k)})^\top(\boldsymbol{R}\boldsymbol{x}^{(k)} + \boldsymbol{q})}{(\boldsymbol{x}^{(k)} - \boldsymbol{z}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)} - \boldsymbol{z}^{(k)})}, 0\right\}, 1\right\}. \qquad (3.14)$$

Since exact line search's big O complexity is the same as that of one iteration of Armijo rule line search while the computed stepsize is optimal, using exact line search is better.

Regarding the convergence rate of GP, it is essentially the same as its unconstrained counterpart, i.e., the gradient descent method. Therefore, in general, GP is a sublinear convergence algorithm where the number of iterations required to achieve $f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^*) < \epsilon$ is $\mathcal{O}(1/\epsilon)$, where $\boldsymbol{x}^*$ denotes the optimal solution. If $\boldsymbol{R}$ is furthermore positie definite, then GP can achieve linear convergence rate [80]. The GP algorithm that solves problem (**QP**) is shown in Algorithm 6.

3.3.1.2 In search of $\alpha^{(k)}$ and $\beta^{(k)}$ (case c)

BBGP is a special case of GP under this parameter settings. Barzilai and Borwein proposed this variation of the gradient descent method in the context of unconstrained minimization of a smooth nonlinear function [76]. It can be viewed as an approximation of Newton's method that achieves superlinear convergence rate in the ideal case. In each iteration of BBGP, the update rule of the decision variable $\boldsymbol{x}$ is

$$\boldsymbol{z}^{(k)} = \Pi_{\mathcal{X}}\left(\boldsymbol{x}^{(k)} - \underbrace{\xi^{(k)}\boldsymbol{H}^{(k)}}_{\beta^{(k)}}\nabla f(\boldsymbol{x}^{(k)})\right), \tag{3.15a}$$

$$\alpha^{(k)} = \arg\min_{0\le\alpha\le 1} f(\boldsymbol{x}^{(k)} + \alpha(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)})), \tag{3.15b}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)}(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}), \tag{3.15c}$$

where $\boldsymbol{H}^{(k)}$ is positive definite and is the so-called scaling matrix in the kth iteration. In gradient descent methods, $\boldsymbol{H}^{(k)} = \boldsymbol{I}$; in Newton's methods, $\boldsymbol{H}^{(k)} = \left[\nabla^2 f(\boldsymbol{x}^{(k)})\right]^{-1}$, which is the inverse of Hessian. Obviously, $\boldsymbol{H}^{(k)}$ plays a core role that affects the convergence rate of the method. The trade-off of the faster convergence rate of Newton's method goes more to the computation of the matrix inverse $(\nabla^2 f(\boldsymbol{x}^{(k)}))^{-1}$. Therefore people later started to pursue methods that use scaling matrices with properties as close as possible to the inverse of Hessian in a more efficient way. Here comes an important branch of gradient-based methods—quasi-Newton's methods. The idea is as follows: $(\nabla^2 f(\boldsymbol{x}^{(k)}))^{-1}$ can be approximated by using the mean-value theorem as

$$\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)} \approx \underbrace{\left[\nabla^2 f(\boldsymbol{x}^{(k)})\right]^{-1}}_{\boldsymbol{H}^{(k)}}(\nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})), \tag{3.16}$$

or equivalently,

$$\underbrace{\nabla^2 f(\boldsymbol{x}^{(k)})}_{(\boldsymbol{H}^{(k)})^{-1}}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}) \approx \nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)}), \tag{3.17}$$

which is called the secant condition when equality holds. Quasi-Newton's methods avoid direct computation of $(\nabla^2 f(\boldsymbol{x}^{(k)}))^{-1}$ by searching a matrix $\boldsymbol{H}^{(k)}$ that approximately satisfies (3.17). Motivated by the scaling matrices approach as quasi-Newton methods do, Barzilai and Borwein take $\xi^{(k)} = 1$ in (3.15a) and predefine a simple structure

$$\boldsymbol{H}^{(k)} = \beta^{(k)}\boldsymbol{I} = \Pi_{[\beta_{\min},\beta_{\max}]}\left(\frac{1}{\eta^{(k)}}\right)\boldsymbol{I} \tag{3.18}$$

for the scaling matrix and search for $\eta^{(k)}$ that best fits the secant condition in a ℓ_2 norm sense. Mathematically, $\eta^{(k)}$ is chosen by

$$\eta^{(k)} = \min_{\eta} \quad g(\boldsymbol{x}^{(k)}, \boldsymbol{z}^{(k)}, \eta) \tag{3.19}$$

where

$$g(\boldsymbol{x}, \boldsymbol{y}, \eta) = \|(\boldsymbol{x} - \boldsymbol{y})\eta - (\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))\|_2^2. \tag{3.20}$$

In practice, with the extra computation cost, the performance is improved comparing with plain Gradient Projection Method in terms convergence rate [76]. The idea is later extended to GP Method by Figueiredo *et al.* in [83], and, in this thesis, is called Barzilai-Borwein Gradient Projection Method (BBGP). Still, we consider the BBGP for problem (**QP**). In each iteration, BBGP searches for both $\alpha^{(k)}$ and $\beta^{(k)}$, where $\alpha^{(k)}$ is obtained by exact line search, i.e., by (3.14) while $\beta^{(k)}$ is computed by solving (3.19). The derivation is shown below.

$$\begin{aligned}
g(\boldsymbol{x}, \boldsymbol{y}, \eta) &= \|(\boldsymbol{x} - \boldsymbol{y})\eta - (\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))\|_2^2 && (3.21a)\\
&= \|\boldsymbol{x} - \boldsymbol{y}\|_2^2\eta^2 - 2(\boldsymbol{x} - \boldsymbol{y})^\top(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))\eta \\
&\quad + \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2^2 && (3.21b)\\
&= \|\boldsymbol{x} - \boldsymbol{y}\|_2^2\eta^2 - 2(\boldsymbol{x} - \boldsymbol{y})^\top\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{y})\eta + \|\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{y})\|_2^2, && (3.21c)\\
\frac{\mathrm{d}g(\boldsymbol{x}, \boldsymbol{y}, \eta)}{\mathrm{d}\eta} &= 2\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\eta - 2(\boldsymbol{x} - \boldsymbol{y})^\top\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{y}). && (3.21d)
\end{aligned}$$

Solving $\dfrac{\mathrm{d}g(\boldsymbol{x},\boldsymbol{y},\eta)}{\mathrm{d}\eta}=0$ gives the solution of (3.19).

$$\eta^* = \frac{(\boldsymbol{x}-\boldsymbol{y})^\top \boldsymbol{R}(\boldsymbol{x}-\boldsymbol{y})}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}. \tag{3.22}$$

Therefore by putting $\boldsymbol{x}=\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}=\boldsymbol{z}^{(k)}$, we can find $\eta^{(k)}$, and thus $\beta^{(k)}$, by

$$\beta^{(k)} = \Pi_{[\beta_{\min},\beta_{\max}]}\left(\frac{1}{\eta^{(k)}}\right) \tag{3.23a}$$

$$= \min\left\{\max\left\{\frac{\|\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)}\|_2^2}{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})},\beta_{\min}\right\},\beta_{\max}\right\}. \tag{3.23b}$$

Note that if we fix $\alpha^{(k)}$ and only update $\beta^{(k)}$ then the objective value is not guaranteed to have monotonic decrease. By using the exact line search on $\alpha^{(k)}$ this property can be kept [80] where the convergence rate is as if in its unconstrained version [84]. The BBGP algorithm that solves problem (**QP**) is shown in Algorithm 7.

Algorithm 6 Gradient Projection (GP) Method (case a)

Input: $\boldsymbol{R}$, $\boldsymbol{q}$, $\boldsymbol{x}^{(0)}\in\mathcal{X}$, $\beta>0$.

1: **for** $k=0,1,2,\cdots$ until stopping criteria holds **do**

2: $\quad \boldsymbol{z}^{(k)} \leftarrow \Pi_{\mathcal{X}}(\boldsymbol{x}^{(k)}-\beta(\boldsymbol{R}\boldsymbol{x}^{(k)}+\boldsymbol{q}))$.

3: $\quad \alpha^{(k)} \leftarrow \Pi_{[0,1]}\left(\frac{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top(\boldsymbol{R}\boldsymbol{x}^{(k)}+\boldsymbol{q})}{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})}\right)$ `// solves` $\min_{\alpha\in[0,1]} f(\boldsymbol{x}^{(k)}+\alpha(\boldsymbol{z}^{(k)}-\boldsymbol{x}^{(k)}))$.

4: $\quad \boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)}+\alpha^{(k)}(\boldsymbol{z}^{(k)}-\boldsymbol{x}^{(k)})$.

5: **end for**

Output: $\boldsymbol{x}^{(k+1)}$.

Algorithm 7 Barzilai-Borwein Gradient Projection (BBGP) Method (case c)

Input: $\boldsymbol{R}$, $\boldsymbol{q}$, $\boldsymbol{x}^{(0)}\in\mathcal{X}$, $\beta^{(0)}\in[\beta_{\min},\beta_{\max}]$.

1: **for** $k=0,1,2,\cdots$ until stopping criteria holds **do**

2: $\quad \boldsymbol{z}^{(k)} \leftarrow \Pi_{\mathcal{X}}(\boldsymbol{x}^{(k)}-\beta^{(k)}(\boldsymbol{R}\boldsymbol{x}^{(k)}+\boldsymbol{q}))$.

3: $\quad \alpha^{(k)} \leftarrow \Pi_{[0,1]}\left(\frac{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top(\boldsymbol{R}\boldsymbol{x}^{(k)}+\boldsymbol{q})}{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})}\right)$ `// solves` $\min_{\alpha\in[0,1]} f(\boldsymbol{x}^{(k)}+\alpha(\boldsymbol{z}^{(k)}-\boldsymbol{x}^{(k)}))$.

4: $\quad \boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)}+\alpha^{(k)}(\boldsymbol{z}^{(k)}-\boldsymbol{x}^{(k)})$.

5: $\quad \eta^{(k+1)} \leftarrow \frac{(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})}{\|\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)}\|_2^2}$ `// solves` $\min_{\eta}\|(\boldsymbol{x}^{(k)}-\boldsymbol{z}^{(k)})\eta-(\nabla f(\boldsymbol{x}^{(k)})-\nabla f(\boldsymbol{z}^{(k)}))\|_2^2$.

6: $\quad \beta^{(k+1)} \leftarrow \Pi_{[\beta_{\min},\beta_{\max}]}\left(1/\eta^{(k+1)}\right)$.

7: **end for**

Output: $\boldsymbol{x}^{(k+1)}$.

3.3.2 Proximal Gradient Method (PG)

Proximal Gradient Method (PG) is another branch of gradient-based method and is extensively used in this decade. Before providing details on PG, we first review the concept of prox-operator. Then we come to the details of PG in solving problem (**QP**).

3.3.2.1 Prox-operator

A prox-operator is an essence of PG. Given a convex function h, its prox-operator is defined as

$$\texttt{prox}_h(\boldsymbol{x}) = \arg\min_{\boldsymbol{u}} \left(h(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2 \right), \tag{3.24}$$

which searches a point $\boldsymbol{u}$ that decreases h while $\boldsymbol{u}$ is not too far away from $\boldsymbol{x}$. The prox-operator (3.24) is strongly convex so that it admits a unique solution. Note that we only require h to be convex, i.e., h can be a nondifferentiable function like the ℓ_1-norm of a vector, indicator function $\mathcal{I}_{\mathcal{X}}$ of a convex set $\mathcal{X}$. In the context of convex constrained QP, we take $h(\cdot)$ as the indicator function $\mathcal{I}_{\mathcal{X}}(\cdot)$. The definition of $\mathcal{I}_{\mathcal{X}}(\cdot)$ is restated below:

$$\mathcal{I}_{\mathcal{X}} = \begin{cases} 0, & \boldsymbol{x} \in \mathcal{X} \\ \infty, & \text{otherwise.} \end{cases} \tag{3.25}$$

It can be verified that

$$\texttt{prox}_{\mathcal{I}_{\mathcal{X}}}(\boldsymbol{x}) = \Pi_{\mathcal{X}}(\boldsymbol{x}). \tag{3.26}$$

Therefore the prox-operator is a generalization of convex set projection. Define $\boldsymbol{x}^*$ as a fixed point of (3.24) such that

$$\texttt{prox}_h(\boldsymbol{x}^*) = \boldsymbol{x}^*. \tag{3.27}$$

An important property of prox-operator is that its fixed point is the minimizer of h.

3.3.2.2 PG in Solving QP

PG is a FOGM equipped with a prox-operator for handling constrained or nondifferentiable convex problems. Specifically, PG is capable of handling unconstrained problems of the form

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) \;=\; \boldsymbol{G}(\boldsymbol{x}) + H(\boldsymbol{x}), \tag{3.28}$$

where $G(\boldsymbol{x})$ is convex and differentiable everywhere and $H(\boldsymbol{x})$ is closed, convex and possibly nondifferentiable. They are widely applicable and have numerous applications especially in areas with large and high-dimensional data. Now we equivalently rewrite problem (**QP**) as

$$\min_{\boldsymbol{x}} \underbrace{\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}}_{G(\boldsymbol{x})} + \underbrace{\mathcal{I}_{\mathcal{X}}(\boldsymbol{x})}_{H(\boldsymbol{x})}. \tag{3.29}$$

It is easy to verify that in (3.29) $G(\boldsymbol{x})$ is convex and differentiable while the $H(\boldsymbol{x})$ is closed, convex and nondifferentiable. Given an initial point $\boldsymbol{x}^{(0)}$, the standard PG recursively computes

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \texttt{prox}_{\mathcal{I}_{\mathcal{X}}}\left(\boldsymbol{x}^{(k)} - \beta^{(k)}\nabla f(\boldsymbol{x}^{(k)})\right) && \text{(3.30a)} \\ &= \Pi_{\mathcal{X}}\left(\boldsymbol{x}^{(k)} - \beta^{(k)}\nabla f(\boldsymbol{x}^{(k)})\right) && \text{(3.30b)} \end{aligned}$$

where $\beta^{(k)}$ is a positive scalar. Here we see a connection between PG and GP — in the context of QP and when we fix $\alpha^{(k)} = 1$ for all k in (3.6), PG is a special case of GP and the only difference goes to the choice of the inner stepsize $\beta^{(k)}$ in each iteration.

There are two popular strategies of choosing $\beta^{(k)}$, namely by Armijo rule line search and by the reciprocal of fixed Lipschitz constant of the gradient. By using either strategies, we can see the PG Method as a case of GP Method: if we use Armijo rule line search for $\beta^{(k)}$, the PG is exactly the same as GP case b) (fixing outer stepsize and searching for inner stepsize); if we choose $\beta^{(k)}$ by using the reciprocal of a fixed Lipschitz constant, PG becomes GP in case d) (fixed both outer and inner stepsizes).

In this thesis, the PG that we focus is the special case of GP using case d) approach. We first define the Lipschitz constant of a function. Then we come to the derivation of Lipschitz constant of the objective function in problem (**QP**).

For all $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and any valid norm $\|\cdot\|$, a function f is Lipschitz continuous in L if

$$\|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \tag{3.31}$$

and Lipschitz smooth in L if

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|. \tag{3.32}$$

Usually the smallest possible value of L of a function is of the most interest. To find L of gradient of f in problem (**QP**), we consider (3.32) using ℓ_2 norm and by mean value theorem we have

$$\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2) = \nabla^2 f(\boldsymbol{x}_m)^\top (\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{3.33a}$$

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\|_2 = \|\nabla^2 f(\boldsymbol{x}_m)^\top (\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \tag{3.33b}$$

$$\leq \|\nabla^2 f(\boldsymbol{x}_m)\|_2 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2, \tag{3.33c}$$

where $\boldsymbol{x}_m$ is a point between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Our aim is to find an upper bound of $\|\nabla^2 f(\boldsymbol{x})\|_2$, i.e., find L such that for all $\boldsymbol{x}$,

$$\|\nabla^2 f(\boldsymbol{x})\|_2 \leq L. \tag{3.34}$$

Since f is a quadratic function, the Hessian of f is the PSD matrix $\boldsymbol{R}$ of f, leaving us

$$L = \|\boldsymbol{R}\|_2 = \lambda_{\max}(\boldsymbol{R}), \tag{3.35}$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of the argument. When $\boldsymbol{R}$ is a big matrix, as encountered in problem (3.2) and (3.3), the eigendecomposition of the system matrix may become formidable by general algorithms. In this case alternative computation of L or exploring efficient formula of L should be considered so it can be computed more effectively.

Since $\boldsymbol{R}$ is unchanged throughout the iterative computation, L is a fixed positive constant. By setting $\beta^{(k)} = \beta \in (0, 2/L)$, for all k, the convergence of the PG method is guaranteed [85, 86]. Since a large amount of movement in each iteration is preferred, $2/L$ should be the best choice although traditionally $1/L$ is a more popular choice. The convergence rate is sublinear for general convex problems, and is linear for strongly convex problems. The PG algorithm that solves problem (**QP**) is shown in Algorithm 8.

Algorithm 8 Proximal Gradient (PG) Method

Input: $\boldsymbol{R}$, $\boldsymbol{q}$, $\boldsymbol{x}^{(0)} \in \mathcal{X}$.

1: $L = \lambda_{\max}(\boldsymbol{R})$.
2: $\beta = 1/L$.
3: **for** $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**
4: $\quad \boldsymbol{x}^{(k+1)} \leftarrow \Pi_{\mathcal{X}} \left(\boldsymbol{x}^{(k)} - \beta(\boldsymbol{R}\boldsymbol{x}^{(k)} + \boldsymbol{q})\right)$.
5: **end for**

Output: $\boldsymbol{x}^{(k+1)}$.

3.3.3 Fast Proximal Gradient Method (FISTA)

FISTA is a more superior PG-type method in terms of runtime performance. The Fast or sometimes called Accelerated PG was firstly proposed by Nesterov in 1983 [77] and was later popularized by Beck and Teboulle in 2009 [78]. Similar to PG, a positive scalar constant β is also required but the suitable range becomes $(0, 1/L]$. Also, the recursive computation of the decision variable does not directly reuse the result of the previous iteration. Instead, FISTA recursively computes the following in each iteration:

$$\boldsymbol{x}^{(k+1)} = \texttt{prox}_{\mathcal{I}_{\mathcal{X}}}\left(\boldsymbol{z}^{(k)} - \beta\nabla f(\boldsymbol{z}^{(k)})\right), \tag{3.36a}$$

$$\boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)} + \frac{\mu^{(k)} - 1}{\mu^{(k+1)}}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}), \tag{3.36b}$$

$$\mu^{(k+1)} = \frac{1 + \sqrt{1 + 4{\mu^{(k)}}^2}}{2}, \tag{3.36c}$$

where $\boldsymbol{z}^{(k)}$ is sometimes called an extrapolation point, which is determined from two previous computed points. The initial setting of FISTA is $\mu^{(0)} = 0$ and $\boldsymbol{x}^{(-1)} = \boldsymbol{x}^{(0)}$.

An inspiring result of FISTA is that, with trivial extra computation, it achieves an improved convergence rate of $\mathcal{O}(1/k^2)$ theoretically. This is the core reason why FISTA becomes so popular. The FISTA algorithm that solves problem (**QP**) is shown in Algorithm 9

Algorithm 9 Fast Proximal Gradient Method (FISTA)

Input: $\boldsymbol{R}$, $\boldsymbol{q}$, $\boldsymbol{x}^{(0)} \in \mathcal{X}$.

1: $L \leftarrow \lambda_{\max}(\boldsymbol{R})$.
2: $\beta \leftarrow 1/L$.
3: $\mu^{(0)} \leftarrow 0$, $\boldsymbol{x}^{(-1)} \leftarrow \boldsymbol{x}^{(0)}$.
4: **for** $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**
5: $\quad \mu^{(k+1)} \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4{\mu^{(k)}}^2}\right)$.
6: $\quad \boldsymbol{z}^{(k)} \leftarrow \boldsymbol{x}^{(k)} + \left[(\mu^{(k)} - 1)/\mu^{(k+1)}\right](\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)})$.
7: $\quad \boldsymbol{x}^{(k+1)} \leftarrow \Pi_{\mathcal{X}}\left(\boldsymbol{z}^{(k)} - \beta(\boldsymbol{R}\boldsymbol{z}^{(k)} + \boldsymbol{q})\right)$.
8: **end for**

Output: $\boldsymbol{x}^{(k+1)}$.

3.3.4 Frank-Wolfe Method (FW)

FW, sometimes called Conditional Gradient Method in other literatures, firstly appeared in a paper by Frank and Wolfe in 1956 [79]. Over the years, FW has seen a revival thanks to its ability to handle optimization problems with structured convex constraints in areas with big data. FW is computationally cheap when the problem is convex and has differentiable objective. A principal property of FW is its projection-free nature. Given a starting point, the standard FW recursively computes

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)}(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}), \tag{3.37}$$

where

$$\begin{aligned} \boldsymbol{z}^{(k)} &= \texttt{LMO}_{\mathcal{X}}\left(\nabla f(\boldsymbol{x}^{(k)})\right) && \text{(3.38a)} \\ &:= \arg\min_{\boldsymbol{z}\in\mathcal{X}} \nabla f(\boldsymbol{x}^{(k)})^\top(\boldsymbol{z} - \boldsymbol{x}^{(k)}) && \text{(3.38b)} \end{aligned}$$

is called the linear minimization orcale (LMO) of $\nabla f(\boldsymbol{x}^{(k)})$ on the convex set $\mathcal{X}$ and $\alpha^{(k)}$ is the stepsize in the kth iteration. An intuitive observation of using `LMO` is as follows. Recall that in convex optimization a necessary and sufficient condition for $\boldsymbol{x}^*$ being a global minimizer of f is

$$\nabla f(\boldsymbol{x}^*)^\top(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall\, \boldsymbol{x} \in \mathcal{X}, \tag{3.39}$$

which is called the first order optimality condition and the equality holds whenever $\boldsymbol{x} = \boldsymbol{x}^*$. By knowing this, FW forces $\boldsymbol{x}^{(k)}$ to update along a direction that violates the inequality most, hoping that the updated point $\boldsymbol{x}^{(k+1)}$ reaches the global optimum.

Specifically, FW finds a LMO (a direction) of $\boldsymbol{x}^{(k)}$ within $\mathcal{X}$ and then comes up with a new point $\boldsymbol{x}^{(k+1)}$ that is along the path between $\boldsymbol{x}^{(k)}$ and the LMO. By this update scheme, $\boldsymbol{x}^{(k+1)}$ is guaranteed to reside in $\mathcal{X}$ due to its convexity, avoiding projection onto $\mathcal{X}$.

Note that each LMO is found by solving a linear program (LP) on $\mathcal{X}$. When $\mathcal{X}$ is a polyhedron, at least one of the vertices of $\mathcal{X}$ belongs tothe solution set. Generally, LMO requires trivial computational resource compared with projection operations. Taking cases relevant to the HSR problem, $\mathcal{X}$ may become one of the following:

- when $\mathcal{X} = [0,1]^n$, we have

$$\texttt{LMO}_{\mathcal{X}}(\boldsymbol{y}) = \Big[\texttt{sgn}(y_i)\Big]_{i=1}^{n}, \tag{3.40}$$

  where $\texttt{sgn}(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0; \end{cases}$

- when $\mathcal{X} = \triangle^n$, we have

$$\texttt{LMO}_{\mathcal{X}}(\boldsymbol{y}) = \boldsymbol{e}_h, \tag{3.41}$$

  where $\boldsymbol{e}_h$ is a unit vector whose h^{th} element is active; h is chosen such that $y_h \leq y_i,\ \forall\, i$.

- when $\mathcal{X} = \mathbb{R}^n_+$, the situation is a bit subtle — $\mathcal{X}$ is a non-polyhedral convex set. Since the optimal $\boldsymbol{x}^*$ must be bounded, we may take a heuristic but reasonable replacement of $\mathbb{R}^n_+$ by $[0,U]^n$, where U is significantly large. The remains are the same as then case when $\mathcal{X} = [0,1]^n$.

FW requires light computation in both $\nabla f(\boldsymbol{x})$ and $\texttt{LMO}_{\mathcal{X}}\left[\nabla f(\boldsymbol{x})\right]$ in order to benefit from the light complexity of FW in each iteration. Thanks to the QP structure of f and the aforementioned convex sets $\mathcal{X}$, the LMO here fulfills both requirement. This advantage becomes more significant when the dimension of the data n increases and the corresponding projections are heavily decided by the dimension.

Frank and Wolfe have shown that FW has monotonic convergence [79]. Beck and Teboulle later showed that it has a sublinear convergence rate in general and a linear convergence rate if the objective is strongly convex and if the optimal point is in the strict interior of the constraint set [87].

Unlike GP and PG Methods, Armijo rule line search of $\alpha^{(k)}$ in (3.37) is unsuitable for FW because the searching arc (the path between $\boldsymbol{x}^{(k)}$ and $\boldsymbol{z}^{(k)}$) is not along the steepest descent direction. Nevertheless, a decreasing stepsize like $\alpha^{(k)} = 2/(k+2)$ or a stepsize obtained from exact line search are still applicable. In our QP case, exact line search scheme is more preferable. The FW algorithm that solves problem (**QP**) is shown in Algorithm 10.

Algorithm 10 Frank-Wolfe Method (FW)

Input: $\boldsymbol{R}$, $\boldsymbol{q}$, $\boldsymbol{x}^{(0)} \in \mathcal{X}$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

  $\gamma^{(k)} \leftarrow \boldsymbol{R}\boldsymbol{x}^{(k)} + \boldsymbol{q}$ `// computes` $\nabla f(\boldsymbol{x}^{(k)})$.

  $\boldsymbol{v}^{(k)} \leftarrow \texttt{LMO}_{\mathcal{X}}(\boldsymbol{\gamma}^{(k)})$ `// solves` $\min_{\boldsymbol{v}\in\mathcal{X}} \boldsymbol{v}^\top \boldsymbol{\gamma}^{(k)}$.

  $\alpha^{(k)} \leftarrow \Pi_{[0,1]}\left(\frac{(\boldsymbol{x}^{(k)}-\boldsymbol{v}^{(k)})^\top(\boldsymbol{R}\boldsymbol{x}^{(k)}+\boldsymbol{q})}{(\boldsymbol{x}^{(k)}-\boldsymbol{v}^{(k)})^\top \boldsymbol{R}(\boldsymbol{x}^{(k)}-\boldsymbol{v}^{(k)})}\right)$ `// solves` $\min_{\alpha\in[0,1]} f(\boldsymbol{x}^{(k)} + \alpha(\boldsymbol{v}^{(k)} - \boldsymbol{x}^{(k)}))$.

  $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \alpha^{(k)}(\boldsymbol{v}^{(k)} - \boldsymbol{x}^{(k)})$.

end for

Output: $\boldsymbol{x}^{(k+1)}$.

3.4 Proposed Algorithm Framework for HSR: ALternating Gradient-based Optimization (ALGO)

Our attention returns to the nonconvex HSR problem $(*)$. Problem $(*)$ has convex constrained quadratic subproblems so that we can tackle the problem by AO in a BCD manner. Let us denote the objective function of $(\mathscr{s})$ and $(\mathscr{a})$ as $f_s(\boldsymbol{A}, \boldsymbol{s})$ and $f_a(\boldsymbol{a}, \boldsymbol{S})$, respectively. We present a traditional algorithm framework of *Exact BCD* to solve problem $(*)$ in Algorithm 11.

Algorithm 11 Algorithm Framework of HSR via *Exact BCD* by FOGM

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

  $\boldsymbol{s}^{(k,0)} \leftarrow \texttt{vec}(\boldsymbol{S}^{(k)})$, $\boldsymbol{a}^{(k,0)} \leftarrow \texttt{vec}(\boldsymbol{A}^{(k)})$.

  `// S subproblem`

  for $j = 1, 2, \cdots, J$ **do**

    $\boldsymbol{s}^{(k,j)} \leftarrow \texttt{FOGM_Upd}(f_s(\boldsymbol{A}^{(k)}, \boldsymbol{s}^{(k,j-1)}))$.

  end for

  $\boldsymbol{S}^{(k+1)} \leftarrow \texttt{reshape}(\boldsymbol{s}^{(k,J)})$.

  `// A subproblem`

  for $j = 1, 2, \cdots, J$ **do**

    $\boldsymbol{a}^{(k,j)} \leftarrow \texttt{FOGM_Upd}(f_a(\boldsymbol{a}^{(k,j-1)}, \boldsymbol{S}^{(k+1)}))$.

  end for

  $\boldsymbol{A}^{(k+1)} \leftarrow \texttt{reshape}(\boldsymbol{a}^{(k,J)})$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

From the algorithm, the $\boldsymbol{S}$ and $\boldsymbol{A}$ subproblems are *exactly* solved by a gradient method. $\texttt{FOGM_Upd}(\cdot)$ refers to a first order gradient method update such that in practice it can

be replaced either by $\texttt{GP_Upd}(\cdot)$, $\texttt{BBGP_Upd}(\cdot)$, $\texttt{PG_Upd}(\cdot)$, $\texttt{FISTA_Upd}(\cdot)$, $\texttt{FW_Upd}(\cdot)$, or any other FOGMs. Typically, in order to solve the subproblems *exactly* or to certain exactness, the gradient update should be repeated many times, i.e., J should be sufficiently large. A drawback is that whenever J is large, the requires much runtime to finish an iteration of BCD.

In this thesis, we propose ALGO, an algorithm framework of *inexact BCD* for solving problem $(*)$. Specifically, we use small value J, with an extreme case as taking $J = 1$, to conduct *inexact solving of the subproblems*, i.e., *Inexact BCD*. The algorithm flow is similar to that in Algorithm 11 except that both $\boldsymbol{S}$ and $\boldsymbol{A}$ subproblems are briefly solved by a one-time gradient update. The algorithm of the proposed ALGO is presented in Algorithm 12.

Algorithm 12 ALGO (Algorithm Framework of HSR via *Inexact BCD* by FOGM)

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

1: **for** $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

2: $\quad \boldsymbol{s}^{(k)} \leftarrow \texttt{vec}(\boldsymbol{S}^{(k)})$, $\boldsymbol{a}^{(k)} \leftarrow \texttt{vec}(\boldsymbol{A}^{(k)})$.

// `S subproblem`

3: $\quad \boldsymbol{S}^{(k+1)} \leftarrow \texttt{reshape}\left(\texttt{FOGM_Upd}(f_s(\boldsymbol{A}^{(k)}, \boldsymbol{s}^{(k)}))\right)$.

// `A subproblem`

4: $\quad \boldsymbol{A}^{(k+1)} \leftarrow \texttt{reshape}\left(\texttt{FOGM_Upd}(f_a(\boldsymbol{a}^{(k)}, \boldsymbol{S}^{(k+1)}))\right)$.

5: **end for**

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

We present the detailed implementation of ALGO using each discussed FOGM in the remaining of this chapter. In the algorithm description, we present a generalized ALGO algorithm that one can decide the exactness of the subproblems in the BCD by assigning J with suitable value. A large J enforces the HSR algorithms to employ exact BCD while a small J enforces the HSR algorithms to employ inexact BCD.

3.4.1 ALGO using GP

Algorithm 13 ALGO using GP

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$, $\beta_S, \beta_A > 0$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

`// S-subproblem`

$\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{A} \leftarrow \boldsymbol{A}^{(k)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top \left(\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}}\right)\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top \left(\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}}\right)$.

$\boldsymbol{Z}_S^{(k,j-1)} \leftarrow \Pi_{\mathcal{S}}\left(\boldsymbol{S}^{(k,j-1)} - \beta_S \boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

$\alpha_S^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G}\,,\, \boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\,,\, \boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\,\rangle}{\|\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\|_{\mathsf{F}}^2}$.

$\alpha_S^{(k,j-1)} \leftarrow \min(\max(\alpha_S^*, 0), 1)$.

$\boldsymbol{S}^{(k,j)} \leftarrow \boldsymbol{S}^{(k,j-1)} + \alpha_S^{(k,j-1)}\left(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)}\right)$.

end for

$\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

`// A-subproblem`

$\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(k+1)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow \left(\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}}\right)(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top \left(\boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}}\right)\boldsymbol{S}^\top$.

$\boldsymbol{Z}_A^{(k,j-1)} \leftarrow \Pi_{\mathcal{A}}\left(\boldsymbol{A}^{(k,j-1)} - \beta_A \boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

$\alpha_A^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G}\,,\, (\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j)})\boldsymbol{S}\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\,,\, \boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\,\rangle}{\|(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\|_{\mathsf{F}}^2}$.

$\alpha_A^{(k,j-1)} \leftarrow \min(\max(\alpha_A^*, 0, 1))$.

$\boldsymbol{A}^{(k,j)} \leftarrow \boldsymbol{A}^{(k,j-1)} + \alpha_A^{(k,j-1)}\left(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)}\right)$.

end for

$\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

3.4.2 ALGO using BBGP

Algorithm 14 ALGO using BBGP

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$, $\beta_S, \beta_A \in [\beta_{\min}, \beta_{\max}]$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

`// S-subproblem`

$\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{A} = \boldsymbol{A}^{(k)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top \left(\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}}\right)\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top \left(\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}}\right)$.

$\boldsymbol{Z}_S^{(k,j-1)} \leftarrow \Pi_{\mathcal{S}}\left(\boldsymbol{S}^{(k,j-1)} - \beta_S^{(k,j-1)}\boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

$\alpha_S^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G}\,,\, \boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\,,\, \boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\,\rangle}{\|\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\|_{\mathsf{F}}^2}$.

$\alpha_S^{(k,j-1)} \leftarrow \min(\max(\alpha_S^*, 0), 1)$.

$\boldsymbol{S}^{(k,j)} \leftarrow \boldsymbol{S}^{(k,j-1)} + \alpha_S^{(k,j-1)}\left(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)}\right)$.

$\eta_S^{(k,j-1)} \leftarrow \frac{\|\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\|_{\mathsf{F}}^2}{\|\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)}\|_{\mathsf{F}}^2}$.

$\beta_S^{(k,j)} \leftarrow \min\left(\max\left(1/\eta_S^{(k,j-1)}, \beta_{\min}\right), \beta_{\max}\right)$.

end for

$\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

`// A-subproblem`

$\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{S} = \boldsymbol{S}^{(k+1)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow \left(\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}}\right)(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top\left(\boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}}\right)\boldsymbol{S}^\top$.

$\boldsymbol{Z}_A^{(k,j-1)} \leftarrow \Pi_{\mathcal{A}}\left(\boldsymbol{A}^{(k,j-1)} - \beta_A^{(k,j-1)}\boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

$\alpha_A^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G}\,,\, (\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\,,\, \boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\,\rangle}{\|(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\|_{\mathsf{F}}^2}$.

$\alpha_A^{(k,j-1)} \leftarrow \min(\max(\alpha_A^*, 0, 1))$.

$\boldsymbol{A}^{(k,j)} \leftarrow \boldsymbol{A}^{(k,j-1)} + \alpha_A^{(k,j-1)}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})$.

$\eta_A^{(k,j-1)} \leftarrow \frac{\|(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\|_{\mathsf{F}}^2}{\|\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)}\|_{\mathsf{F}}^2}$.

$\beta_A^{(k,j)} \leftarrow \min\left(\max\left(1/\eta_A^{(k,j-1)}, \beta_{\min}\right), \beta_{\max}\right)$.

end for

$\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

3.4.3 ALGO using PG

PG requires the objective function to be Lipschitz smooth with constant L. In this section we first derive the Lipschitz constant of $\nabla_{\boldsymbol{s}} f_s(\boldsymbol{A}, \boldsymbol{s})$ and $\nabla_{\boldsymbol{a}} f_a(\boldsymbol{a}, \boldsymbol{S})$ as L_S and L_A, respectively.

Recall that $f_s(\boldsymbol{A}, \boldsymbol{s}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{H}_S \boldsymbol{s}\|_2^2$ indicates $L_S = \lambda_{\max}(\boldsymbol{H}_S^\top \boldsymbol{H}_S)$, where

$$\boldsymbol{H}_S^\top \boldsymbol{H}_S = (\boldsymbol{G}\boldsymbol{G}^\top) \otimes (\boldsymbol{A}^\top \boldsymbol{A}) + \boldsymbol{I}_L \otimes ((\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A}) \tag{3.42}$$

is a big matrix so that the maximum eigenvalue computation with complexity of $\mathcal{O}((NL)^3)$ is impractical. Nevertheless by exploiting the structure of $\boldsymbol{H}_S$ the complexity of computing L_S can be greatly reduced. The derivation is as follows. Let the SVD of $\boldsymbol{G}$ be

$$\boldsymbol{G} = \boldsymbol{U}_G \boldsymbol{\Sigma}_G \boldsymbol{V}_G^\top \tag{3.43}$$

such that $\boldsymbol{U}_G \in \mathbb{R}^{L \times L}$, $\boldsymbol{\Sigma}_G \in \mathbb{R}^{L \times L_{\mathsf{H}}}$ and $\boldsymbol{V}_G \in \mathbb{R}^{L_{\mathsf{H}} \times L_{\mathsf{H}}}$. We can write

$$\begin{aligned}
\boldsymbol{H}_S &= \begin{bmatrix} (\boldsymbol{V}_G \otimes \boldsymbol{I}_M)(\boldsymbol{\Sigma}_G^\top \otimes \boldsymbol{A})(\boldsymbol{U}_G^\top \otimes \boldsymbol{I}_N) \\ \boldsymbol{I}_L \otimes \boldsymbol{F}\boldsymbol{A} \end{bmatrix} && (3.44a) \\
&= \underbrace{\begin{bmatrix} \boldsymbol{V}_G \otimes \boldsymbol{I}_M & \\ & \boldsymbol{I}_{M_M L} \end{bmatrix}}_{\boldsymbol{Q}_1} \begin{bmatrix} (\boldsymbol{\Sigma}_G^\top \otimes \boldsymbol{A})(\boldsymbol{U}_G^\top \otimes \boldsymbol{I}_N) \\ \boldsymbol{I}_L \otimes \boldsymbol{F}\boldsymbol{A} \end{bmatrix} && (3.44b) \\
&= \boldsymbol{Q}_1 \underbrace{\begin{bmatrix} \boldsymbol{\Sigma}_G^\top \otimes \boldsymbol{A} \\ \boldsymbol{U}_G \otimes \boldsymbol{F}\boldsymbol{A} \end{bmatrix}}_{\tilde{\boldsymbol{H}}_S} \underbrace{\boldsymbol{U}_G^\top \otimes \boldsymbol{I}_N}_{\boldsymbol{Q}_2}, && (3.44c)
\end{aligned}$$

where $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are othonormal so that the expression

$$\boldsymbol{H}_S = \boldsymbol{Q}_1 \tilde{\boldsymbol{H}}_S \boldsymbol{Q}_2 \tag{3.45}$$

indicates that $\boldsymbol{H}_S$ and $\tilde{\boldsymbol{H}}_S$ share the same set of singular values. Also, let $\boldsymbol{C}_1 = \boldsymbol{A}^\top \boldsymbol{A}$, $\boldsymbol{C}_2 = (\boldsymbol{F}\boldsymbol{A})^\top (\boldsymbol{F}\boldsymbol{A})$ and $\sigma_{G,1} \geq \cdots \geq \sigma_{G,L_\mathsf{H}}$, we can see that,

$$\begin{aligned}
\tilde{\boldsymbol{H}}_S^\top \tilde{\boldsymbol{H}}_S &= (\boldsymbol{\Sigma}_G \boldsymbol{\Sigma}_G^\top) \otimes (\boldsymbol{A}^\top \boldsymbol{A}) + \boldsymbol{I}_L \otimes ((\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A}) && (3.46a)\\
&= \begin{bmatrix} \sigma_{G,1}^2 \boldsymbol{C}_1 + \boldsymbol{C}_2 & & & & & \\ & \ddots & & & & \\ & & \sigma_{G,L_\mathsf{H}}^2 \boldsymbol{C}_1 + \boldsymbol{C}_2 & & & \\ & & & \boldsymbol{C}_2 & & \\ & & & & \ddots & \\ & & & & & \boldsymbol{C}_2 \end{bmatrix}, && (3.46b)
\end{aligned}$$

which shows the largest singular value of $\tilde{\boldsymbol{H}}_S$ is

$$\begin{aligned}
\sigma_{\max}(\tilde{\boldsymbol{H}}_S) &= \sqrt{\lambda_{\max}(\tilde{\boldsymbol{H}}_S^\top \tilde{\boldsymbol{H}}_S)} && (3.47a)\\
&= \sqrt{\lambda_{\max}(\sigma_{G,1}^2 \boldsymbol{A}^\top \boldsymbol{A} + (\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A})}, && (3.47b)
\end{aligned}$$

where $\sigma_{\max}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the largest singular and largest eigenvalue of its argument, respectively. This concludes that

$$\begin{aligned}
L_S &= \sigma_{\max}(\boldsymbol{H}_S)^2 && (3.48a)\\
&= \sigma_{\max}(\tilde{\boldsymbol{H}}_S)^2 && (3.48b)\\
&= \lambda_{\max}(\sigma_{G,1}^2 \boldsymbol{A}^\top \boldsymbol{A} + (\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A}), && (3.48c)
\end{aligned}$$

where the eigendecomposition complexity is $\mathcal{O}(N^3)$.

A similar derivation can be applied to the Lipschitz constant analysis of $\nabla_{\boldsymbol{a}} f_a(\boldsymbol{a}, \boldsymbol{S})$. The simplified formula of L_A is

$$L_A = \lambda_{\max}(\sigma_{F,1}^2 \boldsymbol{S}\boldsymbol{S}^\top + \boldsymbol{S}\boldsymbol{G}(\boldsymbol{S}\boldsymbol{G})^\top), \tag{3.49}$$

where $\sigma_{F,1}$ denotes the largest singular value of $\boldsymbol{F}$ and the total computation complexity is also $\mathcal{O}(N^3)$.

Algorithm 15 ALGO using PG

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

$\theta_G \leftarrow \sigma_{\max}(\boldsymbol{G})^2$, $\theta_F \leftarrow \sigma_{\max}(\boldsymbol{F})^2$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

// `S-subproblem`

$\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{A} \leftarrow \boldsymbol{A}^{(k)}$.

$L_S \leftarrow \lambda_{\max}(\theta_G \boldsymbol{A}^\top \boldsymbol{A} + (\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A})$.

$\beta_S \leftarrow 1/L_S$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top(\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}})$.

$\boldsymbol{S}^{(k,j)} \leftarrow \Pi_{\mathcal{S}}\left(\boldsymbol{S}^{(k,j-1)} - \beta_S \boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

end for

$\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

// `A-subproblem`

$\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(k+1)}$.

$L_A \leftarrow \lambda_{\max}(\theta_F \boldsymbol{S}\boldsymbol{S}^\top + \boldsymbol{S}\boldsymbol{G}(\boldsymbol{S}\boldsymbol{G})^\top)$.

$\beta_A \leftarrow 1/L_A$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow (\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top(\boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}})\boldsymbol{S}^\top$.

$\boldsymbol{A}^{(k,j)} \leftarrow \Pi_{\mathcal{A}}\left(\boldsymbol{A}^{(k,j-1)} - \beta_A \boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

end for

$\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

3.4.4 ALGO using FISTA

Algorithm 16 ALGO using FISTA

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

$\boldsymbol{S}^{(-1)} \leftarrow \boldsymbol{S}^{(0)}$, $\boldsymbol{A}^{(-1)} \leftarrow \boldsymbol{A}^{(0)}$.

$\theta_G \leftarrow \sigma_{\max}(\boldsymbol{G})^2$, $\theta_F \leftarrow \sigma_{\max}(\boldsymbol{F})^2$, $\mu \leftarrow 0$, $\phi \leftarrow 0$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

`// S-subproblem`

  $\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{S}^{(k,-1)} \leftarrow \boldsymbol{S}^{(k-1)}$, $\boldsymbol{A} \leftarrow \boldsymbol{A}^{(k)}$.

  $L_S \leftarrow \lambda_{\max}\left(\theta_G \boldsymbol{A}^\top \boldsymbol{A} + (\boldsymbol{F}\boldsymbol{A})^\top \boldsymbol{F}\boldsymbol{A}\right)$.

  $\beta_S \leftarrow 1/L_S$.

  for $j = 1, 2, \cdots, J$ **do**

    $\mu' \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4\mu^2}\right)$.

    $\boldsymbol{Z}_S^{(k,j-1)} \leftarrow \boldsymbol{S}^{(k,j-1)} + \frac{\mu'-1}{\mu}\boldsymbol{S}^{(k,j-2)}$.

    $\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{Z}_S^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top(\boldsymbol{F}\boldsymbol{A}\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}})$.

    $\boldsymbol{S}^{(k,j)} \leftarrow \Pi_{\mathcal{S}}\left(\boldsymbol{S}^{(k,j-1)} - \beta_S \boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

    $\mu \leftarrow \mu'$.

  end for

  $\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

`// A-subproblem`

  $\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{A}^{(k,-1)} \leftarrow \boldsymbol{A}^{(k-1)}$ $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(k+1)}$.

  $L_A \leftarrow \lambda_{\max}\left(\theta_F \boldsymbol{S}\boldsymbol{S}^\top + \boldsymbol{S}\boldsymbol{G}(\boldsymbol{S}\boldsymbol{G})^\top\right)$.

  $\beta_A \leftarrow 1/L_A$.

  for $j = 1, 2, \cdots, J$ **do**

    $\phi' \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4\phi^2}\right)$.

    $\boldsymbol{Z}_A^{(k,j-1)} \leftarrow \boldsymbol{A}^{(k,j-1)} + \frac{\phi'-1}{\phi}\boldsymbol{A}^{(k,j-2)}$.

    $\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow (\boldsymbol{Z}_A^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top(\boldsymbol{F}\boldsymbol{Z}_A^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}})\boldsymbol{S}^\top$.

    $\boldsymbol{A}^{(k,j)} \leftarrow \Pi_{\mathcal{A}}\left(\boldsymbol{A}^{(k,j-1)} - \beta_A \boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

    $\phi \leftarrow \phi'$.

  end for

  $\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

3.4.5 ALGO using FW

Algorithm 17 ALGO using FW

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

// `S-subproblem`

$\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{A} \leftarrow \boldsymbol{A}^{(k)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top(\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}})$.

$\boldsymbol{Z}_S^{(k,j-1)} \leftarrow \mathtt{LMO}_{\mathcal{S}}\left(\boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

$\alpha_S^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G}\,,\, \boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\,,\, \boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\,\rangle}{\|\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\|_{\mathsf{F}}^2}$.

$\alpha_S^{(k,j-1)} \leftarrow \min\left(\max\left(\alpha_S^*, 0\right), 1\right)$.

$\boldsymbol{S}^{(k,j)} \leftarrow \boldsymbol{S}^{(k,j-1)} + \alpha_S^{(k,j-1)}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})$.

end for

$\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

// `A-subproblem`

$\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(k+1)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow (\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top(\boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}})\boldsymbol{S}^\top$.

$\boldsymbol{Z}_A^{(k,j-1)} \leftarrow \mathtt{LMO}_{\mathcal{A}}\left(\boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

$\alpha_A^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\boldsymbol{G}\,,\, (\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j)})\boldsymbol{S}\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}^{(k,j-1)}\boldsymbol{S}\,,\, \boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\,\rangle}{\|(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)})\boldsymbol{S}\|_{\mathsf{F}}^2}$.

$\alpha_A^{(k,j-1)} \leftarrow \min\left(\max\left(\alpha_A^*, 0, 1\right)\right)$.

$\boldsymbol{A}^{(k,j)} \leftarrow \boldsymbol{A}^{(k,j-1)} + \alpha_A^{(k,j-1)}\left(\boldsymbol{Z}_A^{(k,j-1)} - \boldsymbol{A}^{(k,j-1)}\right)$.

end for

$\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

3.4.6 ALGO using Hybrid BCD Method

ALGO using Hybrid BCD Method is a heuristic and hybrid approach in the sense that the subproblems can be handled by different kind of FOGMs. The motivation behind is that when certain kind of FOGM can gain advantages in terms of computation efficiency in some certain subproblems, our customized choice of FOGMs for each subproblem further improves the efficiency of ALGO.

Take the HSR problem as example. When the pixel number of HS and MS images grows larger, the projection operation becomes more expensive, requiring much computation time per each iteration. In this thesis we allow ALGO using Hybrid BCD Method to employ FW in the $\boldsymbol{S}$ subproblem and FISTA in the $\boldsymbol{A}$ subproblem, hoping that the runtime performance can have improvement.

Algorithm 18 ALGO using Hybrid BCD

Input: $\boldsymbol{Y}_{\mathsf{H}}$, $\boldsymbol{Y}_{\mathsf{M}}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{S}^{(0)} \in \mathcal{S}$, $\boldsymbol{A}^{(0)} \in \mathcal{A}$.

$\boldsymbol{A}^{(-1)} \leftarrow \boldsymbol{A}^{(0)}$.

$\theta_F \leftarrow \sigma_{\max}(\boldsymbol{F})^2$, $\phi \leftarrow 0$.

for $k = 0, 1, 2, \cdots$ until stopping criteria holds **do**

// `S-subproblem`

$\boldsymbol{S}^{(k,0)} \leftarrow \boldsymbol{S}^{(k)}$, $\boldsymbol{A} \leftarrow \boldsymbol{A}^{(k)}$.

for $j = 1, 2, \cdots, J$ **do**

$\boldsymbol{\Gamma}_S^{(k,j-1)} \leftarrow \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})\boldsymbol{G}^\top + (\boldsymbol{F}\boldsymbol{A})^\top(\boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)} - \boldsymbol{Y}_{\mathsf{M}})$.

$\boldsymbol{Z}_S^{(k,j-1)} \leftarrow \mathtt{LMO}_{\mathcal{S}}\left(\boldsymbol{\Gamma}_S^{(k,j-1)}\right)$.

$\alpha_S^* \leftarrow \frac{\langle\, \boldsymbol{Y}_{\mathsf{H}} - \boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\boldsymbol{G}\,,\, \boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\,\rangle + \langle\, \boldsymbol{Y}_{\mathsf{M}} - \boldsymbol{F}\boldsymbol{A}\boldsymbol{S}^{(k,j-1)}\,,\, \boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\,\rangle}{\|\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\boldsymbol{G}\|_{\mathsf{F}}^2 + \|\boldsymbol{F}\boldsymbol{A}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})\|_{\mathsf{F}}^2}$.

$\alpha_S^{(k,j-1)} \leftarrow \min\left(\max\left(\alpha_S^*, 0\right), 1\right)$.

$\boldsymbol{S}^{(k,j)} \leftarrow \boldsymbol{S}^{(k,j-1)} + \alpha_S^{(k,j-1)}(\boldsymbol{Z}_S^{(k,j-1)} - \boldsymbol{S}^{(k,j-1)})$.

end for

$\boldsymbol{S}^{(k+1)} \leftarrow \boldsymbol{S}^{(k,J)}$.

// `A-subproblem`

$\boldsymbol{A}^{(k,0)} \leftarrow \boldsymbol{A}^{(k)}$, $\boldsymbol{A}^{(k,-1)} \leftarrow \boldsymbol{A}^{(k-1)}$ $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(k+1)}$.

$L_A \leftarrow \lambda_{\max}\left(\theta_F \boldsymbol{S}\boldsymbol{S}^\top + \boldsymbol{S}\boldsymbol{G}(\boldsymbol{S}\boldsymbol{G})^\top\right)$.

$\beta_A \leftarrow 1/L_A$.

for $j = 1, 2, \cdots, J$ **do**

$\phi' \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4\phi^2}\right)$.

$\boldsymbol{Z}_A^{(k,j-1)} \leftarrow \boldsymbol{A}^{(k,j-1)} + \frac{\phi'-1}{\phi}\boldsymbol{A}^{(k,j-2)}$.

$\boldsymbol{\Gamma}_A^{(k,j-1)} \leftarrow (\boldsymbol{Z}_A^{(k,j-1)}\boldsymbol{S}\boldsymbol{G} - \boldsymbol{Y}_{\mathsf{H}})(\boldsymbol{S}\boldsymbol{G})^\top + \boldsymbol{F}^\top(\boldsymbol{F}\boldsymbol{Z}_A^{(k,j-1)}\boldsymbol{S} - \boldsymbol{Y}_{\mathsf{M}})\boldsymbol{S}^\top$.

$\boldsymbol{A}^{(k,j)} \leftarrow \Pi_{\mathcal{A}}\left(\boldsymbol{A}^{(k,j-1)} - \beta_A \boldsymbol{\Gamma}_A^{(k,j-1)}\right)$.

$\phi \leftarrow \phi'$.

end for

$\boldsymbol{A}^{(k+1)} \leftarrow \boldsymbol{A}^{(k,J)}$.

end for

Output: $\boldsymbol{S}^{(k+1)}$, $\boldsymbol{A}^{(k+1)}$.

Chapter 4

Simulations

In this chapter, we compare the effectiveness of ALGO with state-of-the-art algorithms in a number of synthetic data and semi-real data simulations. They are quantitatively compared in terms of super-resolution and unmixing accuracy.

We first discuss the experimental details including environment settings, algorithm implementation details and their stopping criteria. Then we describe the synthetic data and semi-real data generation, followed by the definitions of the numerical quality measures. Finally we show the numerical results on synthetic data simulations and both numerical and pictoral results of the semi-real data simulations.

4.1 Experimental Details

The objective of all HSR experiments is to evaluate the estimation accuracy of the ground truth image (SR image) from its observed HS and MS degraded images. Since there are no real testing datasets for HSR evaluation purpose, a standard and well recognized approach was proposed by Wald *et al.* in [88] such that HS and MS images are synthesized from real spectral images where the optical response of the HS and MS sensors are simulated accordingly, as illustrated in Section 4.1.1. In this thesis we use the sensor response of AVIRIS and TM as the HS and MS camera pair response. The sensor response have already been described in Section 2.1.1 and 2.1.2. Following that, the method of simulating their response are shown in Section 4.1.2 and 4.1.3. Then, the synthetic and semi-real data

generation are specified in Section 4.1.4 and 4.1.5. Finally the initialization and stopping criteria of all algorithms are described in last two subsections.

4.1.1 Wald's Protocol

The proposed Wald's Protocol workflow is shown in Figure 4.1.

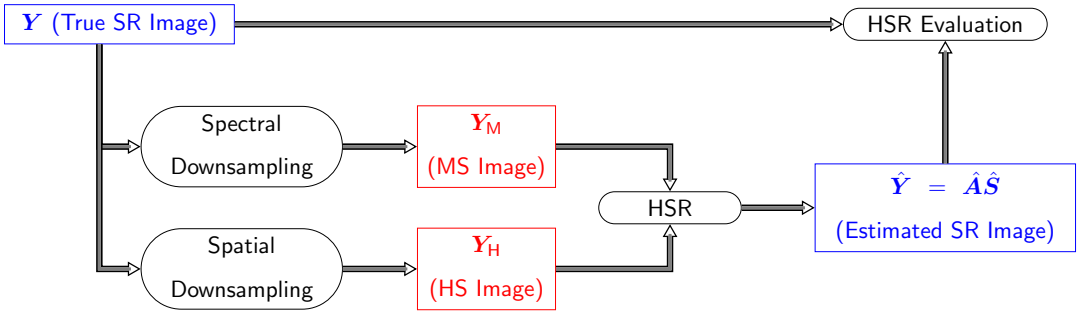


Figure 4.1: Wald's Protocol on HS and MS images synthesis and the evaluation of a testing HSR method.

Usually, to simulate real world scenario, white Gaussian noise of certain SNR level is added linearly to the synthesized HS and MS image pair after the downsampling processes. The addition of noise to both images are modelled in (2.5) and (2.7).

4.1.2 Spatial Degradation

In a recent work by Wei *et al.* , the spatial degradation system is designed to simulate a Gaussian low-pass filtering effect characterized by a 11×11 mask with variance $\sigma = 1.7$. After that the low-passed image undergoes subsampling operation of the image on every 4 pixels horizontally and vertically. The spatial degradation matrix $\boldsymbol{G}$ of the aforementioned specification is generated in a block circular circular block (BCCB) manner described in [48]. In this thesis we stick to the above spatial degradation settings and the $\boldsymbol{G}$ matrix used is exactly the same as the one Wei *et al.* used in their work. We also assume perfect prior information on $\boldsymbol{G}$ when we conduct HSR.

4.1.3 Spectral Degradation

The spectral response is simulated such that the MS image simulates the optical response of Landsat TM Sensors which capture spectral images at six wide bands between wavelength

$0.45\,\mu$m and $2.35\,\mu$m [36, 40]. For each TM band, the observed broad band image is simulated by naively taking the average of all band images within that broad passband. With reference to Table 2.1, the first TM band can be simulated by taking the average of AVIRIS image within $0.45\,\mu m$ and $0.52\,\mu m$, which corresponds to the 10th to the 16th band of an AVIRIS image. The same approach is applied to synthesize the remaining five TM band images.

4.1.4 Synthetic Data Generation

We generate 1000 trials per SNR level where the SR image $\boldsymbol{Y}$ are synthesized using the LMM. In each trial, the endmember matrix $\boldsymbol{A}$, containing N spectral signatures of spectral length $M = 224$, are randomly drawn from USGS Library [89]. Also, the abundance matrix $\boldsymbol{S}$, containing 100×100 pixels of uniform Dirichilet distribution, has maximum pixel purity $\rho = 0.85$. The definition of ρ is

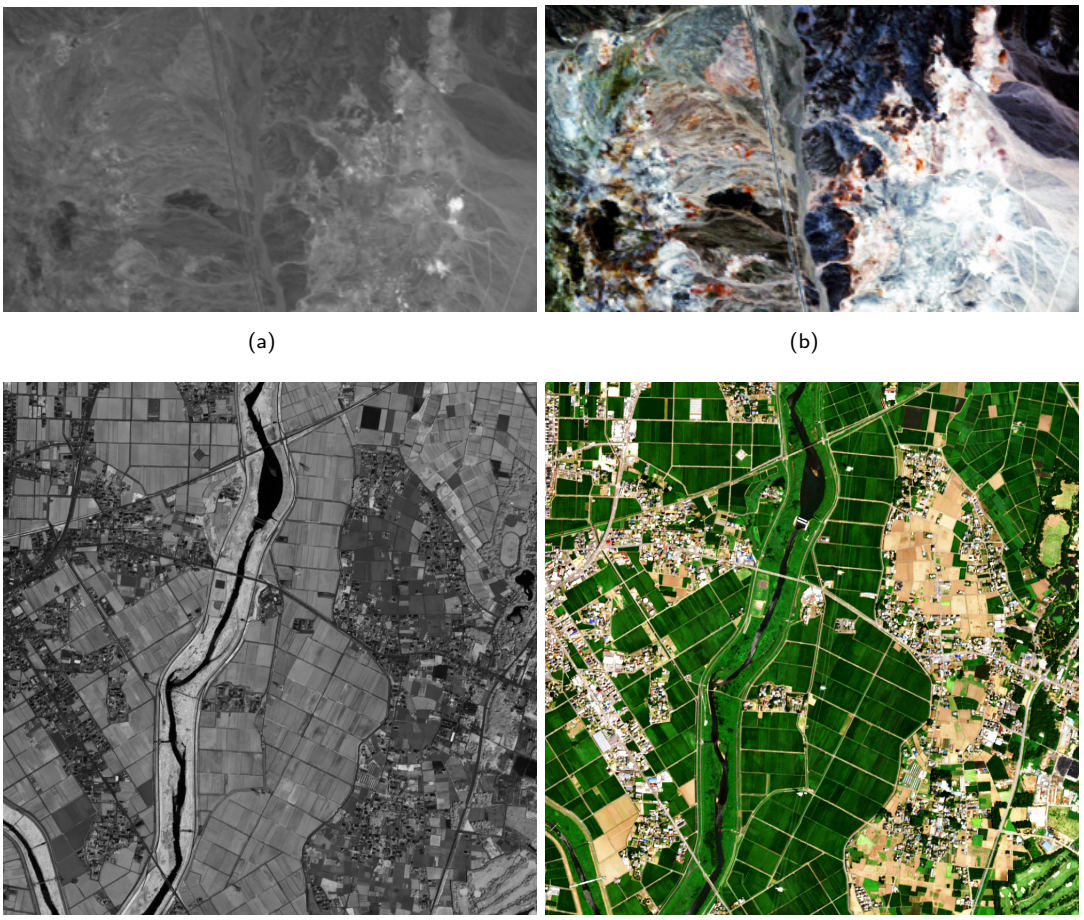
$$\rho = \max_{1 \leq i \leq L} \boldsymbol{s}_i^\top \boldsymbol{s}_i. \tag{4.1}$$

Note that $\boldsymbol{S}$ fulfills columnwise sum-to-one constraint. In the simulation we add white Gaussian noise at three SNR levels (40dB, 30dB and 20dB) to the HS and MS images.

4.1.5 Semi-real Data Generation

In the two sets of semi-real data simulations we regard the real spectral images measured by Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS) [38] and Headwall Hyperspec-VNIR-C Imaging Sensor [90, 91] as the ground truth SR images, respectively. Both sensors are airborne type having 20m and 2.5m GSD, respectively. The AVIRIS sensor takes spectral images at 224 bands covering wavelengths between $0.4\,\mu$m and $2.5\,\mu$m at norminally 10 nm intervals. It has taken many popular spectral image datasets including Cuprite Site (whom we use in the simulations), Indian Pines and Moffet Field. The Hyperspec-VNIR-C sensor takes spectral images at 128 bands covering wavelengths between $0.363\,\mu$m and $1.018\,\mu$m at norminally 5 nm intervals. We use a big spectral image that is taken by this sensor over Chikusei region of Japan in 2014 [90] and was later made public by Yokoya *et al.* .

In the first set of semi-real data simulations we fix a region of 200×348 pixels from the Cuprite Site dataset as the ground truth image. We then trancate the water absorption bands and noisy bands so that only 188 bands remain in the final SR image. In the next set



(a) (b)

(c) (d)

Figure 4.2: 815-nm band and the RGB image of the super-resolution images of Cuprite Site dataset ((a) and (b), 200×348 pixels) and Chikusei dataset ((c) and (d), 1000×1000 pixels).

of semi-real data simulations we fix a region of 1000×1000 pixels from the Chikusei dataset as the ground truth SR image. The semi-real HS and MS images are generated according to the Wald's Protocol. Since the spectral range of the Chikusei dataset only covers four TM bands, its simulated MS images have only four spectral bands. Additive white Gaussian noise of desired SNR level are added to HS and MS images as before. Note that ground truth endmembers and abundances of the datasets are unknown so we do not evaluate the HU performance.

Figure 4.2(a) and 4.2(c) visualize the normalized band images of the two real datasets at 815 nm, respectively.

4.1.6 Algorithm Initialization

In the general descriptions of all HSR Algorithms (ALGO, CNMF and FUMI), they all require an initial guess on the endmember and abundance matrix pair. Here we describe their initialization in our simulations.

The initial guess of the endmember matrix $\boldsymbol{A}^{(0)} \in \mathcal{A}$ is obtained by Successive Projection Algorithm (SPA) [92, 93] followed by projection onto the feasible set $\mathcal{A}$. The SPA algorithm aims at exactly indentifying the signature of all endmembers from a HS image with the following few assumptions: 1) model order N is known; 2) HS image is noiseless; 3) LMM holds; 4) abundance sum-to-one constraint holds; 5) for each material there exists at least a pixel that is entirely occupied by that material (i.e., pure pixel assumption holds). When these assumptions hold, SPA can search for the endmember signatures from the pure pixels of the HS image. We can then further project these signatures onto $\mathcal{A}$ and treat them as $\boldsymbol{A}^{(0)}$, i.e.,

$$\boldsymbol{A}^{(0)} \leftarrow \Pi_{\mathcal{A}}\left(\texttt{SPA}(\boldsymbol{Y}_{\mathsf{H}}, N)\right). \tag{4.2}$$

The algorithm description of SPA is shown below.

Algorithm 19 Successive Projection Algorithm (SPA)

Input: $\boldsymbol{Y}$, N.

$\boldsymbol{A}^{(0)} = [\,]$.
$\boldsymbol{P} \leftarrow \boldsymbol{I}$. `// define projection matrix.`
for $k = 1, 2, \cdots, N$ **do**
  $\ell \leftarrow \arg\max_{n\in[1,L]} \|\boldsymbol{P}\boldsymbol{y}_n\|_2^2$.
  $\hat{\boldsymbol{a}}_k \leftarrow \boldsymbol{y}_\ell$.
  $\boldsymbol{P} \leftarrow \left(\boldsymbol{I} - \left[(\boldsymbol{P}\hat{\boldsymbol{a}}_k)(\boldsymbol{P}\hat{\boldsymbol{a}}_k)^\top\right] / \|\boldsymbol{P}\hat{\boldsymbol{a}}_k\|_2^2\right)\boldsymbol{P}$. `// update projection matrix.`
  $\boldsymbol{A}^{(0)} \leftarrow \left[\boldsymbol{A}^{(0)}, \hat{\boldsymbol{a}}_k\right]$.
end for

Output: $\boldsymbol{A}^{(0)}$.

In [93], the rational of SPA is found to be similar to another endmember identification method called Successive Volume Maximization, which is also a kind of pure-pixel-searching method relying on the pure pixel assumption.

The initial guess of the abundance matrix is rather simple. We naively take a constant flat map as the initial guess of the abundance map, i.e.,

$$\boldsymbol{S}^{(0)} = \frac{1}{N}\mathbf{1}^{N\times L}, \tag{4.3}$$

which is always in the feasible sets considered in this thesis.

4.1.7 Stopping Criteria of Algorithms

We use the following condition to serve as all algorithms' stopping criteria. Let us reuse the definition of f in $(*)$ and further define $f^{(k)} := f(\boldsymbol{A}^{(k)}, \boldsymbol{S}^{(k)})$ as the objective value at the kth iteration. We terminate any algorithms whenever the stopping criteria

$$\left|\frac{f^{(k)} - f^{(k-1)}}{f^{(k)}}\right| \leq \delta \tag{4.4}$$

holds, for δ being a small positive scalar. Unless otherwise specified, we take $\delta = 10^{-4}$ in all simulations.

In ALGO, we always set $J = 1$ in all simulations. In FUMI, stopping criteria similar to (4.4) is used in termination of both subproblems in which the threshold δ_S and δ_A are 10^{-3}.

4.1.8 Algorithm Implementation

4.1.8.1 ALGO

We implement ALGO to solve the HSR problem by inexact BCD purely using one of the following FOGMs: GP, BBGP, PG, FISTA and FW. We also tried ALGO by updating $\boldsymbol{S}$ subproblem with FW and $\boldsymbol{A}$ subproblem with FISTA. For convenience, we call this approach "Hybrid BCD" composed of FW and FISTA. The Hybrid BCD is designed to speed up ALGO by avoiding simplex projection in the $\boldsymbol{S}$ subproblem (recall that in the HSR problem formulation of Wei *et al.* in (2.17), the constraint set of $\boldsymbol{S}$ is a unit simplex for all column). To further reduce runtime, we set $J = 1$ in all ALGO algorithm, i.e., we take a step of gradient update for each subproblem in the inexact BCD. We expect ALGO to be fast and that ALGO by Hybrid BCD has further advantage when model order N increases under the problem formulation by Wei *et al.* .

In general we set $\delta = 10^{-4}$ in most simulations except that in the simulations on Chikusei image we take $\delta = 10^{-2}$ for earlier termination for all algorithms.

4.1.8.2 FUMI

FUMI [48] is an ADMM-based algorithm that solves problem in an alternating optimization sense. Based on Algorithm 5 and its online code [1], FUMI is implemented with standard stopping criteria of ADMM, i.e., the primal feasibility and dual feasibility described in [72] where both primal and dual residual thresholds are set to 10^{-3}.

4.1.9 HU and HSR Quality Measures

The quality measures of HU is mean-squared-error (MSE) and that of HSR are spectral-angular-mapper (SAM) and peak-SNR (PSNR). The algorithm efficiency is compared by their CPU runtime. Our simulations are run on Dell Precision 7910 with dual Intel Xeon CPU E5-2650 v3 and 128GB memory. The definitions of the quality measures are summarized below.

4.1.9.1 MSE

The MSE measures the unmixing error of recovered endmember matrix $\boldsymbol{A}$ defined as

$$\text{MSE} = \min_{\boldsymbol{\pi}\in\mathcal{P}} \frac{1}{N}\sum_{k=1}^{N} \left|\left|\frac{\boldsymbol{a}_k}{\|\boldsymbol{a}_k\|_2} - \frac{\hat{\boldsymbol{a}}_{\pi_k}}{\|\hat{\boldsymbol{a}}_{\pi_k}\|_2}\right|\right|_2^2,$$

where $\mathcal{P}$ is the set of all permutation of $\{1, \cdots, N\}$ and $\hat{\boldsymbol{a}}_k$ is the estimate of $\boldsymbol{a}_k$. HU results with smaller MSE are better.

4.1.9.2 RMSE

The RMSE is a global measure between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ defined as

$$\text{RMSE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\|_{\text{F}}}{\sqrt{ML}}.$$

[1] FUMI's demo code is available on https://github.com/qw245/FUMI.

HSR results with smaller RMSE are better.

4.1.9.3 RSNR

The RSNR is a global measure between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ defined as

$$\mathsf{RSNR}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 10 \log_{10} \left(\frac{\|\boldsymbol{Y}\|_{\mathsf{F}}^2}{\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\|_{\mathsf{F}}^2} \right).$$

HSR results with higher RSNR are better.

4.1.9.4 SAM

We use SAM as a spectral quality measure. The SAM between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ is defined as

$$\mathsf{SAM}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{L} \sum_{j=1}^{L} \arccos \left(\frac{\langle \boldsymbol{y}_j, \hat{\boldsymbol{y}}_j \rangle}{\|\boldsymbol{y}_j\|_2 \|\hat{\boldsymbol{y}}_j\|_2} \right),$$

HSR results with smaller SAM are better.

4.1.9.5 PSNR

We use PSNR as a spatial quality measure. The PSNR is defined via the mean-squared-error of each band image. The PSNR between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ is defined as

$$\mathsf{PSNR}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{M} \sum_{i=1}^{M} 10 \log_{10} \left(\frac{\mathsf{MAX}_i^2}{\mathsf{MSE}_i} \right),$$

where

$$\mathsf{MAX}_i = \max(\boldsymbol{y}^i)$$

is the maximum pixel value in the ith band image and

$$\mathsf{MSE}_i = \frac{1}{L} \|\boldsymbol{y}^i - \hat{\boldsymbol{y}}^i\|_2^2$$

is the MSE in the ith spectral band. HSR results with higher PSNR are better.

4.2 Experiment 1: Benchmark HSR via Exact BCD and Inexact BCD

We compare the performance of HU and HSR on synthetic data using traditional exact BCD and using inexact BCD. Specifically, we only intensively compare FOGMs including GP, BBGP and PG. To take part as an exact BCD, we reuse the ALGO framework by setting the maximum number of iterations $J = 100$; to take part as an inexact BCD, we set $J = 1$ instead. The simulations are conducted under different environment settings including two SNR levels and model order N.

$N = 9$								
SNR (dB)	Method	MSE-en(dB)	MSE-ab(dB)	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)	Outer Iter.
40	Inex.BCD GP	-8.13 ± 1.16	-3.66 ± 0.86	19.34 ± 1.42	-15.32 ± 0.65	4.55 ± 0.69	29.21 ± 0.92	1233.2 ± 246.73
	Ex.BCD: GP	-7.46 ± 1.1	-3.4 ± 0.85	19.24 ± 1.72	-15.27 ± 0.89	4.68 ± 0.76	29.24 ± 1.34	298.53 ± 72.13
20	Inex.BCD GP	-6.75 ± 0.84	-2.73 ± 0.46	17.13 ± 0.87	-14.21 ± 0.58	6.69 ± 0.57	23.07 ± 0.67	165.91 ± 21.13
	Ex.BCD: GP	-5.1 ± 0.62	-2.4 ± 0.43	16.27 ± 0.69	-13.78 ± 0.6	8.17 ± 0.51	22.21 ± 0.97	52.73 ± 6.87
40	Inex.BCD BBGP	-8.24 ± 1.12	-3.62 ± 0.83	19.57 ± 1.41	-15.43 ± 0.75	4.38 ± 0.66	29.38 ± 0.95	1303.63 ± 238.66
	Ex.BCD: BBGP	-7.47 ± 1.1	-3.43 ± 0.83	19.36 ± 1.67	-15.33 ± 0.89	4.66 ± 0.75	29.32 ± 1.3	297.17 ± 70.9
20	Inex.BCD BBGP	-6.88 ± 0.79	-2.45 ± 0.43	16.98 ± 0.85	-14.13 ± 0.6	6.79 ± 0.57	22.93 ± 0.73	245.73 ± 33.62
	Ex.BCD: BBGP	-5.26 ± 0.62	-2.4 ± 0.42	16.38 ± 0.7	-13.84 ± 0.59	8.03 ± 0.5	22.33 ± 0.96	57.3 ± 6.19
40	Inex.BCD PG	-8.23 ± 1.11	-3.98 ± 0.68	21.53 ± 1.72	-16.41 ± 0.62	3.82 ± 0.67	30.69 ± 1.29	3062.73 ± 282.5
	Ex.BCD: PG	-7.75 ± 1.06	-4.01 ± 0.77	21.16 ± 1.82	-16.23 ± 0.7	3.92 ± 0.76	31 ± 1.44	324.91 ± 82.85
20	Inex.BCD PG	-6.65 ± 0.71	-2.97 ± 0.44	17.42 ± 0.75	-14.36 ± 0.54	6.56 ± 0.57	23.31 ± 0.85	626.33 ± 58.86
	Ex.BCD: PG	-5.6 ± 0.66	-2.81 ± 0.42	16.95 ± 0.77	-14.12 ± 0.58	7.25 ± 0.51	22.93 ± 0.87	70.86 ± 5.4

Table 4.1: Average HSR performance on synthetic data by using Exact BCD and Inexact BCD. FOGMs include GP, BBGP and PG. Model order $N = 9$. Pixel number $L = 100 \times 100$.

$N = 16$						
SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)	Outer Iter.
40	Inex.BCD GP	35.53 ± 0.15	-23.18 ± 0.08	0.88 ± 0.01	44.27 ± 0.13	984.38 ± 50.37
	Ex.BCD: GP	32.74 ± 0.32	-21.78 ± 0.16	1.19 ± 0.03	42.06 ± 0.22	243.32 ± 14.47
20	Inex.BCD GP	21.19 ± 0.08	-16.01 ± 0.04	4.27 ± 0.05	28.77 ± 0.09	195.02 ± 10.27
	Ex.BCD: GP	17.7 ± 0.27	-14.27 ± 0.13	6.87 ± 0.25	25.28 ± 0.26	54.99 ± 4.19
40	Inex.BCD BBGP	35.69 ± 0.09	-23.26 ± 0.05	0.86 ± 0.01	44.43 ± 0.08	966.22 ± 122.08
	Ex.BCD: BBGP	32.7 ± 0.33	-21.77 ± 0.16	1.19 ± 0.03	42.06 ± 0.21	248.25 ± 16.46
20	Inex.BCD BBGP	21.3 ± 0.07	-16.07 ± 0.03	4.2 ± 0.04	28.9 ± 0.08	207.79 ± 19.76
	Ex.BCD: BBGP	18.19 ± 0.19	-14.51 ± 0.09	6.44 ± 0.17	25.75 ± 0.18	63.25 ± 4.26
40	Inex.BCD PG	35.93 ± 0.03	-23.38 ± 0.02	0.84 ± 0	44.77 ± 0.04	2516.63 ± 98.48
	Ex.BCD: PG	35.29 ± 0.12	-23.06 ± 0.06	0.91 ± 0.01	44.08 ± 0.1	269.25 ± 20.68
20	Inex.BCD PG	22.19 ± 0.05	-16.51 ± 0.02	3.66 ± 0.02	29.88 ± 0.05	475.63 ± 25.43
	Ex.BCD: PG	20.19 ± 0.08	-15.51 ± 0.04	4.95 ± 0.05	27.72 ± 0.09	80.46 ± 4.28

Table 4.2: Average HSR performance on Cuprite dataset by using Exact BCD and Inexact BCD. FOGMs include GP, BBGP and PG. Model order $N = 16$. Pixel number $L = 200 \times 348$.

In the first part, we conduct synthetic simulation taking model order $N = 9$. The HU and HSR results are the average of 1000 trials of synthetic data generated according to the description in Section 4.1.4. In the second part we set $N = 16$ in the semi-real Cuprite dataset simulation. The HSR results are the average of 100 trials of semi-real data generated according to the description in Section 4.1.5. The results are shown in Table 4.1 and 4.2, respectively.

We highlight the better results in Table 4.1 and 4.2 where we can see the highlighted performance mostly belong to inexact methods. Table 4.2 also shows that inexact BCD methods can even outperform the exact BCD methods roughly by a 2dB improvement.

Regarding the algorithm efficiency, it can be compared by counting on the total number of gradient updates ($J \times$ Outer Iter.) of the algorithms. From the results above, we may conclude that ALGO using inexact BCD is advantageous because it reduces the computation and maintains a comparable or improved performance. In most cases,

4.3 Experiment 2: Benchmark HSR via ALGO using FOGMs

We compare the HU and HSR performance of ALGO using a range of FOGMs by two sets of simulations. The first one is a synthetic data simulation while the second one is a semi-real Cuprite dataset simulation. Our interest focuses on *inexact* BCD since they have generally better performance and higher efficiency as demonstrated in Section 4.2. Since there are too many quality measures, here we only compare the algorithms by MSE-en (for benchmarking HU), SAM and PSNR (for benchmarking HSR spectrally and spatially, respectively). The algorithm runtime is also counted to compare their efficiency.

Similar as before, we highlight the best results in Table 4.3 and 4.4. Table 4.3 shows that FISTA ususally has better HSR performance than ALGO using other FOGMs). However the HU performance is roughly the same as others (generally worse than others by no more than 0.5dB). We are more interested in the runtime performance of ALGO using FISTA because it requires significantly fewer computation time than other algorithms.

When we benchmark the algorithms using semi-real Cuprite dataset, as shown in Table 4.4, we see that ALGO using PG has the best HSR performance although it requires the most computation time.

$N = 9$					
SNR (dB)	ALGO	MSE-en(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	GP	-8.13 ± 1.16	4.55 ± 0.69	29.21 ± 0.92	15.82 ± 4.55
	BBGP	-8.24 ± 1.12	4.38 ± 0.66	29.38 ± 0.95	18.06 ± 4.82
	PG	-8.23 ± 1.11	3.82 ± 0.67	30.69 ± 1.29	26.41 ± 5.75
	FISTA	-7.99 ± 1.17	3.41 ± 0.94	32.48 ± 2.3	5.36 ± 1.3
	FW	-8.13 ± 0.99	4.9 ± 0.68	28.6 ± 0.91	14.12 ± 3.76
30	GP	-8.02 ± 1.16	5.03 ± 0.67	26.81 ± 0.73	6.21 ± 1.66
	BBGP	-8.12 ± 1.11	4.9 ± 0.63	26.93 ± 0.78	7.91 ± 2.01
	PG	-8.04 ± 1.06	4.48 ± 0.6	27.88 ± 1.04	12.94 ± 2.81
	FISTA	-7.6 ± 1.08	4.24 ± 0.79	28.84 ± 1.6	3.02 ± 0.67
	FW	-7.98 ± 1.04	5.26 ± 0.67	26.62 ± 0.78	6.25 ± 1.66
$N = 16$					
SNR (dB)	ALGO	MSE-en(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	GP	-7.69 ± 0.6	4.22 ± 0.4	28.9 ± 0.56	13.5 ± 5.85
	BBGP	-7.73 ± 0.58	4.1 ± 0.36	29.13 ± 0.58	17.11 ± 7.58
	PG	-7.93 ± 0.64	3.58 ± 0.37	30.57 ± 0.83	36.28 ± 14.9
	FISTA	-7.94 ± 0.67	3.07 ± 0.54	32.22 ± 1.55	6.46 ± 2.86
	FW	-7.43 ± 0.53	4.7 ± 0.37	27.95 ± 0.51	10.25 ± 4.93
30	GP	-7.48 ± 0.56	4.58 ± 0.38	26.79 ± 0.46	6.87 ± 3.2
	BBGP	-7.5 ± 0.54	4.51 ± 0.34	26.89 ± 0.48	8.72 ± 4.06
	PG	-7.61 ± 0.54	4.18 ± 0.31	27.81 ± 0.64	15.95 ± 7.2
	FISTA	-7.28 ± 0.56	3.93 ± 0.48	28.49 ± 1.17	4.41 ± 2.14
	FW	-7.27 ± 0.5	5.02 ± 0.36	26.05 ± 0.44	4.43 ± 2.2

Table 4.3: Average HSR performance on synthetic data by Inexact BCD using a range of FOGMs. Pixel number $L = 100 \times 100$.

Overall, PG and FISTA generally have better performance in terms of HSR quality and runtime, respectively. The competitiveness of runtime of FISTA supports our expectation on its convergence rate $\mathcal{O}(1/k^2)$ stated in Section 3.3.3, although we are now in the context of inexact BCD.

FW is also almost the second fastest algorithm. A major reason goes to its projection-free nature described in Section 3.3.4 whom we may predict that the projection operation becomes more expensive when model order N and pixel number L increase.

From the above results, we may conclude that ALGO using PG and FISTA generally perform better in terms of HU and HSR performance while ALGO using FISTA and FW is faster. From now on we focus on ALGO using the aforementioned three FOGMs.

$N = 9$				
SNR (dB)	ALGO	SAM(deg.)	PSNR(dB)	Time(sec.)
40	GP	0.99 ± 0.02	43.24 ± 0.14	39.95 ± 3.45
	BBGP	0.94 ± 0.02	43.78 ± 0.13	37.19 ± 5.91
	PG	0.88 ± 0	44.37 ± 0.03	61.84 ± 5.32
	FISTA	1.17 ± 0.03	42.13 ± 0.21	18.69 ± 1.91
	FW	1.08 ± 0.02	42.64 ± 0.11	24.56 ± 3.11
30	GP	1.76 ± 0.05	36.92 ± 0.2	16.13 ± 1.73
	BBGP	1.67 ± 0.06	37.28 ± 0.24	17.44 ± 2.26
	PG	1.56 ± 0.03	37.82 ± 0.12	34.59 ± 3.21
	FISTA	2.55 ± 0.22	34.61 ± 0.49	9.05 ± 1.02
	FW	1.87 ± 0.05	36.49 ± 0.16	12.03 ± 1.46
$N = 16$				
SNR (dB)	ALGO	SAM(deg.)	PSNR(dB)	Time(sec.)
40	GP	0.88 ± 0.01	44.27 ± 0.13	65.95 ± 6.26
	BBGP	0.86 ± 0.01	44.43 ± 0.08	68.31 ± 10.27
	PG	0.84 ± 0	44.77 ± 0.04	129.73 ± 15.25
	FISTA	1 ± 0.02	43.38 ± 0.11	26.3 ± 2.98
	FW	1.03 ± 0.02	43 ± 0.13	39.62 ± 4.06
30	GP	1.57 ± 0.02	37.73 ± 0.11	26.81 ± 2.6
	BBGP	1.56 ± 0.02	37.76 ± 0.11	27.85 ± 3.95
	PG	1.51 ± 0.01	38.03 ± 0.07	56.64 ± 5.88
	FISTA	2.04 ± 0.05	35.85 ± 0.17	17.22 ± 1.81
	FW	1.82 ± 0.03	36.63 ± 0.12	14.2 ± 1.4

Table 4.4: Average HSR performance on Cuprite dataset by Inexact BCD using a range of FOGMs. Pixel number $L = 200 \times 348$.

4.4 Experiment 3: Benchmark HSR via ALGO and FUMI

We compare the performance of ALGO and FUMI in three parts. First we use synthetic data simulation to compare their HU and HSR performance. After that we use semi-real data generated from the Cuprite dataset and the Chikusei dataset to compare their HSR performance.

4.4.1 Results on Synthetic Data

We apply FUMI and ALGO to synthetic data.

In Table 4.5 we see that ALGO generally has a comparable or better performance than FUMI does. The best HSR performance roughly goes to ALGO using PG and FISTA. In all case, the runtime required by ALGO is significantly fewer than that required by FUMI. Among

$N = 9$					
SNR (dB)	Method	MSE-en(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	FUMI	-7.37 ± 1.1	3.63 ± 0.76	31.37 ± 1.59	411.23 ± 119.49
	ALGO: PG	-8.23 ± 1.11	3.82 ± 0.67	30.69 ± 1.29	26.41 ± 5.75
	ALGO: FISTA	-7.99 ± 1.17	3.41 ± 0.94	32.48 ± 2.3	5.36 ± 1.3
	ALGO: FW	-8.13 ± 0.99	4.9 ± 0.68	28.6 ± 0.91	14.12 ± 3.76
	ALGO: Hybrid [1]	-6.94 ± 1.12	3.99 ± 0.79	31.22 ± 1.64	7.75 ± 1.54
30	FUMI	-6.99 ± 1.03	4.39 ± 0.62	28.37 ± 1.22	293.29 ± 79.24
	ALGO: PG	-8.04 ± 1.06	4.48 ± 0.6	27.88 ± 1.04	12.94 ± 2.81
	ALGO: FISTA	-7.6 ± 1.08	4.24 ± 0.79	28.84 ± 1.6	3.02 ± 0.67
	ALGO: FW	-7.98 ± 1.04	5.26 ± 0.67	26.62 ± 0.78	6.25 ± 1.66
	ALGO: Hybrid [1]	-6.92 ± 1.1	4.68 ± 0.7	28.01 ± 1.14	3.59 ± 0.95
20	FUMI	-4.74 ± 0.59	8.27 ± 0.41	22.09 ± 0.99	248.53 ± 67.97
	ALGO: PG	-6.65 ± 0.71	6.56 ± 0.57	23.31 ± 0.85	5.94 ± 1.25
	ALGO: FISTA	-5.63 ± 0.71	7.16 ± 0.52	23.12 ± 1.02	2.11 ± 0.42
	ALGO: FW	-7 ± 0.79	6.83 ± 0.59	22.99 ± 0.75	2.78 ± 0.66
	ALGO: Hybrid [1]	-5.48 ± 0.72	7.19 ± 0.52	23.02 ± 0.88	1.3 ± 0.43
$N = 16$					
SNR (dB)	Method	MSE-en(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	FUMI	-6.47 ± 0.63	4.19 ± 0.4	28.88 ± 0.71	556.08 ± 146.21
	ALGO: PG	-7.93 ± 0.64	3.58 ± 0.37	30.57 ± 0.83	36.28 ± 14.9
	ALGO: FISTA	-7.94 ± 0.67	3.07 ± 0.54	32.22 ± 1.55	6.46 ± 2.86
	ALGO: FW	-7.43 ± 0.53	4.7 ± 0.37	27.95 ± 0.51	10.25 ± 4.93
	ALGO: Hybrid [1]	-6.96 ± 0.61	4.03 ± 0.45	30.24 ± 0.89	5.7 ± 1.43
30	FUMI	-6.08 ± 0.52	4.6 ± 0.32	26.75 ± 0.56	366.32 ± 108.43
	ALGO: PG	-7.61 ± 0.54	4.18 ± 0.31	27.81 ± 0.64	15.95 ± 7.2
	ALGO: FISTA	-7.28 ± 0.56	3.93 ± 0.48	28.49 ± 1.17	4.41 ± 2.14
	ALGO: FW	-7.27 ± 0.5	5.02 ± 0.36	26.05 ± 0.44	4.43 ± 2.2
	ALGO: Hybrid [1]	-6.77 ± 0.56	4.67 ± 0.39	27.14 ± 0.72	2.82 ± 0.78
20	FUMI	-3.93 ± 0.31	9.04 ± 0.37	20.67 ± 0.63	512.21 ± 167.4
	ALGO: PG	-6.1 ± 0.34	5.97 ± 0.32	23.48 ± 0.54	9.99 ± 4.4
	ALGO: FISTA	-4.81 ± 0.33	7.87 ± 0.26	21.82 ± 0.6	3.57 ± 1.66
	ALGO: FW	-6.21 ± 0.39	7.1 ± 0.29	22.14 ± 0.43	2.71 ± 1.42
	ALGO: Hybrid [1]	-4.82 ± 0.31	8.07 ± 0.24	21.59 ± 0.58	1.62 ± 0.48

[1] Hybrid update on $\boldsymbol{S}$ by FW and on $\boldsymbol{A}$ by FISTA.

Table 4.5: HU and HSR performance on synthetic data by FUMI and ALGO. $L = 100 \times 100$. Highlighted results indicate they are the best under the setting.

the ALGO algorithms, FISTA is almost the fastest when model order is small ($N = 9$) while that of Hybrid BCD is the fastest when model order is large ($N = 16$).

4.4.2 Results on Semi-real Cuprite Dataset

We apply FUMI and ALGO to the semi-real Cuprite dataset.

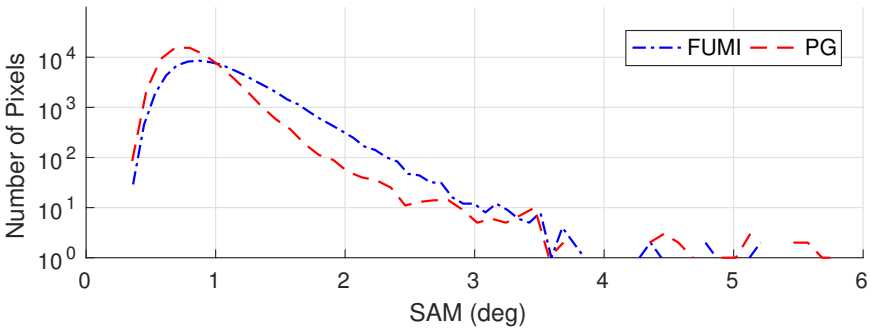
From Table 4.6, we see that ALGO's HSR performance is very close to that of FUMI. Regarding the runtime required by the algorithms, ALGO again outperforms FUMI where ALGO using FISTA is the fastest in small model order case ($N = 9$) while ALGO using FW is the fastest under large model order case ($N = 16$).

$N = 9$					
SNR (dB)	Method	RMSE(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	FUMI	-21.86 ± 0.22	1.16 ± 0.06	42.34 ± 0.29	517.45 ± 113.89
	ALGO: PG	-23.14 ± 0.02	0.88 ± 0	44.37 ± 0.03	61.84 ± 5.32
	ALGO: FISTA	-21.8 ± 0.15	1.17 ± 0.03	42.13 ± 0.21	18.69 ± 1.91
	ALGO: FW	-22.2 ± 0.08	1.08 ± 0.02	42.64 ± 0.11	24.56 ± 3.11
	ALGO: Hybrid [1]	-22.23 ± 0.08	1.07 ± 0.02	42.8 ± 0.13	44.15 ± 14.83
30	FUMI	-18.75 ± 0.18	2.28 ± 0.09	35.24 ± 0.23	478.57 ± 132.86
	ALGO: PG	-20.43 ± 0.06	1.56 ± 0.03	37.82 ± 0.12	34.59 ± 3.21
	ALGO: FISTA	-18.37 ± 0.39	2.55 ± 0.22	34.61 ± 0.49	9.05 ± 1.02
	ALGO: FW	-19.67 ± 0.11	1.87 ± 0.05	36.49 ± 0.16	12.03 ± 1.46
	ALGO: Hybrid [1]	-19.26 ± 0.15	2.08 ± 0.07	35.84 ± 0.22	21.02 ± 6.78
20	FUMI	-14.85 ± 0.23	5.65 ± 0.31	26.65 ± 0.36	565.72 ± 172.4
	ALGO: PG	-16.53 ± 0.04	3.65 ± 0.04	29.89 ± 0.09	13.75 ± 1.63
	ALGO: FISTA	-15.5 ± 0.2	4.84 ± 0.22	27.83 ± 0.34	5.57 ± 0.47
	ALGO: FW	-16.2 ± 0.09	3.97 ± 0.1	29.18 ± 0.17	6.22 ± 0.58
	ALGO: Hybrid [1]	-15.94 ± 0.12	4.31 ± 0.14	28.67 ± 0.25	8.77 ± 2.87
$N = 16$					
SNR (dB)	Method	RMSE(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	FUMI	-22.4 ± 0.13	1.03 ± 0.03	43.03 ± 0.19	1269.38 ± 252.42
	ALGO: PG	-23.38 ± 0.02	0.84 ± 0	44.77 ± 0.04	129.73 ± 15.25
	ALGO: FISTA	-22.63 ± 0.07	1 ± 0.02	43.38 ± 0.11	26.3 ± 2.98
	ALGO: FW	-22.45 ± 0.08	1.03 ± 0.02	43 ± 0.13	39.62 ± 4.06
	ALGO: Hybrid [1]	-22.46 ± 0.1	1.02 ± 0.02	43.15 ± 0.16	61.38 ± 19.56
30	FUMI	-19.48 ± 0.24	1.97 ± 0.09	36.08 ± 0.33	1150.41 ± 280.43
	ALGO: PG	-20.57 ± 0.04	1.51 ± 0.01	38.03 ± 0.07	56.64 ± 5.88
	ALGO: FISTA	-19.39 ± 0.12	2.04 ± 0.05	35.85 ± 0.17	17.22 ± 1.81
	ALGO: FW	-19.86 ± 0.07	1.82 ± 0.03	36.63 ± 0.12	14.2 ± 1.4
	ALGO: Hybrid [1]	-19.61 ± 0.14	1.93 ± 0.06	36.23 ± 0.22	28.8 ± 8.49
20	FUMI	-14.42 ± 0.1	6.6 ± 0.17	25.64 ± 0.2	1575.23 ± 331.09
	ALGO: PG	-16.51 ± 0.02	3.66 ± 0.02	29.88 ± 0.05	25.42 ± 2.75
	ALGO: FISTA	-15.38 ± 0.13	5.15 ± 0.18	27.46 ± 0.29	11.68 ± 1.06
	ALGO: FW	-15.84 ± 0.04	4.52 ± 0.06	28.39 ± 0.1	8.02 ± 0.71
	ALGO: Hybrid [1]	-15.45 ± 0.05	5.07 ± 0.07	27.6 ± 0.11	13.09 ± 3.13

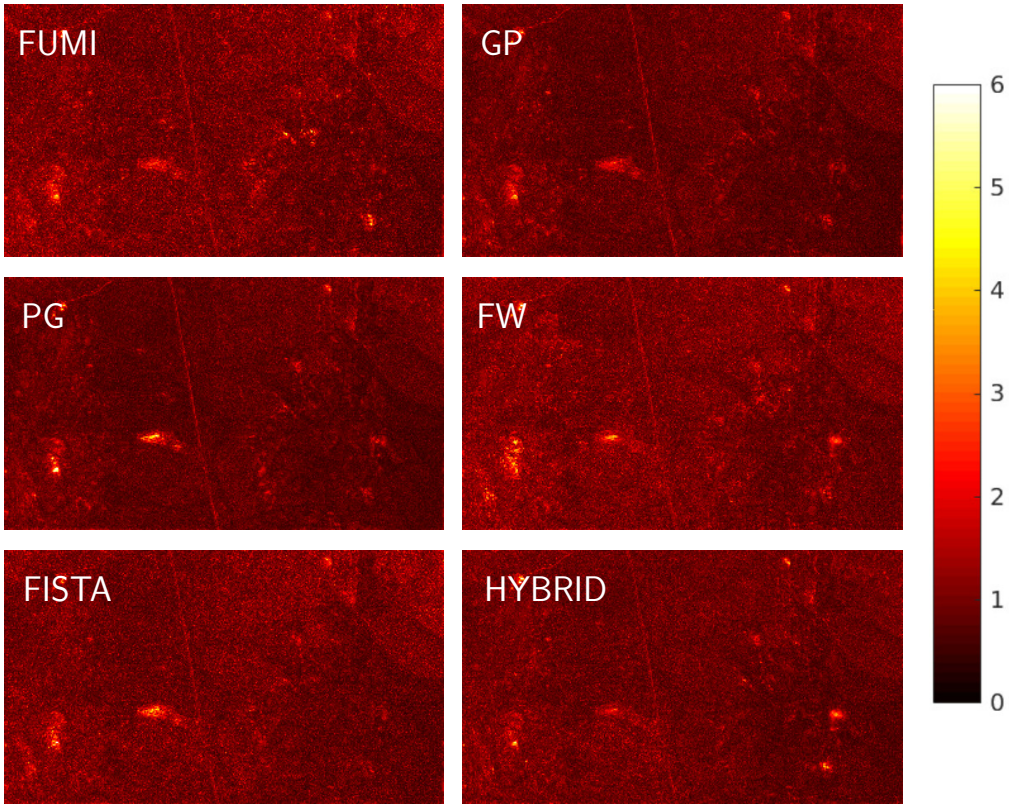
[1] Hybrid update on $\boldsymbol{S}$ by FW and on $\boldsymbol{A}$ by FISTA.

Table 4.6: Average HSR performance on Cuprite dataset by FUMI and ALGO $L = 200 \times 348$. Highlighted results indicate they are the best under the setting.

In Figure 4.3 and 4.4, we compare the HSR performance using SAM and RMSE, respectively. The visualization qualitatively shows there does not have significant difference in the quality of SR image recovery.

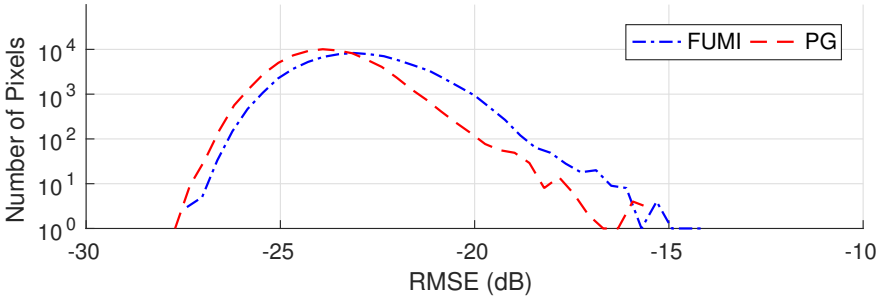


(a) SAM histogram

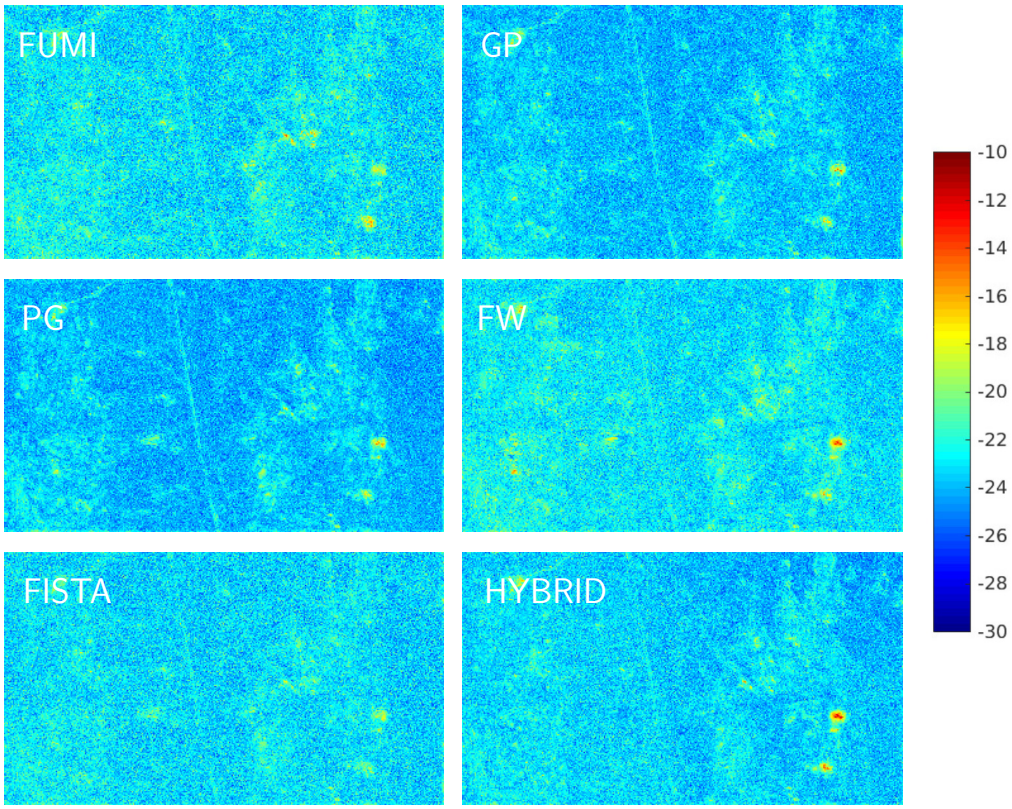


(b) SAM maps

Figure 4.3: The SAM of a trial of semi-real Cuprite dataset simulation. Model order $N = 16$; pixel number $L = 200 \times 348$; SNR $= 40$dB.



(a) RMSE histogram



(b) RMSE maps

Figure 4.4: The RMSE of a trial of semi-real Cuprite dataset simulation. Model order $N = 16$; pixel number $L = 200 \times 348$; SNR $= 40$dB.

4.4.3 Results on Semi-real Chikusei Dataset

We apply FUMI and ALGO to the semi-real Chikusei dataset. Since the image at hand is relatively large, we set $\delta = 10^{-2}$ to terminate the algorithms earlier [2].

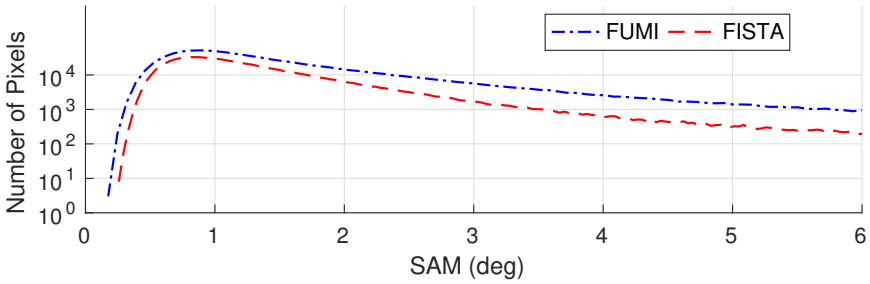
$N = 9$					
SNR (dB)	Method	RMSE(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)[3]
40	FUMI	-22.58 ± 0.03	2.07 ± 0.01	39.61 ± 0.08	2744.03 ± 440.45
	ALGO: PG	-22.01 ± 0.03	3.42 ± 0	36.97 ± 0.06	33.62 ± 8.69
	ALGO: FISTA	-25.12 ± 0	1.72 ± 0	44.5 ± 0.03	87.23 ± 21.39
	ALGO: FW	-21.72 ± 0.18	3.35 ± 0.14	36.8 ± 0.53	40.22 ± 11.01
	ALGO: Hybrid [1]	-23.08 ± 0.05	2.75 ± 0.04	39.58 ± 0.11	32.5 ± 8.64
30	FUMI	-21.88 ± 2.21	2.35 ± 0.25	37.71 ± 3.81	4025.73 ± 948.96
	ALGO: PG	-21.49 ± 2.17	3.56 ± 0.36	35.88 ± 3.62	31.31 ± 8.67
	ALGO: FISTA	-23.44 ± 2.37	2.26 ± 0.23	39.71 ± 4.01	52.68 ± 14.96
	ALGO: FW	-21.06 ± 2.14	3.44 ± 0.37	35.41 ± 3.68	34.64 ± 10.92
	ALGO: Hybrid [1]	-22.43 ± 2.27	2.85 ± 0.29	37.92 ± 3.83	26.93 ± 9.58
20	FUMI[2]	–	–	–	–
	ALGO: PG	-20.25 ± 0.07	4.45 ± 0.13	33.12 ± 0.15	20.83 ± 5.95
	ALGO: FISTA	-20.55 ± 0.07	4.09 ± 0.13	33.56 ± 0.14	13.21 ± 3.64
	ALGO: FW	-19.65 ± 0.2	4.9 ± 0.28	32.07 ± 0.4	21.82 ± 6.14
	ALGO: Hybrid [1]	-20.53 ± 0.13	3.92 ± 0.09	33.66 ± 0.31	14.26 ± 4.06
$N = 16$					
SNR (dB)	Method	RMSE(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	FUMI[2]	–	–	–	–
	ALGO: PG	-22.16 ± 0.06	3.36 ± 0.05	37.33 ± 0.16	56.64 ± 18.45
	ALGO: FISTA	-25.45 ± 0.03	1.57 ± 0.02	45.18 ± 0.13	134.76 ± 39.89
	ALGO: FW	-22.27 ± 0.1	3.05 ± 0.06	38.08 ± 0.24	73.03 ± 25.05
	ALGO: Hybrid [1]	-23.92 ± 0.07	2.32 ± 0.03	41.38 ± 0.19	67.83 ± 23.9
30	FUMI[2]	–	–	–	–
	ALGO: PG	-21.87 ± 0.08	3.52 ± 0.06	36.62 ± 0.19	52.28 ± 17.2
	ALGO: FISTA	-23.82 ± 0.09	2.19 ± 0.04	40.43 ± 0.15	74.44 ± 23.97
	ALGO: FW	-21.71 ± 0.09	3.36 ± 0.06	36.47 ± 0.21	57.25 ± 19.24
	ALGO: Hybrid [1]	-23.01 ± 0.07	2.72 ± 0.04	38.87 ± 0.1	45.14 ± 15.01
20	FUMI[2]	–	–	–	–
	ALGO: PG	-20.36 ± 0.06	4.23 ± 0.12	33.38 ± 0.14	33.36 ± 11.33
	ALGO: FISTA	-20.62 ± 0.08	3.87 ± 0.13	33.79 ± 0.18	23.09 ± 7.91
	ALGO: FW	-19.61 ± 0.12	4.88 ± 0.13	32.08 ± 0.24	29.92 ± 10.06
	ALGO: Hybrid [1]	-20.67 ± 0.11	3.87 ± 0.07	33.94 ± 0.22	20.27 ± 6.84

[1] Hybrid update on $\boldsymbol{S}$ by FW and on $\boldsymbol{A}$ by FISTA.
[2] FUMI is unable to run on Chikusei under the data settings.
[3] The runtime of FUMI is the average total sum of computation time required by all 100 subimages.

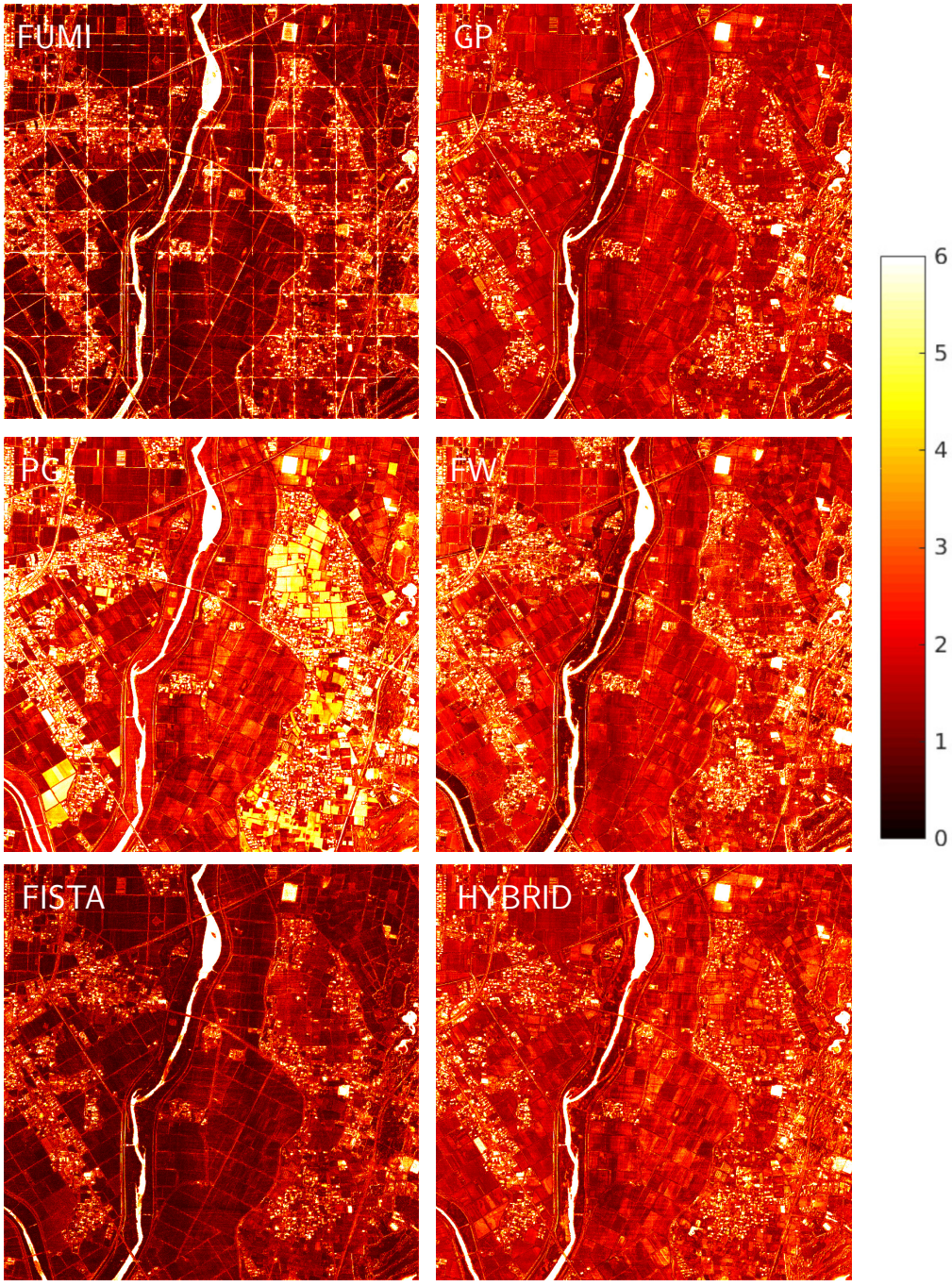
Table 4.7: Average HSR performance on Chikusei dataset by FUMI and ALGO $L = 1000 \times 1000$. Highlighted results indicate they are the best under the setting.

We see from Table 4.7 that ALGO is more capable in processing large images such that FUMI cannot handle the Chikusei image under when $N = 16$ and under low noise level when $N = 9$. Regarding the runtime required by the algorithms, ALGO using Hybrid BCD is almost the fastest. We qualitatively compare their performance using SAM and RMSE in Figure 4.5 and 4.6, respectively, to show that ALGO using FISTA is very comparable to FUMI regardless of FUMI's subtle issues.

[2] FUMI often encounter numerical error when we feed it with big HS and MS image pair In order to enable FUMI to perform HSR successfully, we split the HS and MS image pair into 100 smaller pieces and feed them to FUMI one by one. Nevertheless FUMI is still unapplicable when model order N is large or when the noise level is high. We therefore avoid the comparison whenever it is unavailable.

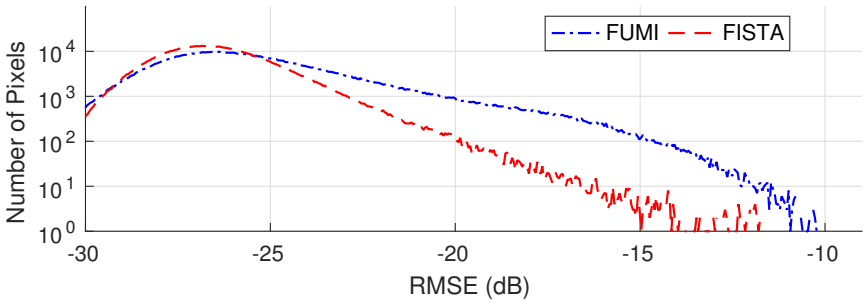


(a) SAM histogram

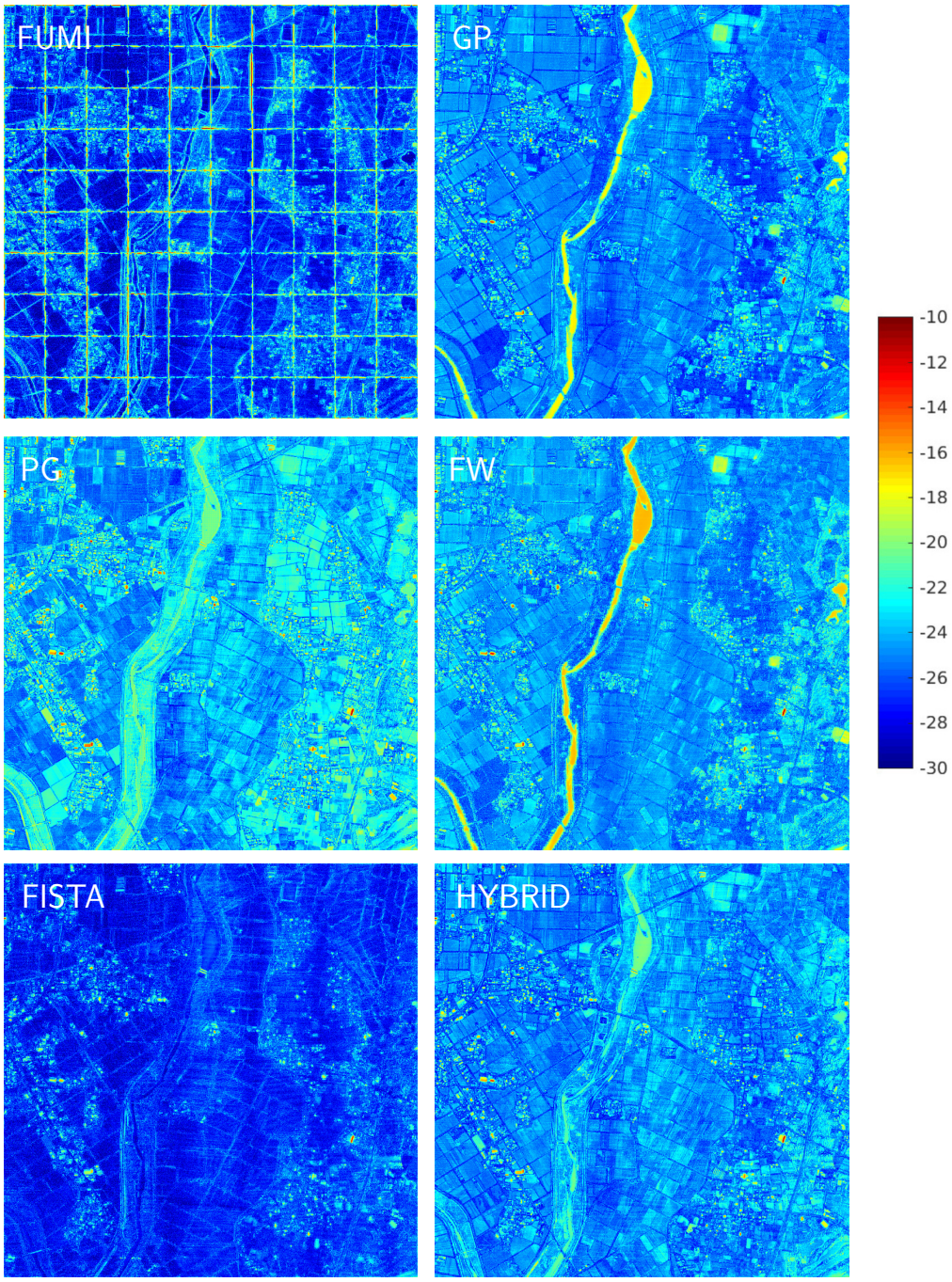


(b) SAM maps

Figure 4.5: The SAM of a trial of semi-real Chikusei dataset simulation. Model order $N = 16$; pixel number $L = 1000 \times 1000$; SNR $= 40$dB.



(a) RMSE histogram



(b) RMSE maps

Figure 4.6: The RMSE of a trial of semi-real Chikusei dataset simulation. Model order $N = 16$; pixel number $L = 1000 \times 1000$; SNR $= 40$dB.

4.5 Experiment 4: Benchmark HSR via ALGO and CNMF

4.5.1 Results on Semi-real Chikusei Dataset

We apply CNMF and ALGO to the semi-real Chikusei dataset. In Yokoya's HSR problem formulation, the feasible set of both $\boldsymbol{S}$ and $\boldsymbol{A}$ are open ($\mathcal{S} = \mathbb{R}_+^{N\times L}$ and $\mathcal{A} = \mathbb{R}_+^{M\times N}$) so that FW is not directly applicable. Although as discussed in Section 3.3.4 that a heuristic approach can be used to *simulate* the nonnegative orthant by a nonnegative box of sufficiently large size, the projection onto nonnegative orthant is in fact inexpensive so that using FW (and therefore Hybrid BCD) would not have further advantages under the current HSR problem setting.

For CNMF is popularly known to be a very fast algorithm, we compare it with FISTA. Again, since the image at hand is relatively large, we set $\delta = 10^{-2}$ to terminate the algorithms earlier.

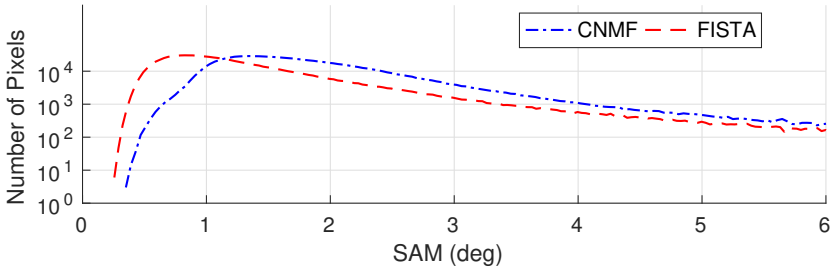
$N = 16$					
SNR (dB)	Method	RMSE(dB)	SAM(deg.)	PSNR(dB)	Time(sec.)
40	CNMF[1]	$28.650.02$	2.18 ± 0.01	44.7 ± 0.04	39.27 ± 6.53
	ALGO: FISTA	$32.180.06$	1.61 ± 0.02	45.32 ± 0.11	75.03 ± 9.76
30	CNMF[1]	$26.590.05$	2.74 ± 0.01	39.73 ± 0.04	37.6 ± 7.36
	ALGO: FISTA	$28.351.03$	2.37 ± 0.31	40.16 ± 1	41.35 ± 12.92
20	CNMF[1]	$21.290.1$	4.58 ± 0.05	32.89 ± 0.07	33.4 ± 8.58
	ALGO: FISTA	$22.440.04$	3.73 ± 0.04	33.98 ± 0.06	11.41 ± 2.03

[1] CNMF subproblem iteration number is set to 100.

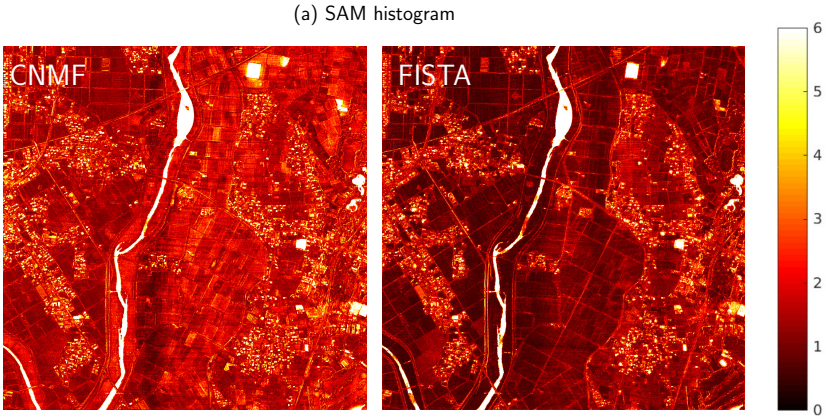
Table 4.8: Average HSR performance on Chikusei dataset by CNMF and ALGO $L = 1000 \times 1000$. Highlighted results indicate they are the best under the setting.

We see from Table 4.8 that the HSR quality by ALGO is similar to that by CNMF in terms of SAM and PSNR and has obvious improvement in terms of RMSE. Although ALGO is generally slower than CNMF, we remark that the CNMF algorithm is not directly solving problem ($*$) because it does not jointly optimize the desired variables with the consideration of the HS and MS recovery altogether, as shown in the update scheme (2.15) and (2.16).

In Figure 4.7 and 4.8, we show the SAM and RMSE map of the recovered Chikusei image. The performance of ALGO using FISTA is shown to be better than that of CNMF.

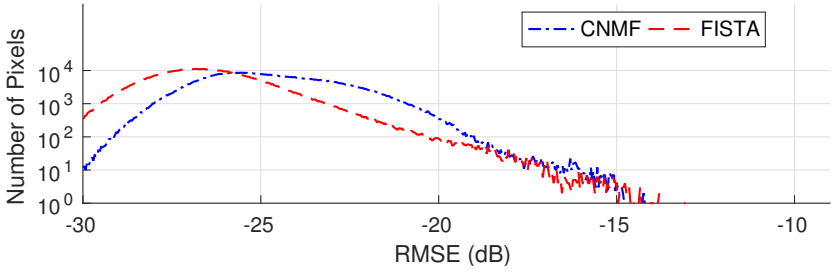


(a) SAM histogram

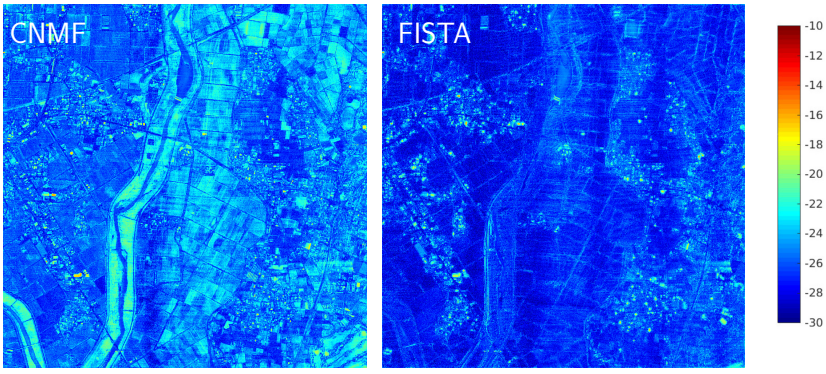


(b) SAM maps

Figure 4.7: The SAM of a trial of semi-real Chikusei dataset simulation. Model order $N = 16$; pixel number $L = 1000 \times 1000$.



(a) RMSE histogram



(b) RMSE maps

Figure 4.8: The RMSE of a trial of semi-real Chikusei dataset simulation. Model order $N = 16$; pixel number $L = 1000 \times 1000$.

Chapter 5

Conclusion and Discussion

With the increasing demand for faster algorithms in the big data era, massive data processing technique gains more attraction in the field of computation, signal processing and data science. This thesis reviewed a general framework of alternating optimization (AO) on solving the hyperspectral super-resolution (HSR) problem, which is a kind of low-rank matrix factorization (MF) problem and is practically attractive in remote sensing. Specifically we explored the use of Block Coordinate Descent (BCD) to tackle the nonconvex HSR problem in two aspects. The first aspect is the use of Exact BCD and Inexact BCD (whom we proposed as ALternating Gradient-based Optimization "ALGO") to solve the HSR problem. Results showed that HSR via ALGO is more competitive than HSR via Exact BCD in terms of both recovery accuracy and algorithm efficiency, which encouraged us to focus on Inexact BCD in the remaining of the thesis. The second aspect is the use of various inexact gradient update rules in solving each subblock problems in the BCD as a practical and intuitive way to reduce complexity. Results showed that proximal type gradient methods are generally better. Proximal Gradient Method (PG) and Fast Proximal Gradient Method (FISTA) of this type were shown to be more competitive in terms of recovery accuracy. Frank Wolfe Method (FW) is also competitive in terms of algorithm efficiency when the image size and model order are large.

We also compared ALGO with two state-of-the-art HSR algorithms, namely FUMI and CNMF. ALGO whom we implemented as a 1-step gradient update in solving each subblock problem generally possesses comparable recovery performance as FUMI and CNMF

have. Regarding the computation time, ALGO also demonstrated to be significantly faster than FUMI even when the data size is very large.

As a future direction, ALGO has great potential in other remote sensing and machine learning applications as long as we can formulate their problems in a block-separable manner. From a practical point of view, ALGO can be applied to existing MF type problems whose first order information is easily computable. The advantage of ALGO in terms of computation efficiency can be revealed when the data size at hand grows to a very large scale and when higher order methods become impractical.

Appendix A

MATLAB Code of ALGO for Solving Wei et al.'s HSR Problem

A.1 ALGO using GP

```
% algorithm initialization %
S       = S_init ; S_curr  = S       ;
A       = A_init ; A_curr  = A       ;
objCurr = 0.5 * ( sum(sum((Y_H-A*S*G).^2)) + sum(sum((Y_M-F*A*S).^2)) ) ;
% the algorithm %
obj_it    = zeros(maxIteraNum+1,1) ; time_it   = zeros(maxIteraNum+1,1) ;
obj_it(1) = objCurr                ;
tic ;
SG = S_curr * G ;
FA = F * A_curr ;
for k = 1 : maxIteraNum
    % S-subproblem %
    for sstep = 1 : SStep
        Scurr_G        = SG                                    ;
        AScurrGminY_H  = A*Scurr_G - Y_H                       ;
        FAScurrminY_M  = FA*S_curr - Y_M                       ;
        GRADS          = ( A' * AScurrGminY_H ) * G'          ...
                           + FA' * FAScurrminY_M               ;
        ZS             = SimplexProj( (S_curr - GRADS)' )'     ;
        ZSminScurr     = ZS - S_curr                           ;
        A_ZSminScurr_G = A * ( ZSminScurr * G )                ;
        FA_ZSminScurr  = FA * ZSminScurr                       ;
        tau            = ( - sum(sum( AScurrGminY_H .* A_ZSminScurr_G )) ...
                           - sum(sum( FAScurrminY_M .* FA_ZSminScurr  )) ) ...
                         / (   sum(sum( A_ZSminScurr_G.^2 ))  ...
                             + sum(sum( FA_ZSminScurr.^2 )) )   ;
        tau            = min( max( tau , 0 ) , 1 )             ;
```

```
        S_curr          = ( 1-tau ) * S_curr + tau * ZS          ;
        SG              = S_curr * G                             ;
    end
    S = S_curr ;
    % A-subproblem %
    for astep = 1 : AStep
        F_Acurr         = FA                                     ;
        AcurrSGminY_H   = A_curr*SG - Y_H                        ;
        FAcurrSminY_M   = F_Acurr * S - Y_M                      ;
        GRADA           = AcurrSGminY_H * SG'                  ...
                          + F' * ( FAcurrSminY_M * S' )          ;
        ZA              = min( max( A_curr - GRADA , 0 ) , 1 )  ;
        ZAminAcurr      = ZA - A_curr                            ;
        ZAminAcurr_SG   = ZAminAcurr * SG                        ;
        F_ZAminAcurr_S  = ( F * ZAminAcurr ) * S                 ;
        tau             = ( - sum(sum( AcurrSGminY_H .* ZAminAcurr_SG  )) ...
                            - sum(sum( FAcurrSminY_M .* F_ZAminAcurr_S )) ) ...
                          / (   sum(sum( ZAminAcurr_SG.^2 ))  ...
                              + sum(sum( F_ZAminAcurr_S.^2 )) ) ;
        tau             = min( max( tau , 0 ) , 1 )              ;
        A_curr          = ( 1-tau ) * A_curr + tau * ZA          ;
        FA              = F * A_curr                             ;
    end
    A = A_curr ;
    % check convergence %
    objPrev       = objCurr                                      ;
    objCurr       = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))        ...
                          + sum(sum((Y_M-FA*S_curr).^2)) )       ;
    obj_chng      = abs( objPrev - objCurr ) / objCurr           ;
    time_it(k+1)  = toc                                          ;
    obj_it(k+1)   = objCurr                                      ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )               ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                    ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

A.2 ALGO using BBGP

```
% algorithm initialization %
S        = S_init ; S_curr  = S       ;
A        = A_init ; A_curr  = A       ;
objCurr = 0.5 * ( sum(sum((Y_H-A*(S*G)).^2)) + sum(sum((Y_M-(F*A)*S).^2)) ) ;
% the algorithm %
obj_it     = zeros(maxIteraNum+1,1) ; time_it    = zeros(maxIteraNum+1,1) ;
obj_it(1) = objCurr                 ;
alphaS     = 1 ; % max( eig( gamma*(A'*A) + (F*A)'*(F*A) ) ) ;
alphaA     = 1 ; % max( eig( (S*G)*(S*G)' + theta * (S*S') ) ) ;
tic ;
SG = S_curr * G ;
FA = F * A_curr ;
for k = 1 : maxIteraNum
    % S-subproblem %
    for sstep = 1 : SStep
        ScurrG         = SG                                                ;
        AScurrGminY_H = A * ScurrG - Y_H                                   ;
        FAScurrminY_M = FA * S_curr - Y_M                                  ;
        GRADS          = ( A' * AScurrGminY_H ) * G'                      ...
                         + FA' * FAScurrminY_M                             ;
        DELTS          = SimplexProj( (S_curr - alphaS*GRADS)' )' - S_curr ;
        ADELTSG        = A * ( DELTS * G )                                 ;
        FADELTS        = FA * DELTS                                        ;
        tau            = ( - sum(sum( AScurrGminY_H .* (ADELTSG) ))       ...
                           - sum(sum( FAScurrminY_M .* FADELTS )) )       ...
                         / ( sum(sum((ADELTSG).^2)) + sum(sum(FADELTS.^2)) ) ;
        tau            = min( max( tau , 0 ) , 1 )                         ;
        S_curr         = S_curr + tau * DELTS                              ;
        gamm           = sum(sum( (ADELTSG).^2 )) + sum(sum( FADELTS.^2 )) ;
        if( gamm > 0 ) % take beta_max as 1 , not to use 1/LS
            alphaS = min( max( sum(sum(DELTS.^2)) / gamm , 0 ) , 1 )       ;
        else
            alphaS = 1                                                     ;
        end
        SG = S_curr * G ;
    end
    S = S_curr ;
    % A-subproblem %
    for astep = 1 : AStep
        FAcurr         = FA                                                ;
        AcurrSGminY_H = A_curr * SG - Y_H                                  ;
        FAcurrSminY_M = FAcurr * S - Y_M                                   ;
        GRADA          = AcurrSGminY_H * SG'                              ...
                         + F' * ( FAcurrSminY_M * S' )                     ;
        DELTA          = min( max( A_curr - alphaA*GRADA , 0 ) , 1 ) - A_curr ;
        FDELTAS        = ( F * DELTA ) * S                                 ;
        DELTASG        = DELTA * SG                                        ;
        tau            = ( - sum(sum( AcurrSGminY_H .* DELTASG ))         ...
                           - sum(sum( FAcurrSminY_M .* FDELTAS )) )       ...
```

```
                        / ( sum(sum(DELTASG.^2)) + sum(sum(FDELTAS.^2)) )    ;
        tau            = min( max( tau , 0 ) , 1 )                           ;
        A_curr         = A_curr + tau * DELTA                                ;
        gamm           = sum(sum( DELTASG.^2 )) + sum(sum( (FDELTAS).^2 ))   ;
        if( gamm > 0 ) % take beta_max as 1 , not to use 1/LA
            alphaA = min( max( sum(sum(DELTA.^2)) / gamm , 0 ) , 1 )         ;
        else
            alphaA = 1                                                       ;
        end
        FA = F * A_curr ;
    end
    A = A_curr ;
    % check convergence %
    objPrev      = objCurr                                                   ;
    objCurr      = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))                  ...
                         + sum(sum((Y_M-FA*S_curr).^2)) )                    ;
    time_it(k+1) = toc                                                       ;
    obj_it(k+1)  = objCurr                                                   ;
    obj_chng     = abs( objPrev - objCurr ) / objPrev                        ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )                           ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                                ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

A.3 ALGO using PG

```
% algorithm initialization %
S        = S_init ; S_curr  = S_init ;
A        = A_init ; A_curr  = A_init ;
objCurr = 0.5 * ( sum(sum((Y_H-A*S*G).^2)) + sum(sum((Y_M-F*A*S).^2)) ) ;
% the algorithm %
obj_it     = zeros(maxIteraNum+1,1) ; time_it    = zeros(maxIteraNum+1,1) ;
obj_it(1) = objCurr                 ;
tic ;
SG = S_curr * G ;
FA = F * A_curr ;
for k = 1 : maxIteraNum
    % S-subproblem %
    AtA = A'*A                              ;
    LS = max( eig( gamma*AtA + FA'*FA ) ) ;
    for sstep = 1 : SStep
        GRADS  = ( (AtA/LS) * SG - (A'/LS) * Y_H ) * G' ...
                 + (FA'/LS) * ( FA * S_curr - Y_M )       ;
        S_curr = SimplexProj( (S_curr - GRADS)' )'        ;
        SG     = S_curr * G                               ;
    end
    S = S_curr ;
    % A-subproblem %
    LA = max( eig( SG*SG' + theta * (S*S') ) ) ;
    for astep = 1 : AStep
        GRADA  = ( A_curr * SG - Y_H ) * (SG'/LA)        ...
                 + F' * ( ( ( FA * S - Y_M ) * S' ) / LA ) ;
        A_curr = min( max( A_curr-GRADA , 0 ) , 1 )        ;
        FA     = F * A_curr                                ;
    end
    A = A_curr ;
    % check convergence %
    objPrev      = objCurr                                          ;
    objCurr      = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))          ...
                         + sum(sum((Y_M-FA*S_curr).^2)) )          ;
    time_it(k+1) = toc                                              ;
    obj_it(k+1)  = objCurr                                          ;
    obj_chng     = abs( objPrev - objCurr ) / objCurr               ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )                  ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                       ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

A.4 ALGO using FISTA

```
% algorithm initialization %
S        = S_init ; S_curr  = S_init ; S_prev  = S_curr ;
A        = A_init ; A_curr  = A_init ; A_prev  = A_curr ;
tk       = 0      ; % Nesterov parameter on S-leastsquares
uk       = 0      ; % Nesterov parameter on A-leastsquares
%restart_tk = ( SStep > 100 ) ;
%restart_uk = ( AStep > 100 ) ;
objCurr = 0.5 * ( sum(sum((Y_H-A*S*G).^2)) + sum(sum((Y_M-F*A*S).^2)) ) ;
% the algorithm %
obj_it    = zeros(maxIteraNum+1,1) ; time_it   = zeros(maxIteraNum+1,1) ;
obj_it(1) = objCurr                ;
tic ;
FA = F * A ;
for k = 1 : maxIteraNum
    % S-subproblem %
    AtA = A'*A                              ;
    LS  = max( eig( gamma*AtA + FA'*FA ) ) ;
    %if( restart_tk ) ; tk  = 0 ; end ;
    for sstep = 1 : SStep
        tk_next = 0.5 + sqrt( 0.25 + tk^2 )                          ;
        ZS      = S_curr + ( (tk-1) / tk_next ) * ( S_curr - S_prev ) ;
        GRADS   = ( (AtA/LS) * ( ZS * G ) - (A'/LS) * Y_H ) * G'    ...
                  + (FA'/LS) * ( FA * ZS - Y_M )                     ;
        tk      = tk_next                                            ;
        S_prev  = S_curr                                             ;
        S_curr  = SimplexProj( (ZS - GRADS)' )'                      ;
    end
    S = S_curr ;
    % A-subproblem %
    SG = S*G                                    ;
    LA = max( eig( SG*SG' + theta * (S*S') ) ) ;
    %if( restart_uk ) ; uk  = 0 ; end ;
    for astep = 1 : AStep
        uk_next = 0.5 + sqrt( 0.25 + uk^2 )                          ;
        ZA      = A_curr + ( (uk-1) / uk_next ) * ( A_curr - A_prev ) ;
        GRADA   = ( ( ZA * SG - Y_H ) * SG'                         ...
                  + F' * ( ( (F * ZA) * S - Y_M ) * S' ) ) / LA      ;
        uk      = uk_next                                            ;
        A_prev  = A_curr                                             ;
        A_curr  = min( max( ZA - GRADA , 0 ) , 1 )                   ;
    end
    A = A_curr ;
    % check convergence %
    FA           = F * A                                        ;
    objPrev      = objCurr                                      ;
    objCurr      = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))        ...
                       + sum(sum((Y_M-FA*S_curr).^2)) )         ;
    time_it(k+1) = toc                                          ;
    obj_it(k+1)  = objCurr                                      ;
```

```
    obj_chng      = abs( objPrev - objCurr ) / objCurr              ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )                  ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                       ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

A.5 ALGO using FW

```
% algorithm initialization %
S           = S_init ; S_curr      = S       ;
A           = A_init ; A_curr      = A       ;
objCurr     = 0.5 * ( sum(sum((Y_H-A*S*G).^2)) + sum(sum((Y_M-F*A*S).^2)) ) ;
modelOrder  = size( A , 2 ) ;
zeros_N_L   = false( modelOrder , MS_pixelNum )        ; % boolean false matrix
FW_bases    = 0:modelOrder:modelOrder*(MS_pixelNum-1) ;
% the algorithm %
obj_it     = zeros(maxIteraNum+1,1) ;
time_it    = zeros(maxIteraNum+1,1) ;
obj_it(1)  = objCurr                ;
tic ;
SG = S_curr * G ;
FA = F * A_curr ;
for k = 1 : maxIteraNum
    % S-subproblem %
    for sstep = 1 : SStep
        Scurr_G         = SG                                        ;
        AScurrGminY_H   = A*Scurr_G - Y_H                           ;
        FAScurrminY_M   = FA*S_curr - Y_M                           ;
        GRADS           = ( A' * AScurrGminY_H ) * G'             ...
                          + FA' * ( FA * S_curr - Y_M )             ;
        [~,minIdx]      = min( GRADS )                              ;
        ZS              = zeros_N_L                                 ;
        ZS( minIdx + FW_bases ) = 1                                 ;
        ZSminScurr      = ZS - S_curr                               ;
        A_ZSminScurr_G  = A * ( ZSminScurr * G )                    ;
        FA_ZSminScurr   = FA * ZSminScurr                           ;
        tau             = ( - sum(sum( AScurrGminY_H .* A_ZSminScurr_G )) ...
                            - sum(sum( FAScurrminY_M .* FA_ZSminScurr  )) ) ...
                          / (   sum(sum( A_ZSminScurr_G.^2 ))    ...
                              + sum(sum( FA_ZSminScurr.^2 )) )    ;
        tau             = min( max( tau , 0 ) , 1 )                 ;
        S_curr          = S_curr + tau * ZSminScurr                 ;
        SG              = S_curr * G                                ;
    end
    S = S_curr ;
    % A-subproblem %
    for astep = 1 : AStep
        F_Acurr         = FA                                      ;
        AcurrSGminY_H   = A_curr*SG - Y_H                         ;
        FAcurrSminY_M   = F_Acurr*S  - Y_M                        ;
        GRADA           = ( AcurrSGminY_H ) * SG'               ...
                          + F' * ( FAcurrSminY_M * S' )           ;
        ZA              = ( GRADA < 0 )                           ;
        ZAminAcurr      = ZA - A_curr                             ;
        ZAminAcurr_SG   = ZAminAcurr * SG                         ;
        F_ZAminAcurr_S  = ( F * ZAminAcurr ) * S                  ;
        tau             = ( - sum(sum( AcurrSGminY_H .* ZAminAcurr_SG  )) ...
```

```
                            - sum(sum( FAcurrSminY_M .* F_ZAminAcurr_S )) ) ...
                          / (   sum(sum( ZAminAcurr_SG.^2 ))  ...
                              + sum(sum( F_ZAminAcurr_S.^2 )) ) ;
        tau             = min( max( tau , 0 ) , 1 )             ;
        A_curr          = A_curr + tau * ZAminAcurr             ;
        FA              = F * A_curr                            ;
    end
    A = A_curr ;
    % check convergence %
    objPrev       = objCurr                                       ;
    objCurr       = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))         ...
                          + sum(sum((Y_M-FA*S_curr).^2)) )        ;
    time_it(k+1) = toc                                            ;
    obj_it(k+1)  = objCurr                                        ;
    obj_chng      = abs( objPrev - objCurr ) / objCurr            ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )                ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                     ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

A.6 ALGO using Hybrid BCD

```
% algorithm initialization %
S           = S_init ; S_curr      = S       ;
if( strcmp( S_HYBRID , 'FISTA' ) )
S_prev      = S_curr ;
tk       = 0       ; % Nesterov parameter on S-leastsquares
end
A           = A_init ; A_curr      = A       ;
if( strcmp( A_HYBRID , 'FISTA' ) )
A_prev      = A_curr ;
uk       = 0       ; % Nesterov parameter on A-leastsquares
end
objCurr     = 0.5 * ( sum(sum((Y_H-A*S*G).^2)) + sum(sum((Y_M-F*A*S).^2)) ) ;
modelOrder = size( A , 2 ) ;
zeros_N_L   = false( modelOrder , MS_pixelNum )        ; % boolean false matrix
FW_bases    = 0:modelOrder:modelOrder*(MS_pixelNum-1) ;
% the algorithm %
obj_it     = zeros(maxIteraNum+1,1) ; time_it    = zeros(maxIteraNum+1,1) ;
obj_it(1) = objCurr                 ;
tic ;
SG = S_curr * G ;
FA = F * A_curr ;
for k = 1 : maxIteraNum
    % S-subproblem %
    if( S_HYBRID_RANDI_STEP )
        SStep = randi(S_HYBRID_RANDI_BASE) ;
    else
        SStep = 1 ;
    end
    switch S_HYBRID
    case 'FW'
    for sstep = 1 : SStep
        Scurr_G         = SG                                    ;
        AScurrGminY_H   = A*Scurr_G - Y_H                       ;
        FAScurrminY_M   = FA*S_curr - Y_M                       ;
        GRADS           = ( A' * AScurrGminY_H ) * G'          ...
                          + FA' * ( FA * S_curr - Y_M )         ;
        [~,minIdx]      = min( GRADS )                          ;
        ZS              = zeros_N_L                             ;
        ZS( minIdx + FW_bases ) = 1                             ;
        ZSminScurr      = ZS - S_curr                           ;
        A_ZSminScurr_G = A * ( ZSminScurr * G )                 ;
        FA_ZSminScurr   = FA * ZSminScurr                       ;
        tau             = ( - sum(sum( AScurrGminY_H .* A_ZSminScurr_G )) ...
                            - sum(sum( FAScurrminY_M .* FA_ZSminScurr  )) ) ...
                          / (   sum(sum( A_ZSminScurr_G.^2 ))    ...
                              + sum(sum( FA_ZSminScurr.^2 )) )    ;
        tau             = min( max( tau , 0 ) , 1 )             ;
        S_curr          = S_curr + tau * ZSminScurr             ;
        SG              = S_curr * G                            ;
```

```
    end
    % end case 'FRANKWOLFE'
    end
    S = S_curr ;
    % A-subproblem %
    if( A_HYBRID_RANDI_STEP )
        AStep = randi(A_HYBRID_RANDI_BASE) ;
    else
        AStep = 1 ;
    end
    switch A_HYBRID
    case 'PG'
    LA = max( eig( SG*SG' + theta * (S*S') ) ) ;
    for astep = 1 : AStep
        GRADA  = ( A_curr * SG - Y_H ) * (SG'/LA)        ...
                 + F' * ( ( ( FA * S - Y_M ) * S' ) / LA ) ;
        A_curr = min( max( A_curr-GRADA , 0 ) , 1 )        ;
        FA     = F * A_curr                                ;
    end
    % end case 'PROXGRAD'
    case 'FISTA'
    LA = max( eig( SG*SG' + theta * (S*S') ) ) ;
    %if( restart_uk ) ; uk  = 0 ; end ;
    for astep = 1 : AStep
        uk_next = 0.5 + sqrt( 0.25 + uk^2 )                          ;
        ZA      = A_curr + ( (uk-1) / uk_next ) * ( A_curr - A_prev ) ;
        GRADA   = ( ( ZA * SG - Y_H ) * SG'                         ...
                  + F' * ( ( (F * ZA) * S - Y_M ) * S' ) ) / LA       ;
        uk      = uk_next                                            ;
        A_prev  = A_curr                                             ;
        A_curr  = min( max( ZA - GRADA , 0 ) , 1 )                   ;
    end
    FA = F * A_curr ;
    % end case 'FISTA'
    end
    A = A_curr ;
    % check convergence %
    objPrev      = objCurr                                       ;
    objCurr      = 0.5 * ( sum(sum((Y_H-A_curr*SG).^2))          ...
                         + sum(sum((Y_M-FA*S_curr).^2)) )        ;
    time_it(k+1) = toc                                           ;
    obj_it(k+1)  = objCurr                                       ;
    obj_chng     = abs( objPrev - objCurr ) / objCurr            ;
    if( obj_chng < convthresh )
        fprintf( 'objective value converges !\n' )               ;
        fprintf( 'early break @ it: %d / %d\n' , k , maxIteraNum ) ;
        break                                                    ;
    end
end
TIME_MFbA = toc ; IT_used_MF = k ;
HSI_MF = reshape( S'*A' , MS_imgSizeM , MS_imgSizeN , HS_bandNum ) ;
```

Appendix B

Full Synthetic Simulations Results

In this section we present the full set of numerical results on synthetic simulations. The synthetic simulation settings are detailed in Section 4.1. In the numerical results we take two more quantitative measure than we took in Chapter 4. The additional performance measure are normalized mean-squared-error (NMSE) of the recovered endmember and abundance matrix, which measure their unmixing error, and reconstruction SNR (RSNR) of the recovered SR image, which measures the super-resolution error.

The NMSE of estimated endmember matrix $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{S}}$ are defined as

$$\text{NMSE}(\boldsymbol{A}, \hat{\boldsymbol{A}}) = 10 \log_{10} \left(\frac{\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathsf{F}}^2}{\|\boldsymbol{A}\|_{\mathsf{F}}^2} \right) \tag{B.1}$$

$$\text{NMSE}(\boldsymbol{S}, \hat{\boldsymbol{S}}) = 10 \log_{10} \left(\frac{\|\boldsymbol{S} - \hat{\boldsymbol{S}}\|_{\mathsf{F}}^2}{\|\boldsymbol{S}\|_{\mathsf{F}}^2} \right), \tag{B.2}$$

respectively.

The RSNR of recovered SR image $\hat{\boldsymbol{Y}}$ is defined as

$$\text{RSNR}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 10 \log_{10} \left(\frac{\|\boldsymbol{Y}\|_{\mathsf{F}}^2}{\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\|_{\mathsf{F}}^2} \right). \tag{B.3}$$

B.1 Performance of ALGO Against the Exactness in solving Subproblems

We group the results of ALGO into individual tables according to the FOGM that ALGO uses. The objective here is to see how the exactness of solving the subproblems affects ALGO in terms of unmixing and super-resolution quality in the HSR problem.

In the tables, J refers to the inner iteration number when ALGO solves the subproblems. In the context of Hybrid BCD, the notation "$J \sim \mathcal{U}(a, b)$" reads "J is an integer between a and b following a discrete uniform distribution". The results by ALGO-GP, ALGO-BBGP, ALGO-PG, ALGO-FISTA, ALGO-FW, ALGO-Hybrid(S: FW A: PG) and ALGO-Hybrid(S: FW A: PG) are shown from Table B.1 to B.7 where model order $N = 9$. When $N = 16$, the results are shown from Table B.8 to B.14.

Basically the performance of non-Hybrid type ALGO algorithms are similar to each other under inexact BCD settings, i.e., when $J \in \{1, 5, 10\}$. Comparing with exact BCD setting where we take $J = 100$, the HU and HSR are shown to have worse quality. ALGO-FISTA's performance in Table B.4 and B.11 (where N is 9 and 16, respectively) can even show a significant difference between the use of exact and inexact BCD.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP ($J = 1$)	-6.4 ± 1.76	-8.13 ± 1.16	-3.81 ± 0.85	-3.66 ± 0.86
	ALGO-GP ($J = 5$)	-6.54 ± 1.83	-8.11 ± 1.19	-3.84 ± 0.94	-3.72 ± 0.95
	ALGO-GP ($J = 10$)	-6.37 ± 1.8	-8.13 ± 1.17	-3.85 ± 0.9	-3.75 ± 0.91
	ALGO-GP ($J = 100$)	-5.85 ± 1.57	-7.46 ± 1.1	-2.75 ± 0.9	-3.4 ± 0.85
30	ALGO-GP ($J = 1$)	-6.27 ± 1.59	-8.12 ± 1.11	-3.25 ± 0.65	-3.2 ± 0.64
	ALGO-GP ($J = 5$)	-6.31 ± 1.63	-8.09 ± 1.17	-3.33 ± 0.68	-3.27 ± 0.66
	ALGO-GP ($J = 10$)	-6.23 ± 1.65	-7.92 ± 1.18	-3.25 ± 0.68	-3.23 ± 0.66
	ALGO-GP ($J = 100$)	-5.67 ± 1.38	-7.2 ± 1.02	-2.44 ± 0.76	-3.09 ± 0.67
20	ALGO-GP ($J = 1$)	-5.65 ± 1.3	-6.75 ± 0.84	-2.95 ± 0.48	-2.73 ± 0.46
	ALGO-GP ($J = 5$)	-5.61 ± 1.32	-6.54 ± 0.83	-2.86 ± 0.48	-2.69 ± 0.46
	ALGO-GP ($J = 10$)	-5.17 ± 1.28	-6.24 ± 0.75	-2.81 ± 0.47	-2.69 ± 0.44
	ALGO-GP ($J = 100$)	-3.96 ± 1.02	-5.1 ± 0.62	-1.98 ± 0.55	-2.4 ± 0.43
10	ALGO-GP ($J = 1$)	-2.23 ± 0.79	-3.36 ± 0.42	-1.28 ± 0.36	-1.43 ± 0.29
	ALGO-GP ($J = 5$)	-1.65 ± 0.71	-2.81 ± 0.34	-1.28 ± 0.31	-1.45 ± 0.27
	ALGO-GP ($J = 10$)	-1.07 ± 0.68	-2.38 ± 0.33	-1.22 ± 0.31	-1.39 ± 0.26
	ALGO-GP ($J = 100$)	-0.13 ± 0.63	-1.98 ± 0.31	-1 ± 0.27	-1.3 ± 0.24

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP ($J = 1$)	19.34 ± 1.42	-15.32 ± 0.65	4.55 ± 0.69	29.21 ± 0.92
	ALGO-GP ($J = 5$)	19.36 ± 1.43	-15.33 ± 0.67	4.54 ± 0.7	29.37 ± 0.98
	ALGO-GP ($J = 10$)	19.49 ± 1.43	-15.39 ± 0.64	4.49 ± 0.71	29.47 ± 1.05
	ALGO-GP ($J = 100$)	19.24 ± 1.72	-15.27 ± 0.89	4.68 ± 0.76	29.24 ± 1.34
30	ALGO-GP ($J = 1$)	19.03 ± 1.31	-15.17 ± 0.71	4.9 ± 0.63	26.93 ± 0.78
	ALGO-GP ($J = 5$)	18.99 ± 1.37	-15.15 ± 0.69	4.92 ± 0.69	27.02 ± 0.77
	ALGO-GP ($J = 10$)	18.85 ± 1.36	-15.08 ± 0.66	5 ± 0.71	26.93 ± 0.78
	ALGO-GP ($J = 100$)	18.77 ± 1.46	-15.04 ± 0.76	5.32 ± 0.71	26.73 ± 0.98
20	ALGO-GP ($J = 1$)	17.13 ± 0.87	-14.21 ± 0.58	6.69 ± 0.57	23.07 ± 0.67
	ALGO-GP ($J = 5$)	17.03 ± 0.88	-14.16 ± 0.58	6.82 ± 0.56	23.03 ± 0.7
	ALGO-GP ($J = 10$)	16.8 ± 0.83	-14.05 ± 0.58	7.12 ± 0.52	22.83 ± 0.72
	ALGO-GP ($J = 100$)	16.27 ± 0.69	-13.78 ± 0.6	8.17 ± 0.51	22.21 ± 0.97
10	ALGO-GP ($J = 1$)	11.36 ± 0.33	-11.34 ± 0.7	14.11 ± 0.54	16.96 ± 0.79
	ALGO-GP ($J = 5$)	10.69 ± 0.26	-11 ± 0.74	15.44 ± 0.49	16.27 ± 0.87
	ALGO-GP ($J = 10$)	10.11 ± 0.25	-10.72 ± 0.74	16.61 ± 0.49	15.68 ± 0.91
	ALGO-GP ($J = 100$)	9.17 ± 0.29	-10.24 ± 0.66	18.6 ± 0.6	14.72 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP ($J = 1$)	15.82 ± 4.55	1233.2 ± 246.73
	ALGO-GP ($J = 5$)	35.67 ± 11.22	673.81 ± 149.42
	ALGO-GP ($J = 10$)	50.87 ± 16.77	493.34 ± 128.56
	ALGO-GP ($J = 100$)	322.16 ± 109.76	298.53 ± 72.13
30	ALGO-GP ($J = 1$)	7.91 ± 2.01	558.76 ± 91.89
	ALGO-GP ($J = 5$)	14.73 ± 3.74	261.58 ± 34.84
	ALGO-GP ($J = 10$)	22.58 ± 5.67	202.9 ± 26.3
	ALGO-GP ($J = 100$)	133.94 ± 39.43	111.88 ± 16.49
20	ALGO-GP ($J = 1$)	2.32 ± 0.54	165.91 ± 21.13
	ALGO-GP ($J = 5$)	5.42 ± 1.29	103.28 ± 9.86
	ALGO-GP ($J = 10$)	9.32 ± 2.3	90 ± 10.86
	ALGO-GP ($J = 100$)	58.3 ± 17.71	52.73 ± 6.87
10	ALGO-GP ($J = 1$)	1.76 ± 0.4	124.84 ± 11.45
	ALGO-GP ($J = 5$)	3.26 ± 0.81	62.09 ± 6.09
	ALGO-GP ($J = 10$)	5.21 ± 1.47	50.62 ± 8.73
	ALGO-GP ($J = 100$)	49.51 ± 15.78	43.49 ± 7.73

Table B.1: Average HU and HSR performance of ALGO-GP on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-BBGP ($J = 1$)	-6.4 ± 1.67	-8.24 ± 1.12	-3.65 ± 0.83	-3.62 ± 0.83
	ALGO-BBGP ($J = 5$)	-6.46 ± 1.69	-8.2 ± 1.13	-3.77 ± 0.83	-3.72 ± 0.83
	ALGO-BBGP ($J = 10$)	-6.43 ± 1.7	-8.06 ± 1.15	-3.75 ± 0.86	-3.74 ± 0.88
	ALGO-BBGP ($J = 100$)	-5.92 ± 1.56	-7.47 ± 1.1	-2.75 ± 0.9	-3.43 ± 0.83
30	ALGO-BBGP ($J = 1$)	-6.27 ± 1.59	-8.12 ± 1.11	-3.25 ± 0.65	-3.2 ± 0.64
	ALGO-BBGP ($J = 5$)	-6.31 ± 1.63	-8.09 ± 1.17	-3.33 ± 0.68	-3.27 ± 0.66
	ALGO-BBGP ($J = 10$)	-6.23 ± 1.65	-7.92 ± 1.18	-3.25 ± 0.68	-3.23 ± 0.66
	ALGO-BBGP ($J = 100$)	-5.67 ± 1.38	-7.2 ± 1.02	-2.44 ± 0.76	-3.09 ± 0.67
20	ALGO-BBGP ($J = 1$)	-5.64 ± 1.2	-6.88 ± 0.79	-2.48 ± 0.47	-2.45 ± 0.43
	ALGO-BBGP ($J = 5$)	-5.47 ± 1.24	-6.6 ± 0.78	-2.6 ± 0.47	-2.55 ± 0.43
	ALGO-BBGP ($J = 10$)	-5.3 ± 1.23	-6.35 ± 0.79	-2.52 ± 0.47	-2.5 ± 0.42
	ALGO-BBGP ($J = 100$)	-4.18 ± 1.02	-5.26 ± 0.62	-1.99 ± 0.54	-2.4 ± 0.42
10	ALGO-BBGP ($J = 1$)	-2.27 ± 0.71	-3.52 ± 0.37	-1 ± 0.31	-1.31 ± 0.25
	ALGO-BBGP ($J = 5$)	-1.45 ± 0.67	-2.84 ± 0.31	-1.09 ± 0.3	-1.37 ± 0.25
	ALGO-BBGP ($J = 10$)	-1.22 ± 0.66	-2.58 ± 0.31	-1.11 ± 0.29	-1.36 ± 0.25
	ALGO-BBGP ($J = 100$)	-0.2 ± 0.63	-2 ± 0.31	-1 ± 0.26	-1.3 ± 0.24

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-BBGP ($J = 1$)	19.57 ± 1.41	-15.43 ± 0.75	4.38 ± 0.66	29.38 ± 0.95
	ALGO-BBGP ($J = 5$)	19.48 ± 1.44	-15.39 ± 0.75	4.43 ± 0.69	29.47 ± 0.97
	ALGO-BBGP ($J = 10$)	19.42 ± 1.41	-15.36 ± 0.72	4.46 ± 0.69	29.43 ± 1
	ALGO-BBGP ($J = 100$)	19.36 ± 1.67	-15.33 ± 0.89	4.66 ± 0.75	29.32 ± 1.3
30	ALGO-BBGP ($J = 1$)	19.03 ± 1.31	-15.17 ± 0.71	4.9 ± 0.63	26.93 ± 0.78
	ALGO-BBGP ($J = 5$)	18.99 ± 1.37	-15.15 ± 0.69	4.92 ± 0.69	27.02 ± 0.77
	ALGO-BBGP ($J = 10$)	18.85 ± 1.36	-15.08 ± 0.66	5 ± 0.71	26.93 ± 0.78
	ALGO-BBGP ($J = 100$)	18.77 ± 1.46	-15.04 ± 0.76	5.32 ± 0.71	26.73 ± 0.98
20	ALGO-BBGP ($J = 1$)	16.98 ± 0.85	-14.13 ± 0.6	6.79 ± 0.57	22.93 ± 0.73
	ALGO-BBGP ($J = 5$)	17 ± 0.89	-14.15 ± 0.6	6.87 ± 0.57	23.01 ± 0.75
	ALGO-BBGP ($J = 10$)	16.78 ± 0.86	-14.04 ± 0.6	7.08 ± 0.56	22.84 ± 0.76
	ALGO-BBGP ($J = 100$)	16.38 ± 0.7	-13.84 ± 0.59	8.03 ± 0.5	22.33 ± 0.96
10	ALGO-BBGP ($J = 1$)	11.29 ± 0.33	-11.31 ± 0.68	14.22 ± 0.54	16.89 ± 0.78
	ALGO-BBGP ($J = 5$)	10.55 ± 0.26	-10.93 ± 0.73	15.7 ± 0.48	16.12 ± 0.88
	ALGO-BBGP ($J = 10$)	10.25 ± 0.24	-10.79 ± 0.74	16.32 ± 0.47	15.82 ± 0.89
	ALGO-BBGP ($J = 100$)	9.23 ± 0.28	-10.27 ± 0.67	18.47 ± 0.57	14.78 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-BBGP ($J = 1$)	18.06 ± 4.82	1303.63 ± 238.66
	ALGO-BBGP ($J = 5$)	40 ± 11.75	697.8 ± 139.47
	ALGO-BBGP ($J = 10$)	61.17 ± 19.16	541.8 ± 123.03
	ALGO-BBGP ($J = 100$)	352.19 ± 119.94	297.17 ± 70.9
30	ALGO-BBGP ($J = 1$)	7.91 ± 2.01	558.76 ± 91.89
	ALGO-BBGP ($J = 5$)	14.73 ± 3.74	261.58 ± 34.84
	ALGO-BBGP ($J = 10$)	22.58 ± 5.67	202.9 ± 26.3
	ALGO-BBGP ($J = 100$)	133.94 ± 39.43	111.88 ± 16.49
20	ALGO-BBGP ($J = 1$)	3.66 ± 0.84	245.73 ± 33.62
	ALGO-BBGP ($J = 5$)	6.52 ± 1.59	115.68 ± 13.43
	ALGO-BBGP ($J = 10$)	11.71 ± 2.69	104.36 ± 8.82
	ALGO-BBGP ($J = 100$)	69.44 ± 19.71	57.3 ± 6.19
10	ALGO-BBGP ($J = 1$)	2.04 ± 0.49	136.03 ± 17.53
	ALGO-BBGP ($J = 5$)	4.3 ± 1.03	75.21 ± 8.62
	ALGO-BBGP ($J = 10$)	7.81 ± 1.89	69.14 ± 8.14
	ALGO-BBGP ($J = 100$)	54.66 ± 16.97	44.92 ± 7.51

Table B.2: Average HU and HSR performance of ALGO-BBGP on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-PG ($J = 1$)	-6.4 ± 1.67	-8.24 ± 1.12	-3.65 ± 0.83	-3.62 ± 0.83
	ALGO-PG ($J = 5$)	-6.46 ± 1.69	-8.2 ± 1.13	-3.77 ± 0.83	-3.72 ± 0.83
	ALGO-PG ($J = 10$)	-6.43 ± 1.7	-8.06 ± 1.15	-3.75 ± 0.86	-3.74 ± 0.88
	ALGO-PG ($J = 100$)	-5.92 ± 1.56	-7.47 ± 1.1	-2.75 ± 0.9	-3.43 ± 0.83
30	ALGO-PG ($J = 1$)	-6.27 ± 1.59	-8.12 ± 1.11	-3.25 ± 0.65	-3.2 ± 0.64
	ALGO-PG ($J = 5$)	-6.31 ± 1.63	-8.09 ± 1.17	-3.33 ± 0.68	-3.27 ± 0.66
	ALGO-PG ($J = 10$)	-6.23 ± 1.65	-7.92 ± 1.18	-3.25 ± 0.68	-3.23 ± 0.66
	ALGO-PG ($J = 100$)	-5.67 ± 1.38	-7.2 ± 1.02	-2.44 ± 0.76	-3.09 ± 0.67
20	ALGO-PG ($J = 1$)	-5.64 ± 1.2	-6.88 ± 0.79	-2.48 ± 0.47	-2.45 ± 0.43
	ALGO-PG ($J = 5$)	-5.47 ± 1.24	-6.6 ± 0.78	-2.6 ± 0.47	-2.55 ± 0.43
	ALGO-PG ($J = 10$)	-5.3 ± 1.23	-6.35 ± 0.79	-2.52 ± 0.47	-2.5 ± 0.42
	ALGO-PG ($J = 100$)	-4.18 ± 1.02	-5.26 ± 0.62	-1.99 ± 0.54	-2.4 ± 0.42
10	ALGO-PG ($J = 1$)	-2.27 ± 0.71	-3.52 ± 0.37	-1 ± 0.31	-1.31 ± 0.25
	ALGO-PG ($J = 5$)	-1.45 ± 0.67	-2.84 ± 0.31	-1.09 ± 0.3	-1.37 ± 0.25
	ALGO-PG ($J = 10$)	-1.22 ± 0.66	-2.58 ± 0.31	-1.11 ± 0.29	-1.36 ± 0.25
	ALGO-PG ($J = 100$)	-0.2 ± 0.63	-2 ± 0.31	-1 ± 0.26	-1.3 ± 0.24

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-PG ($J = 1$)	19.57 ± 1.41	-15.43 ± 0.75	4.38 ± 0.66	29.38 ± 0.95
	ALGO-PG ($J = 5$)	19.48 ± 1.44	-15.39 ± 0.75	4.43 ± 0.69	29.47 ± 0.97
	ALGO-PG ($J = 10$)	19.42 ± 1.41	-15.36 ± 0.72	4.46 ± 0.69	29.43 ± 1
	ALGO-PG ($J = 100$)	19.36 ± 1.67	-15.33 ± 0.89	4.66 ± 0.75	29.32 ± 1.3
30	ALGO-PG ($J = 1$)	19.03 ± 1.31	-15.17 ± 0.71	4.9 ± 0.63	26.93 ± 0.78
	ALGO-PG ($J = 5$)	18.99 ± 1.37	-15.15 ± 0.69	4.92 ± 0.69	27.02 ± 0.77
	ALGO-PG ($J = 10$)	18.85 ± 1.36	-15.08 ± 0.66	5 ± 0.71	26.93 ± 0.78
	ALGO-PG ($J = 100$)	18.77 ± 1.46	-15.04 ± 0.76	5.32 ± 0.71	26.73 ± 0.98
20	ALGO-PG ($J = 1$)	16.98 ± 0.85	-14.13 ± 0.6	6.79 ± 0.57	22.93 ± 0.73
	ALGO-PG ($J = 5$)	17 ± 0.89	-14.15 ± 0.6	6.87 ± 0.57	23.01 ± 0.75
	ALGO-PG ($J = 10$)	16.78 ± 0.86	-14.04 ± 0.6	7.08 ± 0.56	22.84 ± 0.76
	ALGO-PG ($J = 100$)	16.38 ± 0.7	-13.84 ± 0.59	8.03 ± 0.5	22.33 ± 0.96
10	ALGO-PG ($J = 1$)	11.29 ± 0.33	-11.31 ± 0.68	14.22 ± 0.54	16.89 ± 0.78
	ALGO-PG ($J = 5$)	10.55 ± 0.26	-10.93 ± 0.73	15.7 ± 0.48	16.12 ± 0.88
	ALGO-PG ($J = 10$)	10.25 ± 0.24	-10.79 ± 0.74	16.32 ± 0.47	15.82 ± 0.89
	ALGO-PG ($J = 100$)	9.23 ± 0.28	-10.27 ± 0.67	18.47 ± 0.57	14.78 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-PG ($J = 1$)	18.06 ± 4.82	1303.63 ± 238.66
	ALGO-PG ($J = 5$)	40 ± 11.75	697.8 ± 139.47
	ALGO-PG ($J = 10$)	61.17 ± 19.16	541.8 ± 123.03
	ALGO-PG ($J = 100$)	352.19 ± 119.94	297.17 ± 70.9
30	ALGO-PG ($J = 1$)	7.91 ± 2.01	558.76 ± 91.89
	ALGO-PG ($J = 5$)	14.73 ± 3.74	261.58 ± 34.84
	ALGO-PG ($J = 10$)	22.58 ± 5.67	202.9 ± 26.3
	ALGO-PG ($J = 100$)	133.94 ± 39.43	111.88 ± 16.49
20	ALGO-PG ($J = 1$)	3.66 ± 0.84	245.73 ± 33.62
	ALGO-PG ($J = 5$)	6.52 ± 1.59	115.68 ± 13.43
	ALGO-PG ($J = 10$)	11.71 ± 2.69	104.36 ± 8.82
	ALGO-PG ($J = 100$)	69.44 ± 19.71	57.3 ± 6.19
10	ALGO-PG ($J = 1$)	2.04 ± 0.49	136.03 ± 17.53
	ALGO-PG ($J = 5$)	4.3 ± 1.03	75.21 ± 8.62
	ALGO-PG ($J = 10$)	7.81 ± 1.89	69.14 ± 8.14
	ALGO-PG ($J = 100$)	54.66 ± 16.97	44.92 ± 7.51

Table B.3: Average HU and HSR performance of ALGO-PG on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-FISTA ($J = 1$)	-6.89 ± 1.99	-7.99 ± 1.17	-4.27 ± 0.82	-4.08 ± 0.84
	ALGO-FISTA ($J = 5$)	-6.94 ± 1.99	-7.95 ± 1.19	-4.35 ± 0.85	-4.17 ± 0.87
	ALGO-FISTA ($J = 10$)	-6.8 ± 1.95	-7.96 ± 1.21	-4.29 ± 0.87	-4.16 ± 0.89
	ALGO-FISTA ($J = 100$)	-5.72 ± 1.75	-7.4 ± 1.18	-2.76 ± 0.85	-3.42 ± 0.84
30	ALGO-FISTA ($J = 1$)	-6.42 ± 1.71	-7.6 ± 1.08	-3.77 ± 0.62	-3.55 ± 0.61
	ALGO-FISTA ($J = 5$)	-6.21 ± 1.71	-7.19 ± 1.02	-3.86 ± 0.63	-3.65 ± 0.62
	ALGO-FISTA ($J = 10$)	-6.06 ± 1.66	-7.05 ± 0.99	-3.82 ± 0.66	-3.65 ± 0.65
	ALGO-FISTA ($J = 100$)	-4.88 ± 1.51	-6.6 ± 0.97	-2.42 ± 0.76	-3.07 ± 0.67
20	ALGO-FISTA ($J = 1$)	-5 ± 1.24	-5.63 ± 0.71	-2.86 ± 0.44	-2.7 ± 0.42
	ALGO-FISTA ($J = 5$)	-4.05 ± 1.12	-4.76 ± 0.6	-2.82 ± 0.44	-2.69 ± 0.42
	ALGO-FISTA ($J = 10$)	-3.76 ± 1.11	-4.56 ± 0.58	-2.77 ± 0.46	-2.66 ± 0.43
	ALGO-FISTA ($J = 100$)	-2.95 ± 1.05	-4.68 ± 0.58	-1.74 ± 0.58	-2.34 ± 0.44
10	ALGO-FISTA ($J = 1$)	-1 ± 0.71	-2.49 ± 0.31	-1.03 ± 0.3	-1.35 ± 0.25
	ALGO-FISTA ($J = 5$)	-0.09 ± 0.73	-2.09 ± 0.31	-1.05 ± 0.3	-1.36 ± 0.25
	ALGO-FISTA ($J = 10$)	-0.05 ± 0.71	-1.96 ± 0.32	-1.11 ± 0.28	-1.36 ± 0.25
	ALGO-FISTA ($J = 100$)	0.01 ± 0.64	-1.92 ± 0.32	-0.93 ± 0.26	-1.26 ± 0.24

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-FISTA ($J = 1$)	23.24 ± 2.37	-17.27 ± 1.03	3.41 ± 0.94	32.48 ± 2.3
	ALGO-FISTA ($J = 5$)	22.91 ± 2.1	-17.1 ± 0.89	3.52 ± 0.78	32.31 ± 1.73
	ALGO-FISTA ($J = 10$)	22.58 ± 2.11	-16.94 ± 0.95	3.64 ± 0.81	32.04 ± 1.78
	ALGO-FISTA ($J = 100$)	20.4 ± 2.44	-15.85 ± 1.34	4.61 ± 1.12	30.37 ± 2.03
30	ALGO-FISTA ($J = 1$)	21.57 ± 1.74	-16.44 ± 0.79	4.24 ± 0.79	28.84 ± 1.6
	ALGO-FISTA ($J = 5$)	21.21 ± 1.53	-16.26 ± 0.67	4.44 ± 0.7	28.52 ± 1.23
	ALGO-FISTA ($J = 10$)	20.85 ± 1.53	-16.08 ± 0.7	4.62 ± 0.72	28.21 ± 1.2
	ALGO-FISTA ($J = 100$)	19.05 ± 1.83	-15.18 ± 1.07	5.58 ± 0.95	26.85 ± 1.41
20	ALGO-FISTA ($J = 1$)	17.26 ± 0.76	-14.28 ± 0.61	7.16 ± 0.52	23.12 ± 1.02
	ALGO-FISTA ($J = 5$)	16.39 ± 0.61	-13.84 ± 0.63	8.09 ± 0.49	22.31 ± 0.98
	ALGO-FISTA ($J = 10$)	16.05 ± 0.63	-13.67 ± 0.65	8.43 ± 0.53	22 ± 0.96
	ALGO-FISTA ($J = 100$)	15.51 ± 0.8	-13.4 ± 0.69	8.99 ± 0.68	21.47 ± 1.05
10	ALGO-FISTA ($J = 1$)	10.06 ± 0.3	-10.69 ± 0.72	16.69 ± 0.6	15.63 ± 0.9
	ALGO-FISTA ($J = 5$)	9.53 ± 0.26	-10.42 ± 0.7	17.79 ± 0.51	15.07 ± 0.91
	ALGO-FISTA ($J = 10$)	9.37 ± 0.27	-10.34 ± 0.68	18.14 ± 0.55	14.91 ± 0.89
	ALGO-FISTA ($J = 100$)	8.99 ± 0.33	-10.16 ± 0.64	18.98 ± 0.7	14.55 ± 0.87

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-FISTA ($J = 1$)	5.36 ± 1.3	550.78 ± 85.34
	ALGO-FISTA ($J = 5$)	6.88 ± 2.04	198.9 ± 38.23
	ALGO-FISTA ($J = 10$)	9.24 ± 3.07	138.41 ± 32.93
	ALGO-FISTA ($J = 100$)	128.12 ± 40.99	191.93 ± 41.84
30	ALGO-FISTA ($J = 1$)	3.02 ± 0.67	299.52 ± 26.05
	ALGO-FISTA ($J = 5$)	3.56 ± 0.88	103.62 ± 11.65
	ALGO-FISTA ($J = 10$)	4.58 ± 1.19	69.3 ± 9.48
	ALGO-FISTA ($J = 100$)	53.97 ± 15.75	79.78 ± 13.29
20	ALGO-FISTA ($J = 1$)	2.11 ± 0.42	195.63 ± 10.07
	ALGO-FISTA ($J = 5$)	2.46 ± 0.57	71.22 ± 5.47
	ALGO-FISTA ($J = 10$)	3.23 ± 0.79	48.67 ± 5.11
	ALGO-FISTA ($J = 100$)	29.77 ± 8.7	43.6 ± 5.87
10	ALGO-FISTA ($J = 1$)	1.63 ± 0.35	148.76 ± 12.72
	ALGO-FISTA ($J = 5$)	2.03 ± 0.52	58.84 ± 7.45
	ALGO-FISTA ($J = 10$)	2.73 ± 0.74	41.55 ± 6.18
	ALGO-FISTA ($J = 100$)	28.08 ± 8.71	40.89 ± 7.36

Table B.4: Average HU and HSR performance of ALGO-FISTA on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-FW ($J = 1$)	-5.82 ± 1.49	-8.13 ± 0.99	-1.94 ± 1	-3.32 ± 0.78
	ALGO-FW ($J = 5$)	-5.58 ± 1.81	-8.11 ± 1.1	-2.5 ± 1.06	-3.5 ± 0.94
	ALGO-FW ($J = 10$)	-5.62 ± 1.89	-8.08 ± 1.14	-2.66 ± 1.12	-3.57 ± 0.99
	ALGO-FW ($J = 100$)	-5.79 ± 1.97	-8.2 ± 1.23	-2.68 ± 1.11	-3.9 ± 1.01
30	ALGO-FW ($J = 1$)	-5.85 ± 1.53	-7.98 ± 1.04	-1.71 ± 0.93	-3 ± 0.65
	ALGO-FW ($J = 5$)	-5.64 ± 1.72	-7.94 ± 1.11	-2.26 ± 0.91	-3.16 ± 0.75
	ALGO-FW ($J = 10$)	-5.59 ± 1.76	-7.89 ± 1.14	-2.35 ± 0.98	-3.2 ± 0.79
	ALGO-FW ($J = 100$)	-5.51 ± 1.83	-7.86 ± 1.19	-2.36 ± 0.99	-3.42 ± 0.79
20	ALGO-FW ($J = 1$)	-5.25 ± 1.25	-7 ± 0.79	-1.5 ± 0.71	-2.37 ± 0.46
	ALGO-FW ($J = 5$)	-4.96 ± 1.35	-6.73 ± 0.79	-2 ± 0.63	-2.53 ± 0.47
	ALGO-FW ($J = 10$)	-4.84 ± 1.3	-6.47 ± 0.77	-2.07 ± 0.64	-2.54 ± 0.48
	ALGO-FW ($J = 100$)	-4.07 ± 1.19	-5.72 ± 0.67	-2.07 ± 0.64	-2.63 ± 0.48
10	ALGO-FW ($J = 1$)	-2.66 ± 0.83	-4.11 ± 0.38	-1 ± 0.33	-1.37 ± 0.27
	ALGO-FW ($J = 5$)	-1.75 ± 0.77	-3.23 ± 0.34	-1.25 ± 0.31	-1.49 ± 0.28
	ALGO-FW ($J = 10$)	-1.41 ± 0.73	-2.94 ± 0.33	-1.26 ± 0.31	-1.49 ± 0.27
	ALGO-FW ($J = 100$)	-0.49 ± 0.64	-2.24 ± 0.31	-1.19 ± 0.29	-1.43 ± 0.26

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-FW ($J = 1$)	18.7 ± 1.39	-15 ± 0.81	4.9 ± 0.68	28.6 ± 0.91
	ALGO-FW ($J = 5$)	18.51 ± 1.71	-14.9 ± 0.88	4.93 ± 0.8	28.77 ± 1.06
	ALGO-FW ($J = 10$)	18.56 ± 1.9	-14.93 ± 0.96	4.88 ± 0.85	28.89 ± 1.22
	ALGO-FW ($J = 100$)	20.33 ± 2.29	-15.81 ± 1.12	4.25 ± 0.86	30.41 ± 1.6
30	ALGO-FW ($J = 1$)	18.43 ± 1.33	-14.87 ± 0.74	5.26 ± 0.67	26.62 ± 0.78
	ALGO-FW ($J = 5$)	18.26 ± 1.56	-14.79 ± 0.78	5.32 ± 0.77	26.61 ± 0.77
	ALGO-FW ($J = 10$)	18.38 ± 1.71	-14.85 ± 0.81	5.27 ± 0.8	26.73 ± 0.88
	ALGO-FW ($J = 100$)	19.86 ± 1.78	-15.58 ± 0.84	4.82 ± 0.75	27.81 ± 1.11
20	ALGO-FW ($J = 1$)	16.89 ± 0.91	-14.09 ± 0.62	6.83 ± 0.59	22.99 ± 0.75
	ALGO-FW ($J = 5$)	16.78 ± 0.98	-14.04 ± 0.62	7.04 ± 0.61	22.94 ± 0.72
	ALGO-FW ($J = 10$)	16.77 ± 0.96	-14.03 ± 0.62	7.16 ± 0.59	22.93 ± 0.73
	ALGO-FW ($J = 100$)	16.79 ± 0.71	-14.04 ± 0.63	7.62 ± 0.49	22.7 ± 0.93
10	ALGO-FW ($J = 1$)	11.88 ± 0.36	-11.6 ± 0.67	13.11 ± 0.53	17.5 ± 0.73
	ALGO-FW ($J = 5$)	11.1 ± 0.29	-11.21 ± 0.7	14.62 ± 0.49	16.68 ± 0.81
	ALGO-FW ($J = 10$)	10.74 ± 0.26	-11.03 ± 0.71	15.32 ± 0.46	16.31 ± 0.85
	ALGO-FW ($J = 100$)	9.72 ± 0.26	-10.52 ± 0.68	17.4 ± 0.5	15.26 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-FW ($J = 1$)	14.12 ± 3.76	1368.58 ± 202.94
	ALGO-FW ($J = 5$)	34.32 ± 9.89	826.36 ± 130.66
	ALGO-FW ($J = 10$)	54.18 ± 15.78	662.62 ± 111.2
	ALGO-FW ($J = 100$)	279.61 ± 90.41	324.34 ± 58.81
30	ALGO-FW ($J = 1$)	6.25 ± 1.66	591.12 ± 84.88
	ALGO-FW ($J = 5$)	12.64 ± 3.51	308.62 ± 48.58
	ALGO-FW ($J = 10$)	18.85 ± 5.21	234.83 ± 35.44
	ALGO-FW ($J = 100$)	105.98 ± 33.02	120.38 ± 14.36
20	ALGO-FW ($J = 1$)	2.78 ± 0.66	247.13 ± 31.71
	ALGO-FW ($J = 5$)	5.21 ± 1.35	126.8 ± 14.12
	ALGO-FW ($J = 10$)	8.12 ± 2.02	101.2 ± 10.17
	ALGO-FW ($J = 100$)	53.2 ± 16.14	60.19 ± 5.32
10	ALGO-FW ($J = 1$)	1.93 ± 0.45	169.92 ± 15.76
	ALGO-FW ($J = 5$)	4.06 ± 0.99	98.08 ± 7.28
	ALGO-FW ($J = 10$)	6.24 ± 1.48	76.62 ± 5.96
	ALGO-FW ($J = 100$)	43.15 ± 14.27	48.27 ± 7.2

Table B.5: Average HU and HSR performance of ALGO-FW on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-5.92 ± 1.72	-7.93 ± 1.07	-2.72 ± 0.93	-3.14 ± 0.87
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-6.11 ± 1.67	-8.02 ± 1.09	-2.79 ± 0.95	-3.26 ± 0.89
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-6.16 ± 1.71	-8.07 ± 1.09	-2.83 ± 0.97	-3.35 ± 0.92
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-5.75 ± 1.62	-7.76 ± 1.03	-2.32 ± 0.81	-2.73 ± 0.71
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-5.97 ± 1.62	-7.88 ± 1.07	-2.45 ± 0.83	-2.9 ± 0.75
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-6.1 ± 1.61	-7.95 ± 1.1	-2.53 ± 0.81	-3.02 ± 0.74
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-5.32 ± 1.3	-6.8 ± 0.8	-1.87 ± 0.58	-2.11 ± 0.48
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-5.55 ± 1.22	-6.84 ± 0.8	-2.08 ± 0.58	-2.31 ± 0.47
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-5.63 ± 1.22	-6.82 ± 0.81	-2.16 ± 0.56	-2.4 ± 0.46
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-2.93 ± 0.76	-4.15 ± 0.4	-0.88 ± 0.32	-1.18 ± 0.27
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-2.8 ± 0.72	-3.87 ± 0.4	-1.03 ± 0.3	-1.29 ± 0.26
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-2.59 ± 0.71	-3.59 ± 0.41	-1.12 ± 0.3	-1.36 ± 0.26

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	18.21 ± 1.39	-14.75 ± 0.64	5.09 ± 0.75	28.1 ± 0.87
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	18.47 ± 1.42	-14.88 ± 0.65	4.96 ± 0.74	28.39 ± 0.87
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	18.72 ± 1.46	-15.01 ± 0.66	4.85 ± 0.74	28.7 ± 0.91
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	17.72 ± 1.28	-14.51 ± 0.6	5.61 ± 0.72	25.93 ± 0.68
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	17.96 ± 1.32	-14.63 ± 0.6	5.47 ± 0.72	26.14 ± 0.7
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	18.22 ± 1.36	-14.76 ± 0.6	5.34 ± 0.72	26.38 ± 0.72
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	16.45 ± 0.96	-13.87 ± 0.54	7.07 ± 0.66	22.55 ± 0.65
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	16.65 ± 0.98	-13.97 ± 0.54	6.97 ± 0.65	22.71 ± 0.69
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	16.83 ± 0.97	-14.06 ± 0.54	6.89 ± 0.63	22.89 ± 1.36
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	11.82 ± 0.41	-11.57 ± 0.64	13.09 ± 0.58	17.39 ± 0.67
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	11.66 ± 0.37	-11.49 ± 0.65	13.5 ± 0.57	17.25 ± 0.73
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	11.46 ± 0.5	-11.39 ± 0.76	13.9 ± 0.71	17.08 ± 1.29

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	14.5 ± 5.7	2311.51 ± 310.8
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	18.18 ± 6.41	501.18 ± 71.55
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	19.9 ± 8.12	353.52 ± 62.71
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	6.06 ± 2.31	960.04 ± 121.7
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	6.96 ± 2.69	227.66 ± 37.07
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	9.43 ± 3.66	164.68 ± 30.88
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	2.64 ± 1	395.43 ± 46.69
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	3.16 ± 1.27	101.66 ± 16.65
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	4.39 ± 1.65	75.07 ± 13.25
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	1.67 ± 0.6	234.83 ± 29.38
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	2.46 ± 1.02	70.46 ± 13.19
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	3.41 ± 1.34	58.52 ± 12.66

Table B.6: Average HU and HSR performance of ALGO-Hybrid (S: FW A: PG) on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-5.79 ± 1.64	-6.94 ± 1.12	-2.74 ± 0.84	-3.75 ± 0.75
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-5.98 ± 1.54	-7.19 ± 1.12	-2.74 ± 0.89	-3.93 ± 0.8
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-5.69 ± 1.62	-7.18 ± 1.13	-2.69 ± 0.96	-3.96 ± 0.85
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-5.7 ± 1.6	-6.92 ± 1.1	-2.7 ± 0.75	-3.33 ± 0.6
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-5.68 ± 1.45	-6.92 ± 1.07	-2.78 ± 0.83	-3.55 ± 0.66
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-5.37 ± 1.41	-6.77 ± 0.99	-2.74 ± 0.91	-3.69 ± 0.69
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-4.72 ± 1.19	-5.48 ± 0.72	-2.46 ± 0.53	-2.6 ± 0.42
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-4.1 ± 0.99	-4.91 ± 0.64	-2.56 ± 0.57	-2.77 ± 0.44
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-3.51 ± 0.95	-4.63 ± 0.57	-2.48 ± 0.61	-2.82 ± 0.46
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-0.98 ± 0.68	-2.42 ± 0.33	-1.52 ± 0.34	-1.63 ± 0.27
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-0.28 ± 0.69	-2 ± 0.34	-1.48 ± 0.33	-1.58 ± 0.28
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-0.09 ± 0.68	-1.89 ± 0.33	-1.45 ± 0.32	-1.56 ± 0.28

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	21.38 ± 1.98	-16.34 ± 0.95	3.99 ± 0.79	31.22 ± 1.64
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	21.71 ± 1.94	-16.5 ± 0.96	3.91 ± 0.76	31.33 ± 1.61
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	21.78 ± 1.96	-16.54 ± 0.99	3.93 ± 0.77	31.24 ± 1.69
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	20.18 ± 1.56	-15.75 ± 0.74	4.68 ± 0.7	28.01 ± 1.14
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	20.41 ± 1.47	-15.86 ± 0.74	4.71 ± 0.67	28.01 ± 1.17
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	20.47 ± 1.35	-15.89 ± 0.73	4.75 ± 0.63	27.92 ± 1.16
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	17.09 ± 0.78	-14.19 ± 0.59	7.19 ± 0.52	23.02 ± 0.88
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	16.63 ± 0.65	-13.96 ± 0.61	7.78 ± 0.49	22.54 ± 0.95
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$	16.34 ± 0.58	-13.81 ± 0.63	8.14 ± 0.45	22.25 ± 1.52
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	10.67 ± 0.31	-10.99 ± 0.69	15.44 ± 0.58	16.21 ± 0.89
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	9.85 ± 0.44	-10.58 ± 0.74	17.08 ± 0.82	15.38 ± 1.02
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	9.56 ± 0.43	-10.43 ± 0.73	17.68 ± 0.83	15.12 ± 1.28

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	7.75 ± 1.54	925.46 ± 183.12
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	8.4 ± 3.28	232.91 ± 52.6
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	10.7 ± 4.55	190.79 ± 38.24
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	3.59 ± 0.95	448.93 ± 85.72
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	3.64 ± 1.48	120.76 ± 24.67
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	5.43 ± 2.1	95.07 ± 16.93
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	1.3 ± 0.43	203.93 ± 20.73
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	1.78 ± 0.72	59.38 ± 8.86
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$	2.73 ± 1	46.53 ± 6.85
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	1.01 ± 0.32	146.19 ± 13.65
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	1.56 ± 0.66	47.06 ± 7.65
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	2.23 ± 0.86	39.1 ± 7.16

Table B.7: Average HU and HSR performance of ALGO-Hybrid (S: FW A: FISTA) on synthetic data. $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP ($J = 1$)	-5.21 ± 0.97	-7.69 ± 0.6	-2.12 ± 0.35	-2.07 ± 0.3
	ALGO-GP ($J = 5$)	-5.29 ± 0.98	-7.68 ± 0.59	-2.03 ± 0.33	-2.01 ± 0.3
	ALGO-GP ($J = 10$)	-5.25 ± 0.96	-7.7 ± 0.59	-2.16 ± 0.31	-2.09 ± 0.29
	ALGO-GP ($J = 100$)	-4.88 ± 0.74	-7.06 ± 0.55	0 ± 0.46	-1.2 ± 0.27
30	ALGO-GP ($J = 1$)	-5.14 ± 0.9	-7.48 ± 0.56	-1.89 ± 0.34	-1.85 ± 0.27
	ALGO-GP ($J = 5$)	-5.21 ± 0.87	-7.46 ± 0.55	-1.7 ± 0.29	-1.72 ± 0.24
	ALGO-GP ($J = 10$)	-5.13 ± 0.87	-7.42 ± 0.53	-1.82 ± 0.26	-1.78 ± 0.23
	ALGO-GP ($J = 100$)	-4.74 ± 0.7	-6.71 ± 0.52	0.09 ± 0.41	-0.98 ± 0.23
20	ALGO-GP ($J = 1$)	-4.62 ± 0.69	-6.01 ± 0.4	-1.18 ± 0.26	-1.22 ± 0.19
	ALGO-GP ($J = 5$)	-4.55 ± 0.64	-5.78 ± 0.35	-0.67 ± 0.21	-0.94 ± 0.16
	ALGO-GP ($J = 10$)	-4.23 ± 0.63	-5.5 ± 0.33	-0.58 ± 0.21	-0.89 ± 0.16
	ALGO-GP ($J = 100$)	-3.55 ± 0.51	-4.74 ± 0.3	0.81 ± 0.26	-0.31 ± 0.16
10	ALGO-GP ($J = 1$)	-1.31 ± 0.36	-2.58 ± 0.2	0.75 ± 0.22	-0.04 ± 0.11
	ALGO-GP ($J = 5$)	-0.77 ± 0.31	-2.17 ± 0.15	1.26 ± 0.15	0.16 ± 0.09
	ALGO-GP ($J = 10$)	-0.42 ± 0.31	-1.89 ± 0.18	1.51 ± 0.15	0.25 ± 0.09
	ALGO-GP ($J = 100$)	0.13 ± 0.3	-1.63 ± 0.17	2.01 ± 0.08	0.42 ± 0.07

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP ($J = 1$)	20.21 ± 0.9	-15.82 ± 0.38	4.22 ± 0.4	28.9 ± 0.56
	ALGO-GP ($J = 5$)	20.11 ± 0.83	-15.77 ± 0.39	4.28 ± 0.38	28.78 ± 0.55
	ALGO-GP ($J = 10$)	20.19 ± 0.84	-15.81 ± 0.34	4.22 ± 0.4	28.79 ± 0.59
	ALGO-GP ($J = 100$)	20.01 ± 0.79	-15.72 ± 0.4	4.37 ± 0.36	28.19 ± 0.53
30	ALGO-GP ($J = 1$)	19.78 ± 0.81	-15.61 ± 0.34	4.58 ± 0.38	26.79 ± 0.46
	ALGO-GP ($J = 5$)	19.68 ± 0.75	-15.55 ± 0.36	4.67 ± 0.36	26.6 ± 0.44
	ALGO-GP ($J = 10$)	19.66 ± 0.73	-15.55 ± 0.33	4.69 ± 0.35	26.44 ± 0.45
	ALGO-GP ($J = 100$)	19.26 ± 0.67	-15.35 ± 0.38	5.12 ± 0.32	25.73 ± 0.43
20	ALGO-GP ($J = 1$)	17.44 ± 0.45	-14.43 ± 0.36	6.64 ± 0.28	22.66 ± 0.46
	ALGO-GP ($J = 5$)	16.84 ± 0.33	-14.13 ± 0.4	7.36 ± 0.22	22.07 ± 0.49
	ALGO-GP ($J = 10$)	16.22 ± 0.28	-13.82 ± 0.41	8.03 ± 0.19	21.47 ± 0.49
	ALGO-GP ($J = 100$)	14.85 ± 0.23	-13.13 ± 0.44	9.84 ± 0.22	20.05 ± 0.6
10	ALGO-GP ($J = 1$)	10.1 ± 0.16	-10.74 ± 0.47	16.58 ± 0.32	15.2 ± 0.58
	ALGO-GP ($J = 5$)	8.95 ± 0.11	-10.16 ± 0.46	19.05 ± 0.24	14.03 ± 0.59
	ALGO-GP ($J = 10$)	8.29 ± 0.13	-9.84 ± 0.43	20.53 ± 0.3	13.37 ± 0.58
	ALGO-GP ($J = 100$)	7.27 ± 0.21	-9.32 ± 0.36	23.01 ± 0.5	12.35 ± 0.56

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP ($J = 1$)	13.5 ± 5.85	878.83 ± 133.88
	ALGO-GP ($J = 5$)	36.52 ± 14.08	383.95 ± 41.03
	ALGO-GP ($J = 10$)	48.87 ± 18.85	256.69 ± 26.62
	ALGO-GP ($J = 100$)	246.94 ± 74.52	159.92 ± 24.28
30	ALGO-GP ($J = 1$)	6.87 ± 3.2	426.98 ± 59.75
	ALGO-GP ($J = 5$)	17.99 ± 6.83	193.5 ± 15.97
	ALGO-GP ($J = 10$)	28.74 ± 10.51	155.86 ± 10.26
	ALGO-GP ($J = 100$)	136.78 ± 36.5	86.22 ± 11.89
20	ALGO-GP ($J = 1$)	3.22 ± 1.55	178.06 ± 17.08
	ALGO-GP ($J = 5$)	11.49 ± 3.98	122.21 ± 6.89
	ALGO-GP ($J = 10$)	18.83 ± 6.51	100.81 ± 6.66
	ALGO-GP ($J = 100$)	78.48 ± 18.32	50.04 ± 5.78
10	ALGO-GP ($J = 1$)	2.38 ± 1.01	134.34 ± 10.23
	ALGO-GP ($J = 5$)	6.8 ± 2.34	72.59 ± 4.25
	ALGO-GP ($J = 10$)	10.99 ± 3.99	58.89 ± 6.38
	ALGO-GP ($J = 100$)	59.05 ± 16.55	38.73 ± 3.31

Table B.8: Average HU and HSR performance of ALGO-GP on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-BBGP ($J = 1$)	-5.29 ± 0.88	-7.73 ± 0.58	-1.87 ± 0.32	-1.95 ± 0.28
	ALGO-BBGP ($J = 5$)	-5.29 ± 0.88	-7.73 ± 0.58	-1.96 ± 0.33	-1.99 ± 0.29
	ALGO-BBGP ($J = 10$)	-5.26 ± 0.9	-7.65 ± 0.59	-1.92 ± 0.32	-1.97 ± 0.27
	ALGO-BBGP ($J = 100$)	-4.92 ± 0.72	-7.08 ± 0.54	-0.01 ± 0.45	-1.21 ± 0.26
30	ALGO-BBGP ($J = 1$)	-5.21 ± 0.83	-7.48 ± 0.67	-1.54 ± 0.3	-1.66 ± 0.25
	ALGO-BBGP ($J = 5$)	-5.23 ± 0.85	-7.48 ± 0.68	-1.61 ± 0.3	-1.68 ± 0.26
	ALGO-BBGP ($J = 10$)	-5.22 ± 0.85	-7.4 ± 0.67	-1.46 ± 0.28	-1.6 ± 0.24
	ALGO-BBGP ($J = 100$)	-4.82 ± 0.73	-6.74 ± 0.63	0.09 ± 0.4	-0.99 ± 0.22
20	ALGO-BBGP ($J = 1$)	-4.74 ± 0.62	-6.22 ± 0.38	-0.52 ± 0.22	-0.87 ± 0.16
	ALGO-BBGP ($J = 5$)	-4.48 ± 0.6	-5.76 ± 0.33	-0.57 ± 0.2	-0.9 ± 0.15
	ALGO-BBGP ($J = 10$)	-4.45 ± 0.61	-5.65 ± 0.34	-0.14 ± 0.19	-0.67 ± 0.15
	ALGO-BBGP ($J = 100$)	-3.65 ± 0.51	-4.83 ± 0.28	0.83 ± 0.24	-0.3 ± 0.15
10	ALGO-BBGP ($J = 1$)	-1.37 ± 0.31	-2.68 ± 0.18	1.14 ± 0.14	0.1 ± 0.09
	ALGO-BBGP ($J = 5$)	-0.66 ± 0.29	-2.16 ± 0.16	1.46 ± 0.12	0.22 ± 0.08
	ALGO-BBGP ($J = 10$)	-0.59 ± 0.29	-2.03 ± 0.16	1.69 ± 0.11	0.32 ± 0.07
	ALGO-BBGP ($J = 100$)	0.09 ± 0.29	-1.65 ± 0.18	2.01 ± 0.1	0.42 ± 0.07

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-BBGP ($J = 1$)	20.44 ± 0.82	-15.93 ± 0.4	4.1 ± 0.36	29.13 ± 0.58
	ALGO-BBGP ($J = 5$)	20.31 ± 0.84	-15.87 ± 0.43	4.16 ± 0.37	28.98 ± 0.55
	ALGO-BBGP ($J = 10$)	20.17 ± 0.82	-15.8 ± 0.39	4.21 ± 0.38	28.81 ± 0.57
	ALGO-BBGP ($J = 100$)	20.08 ± 0.78	-15.75 ± 0.4	4.36 ± 0.36	28.22 ± 0.53
30	ALGO-BBGP ($J = 1$)	19.86 ± 1.3	-15.63 ± 0.93	4.5 ± 0.42	26.81 ± 1.55
	ALGO-BBGP ($J = 5$)	19.86 ± 1.32	-15.63 ± 0.94	4.53 ± 0.42	26.71 ± 1.53
	ALGO-BBGP ($J = 10$)	19.62 ± 1.3	-15.51 ± 0.93	4.63 ± 0.42	26.45 ± 1.51
	ALGO-BBGP ($J = 100$)	19.3 ± 1.25	-15.35 ± 0.92	5.09 ± 0.42	25.74 ± 1.48
20	ALGO-BBGP ($J = 1$)	17.3 ± 0.41	-14.36 ± 0.38	6.74 ± 0.27	22.52 ± 0.47
	ALGO-BBGP ($J = 5$)	16.78 ± 0.32	-14.1 ± 0.4	7.45 ± 0.21	22 ± 0.49
	ALGO-BBGP ($J = 10$)	16.29 ± 0.3	-13.86 ± 0.41	7.91 ± 0.2	21.53 ± 0.49
	ALGO-BBGP ($J = 100$)	14.92 ± 0.21	-13.17 ± 0.45	9.75 ± 0.2	20.12 ± 0.6
10	ALGO-BBGP ($J = 1$)	9.87 ± 0.34	-10.62 ± 0.56	17.01 ± 0.6	14.98 ± 0.73
	ALGO-BBGP ($J = 5$)	8.71 ± 0.3	-10.04 ± 0.54	19.53 ± 0.67	13.79 ± 0.72
	ALGO-BBGP ($J = 10$)	8.31 ± 0.29	-9.84 ± 0.53	20.45 ± 0.69	13.39 ± 0.71
	ALGO-BBGP ($J = 100$)	7.31 ± 0.31	-9.34 ± 0.47	22.88 ± 0.87	12.38 ± 0.69

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-BBGP ($J = 1$)	17.11 ± 7.58	1053.71 ± 109.26
	ALGO-BBGP ($J = 5$)	40.33 ± 15.9	395.03 ± 46.67
	ALGO-BBGP ($J = 10$)	58.6 ± 22.3	291.66 ± 34.16
	ALGO-BBGP ($J = 100$)	273 ± 82.88	163.54 ± 21.97
30	ALGO-BBGP ($J = 1$)	8.7 ± 4.08	512.71 ± 47.98
	ALGO-BBGP ($J = 5$)	18.87 ± 7.71	193.52 ± 20.52
	ALGO-BBGP ($J = 10$)	32.73 ± 12.51	167.6 ± 12.07
	ALGO-BBGP ($J = 100$)	150.23 ± 34.68	88.32 ± 9.02
20	ALGO-BBGP ($J = 1$)	4.79 ± 2.26	253.77 ± 18.48
	ALGO-BBGP ($J = 5$)	10.96 ± 3.91	109.97 ± 10.37
	ALGO-BBGP ($J = 10$)	24.37 ± 8.68	122.23 ± 6.47
	ALGO-BBGP ($J = 100$)	92.49 ± 20.72	54.18 ± 4.07
10	ALGO-BBGP ($J = 1$)	2.54 ± 1.13	136.7 ± 11.42
	ALGO-BBGP ($J = 5$)	7.83 ± 2.78	77.65 ± 4.92
	ALGO-BBGP ($J = 10$)	14.88 ± 5.26	75.16 ± 4.82
	ALGO-BBGP ($J = 100$)	69.47 ± 21.82	41.36 ± 3.41

Table B.9: Average HU and HSR performance of ALGO-BBGP on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-PG ($J = 1$)	-5.59 ± 0.94	-7.93 ± 0.64	-3.22 ± 0.26	-2.8 ± 0.25
	ALGO-PG ($J = 5$)	-5.57 ± 0.95	-7.87 ± 0.64	-3.18 ± 0.26	-2.77 ± 0.25
	ALGO-PG ($J = 10$)	-5.57 ± 0.95	-7.83 ± 0.64	-3.18 ± 0.26	-2.77 ± 0.25
	ALGO-PG ($J = 100$)	-5.11 ± 0.99	-7.52 ± 0.58	-3.18 ± 0.25	-2.74 ± 0.25
30	ALGO-PG ($J = 1$)	-5.43 ± 0.86	-7.61 ± 0.54	-3.14 ± 0.24	-2.71 ± 0.23
	ALGO-PG ($J = 5$)	-5.4 ± 0.87	-7.51 ± 0.55	-2.99 ± 0.22	-2.57 ± 0.21
	ALGO-PG ($J = 10$)	-5.36 ± 0.86	-7.42 ± 0.54	-2.95 ± 0.22	-2.54 ± 0.21
	ALGO-PG ($J = 100$)	-4.89 ± 0.86	-7.02 ± 0.49	-2.88 ± 0.21	-2.46 ± 0.21
20	ALGO-PG ($J = 1$)	-4.66 ± 0.63	-6.1 ± 0.34	-2.44 ± 0.17	-2.06 ± 0.16
	ALGO-PG ($J = 5$)	-4.43 ± 0.61	-5.59 ± 0.31	-2.09 ± 0.16	-1.79 ± 0.15
	ALGO-PG ($J = 10$)	-4.3 ± 0.6	-5.42 ± 0.3	-1.97 ± 0.16	-1.7 ± 0.15
	ALGO-PG ($J = 100$)	-3.56 ± 0.63	-4.81 ± 0.26	-1.54 ± 0.15	-1.41 ± 0.15
10	ALGO-PG ($J = 1$)	-2.09 ± 0.34	-3.23 ± 0.19	-0.26 ± 0.12	-0.52 ± 0.1
	ALGO-PG ($J = 5$)	-1.31 ± 0.3	-2.54 ± 0.16	0.22 ± 0.11	-0.28 ± 0.09
	ALGO-PG ($J = 10$)	-0.99 ± 0.3	-2.32 ± 0.15	0.41 ± 0.11	-0.19 ± 0.09
	ALGO-PG ($J = 100$)	-0.23 ± 0.31	-1.82 ± 0.16	1.14 ± 0.1	0.11 ± 0.08

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-PG ($J = 1$)	22.3 ± 0.98	-16.86 ± 0.35	3.58 ± 0.37	30.57 ± 0.83
	ALGO-PG ($J = 5$)	22.42 ± 1.02	-16.92 ± 0.35	3.52 ± 0.38	30.9 ± 0.84
	ALGO-PG ($J = 10$)	22.41 ± 1.02	-16.92 ± 0.35	3.52 ± 0.39	30.93 ± 0.85
	ALGO-PG ($J = 100$)	21.36 ± 0.92	-16.39 ± 0.38	3.76 ± 0.38	29.89 ± 0.73
30	ALGO-PG ($J = 1$)	21.16 ± 0.75	-16.3 ± 0.3	4.18 ± 0.31	27.81 ± 0.64
	ALGO-PG ($J = 5$)	21.54 ± 0.84	-16.48 ± 0.31	3.98 ± 0.35	28.24 ± 0.66
	ALGO-PG ($J = 10$)	21.52 ± 0.84	-16.47 ± 0.31	4 ± 0.35	28.21 ± 0.65
	ALGO-PG ($J = 100$)	20.6 ± 0.78	-16.02 ± 0.35	4.32 ± 0.34	27.32 ± 0.56
20	ALGO-PG ($J = 1$)	18.3 ± 0.46	-14.86 ± 0.33	5.97 ± 0.32	23.48 ± 0.54
	ALGO-PG ($J = 5$)	18.09 ± 0.43	-14.75 ± 0.34	6.26 ± 0.29	23.26 ± 0.54
	ALGO-PG ($J = 10$)	17.87 ± 0.41	-14.65 ± 0.34	6.49 ± 0.27	23.05 ± 0.53
	ALGO-PG ($J = 100$)	16.39 ± 0.31	-13.91 ± 0.4	7.94 ± 0.21	21.66 ± 0.5
10	ALGO-PG ($J = 1$)	11.85 ± 0.23	-11.61 ± 0.41	13.23 ± 0.33	17.09 ± 0.47
	ALGO-PG ($J = 5$)	10.63 ± 0.17	-11.01 ± 0.42	15.53 ± 0.3	15.76 ± 0.53
	ALGO-PG ($J = 10$)	10.09 ± 0.16	-10.73 ± 0.43	16.61 ± 0.29	15.19 ± 0.55
	ALGO-PG ($J = 100$)	8.48 ± 0.16	-9.93 ± 0.4	20.06 ± 0.35	13.56 ± 0.56

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-PG ($J = 1$)	36.28 ± 14.9	3150.36 ± 203.2
	ALGO-PG ($J = 5$)	82.03 ± 30.03	1265.71 ± 87.77
	ALGO-PG ($J = 10$)	107.78 ± 38.83	841.97 ± 55.47
	ALGO-PG ($J = 100$)	219.61 ± 58.6	206.66 ± 18.57
30	ALGO-PG ($J = 1$)	15.95 ± 7.2	1329.99 ± 110.83
	ALGO-PG ($J = 5$)	44.11 ± 15.67	691.99 ± 19.99
	ALGO-PG ($J = 10$)	55.52 ± 20.28	439.24 ± 10.94
	ALGO-PG ($J = 100$)	146.86 ± 31.24	135 ± 6.44
20	ALGO-PG ($J = 1$)	9.99 ± 4.4	758.43 ± 26.36
	ALGO-PG ($J = 5$)	20.55 ± 6.86	318.24 ± 6.67
	ALGO-PG ($J = 10$)	29.23 ± 10.03	227.55 ± 5.55
	ALGO-PG ($J = 100$)	97.46 ± 19.34	90.13 ± 3.81
10	ALGO-PG ($J = 1$)	4.92 ± 2.16	369.88 ± 9.45
	ALGO-PG ($J = 5$)	12 ± 3.99	186.19 ± 4.76
	ALGO-PG ($J = 10$)	18.05 ± 6.07	141.09 ± 4.16
	ALGO-PG ($J = 100$)	58.97 ± 15.05	56.15 ± 2.91

Table B.10: Average HU and HSR performance of ALGO-PG on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-FISTA ($J = 1$)	-5.89 ± 1.07	-7.94 ± 0.67	-3 ± 0.28	-2.68 ± 0.27
	ALGO-FISTA ($J = 5$)	-5.91 ± 1.06	-7.95 ± 0.67	-2.93 ± 0.28	-2.62 ± 0.27
	ALGO-FISTA ($J = 10$)	-5.83 ± 1.09	-7.88 ± 0.67	-2.92 ± 0.28	-2.6 ± 0.27
	ALGO-FISTA ($J = 100$)	-4.2 ± 0.73	-6.32 ± 0.53	-0.17 ± 0.36	-1.01 ± 0.23
30	ALGO-FISTA ($J = 1$)	-5.5 ± 0.95	-7.28 ± 0.56	-2.66 ± 0.25	-2.35 ± 0.23
	ALGO-FISTA ($J = 5$)	-5.39 ± 0.91	-7.1 ± 0.51	-2.46 ± 0.23	-2.2 ± 0.22
	ALGO-FISTA ($J = 10$)	-5.27 ± 0.96	-7.02 ± 0.51	-2.4 ± 0.24	-2.15 ± 0.22
	ALGO-FISTA ($J = 100$)	-4.11 ± 0.73	-5.97 ± 0.43	0.02 ± 0.35	-0.81 ± 0.2
20	ALGO-FISTA ($J = 1$)	-3.76 ± 0.65	-4.81 ± 0.33	-1.41 ± 0.25	-1.36 ± 0.2
	ALGO-FISTA ($J = 5$)	-3.14 ± 0.62	-4.38 ± 0.25	-0.85 ± 0.17	-1.03 ± 0.15
	ALGO-FISTA ($J = 10$)	-3.07 ± 0.63	-4.38 ± 0.26	-0.6 ± 0.17	-0.88 ± 0.15
	ALGO-FISTA ($J = 100$)	-3.05 ± 0.54	-4.37 ± 0.26	1.32 ± 0.18	-0.06 ± 0.13
10	ALGO-FISTA ($J = 1$)	-0.19 ± 0.32	-1.85 ± 0.16	1.03 ± 0.11	0.06 ± 0.08
	ALGO-FISTA ($J = 5$)	0.17 ± 0.34	-1.72 ± 0.17	1.6 ± 0.1	0.27 ± 0.07
	ALGO-FISTA ($J = 10$)	0.16 ± 0.33	-1.66 ± 0.17	1.78 ± 0.09	0.34 ± 0.07
	ALGO-FISTA ($J = 100$)	0.2 ± 0.31	-1.6 ± 0.18	2.11 ± 0.08	0.46 ± 0.07

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-FISTA ($J = 1$)	24.18 ± 1.47	-17.8 ± 0.64	3.07 ± 0.54	32.22 ± 1.55
	ALGO-FISTA ($J = 5$)	24.24 ± 1.16	-17.83 ± 0.46	3.05 ± 0.37	32.23 ± 1
	ALGO-FISTA ($J = 10$)	23.87 ± 1.19	-17.65 ± 0.46	3.15 ± 0.39	31.92 ± 0.99
	ALGO-FISTA ($J = 100$)	19.48 ± 0.96	-15.45 ± 0.46	4.96 ± 0.47	27.87 ± 0.82
30	ALGO-FISTA ($J = 1$)	22.31 ± 1.17	-16.87 ± 0.58	3.93 ± 0.48	28.49 ± 1.17
	ALGO-FISTA ($J = 5$)	22.04 ± 0.73	-16.74 ± 0.35	4.11 ± 0.29	28.11 ± 0.7
	ALGO-FISTA ($J = 10$)	21.67 ± 0.78	-16.55 ± 0.35	4.27 ± 0.32	27.77 ± 0.67
	ALGO-FISTA ($J = 100$)	18.77 ± 0.88	-15.1 ± 0.46	5.73 ± 0.45	25.37 ± 0.6
20	ALGO-FISTA ($J = 1$)	16.65 ± 0.28	-14.04 ± 0.41	7.87 ± 0.26	21.82 ± 0.6
	ALGO-FISTA ($J = 5$)	15.35 ± 0.23	-13.39 ± 0.43	9.27 ± 0.23	20.57 ± 0.59
	ALGO-FISTA ($J = 10$)	14.96 ± 0.26	-13.19 ± 0.44	9.69 ± 0.27	20.21 ± 0.59
	ALGO-FISTA ($J = 100$)	13.77 ± 0.31	-12.59 ± 0.44	11.15 ± 0.33	19.03 ± 0.6
10	ALGO-FISTA ($J = 1$)	8.67 ± 0.16	-10.03 ± 0.41	19.6 ± 0.36	13.75 ± 0.57
	ALGO-FISTA ($J = 5$)	7.86 ± 0.17	-9.62 ± 0.39	21.5 ± 0.4	12.93 ± 0.57
	ALGO-FISTA ($J = 10$)	7.6 ± 0.2	-9.49 ± 0.37	22.15 ± 0.46	12.68 ± 0.56
	ALGO-FISTA ($J = 100$)	7.12 ± 0.24	-9.25 ± 0.35	23.39 ± 0.59	12.2 ± 0.56

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-FISTA ($J = 1$)	6.46 ± 2.86	495 ± 21.42
	ALGO-FISTA ($J = 5$)	10.31 ± 3.66	152.81 ± 7.52
	ALGO-FISTA ($J = 10$)	12.26 ± 4.4	93.19 ± 5.21
	ALGO-FISTA ($J = 100$)	84.06 ± 35.64	77.14 ± 24.48
30	ALGO-FISTA ($J = 1$)	4.41 ± 2.14	333.85 ± 7.3
	ALGO-FISTA ($J = 5$)	7.47 ± 2.93	111.15 ± 4.47
	ALGO-FISTA ($J = 10$)	9.07 ± 3.37	69.72 ± 3.49
	ALGO-FISTA ($J = 100$)	42.44 ± 12.98	37.76 ± 7.76
20	ALGO-FISTA ($J = 1$)	3.57 ± 1.66	236.07 ± 5.91
	ALGO-FISTA ($J = 5$)	5.87 ± 1.99	86.33 ± 3.68
	ALGO-FISTA ($J = 10$)	7.56 ± 2.57	57.3 ± 2.9
	ALGO-FISTA ($J = 100$)	38.01 ± 8.16	34.36 ± 2.87
10	ALGO-FISTA ($J = 1$)	2.51 ± 1.05	164.86 ± 4.92
	ALGO-FISTA ($J = 5$)	4.66 ± 1.56	68.48 ± 3.46
	ALGO-FISTA ($J = 10$)	6.35 ± 2.14	48.22 ± 3.31
	ALGO-FISTA ($J = 100$)	40.21 ± 11.03	37.4 ± 3.37

Table B.11: Average HU and HSR performance of ALGO-FISTA on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-FW ($J = 1$)	-5.02 ± 0.78	-7.43 ± 0.53	0.41 ± 0.54	-1.44 ± 0.24
	ALGO-FW ($J = 5$)	-4.91 ± 0.83	-7.38 ± 0.55	0.02 ± 0.46	-1.42 ± 0.26
	ALGO-FW ($J = 10$)	-4.8 ± 0.83	-7.36 ± 0.57	0.22 ± 0.5	-1.34 ± 0.26
	ALGO-FW ($J = 100$)	-4.64 ± 0.89	-7.35 ± 0.58	0.49 ± 0.49	-1.31 ± 0.26
30	ALGO-FW ($J = 1$)	-4.95 ± 0.74	-7.26 ± 0.55	0.69 ± 0.5	-1.12 ± 0.21
	ALGO-FW ($J = 5$)	-4.87 ± 0.77	-7.21 ± 0.57	0.2 ± 0.4	-1.15 ± 0.22
	ALGO-FW ($J = 10$)	-4.77 ± 0.82	-7.17 ± 0.58	0.35 ± 0.42	-1.08 ± 0.23
	ALGO-FW ($J = 100$)	-4.53 ± 0.85	-7.02 ± 0.58	0.68 ± 0.42	-1.01 ± 0.23
20	ALGO-FW ($J = 1$)	-4.36 ± 0.65	-6.21 ± 0.39	0.52 ± 0.31	-0.64 ± 0.15
	ALGO-FW ($J = 5$)	-4.17 ± 0.63	-5.9 ± 0.35	0.24 ± 0.22	-0.64 ± 0.15
	ALGO-FW ($J = 10$)	-4.07 ± 0.61	-5.7 ± 0.34	0.37 ± 0.21	-0.57 ± 0.15
	ALGO-FW ($J = 100$)	-3.48 ± 0.6	-5.13 ± 0.3	0.78 ± 0.22	-0.42 ± 0.15
10	ALGO-FW ($J = 1$)	-1.64 ± 0.36	-3.13 ± 0.2	0.9 ± 0.12	-0.02 ± 0.09
	ALGO-FW ($J = 5$)	-0.96 ± 0.32	-2.45 ± 0.18	1 ± 0.11	0.04 ± 0.08
	ALGO-FW ($J = 10$)	-0.72 ± 0.31	-2.23 ± 0.18	1.18 ± 0.11	0.11 ± 0.08
	ALGO-FW ($J = 100$)	-0.11 ± 0.29	-1.8 ± 0.18	1.63 ± 0.1	0.28 ± 0.07

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-FW ($J = 1$)	19.14 ± 0.77	-15.28 ± 0.44	4.7 ± 0.37	27.95 ± 0.51
	ALGO-FW ($J = 5$)	18.87 ± 0.87	-15.15 ± 0.44	4.8 ± 0.41	27.7 ± 0.48
	ALGO-FW ($J = 10$)	18.8 ± 0.94	-15.11 ± 0.46	4.82 ± 0.42	27.64 ± 0.5
	ALGO-FW ($J = 100$)	19.7 ± 1.02	-15.56 ± 0.5	4.54 ± 0.39	28.29 ± 0.64
30	ALGO-FW ($J = 1$)	18.83 ± 0.93	-15.12 ± 0.63	5.02 ± 0.39	26.02 ± 0.94
	ALGO-FW ($J = 5$)	18.63 ± 0.98	-15.02 ± 0.62	5.15 ± 0.41	25.8 ± 0.9
	ALGO-FW ($J = 10$)	18.63 ± 1.03	-15.02 ± 0.64	5.19 ± 0.42	25.72 ± 0.91
	ALGO-FW ($J = 100$)	19.45 ± 1.04	-15.43 ± 0.64	5.1 ± 0.39	26.14 ± 0.96
20	ALGO-FW ($J = 1$)	16.83 ± 0.45	-14.12 ± 0.38	7.1 ± 0.29	22.14 ± 0.43
	ALGO-FW ($J = 5$)	16.33 ± 0.4	-13.88 ± 0.38	7.74 ± 0.25	21.69 ± 0.43
	ALGO-FW ($J = 10$)	16.03 ± 0.38	-13.72 ± 0.38	8.15 ± 0.24	21.37 ± 0.45
	ALGO-FW ($J = 100$)	15.24 ± 0.27	-13.33 ± 0.41	9.28 ± 0.22	20.47 ± 0.54
10	ALGO-FW ($J = 1$)	10.7 ± 0.38	-11.04 ± 0.54	15.31 ± 0.56	15.85 ± 0.71
	ALGO-FW ($J = 5$)	9.58 ± 0.34	-10.48 ± 0.53	17.58 ± 0.62	14.67 ± 0.71
	ALGO-FW ($J = 10$)	9.07 ± 0.32	-10.22 ± 0.52	18.7 ± 0.66	14.14 ± 0.71
	ALGO-FW ($J = 100$)	7.91 ± 0.31	-9.64 ± 0.49	21.36 ± 0.79	12.98 ± 0.69

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-FW ($J = 1$)	10.25 ± 4.93	928.87 ± 67.31
	ALGO-FW ($J = 5$)	34.17 ± 14.66	450.45 ± 33.36
	ALGO-FW ($J = 10$)	51.11 ± 22.48	339.87 ± 25.33
	ALGO-FW ($J = 100$)	221.52 ± 64.92	197.24 ± 18.69
30	ALGO-FW ($J = 1$)	4.43 ± 2.2	383.74 ± 25.04
	ALGO-FW ($J = 5$)	14.14 ± 5.9	196.51 ± 9.89
	ALGO-FW ($J = 10$)	22.11 ± 9.16	156.96 ± 7.76
	ALGO-FW ($J = 100$)	108.64 ± 23.68	94.74 ± 7.18
20	ALGO-FW ($J = 1$)	2.71 ± 1.42	205.78 ± 8.34
	ALGO-FW ($J = 5$)	9.3 ± 3.71	126.13 ± 4.73
	ALGO-FW ($J = 10$)	15.19 ± 6.04	104.89 ± 4.35
	ALGO-FW ($J = 100$)	68.41 ± 14.73	59.32 ± 3.23
10	ALGO-FW ($J = 1$)	2.14 ± 1.05	167.31 ± 5.03
	ALGO-FW ($J = 5$)	7.32 ± 2.88	99.94 ± 3.74
	ALGO-FW ($J = 10$)	11.46 ± 4.45	79.16 ± 3.14
	ALGO-FW ($J = 100$)	49.9 ± 14.79	44.62 ± 2.72

Table B.12: Average HU and HSR performance of ALGO-FW on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-4.53 ± 0.8	-7.19 ± 0.54	0.24 ± 0.34	-0.71 ± 0.24
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-4.64 ± 0.85	-7.26 ± 0.55	0.22 ± 0.38	-0.8 ± 0.26
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-4.72 ± 0.87	-7.3 ± 0.55	0.18 ± 0.39	-0.91 ± 0.25
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-4.53 ± 0.82	-7.02 ± 0.64	0.68 ± 0.3	-0.37 ± 0.21
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-4.66 ± 0.86	-7.11 ± 0.65	0.56 ± 0.35	-0.52 ± 0.23
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-4.8 ± 0.86	-7.15 ± 0.66	0.48 ± 0.35	-0.64 ± 0.23
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-4.47 ± 0.7	-6.31 ± 0.46	0.96 ± 0.21	-0.09 ± 0.15
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-4.62 ± 0.66	-6.28 ± 0.45	0.72 ± 0.24	-0.26 ± 0.17
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-4.63 ± 0.64	-6.19 ± 0.44	0.62 ± 0.24	-0.34 ± 0.16
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	-2.39 ± 0.4	-3.63 ± 0.23	1.55 ± 0.13	0.3 ± 0.09
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	-2.1 ± 0.4	-3.22 ± 0.26	1.41 ± 0.12	0.22 ± 0.08
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	-1.77 ± 0.39	-2.89 ± 0.26	1.38 ± 0.12	0.19 ± 0.08

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	19.15 ± 0.83	-15.29 ± 0.36	4.69 ± 0.42	28.02 ± 0.54
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	19.31 ± 0.84	-15.37 ± 0.37	4.65 ± 0.41	28.14 ± 0.54
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	19.5 ± 0.86	-15.46 ± 0.37	4.59 ± 0.41	28.29 ± 0.56
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	18.78 ± 1.28	-15.09 ± 0.89	5 ± 0.48	26.01 ± 1.5
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	18.94 ± 1.29	-15.17 ± 0.9	4.96 ± 0.48	26.09 ± 1.5
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	19.11 ± 1.3	-15.25 ± 0.9	4.92 ± 0.47	26.18 ± 1.51
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	17.06 ± 0.73	-14.24 ± 0.56	6.8 ± 0.38	22.34 ± 0.84
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	17 ± 0.71	-14.2 ± 0.56	6.97 ± 0.38	22.24 ± 0.85
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	16.91 ± 0.7	-14.16 ± 0.56	7.15 ± 0.38	22.13 ± 0.86
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	11.05 ± 0.39	-11.21 ± 0.54	14.7 ± 0.54	16.19 ± 0.71
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	10.53 ± 0.41	-10.95 ± 0.55	15.73 ± 0.67	15.65 ± 0.73
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	10.06 ± 0.41	-10.71 ± 0.55	16.68 ± 0.73	15.16 ± 0.74

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	13.11 ± 3.3	1903.32 ± 168.58
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	20.44 ± 6.48	423.84 ± 49.46
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	26.92 ± 8.68	295.24 ± 41.59
30	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	5.83 ± 1.64	784.67 ± 60.74
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	8.29 ± 3.41	192.38 ± 25.16
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	11.07 ± 4.74	137.54 ± 21.37
20	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	2.92 ± 0.87	375.19 ± 21.76
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	4.4 ± 1.81	98.61 ± 14.36
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	6.16 ± 2.66	74.94 ± 13.34
10	ALGO-Hybrid(S:FW A:PG) ($J = 1$)	2.11 ± 0.62	263.95 ± 13.04
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 10)$)	2.88 ± 1.14	74.2 ± 12.56
	ALGO-Hybrid(S:FW A:PG) ($J \sim \mathcal{U}(1, 20)$)	4.22 ± 1.87	58.44 ± 11.69

Table B.13: Average HU and HSR performance of ALGO-Hybrid (S: FW A: PG) on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-4.78 ± 0.97	-6.96 ± 0.61	0.26 ± 0.45	-1.11 ± 0.24
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-5.04 ± 0.82	-6.97 ± 0.6	0.27 ± 0.5	-1.41 ± 0.25
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-4.93 ± 0.78	-6.79 ± 0.6	0.37 ± 0.53	-1.51 ± 0.26
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-4.76 ± 0.88	-6.75 ± 0.67	0.18 ± 0.4	-0.81 ± 0.21
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-4.8 ± 0.79	-6.57 ± 0.63	0.09 ± 0.45	-1.14 ± 0.24
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-4.54 ± 0.74	-6.21 ± 0.59	0.25 ± 0.46	-1.25 ± 0.24
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-3.65 ± 0.62	-4.81 ± 0.35	-0.15 ± 0.23	-0.65 ± 0.14
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	-3.26 ± 0.53	-4.34 ± 0.33	-0.21 ± 0.27	-0.83 ± 0.17
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	-2.94 ± 0.49	-4.12 ± 0.3	-0.02 ± 0.29	-0.82 ± 0.17
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	-0.06 ± 0.32	-1.77 ± 0.18	0.56 ± 0.14	-0.12 ± 0.09
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	0.16 ± 0.32	-1.58 ± 0.19	0.67 ± 0.13	-0.09 ± 0.09
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	0.22 ± 0.32	-1.53 ± 0.19	0.84 ± 0.13	-0.02 ± 0.09

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	21.24 ± 1.13	-16.33 ± 0.49	4.03 ± 0.45	30.24 ± 0.89
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	21.49 ± 1.1	-16.46 ± 0.47	4.01 ± 0.44	30.03 ± 0.94
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	21.5 ± 1	-16.47 ± 0.45	4.07 ± 0.41	29.73 ± 0.94
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	20.22 ± 1.41	-15.81 ± 0.95	4.66 ± 0.46	27.06 ± 1.65
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	20.25 ± 1.37	-15.82 ± 0.95	4.82 ± 0.46	26.71 ± 1.64
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	20.12 ± 1.31	-15.76 ± 0.94	4.98 ± 0.44	26.36 ± 1.61
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	16.35 ± 0.62	-13.88 ± 0.59	8.06 ± 0.35	21.56 ± 0.9
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	15.56 ± 0.58	-13.48 ± 0.6	8.97 ± 0.4	20.76 ± 0.91
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$	15.12 ± 0.55	-13.26 ± 0.6	9.48 ± 0.39	20.32 ± 0.91
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	9.27 ± 0.34	-10.32 ± 0.51	18.23 ± 0.68	14.33 ± 0.72
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	8.64 ± 0.35	-10.01 ± 0.49	19.6 ± 0.77	13.7 ± 0.71
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	8.33 ± 0.34	-9.85 ± 0.48	20.3 ± 0.8	13.39 ± 0.71

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	5.7 ± 1.43	833.37 ± 90.3
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	8.77 ± 2.71	183.65 ± 22.78
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	12.6 ± 3.98	139.75 ± 16.94
30	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	2.82 ± 0.78	386.16 ± 31.5
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	4.16 ± 1.72	97.73 ± 11.16
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	5.94 ± 2.5	73.95 ± 9.82
20	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	1.62 ± 0.48	216.66 ± 11.66
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	2.63 ± 1.1	60.14 ± 7.5
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$	3.63 ± 1.59	44.5 ± 6.64
10	ALGO-Hybrid(S:FW A:FISTA) ($J = 1$)	1.18 ± 0.35	153.23 ± 6.79
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 10)$)	1.69 ± 0.71	43.94 ± 6.14
	ALGO-Hybrid(S:FW A:FISTA) ($J \sim \mathcal{U}(1, 20)$)	2.33 ± 1.02	32.51 ± 5.84

Table B.14: Average HU and HSR performance of ALGO-Hybrid (S: FW A: FISTA) on synthetic data. $N = 16$; $L = 100 \times 100$; 1000 trials.

B.2 Performance of ALGO Against the Choice of FOGMs

We group the results of ALGO into individual tables according to implementation (the inner iteration number J) and environment settings (model order N and SNR level). In each table we compare the FOGM used by ALGO and compare their competitiveness. The results shown from B.15 to B.22 are synthetic simulations with model order $N = 9$. The results shown from B.23 to B.30 are synthetic simulations with model order $N = 16$.

In general the performance of ALGO-FISTA is the better when ALGO is operating as inexact BCD with $J = 1$ (readers may refer to Table B.15, B.23). However, when $J = 100$, i.e., in the context of exact BCD, ALGO-PG significantly becomes the best (readers may refer to Table B.21, B.22, B.29, B.30).

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-6.4 ± 1.76	-8.13 ± 1.16	-3.81 ± 0.85	-3.66 ± 0.86
	ALGO-BBGP	-6.4 ± 1.67	-8.24 ± 1.12	-3.65 ± 0.83	-3.62 ± 0.83
	ALGO-PG	-6.57 ± 1.66	-8.23 ± 1.11	-4.27 ± 0.67	-3.98 ± 0.68
	ALGO-FISTA	-6.89 ± 1.99	-7.99 ± 1.17	-4.27 ± 0.82	-4.08 ± 0.84
	ALGO-FW	-5.82 ± 1.49	-8.13 ± 0.99	-1.94 ± 1	-3.32 ± 0.78
	ALGO-Hybrid (S: FW A:PG)	-5.92 ± 1.72	-7.93 ± 1.07	-2.72 ± 0.93	-3.14 ± 0.87
	ALGO-Hybrid (S: FW A:FISTA)	-5.79 ± 1.64	-6.94 ± 1.12	-2.74 ± 0.84	-3.75 ± 0.75
30	ALGO-GP	-6.28 ± 1.69	-8.02 ± 1.16	-3.54 ± 0.69	-3.36 ± 0.69
	ALGO-BBGP	-6.27 ± 1.59	-8.12 ± 1.11	-3.25 ± 0.65	-3.2 ± 0.64
	ALGO-PG	-6.32 ± 1.53	-8.04 ± 1.06	-3.97 ± 0.57	-3.66 ± 0.57
	ALGO-FISTA	-6.42 ± 1.71	-7.6 ± 1.08	-3.77 ± 0.62	-3.55 ± 0.61
	ALGO-FW	-5.85 ± 1.53	-7.98 ± 1.04	-1.71 ± 0.93	-3 ± 0.65
	ALGO-Hybrid (S: FW A:PG)	-5.75 ± 1.62	-7.76 ± 1.03	-2.32 ± 0.81	-2.73 ± 0.71
	ALGO-Hybrid (S: FW A:FISTA)	-5.7 ± 1.6	-6.92 ± 1.1	-2.7 ± 0.75	-3.33 ± 0.6

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	19.34 ± 1.42	-15.32 ± 0.65	4.55 ± 0.69	29.21 ± 0.92
	ALGO-BBGP	19.57 ± 1.41	-15.43 ± 0.75	4.38 ± 0.66	29.38 ± 0.95
	ALGO-PG	21.53 ± 1.72	-16.41 ± 0.62	3.82 ± 0.67	30.69 ± 1.29
	ALGO-FISTA	23.24 ± 2.37	-17.27 ± 1.03	3.41 ± 0.94	32.48 ± 2.3
	ALGO-FW	18.7 ± 1.39	-15 ± 0.81	4.9 ± 0.68	28.6 ± 0.91
	ALGO-Hybrid (S: FW A:PG)	18.21 ± 1.39	-14.75 ± 0.64	5.09 ± 0.75	28.1 ± 0.87
	ALGO-Hybrid (S: FW A:FISTA)	21.38 ± 1.98	-16.34 ± 0.95	3.99 ± 0.79	31.22 ± 1.64
30	ALGO-GP	18.85 ± 1.34	-15.08 ± 0.61	5.03 ± 0.67	26.81 ± 0.73
	ALGO-BBGP	19.03 ± 1.31	-15.17 ± 0.71	4.9 ± 0.63	26.93 ± 0.78
	ALGO-PG	20.48 ± 1.37	-15.89 ± 0.54	4.48 ± 0.6	27.88 ± 1.04
	ALGO-FISTA	21.57 ± 1.74	-16.44 ± 0.79	4.24 ± 0.79	28.84 ± 1.6
	ALGO-FW	18.43 ± 1.33	-14.87 ± 0.74	5.26 ± 0.67	26.62 ± 0.78
	ALGO-Hybrid (S: FW A:PG)	17.72 ± 1.28	-14.51 ± 0.6	5.61 ± 0.72	25.93 ± 0.68
	ALGO-Hybrid (S: FW A:FISTA)	20.18 ± 1.56	-15.75 ± 0.74	4.68 ± 0.7	28.01 ± 1.14

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	15.82 ± 4.55	1233.2 ± 246.73
	ALGO-BBGP	18.06 ± 4.82	1303.63 ± 238.66
	ALGO-PG	26.41 ± 5.75	3062.73 ± 282.5
	ALGO-FISTA	5.36 ± 1.3	550.78 ± 85.34
	ALGO-FW	14.12 ± 3.76	1368.58 ± 202.94
	ALGO-Hybrid (S: FW A:PG)	14.5 ± 5.7	2311.51 ± 310.8
	ALGO-Hybrid (S: FW A:FISTA)	7.75 ± 1.54	925.46 ± 183.12
30	ALGO-GP	6.21 ± 1.66	473.6 ± 86.34
	ALGO-BBGP	7.91 ± 2.01	558.76 ± 91.89
	ALGO-PG	12.94 ± 2.81	1456.56 ± 125.69
	ALGO-FISTA	3.02 ± 0.67	299.52 ± 26.05
	ALGO-FW	6.25 ± 1.66	591.12 ± 84.88
	ALGO-Hybrid (S: FW A:PG)	6.06 ± 2.31	960.04 ± 121.7
	ALGO-Hybrid (S: FW A:FISTA)	3.59 ± 0.95	448.93 ± 85.72

Table B.15: Average HU and HSR performance of ALGO (with $J = 1$) on synthetic data at 40dB and 30dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-5.65 ± 1.3	-6.75 ± 0.84	-2.95 ± 0.48	-2.73 ± 0.46
	ALGO-BBGP	-5.64 ± 1.2	-6.88 ± 0.79	-2.48 ± 0.47	-2.45 ± 0.43
	ALGO-PG	-5.51 ± 1.12	-6.65 ± 0.71	-3.28 ± 0.45	-2.97 ± 0.44
	ALGO-FISTA	-5 ± 1.24	-5.63 ± 0.71	-2.86 ± 0.44	-2.7 ± 0.42
	ALGO-FW	-5.25 ± 1.25	-7 ± 0.79	-1.5 ± 0.71	-2.37 ± 0.46
	ALGO-Hybrid (S: FW A:PG)	-5.32 ± 1.3	-6.8 ± 0.8	-1.87 ± 0.58	-2.11 ± 0.48
	ALGO-Hybrid (S: FW A:FISTA)	-4.72 ± 1.19	-5.48 ± 0.72	-2.46 ± 0.53	-2.6 ± 0.42
10	ALGO-GP	-2.23 ± 0.79	-3.36 ± 0.42	-1.28 ± 0.36	-1.43 ± 0.29
	ALGO-BBGP	-2.27 ± 0.71	-3.52 ± 0.37	-1 ± 0.31	-1.31 ± 0.25
	ALGO-PG	-2.93 ± 0.69	-3.96 ± 0.39	-1.33 ± 0.3	-1.4 ± 0.27
	ALGO-FISTA	-1 ± 0.71	-2.49 ± 0.31	-1.03 ± 0.3	-1.35 ± 0.25
	ALGO-FW	-2.66 ± 0.83	-4.11 ± 0.38	-1 ± 0.33	-1.37 ± 0.27
	ALGO-Hybrid (S: FW A:PG)	-2.93 ± 0.76	-4.15 ± 0.4	-0.88 ± 0.32	-1.18 ± 0.27
	ALGO-Hybrid (S: FW A:FISTA)	-0.98 ± 0.68	-2.42 ± 0.33	-1.52 ± 0.34	-1.63 ± 0.27

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	17.13 ± 0.87	-14.21 ± 0.58	6.69 ± 0.57	23.07 ± 0.67
	ALGO-BBGP	16.98 ± 0.85	-14.13 ± 0.6	6.79 ± 0.57	22.93 ± 0.73
	ALGO-PG	17.42 ± 0.75	-14.36 ± 0.54	6.56 ± 0.57	23.31 ± 0.85
	ALGO-FISTA	17.26 ± 0.76	-14.28 ± 0.61	7.16 ± 0.52	23.12 ± 1.02
	ALGO-FW	16.89 ± 0.91	-14.09 ± 0.62	6.83 ± 0.59	22.99 ± 0.75
	ALGO-Hybrid (S: FW A:PG)	16.45 ± 0.96	-13.87 ± 0.54	7.07 ± 0.66	22.55 ± 0.65
	ALGO-Hybrid (S: FW A:FISTA)	17.09 ± 0.78	-14.19 ± 0.59	7.19 ± 0.52	23.02 ± 0.88
10	ALGO-GP	11.36 ± 0.33	-11.34 ± 0.7	14.11 ± 0.54	16.96 ± 0.79
	ALGO-BBGP	11.29 ± 0.33	-11.31 ± 0.68	14.22 ± 0.54	16.89 ± 0.78
	ALGO-PG	11.85 ± 0.42	-11.58 ± 0.64	13.05 ± 0.62	17.43 ± 0.67
	ALGO-FISTA	10.06 ± 0.3	-10.69 ± 0.72	16.69 ± 0.6	15.63 ± 0.9
	ALGO-FW	11.88 ± 0.36	-11.6 ± 0.67	13.11 ± 0.53	17.5 ± 0.73
	ALGO-Hybrid (S: FW A:PG)	11.82 ± 0.41	-11.57 ± 0.64	13.09 ± 0.58	17.39 ± 0.67
	ALGO-Hybrid (S: FW A:FISTA)	10.67 ± 0.31	-10.99 ± 0.69	15.44 ± 0.58	16.21 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	2.32 ± 0.54	165.91 ± 21.13
	ALGO-BBGP	3.66 ± 0.84	245.73 ± 33.62
	ALGO-PG	5.94 ± 1.25	626.33 ± 58.86
	ALGO-FISTA	2.11 ± 0.42	195.63 ± 10.07
	ALGO-FW	2.78 ± 0.66	247.13 ± 31.71
	ALGO-Hybrid (S: FW A:PG)	2.64 ± 1	395.43 ± 46.69
	ALGO-Hybrid (S: FW A:FISTA)	1.3 ± 0.43	203.93 ± 20.73
10	ALGO-GP	1.76 ± 0.4	124.84 ± 11.45
	ALGO-BBGP	2.04 ± 0.49	136.03 ± 17.53
	ALGO-PG	2.93 ± 0.67	303.65 ± 30.63
	ALGO-FISTA	1.63 ± 0.35	148.76 ± 12.72
	ALGO-FW	1.93 ± 0.45	169.92 ± 15.76
	ALGO-Hybrid (S: FW A:PG)	1.67 ± 0.6	234.83 ± 29.38
	ALGO-Hybrid (S: FW A:FISTA)	1.01 ± 0.32	146.19 ± 13.65

Table B.16: Average HU and HSR performance of ALGO (with $J = 1$) on synthetic data at 20dB and 10dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-6.54 ± 1.83	-8.11 ± 1.19	-3.84 ± 0.94	-3.72 ± 0.95
	ALGO-BBGP	-6.46 ± 1.69	-8.2 ± 1.13	-3.77 ± 0.83	-3.72 ± 0.83
	ALGO-PG	-6.5 ± 1.71	-8.11 ± 1.12	-4.31 ± 0.69	-4.03 ± 0.7
	ALGO-FISTA	-6.94 ± 1.99	-7.95 ± 1.19	-4.35 ± 0.85	-4.17 ± 0.87
	ALGO-FW	-5.58 ± 1.81	-8.11 ± 1.1	-2.5 ± 1.06	-3.5 ± 0.94
30	ALGO-GP	-6.29 ± 1.73	-8.01 ± 1.19	-3.46 ± 0.71	-3.32 ± 0.71
	ALGO-BBGP	-6.31 ± 1.63	-8.09 ± 1.17	-3.33 ± 0.68	-3.27 ± 0.66
	ALGO-PG	-6.33 ± 1.6	-7.99 ± 1.11	-3.97 ± 0.57	-3.66 ± 0.57
	ALGO-FISTA	-6.21 ± 1.71	-7.19 ± 1.02	-3.86 ± 0.63	-3.65 ± 0.62
	ALGO-FW	-5.64 ± 1.72	-7.94 ± 1.11	-2.26 ± 0.91	-3.16 ± 0.75

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	19.36 ± 1.43	-15.33 ± 0.67	4.54 ± 0.7	29.37 ± 0.98
	ALGO-BBGP	19.48 ± 1.44	-15.39 ± 0.75	4.43 ± 0.69	29.47 ± 0.97
	ALGO-PG	21.67 ± 1.81	-16.48 ± 0.65	3.72 ± 0.69	31.22 ± 1.33
	ALGO-FISTA	22.91 ± 2.1	-17.1 ± 0.89	3.52 ± 0.78	32.31 ± 1.73
	ALGO-FW	18.51 ± 1.71	-14.9 ± 0.88	4.93 ± 0.8	28.77 ± 1.06
30	ALGO-GP	18.84 ± 1.34	-15.07 ± 0.61	5.02 ± 0.69	26.9 ± 0.75
	ALGO-BBGP	18.99 ± 1.37	-15.15 ± 0.69	4.92 ± 0.69	27.02 ± 0.77
	ALGO-PG	20.79 ± 1.54	-16.05 ± 0.57	4.27 ± 0.65	28.31 ± 1.06
	ALGO-FISTA	21.21 ± 1.53	-16.26 ± 0.67	4.44 ± 0.7	28.52 ± 1.23
	ALGO-FW	18.26 ± 1.56	-14.79 ± 0.78	5.32 ± 0.77	26.61 ± 0.77

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	35.67 ± 11.22	673.81 ± 149.42
	ALGO-BBGP	40 ± 11.75	697.8 ± 139.47
	ALGO-PG	43.46 ± 11.39	1300.47 ± 163
	ALGO-FISTA	6.88 ± 2.04	198.9 ± 38.23
	ALGO-FW	34.32 ± 9.89	826.36 ± 130.66
30	ALGO-GP	12.85 ± 3.31	244.16 ± 34.23
	ALGO-BBGP	14.73 ± 3.74	261.58 ± 34.84
	ALGO-PG	18.79 ± 4.53	563.39 ± 48.39
	ALGO-FISTA	3.56 ± 0.88	103.62 ± 11.65
	ALGO-FW	12.64 ± 3.51	308.62 ± 48.58

Table B.17: Average HU and HSR performance of ALGO (with $J = 5$) on synthetic data at 40dB and 30dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-5.61 ± 1.32	-6.54 ± 0.83	-2.86 ± 0.48	-2.69 ± 0.46
	ALGO-BBGP	-5.47 ± 1.24	-6.6 ± 0.78	-2.6 ± 0.47	-2.55 ± 0.43
	ALGO-PG	-5.5 ± 1.21	-6.54 ± 0.75	-3.19 ± 0.44	-2.91 ± 0.43
	ALGO-FISTA	-4.05 ± 1.12	-4.76 ± 0.6	-2.82 ± 0.44	-2.69 ± 0.42
	ALGO-FW	-4.96 ± 1.35	-6.73 ± 0.79	-2 ± 0.63	-2.53 ± 0.47
10	ALGO-GP	-1.65 ± 0.71	-2.81 ± 0.34	-1.28 ± 0.31	-1.45 ± 0.27
	ALGO-BBGP	-1.45 ± 0.67	-2.84 ± 0.31	-1.09 ± 0.3	-1.37 ± 0.25
	ALGO-PG	-2.36 ± 0.71	-3.44 ± 0.37	-1.27 ± 0.29	-1.41 ± 0.26
	ALGO-FISTA	-0.09 ± 0.73	-2.09 ± 0.31	-1.05 ± 0.3	-1.36 ± 0.25
	ALGO-FW	-1.75 ± 0.77	-3.23 ± 0.34	-1.25 ± 0.31	-1.49 ± 0.28

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	17.03 ± 0.88	-14.16 ± 0.58	6.82 ± 0.56	23.03 ± 0.7
	ALGO-BBGP	17 ± 0.89	-14.15 ± 0.6	6.87 ± 0.57	23.01 ± 0.75
	ALGO-PG	17.79 ± 0.85	-14.54 ± 0.53	6.38 ± 0.56	23.61 ± 0.87
	ALGO-FISTA	16.39 ± 0.61	-13.84 ± 0.63	8.09 ± 0.49	22.31 ± 0.98
	ALGO-FW	16.78 ± 0.98	-14.04 ± 0.62	7.04 ± 0.61	22.94 ± 0.72
10	ALGO-GP	10.69 ± 0.26	-11 ± 0.74	15.44 ± 0.49	16.27 ± 0.87
	ALGO-BBGP	10.55 ± 0.26	-10.93 ± 0.73	15.7 ± 0.48	16.12 ± 0.88
	ALGO-PG	11.42 ± 0.35	-11.37 ± 0.67	13.99 ± 0.58	17.02 ± 0.77
	ALGO-FISTA	9.53 ± 0.26	-10.42 ± 0.7	17.79 ± 0.51	15.07 ± 0.91
	ALGO-FW	11.1 ± 0.29	-11.21 ± 0.7	14.62 ± 0.49	16.68 ± 0.81

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	5.42 ± 1.29	103.28 ± 9.86
	ALGO-BBGP	6.52 ± 1.59	115.68 ± 13.43
	ALGO-PG	8.4 ± 2.02	254.55 ± 19.07
	ALGO-FISTA	2.46 ± 0.57	71.22 ± 5.47
	ALGO-FW	5.21 ± 1.35	126.8 ± 14.12
10	ALGO-GP	3.26 ± 0.81	62.09 ± 6.09
	ALGO-BBGP	4.3 ± 1.03	75.21 ± 8.62
	ALGO-PG	5.05 ± 1.33	153.8 ± 15.13
	ALGO-FISTA	2.03 ± 0.52	58.84 ± 7.45
	ALGO-FW	4.06 ± 0.99	98.08 ± 7.28

Table B.18: Average HU and HSR performance of ALGO (with $J = 5$) on synthetic data at 20dB and 10dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-6.37 ± 1.8	-8.13 ± 1.17	-3.85 ± 0.9	-3.75 ± 0.91
	ALGO-BBGP	-6.43 ± 1.7	-8.06 ± 1.15	-3.75 ± 0.86	-3.74 ± 0.88
	ALGO-PG	-6.47 ± 1.74	-8.05 ± 1.11	-4.33 ± 0.71	-4.05 ± 0.72
	ALGO-FISTA	-6.8 ± 1.95	-7.96 ± 1.21	-4.29 ± 0.87	-4.16 ± 0.89
	ALGO-FW	-5.62 ± 1.89	-8.08 ± 1.14	-2.66 ± 1.12	-3.57 ± 0.99
30	ALGO-GP	-6.11 ± 1.71	-7.98 ± 1.15	-3.42 ± 0.7	-3.32 ± 0.7
	ALGO-BBGP	-6.23 ± 1.65	-7.92 ± 1.18	-3.25 ± 0.68	-3.23 ± 0.66
	ALGO-PG	-6.29 ± 1.61	-7.92 ± 1.11	-3.97 ± 0.58	-3.66 ± 0.58
	ALGO-FISTA	-6.06 ± 1.66	-7.05 ± 0.99	-3.82 ± 0.66	-3.65 ± 0.65
	ALGO-FW	-5.59 ± 1.76	-7.89 ± 1.14	-2.35 ± 0.98	-3.2 ± 0.79

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	19.49 ± 1.43	-15.39 ± 0.64	4.49 ± 0.71	29.47 ± 1.05
	ALGO-BBGP	19.42 ± 1.41	-15.36 ± 0.72	4.46 ± 0.69	29.43 ± 1
	ALGO-PG	21.67 ± 1.83	-16.48 ± 0.66	3.71 ± 0.7	31.33 ± 1.34
	ALGO-FISTA	22.58 ± 2.11	-16.94 ± 0.95	3.64 ± 0.81	32.04 ± 1.78
	ALGO-FW	18.56 ± 1.9	-14.93 ± 0.96	4.88 ± 0.85	28.89 ± 1.22
30	ALGO-GP	18.86 ± 1.36	-15.08 ± 0.58	5.03 ± 0.7	26.92 ± 0.8
	ALGO-BBGP	18.85 ± 1.36	-15.08 ± 0.66	5 ± 0.71	26.93 ± 0.78
	ALGO-PG	20.81 ± 1.58	-16.06 ± 0.58	4.25 ± 0.67	28.41 ± 1.06
	ALGO-FISTA	20.85 ± 1.53	-16.08 ± 0.7	4.62 ± 0.72	28.21 ± 1.2
	ALGO-FW	18.38 ± 1.71	-14.85 ± 0.81	5.27 ± 0.8	26.73 ± 0.88

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	50.87 ± 16.77	493.34 ± 128.56
	ALGO-BBGP	61.17 ± 19.16	541.8 ± 123.03
	ALGO-PG	60.14 ± 16.69	918.78 ± 143.05
	ALGO-FISTA	9.24 ± 3.07	138.41 ± 32.93
	ALGO-FW	54.18 ± 15.78	662.62 ± 111.2
30	ALGO-GP	18.95 ± 4.97	180.56 ± 25.35
	ALGO-BBGP	22.58 ± 5.67	202.9 ± 26.3
	ALGO-PG	24.68 ± 5.96	380.79 ± 34.18
	ALGO-FISTA	4.58 ± 1.19	69.3 ± 9.48
	ALGO-FW	18.85 ± 5.21	234.83 ± 35.44

Table B.19: Average HU and HSR performance of ALGO (with $J = 10$) on synthetic data at 40dB and 30dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-5.17 ± 1.28	-6.24 ± 0.75	-2.81 ± 0.47	-2.69 ± 0.44
	ALGO-BBGP	-5.3 ± 1.23	-6.35 ± 0.79	-2.52 ± 0.47	-2.5 ± 0.42
	ALGO-PG	-5.43 ± 1.24	-6.42 ± 0.75	-3.17 ± 0.44	-2.9 ± 0.43
	ALGO-FISTA	-3.76 ± 1.11	-4.56 ± 0.58	-2.77 ± 0.46	-2.66 ± 0.43
	ALGO-FW	-4.84 ± 1.3	-6.47 ± 0.77	-2.07 ± 0.64	-2.54 ± 0.48
10	ALGO-GP	-1.07 ± 0.68	-2.38 ± 0.33	-1.22 ± 0.31	-1.39 ± 0.26
	ALGO-BBGP	-1.22 ± 0.66	-2.58 ± 0.31	-1.11 ± 0.29	-1.36 ± 0.25
	ALGO-PG	-2.06 ± 0.7	-3.18 ± 0.35	-1.26 ± 0.3	-1.43 ± 0.26
	ALGO-FISTA	-0.05 ± 0.71	-1.96 ± 0.32	-1.11 ± 0.28	-1.36 ± 0.25
	ALGO-FW	-1.41 ± 0.73	-2.94 ± 0.33	-1.26 ± 0.31	-1.49 ± 0.27

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	16.8 ± 0.83	-14.05 ± 0.58	7.12 ± 0.52	22.83 ± 0.72
	ALGO-BBGP	16.78 ± 0.86	-14.04 ± 0.6	7.08 ± 0.56	22.84 ± 0.76
	ALGO-PG	17.78 ± 0.86	-14.54 ± 0.53	6.42 ± 0.56	23.62 ± 0.87
	ALGO-FISTA	16.05 ± 0.63	-13.67 ± 0.65	8.43 ± 0.53	22 ± 0.96
	ALGO-FW	16.77 ± 0.96	-14.03 ± 0.62	7.16 ± 0.59	22.93 ± 0.73
10	ALGO-GP	10.11 ± 0.25	-10.72 ± 0.74	16.61 ± 0.49	15.68 ± 0.91
	ALGO-BBGP	10.25 ± 0.24	-10.79 ± 0.74	16.32 ± 0.47	15.82 ± 0.89
	ALGO-PG	11.14 ± 0.32	-11.23 ± 0.69	14.55 ± 0.55	16.73 ± 0.81
	ALGO-FISTA	9.37 ± 0.27	-10.34 ± 0.68	18.14 ± 0.55	14.91 ± 0.89
	ALGO-FW	10.74 ± 0.26	-11.03 ± 0.71	15.32 ± 0.46	16.31 ± 0.85

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	9.32 ± 2.3	90 ± 10.86
	ALGO-BBGP	11.71 ± 2.69	104.36 ± 8.82
	ALGO-PG	11.63 ± 2.76	179.68 ± 13.49
	ALGO-FISTA	3.23 ± 0.79	48.67 ± 5.11
	ALGO-FW	8.12 ± 2.02	101.2 ± 10.17
10	ALGO-GP	5.21 ± 1.47	50.62 ± 8.73
	ALGO-BBGP	7.81 ± 1.89	69.14 ± 8.14
	ALGO-PG	7.64 ± 1.94	117.67 ± 11.11
	ALGO-FISTA	2.73 ± 0.74	41.55 ± 6.18
	ALGO-FW	6.24 ± 1.48	76.62 ± 5.96

Table B.20: Average HU and HSR performance of ALGO (with $J = 10$) on synthetic data at 20dB and 10dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-5.85 ± 1.57	-7.46 ± 1.1	-2.75 ± 0.9	-3.4 ± 0.85
	ALGO-BBGP	-5.92 ± 1.56	-7.47 ± 1.1	-2.75 ± 0.9	-3.43 ± 0.83
	ALGO-PG	-6.2 ± 1.72	-7.75 ± 1.06	-4.28 ± 0.76	-4.01 ± 0.77
	ALGO-FISTA	-5.72 ± 1.75	-7.4 ± 1.18	-2.76 ± 0.85	-3.42 ± 0.84
	ALGO-FW	-5.79 ± 1.97	-8.2 ± 1.23	-2.68 ± 1.11	-3.9 ± 1.01
30	ALGO-GP	-5.56 ± 1.41	-7.14 ± 1.03	-2.45 ± 0.76	-3.06 ± 0.66
	ALGO-BBGP	-5.67 ± 1.38	-7.2 ± 1.02	-2.44 ± 0.76	-3.09 ± 0.67
	ALGO-PG	-5.87 ± 1.54	-7.51 ± 1.02	-3.88 ± 0.62	-3.6 ± 0.61
	ALGO-FISTA	-4.88 ± 1.51	-6.6 ± 0.97	-2.42 ± 0.76	-3.07 ± 0.67
	ALGO-FW	-5.51 ± 1.83	-7.86 ± 1.19	-2.36 ± 0.99	-3.42 ± 0.79

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	19.24 ± 1.72	-15.27 ± 0.89	4.68 ± 0.76	29.24 ± 1.34
	ALGO-BBGP	19.36 ± 1.67	-15.33 ± 0.89	4.66 ± 0.75	29.32 ± 1.3
	ALGO-PG	21.16 ± 1.82	-16.23 ± 0.7	3.92 ± 0.76	31 ± 1.44
	ALGO-FISTA	20.4 ± 2.44	-15.85 ± 1.34	4.61 ± 1.12	30.37 ± 2.03
	ALGO-FW	20.33 ± 2.29	-15.81 ± 1.12	4.25 ± 0.86	30.41 ± 1.6
30	ALGO-GP	18.64 ± 1.49	-14.97 ± 0.76	5.35 ± 0.72	26.62 ± 1.01
	ALGO-BBGP	18.77 ± 1.46	-15.04 ± 0.76	5.32 ± 0.71	26.73 ± 0.98
	ALGO-PG	20.3 ± 1.58	-15.8 ± 0.6	4.51 ± 0.71	28.11 ± 1.07
	ALGO-FISTA	19.05 ± 1.83	-15.18 ± 1.07	5.58 ± 0.95	26.85 ± 1.41
	ALGO-FW	19.86 ± 1.78	-15.58 ± 0.84	4.82 ± 0.75	27.81 ± 1.11

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	322.16 ± 109.76	298.53 ± 72.13
	ALGO-BBGP	352.19 ± 119.94	297.17 ± 70.9
	ALGO-PG	215.94 ± 75.76	324.91 ± 82.85
	ALGO-FISTA	128.12 ± 40.99	191.93 ± 41.84
	ALGO-FW	279.61 ± 90.41	324.34 ± 58.81
30	ALGO-GP	121.26 ± 35.84	110.39 ± 17.02
	ALGO-BBGP	133.94 ± 39.43	111.88 ± 16.49
	ALGO-PG	81.49 ± 22.71	122.13 ± 16.3
	ALGO-FISTA	53.97 ± 15.75	79.78 ± 13.29
	ALGO-FW	105.98 ± 33.02	120.38 ± 14.36

Table B.21: Average HU and HSR performance of ALGO (with $J = 100$) on synthetic data at 40dB and 30dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-3.96 ± 1.02	-5.1 ± 0.62	-1.98 ± 0.55	-2.4 ± 0.43
	ALGO-BBGP	-4.18 ± 1.02	-5.26 ± 0.62	-1.99 ± 0.54	-2.4 ± 0.42
	ALGO-PG	-4.63 ± 1.15	-5.6 ± 0.66	-3.04 ± 0.44	-2.81 ± 0.42
	ALGO-FISTA	-2.95 ± 1.05	-4.68 ± 0.58	-1.74 ± 0.58	-2.34 ± 0.44
	ALGO-FW	-4.07 ± 1.19	-5.72 ± 0.67	-2.07 ± 0.64	-2.63 ± 0.48
10	ALGO-GP	-0.13 ± 0.63	-1.98 ± 0.31	-1 ± 0.27	-1.3 ± 0.24
	ALGO-BBGP	-0.2 ± 0.63	-2 ± 0.31	-1 ± 0.26	-1.3 ± 0.24
	ALGO-PG	-0.81 ± 0.68	-2.32 ± 0.3	-1.16 ± 0.28	-1.38 ± 0.25
	ALGO-FISTA	0.01 ± 0.64	-1.92 ± 0.32	-0.93 ± 0.26	-1.26 ± 0.24
	ALGO-FW	-0.49 ± 0.64	-2.24 ± 0.31	-1.19 ± 0.29	-1.43 ± 0.26

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	16.27 ± 0.69	-13.78 ± 0.6	8.17 ± 0.51	22.21 ± 0.97
	ALGO-BBGP	16.38 ± 0.7	-13.84 ± 0.59	8.03 ± 0.5	22.33 ± 0.96
	ALGO-PG	16.95 ± 0.77	-14.12 ± 0.58	7.25 ± 0.51	22.93 ± 0.87
	ALGO-FISTA	15.51 ± 0.8	-13.4 ± 0.69	8.99 ± 0.68	21.47 ± 1.05
	ALGO-FW	16.79 ± 0.71	-14.04 ± 0.63	7.62 ± 0.49	22.7 ± 0.93
10	ALGO-GP	9.17 ± 0.29	-10.24 ± 0.66	18.6 ± 0.6	14.72 ± 0.89
	ALGO-BBGP	9.23 ± 0.28	-10.27 ± 0.67	18.47 ± 0.57	14.78 ± 0.89
	ALGO-PG	9.92 ± 0.25	-10.62 ± 0.71	16.99 ± 0.49	15.48 ± 0.9
	ALGO-FISTA	8.99 ± 0.33	-10.16 ± 0.64	18.98 ± 0.7	14.55 ± 0.87
	ALGO-FW	9.72 ± 0.26	-10.52 ± 0.68	17.4 ± 0.5	15.26 ± 0.89

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	58.3 ± 17.71	52.73 ± 6.87
	ALGO-BBGP	69.44 ± 19.71	57.3 ± 6.19
	ALGO-PG	47.69 ± 12.47	70.86 ± 5.4
	ALGO-FISTA	29.77 ± 8.7	43.6 ± 5.87
	ALGO-FW	53.2 ± 16.14	60.19 ± 5.32
10	ALGO-GP	49.51 ± 15.78	43.49 ± 7.73
	ALGO-BBGP	54.66 ± 16.97	44.92 ± 7.51
	ALGO-PG	34.79 ± 10.16	51.78 ± 6.86
	ALGO-FISTA	28.08 ± 8.71	40.89 ± 7.36
	ALGO-FW	43.15 ± 14.27	48.27 ± 7.2

Table B.22: Average HU and HSR performance of ALGO (with $J = 100$) on synthetic data at 20dB and 10dB SNR; $N = 9$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-5.21 ± 0.97	-7.69 ± 0.6	-2.12 ± 0.35	-2.07 ± 0.3
	ALGO-BBGP	-5.29 ± 0.88	-7.73 ± 0.58	-1.87 ± 0.32	-1.95 ± 0.28
	ALGO-PG	-5.59 ± 0.94	-7.93 ± 0.64	-3.22 ± 0.26	-2.8 ± 0.25
	ALGO-FISTA	-5.89 ± 1.07	-7.94 ± 0.67	-3 ± 0.28	-2.68 ± 0.27
	ALGO-FW	-5.02 ± 0.78	-7.43 ± 0.53	0.41 ± 0.54	-1.44 ± 0.24
	ALGO-Hybrid (S: FW A:PG)	-4.53 ± 0.8	-7.19 ± 0.54	0.24 ± 0.34	-0.71 ± 0.24
	ALGO-Hybrid (S: FW A:FISTA)	-4.78 ± 0.97	-6.96 ± 0.61	0.26 ± 0.45	-1.11 ± 0.24
30	ALGO-GP	-5.14 ± 0.9	-7.48 ± 0.56	-1.89 ± 0.34	-1.85 ± 0.27
	ALGO-BBGP	-5.21 ± 0.83	-7.48 ± 0.67	-1.54 ± 0.3	-1.66 ± 0.25
	ALGO-PG	-5.43 ± 0.86	-7.61 ± 0.54	-3.14 ± 0.24	-2.71 ± 0.23
	ALGO-FISTA	-5.5 ± 0.95	-7.28 ± 0.56	-2.66 ± 0.25	-2.35 ± 0.23
	ALGO-FW	-4.95 ± 0.74	-7.26 ± 0.55	0.69 ± 0.5	-1.12 ± 0.21
	ALGO-Hybrid (S: FW A:PG)	-4.53 ± 0.82	-7.02 ± 0.64	0.68 ± 0.3	-0.37 ± 0.21
	ALGO-Hybrid (S: FW A:FISTA)	-4.76 ± 0.88	-6.75 ± 0.67	0.18 ± 0.4	-0.81 ± 0.21

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	20.21 ± 0.9	-15.82 ± 0.38	4.22 ± 0.4	28.9 ± 0.56
	ALGO-BBGP	20.44 ± 0.82	-15.93 ± 0.4	4.1 ± 0.36	29.13 ± 0.58
	ALGO-PG	22.3 ± 0.98	-16.86 ± 0.35	3.58 ± 0.37	30.57 ± 0.83
	ALGO-FISTA	24.18 ± 1.47	-17.8 ± 0.64	3.07 ± 0.54	32.22 ± 1.55
	ALGO-FW	19.14 ± 0.77	-15.28 ± 0.44	4.7 ± 0.37	27.95 ± 0.51
	ALGO-Hybrid (S: FW A:PG)	19.15 ± 0.83	-15.29 ± 0.36	4.69 ± 0.42	28.02 ± 0.54
	ALGO-Hybrid (S: FW A:FISTA)	21.24 ± 1.13	-16.33 ± 0.49	4.03 ± 0.45	30.24 ± 0.89
30	ALGO-GP	19.78 ± 0.81	-15.61 ± 0.34	4.58 ± 0.38	26.79 ± 0.46
	ALGO-BBGP	19.86 ± 1.3	-15.63 ± 0.93	4.5 ± 0.42	26.81 ± 1.55
	ALGO-PG	21.16 ± 0.75	-16.3 ± 0.3	4.18 ± 0.31	27.81 ± 0.64
	ALGO-FISTA	22.31 ± 1.17	-16.87 ± 0.58	3.93 ± 0.48	28.49 ± 1.17
	ALGO-FW	18.83 ± 0.93	-15.12 ± 0.63	5.02 ± 0.39	26.02 ± 0.94
	ALGO-Hybrid (S: FW A:PG)	18.78 ± 1.28	-15.09 ± 0.89	5 ± 0.48	26.01 ± 1.5
	ALGO-Hybrid (S: FW A:FISTA)	20.22 ± 1.41	-15.81 ± 0.95	4.66 ± 0.46	27.06 ± 1.65

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	13.5 ± 5.85	878.83 ± 133.88
	ALGO-BBGP	17.11 ± 7.58	1053.71 ± 109.26
	ALGO-PG	36.28 ± 14.9	3150.36 ± 203.2
	ALGO-FISTA	6.46 ± 2.86	495 ± 21.42
	ALGO-FW	10.25 ± 4.93	928.87 ± 67.31
	ALGO-Hybrid (S: FW A:PG)	13.11 ± 3.3	1903.32 ± 168.58
	ALGO-Hybrid (S: FW A:FISTA)	5.7 ± 1.43	833.37 ± 90.3
30	ALGO-GP	6.87 ± 3.2	426.98 ± 59.75
	ALGO-BBGP	8.7 ± 4.08	512.71 ± 47.98
	ALGO-PG	15.95 ± 7.2	1329.99 ± 110.83
	ALGO-FISTA	4.41 ± 2.14	333.85 ± 7.3
	ALGO-FW	4.43 ± 2.2	383.74 ± 25.04
	ALGO-Hybrid (S: FW A:PG)	5.83 ± 1.64	784.67 ± 60.74
	ALGO-Hybrid (S: FW A:FISTA)	2.82 ± 0.78	386.16 ± 31.5

Table B.23: Average HU and HSR performance of ALGO (with $J = 1$) on synthetic data at 40dB and 30dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-4.62 ± 0.69	-6.01 ± 0.4	-1.18 ± 0.26	-1.22 ± 0.19
	ALGO-BBGP	-4.74 ± 0.62	-6.22 ± 0.38	-0.52 ± 0.22	-0.87 ± 0.16
	ALGO-PG	-4.66 ± 0.63	-6.1 ± 0.34	-2.44 ± 0.17	-2.06 ± 0.16
	ALGO-FISTA	-3.76 ± 0.65	-4.81 ± 0.33	-1.41 ± 0.25	-1.36 ± 0.2
	ALGO-FW	-4.36 ± 0.65	-6.21 ± 0.39	0.52 ± 0.31	-0.64 ± 0.15
	ALGO-Hybrid (S: FW A:PG)	-4.47 ± 0.7	-6.31 ± 0.46	0.96 ± 0.21	-0.09 ± 0.15
	ALGO-Hybrid (S: FW A:FISTA)	-3.65 ± 0.62	-4.81 ± 0.35	-0.15 ± 0.23	-0.65 ± 0.14
10	ALGO-GP	-1.31 ± 0.36	-2.58 ± 0.2	0.75 ± 0.22	-0.04 ± 0.11
	ALGO-BBGP	-1.37 ± 0.31	-2.68 ± 0.18	1.14 ± 0.14	0.1 ± 0.09
	ALGO-PG	-2.09 ± 0.34	-3.23 ± 0.19	-0.26 ± 0.12	-0.52 ± 0.1
	ALGO-FISTA	-0.19 ± 0.32	-1.85 ± 0.16	1.03 ± 0.11	0.06 ± 0.08
	ALGO-FW	-1.64 ± 0.36	-3.13 ± 0.2	0.9 ± 0.12	-0.02 ± 0.09
	ALGO-Hybrid (S: FW A:PG)	-2.39 ± 0.4	-3.63 ± 0.23	1.55 ± 0.13	0.3 ± 0.09
	ALGO-Hybrid (S: FW A:FISTA)	-0.06 ± 0.32	-1.77 ± 0.18	0.56 ± 0.14	-0.12 ± 0.09

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	17.44 ± 0.45	-14.43 ± 0.36	6.64 ± 0.28	22.66 ± 0.46
	ALGO-BBGP	17.3 ± 0.41	-14.36 ± 0.38	6.74 ± 0.27	22.52 ± 0.47
	ALGO-PG	18.3 ± 0.46	-14.86 ± 0.33	5.97 ± 0.32	23.48 ± 0.54
	ALGO-FISTA	16.65 ± 0.28	-14.04 ± 0.41	7.87 ± 0.26	21.82 ± 0.6
	ALGO-FW	16.83 ± 0.45	-14.12 ± 0.38	7.1 ± 0.29	22.14 ± 0.43
	ALGO-Hybrid (S: FW A:PG)	17.06 ± 0.73	-14.24 ± 0.56	6.8 ± 0.38	22.34 ± 0.84
	ALGO-Hybrid (S: FW A:FISTA)	16.35 ± 0.62	-13.88 ± 0.59	8.06 ± 0.35	21.56 ± 0.9
10	ALGO-GP	10.1 ± 0.16	-10.74 ± 0.47	16.58 ± 0.32	15.2 ± 0.58
	ALGO-BBGP	9.87 ± 0.34	-10.62 ± 0.56	17.01 ± 0.6	14.98 ± 0.73
	ALGO-PG	11.85 ± 0.23	-11.61 ± 0.41	13.23 ± 0.33	17.09 ± 0.47
	ALGO-FISTA	8.67 ± 0.16	-10.03 ± 0.41	19.6 ± 0.36	13.75 ± 0.57
	ALGO-FW	10.7 ± 0.38	-11.04 ± 0.54	15.31 ± 0.56	15.85 ± 0.71
	ALGO-Hybrid (S: FW A:PG)	11.05 ± 0.39	-11.21 ± 0.54	14.7 ± 0.54	16.19 ± 0.71
	ALGO-Hybrid (S: FW A:FISTA)	9.27 ± 0.34	-10.32 ± 0.51	18.23 ± 0.68	14.33 ± 0.72

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	3.22 ± 1.55	178.06 ± 17.08
	ALGO-BBGP	4.79 ± 2.26	253.77 ± 18.48
	ALGO-PG	9.99 ± 4.4	758.43 ± 26.36
	ALGO-FISTA	3.57 ± 1.66	236.07 ± 5.91
	ALGO-FW	2.71 ± 1.42	205.78 ± 8.34
	ALGO-Hybrid (S: FW A:PG)	2.92 ± 0.87	375.19 ± 21.76
	ALGO-Hybrid (S: FW A:FISTA)	1.62 ± 0.48	216.66 ± 11.66
10	ALGO-GP	2.38 ± 1.01	134.34 ± 10.23
	ALGO-BBGP	2.54 ± 1.13	136.7 ± 11.42
	ALGO-PG	4.92 ± 2.16	369.88 ± 9.45
	ALGO-FISTA	2.51 ± 1.05	164.86 ± 4.92
	ALGO-FW	2.14 ± 1.05	167.31 ± 5.03
	ALGO-Hybrid (S: FW A:PG)	2.11 ± 0.62	263.95 ± 13.04
	ALGO-Hybrid (S: FW A:FISTA)	1.18 ± 0.35	153.23 ± 6.79

Table B.24: Average HU and HSR performance of ALGO (with $J = 1$) on synthetic data at 20dB and 10dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-5.29 ± 0.98	-7.68 ± 0.59	-2.03 ± 0.33	-2.01 ± 0.3
	ALGO-BBGP	-5.29 ± 0.88	-7.73 ± 0.58	-1.96 ± 0.33	-1.99 ± 0.29
	ALGO-PG	-5.57 ± 0.95	-7.87 ± 0.64	-3.18 ± 0.26	-2.77 ± 0.25
	ALGO-FISTA	-5.91 ± 1.06	-7.95 ± 0.67	-2.93 ± 0.28	-2.62 ± 0.27
	ALGO-FW	-4.91 ± 0.83	-7.38 ± 0.55	0.02 ± 0.46	-1.42 ± 0.26
30	ALGO-GP	-5.21 ± 0.87	-7.46 ± 0.55	-1.7 ± 0.29	-1.72 ± 0.24
	ALGO-BBGP	-5.23 ± 0.85	-7.48 ± 0.68	-1.61 ± 0.3	-1.68 ± 0.26
	ALGO-PG	-5.4 ± 0.87	-7.51 ± 0.55	-2.99 ± 0.22	-2.57 ± 0.21
	ALGO-FISTA	-5.39 ± 0.91	-7.1 ± 0.51	-2.46 ± 0.23	-2.2 ± 0.22
	ALGO-FW	-4.87 ± 0.77	-7.21 ± 0.57	0.2 ± 0.4	-1.15 ± 0.22

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	20.11 ± 0.83	-15.77 ± 0.39	4.28 ± 0.38	28.78 ± 0.55
	ALGO-BBGP	20.31 ± 0.84	-15.87 ± 0.43	4.16 ± 0.37	28.98 ± 0.55
	ALGO-PG	22.42 ± 1.02	-16.92 ± 0.35	3.52 ± 0.38	30.9 ± 0.84
	ALGO-FISTA	24.24 ± 1.16	-17.83 ± 0.46	3.05 ± 0.37	32.23 ± 1
	ALGO-FW	18.87 ± 0.87	-15.15 ± 0.44	4.8 ± 0.41	27.7 ± 0.48
30	ALGO-GP	19.68 ± 0.75	-15.55 ± 0.36	4.67 ± 0.36	26.6 ± 0.44
	ALGO-BBGP	19.86 ± 1.32	-15.63 ± 0.94	4.53 ± 0.42	26.71 ± 1.53
	ALGO-PG	21.54 ± 0.84	-16.48 ± 0.31	3.98 ± 0.35	28.24 ± 0.66
	ALGO-FISTA	22.04 ± 0.73	-16.74 ± 0.35	4.11 ± 0.29	28.11 ± 0.7
	ALGO-FW	18.63 ± 0.98	-15.02 ± 0.62	5.15 ± 0.41	25.8 ± 0.9

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	36.52 ± 14.08	383.95 ± 41.03
	ALGO-BBGP	40.33 ± 15.9	395.03 ± 46.67
	ALGO-PG	82.03 ± 30.03	1265.71 ± 87.77
	ALGO-FISTA	10.31 ± 3.66	152.81 ± 7.52
	ALGO-FW	34.17 ± 14.66	450.45 ± 33.36
30	ALGO-GP	17.99 ± 6.83	193.5 ± 15.97
	ALGO-BBGP	18.87 ± 7.71	193.52 ± 20.52
	ALGO-PG	44.11 ± 15.67	691.99 ± 19.99
	ALGO-FISTA	7.47 ± 2.93	111.15 ± 4.47
	ALGO-FW	14.14 ± 5.9	196.51 ± 9.89

Table B.25: Average HU and HSR performance of ALGO (with $J = 5$) on synthetic data at 40dB and 30dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-4.55 ± 0.64	-5.78 ± 0.35	-0.67 ± 0.21	-0.94 ± 0.16
	ALGO-BBGP	-4.48 ± 0.6	-5.76 ± 0.33	-0.57 ± 0.2	-0.9 ± 0.15
	ALGO-PG	-4.43 ± 0.61	-5.59 ± 0.31	-2.09 ± 0.16	-1.79 ± 0.15
	ALGO-FISTA	-3.14 ± 0.62	-4.38 ± 0.25	-0.85 ± 0.17	-1.03 ± 0.15
	ALGO-FW	-4.17 ± 0.63	-5.9 ± 0.35	0.24 ± 0.22	-0.64 ± 0.15
10	ALGO-GP	-0.77 ± 0.31	-2.17 ± 0.15	1.26 ± 0.15	0.16 ± 0.09
	ALGO-BBGP	-0.66 ± 0.29	-2.16 ± 0.16	1.46 ± 0.12	0.22 ± 0.08
	ALGO-PG	-1.31 ± 0.3	-2.54 ± 0.16	0.22 ± 0.11	-0.28 ± 0.09
	ALGO-FISTA	0.17 ± 0.34	-1.72 ± 0.17	1.6 ± 0.1	0.27 ± 0.07
	ALGO-FW	-0.96 ± 0.32	-2.45 ± 0.18	1 ± 0.11	0.04 ± 0.08

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	16.84 ± 0.33	-14.13 ± 0.4	7.36 ± 0.22	22.07 ± 0.49
	ALGO-BBGP	16.78 ± 0.32	-14.1 ± 0.4	7.45 ± 0.21	22 ± 0.49
	ALGO-PG	18.09 ± 0.43	-14.75 ± 0.34	6.26 ± 0.29	23.26 ± 0.54
	ALGO-FISTA	15.35 ± 0.23	-13.39 ± 0.43	9.27 ± 0.23	20.57 ± 0.59
	ALGO-FW	16.33 ± 0.4	-13.88 ± 0.38	7.74 ± 0.25	21.69 ± 0.43
10	ALGO-GP	8.95 ± 0.11	-10.16 ± 0.46	19.05 ± 0.24	14.03 ± 0.59
	ALGO-BBGP	8.71 ± 0.3	-10.04 ± 0.54	19.53 ± 0.67	13.79 ± 0.72
	ALGO-PG	10.63 ± 0.17	-11.01 ± 0.42	15.53 ± 0.3	15.76 ± 0.53
	ALGO-FISTA	7.86 ± 0.17	-9.62 ± 0.39	21.5 ± 0.4	12.93 ± 0.57
	ALGO-FW	9.58 ± 0.34	-10.48 ± 0.53	17.58 ± 0.62	14.67 ± 0.71

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	11.49 ± 3.98	122.21 ± 6.89
	ALGO-BBGP	10.96 ± 3.91	109.97 ± 10.37
	ALGO-PG	20.55 ± 6.86	318.24 ± 6.67
	ALGO-FISTA	5.87 ± 1.99	86.33 ± 3.68
	ALGO-FW	9.3 ± 3.71	126.13 ± 4.73
10	ALGO-GP	6.8 ± 2.34	72.59 ± 4.25
	ALGO-BBGP	7.83 ± 2.78	77.65 ± 4.92
	ALGO-PG	12 ± 3.99	186.19 ± 4.76
	ALGO-FISTA	4.66 ± 1.56	68.48 ± 3.46
	ALGO-FW	7.32 ± 2.88	99.94 ± 3.74

Table B.26: Average HU and HSR performance of ALGO (with $J = 5$) on synthetic data at 20dB and 10dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-5.25 ± 0.96	-7.7 ± 0.59	-2.16 ± 0.31	-2.09 ± 0.29
	ALGO-BBGP	-5.26 ± 0.9	-7.65 ± 0.59	-1.92 ± 0.32	-1.97 ± 0.27
	ALGO-PG	-5.57 ± 0.95	-7.83 ± 0.64	-3.18 ± 0.26	-2.77 ± 0.25
	ALGO-FISTA	-5.83 ± 1.09	-7.88 ± 0.67	-2.92 ± 0.28	-2.6 ± 0.27
	ALGO-FW	-4.8 ± 0.83	-7.36 ± 0.57	0.22 ± 0.5	-1.34 ± 0.26
30	ALGO-GP	-5.13 ± 0.87	-7.42 ± 0.53	-1.82 ± 0.26	-1.78 ± 0.23
	ALGO-BBGP	-5.22 ± 0.85	-7.4 ± 0.67	-1.46 ± 0.28	-1.6 ± 0.24
	ALGO-PG	-5.36 ± 0.86	-7.42 ± 0.54	-2.95 ± 0.22	-2.54 ± 0.21
	ALGO-FISTA	-5.27 ± 0.96	-7.02 ± 0.51	-2.4 ± 0.24	-2.15 ± 0.22
	ALGO-FW	-4.77 ± 0.82	-7.17 ± 0.58	0.35 ± 0.42	-1.08 ± 0.23

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	20.19 ± 0.84	-15.81 ± 0.34	4.22 ± 0.4	28.79 ± 0.59
	ALGO-BBGP	20.17 ± 0.82	-15.8 ± 0.39	4.21 ± 0.38	28.81 ± 0.57
	ALGO-PG	22.41 ± 1.02	-16.92 ± 0.35	3.52 ± 0.39	30.93 ± 0.85
	ALGO-FISTA	23.87 ± 1.19	-17.65 ± 0.46	3.15 ± 0.39	31.92 ± 0.99
	ALGO-FW	18.8 ± 0.94	-15.11 ± 0.46	4.82 ± 0.42	27.64 ± 0.5
30	ALGO-GP	19.66 ± 0.73	-15.55 ± 0.33	4.69 ± 0.35	26.44 ± 0.45
	ALGO-BBGP	19.62 ± 1.3	-15.51 ± 0.93	4.63 ± 0.42	26.45 ± 1.51
	ALGO-PG	21.52 ± 0.84	-16.47 ± 0.31	4 ± 0.35	28.21 ± 0.65
	ALGO-FISTA	21.67 ± 0.78	-16.55 ± 0.35	4.27 ± 0.32	27.77 ± 0.67
	ALGO-FW	18.63 ± 1.03	-15.02 ± 0.64	5.19 ± 0.42	25.72 ± 0.91

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	48.87 ± 18.85	256.69 ± 26.62
	ALGO-BBGP	58.6 ± 22.3	291.66 ± 34.16
	ALGO-PG	107.78 ± 38.83	841.97 ± 55.47
	ALGO-FISTA	12.26 ± 4.4	93.19 ± 5.21
	ALGO-FW	51.11 ± 22.48	339.87 ± 25.33
30	ALGO-GP	28.74 ± 10.51	155.86 ± 10.26
	ALGO-BBGP	32.73 ± 12.51	167.6 ± 12.07
	ALGO-PG	55.52 ± 20.28	439.24 ± 10.94
	ALGO-FISTA	9.07 ± 3.37	69.72 ± 3.49
	ALGO-FW	22.11 ± 9.16	156.96 ± 7.76

Table B.27: Average HU and HSR performance of ALGO (with $J = 10$) on synthetic data at 40dB and 30dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-4.23 ± 0.63	-5.5 ± 0.33	-0.58 ± 0.21	-0.89 ± 0.16
	ALGO-BBGP	-4.45 ± 0.61	-5.65 ± 0.34	-0.14 ± 0.19	-0.67 ± 0.15
	ALGO-PG	-4.3 ± 0.6	-5.42 ± 0.3	-1.97 ± 0.16	-1.7 ± 0.15
	ALGO-FISTA	-3.07 ± 0.63	-4.38 ± 0.26	-0.6 ± 0.17	-0.88 ± 0.15
	ALGO-FW	-4.07 ± 0.61	-5.7 ± 0.34	0.37 ± 0.21	-0.57 ± 0.15
10	ALGO-GP	-0.42 ± 0.31	-1.89 ± 0.18	1.51 ± 0.15	0.25 ± 0.09
	ALGO-BBGP	-0.59 ± 0.29	-2.03 ± 0.16	1.69 ± 0.11	0.32 ± 0.07
	ALGO-PG	-0.99 ± 0.3	-2.32 ± 0.15	0.41 ± 0.11	-0.19 ± 0.09
	ALGO-FISTA	0.16 ± 0.33	-1.66 ± 0.17	1.78 ± 0.09	0.34 ± 0.07
	ALGO-FW	-0.72 ± 0.31	-2.23 ± 0.18	1.18 ± 0.11	0.11 ± 0.08

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	16.22 ± 0.28	-13.82 ± 0.41	8.03 ± 0.19	21.47 ± 0.49
	ALGO-BBGP	16.29 ± 0.3	-13.86 ± 0.41	7.91 ± 0.2	21.53 ± 0.49
	ALGO-PG	17.87 ± 0.41	-14.65 ± 0.34	6.49 ± 0.27	23.05 ± 0.53
	ALGO-FISTA	14.96 ± 0.26	-13.19 ± 0.44	9.69 ± 0.27	20.21 ± 0.59
	ALGO-FW	16.03 ± 0.38	-13.72 ± 0.38	8.15 ± 0.24	21.37 ± 0.45
10	ALGO-GP	8.29 ± 0.13	-9.84 ± 0.43	20.53 ± 0.3	13.37 ± 0.58
	ALGO-BBGP	8.31 ± 0.29	-9.84 ± 0.53	20.45 ± 0.69	13.39 ± 0.71
	ALGO-PG	10.09 ± 0.16	-10.73 ± 0.43	16.61 ± 0.29	15.19 ± 0.55
	ALGO-FISTA	7.6 ± 0.2	-9.49 ± 0.37	22.15 ± 0.46	12.68 ± 0.56
	ALGO-FW	9.07 ± 0.32	-10.22 ± 0.52	18.7 ± 0.66	14.14 ± 0.71

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	18.83 ± 6.51	100.81 ± 6.66
	ALGO-BBGP	24.37 ± 8.68	122.23 ± 6.47
	ALGO-PG	29.23 ± 10.03	227.55 ± 5.55
	ALGO-FISTA	7.56 ± 2.57	57.3 ± 2.9
	ALGO-FW	15.19 ± 6.04	104.89 ± 4.35
10	ALGO-GP	10.99 ± 3.99	58.89 ± 6.38
	ALGO-BBGP	14.88 ± 5.26	75.16 ± 4.82
	ALGO-PG	18.05 ± 6.07	141.09 ± 4.16
	ALGO-FISTA	6.35 ± 2.14	48.22 ± 3.31
	ALGO-FW	11.46 ± 4.45	79.16 ± 3.14

Table B.28: Average HU and HSR performance of ALGO (with $J = 10$) on synthetic data at 20dB and 10dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
40	ALGO-GP	-4.88 ± 0.74	-7.06 ± 0.55	0 ± 0.46	-1.2 ± 0.27
	ALGO-BBGP	-4.92 ± 0.72	-7.08 ± 0.54	-0.01 ± 0.45	-1.21 ± 0.26
	ALGO-PG	-5.11 ± 0.99	-7.52 ± 0.58	-3.18 ± 0.25	-2.74 ± 0.25
	ALGO-FISTA	-4.2 ± 0.73	-6.32 ± 0.53	-0.17 ± 0.36	-1.01 ± 0.23
	ALGO-FW	-4.64 ± 0.89	-7.35 ± 0.58	0.49 ± 0.49	-1.31 ± 0.26
30	ALGO-GP	-4.74 ± 0.7	-6.71 ± 0.52	0.09 ± 0.41	-0.98 ± 0.23
	ALGO-BBGP	-4.82 ± 0.73	-6.74 ± 0.63	0.09 ± 0.4	-0.99 ± 0.22
	ALGO-PG	-4.89 ± 0.86	-7.02 ± 0.49	-2.88 ± 0.21	-2.46 ± 0.21
	ALGO-FISTA	-4.11 ± 0.73	-5.97 ± 0.43	0.02 ± 0.35	-0.81 ± 0.2
	ALGO-FW	-4.53 ± 0.85	-7.02 ± 0.58	0.68 ± 0.42	-1.01 ± 0.23

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
40	ALGO-GP	20.01 ± 0.79	-15.72 ± 0.4	4.37 ± 0.36	28.19 ± 0.53
	ALGO-BBGP	20.08 ± 0.78	-15.75 ± 0.4	4.36 ± 0.36	28.22 ± 0.53
	ALGO-PG	21.36 ± 0.92	-16.39 ± 0.38	3.76 ± 0.38	29.89 ± 0.73
	ALGO-FISTA	19.48 ± 0.96	-15.45 ± 0.46	4.96 ± 0.47	27.87 ± 0.82
	ALGO-FW	19.7 ± 1.02	-15.56 ± 0.5	4.54 ± 0.39	28.29 ± 0.64
30	ALGO-GP	19.26 ± 0.67	-15.35 ± 0.38	5.12 ± 0.32	25.73 ± 0.43
	ALGO-BBGP	19.3 ± 1.25	-15.35 ± 0.92	5.09 ± 0.42	25.74 ± 1.48
	ALGO-PG	20.6 ± 0.78	-16.02 ± 0.35	4.32 ± 0.34	27.32 ± 0.56
	ALGO-FISTA	18.77 ± 0.88	-15.1 ± 0.46	5.73 ± 0.45	25.37 ± 0.6
	ALGO-FW	19.45 ± 1.04	-15.43 ± 0.64	5.1 ± 0.39	26.14 ± 0.96

SNR (dB)	Method	Time(sec)	Outer Iter.
40	ALGO-GP	246.94 ± 74.52	159.92 ± 24.28
	ALGO-BBGP	273 ± 82.88	163.54 ± 21.97
	ALGO-PG	219.61 ± 58.6	206.66 ± 18.57
	ALGO-FISTA	84.06 ± 35.64	77.14 ± 24.48
	ALGO-FW	221.52 ± 64.92	197.24 ± 18.69
30	ALGO-GP	136.78 ± 36.5	86.22 ± 11.89
	ALGO-BBGP	150.23 ± 34.68	88.32 ± 9.02
	ALGO-PG	146.86 ± 31.24	135 ± 6.44
	ALGO-FISTA	42.44 ± 12.98	37.76 ± 7.76
	ALGO-FW	108.64 ± 23.68	94.74 ± 7.18

Table B.29: Average HU and HSR performance of ALGO (with $J = 100$) on synthetic data at 40dB and 30dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

SNR (dB)	Method	NMSE-en(dB)	MSE-en(dB)	NMSE-ab(dB)	MSE-ab(dB)
20	ALGO-GP	-3.55 ± 0.51	-4.74 ± 0.3	0.81 ± 0.26	-0.31 ± 0.16
	ALGO-BBGP	-3.65 ± 0.51	-4.83 ± 0.28	0.83 ± 0.24	-0.3 ± 0.15
	ALGO-PG	-3.56 ± 0.63	-4.81 ± 0.26	-1.54 ± 0.15	-1.41 ± 0.15
	ALGO-FISTA	-3.05 ± 0.54	-4.37 ± 0.26	1.32 ± 0.18	-0.06 ± 0.13
	ALGO-FW	-3.48 ± 0.6	-5.13 ± 0.3	0.78 ± 0.22	-0.42 ± 0.15
10	ALGO-GP	0.13 ± 0.3	-1.63 ± 0.17	2.01 ± 0.08	0.42 ± 0.07
	ALGO-BBGP	0.09 ± 0.29	-1.65 ± 0.18	2.01 ± 0.1	0.42 ± 0.07
	ALGO-PG	-0.23 ± 0.31	-1.82 ± 0.16	1.14 ± 0.1	0.11 ± 0.08
	ALGO-FISTA	0.2 ± 0.31	-1.6 ± 0.18	2.11 ± 0.08	0.46 ± 0.07
	ALGO-FW	-0.11 ± 0.29	-1.8 ± 0.18	1.63 ± 0.1	0.28 ± 0.07

SNR (dB)	Method	RSNR(dB)	RMSE(dB)	SAM(deg.)	PSNR(dB)
20	ALGO-GP	14.85 ± 0.23	-13.13 ± 0.44	9.84 ± 0.22	20.05 ± 0.6
	ALGO-BBGP	14.92 ± 0.21	-13.17 ± 0.45	9.75 ± 0.2	20.12 ± 0.6
	ALGO-PG	16.39 ± 0.31	-13.91 ± 0.4	7.94 ± 0.21	21.66 ± 0.5
	ALGO-FISTA	13.77 ± 0.31	-12.59 ± 0.44	11.15 ± 0.33	19.03 ± 0.6
	ALGO-FW	15.24 ± 0.27	-13.33 ± 0.41	9.28 ± 0.22	20.47 ± 0.54
10	ALGO-GP	7.27 ± 0.21	-9.32 ± 0.36	23.01 ± 0.5	12.35 ± 0.56
	ALGO-BBGP	7.31 ± 0.31	-9.34 ± 0.47	22.88 ± 0.87	12.38 ± 0.69
	ALGO-PG	8.48 ± 0.16	-9.93 ± 0.4	20.06 ± 0.35	13.56 ± 0.56
	ALGO-FISTA	7.12 ± 0.24	-9.25 ± 0.35	23.39 ± 0.59	12.2 ± 0.56
	ALGO-FW	7.91 ± 0.31	-9.64 ± 0.49	21.36 ± 0.79	12.98 ± 0.69

SNR (dB)	Method	Time(sec)	Outer Iter.
20	ALGO-GP	78.48 ± 18.32	50.04 ± 5.78
	ALGO-BBGP	92.49 ± 20.72	54.18 ± 4.07
	ALGO-PG	97.46 ± 19.34	90.13 ± 3.81
	ALGO-FISTA	38.01 ± 8.16	34.36 ± 2.87
	ALGO-FW	68.41 ± 14.73	59.32 ± 3.23
10	ALGO-GP	59.05 ± 16.55	38.73 ± 3.31
	ALGO-BBGP	69.47 ± 21.82	41.36 ± 3.41
	ALGO-PG	58.97 ± 15.05	56.15 ± 2.91
	ALGO-FISTA	40.21 ± 11.03	37.4 ± 3.37
	ALGO-FW	49.9 ± 14.79	44.62 ± 2.72

Table B.30: Average HU and HSR performance of ALGO (with $J = 100$) on synthetic data at 20dB and 10dB SNR; $N = 16$; $L = 100 \times 100$; 1000 trials.

B.3 Algorithm Speed of ALGO-FISTA and ALGO-FW Against Model Order

In this section we compare the algorithm speed against model order N using the semi-real Cuprite dataset. The simulation is to solve the HSR problem formulated by Wei *et al.* , i.e., the feasible set $\mathcal{S}$ is a simplex. Since the projection operation onto $\mathcal{S}$ is expensive ($\mathcal{O}(LN \log N)$, per iteration [75]), we expect a larger N requires more computation time by ALGO-FISTA. We run 100 trials in each model order setting to obtain the average runtime and HSR recovery error (SAM and PSNR). For simplicity we only compare the two fastest algorithms, i.e., ALGO-FISTA and ALGO-FW.

In Figure B.1 ALGO-FW is slower than ALGO-FISTA when N is small. When N increases to 30 or larger, ALGO-FW becomes faster than ALGO-FISTA. A suggested explaination is that the impact of the projection-free nature of ALGO-FW becomes more significant when N is large.

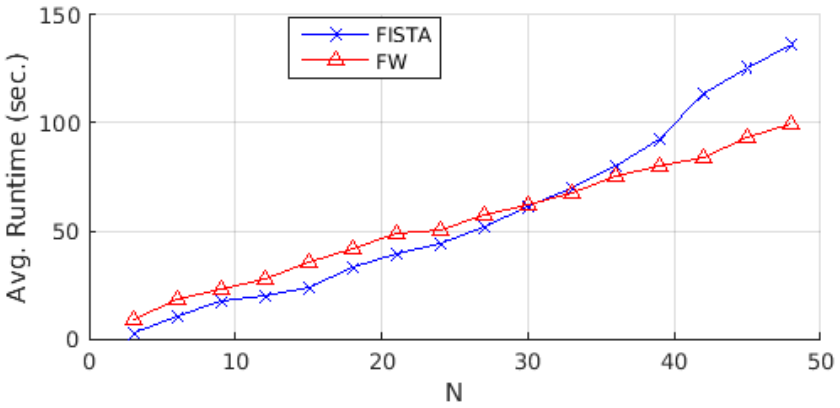


Figure B.1: Runtime by ALGO-FISTA and ALGO-FW against model order N.

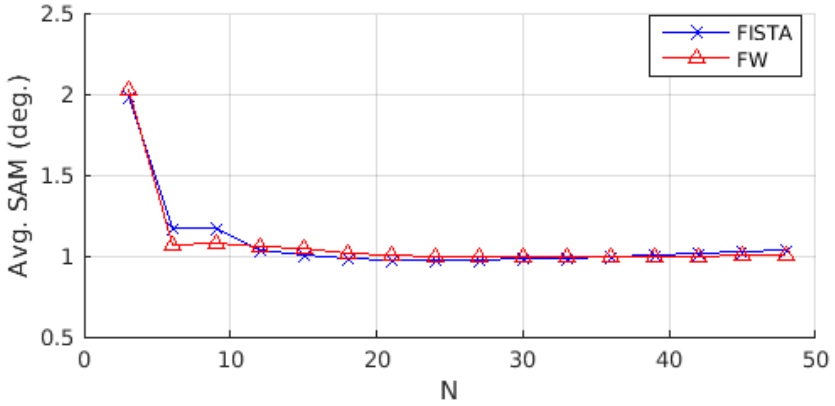


Figure B.2: SAM by ALGO-FISTA and ALGO-FW against model order N.

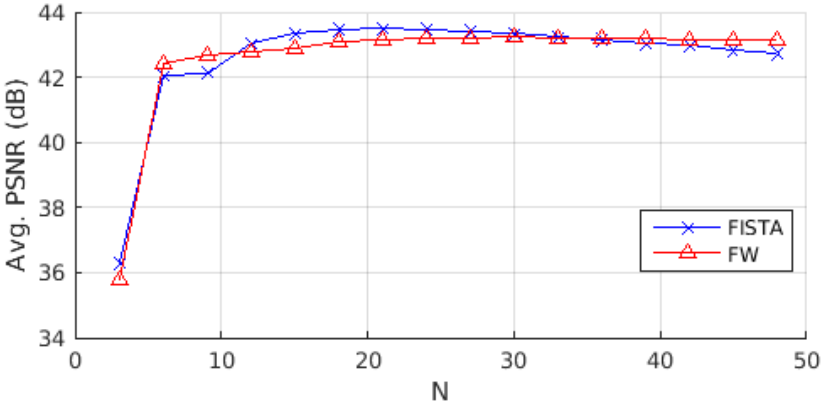


Figure B.3: PSNR by ALGO-FISTA and ALGO-FW against model order N.

B.4 Algorithm Speed of ALGO-FISTA and ALGO-FW Against Pixel Number

In this section we compare the algorithm speed against pixel number L by synthetic simulation. Again, the simulation is to solve the HSR problem formulated by Wei *et al.* , i.e., the feasible set $\mathcal{S}$ is a simplex. We run 100 trials in each pixel number setting to obtain the average runtime.

In Figure B.4 we see that both ALGO-FISTA and ALGO-FW requires more runtime when L increases, which is reasonable. However we also observe that with a larger N, the increase in runtime by ALGO-FISTA is more significant than that by ALGO-FW. This can be explained by the fact that ALGO-FISTA requires more computation during the simplex projection per each iteration when N is large, while this is not the case for ALGO-FW due to its projection-free nature that prevents ALGO-FW from spending extra computation in projection operation.

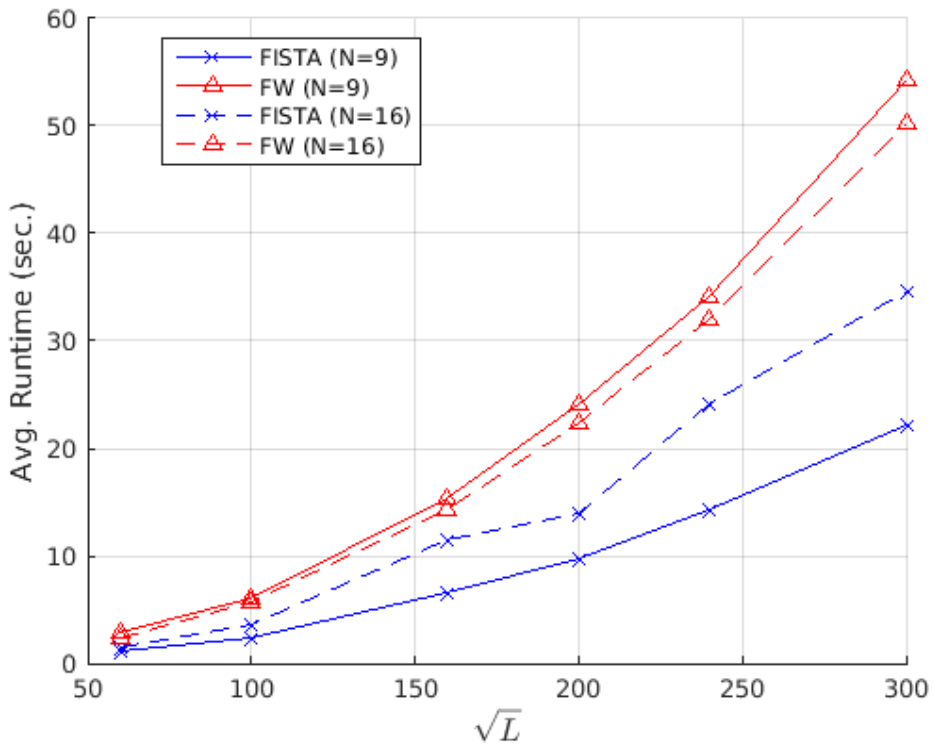


Figure B.4: Run time by ALGO-FISTA and ALGO-FW against pixel number L.

Bibliography

[1] F. A. Kruse. Comparison of AVIRIS and Hyperion for Hyperspectral Mineral Mapping. In *Proceedings of the 11th JPL Airborne Geoscience Workshop, 4-8 March 2002, Pasadena, California*, 2002.

[2] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. S., M. R. Olah, and O. Williams. Imaging Spectroscopy and the Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS). *Remote Sensing of Environment*, 65(3):227–248, 1998.

[3] R. O. Green and J. Boardman. Exploration of the Relationship between Information Content and Signal-to-Noise Ratio and Spatial Resolution in AVIRIS Spectral Data. In *AVIRIS Workshop (2001), Pasadena, CA*, 2001.

[4] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock. Imaging Spectrometry for Earth Remote Sensing. *Science*, 228(4704):1147–1153, 1985.

[5] F. A. Kruse. Advances in hyperspectral remote sensing for geologic mapping and exploration. In *Proceedings 9th Australasian Remote Sensing Conference, Sydney, Australia*, 1998.

[6] C. Kumaravelu and A. Gopal. A Review on the Applications of near–Infrared Spectrometer and Chemometrics for the Agro-Food Processing Industries. In *2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR)*, pages 8–12, July 2015.

[7] S. Marshall, T. Kelman, T. Qiao, P. Murray, and J. Zabalza. Hyperspectral Imaging for Food Applications. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 2854–2858, Aug 2015.

[8] P. Mishra, C. B. Y. Cordella, D. N. Rutledge, P. Barreiro, J. M. Roger, and B. Diezma. Application of Independent Components Analysis with the JADE Algorithm and NIR Hyperspectral Imaging for Revealing Food Adulteration. *Journal of Food Engineering*, 168:7–15, 2016.

[9] N. Yokoya, C. Grohnfeldt, and J. Chanussot. Hyperspectral and Multispectral Data Fusion: A Comparative Review of the Recent Literature. *IEEE Geoscience and Remote Sensing Magazine*, 5(2):29–56, June 2017.

[10] W. Carper, T. M. Lillesand, and P. W. Kiefer. The Use of Intensity-Hue-Saturation Transformations for Merging Spot Panchromatic and Multispectral Image Data. *Photogramm. Eng. Remote Sens.*, 56(4):459–467, April 1990.

[11] C. A. Laben and B. V. Brower. Process for Enhancing the Spatial Resolution of Multispectral Imagery using Pan-sharpening, 2000. Patent US 6011875 A.

[12] B. Aiazzi, S. Baronti, and M. Selva. Improving Component Substitution Pansharpening Through Multivariate Regression of MS +Pan Data. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3230–3239, Oct 2007.

[13] J. G. Liu. Smoothing Filter-based Intensity Modulation: A Spectral Preserve Image Fusion Technique for Improving Spatial Details. *International Journal of Remote Sensing*, 21(18):3461–3472, 2000.

[14] B. Aiazzi, L. Alparone, S. Baronti, A. Garzelli, and M. Selva. MTF-Tailored Multiscale Fusion of High-Resolution MS and Pan Imagery. *Photogramm. Eng. Remote Sens.*, 72 (5):591–596, May 2006.

[15] M. T. Eismann and R. C. Hardie. Application of the Stochastic Mixing Model to Hyperspectral Resolution Enhancement. *IEEE Transactions on Geoscience and Remote Sensing*, 42(9):1924–1933, Sept 2004.

[16] R. C. Hardie, M. T. Eismann, and G. L. Wilson. MAP Estimation for Hyperspectral Image Resolution Enhancement using an Auxiliary Sensor. *IEEE Transactions on Image Processing*, 13(9):1174–1184, Sept 2004.

[17] M. T. Eismann and R. C. Hardie. Hyperspectral Resolution Enhancement using High-Resolution Multispectral Imagery with Arbitrary Response Functions. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):455–465, March 2005.

[18] N. Yokoya, T. Yairi, and A. Iwasaki. Coupled Nonnegative Matrix Factorization Unmixing for Hyperspectral and Multispectral Data Fusion. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):528–537, Feb 2012.

[19] O. Bern, A. Helens, P. Pilleri, and C. Joblin. Non-negative Matrix Factorization Pansharpening of Hyperspectral Data: An Application to Mid-Infrared Astronomy. In *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pages 1–4, June 2010.

[20] R. Kawakami, Y. Matsushita, J. Wright, M. Ben-Ezra, Y. W. Tai, and K. Ikeuchi. High-Resolution Hyperspectral Imaging Via Matrix Factorization. In *CVPR 2011*, pages 2329–2336, June 2011.

[21] B. Huang, H. Song, H. Cui, J. Peng, and Z. Xu. Spatial and Spectral Image Fusion Using Sparse Matrix Factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 52(3):1693–1704, March 2014.

[22] M. A. Veganzones, M. Simes, G. Licciardi, N. Yokoya, J. M. Bioucas-Dias, and J. Chanussot. Hyperspectral Super-Resolution of Locally Low Rank Images from Complementary Multisource Data. *IEEE Transactions on Image Processing*, 25(1):274–288, Jan 2016.

[23] C. Yi, Y. Q. Zhao, J. Yang, J. C. W. Chan, and S. G. Kong. Joint Hyperspectral Superresolution and Unmixing with Interactive Feedback. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3823–3834, July 2017.

[24] L. Loncan, L. B. de Almeida, J. M. Bioucas-Dias, X. Briottet, J. Chanussot, N. Dobigeon, S. Fabre, W. Liao, G. A. Licciardi, M. Simoes, J. Y. Tourneret, M. A. Veganzones, G. Vivone, Q. Wei, and N. Yokoya. Hyperspectral Pansharpening: A Review. *IEEE Geoscience and Remote Sensing Magazine*, 3(3):27–46, Sept 2015.

[25] S. J. Wright. Coordinate Descent Algorithms. *Mathematical Programming*, 151(1): 3–34, Jun 2015.

[26] Y. Nesterov. Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[27] C. A. Bouman and K. Sauer. A Unified Approach to Statistical Tomography using Coordinate Descent Optimization. *IEEE Transactions on Image Processing*, 5(3):480–492, Mar 1996.

[28] J. C. Ye, K. J. Webb, C. A. Bouman, and R. P. Millane. Optical Diffusion Tomography by Iterative-Coordinate-Descent Optimization in a Bayesian Framework. *Journal of the Optical Society of America A-Optics Image Science and Vision*, 16(10):2400–2412, Oct 1999.

[29] H. Liu, M. Palatucci, and J. Zhang. Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. In *International Conference on Machine Learning*, Pittsburgh, PA, June 2009.

[30] A. A. Canutescu and R. L. Dunbrack. Cyclic Coordinate Descent: A Robotics Algorithm for Protein Loop Closure. *Protein Science*, 12(5):963–972, May 2003.

[31] M. Florian and Y. Chen. A Coordinate Descent Method for the Bi-level O-D Matrix Adjustment Problem. *International Transactions in Operational Research*, 2(2):165–179, 1995.

[32] P. Breheny and J. Huang. Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection. *Annals of Applied Statistics*, 5(1):232–253, 2011.

[33] J. Friedman, T. Hastie, and R. Tibshirani. Sparse Inverse Covariance Estimation with the Graphical LASSO. *Biostatistics*, 9(3):432–441, Jul 2008.

[34] K. W. Chang, C. J. Hsieh, and C. J. Lin. Coordinate Descent Method for Large-Scale L2-loss Linear Support Vector Machines. *J. Mach. Learn. Res.*, 9:1369–1398, Jun 2008.

[35] Spectral Response of the Operational Land Imager In-Band, Band-Average Relative Spectral Response. `https://landsat.gsfc.nasa.gov/preliminary-spectral-response-of-the-operational-land-imager-in-band-band-averag` 2016. Accessed: 2016-09-29.

[36] Landsat 7 Science Data Users Handbook. `http://landsat.gsfc.nasa.gov/wp-content/uploads/2016/08/Landsat7_Handbook.pdf`, 2016. Accessed: 2016-09-29.

[37] J. A. Barsi, K. Lee, G. Kvaran, B. L. Markham, and J. A. Pedelty. The Spectral Response of the Landsat-8 Operational Land Imager. *Remote Sensing*, 6(10):10232–10251, 2014.

[38] AVIRIS Data - Ordering Free AVIRIS Standard Data Products. [Online]. Available"`http://aviris.jpl.nasa.gov/html/aviris.freedata.html`, 2016. [Accessed: 2016-05-01].

[39] M. T. Eismann and R. C. Hardie. Hyperspectral Resolution Enhancement using High-Resolution Multispectral Imagery with Arbitrary Response Functions. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):455–465, March 2005.

[40] R. Anderson, W. Malila., R. Maxwell, and L. Reed. Military Utility of Multispectral and Hyperspectral Sensors. Technical report, Environmental Research Inst. of Michigan Ann Arbor, Nov 1994.

[41] A. R. Gillespie, M. O. Smith, J. B. Adams, S. C. Willis, and A. F. Fischer. Interpretation of Residual Images: Spectral Mixture Analysis of AVIRIS Images. *Proc. 2nd AVIRIS Workshop, Owens Valley, California*, pages 243–270, 1990.

[42] L. Drumetz, M. A. Veganzones, S. Henrot, R. Phlypo, J. Chanussot, and C. Jutten. Blind Hyperspectral Unmixing Using an Extended Linear Mixing Model to Address Spectral Variability. *IEEE Transactions on Image Processing*, 25(8):3890–3905, Aug 2016.

[43] C. I. Chang and D. C. Heinz. Constrained Subpixel Target Detection for Remotely Sensed Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 38(3):1144–1159, May 2000.

[44] M. Simes, J. BioucasDias, L. B. Almeida, and J. Chanussot. A Convex Formulation for Hyperspectral Image Superresolution via Subspace-Based Regularization. *IEEE Transactions on Geoscience and Remote Sensing*, 53(6):3373–3388, June 2015.

[45] M. Wang, J. Yu, J. H. Xue, and W. Sun. Denoising of Hyperspectral Images Using Group Low-Rank Representation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(9):4420–4427, Sept 2016.

[46] C. G. Tsinos, A. A. Rontogiannis, and K. Berberidis. Distributed Blind Hyperspectral Unmixing Via Joint Sparsity and Low-Rank Constrained Non-Negative Matrix Factorization. *IEEE Transactions on Computational Imaging*, 3(2):160–174, June 2017.

[47] D. C. Heinz and Chein-I-Chang. Fully Constrained Least Squares Linear Spectral Mixture Analysis Method for Material Quantification in Hyperspectral Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(3):529–545, Mar 2001.

[48] Q. Wei, J. Bioucas-Dias, N. Dobigeon, J. Y. Tourneret, M. Chen, and S. Godsill. Multiband Image Fusion Based on Spectral Unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7236–7249, Dec 2016.

[49] Q. Wei, N. Dobigeon, and J. Y. Tourneret. Fast Fusion of Multi-Band Images Based on Solving a Sylvester Equation. *IEEE Transactions on Image Processing*, 24(11): 4109–4121, Nov 2015.

[50] N. Gillis, D. Kuang, and H. Park. Hierarchical Clustering of Hyperspectral Images Using Rank-Two Nonnegative Matrix Factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):2066–2078, April 2015.

[51] L. Sun, B. Jeon, Y. Zheng, and Z. Wu. Hyperspectral Image Restoration Using Low-Rank Representation on Spectral Difference Image. *IEEE Geoscience and Remote Sensing Letters*, PP(99):1–5, 2017.

[52] H. Zhang, W. He, L. Zhang, H. Shen, and Q. Yuan. Hyperspectral Image Restoration Using Low-Rank Matrix Recovery. *IEEE Transactions on Geoscience and Remote Sensing*, 52(8):4729–4743, Aug 2014.

[53] M. S. Karoui, Y. Deville, F. Z. Benhalouche, and I. Boukerch. Hypersharpening by Joint-Criterion Nonnegative Matrix Factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 55(3):1660–1670, March 2017.

[54] J. M. Bioucas-Dias and J. M. P. Nascimento. Hyperspectral Subspace Identification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2435–2445, Aug 2008.

[55] C. Yi, Y. Q. Zhao, J. Yang, J. C. W. Chan, and S. G. Kong. Joint Hyperspectral Superresolution and Unmixing With Interactive Feedback. *IEEE Transactions on Geoscience and Remote Sensing*, PP(99):1–12, April 2017.

[56] C. I. Chang, S. S. Chiang, J. A. Smith, and I. W. Ginsberg. Linear Spectral Random Mixture Analysis for Hyperspectral Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 40(2):375–392, Feb 2002.

[57] A. Sumarsono and Q. Du. Low-Rank Subspace Representation for Estimating the Number of Signal Subspaces in Hyperspectral Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 53(11):6286–6292, Nov 2015.

[58] T. H. Chan, C. Y. Chi, Y. M. Huang, and W. K. Ma. A Convex Analysis-Based Minimum-Volume Enclosing Simplex Algorithm for Hyperspectral Unmixing. *IEEE Transactions on Signal Processing*, 57(11):4418–4432, Nov 2009.

[59] E. Wycoff, T. H. Chan, K. Jia, W. K. Ma, and Y. Ma. A Non-Negative Sparse Promoting Algorithm for High Resolution Hyperspectral Imaging. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1409–1413, May 2013.

[60] 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*, 42(10):2085–2095, Oct 2004.

[61] 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*, 45(3):765–777, March 2007.

[62] X. Fu, K. Huang, B. Yang, W. K. Ma, and N. D. Sidiropoulos. Robust Volume Minimization-Based Matrix Factorization for Remote Sensing and Document Clustering. *IEEE Transactions on Signal Processing*, 64(23):6254–6268, Dec 2016.

[63] J. M. Bioucas-Dias. A Variable Splitting Augmented Lagrangian Approach to Linear Spectral Unmixing. In *2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pages 1–4, Aug 2009.

[64] J. M. P. Nascimento and J. M. B. Dias. Vertex Component Analysis: A Fast Algorithm to Unmix Hyperspectral Data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(4):898–910, April 2005.

[65] N. Keshava and J. F. Mustard. Spectral Unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, Jan 2002.

[66] P. E. Johnson, M. O. Smith, S. Taylor-George, and J. B. Adams. A Semiempirical Method for Analysis of the Reflectance Spectra of Binary Mineral Mixtures. *Journal of Geophysical Research*, 88(B4):3357–3561, April 1983.

[67] P. E. Johnson, M. O. Smith, and J. B. Adams. Simple Algorithms for Remote Determination of Mineral Abundances and Particle Sizes from Reflectance Spectra. *Journal of Geophysical Research: Planets*, 97(E2):2649–2657, Feb 1992.

[68] J. F. Mustard and C. M. Pieters. Quantitative Abundance Estimates from Bidirectional Reflectance Measurements. *Journal of Geophysical Research: Solid Earth*, 92(B4): E617–E626, March 1987.

[69] J. F. Mustard and C. M. Pieters. Photometric Phase Functions of Common Geologic Minerals and Applications to Quantitative Analysis of Mineral Mixture Reflectance Spectra. *Journal of Geophysical Research: Solid Earth*, 94(B10):13619–13634, Oct 1989.

[70] D. D. Lee and H. S. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401(6755):788–791, Oct 1999.

[71] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

[72] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipiliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[73] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A General Analysis of the Convergence of ADMM. In *International Conference on Machine Learning 32*, pages 343–352, 2015.

[74] M. Hong and Z. Q. Luo. On the Linear Convergence of the Alternating Direction Method of Multipliers. *Mathematical Programming*, 162(1):165–199, 2017.

[75] W. Wang and M. Á. Carreira-Perpiñán. Projection onto The Probability Simplex: An Efficient Algorithm with A Simple Proof, and An Application. *arXiv preprint*, arXiv: 1309.1541v1, 2013.

[76] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141, 1988.

[77] Y. Nesterov. *Introductory Lectures on Convex Optimization : A Basic Course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London, 2004.

[78] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[79] M. Frank and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 3:95–110, March-June 1956.

[80] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, 1999.

[81] S. M. Goldfeld, R. E. Quandt, and H. F. Trotter. Maximization by Quadratic Hill-Climbing. *Econometrica*, 34(3):541–551, 1966.

[82] Jr. J. E. Dennis and Jorge J. Mor. Quasi-Newton Methods, Motivation and Theory. *SIAM Review*, 19(1):46–89, 1977.

[83] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, Dec 2007.

[84] T. Serafini, G. Zanghirati, and L. Zanni. Gradient Projection Methods for Quadratic Programs and Applications in Training Support Vector Machines. *Optimization Methods and Software*, 20(2-3):353–378, 2005.

[85] N. Parikh and S. Boyd. Proximal Algorithms. *Foundations and Trends in Optimzation*, 1(3):127–239, Jan 2014.

[86] P. L. Combettes and J.-C. Pesquet. *Proximal Splitting Methods in Signal Processing*, pages 185–212. Springer New York, New York, NY, 2011.

[87] A. Beck and M. Teboulle. A Conditional Gradient Method with Linear Rate of Convergence for Solving Convex Linear Systems. *Mathematical Methods of Operations Research*, 59(2):235–247, 2004.

[88] L. Wald, T. Ranchin, and M. Mangolini. Fusion of Satellite Images of Different Spatial Resolutions: Assessing the Quality of Resulting Images. *hotogrammetric Engineering and Remote Sensing*, 63(6):691–699, 1997.

[89] R. F. Kokaly, R. N. Clark, G. A. Swayze, K. E. Livo, T. M Hoefen, N. C.; Pearson, R. A. Wise, W. M. Benzel, H. A. Lowers, R. L. Driscoll, and A. J. Klein. USGS Spectral Library Version 7: U.S. Geological Survey Data Series 1035. `http://speclab.cr.usgs.gov/`, 2017.

[90] N. Yokoya and A. Iwasaki. Airborne Hyperspectral Data over Chikusei. Technical Report SAL-2016-05-27, Space Application Laboratory, University of Tokyo, Japan, May 2016. URL `http://park.itc.u-tokyo.ac.jp/sal/hyperdata/TechRepSAL20160527.pdf`.

[91] Hyperspec VNIR Imaging Sensor for the 400nm to 1000 nm Spectral Range. [Online]. Available"`http://www.alava-ing.es/repositorio/1718/pdf/4334/2/hyperspec-vnir.pdf`", 2016. [Accessed: 2016-05-01].

[92] M. C. U. Arajo, T. C. B. Saldanha, R.K.H. Galvo, T. Yoneyama, H. C. Chame, and V. Visani. The Successive Projections Algorithm for Variable Selection in Spectroscopic Multicomponent Analysis. *Chemometrics and Intelligent Laboratory Systems*, 57(2): 65–73, 2001.

[93] W. K. Ma, J. M. Bioucas-Dias, T. H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi, and C. Y. Chi. A Signal Processing Perspective on Hyperspectral Unmixing: Insights from Remote Sensing. *IEEE Signal Processing Magazine*, 31(1):67–81, Jan 2014. ISSN 1053-5888.