SISAL Revisited*

Chujun Huang[†], Mingjie Shao[†‡], Wing-Kin Ma[†], and Anthony Man–Cho So[§]

Abstract. Simplex identification via split augmented Lagrangian (SISAL) is a popularly-used algorithm in blind unmixing of hyperspectral images. Developed by José M. Bioucas-Dias in 2009, the algorithm is fundamentally relevant to tackling simplex-structured matrix factorization, and by extension, non-negative matrix factorization, which have many applications under their umbrellas. In this article, we revisit SISAL and provide new meanings to this quintessential algorithm. The formulation of SISAL was motivated from a geometric perspective, with no noise. We show that SISAL can be explained as an approximation scheme from a probabilistic simplex component analysis framework, which is statistical and is principally more powerful in accommodating the presence of noise. The algorithm for SISAL was designed based on a successive convex approximation method, with a focus on practical utility. It was not known, by analyses, whether the SISAL algorithm has any kind of guarantee of convergence to a stationary point. By establishing associations between the SISAL algorithm and a line-search-based proximal gradient method, we confirm that SISAL can indeed guarantee convergence to a stationary point. Our re-explanation of SISAL also reveals new formulations and algorithms. The performance of these new possibilities is demonstrated by numerical experiments.

Key words. simplex-structured matrix factorization, probabilistic simplex component analysis, hyperspectral unmixing, first-order optimization

AMS subject classifications. 15A23, 90C26

1. Introduction.

Simplex identification via split augmented Lagrangian (SISAL) is an algorithm developed by José M. Bioucas-Dias in 2009 [4]. It appears in a 4-page conference paper, with open source code (in MATLAB). It basically deals with a simplex-structured matrix factorization problem from hyperspectral imaging; the problem is famously known as hyperspectral unmixing (HU) in the community of hyperspectral remote sensing. It is worth mentioning that HU is not only a key topic in hyperspectral imaging [5, 21], it also has strong relationships with non-negative matrix factorization and the various machine learning applications thereof; see, e.g., [11, 15] and the references therein. The development of SISAL revolves around problem formulation and optimization algorithm design. SISAL has a unique place in the course of history of HU: it offered one of the first, and most pioneering, practical algorithms for a promising but difficult-to-implement strategy for HU, namely, simplex volume minimization (SVMin). It has become a benchmark and has been frequently used by researchers. By the authors' understanding, the reasons boil down to one: *it works well in practice.* SISAL has good running speed, scales well with the data sizes (very large ones)

[*] February 10, 2022.

Funding: This work was supported by a General Research Fund of Hong Kong Research Grant Council under Project ID CUHK 14205717.

[†] Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong SAR of China (cjhuang@link.cuhk.edu.hk,mingjieshao@cuhk.edu.hk,wkma@ee.cuhk.edu.hk).

[‡] Equal contribution with Chunjun Huang.

[§] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong SAR of China (manchoso@se.cuhk.edu.hk).

computationally, delivers reasonably good unmixing results, and demonstrates resilience to noise and modeling error effects. SISAL shows powerful intuitions by its inventor. As an article to pay tribute to Bioucas-Dias' tremendous insights to hyperspectral imaging, allow us to quote a saying by Steve Jobs: "Intuition is a very powerful thing, more powerful than intellect, in my opinion."

This article serves as an endeavor to continue the legacy of Bioucas-Dias' SISAL. It can also be regarded as the sequel of [34]. The SISAL work has left some open questions. First and foremost, SISAL requires tuning of a regularization parameter. That parameter has an impact on SISAL's noise resilience behaviors. It is not clear how we should choose that parameter, apart from empirical or human experience. To make the story more complicated, SISAL was motivated by the noiseless case, and the subsequent explanation of why SISAL works in the noisy case was intuitive. Our question is whether there exists an alternative explanation for the noisy case. To answer that, we pursue a probabilistic simplex component analysis (SCA) framework, wherein we employ a principled formulation, namely, the maximum likelihood, to deal with the problem under a pertinent statistical model (to be specified later). This statistical strategy for unmixing is different from SISAL or SVMin, which is geometric. The former, by principle, has the upper hand in the noisy case; it also frees us from parameter tuning. We will show that SISAL can be seen as an approximation scheme of probabilistic SCA. Moreover, the connections we build suggest a different concept: Rather than considering parameter tuning, we should work on a more general formulation of SISAL, which is induced from probabilistic SCA and has no pre-selected parameter (except for the noise variance which can be estimated from data).

Some prior work on the aforementioned direction should be recognized. The links between SVMin (but not SISAL) and statistical inference were noted in earlier works [23, 24], [10, Appendix]. The prequel of this article [34] describes the connections between SVMin and probabilistic SCA more explicitly, but it only showed similarities, not a direct connection, between SISAL and probabilistic SCA. This article shows a close connection between SISAL and probabilistic SCA, compared to the previous work. Curiously, a simple second-order statistics observation (to be shown in Section 3.4) provides the very crucial piece of jigsaw to complete the puzzle.

Second, it is intriguing to study the optimization aspects of SISAL. The problem formulated in SISAL is non-convex, and Bioucas-Dias derived a successive convex approximation algorithm to tackle the problem. The algorithm can be seen a first-order method, as will be elaborated upon later, and it is worth mentioning that, in 2009, non-convex first-order optimization was not as extensively studied as today. As mentioned, the algorithm proved to be a success in practice. Our question is whether the SISAL algorithm actually possesses any form of guarantees of finding a stationary point, leveraging on our much better understanding of non-convex first-order optimization today. We will see that the SISAL algorithm can be viewed as an instance of the proximal gradient method, with line search along the feasible direction. There are, however, caveats that prevent us from directly claiming convergence to a stationary point—a key component in the objective function does not have Lipschitz gradient, and its domain is the set of all invertible matrices (which is a non-convex set). In this connection we should mention that, in the current non-convex first-order optimization literature, it is very common to assume the aforementioned component to have Lipschitz gradient.

We will confirm that the SISAL algorithm, with a minor adjustment, can indeed guarantee convergence to a stationary point (more accurately, limit-point convergence). This is made possible by establishing associations between the SISAL algorithm and the line-search-based proximal gradient framework in [6].

Our endeavor to re-explain SISAL also gives rise to new insights for algorithms. Through connecting SISAL and probabilistic SCA, we see a more general formulation that resembles SISAL. The new formulation replaces SISAL's penalty term with a probabilistic penalty term, and it has the regularization parameter (which requires tuning in SISAL) eliminated. We custom-design a practical algorithm for the formulation (which is more difficult than the SISAL), and we will illustrate by numerical experiments that this probabilistic SISAL performs well under the high SNR regime. We also study a SISAL variant that is easier to work with from an optimization algorithm design viewpoint, and numerical results suggest that the variant is computationally competitive.

We organize this paper as follows. Section 2 provides the problem statement and reviews the formulation of SISAL. Section 3 studies probabilistic SCA, shows how probabilistic SCA and SISAL are connected, and, in the process, reveals new formulations. Section 4 considers the optimization aspects of SISAL, particularly, the stationarity guarantee of SISAL. Section 5 develops a practical algorithm for the new formulation of probabilistic SISAL. Section 6 provides synthetic and semi-real data experiments. Section 7 concludes this work.

Our basic notations are as follows. The sets of all real, non-negative and positive numbers are denoted by $\mathbb{R}, \mathbb{R}_+, \mathbb{R}_{++}$, respectively; boldface lowercase letters, such as $\boldsymbol{x}$, represent column vectors; boldface capital letters, such as $\boldsymbol{X}$, represent matrices; we may use the notation $(x_1, \ldots, x_n)$ to represent a column vector; the superscripts $^\top$, $^{-1}$ and † denote transpose, inverse and pseudo-inverse, respectively; $\det(\boldsymbol{X})$ denotes the determinant of $\boldsymbol{X}$; $\mathrm{Diag}(x_1, \ldots, x_n)$ denotes a diagonal matrix with the ith diagonal element given by x_i; $\boldsymbol{0}$ and $\boldsymbol{1}$ denote all-zero and all-one vectors of appropriate sizes, respectively; $\boldsymbol{x} \geq \boldsymbol{0}$ means that $\boldsymbol{x}$ is element-wise non-negative, and similarly $\boldsymbol{X} \geq \boldsymbol{0}$ means that $\boldsymbol{X}$ is element-wise non-negative; $\|\cdot\|$ denotes the Euclidean norm for both vectors and matrices; $\mathrm{conv}(\boldsymbol{A}) = \{\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \mid \boldsymbol{x} \geq 0, \boldsymbol{1}^\top \boldsymbol{x} = 1\}$ denotes the convex hull of the columns of $\boldsymbol{A}$; $p(\boldsymbol{x}; \boldsymbol{\theta})$ denotes the probability distribution of a random variable $\boldsymbol{x}$, with the distribution parameter given by $\boldsymbol{\theta}$; $p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ denotes the joint probability distribution of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, with distribution parameter $\boldsymbol{\theta}$; $p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\theta})$ denotes the probability distribution of $\boldsymbol{x}$ conditioned on $\boldsymbol{y}$, with distribution parameter $\boldsymbol{\theta}$; $\mathbb{E}[\cdot]$ denotes the expectation. More notations will be defined in appropriate places.

2. Background.

2.1. Problem Statement.

The problem of interest, in its most basic form, is as follows. We are given a collection of data points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T \in \mathbb{R}^M$. We postulate that

$$\boldsymbol{y}_t = \boldsymbol{A}_0 \boldsymbol{s}_t, \tag{2.1}$$

where $\boldsymbol{A}_0 \in \mathbb{R}^{M \times N}$, with $M \geq N$; $\boldsymbol{s}_t$ is a latent (and thus unknown) variable. The latent variables lie in the unit simplex, i.e., $\boldsymbol{s}_t \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{s}_t = 1$. The matrix $\boldsymbol{A}_0$ is unknown. The problem is to recover $\boldsymbol{A}_0$ from $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$. Note that after recovering $\boldsymbol{A}_0$, we can recover $\boldsymbol{s}_t$

by solving the regression problem $\min_{\boldsymbol{s}_t \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{s}_t = 1} \|\boldsymbol{y}_t - \boldsymbol{A}_0 \boldsymbol{s}_t\|^2$. For convenience, the above problem of recovering $\boldsymbol{A}_0$ from $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ will be called SCA in the sequel.

From a geometrical viewpoint, SCA is a problem of finding the vertices of a hidden simplex from a collection of data points that lie in that simplex. To be specific, observe from (2.1) that $\boldsymbol{y}_t \in \operatorname{conv}(\boldsymbol{A}_0)$; or, in words, the data points lie in $\operatorname{conv}(\boldsymbol{A}_0)$. The set $\operatorname{conv}(\boldsymbol{A}_0)$ is a simplex under the assumption of full-column rank $\boldsymbol{A}_0$, and, by the definition of simplices, the vertices of $\operatorname{conv}(\boldsymbol{A}_0)$ are the columns of $\boldsymbol{A}_0$.[1] Hence, the $\boldsymbol{y}_t$'s are simplicially distributed data, and recovering $\boldsymbol{A}_0$ is the same as finding the vertices. Such viewpoint is commonly used in the context of hyperspectral unmixing; see, e.g., [5, 21]. From a statistical viewpoint, SCA is reminiscent of latent factor analyses such as independent component analysis (ICA). Specifically they share the common goal of exploiting the underlying natures of the latent variables, which are based upon further postulates on the statistics of the $\boldsymbol{s}_t$'s, to recover $\boldsymbol{A}_0$. Note that unit-simplex distributed $\boldsymbol{s}_t$'s do not have element-wise independent $\boldsymbol{s}_t$'s, the latter being the key postulate of ICA.

An important application of SCA is hyperspectral unmixing (HU) in remote sensing [5,21]. In fact, HU has provided strong motivations for researchers to study SCA, and one can argue that HU is central to the developments of SCA. A concise problem statement of HU is as follows. We are given a hyperspectral image taken from a scene. The image is represented by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$, where each $\boldsymbol{y}_t \in \mathbb{R}^M$ is a collection of reflectance measurements over a number of M (over a hundred) fine-resolution spectral bands at a particular pixel. Under some assumptions we may postulate that $\boldsymbol{y}_t$ follows the SCA model (2.1) [5]. In particular, each column of $\boldsymbol{A}_0$ describes the spectral response of a distinct material (or endmember), and each $\boldsymbol{s}_t$ describes the proportional distribution (or abundance) of the various materials at pixel t. The problem of HU is to identify the unknown materials and how they compose the scene, specifically, by uncovering the materials' spectral responses and the proportional distributions from the image. The problem is, in essence, SCA. The reader is refered to [5,10,12,21–24,34] for further details of HU.

SCA has strong connections with non-negative matrix factorization (NMF). To describe, consider an NMF data model $\boldsymbol{z}_t = \boldsymbol{B}\boldsymbol{c}_t$ for $t = 1, \ldots, T$, where $\boldsymbol{B} \geq \boldsymbol{0}$ and $\boldsymbol{c}_t \geq \boldsymbol{0}$ for all t. Note that $\boldsymbol{c}_t$ may not satisfy $\boldsymbol{1}^\top \boldsymbol{c}_t = 1$. Consider normalizing the data points $\boldsymbol{z}_t$'s by $\boldsymbol{y}_t = \boldsymbol{z}_t / (\boldsymbol{1}^\top \boldsymbol{z}_t)$. One can show that

$$\boldsymbol{y}_t = \sum_{i=1}^{N} \underbrace{\frac{\boldsymbol{b}_i}{\boldsymbol{1}^\top \boldsymbol{b}_i}}_{:= \boldsymbol{a}_{i,0}} \underbrace{\frac{\boldsymbol{1}^\top \boldsymbol{b}_i c_{i,t}}{\sum_{j=1}^{N} \boldsymbol{1}^\top \boldsymbol{b}_j c_{j,t}}}_{:= s_{i,t}} = \boldsymbol{A}_0 \boldsymbol{s}_t,$$

where $\boldsymbol{b}_i$ and $\boldsymbol{a}_{i,0}$ denote the ith column of $\boldsymbol{B}$ and $\boldsymbol{A}_0$, respectively, and the above defined $\boldsymbol{s}_t$ is seen to satisfy $\boldsymbol{s}_t \geq \boldsymbol{0}$ and $\boldsymbol{1}^\top \boldsymbol{s}_t = 1$; see [11, 15] and the references therein. Thus, NMF can be cast as an SCA problem by the above normalization process. It is worth noting that the application of SCA to NMF does not exploit the non-negativity of $\boldsymbol{A}_0$ in general; rather,

[1] We should recall that a set $\mathcal{S} \subseteq \mathbb{R}^m$ is called a simplex if it takes the form $\mathcal{S} = \operatorname{conv}(\boldsymbol{A})$, where $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n] \in \mathbb{R}^{m \times n}$ has $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ being affinely independent. A simplex $\operatorname{conv}(\boldsymbol{A})$ has the property that the set of vertices of $\operatorname{conv}(\boldsymbol{A})$ is $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$. Also, it should be noted that if $\boldsymbol{A}$ has full column rank, then $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ is affinely independent; the converse is not true.

it focuses on leveraging the structures of the unit-simplex-distributed $\boldsymbol{s}_t$'s to recover $\boldsymbol{A}_0$. The reader is referred to [11, 15] for details.

2.2. Simplex Volume Minimization and SISAL.

There are various ways to tackle SCA, and, among them, simplex volume minimization (SVMin) stands as a powerful approach. SVMin is built on the geometrical intuition that, if we can find a simplex that circumscribes all the data points and yields the minimum volume, that simplex is expected to be the ground-truth simplex $\mathrm{conv}(\boldsymbol{A}_0)$; see the literature [5, 11, 15, 21] for more inspirations. The problem of finding the minimum-volume data circumscribing simplex can be formulated as

$$\begin{aligned} \min_{\boldsymbol{A}\in\mathbb{R}^{M\times N}} \quad & \mathrm{vol}(\boldsymbol{A}) := (N-1)! \cdot (\det(\bar{\boldsymbol{A}}^\top \bar{\boldsymbol{A}}))^{1/2} \\ \text{s.t.} \quad & \boldsymbol{y}_t \in \mathrm{conv}(\boldsymbol{A}), \quad t = 1, \ldots, T, \end{aligned} \tag{2.2}$$

where $\mathrm{vol}(\boldsymbol{A})$ is the volume of the simplex $\mathrm{conv}(\boldsymbol{A})$ [16] (we assume that every feasible point $\boldsymbol{A}$ of (2.2) has full column rank); $\bar{\boldsymbol{A}} = [\ \boldsymbol{a}_1 - \boldsymbol{a}_N, \ldots, \boldsymbol{a}_{N-1} - \boldsymbol{a}_N\ ]$, with $\boldsymbol{a}_i$ being the ith column of $\boldsymbol{A}$. Recent studies have revealed that SVMin is more than an intuition. It is shown that, under some technical conditions which should hold for sufficiently well-spread $\boldsymbol{s}_t$'s, the optimal solution to the SVMin problem (2.2) is the ground truth $\boldsymbol{A}_0$ or its column permutation [12, 13, 19]. In other words, SVMin is equipped with provable recovery guarantees.

SISAL [4] is arguably the most popular algorithm for SVMin. Here we shed light onto how SVMin is formulated in SISAL. Bioucas-Dias, the author of SISAL, derived the SISAL formulation in an intuitively powerful way. In particular, he focused on rewriting SVMin to a form that is algorithmically friendly to handle. Assume $M = N$; this is not a problem since we can apply dimensionality reduction to project the data points to a lower dimensional space [5, 21]. SISAL starts with the following variation of writing the SVMin problem

$$\begin{aligned} \min_{\boldsymbol{A}\in\mathbb{R}^{N\times N}, \boldsymbol{S}\in\mathbb{R}^{N\times T}} \quad & |\det(\boldsymbol{A})| \\ \text{s.t.} \quad & \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S},\ \boldsymbol{S} \geq \boldsymbol{0},\ \boldsymbol{S}^\top \boldsymbol{1} = \boldsymbol{1}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{Y} = [\ \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\ ]$. In particular the above problem replaces the simplex volume $\mathrm{vol}(\boldsymbol{A}) \propto (\det(\bar{\boldsymbol{A}}^\top \bar{\boldsymbol{A}}))^{1/2}$ in problem (2.2) with $|\det(\boldsymbol{A})|$—which is easier to work with. The first key idea leading to SISAL is to perform a transformation

$$\boldsymbol{B} = \boldsymbol{A}^{-1},$$

for which we assume that every feasible point $\boldsymbol{A}$ of problem (2.3) is invertible. By $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S} \Longleftrightarrow \boldsymbol{B}\boldsymbol{Y} = \boldsymbol{S}$, we can transform problem (2.3) to

$$\begin{aligned} \min_{\boldsymbol{B}\in\mathbb{R}^{N\times N}} \quad & 1/|\det(\boldsymbol{B})| \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{Y} \geq \boldsymbol{0},\ \boldsymbol{Y}^\top \boldsymbol{B}^\top \boldsymbol{1} = \boldsymbol{1}. \end{aligned} \tag{2.4}$$

The transformed problem above is a non-convex optimization problem with convex constraints, and in this regard we should note that the constraint $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{S}$ in the SVMin problem (2.3) is non-convex. The second idea, which looks minor but will be relevant to a key aspect later, is to assume that

$$\boldsymbol{Y}^\top \boldsymbol{B}^\top \mathbf{1} = \mathbf{1} \quad \Longleftrightarrow \quad \boldsymbol{B}^\top \mathbf{1} = (\boldsymbol{Y}^\top)^\dagger \mathbf{1}. \tag{2.5}$$

Note that (2.5) is true for "$\Longrightarrow$", but (2.5) is not necessarily true for "$\Longleftarrow$" when we are given an arbitrary $\boldsymbol{Y}$. Applying (2.5), we rewrite problem (2.4) as

$$\begin{aligned} \min_{\boldsymbol{B} \in \mathbb{R}^{N \times N}} \quad & 1/|\det(\boldsymbol{B})| \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{Y} \geq \mathbf{0}, \ \boldsymbol{B}^\top \mathbf{1} = (\boldsymbol{Y}^\top)^\dagger \mathbf{1}. \end{aligned} \tag{2.6}$$

The constraint $\boldsymbol{B}\boldsymbol{Y} \geq \mathbf{0}$, albeit convex, is a number of NT linear inequalities. These linear inequalities are unstructured, meaning that there is no special structure that we can utilize to handle the inequalities efficiently. When T is large, which is often the case in practice, forcing the numerous linear inequalities to hold can be a computational challenge. The third idea, which is a compromise, is to approximate the constraint $\boldsymbol{B}\boldsymbol{Y} \geq \mathbf{0}$ by soft constraints. This gives rise to the final formulation of SISAL:

Formulation 1, SISAL Formulation by Bioucas-Dias [4]:

$$\begin{aligned} \min_{\boldsymbol{B} \in \mathbb{R}^{N \times N}} \quad & -\log(|\det(\boldsymbol{B})|) + \lambda \sum_{t=1}^{T} \sum_{i=1}^{N} \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t) \\ \text{s.t.} \quad & \boldsymbol{B}^\top \mathbf{1} = (\boldsymbol{Y}^\top)^\dagger \mathbf{1}, \end{aligned}$$

where $\text{hinge}(x) = \max\{-x, 0\}$ is a hinge function, and it serves as a penalty function for non-negative x; $\boldsymbol{b}_i$ denotes the ith row of $\boldsymbol{B}$; $\lambda > 0$ is a pre-selected penalty parameter; recall $\boldsymbol{B} = \boldsymbol{A}^{-1}$.

Our description of the formulation of SISAL is complete. Let us summarize the ideas that led to the SISAL formulation:

i) use the SVMin formulation (2.3), which considers $M = N$ and replaces the simplex volume $\text{vol}(\boldsymbol{A})$ in (2.2) with $|\det(\boldsymbol{A})|$;

ii) apply the variable transformation $\boldsymbol{B} = \boldsymbol{A}^{-1}$;

iii) assume that the equivalence in (2.5) is true;

iv) apply the soft constraint approximations, replacing the constraints $\boldsymbol{B}\boldsymbol{Y} \geq \mathbf{0}$ with a penalty function $\lambda \sum_{t=1}^{T} \sum_{i=1}^{N} \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t)$ in the objective function.

All these operations aim at simplifying the problem for efficient optimization. Interestingly it is recently shown that, except for operation iv), and under appropriate model assumptions, all the above operations lead us to the same problem as the basic SVMin formulation in (2.2).

Proposition 1 ([20]). *Suppose that the data points exactly follow the data model $\boldsymbol{y}_t = \boldsymbol{A}_0 \boldsymbol{s}_t$, with $M = N$; that $\boldsymbol{A}_0$ has full column rank; and that $\boldsymbol{S} = [\ \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\ ]$ has full row rank. Then, the SVMin problem* (2.2) *is equivalent to problem* (2.6)*. Particularly, given any feasible point $\boldsymbol{A}$ of problem* (2.2)*, (a) $\boldsymbol{A}$ is invertible; (b) the both sides of the implications of* (2.5) *are true; (c) it holds that $\text{vol}(\boldsymbol{A}) = C \cdot |\det(\boldsymbol{A})|$ for some constant C.*

2.3. Why is SISAL Successful?. There are two reasons for the success of SISAL. The first is with computational efficiency. Bioucas-Dias built a specialized algorithm for Formulation 1, which is a combination of successive convex approximation and the variable splitting augmented Lagrangian method. The result is a computationally efficient algorithm that scales well with the data size T, particularly compared to other SVMin algorithms that deal with the hard constraint $\boldsymbol{BY} \geq \boldsymbol{0}$. The second is with noise robustness. The reader may have noticed that the SISAL formulation was derived under a data model that postulates that every data point is perfectly drawn from $\boldsymbol{y}_t = \boldsymbol{A}_0\boldsymbol{s}_t$—with no noise. As it turns out, the key success of SISAL lies in the noisy case. The soft constraint approximation, which was at first introduced to avoid the hard constraint $\boldsymbol{BY} \geq \boldsymbol{0}$, provides SISAL with resilience to noise effects. It was noticed that SISAL can be robust to outlying data points, while SVMin algorithms that faithfully implement the hard constraint $\boldsymbol{BY} \geq \boldsymbol{0}$ may not. This gives SISAL a significant advantage in practice.

SISAL does have a weakness. It is not clear how the penalty parameter λ should be chosen, and usually it is manually tuned.

3. SISAL as Probabilistic SCA, and Beyond.

Intriguingly, we can provide an explanation of why SISAL works in the noisy case. The idea is to build a connection between SISAL and a probabilistic SCA framework, and this is the focus of this section.

3.1. Probabilistic SCA. To put into context, consider a noisy data model

$$\boldsymbol{y}_t = \boldsymbol{A}_0\boldsymbol{s}_t + \boldsymbol{v}_t, \quad t = 1, \ldots, T, \tag{3.1}$$

where $\boldsymbol{v}_t$ is noise. The model is accompanied with the following assumptions:

i) $\boldsymbol{A}_0$ is square and invertible;

ii) every $\boldsymbol{s}_t$ is uniformly distributed on the unit simplex; or, equivalently, every $\boldsymbol{s}_t$ follows a Dirichlet distribution with concentration parameter $\boldsymbol{1}$;

iii) every $\boldsymbol{v}_t$ is Gaussian distributed with mean zero and covariance $\sigma^2\boldsymbol{I}$;

iv) the $\boldsymbol{s}_t$'s are independent and identically distributed (i.i.d.), the $\boldsymbol{v}_t$'s are i.i.d., and the $\boldsymbol{s}_t$'s are independent of the $\boldsymbol{v}_t$'s.

Our point of departure is the maximum-likelihood (ML) estimator

$$\begin{aligned} \hat{\boldsymbol{A}} \in \arg \max_{\boldsymbol{A} \in \mathbb{R}^{N \times N}} \; & \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{y}_t; \boldsymbol{A}) \\ \text{s.t. } & \boldsymbol{A} \text{ is invertible,} \end{aligned} \tag{3.2}$$

where $p(\boldsymbol{y}; \boldsymbol{A})$ is the probability distribution of a data point $\boldsymbol{y}$ parameterized by $\boldsymbol{A}$, which will be specified shortly. The ML estimator (3.2) has been shown to possess a desirable identifiability characteristic [34]. In addition, ML estimation is deemed a principled and powerful approach for estimating $\boldsymbol{A}_0$ in the noisy case, and the same type of ML estimation is also seen in probabilistic forms of principal component analysis (PCA) and ICA [1, 17, 25, 30].

3.2. Approximating the Likelihood. The expression of $p(\boldsymbol{y}; \boldsymbol{A})$ and how we handle it hold the first key of connecting SISAL and the ML estimator. To derive $p(\boldsymbol{y}; \boldsymbol{A})$, let $p(\boldsymbol{y}, \boldsymbol{s}; \boldsymbol{A})$ be the joint distribution of a data point $\boldsymbol{y}$ and its associated latent variable $\boldsymbol{s}$ (parameterized by $\boldsymbol{A}$). From the model in (3.1) and its accompanying assumptions, $p(\boldsymbol{y},\boldsymbol{s};\boldsymbol{A})$ is given by

$$p(\boldsymbol{y},\boldsymbol{s};\boldsymbol{A}) = p(\boldsymbol{y}|\boldsymbol{s};\boldsymbol{A})p(\boldsymbol{s}), \tag{3.3}$$

$$p(\boldsymbol{y}|\boldsymbol{s};\boldsymbol{A}) = \mathcal{N}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{s},\sigma^2\boldsymbol{I}), \tag{3.4}$$

$$p(\boldsymbol{s}) = (N-1)!\cdot \mathbb{1}_{\Delta}(\boldsymbol{s}), \quad \Delta = \{\boldsymbol{s}\in\mathbb{R}^N_{++} \mid \boldsymbol{1}^\top\boldsymbol{s} = 1\}, \tag{3.5}$$

where $p(\boldsymbol{s})$ is the latent prior; $p(\boldsymbol{y}|\boldsymbol{s};\boldsymbol{A})$ is the distribution of $\boldsymbol{y}$ conditioned on $\boldsymbol{s}$ (and parameterized by $\boldsymbol{A}$); $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes a real-valued multivariate Gaussian distribution function with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$;

$$\mathbb{1}_{\mathcal{X}}(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x}\notin\mathcal{X} \\ 1 & \text{if } \boldsymbol{x}\in\mathcal{X} \end{cases}.$$

The distribution $p(\boldsymbol{y};\boldsymbol{A})$ is the marginalization of $p(\boldsymbol{y},\boldsymbol{s};\boldsymbol{A})$ over $\boldsymbol{s}$:

$$p(\boldsymbol{y};\boldsymbol{A}) = \int p(\boldsymbol{y},\boldsymbol{s};\boldsymbol{A})\mathrm{d}\mu(\boldsymbol{s}), \tag{3.6}$$

where μ is the Lebesgue measure on $\{\boldsymbol{s}\in\mathbb{R}^N \mid \boldsymbol{1}^\top\boldsymbol{s}=1\}$. At first sight, and by intuition, one may be tempted to further write (3.6) as

$$p(\boldsymbol{y};\boldsymbol{A}) = \int_{\mathbb{R}^N} p(\boldsymbol{y},\boldsymbol{s};\boldsymbol{A})\mathrm{d}\boldsymbol{s}. \tag{3.7}$$

But the correct way should be

$$p(\boldsymbol{y};\boldsymbol{A}) = \int_{\mathbb{R}^{N-1}} p(\boldsymbol{y},(\boldsymbol{s}_{1:N-1}, 1-\boldsymbol{1}^\top\boldsymbol{s}_{1:N-1});\boldsymbol{A})\mathrm{d}\boldsymbol{s}_{1:N-1},$$

where $\boldsymbol{s}_{1:N-1} = (s_1,\ldots,s_{N-1})$, and we use the relation $\boldsymbol{1}^\top\boldsymbol{s}=1$ to explicitly represent s_N by $s_N = 1-\boldsymbol{1}^\top\boldsymbol{s}_{1:N-1}$. Simply speaking, (3.7) does not consider the mathematical caveat that $\mathbb{1}_{\Delta}(\boldsymbol{s})$ is not measurable on $\mathbb{R}^N$. There is however a simple trick to get around this caveat and thereby allow us to use (3.7) (which is simpler), as we will study later.

The function in (3.6) requires us to solve an integral. Unfortunately, that integral is intractable in general. To be more precise, we do not know if there exists a simple analytical expression or a computationally efficient method to solve the integral, given an arbitrary instance of $\boldsymbol{y},\boldsymbol{A},N$. As with many scientific and engineering studies, we pursue approximations and heuristics. Firstly, we adopt a quasi latent prior

$$p(\boldsymbol{s}) \simeq C\cdot\mathbb{1}_{\hat{\Delta}}(\boldsymbol{s}), \quad \hat{\Delta} = \{\boldsymbol{s}\in\mathbb{R}^N_{++} \mid |\boldsymbol{1}^\top\boldsymbol{s}-1| < \delta/2\}, \tag{3.8}$$

where $\delta>0$ is given and is small; C is a normalizing constant. Clearly, (3.8) should closely approximate the true latent prior when δ is very small. Since the quasi latent prior (3.8) is measurable on $\mathbb{R}^N$, we can use the expression (3.7) and write

$$p(\boldsymbol{y};\boldsymbol{A}) \simeq C\int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{s},\sigma^2\boldsymbol{I})\mathbb{1}_{\hat{\Delta}}(\boldsymbol{s})\mathrm{d}\boldsymbol{s}. \tag{3.9}$$

Let $\boldsymbol{B} = \boldsymbol{A}^{-1}$. By the change of variable $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$, (3.9) can be rewritten as

$$\begin{aligned} p(\boldsymbol{y};\boldsymbol{A}) &\simeq C|\det(\boldsymbol{B})| \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{y};\boldsymbol{x},\sigma^2\boldsymbol{I})\mathbb{1}_{\hat{\Delta}}(\boldsymbol{B}\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= C|\det(\boldsymbol{B})| \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{x};\boldsymbol{y},\sigma^2\boldsymbol{I})\mathbb{1}_{\hat{\Delta}}(\boldsymbol{B}\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{3.10}$$

By another change of variable $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{y}$, we can further rewrite (3.10) as

$$\begin{aligned} p(\boldsymbol{y};\boldsymbol{A}) &\simeq C|\det(\boldsymbol{B})| \int_{\mathbb{R}^N} \mathcal{N}(\boldsymbol{v};\boldsymbol{0},\sigma^2\boldsymbol{I})\mathbb{1}_{\hat{\Delta}}(\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}))\mathrm{d}\boldsymbol{v} \\ &= C|\det(\boldsymbol{B})| \cdot \mathrm{Prob}(\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) \in \hat{\Delta}), \end{aligned} \tag{3.11}$$

where $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})$. By noting the definition of $\hat{\Delta}$ in (3.8), the probability term in (3.11) can be expressed as

$$\mathrm{Prob}(\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) \in \hat{\Delta}) = \mathrm{Prob}\left(\boldsymbol{b}_1^\top(\boldsymbol{y}+\boldsymbol{v}) > 0, \ldots, \boldsymbol{b}_N^\top(\boldsymbol{y}+\boldsymbol{v}) > 0, |\boldsymbol{1}^\top\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) - 1| < \delta/2\right), \tag{3.12}$$

where $\boldsymbol{b}_i$ denotes the ith row of $\boldsymbol{B}$. For convenience, let

$$\mathcal{E}_i = \{\boldsymbol{b}_i^\top(\boldsymbol{y}+\boldsymbol{v}) > 0\}, \quad i = 1,\ldots,N, \tag{3.13a}$$

$$\mathcal{E}_{N+1} = \{|\boldsymbol{1}^\top\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) - 1| < \delta/2\}, \tag{3.13b}$$

and write

$$\mathrm{Prob}\left(\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) \in \hat{\Delta}\right) = \mathrm{Prob}\left(\cap_{i=1}^{N+1}\mathcal{E}_i\right).$$

The following heuristic is very crucial.

Heuristic 1. *Approximate* (3.12) *by*

$$\mathrm{Prob}\left(\cap_{i=1}^{N+1}\mathcal{E}_i\right) \approx \prod_{i=1}^{N+1}\mathrm{Prob}(\mathcal{E}_i).$$

We will discuss how to make sense of Heuristic 1 in the next subsection. One can show from (3.13a) that

$$\mathrm{Prob}(\mathcal{E}_i) = \Phi\left(\frac{\boldsymbol{b}_i^\top\boldsymbol{y}}{\sigma\|\boldsymbol{b}_i\|}\right), \quad i = 1,\ldots,N,$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^x e^{-z^2/2}\mathrm{d}z$; the idea is that, for $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})$, we have $\boldsymbol{b}_i^\top(\boldsymbol{y}+\boldsymbol{v}) \sim \mathcal{N}(\boldsymbol{b}_i^\top\boldsymbol{y},\sigma^2\|\boldsymbol{b}_i\|^2)$. Also, we see from (3.13b) that

$$\mathrm{Prob}(\mathcal{E}_{N+1}) = \int_{-\delta/2}^{\delta/2}\mathcal{N}(\eta;\boldsymbol{1}^\top\boldsymbol{B}\boldsymbol{y}-1,\sigma^2\|\boldsymbol{B}^\top\boldsymbol{1}\|^2)d\eta \simeq \delta\cdot\mathcal{N}(0;\boldsymbol{1}^\top\boldsymbol{B}\boldsymbol{y}-1,\sigma^2\|\boldsymbol{B}^\top\boldsymbol{1}\|^2)$$

for a very small δ; again, the idea is that, for $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})$, we have $\boldsymbol{1}^\top\boldsymbol{B}(\boldsymbol{y}+\boldsymbol{v}) - 1 \sim \mathcal{N}(\boldsymbol{1}^\top\boldsymbol{B}\boldsymbol{y}-1,\sigma^2\|\boldsymbol{B}^\top\boldsymbol{1}\|^2)$. Putting the components together, we obtain an approximate expression of $p(\boldsymbol{y};\boldsymbol{A})$ as follows

$$p(\boldsymbol{y};\boldsymbol{A}) \approx \delta C|\det(\boldsymbol{B})| \cdot \left(\prod_{i=1}^N \Phi\left(\frac{\boldsymbol{b}_i^\top\boldsymbol{y}}{\sigma\|\boldsymbol{b}_i\|}\right)\right) \cdot \mathcal{N}(0;\boldsymbol{1}^\top\boldsymbol{B}\boldsymbol{y}-1,\sigma^2\|\boldsymbol{B}^\top\boldsymbol{1}\|^2). \tag{3.14}$$

3.3. Insights Revealed and Discussion.

Allow us to pause a moment to examine how the ML problem looks like under the likelihood approximation derived in the preceding subsection. By applying (3.14) to the ML problem (3.2), the following formulation can be shown.

Formulation 2, An Approximate Formulation of the ML Problem (3.2), Principally by Heuristic 1:

$$\min_{\boldsymbol{B}\in\mathbb{R}^{N\times N}} \quad -\log(|\det(\boldsymbol{B})|) + g(\boldsymbol{B}) - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\log\Phi\left(\frac{\boldsymbol{b}_i^\top\boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right),$$

where we recall $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-z^2/2}\mathrm{d}z$;

$$g(\boldsymbol{B}) = \log(\|\boldsymbol{B}^\top\boldsymbol{1}\|) + \frac{\|\boldsymbol{Y}^\top\boldsymbol{B}^\top\boldsymbol{1}-\boldsymbol{1}\|^2}{2\sigma^2T\|\boldsymbol{B}^\top\boldsymbol{1}\|^2}.$$

As a minor point of note for Formulation 2, we do not explicitly write down the constraint of invertible $\boldsymbol{B}$, which comes from the constraint of invertible $\boldsymbol{A}$ in the ML problem (3.2). This is because $-\log|\det(\boldsymbol{B})| = +\infty$ for non-invertible matrices, which means that the invertible matrix constraint is already taken care of.

Let us compare Formulation 2 and the SISAL formulation (Formulation 1). We see that both have penalty terms related to negative $\boldsymbol{b}_i^\top\boldsymbol{y}_t$. To better illustrate, Fig. 1 plots $-\log\Phi(x)$ and the hinge function. It is observed that $-\log\Phi(x)$ is monotone decreasing, and it gives stronger outputs as x is more negative. Hence we may see $-\log\Phi(x)$ as a penalty function for negative x, serving a similar aim as the hinge function. Moreover, the constraint $\boldsymbol{B}^\top\boldsymbol{1} = (\boldsymbol{Y}^\top)^\dagger\boldsymbol{1}$ in the SISAL formulation, which comes from $\boldsymbol{Y}^\top\boldsymbol{B}^\top\boldsymbol{1} = \boldsymbol{1}$, is seen to bear some resemblance to the penalty function g in Formulation 2. In the next subsection, we will put forth another element that will bring Formulation 2 even closer to the SISAL formulation. Some discussions are as follows.

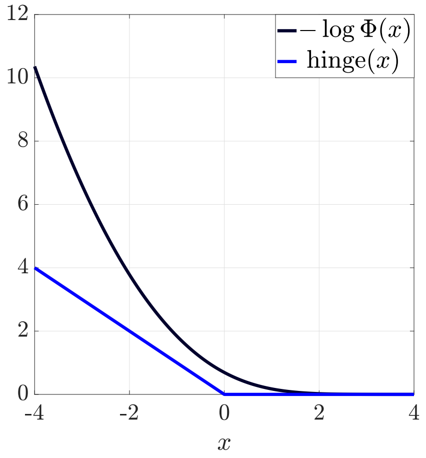


Figure 1. *Comparison of $-\log\Phi(x)$ and the hinge function.*

Remark 1. Some related work should be mentioned. In [34], we derived an approximate ML formulation similar to Formulation 2. We applied an approximation similar to Heuristic 1, but we did not use the quasi latent prior in (3.8). As a result, our previous approximate ML formulation is still not as similar to SISAL as Formulation 2.

Remark 2. We return to the question of how we can make sense of Heuristic 1. Here is our intuition: By the probability result $\text{Prob}\left(\cap_{i=1}^{N+1}\mathcal{E}_i\right) \leq \text{Prob}(\mathcal{E}_i)$ for any i, we have

$$\text{Prob}\left(\cap_{i=1}^{N+1}\mathcal{E}_i\right) \leq \left(\prod_{i=1}^{N+1} \text{Prob}(\mathcal{E}_i)\right)^{1/(N+1)}.$$

From the above inequality, we can show that

$$-\frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{y};\boldsymbol{A}) \geq -\log(|\det(\boldsymbol{B})|) + \frac{1}{N+1}\left[g(\boldsymbol{B}) - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} \log \Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right)\right], \tag{3.15}$$

which is a lower-bound approximation and sounds better in terms of being equipped with a rationale. Empirically, we however found that (3.15) tends to underestimate the negative log likelihood value $-\frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{y};\boldsymbol{A})$ quite significantly. Instead, removing the scaling $1/(N+1)$ from (3.15) would give better results. As future work, it would be interesting to analyze the approximation accuracy of Heuristic 1 or to study better approximations under the genre of Heuristic 1.

3.4. Bringing SISAL and ML Closer. We start with an assumption that does not seem to make sense at first. Let

$$\boldsymbol{p} = \boldsymbol{A}_0^{-\top}\boldsymbol{1},$$

and *suppose* that we know $\boldsymbol{p}$. Consider the following modified ML problem

$$\begin{aligned} \max_{\boldsymbol{A}\in\mathbb{R}^{N\times N}} \ & \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{y}_t;\boldsymbol{A}) \\ \text{s.t.} \ & \boldsymbol{A}^{-\top}\boldsymbol{1} = \boldsymbol{p}, \quad \boldsymbol{A} \text{ is invertible,} \end{aligned} \tag{3.16}$$

wherein we include our prior information of $\boldsymbol{p}$ to better guide the estimation. By applying the preceding likelihood approximation to problem (3.16) (or by adding the constraint $\boldsymbol{A}^{-\top}\boldsymbol{1} = \boldsymbol{p}$ to Formulation 2), we have the following formulation.

Formulation 3, An Approximate Formulation of the modified ML Problem (3.16), Principally by Heuristic 1:

$$\begin{aligned} \min_{\boldsymbol{B}\in\mathbb{R}^{N\times N}} \ & -\log(|\det(\boldsymbol{B})|) - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} \log \Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right) \\ \text{s.t.} \ & \boldsymbol{B}^\top\boldsymbol{1} = \boldsymbol{p}. \end{aligned}$$

Formulation 3 is very similar to the SISAL formulation (Formulation 1) if $\boldsymbol{p} = (\boldsymbol{Y}^\top)^\dagger \mathbf{1}$. In fact, we have this surprising result.

Fact 1 ([20]). *Suppose that the data points $\boldsymbol{y}_t$'s follow the noiseless model $\boldsymbol{y}_t = \boldsymbol{A}_0 \boldsymbol{s}_t$ (with $M = N$); that $\boldsymbol{A}_0$ has full column rank; and that $\boldsymbol{S} = [\ \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\ ]$ has full row rank. Then,*

$$(\boldsymbol{Y}^\top)^\dagger \mathbf{1} = \boldsymbol{A}_0^{-\top} \mathbf{1}.$$

Fact 1 was shown in [20], and we shall not repeat the proof. Rather, we are interested in its extension to the noisy case.

Fact 2. *Suppose that the data points $\boldsymbol{y}_t$'s follow the model in (3.1) and the accompanying assumptions. Let $\boldsymbol{\mu}_y = \mathbb{E}[\boldsymbol{y}_t]$ and $\boldsymbol{R}_{yy} = \mathbb{E}[\boldsymbol{y}_t \boldsymbol{y}_t^\top]$ be the mean and correlation matrix of $\boldsymbol{y}_t$, respectively. Then,*

$$(\boldsymbol{R}_{yy} - \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\mu}_y = \boldsymbol{A}_0^{-\top} \mathbf{1}.$$

Proof of Fact 2: Let $\boldsymbol{R}_{ss} = \mathbb{E}[\boldsymbol{s}_t \boldsymbol{s}_t^\top]$, $\boldsymbol{\mu}_s = \mathbb{E}[\boldsymbol{s}_t]$. It can be verified that $\boldsymbol{R}_{ss}$ is positive definite. Also, from the data model (3.1), we can show that

$$\boldsymbol{R}_{yy} = \boldsymbol{A}_0 \boldsymbol{R}_{ss} \boldsymbol{A}_0^\top + \sigma^2 \boldsymbol{I}, \quad \boldsymbol{\mu}_y = \boldsymbol{A}_0 \boldsymbol{\mu}_s.$$

It follows that

$$(\boldsymbol{R}_{yy} - \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\mu}_y = (\boldsymbol{A}_0 \boldsymbol{R}_{ss} \boldsymbol{A}_0^\top)^{-1} \boldsymbol{A}_0 \boldsymbol{\mu}_s = \boldsymbol{A}_0^{-\top} \boldsymbol{R}_{ss}^{-1} \boldsymbol{\mu}_s.$$

It can be shown that $\boldsymbol{R}_{ss}^{-1} \boldsymbol{\mu}_s = \mathbf{1}$. Specifically,

$$\boldsymbol{R}_{ss} \mathbf{1} = \mathbb{E}[\boldsymbol{s}_t \underbrace{\boldsymbol{s}_t^\top \mathbf{1}}_{=1}] = \mathbb{E}[\boldsymbol{s}_t] = \boldsymbol{\mu}_s.$$

The proof is complete. Note that this result also applies to a more general case wherein $\boldsymbol{s}_t$ follows a (and possibly non-uniform) Δ-supported distribution with positive definite $\boldsymbol{R}_{ss}$. ■

Fact 2 provides us with an implication that, in practice, we can estimate $\boldsymbol{p}$ by

$$\hat{\boldsymbol{p}} = (\hat{\boldsymbol{R}}_{yy} - \sigma^2 \boldsymbol{I})^{-1} \hat{\boldsymbol{\mu}}_y, \quad \hat{\boldsymbol{R}}_{yy} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t^\top, \quad \hat{\boldsymbol{\mu}}_y = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{y}_t. \tag{3.17}$$

Our final touch is to explain how the negative penalty terms in Formulation 3 and the SISAL formulation are related. We start from the direction of Formulation 3. Consider the following result.

Fact 3. **([33], [27, footnote 1])** *It holds that $\Phi(x) \leq \frac{1}{2} e^{\sqrt{\frac{2}{\pi}} x}$. Also, as a direct consequence,*

$$-\log \Phi(x) \geq -\log \left(\max \left\{ \frac{1}{2} e^{\sqrt{\frac{2}{\pi}} x}, 1 \right\} \right) = \max \left\{ \log(2) - \sqrt{\frac{2}{\pi}} x, 0 \right\}.$$

Using Fact 3, the penalty terms of Formulation 3 can be approximated by

$$
\begin{aligned}
-\log\Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right) &\geq \max\left\{\log(2) - \sqrt{\frac{2}{\pi}}\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}, 0\right\} \\
&\geq \max\left\{-\sqrt{\frac{2}{\pi}}\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}, 0\right\} \\
&= \sqrt{\frac{2}{\pi}}\frac{1}{\sigma\|\boldsymbol{b}_i\|}\mathrm{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t).
\end{aligned}
\tag{3.18}
$$

The normalizing term $\|\boldsymbol{b}_i\|$ is hard to deal with. By pretending as if $\|\boldsymbol{b}_i\|$ were a constant, and by setting $\sqrt{\frac{2}{\pi}}\frac{1}{\sigma\|\boldsymbol{b}_i\|T} = \lambda$ for some pre-selected $\lambda > 0$, we have

$$
-\frac{1}{T}\log\Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right) \approx \lambda \cdot \mathrm{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t). \tag{3.19}
$$

Now, we are ready to draw our main conclusion: *SISAL can be explained as an approximation of the ML estimator* (3.16). In particular, the connection is made by applying Fact 1 and (3.19) to Formulation 3.

3.5. A Hinge-Square Variant of SISAL.

The explanation of SISAL as an approximate ML estimator in the preceding subsection gives us a new insight, namely, that the hinge function serves as a surrogate of the penalty function $-\log\Phi(x)$ from the ML viewpoint. In that regard, we can choose a different surrogate of $-\log\Phi(x)$. From Fig. 1 we see that, as x becomes more negative, the hinge function is a poor approximation of $-\log\Phi(x)$. Consider the following result.

Fact 4. **(Chernoff bound; see, e.g., [33])** *It holds that, for $x \leq 0$, $\Phi(x) \leq \frac{1}{2}e^{-x^2/2}$. Also, as a direct consequence, we may approximate*

$$
-\log\Phi(x) \approx -\log\left(\frac{1}{2}e^{-\max\{-x,0\}^2/2}\right) = \log(2) + \frac{1}{2}\mathrm{hinge}(x)^2.
$$

Fig. 2 compares the above surrogate and $-\log\Phi(x)$. We see that this new surrogate approximates $-\log\Phi(x)$ better for negative x. By approximating

$$
-\frac{1}{T}\log\Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right) \approx \lambda \cdot \mathrm{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t)^2 + \mathrm{constant}, \tag{3.20}
$$

as before, we have the following variant of SISAL.

Formulation 4, H^2-SISAL; a Chernoff bound-based heuristic of the approximate ML problem in Formulation 3, or a hinge-square variant of SISAL in Formulation 1:

$$\begin{aligned} \min_{\boldsymbol{B}\in\mathbb{R}^{N\times N}} \quad & -\log(|\det(\boldsymbol{B})|) + \lambda \sum_{t=1}^{T}\sum_{i=1}^{N} \mathrm{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t)^2 \\ \text{s.t.} \quad & \boldsymbol{B}^\top \boldsymbol{1} = \boldsymbol{p}, \end{aligned}$$

where $\lambda > 0$ is a pre-selected penalty parameter.

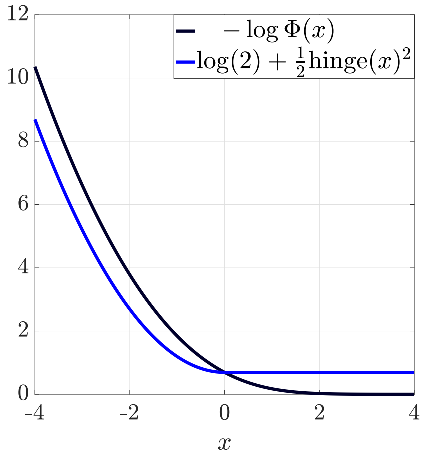


Figure 2. *Comparison of* $-\log\Phi(x)$ *and a hinge-square based function.*

Observe that the difference between Formulation 4 and the SISAL formulation (Formulation 1) is that the former puts a square on the hinge function. From an optimization viewpoint, this H^2-SISAL formulation has the advantage that the hinge-square penalty terms, as well as the whole objective function, are continuously differentiable.

4. SISAL as an Algorithm, and More.

Having explored the formulation aspects with SISAL, we turn to the algorithmic aspects. To facilitate our subsequent development, let us introduce some notations. Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be an extended real-valued function. We denote $\operatorname{dom} f = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) < +\infty\}$ as the domain of f; $\nabla f(\boldsymbol{x})$ as the gradient of f (when f is differentiable at $\boldsymbol{x}$);

$$\mathrm{prox}_f(\boldsymbol{x}) \in \arg\min_{\boldsymbol{z}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{x}\|^2 + f(\boldsymbol{x})$$

as a proximal operator associated with f. We also denote $\langle\cdot,\cdot\rangle$ as the inner product;

$$\Pi_{\mathcal{X}}(\boldsymbol{x}) \in \arg\min_{\boldsymbol{z}\in\mathcal{X}} \|\boldsymbol{z}-\boldsymbol{x}\|^2$$

as a projection of $\boldsymbol{x}$ onto a closed set $\mathcal{X} \subseteq \mathbb{R}^n$;

$$\mathbb{I}_{\mathcal{X}}(\boldsymbol{x}) = \begin{cases} +\infty & \text{if } \boldsymbol{x} \notin \mathcal{X} \\ 0 & \text{if } \boldsymbol{x} \in \mathcal{X} \end{cases}$$

as the indicator function associated with $\mathcal{X}$. Furthermore, we call f to have Lipschitz continuous gradient on $\mathcal{X}$ if ∇f is Lipschitz continuous on $\mathcal{X}$; i.e., there exists $\alpha > 0$ such that $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq \alpha \|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.

4.1. The SISAL Algorithm.

To describe the algorithm used in SISAL, we start with describing the basic natures of the SISAL problem. Recall from Formulation 1 the SISAL problem:

$$\min_{\boldsymbol{B} \in \mathbb{R}^{N \times N}, \boldsymbol{B}^\top \boldsymbol{1} = \boldsymbol{p}} \; f(\boldsymbol{B}) = \underbrace{- \log |\det(\boldsymbol{B})|}_{:= f_0(\boldsymbol{B})} + \lambda \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t), \tag{4.1}$$

where $\boldsymbol{p} = (\boldsymbol{Y}^\top)^\dagger \boldsymbol{1}$. The problem is non-convex and non-smooth: the second term of f, which has the hinge function involved, is convex and non-differentiable; f_0 is non-convex and continuously differentiable on its domain $\text{dom}\, f_0$; $\text{dom}\, f_0$ is the set of all invertible matrices on $\mathbb{R}^{N \times N}$; f_0 does *not* have Lipschitz continuous gradient on $\text{dom}\, f_0$. If one wants to find an off-the-shelf optimization method that offers some form of guarantee of finding a stationary point of problem (4.1), that will not be immediately obvious. The non-triviality comes in two ways:

1. Implementation: One can actually apply an off-the-shelf method from the recent advances of optimization, particularly, first-order optimization. Take the proximal gradient method as an example. One needs to choose the step size, which is typically guided by the Lipschitz constant of ∇f_0. The absence of Lipschitz continuous ∇f_0 in our problem necessitates a different strategy to deal with the problem. Also, the problem domain, the set of all invertible matrices, is non-standard at first sight.
2. Theory: The Lipschitz continuity of ∇f_0 is needed in most convergence proofs. Again, we do not have Lipschitz continuous ∇f_0.

Back to 2009, Bioucas-Dias dealt with the problem by successive convex approximation. The ideas are to form a quadratic approximation of f_0 at a given point $\tilde{\boldsymbol{B}} \in \text{dom}\, f_0$

$$f(\boldsymbol{B}) \approx f_0(\tilde{\boldsymbol{B}}) + \langle \nabla f_0(\tilde{\boldsymbol{B}}), \boldsymbol{B} - \tilde{\boldsymbol{B}} \rangle + \frac{\mu}{2} \|\boldsymbol{B} - \tilde{\boldsymbol{B}}\|^2 := g_\mu(\boldsymbol{B}, \tilde{\boldsymbol{B}}),$$

for some $\mu > 0$; and to solve, iteratively,

$$\boldsymbol{B}^{k+1} = \arg \min_{\boldsymbol{B} \in \mathbb{R}^{N \times N}, \boldsymbol{B}^\top \boldsymbol{1} = \boldsymbol{p}} g_{\mu_k}(\boldsymbol{B}, \boldsymbol{B}^k) + \lambda \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t), \quad k = 0, 1, 2, \cdots \tag{4.2}$$

for some $\mu_k > 0$ for all k. The problems encountered in (4.2) are convex (in fact, strictly convex). Bioucas-Dias solved these problems by a variable splitting augmented Lagrangian algorithm, which is now more popularly known as the alternating direction method of multipliers (ADMM). That ADMM algorithm exploits the problem structure of (4.2) and is computationally efficient. But (4.2) has a caveat: depending on how μ_k is chosen, a new iterate $\boldsymbol{B}^{k+1}$ may not be invertible; and when that happens, the successive convex optimization in (4.2) will crash.

Algorithm 4.1 is the actual form of the SISAL algorithm. Intuitively, we expect that there should exist a $\theta_k \in (0, 1]$, no matter how small it may be, such that $\boldsymbol{B}^{k+1} = \boldsymbol{B}^k + \theta_k (\bar{\boldsymbol{B}}^k - \boldsymbol{B}^k)$ remains invertible. As mentioned, empirical studies suggest that SISAL works. This leads to an intriguing, and previously unanswered, basic question: Does Algorithm 4.1 have any guarantee of finding a stationary point of problem (4.1)?

Algorithm 4.1 SISAL by Bioucas-Dias [4], successive convex optimization for Formulation 1

```
given: an invertible starting point $\boldsymbol{B}^0$ and a constant $\mu > 0$
$k = 0$
repeat
    $\bar{\boldsymbol{B}}^k = \arg \min_{\boldsymbol{B} \in \mathbb{R}^{N \times N}, \boldsymbol{B}^\top \boldsymbol{1} = \boldsymbol{p}} g_\mu(\boldsymbol{B}, \boldsymbol{B}^k) + \lambda \sum_{t=1}^T \sum_{i=1}^N \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t)$, by ADMM (see [4])
    find a $\theta_k \in (0,1]$ such that $f(\boldsymbol{B}^k + \theta_k(\bar{\boldsymbol{B}}^k - \boldsymbol{B}^k)) \leq f(\boldsymbol{B}^k)$, by line search
    $\boldsymbol{B}^{k+1} = \boldsymbol{B}^k + \theta_k(\bar{\boldsymbol{B}}^k - \boldsymbol{B}^k)$
    $k = k + 1$
until a stopping rule is satisfied
output: $\boldsymbol{B}^k$
```

4.2. Line Search-Based Proximal Gradient Method. Our study found that the optimization framework by Bonettini *et al.* [6] can be used to answer the question. To put into context, consider a problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) := f_0(\boldsymbol{x}) + f_1(\boldsymbol{x}), \tag{4.3}$$

where f_0 is continuously differentiable on its domain $\operatorname{dom} f_0$; $\operatorname{dom} f_0$ is open; f_1 is convex, proper, lower semicontinuous, and bounded from below; $\operatorname{dom} f_1$ is closed and nonempty. For this problem, a point $\bar{\boldsymbol{x}} \in \operatorname{dom} f$ is called a stationary point of problem (4.3) if the directional derivative of f, defined as $f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{t \downarrow 0}(f(\boldsymbol{x} + t\boldsymbol{d}) - f(\boldsymbol{x}))/t$, satisfies $f'(\bar{\boldsymbol{x}}; \boldsymbol{d}) \geq 0$ for all $\boldsymbol{d} \in \mathbb{R}^n$. To describe the method, let

$$h_\mu(\boldsymbol{z}, \boldsymbol{x}) = \langle \nabla f_0(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{z} - \boldsymbol{x}\|^2 + f_1(\boldsymbol{z}) - f_1(\boldsymbol{x}), \qquad \mu > 0.$$

Consider the following line search-based proximal gradient (LSB-PG) method: given $\beta \in (0,1)$, $\boldsymbol{x}^0 \in \operatorname{dom} f$, recursively compute

$$\boldsymbol{y}^k = \arg \min_{\boldsymbol{z} \in \mathbb{R}^n} h_{\mu_k}(\boldsymbol{z}, \boldsymbol{x}^k) = \operatorname{prox}_{\mu_k^{-1} f_1}(\boldsymbol{x}^k - \mu_k^{-1} \nabla f_0(\boldsymbol{x}^k)), \quad \text{for some } \mu_k > 0, \tag{4.4}$$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \theta_k(\boldsymbol{y}^k - \boldsymbol{x}^k), \tag{4.5}$$

for $k = 0, 1, 2, \cdots$, where $\theta_k \in (0,1]$ is chosen such that

$$f(\boldsymbol{x}^k + \theta_k(\boldsymbol{y}^k - \boldsymbol{x}^k)) \leq f(\boldsymbol{x}^k) + \beta \theta_k h_{\mu_k}(\boldsymbol{y}^k, \boldsymbol{x}^k). \tag{4.6}$$

To be precise, we use an Armijo line search rule to find θ_k: find the smallest non-negative integer j such that

$$f(\boldsymbol{x}^k + \delta^j(\boldsymbol{y}^k - \boldsymbol{x}^k)) \leq f(\boldsymbol{x}^k) + \beta \delta^j h_{\mu_k}(\boldsymbol{y}^k, \boldsymbol{x}^k). \tag{4.7}$$

for some given $\delta \in (0,1)$, and then choose $\theta_k = \delta^j$. It is worth noting that (4.6) is a sufficient decrease condition with the objective value, since $h_{\mu_k}(\boldsymbol{y}^k, \boldsymbol{x}^k) \leq 0$. Also, the framework in [6] is much more general than the LSB-PG, and here we reduce the framework to the above minimal form which is enough to answer our question.

The LSB-PG method is equipped with the following stationarity guarantee.

Proposition 2 (a rephrased, simplified, version of Corollary 3.1 in [6]). *Consider problem* (4.3) *and its associated LSB-PG method in* (4.4)–(4.7)*. Suppose* $\mathrm{dom}\, f_0 \supseteq \mathrm{dom}\, f_1$*. Also, assume that* $\{\mu_k\} \subset [\mu_{\min}, \mu_{\max}]$ *for some* $0 < \mu_{\min} \leq \mu_{\max} < +\infty$*, and that* $\{\boldsymbol{x}_k\}$ *has a limit point. Then any limit point of* $\{\boldsymbol{x}_k\}$ *is a stationary point of problem* (4.3)*.*

As we will discuss in the next subsection, the application of the LSB-PG method to the SISAL problem does not have $\mathrm{dom}\, f_0 \supseteq \mathrm{dom}\, f_1$ satisfied. This led us to rework the whole proof to see if the above assumption can be relaxed. The answer, fortunately, is yes.

Corollary 1. *The same stationarity result in Proposition 2 holds if we replace* $\mathrm{dom}\, f_0 \supseteq \mathrm{dom}\, f_1$ *by* $\mathrm{dom}\, f_0 \cap \mathrm{dom}\, f_1 \neq \emptyset$*. As a comment, the assumption of open* $\mathrm{dom}\, f_0$ *plays a crucial role.*

The proof of Corollary 1 is a meticulous re-examination of the whole proof of Corollary 3.1 in [6], including the proof of the theorems and propositions that precede it. We shall omit the proof. The following remark describes the unique aspect of proving Corollary 1, and the reader may choose to skip it and jump to the next subsection for the application of Corollary 1 to the SISAL problem.

Remark 3. We discuss the key proof differences of Proposition 2 and Corollary 1. In the proof, an important issue is to show that there exists a $\theta_k \in (0, 1]$ such that the sufficient decrease condition (4.6) holds. To achieve the latter, a prerequisite is to ensure $\boldsymbol{x}^{k+1} \in \mathrm{dom}\, f_0$. One can readily see from (4.4)–(4.5) that $\boldsymbol{y}^k \in \mathrm{dom}\, f_1$, and then $\boldsymbol{x}^{k+1} \in \mathrm{dom}\, f_1$ (due to the convexity of $\mathrm{dom}\, f_1$). For the case of $\mathrm{dom}\, f_0 \supseteq \mathrm{dom}\, f_1$, or Proposition 2, we automatically get $\boldsymbol{x}^{k+1} \in \mathrm{dom}\, f_0$. For the case of $\mathrm{dom}\, f_0 \not\supseteq \mathrm{dom}\, f_1$, or Corollary 1, we need to leverage on the assumption of open $\mathrm{dom}\, f_0$. Since $\mathrm{dom}\, f_0$ is open, there exists $\epsilon_k > 0$ such that, for any $\boldsymbol{u} \in \mathbb{R}^n$ with $\|\boldsymbol{u}\| \leq \epsilon$, we have $\boldsymbol{x}^k + \boldsymbol{u} \in \mathrm{dom}\, f_0$. This implies that there must exist a $\theta_k > 0$, no matter how small it is, such that $\boldsymbol{x}^k + \theta_k(\boldsymbol{y}^k - \boldsymbol{x}^k) \in \mathrm{dom}\, f_0$. The above is the distinct part of the proof of Corollary 1.

4.3. Stationarity Guarantee of SISAL.

Now we apply the framework in the preceding subsection to the SISAL problem. Let

$$\begin{aligned} f_0(\boldsymbol{B}) &= \; -\log|\det(\boldsymbol{B})|, \\ f_1(\boldsymbol{B}) &= \lambda \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} \mathrm{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t) + \mathbb{I}_{\mathcal{B}}(\boldsymbol{B}), \quad \mathcal{B} = \; \{\boldsymbol{B} \in \mathbb{R}^{N \times N} \mid \boldsymbol{B}^\top \mathbf{1} = \boldsymbol{p}\}, \end{aligned}$$

and let $\mu_k = \mu$ for some pre-selected constant $\mu > 0$. We observe that the SISAL algorithm in Algorithm 4.1 is very similar to the LSB-PG method in (4.4)–(4.7), with β being nearly zero. Or, more specifically, if we modify Algorithm 4.1 by changing the line search in Step 5 to the Armijo rule in (4.7), the algorithm is, faithfully, an instance of the LSB-PG method. To answer the question of stationarity guarantees, note that $\mathrm{dom}\, f_0$ is the set of all invertible matrices on $\mathbb{R}^{N \times N}$, while $\mathrm{dom}\, f_1 = \mathcal{B}$. Clearly, we have $\mathrm{dom}\, f_0 \not\supseteq \mathrm{dom}\, f_1$, and Proposition 2 is not applicable. Corollary 1 is applicable if $\mathrm{dom}\, f_0$ is open. In fact, it is known in topology that the set of invertible matrices is open.[2] Let us conclude. By Corollary 1, the SISAL

[2] For the reader's interest, here is a simple proof by matrix analysis. Let $\mathcal{S}$ be the set of invertible matrices on $\mathbb{R}^{N \times N}$. Let $\boldsymbol{X} \in \mathcal{S}$, and let $\sigma_1 \geq \cdots \geq \sigma_N > 0$ be its singular values. Let $\epsilon > 0$. Let $\boldsymbol{Y}$ be any matrix such that $\|\boldsymbol{X} - \boldsymbol{Y}\| \leq \epsilon$, and let $d_1 \geq \cdots \geq d_N \geq 0$ be its singular values. By the singular value inequality $\|\boldsymbol{X} - \boldsymbol{Y}\|^2 \geq \sum_{i=1}^{N} |\sigma_i - d_i|^2$, and letting $\epsilon = \sigma_N/2$, one can verify that $d_N \geq \sigma_N/2 > 0$.

algorithm, upon a minor modification with its line search rule, is equipped with a stationarity guarantee.

4.4. Application to H^2-SISAL and Formulation 3. It is exciting to point out that we can also use the LSB-PG method in Section 4.2 to deal with the H^2-SISAL problem in Formulation 4. Specifically we choose

$$f_0(\boldsymbol{B}) = -\log(|\det(\boldsymbol{B})|) + \lambda \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} \text{hinge}(\boldsymbol{b}_i^\top \boldsymbol{y}_t)^2, \quad f_1(\boldsymbol{B}) = \mathbb{I}_{\mathcal{B}}(\boldsymbol{B}); \tag{4.8}$$

note that we put the (continuously differentiable) hinge-square penalty term to f_0, which is different compared to SISAL. The resulting LSB-PG method has the proximal operation (4.4) reduced to

$$\bar{\boldsymbol{B}}^k = \text{prox}_{\mu_k^{-1} f_1}(\boldsymbol{B}^k - \mu_k^{-1} \nabla f_0(\boldsymbol{B}^k)) = \Pi_{\mathcal{B}}(\boldsymbol{B}^k - \mu_k^{-1} \nabla f_0(\boldsymbol{B}^k)),$$

which has a simple closed form and is cheap to compute. We should recall that the proximal operation in SISAL has no closed form and requires us to call a solver (ADMM). We take advantage of the computational efficiency of the proximal operation by considering the following rule of choosing μ_k: find the smallest non-negative integer j such that

$$f(\bar{\boldsymbol{B}}^{k,j}) \le f(\boldsymbol{B}^k) + \beta h_{\nu c^j}(\bar{\boldsymbol{B}}^{k,j}, \boldsymbol{B}^k), \tag{4.9a}$$

$$\bar{\boldsymbol{B}}^{k,j} = \Pi_{\mathcal{B}}(\boldsymbol{B}^k - (\nu c^j)^{-1} \nabla f_0(\boldsymbol{B}^k)), \tag{4.9b}$$

for some given $\nu > 0, c > 1$, and then choose $\mu_k = \nu c^j$. Consequently, the sufficient decrease condition (4.6) will be satisfied for $\theta_k = 1$, and we can simply set $\theta_k = 1$, $\boldsymbol{B}^{k+1} = \bar{\boldsymbol{B}}^{k,j}$. Note that this is a typical scheme in proximal gradient methods (see, e.g., [2]), and (4.9) is popularly called the backtracking line search. We should also mention that the above LSB-PG scheme is identical to the projected gradient method, with a suitably chosen step size. By Corollary 1, this LSB-PG scheme is equipped with a stationarity guarantee under the assumption that the μ_k's found by the backtracking line search are bounded.

Our actual algorithm, shown in Algorithm 4.2, is an extrapolated variant of the above scheme. Note that, by choosing $\alpha_k = 0$, Algorithm 4.2 reduces to the previous LSB-PG

Algorithm 4.2 H^2-SISAL, an extrapolated proximal gradient scheme for Formulation 4

```
1: given: an invertible starting point $\boldsymbol{B}^0$; a constant $\beta \in (0,1)$; and an extrapolation
   sequence $\{\alpha_k\}$, typically the FISTA sequence [2]
2: $k = 0$, $\boldsymbol{B}^{-1} = \boldsymbol{B}^0$
3: repeat
4:     $\boldsymbol{B}_{\text{ex}}^k = \boldsymbol{B}^k + \alpha_k(\boldsymbol{B}^k - \boldsymbol{B}^{k-1})$
5:     $\boldsymbol{B}^{k+1} = \Pi_{\mathcal{B}}(\boldsymbol{B}_{\text{ex}}^k - \mu_k^{-1}\nabla f_0(\boldsymbol{B}_{\text{ex}}^k))$, where $\mu_k$ is chosen such that $f(\boldsymbol{B}^{k+1}) \le f(\boldsymbol{B}_{\text{ex}}^k)+$
6:     $\beta h_{\mu_k}(\boldsymbol{B}^{k+1}, \boldsymbol{B}_{\text{ex}}^k)$, done by the backtracking line search (4.9); $f_0$ is given in (4.8)
7:     $k = k + 1$
8: until a stopping rule is satisfied
9: output: $\boldsymbol{B}^k$
```

scheme. Our consideration is more from the practical side. The LSB-PG framework does not cover the extrapolated variant, and hence it is not known if Algorithm 4.2 is equipped with stationarity guarantees. On the other hand, we want to leverage on the merits of extrapolation demonstrated in prior works. It is known that, when f_0 is convex and has Lipschitz continuous gradient, the extrapolated proximal gradient method can lead to faster convergence rates than the proximal gradient method, both provably and empirically [3]; and that, when f_0 is non-convex and has Lipschitz continuous gradient, the extrapolated proximal gradient method is shown to yield some stationarity guarantee [14, 37], and similar methods were empirically found to lead to faster convergence speeds in some applications [12, 27, 35, 36]. Our empirical experience with Algorithm 4.2 is good in terms of runtime speed and stability.

We should further note that all the developments in this subsection apply to the approximate ML problem in Formulation 3; change

$$f_0(\boldsymbol{B}) = -\log(|\det(\boldsymbol{B})|) - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\log\Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right)$$

(this f_0 can be shown to be continuously differentiable on the set of all invertible matrices). Unfortunately, by our numerical experience, the adaptation of Algorithm 4.2 (with or without extrapolation) to Formulation 3 is not promising: its convergence tends to be slow; and numerical instability could happen, if not careful enough. The culprit is most likely the normalizing terms $\|\boldsymbol{b}_i\|$: the term $1/\|\boldsymbol{b}_i\|$ becomes very large for small $\|\boldsymbol{b}_i\|$, and the occurrence of such event can cause numerical instability. These setbacks drove us to rethink our strategy for dealing with Formulation 3.

5. Probabilistic SISAL via Inexact Block Coordinate Descent.

In this section we devise an algorithm for tackling the approximate ML problem in Formulation 3, with a focus on practicality and efficiency in our design.

5.1. Reformulation and Inexact Block Coordinate Descent.

As mentioned previously, the normalizing terms $\|\boldsymbol{b}_i\|$ in the objective function are troublesome. We deal with them by considering the change of variable

$$\boldsymbol{B} = \boldsymbol{D}\boldsymbol{C}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{c}_1^\top \\ \vdots \\ \boldsymbol{c}_N^\top \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{bmatrix}, \quad d_i > 0, \quad \boldsymbol{c}_i \in \mathcal{U} := \{\boldsymbol{c} \in \mathbb{R}^N \mid \|\boldsymbol{c}\| = 1\}, \ \forall i.$$

Applying the above transformation to Formulation 3 leads to the following reformulation

$$\begin{aligned} \min_{\boldsymbol{C}\in\mathbb{R}^{N\times N}, \boldsymbol{d}\in\mathbb{R}^N} \quad & -\log|\det(\boldsymbol{C})| - \sum_{i=1}^{N}\log d_i - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\log\Phi(\boldsymbol{c}_i^\top \bar{\boldsymbol{y}}_t) \\ \text{s.t.} \quad & \boldsymbol{C}^\top \boldsymbol{d} = \boldsymbol{p}, \ \boldsymbol{C} \in \mathcal{U}^N, \end{aligned} \tag{5.1}$$

where, for convenience, we denote $\bar{\boldsymbol{y}}_t = \boldsymbol{y}_t/\sigma$, $\mathcal{U}^N = \{\boldsymbol{C} = [\ \boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\ ]^\top \mid \boldsymbol{c}_i \in \mathcal{U}\ \forall i\}$, and $\boldsymbol{d} = (d_1, \ldots, d_N)$; note dom$(-\log) = \mathbb{R}_{++}$. The upshot of the reformulation in (5.1) is that the normalizing terms disappear. The new challenges are that we are now faced with unit modulus constraints, and handling both the equality constraint $\boldsymbol{C}^\top \boldsymbol{d} = \boldsymbol{p}$ and the unit modulus constraints is difficult. We make a compromise by considering a penalized alternation of problem (5.1)

$$\min_{\boldsymbol{C}\in\mathcal{U}^N, \boldsymbol{d}\in\mathbb{R}^N} F_\eta(\boldsymbol{C},\boldsymbol{d}) := -\log|\det(\boldsymbol{C})| - \sum_{i=1}^N \log d_i - \frac{1}{T}\sum_{t=1}^T\sum_{i=1}^N \log\Phi(\boldsymbol{c}_i^\top \bar{\boldsymbol{y}}_t) + \eta\|\boldsymbol{C}^\top\boldsymbol{d} - \boldsymbol{p}\|^2 \tag{5.2}$$

for a given penalty parameter $\eta > 0$ that is presumably large. Observe that F_η is convex in $\boldsymbol{d}$, and non-convex in $\boldsymbol{C}$.

We employ a block coordinate descent (BCD) strategy to handle problem (5.2). The first layer of our algorithm is shown in Algorithm 5.1. We minimize F_η over $\boldsymbol{C}$ and $\boldsymbol{d}$ in an alternating fashion. To be more precise, the minimization F_η over $\boldsymbol{C} \in \mathcal{U}^N$ is only approximate since the problem is non-convex. Moreover, we gradually increase η. By experience, graduating increasing η is better than applying a large fixed η. The second layer of our design deals with the computations of the coordinate minimizers in Steps 5–6 of Algorithm 5.1, which is detailed next.

Algorithm 5.1 Pr-SISAL, an inexact BCD algorithm for the altered problem (5.2) of Formulation 3

```
 1: given: an invertible starting point $\boldsymbol{B}^0$, a starting penalty value $\eta > 0$, $c > 1$, and a rule
    for increasing $\eta$
 2: $k = 0$, $\boldsymbol{d}^0 = (\|\boldsymbol{b}_1^0\|, \ldots, \|\boldsymbol{b}_N^0\|)$, $\boldsymbol{C}^0 = [\ \boldsymbol{b}_1^0/d_1^0, \ldots, \boldsymbol{b}_N^0/d_N^0\ ]^\top$
 3: repeat
 4:     repeat
 5:         $\boldsymbol{d}^{k+1} = \arg\min_{\boldsymbol{d}\in\mathbb{R}^N} F_\eta(\boldsymbol{C}^k, \boldsymbol{d})$ by Algorithm 5.2 with $\boldsymbol{d}^k$ as the starting point
 6:         $\boldsymbol{C}^{k+1} \approx \arg\min_{\boldsymbol{C}\in\mathcal{U}^N} F_\eta(\boldsymbol{C}, \boldsymbol{d}^{k+1})$ by Algorithm 5.3 with $\boldsymbol{C}^k$ as the starting point
 7:         $k = k + 1$
 8:     until a stopping rule is satisfied
 9:     $\eta = \eta\, c$
10: until a stopping rule is satisfied
11: output: $\boldsymbol{B}^k = \boldsymbol{D}^k\boldsymbol{C}^k$, where $\boldsymbol{D}^k = \mathrm{Diag}(\boldsymbol{d}_k)$
```

5.2. Coordinate Minimization Over $\boldsymbol{d}$.

Let us first consider the coordinate minimization over $\boldsymbol{d}$ in Step 5 of Algorithm 5.1. The problem amounts to solving

$$\min_{\boldsymbol{d}\in\mathbb{R}^N} f(\boldsymbol{d}) := \underbrace{\eta\|\boldsymbol{C}^\top\boldsymbol{d} - \boldsymbol{p}\|^2}_{:=f_0(\boldsymbol{d})} \underbrace{- \textstyle\sum_{i=1}^N \log(d_i)}_{:=f_1(\boldsymbol{d})}. \tag{5.3}$$

The above problem is convex. It also falls into the scope of proximal gradient methods (cf. Section 4.2), with Lipschitz continuous ∇f_0. We employ the (standard) extrapolated proximal gradient method to compute the solution to problem (5.3). The algorithm is shown in Algorithm 5.2. Note that

$$\mathrm{prox}_{\mu^{-1}f_1}(\boldsymbol{d}) = \left(\tfrac{d_1 + \sqrt{d_1^2 + 4/\mu}}{2}, \cdots, \tfrac{d_N + \sqrt{d_N^2 + 4/\mu}}{2} \right). \tag{5.4}$$

Algorithm 5.2 an extrapolated proximal gradient algorithm for $\min_{\boldsymbol{d}\in\mathbb{R}^N} F_\eta(\boldsymbol{C},\boldsymbol{d})$

```
given: a starting point $\boldsymbol{d}^0$; and an extrapolation sequence $\{\alpha_k\}$, typically the FISTA sequence [2]
$k = 0$, $\boldsymbol{d}^{-1} = \boldsymbol{d}^0$,
$\mu = 2\eta\sigma_{\max}(\boldsymbol{C})^2$, where $\sigma_{\max}(\boldsymbol{C})$ is the largest singular value of $\boldsymbol{C}$
repeat
    $\boldsymbol{d}_{\rm ex}^k = \boldsymbol{d}^k + \alpha_k(\boldsymbol{d}^k - \boldsymbol{d}^{k-1})$
    $\boldsymbol{d}^{k+1} = \mathrm{prox}_{\mu^{-1}f_1}(\boldsymbol{d}_{\rm ex}^k - \mu^{-1}\nabla f_0(\boldsymbol{d}_{\rm ex}^k))$; $f_0$ is given in (5.3); $\mathrm{prox}_{\mu^{-1}f_1}$ is given in (5.4)
    $k = k + 1$
until a stopping rule is satisfied
output: $\boldsymbol{d}^k$
```

5.3. Coordinate Minimization Over $\boldsymbol{C}$. Next, consider the coordinate minimization over $\boldsymbol{C}$. The problem can be presented as

$$\min_{\boldsymbol{C}\in\mathbb{R}^{N\times N}} \; f(\boldsymbol{C}) := \underbrace{-\log|\det(\boldsymbol{C})| - \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\log\Phi(\boldsymbol{c}_i^\top\bar{\boldsymbol{y}}_t) + \eta\|\boldsymbol{C}^\top\boldsymbol{d}-\boldsymbol{p}\|^2}_{:=f_0(\boldsymbol{C})} + \underbrace{\mathbb{I}_{\mathcal{U}^N}(\boldsymbol{C})}_{:=f_1(\boldsymbol{C})} \tag{5.5}$$

We begin by considering the proximal gradient method:

$$\boldsymbol{C}^{k+1} = \mathrm{prox}_{\mu_k^{-1}f_1}(\boldsymbol{C}^k - \mu_k^{-1}\nabla f_0(\boldsymbol{C}^k)) = \Pi_{\mathcal{U}^N}(\boldsymbol{C}^k - \mu_k^{-1}\nabla f_0(\boldsymbol{C}^k)), \tag{5.6}$$

where $\mu_k > 0$ is chosen such that the sufficient decrease condition is satisfied, and it is done by the backtracking line search (cf. (4.9)); we have

$$\Pi_{\mathcal{U}^N}(\boldsymbol{C}) = [\,\Pi_{\mathcal{U}}(\boldsymbol{c}_1),\ldots,\Pi_{\mathcal{U}}(\boldsymbol{c}_N)\,]^\top, \quad \Pi_{\mathcal{U}}(\boldsymbol{c}) = \begin{cases} \boldsymbol{c}/\|\boldsymbol{c}\| & \text{if } \boldsymbol{c}\neq\boldsymbol{0} \\ \text{any } \boldsymbol{u}\in\mathcal{U} & \text{if } \boldsymbol{c}=\boldsymbol{0} \end{cases}$$

The method, by operations, is the same as the standard proximal gradient method. But the problem does not fall within the scope of the stationarity-guaranteed LSB-PG framework, because $\mathcal{U}^N$ is non-convex. We adopt this method mostly based on practicality: It is simple, and the same method or similar methods have been used in practice [7, 26, 31], with reasonable results demonstrated. Moreover, as a supporting argument, the method is shown to be equipped with some stationarity guarantee under the assumption of Lipschitz continuous ∇f_0 [31].

The above method is just a vanilla version of our actual algorithm. There is a practical issue: the computation of ∇f_0 is expensive, and the direct use of the proximal gradient method can be slow in terms of the runtimes. To give an idea, let us show ∇f_0:

$$\nabla f_0(\boldsymbol{C}) = -\boldsymbol{C}^{-\top} - \frac{1}{T}\sum_{t=1}^{T}\begin{bmatrix} \frac{1}{\Phi(\boldsymbol{c}_1^\top\bar{\boldsymbol{y}}_t)}\frac{1}{\sqrt{2\pi}}e^{-(\boldsymbol{c}_1^\top\bar{\boldsymbol{y}}_t)^2/2}\bar{\boldsymbol{y}}_t^\top \\ \vdots \\ \frac{1}{\Phi(\boldsymbol{c}_N^\top\bar{\boldsymbol{y}}_t)}\frac{1}{\sqrt{2\pi}}e^{-(\boldsymbol{c}_N^\top\bar{\boldsymbol{y}}_t)^2/2}\bar{\boldsymbol{y}}_t^\top \end{bmatrix} + 2\eta\,\boldsymbol{d}(\boldsymbol{C}^\top\boldsymbol{d}-\boldsymbol{p})^\top.$$

We see that computing ∇f_0 requires evaluating Φ for a number of NT times (recall that T is large in practice). The function Φ does not have a closed form and is evaluated by a numerical method. While this should not be an issue when we are required to call Φ a few times, the problem here requires us to evaluate Φ numerous times (and at every iteration).

To reduce the number of times Φ is called, and thereby alleviate the computational burden, we consider a combination of the majorization-minimization (MM) and proximal gradient method. Recall the idea of MM: i) build a surrogate of f by finding a majorant $g(\boldsymbol{C}, \tilde{\boldsymbol{C}})$ of f at $\tilde{\boldsymbol{C}}$, i.e., $f(\boldsymbol{C}) \leq g(\boldsymbol{C}, \tilde{\boldsymbol{C}})$ for all $\boldsymbol{C}, \tilde{\boldsymbol{C}}$, and $f(\boldsymbol{C}) = g(\boldsymbol{C}, \boldsymbol{C})$; ii) handle the problem by recursively solving $\boldsymbol{C}^{k+1} = \min_{\boldsymbol{C}} g(\boldsymbol{C}, \boldsymbol{C}^k)$. Consider the following fact.

Fact 5 ([28] and the references therein). *It holds that, for any $\tilde{x} \in \mathbb{R}$,*

$$-\log \Phi(x) \leq g(x, \tilde{x}) := \frac{1}{2}|x + w(\tilde{x})|^2 + r(\tilde{x}),$$

where $r(\tilde{x})$ does not depend on x;

$$w(\tilde{x}) = -\tilde{x} - \frac{1}{\Phi(\tilde{x})} \frac{1}{\sqrt{2\pi}} e^{-\tilde{x}^2/2}.$$

Also, we have $g(x, x) = -\log \Phi(x)$.

Let us apply Fact 5 to build a majorant of f_0:

$$g_0(\boldsymbol{C}, \tilde{\boldsymbol{C}}) = -\log|\det(\boldsymbol{C})| + \frac{1}{2T}\sum_{t=1}^{T}\sum_{i=1}^{N}\left|\boldsymbol{c}_i^\top \bar{\boldsymbol{y}}_t - w(\tilde{\boldsymbol{c}}_i^\top \bar{\boldsymbol{y}}_t)\right|^2 + \eta\|\boldsymbol{C}^\top \boldsymbol{d} - \boldsymbol{p}\|^2 + r(\tilde{\boldsymbol{C}}), \tag{5.7}$$

for some r that does not depend on $\boldsymbol{C}$. Also, let $g(\boldsymbol{C}, \tilde{\boldsymbol{C}}) = g_0(\boldsymbol{C}, \tilde{\boldsymbol{C}}) + f_1(\boldsymbol{C})$, which is a majorant of f. We carry out MM, in an inexact sense, by approximating $\boldsymbol{C}^{k+1} = \arg\min_{\boldsymbol{C}} g(\boldsymbol{C}, \boldsymbol{C}^k)$ via the proximal gradient method. By doing so, we hope that the number of times Φ is called can be reduced: the evaluations of Φ happen in the majorant construction step (5.7), but not in the (more intensively operating) proximal gradient iterations. Our high-level algorithm description is complete, and the algorithm is shown below. Note that the actual proximal gradient method we employ is extrapolated.

6. Numerical Results.

Now we proceed to numerical results. While we focused on giving a novel explanation of SISAL, the study itself showed new possibilities which we would like to examine by numerical experiments. The most interesting one is the approximate ML estimator in Formulation 3, which resembles a SISAL variant that adopts a probabilistic penalty term. This probabilistic SISAL does not have the regularization parameter λ, and we want to see how well it works compared to SISAL (which requires tuning λ). Also we are interested in the hinge-square SISAL variant in Formulation 4, in terms of runtimes.

6.1. Settings of the Algorithms.

The implementations of the hinge-square and probabilistic SISAL formulations in Formulations 4 and 3 are accomplished by Algorithms 4.2 and 5.1, respectively. For convenience, Algorithms 4.2 and 5.1 will be called H^2-SISAL and Pr-SISAL, respectively, in the sequel. We first specify the dimensionality reduction (DR) preprocessing, which is required by the SISAL algorithms. The standard PCA is used to perform

Algorithm 5.3 a combined MM and extrapolated proximal gradient algorithm for $\min_{\boldsymbol{C} \in \mathcal{U}^N} F_\eta(\boldsymbol{C}, \boldsymbol{d})$

```
1:  given: an invertible starting point $\boldsymbol{C}^0$; and an extrapolation sequence $\{\alpha_k\}$, typically the FISTA sequence [2]
2:  $k = 0$,
3:  repeat    % MM iterations
4:      compute $w((\boldsymbol{c}_i^k)^\top \bar{\boldsymbol{y}}_t)$ for all $i, t$
5:      $l = 0$, $\boldsymbol{C}^{k,-1} = \boldsymbol{C}^{k,0} = \boldsymbol{C}^k$
6:      repeat    % extrapolated proximal gradient iterations
7:          $\boldsymbol{C}_{\rm ex}^{k,l} = \boldsymbol{C}^{k,l} + \alpha_l(\boldsymbol{C}^{k,l} - \boldsymbol{C}^{k,l-1})$
8:          $\boldsymbol{C}^{k,l+1} = \Pi_{\mathcal{U}^N}(\boldsymbol{C}_{\rm ex}^{k,l} - \mu_{k,l}^{-1} \nabla g_0(\boldsymbol{C}_{\rm ex}^{k,l}, \boldsymbol{C}^k))$, where $\mu_{k,l}$ is chosen such that

            $$g_0(\boldsymbol{C}^{k,l+1}, \boldsymbol{C}^k) \leq g_0(\boldsymbol{C}_{\rm ex}^{k,l}, \boldsymbol{C}^k) + \langle \nabla g_0(\boldsymbol{C}_{\rm ex}^{k,l}, \boldsymbol{C}^k), \boldsymbol{C}^{k,l+1} - \boldsymbol{C}_{\rm ex}^{k,l} \rangle + \frac{\mu_{k,l}}{2} \|\boldsymbol{C}^{k,l+1} - \boldsymbol{C}_{\rm ex}^{k,l}\|^2$$

            (i.e., sufficient decrease) is satisfied, and it is done by the backtracking line search; $g_0$ is given in (5.7)
9:          $l = l + 1$
10:     until a stopping rule is satisfied
11:     $\boldsymbol{C}^{k+1} = \boldsymbol{C}^{k,l}$
12:     $k = k + 1$
13: until a stopping rule is satisfied
14: output: $\boldsymbol{C}^k$
```

DR. To be specific, let $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T \in \mathbb{R}^M$ be the data points. We compute $\hat{\boldsymbol{R}}_{yy} = \frac{1}{T} \sum_{t=1}^T \boldsymbol{y}_t \boldsymbol{y}_t^\top$, compute the N-principal eigenvector matrix $\boldsymbol{U} \in \mathbb{R}^{M \times N}$ of $\hat{\boldsymbol{R}}_{yy}$, and take $\tilde{\boldsymbol{y}}_t = \boldsymbol{U}^\top \boldsymbol{y}_t \in \mathbb{R}^N$ as the dimension-reduced data points. Pr-SISAL or H^2-SISAL is then applied to $\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_T$ to get an estimate of $\tilde{\boldsymbol{A}}_0 = \boldsymbol{U}^\top \boldsymbol{A}_0$, and we use the relation $\boldsymbol{A}_0 = \boldsymbol{U} \tilde{\boldsymbol{A}}_0$ to form the estimate of $\boldsymbol{A}_0$. In this connection, it is worth noting that, for the case of $M \geq N+1$, we can also estimate the noise power σ^2 from $\hat{\boldsymbol{R}}_{yy}$, specifically, by taking the $(N+1)$th eigenvalue of $\hat{\boldsymbol{R}}_{yy}$ as the estimate of σ^2; this is a commonly-used trick in statistical signal processing [29, Chapter 4.5].

The settings of Pr-SISAL in Algorithm 5.1 are as follows. The vector $\boldsymbol{p}$ is estimated by (3.17). The starting point is generated by expanded vertex component analysis (VCA), a built-in function of SISAL and a slight modification of the output by the VCA algorithm [22]. We set the initial value of η to 1 and set $c = 5$. We stop the inner loop (Steps 4–8) if $\text{rc}(\boldsymbol{B}^{k+1}, \boldsymbol{B}^k) := \|\boldsymbol{B}^{k+1} - \boldsymbol{B}^k\| / \|\boldsymbol{B}^k\| \leq 10^{-7}$ (rc stands for relative change) or if the number of inner loops exceeds 4×10^5. We stop the outer loop if the number of outer loops exceeds 10. For the sub-algorithm Algorithm 5.2, we stop if $\text{rc}(\boldsymbol{d}^{k+1}, \boldsymbol{d}^k) \leq 10^{-5}$. For the sub-algorithm Algorithm 5.3, we stop the MM loop and the proximal gradient loop if $\text{rc}(\boldsymbol{C}^{k+1}, \boldsymbol{C}^k) \leq 10^{-5}$ and $\text{rc}(\boldsymbol{C}^{k,l+1}, \boldsymbol{C}^{k,l}) \leq 10^{-3}$, respectively. The extrapolation sequence $\{\alpha_k\}$ in Algorithms 5.2 and 5.3 is chosen as the (standard) FISTA sequence [2].

The settings of H^2-SISAL in Algorithm 4.2 are as follows. We choose $\boldsymbol{p} = (\boldsymbol{Y}^\top)^\dagger \boldsymbol{1}$. The starting point is generated by expanded VCA. The FISTA extrapolation sequence is used. We stop Algorithm 4.2 if $\text{rc}(\boldsymbol{B}^{k+1}, \boldsymbol{B}^k) \leq 10^{-6}$.

We will benchmark Pr-SISAL and H^2-SISAL against SISAL itself, VCA [22], ISA-PRISM and VIA-PRISM [34]. SISAL and VCA have open source codes, and we use them directly. The stopping rule of SISAL is that the number of iterations exceeds 250. ISA-PRISM is an importance sampling scheme for implementing the ML estimator (3.2), and VIA-PRISM is a variational inference approximation scheme for the ML estimator (3.2). We run ISA-PRISM only for small N, due to its demanding computational cost to achieve reasonable performance for large N. We stop ISA-PRISM when the number of iterations exceeds 100, and we use rejection sampling, with 500 initial samples, to implement ISA-PRISM. We stop VIA-PRISM when the number of iterations exceeds 500. Also, our VIA-PRISM implementation has some differences from that in the original work [34]; we replace the optimization algorithm for the variational variables, Algorithm 1 in [34], with a projected gradient algorithm, which was found to be more efficient.

6.2. Comparisons of SISAL, H^2-SISAL and Pr-SISAL By Simulations.

We conduct our simulations by the following way. We generate the data points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ by the model in (3.1), i.e., $\boldsymbol{y}_t = \boldsymbol{A}_0 \boldsymbol{s}_t + \boldsymbol{v}_t$, where the $\boldsymbol{s}_t$'s are i.i.d. uniform distributed on the unit simplex; the $\boldsymbol{v}_t$'s are i.i.d. Gaussian with mean zero and covariance $\sigma^2 \boldsymbol{I}$. In addition, for each simulation trial, $\boldsymbol{A}_0$ is drawn from an element-wise i.i.d. $[0,1]$ distribution; we also restrict the condition number of the admitted $\boldsymbol{A}_0$ to be no greater than 100. We use a number of 100 simulation trials to evaluate the mean square error (MSE)

$$\mathsf{MSE}(\boldsymbol{A}_0, \hat{\boldsymbol{A}}) = \min_{\boldsymbol{P} \in \mathcal{P}} \frac{1}{MN} \|\boldsymbol{A}_0 - \hat{\boldsymbol{A}} \boldsymbol{P}\|^2,$$

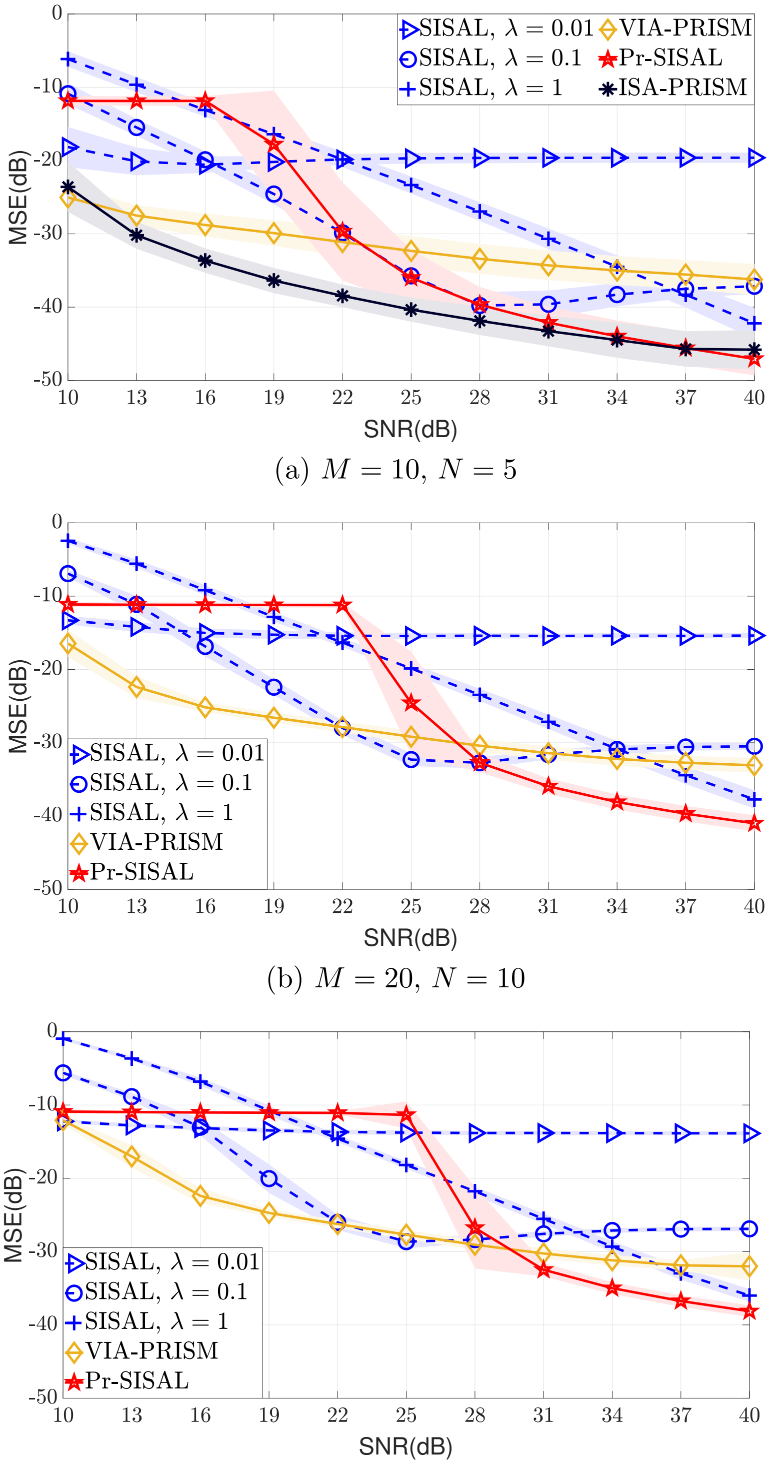
where $\hat{\boldsymbol{A}}$ denotes an estimate of $\boldsymbol{A}_0$ by some algorithm; $\mathcal{P}$ is the set of all permutation matrices on $\mathbb{R}^{N \times N}$. We should also note that the signal-to-noise ratio (SNR) is defined as

$$\mathsf{SNR} = \frac{\frac{1}{T} \sum_{t=1}^{T} \|\boldsymbol{A}_0 \boldsymbol{s}_t\|^2}{M \sigma^2}$$

Fig. 3 compares Pr-SISAL and SISAL for various values of (M, N) and for $T = 1,000$. Our observations are as follows. First, the recovery performance behaviors of SISAL vary from one choice of λ to another. There is no single λ that works best for all SNRs, which suggests the need for parameter tuning in practice. Second, Pr-SISAL performs unsatisfactorily for low SNRs, particularly when compared to VIA-PRISM. But we also see that the performance of Pr-SISAL improves drastically as the SNRs are greater than certain thresholds. Also, for $(M, N) = (10, 5)$, Pr-SISAL achieves performance close to the ML estimator by ISA-PRISM when the SNR is high enough. These results indicate that Pr-SISAL is a good estimator for the high SNR regime.

Fig. 4 compares H^2-SISAL and SISAL under the same settings as above. We see that H^2-SISAL works reasonably and is comparable to SISAL. Also, H^2-SISAL behaves differently for different regularization parameters λ, which suggests that H^2-SISAL requires parameter tuning in practice (just like SISAL).

We move on to the comparison of computational efficiency. Tables 1–2 illustrate some runtime results. The runtimes were measured on a small server with the Intel Core i7-5820K

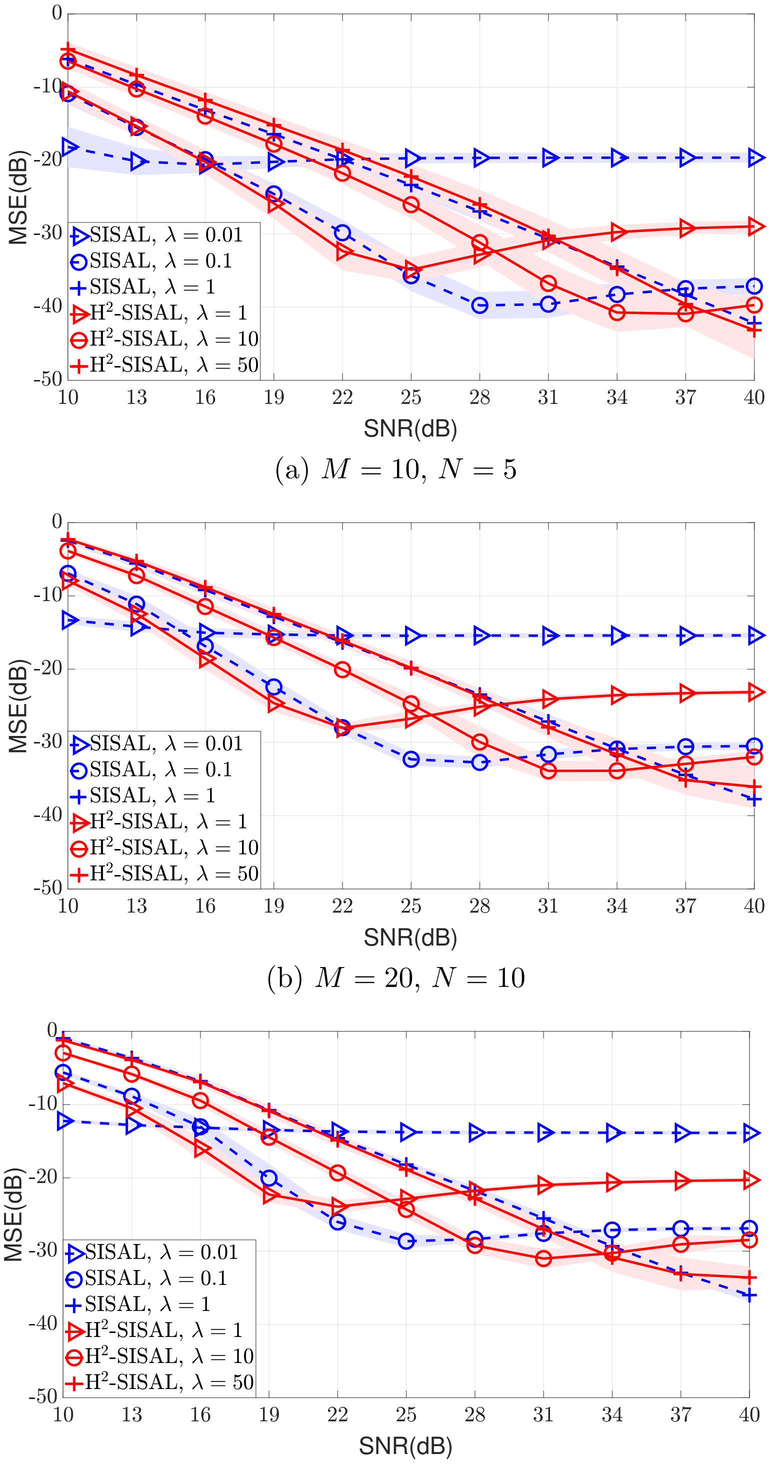


(a) $M = 10,\ N = 5$

(b) $M = 20,\ N = 10$

(c) $M = 30,\ N = 15$

Figure 3. *Comparison of Pr-SISAL, SISAL and VIA-PRISM. The lines are the average MSEs, while the shaded areas show the standard deviations of the MSEs.*



(a) $M = 10$, $N = 5$

(b) $M = 20$, $N = 10$

(c) $M = 30$, $N = 15$

Figure 4. *Comparison of H^2-SISAL and SISAL.*

CPU processor and 64GB memory, and with implementations using MATLAB 2019a. H^2-SISAL is seen to run faster than SISAL. Pr-SISAL, in comparison, is slow, although this is so far the best algorithm we can build for the difficult formulation of probabilistic SISAL. The reader will see in the extra simulation results in Appendix A that the proximal gradient method for tackling SISAL and H^2-SISAL is even slower for probabilistic SISAL.

Table 1

Average runtimes (in sec.) of SISAL, H^2-SISAL, Pr-SISAL and VIA-PRISM. $T = 1,000$, $\mathsf{SNR} = 30dB$.

(M, N)	$(10, 5)$	$(20, 10)$	$(30, 15)$
SISAL, $\lambda = 0.1$	0.078	0.129	0.162
H^2-SISAL, $\lambda = 10.0$	0.033	0.066	0.132
Pr-SISAL	8.336	21.854	42.785
VIA-PRISM	0.632	0.974	1.212

Table 2

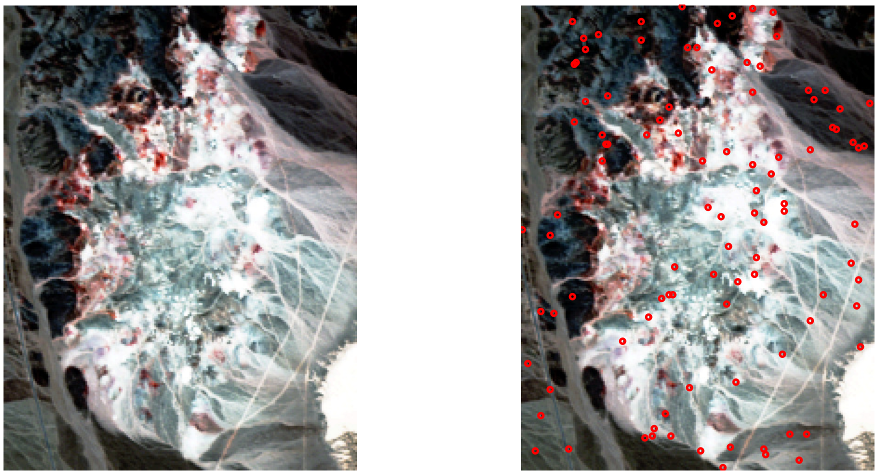
Average runtimes (in sec.) of SISAL, H^2-SISAL, Pr-SISAL and VIA-PRISM. $(M, N) = (20, 10)$, $\mathsf{SNR} = 30dB$.

T	1000	2000	3000	4000	5000	6000	7000	8000
SISAL, $\lambda = 0.1$	0.119	0.201	0.295	0.353	0.401	0.455	0.539	0.587
H^2-SISAL, $\lambda = 10.0$	0.064	0096	0.139	0.192	0.230	0.246	0.281	0.325
Pr-SISAL	23.145	24.656	50.609	56.395	75.753	75.278	100.100	100.917
VIA-PRISM	0.986	1.600	2.276	2.860	3.349	3.928	4.746	4.961

6.3. A Semi-Real Data Experiment.

We further test Pr-SISAL by using real data. The application of interest is hyperspectral unmixing (HU). The real data set used to perform our experiment is the Cuprite hyperspectral image [32]; we will simply call it Cuprite for convenience. Cuprite is interesting in the sense that, among the popular and publicly available data sets in hyperspectral remote sensing, Cuprite is the only one that has more than 10 materials (to our best knowledge). Cuprite has been used to demonstrate many HU algorithms, e.g., [8, 18, 22, 34], and real data experiments by Cuprite have almost become a standard. An illustration of the Cuprite image is shown in Fig. 5(a).

The settings of our experiment are as follows. We largely follow the standard procedure in the literature [8, 18, 22, 34], particularly, the one in [34]. Some additional details are as follows. We adopt the band selection in [18]. It was argued that Cuprite is composed of 12 materials, namely, those shown in Table 3; we refer the reader to [38] and the references therein for details. The ground-truth $\boldsymbol{A}_0$ corresponds to the reference spectral responses of those materials, taken from the USGS library [9]. We test VCA, VIA-PRISM, SISAL, H^2-SISAL and Pr-SISAL. For all the tested algorithms, we additionally do the following: we apply the data normalization preprocessing, described in Section 2.1, to the data points before DR; also, for Pr-SISAL and VIA-PRISM, we estimate the noise variance σ^2 by the eigenvalue method described in Section 6.1. Moreover, some of the stopping rules are modified: We



(a) Cuprite image

(b) Cuprite image with artificially added outliers; red circles represent outlying pixels.

Figure 5. *Cuprite image; constructed by RGB bands.*

stop SISAL if the number of iterations exceeds $1,000$; we stop the inner loop of Pr-SISAL if $\mathrm{rc}(\boldsymbol{B}^{k+1}, \boldsymbol{B}^k) \leq 2 \times 10^{-7}$ or if the number of iterations exceeds 10^7. We evaluate the recovery performance by the spectral angle distance (SAD)

$$\mathsf{SAD}(\boldsymbol{a}_{0,i}, \hat{\boldsymbol{a}}_{\pi_i}) = \cos^{-1}\left(\frac{\boldsymbol{a}_{0,i}^\top \hat{\boldsymbol{a}}_{\pi_i}}{\|\boldsymbol{a}_{0,i}\|\|\hat{\boldsymbol{a}}_{\pi_i}\|}\right),$$

where $\boldsymbol{a}_{0,i}$ and $\hat{\boldsymbol{a}}_i$ denote the ith column of $\boldsymbol{A}_0$ and $\hat{\boldsymbol{A}}$, respectively; $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ is a set of permutation indices for $\{1, \ldots, N\}$ (i.e. $\pi_i \in \{1, \ldots, N\}$ and $\pi_i \neq \pi_j$ for all $i \neq j$), obtained by minimizing $\sum_{i=1}^{N} \mathsf{SAD}(\boldsymbol{a}_{0,i}, \hat{\boldsymbol{a}}_{\pi_i})$ over all possible permutations.

Table 3 shows the SADs of the tested algorithms. We see that all the algorithms give reasonable SAD performance, with VCA achieving the best average SAD. We also see that SISAL and H^2-SISAL, with the regularization parameter tuned to $\lambda = 0.001$ and $\lambda = 0.01$, respectively, provide comparable performance to Pr-SISAL. But note that Pr-SISAL has no parameter to manually tune.

We also consider an experiment that puts some twist on the Cuprite data experiment. Specifically, we randomly pick some pixels and replace them with outliers; see Fig. 5(b) for an illustration. Our aim is to examine how robust the algorithms are. The experimental settings are the same as above, and additionally we randomly select 100 pixels and replace them with randomly selected spectral responses from the USGS library [9].

Table 4 displays the SAD performance of the tested algorithms for 10 trials (The locations and spectral responses of the outliers are changed at each trial). It is seen that VCA gives the worst average SAD, which suggests that VCA is sensitive to outliers. The other algorithms, including the new possibility of H^2-SISAL and Pr-SISAL, are more robust as indicated by their SAD performance. Fig. 6 shows the estimated spectral signatures $\hat{\boldsymbol{a}}_i$ of the various materials

Table 3

SAD performances on the Cuprite dataset. The best SADs among all the tested algorithms are marked in **bold***.*

Alg. / Endmember	VCA	SISAL		H^2-SISAL		Pr-SISAL	VIA-PRISM
		$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.01$	$\lambda = 0.1$		
Alunite	2.07	4.55	6.82	**1.65**	3.83	3.27	4.54
Andradite	2.07	2.35	5.66	2.37	3.69	**1.89**	3.10
Buddingtonite	**2.11**	5.20	3.68	2.92	3.19	3.43	3.88
Dumortierite	**2.66**	3.25	8.07	3.32	6.49	3.51	3.39
Kaolinite$_1$	2.51	2.22	2.78	**2.16**	3.06	2.67	3.90
Kaolinite$_2$	**1.99**	2.48	7.77	2.29	6.20	**1.99**	2.79
Muscovite	**2.12**	2.80	3.15	6.07	4.30	3.64	2.67
Montmorillonite	1.74	2.53	3.88	1.99	2.77	**1.27**	3.22
Nontronite	**1.97**	3.81	2.84	3.03	3.72	2.75	3.14
Pyrope	2.10	1.45	3.93	1.94	2.76	1.70	**1.32**
Sphene	**1.49**	3.19	7.85	3.47	6.95	4.49	1.83
Chalcedony	2.86	3.82	3.85	3.09	3.38	**1.59**	4.35
Average SAD	**2.14**	3.14	5.02	2.86	4.19	3.13	3.07

from one random trial. We observe that SISAL, H^2-SISAL and Pr-SISAL yield good recovery; VCA and VIA-PRISM are not as promising in comparison.

Table 4

SAD performances on the Cuprite dataset with outliers. The best SADs averaged over 10 trials among all the tested algorithms are marked in **bold***.*

Alg. / Endmember	VCA	SISAL		H^2-SISAL		Pr-SISAL	VIA-PRISM
		$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.01$	$\lambda = 0.1$		
Alunite	9.64±4.59	4.74±0.26	6.72±1.21	**2.82**±1.30	5.84±1.50	3.91±0.77	11.65±2.72
Andradite	8.38±5.21	3.45±0.48	7.50±1.97	2.95±0.61	6.16±0.96	**2.27** ±0.31	3.31±0.41
Buddingtonite	13.42±4.14	4.07±1.12	3.93±0.56	**3.23**±0.69	5.49±0.90	3.47±0.31	3.85±1.02
Dumortierite	12.43±3.74	**2.93**±0.83	6.51±1.31	3.17±0.52	5.38±0.79	3.17±0.54	6.85±2.87
Kaolinite$_1$	9.00±4.05	**2.33**±0.43	4.42±1.48	3.18±0.72	5.41±1.01	2.39±0.28	4.38±1.47
Kaolinite$_2$	7.33±4.86	2.53±0.75	5.39±2.09	2.59±0.56	5.52±1.57	**2.34** ±0.59	3.36±1.06
Muscovite	15.40±5.50	**3.11** ±0.59	5.14±2.25	3.66±1.24	5.30±1.32	3.25±0.58	4.57±0.64
Montmorillonite	10.31±3.65	3.47±0.57	3.29±0.24	2.31±0.91	3.42±0.53	**2.11** ±0.48	2.79±0.28
Nontronite	5.92±2.96	3.66±0.57	3.75±0.64	3.33±0.98	4.46±1.03	**2.58** ±0.42	3.36±0.73
Pyrope	12.59±3.87	2.72±1.00	5.79±2.17	3.44±0.89	5.15±1.45	**2.62** ±0.53	3.11±0.65
Sphene	11.96±1.34	**2.35**±0.91	5.91±1.37	2.99±0.62	6.30±2.09	3.69±0.66	9.85±1.39
Chalcedony	14.61±4.89	2.68±0.40	4.96±2.06	2.98±0.86	5.78±1.11	**2.58** ±0.78	6.27±4.76
Average SAD	10.91	3.17	5.28	3.05	5.35	**2.86**	5.28

7. Conclusions. In this article we showed that the famous SISAL algorithm, developed by Bioucas-Dias in hyperspectral unmixing in 2009, can be explained as a probabilistic method for SCA. In particular, SISAL was derived from the noiseless case, and our study provides an explanation of why SISAL can be robust to noise. Moreover, we gave a positive answer to the question of whether the SISAL algorithm can lead to provable convergence to a stationary point. This was done by casting SISAL as an instance of a proximal gradient framework in non-convex first-order optimization. Furthermore, through connecting SISAL and probabilistic

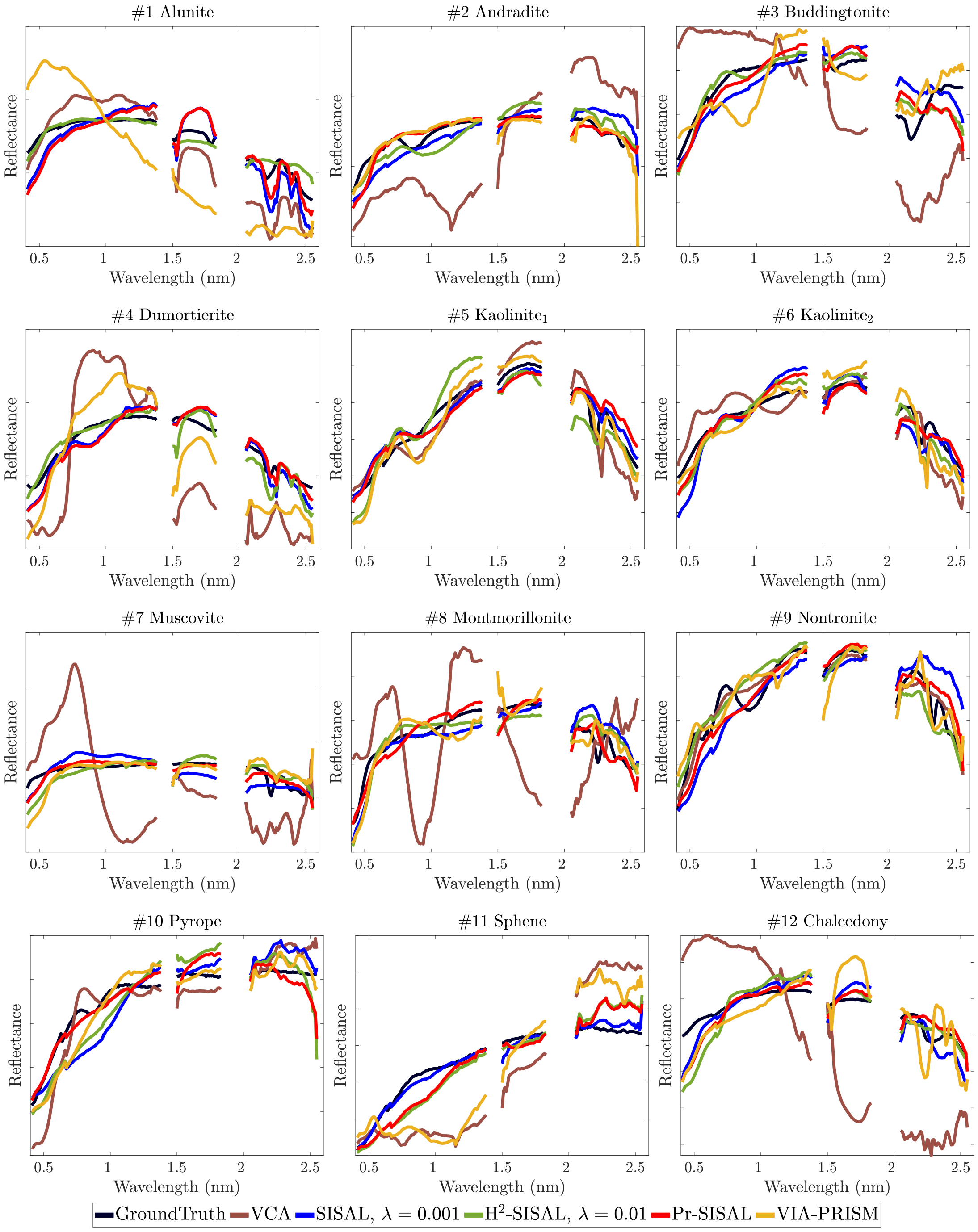


Figure 6. *Estimated spectrums of Cuprite. Algorithms: VCA, SISAL with* $\lambda = 0.001$*,* H^2*-SISAL with* $\lambda = 0.01$*, Pr-SISAL, and VIA-PRISM.*

SCA, we also found new SCA formulations that resemble SISAL. To allow us to numerically study the new SCA formulations, we built customized algorithms for them. The potential of the new algorithms was demonstrated by numerical experiments.

Appendix A. Additional Simulation Results.

We display two more numerical results for Pr-SISAL. The first is with Heuristic 1, which is used to build the approximate ML formulation in Formulation 3. To put into context, let us write down a slightly more general form of Formulation 3:

$$\min_{\boldsymbol{B}^\top \boldsymbol{1}=\boldsymbol{p}} \; -\log(|\det(\boldsymbol{B})|) - \frac{\tau}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\log\Phi\left(\frac{\boldsymbol{b}_i^\top \boldsymbol{y}_t}{\sigma\|\boldsymbol{b}_i\|}\right), \tag{A.1}$$

where $\tau > 0$, and Formulation 3 is the special case of $\tau = 1$. In Remark 2, we argue that $\tau = 1/(N+1)$ is arguably equipped with a better rationale (lower-bound approximation of the ML objective), but eventually the heuristic (and, intuitively, more progressive) choice of $\tau = 1$ prevails in terms of approximating the ML problem better in practice. We want to illustrate that. Fig. 7 shows the performance of formulation in (A.1) for different values of τ and for $(M,N) = (10,5)$, $T = 1,000$; the simulation is done by exactly the same way as in Section 6.2. We see that $\tau = 1/(N+1)$ does not work well, except for very high SNRs. We also try $\tau = N+1$ (more progressive than $\tau = 1$), and the result is not as good as $\tau = 1$.

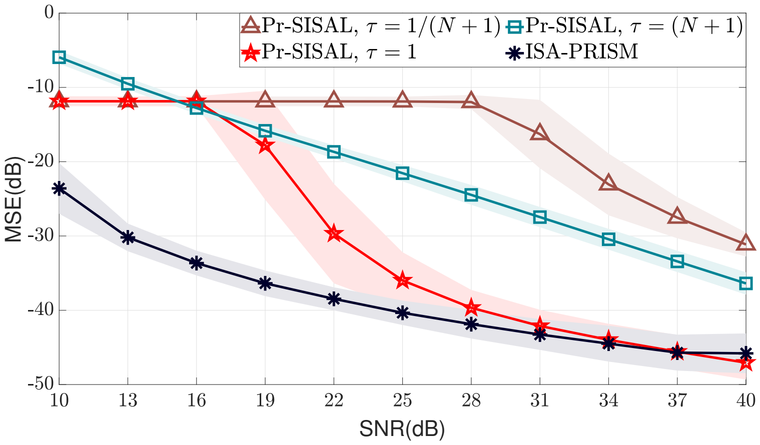


Figure 7. *Performance of the formulation in* (A.1) *for different values of* τ.

The second result is about the implementations of Formulation 3. It was mentioned that the proximal gradient method can be used to handle Formulation 3, but the results are not promising. Here we show the results. We implement Formulation 3 using the same proximal gradient algorithm in Algorithm 4.2, with or without extrapolation. We stop the algorithm if $\mathrm{rc}(\boldsymbol{B}^{k+1}, \boldsymbol{B}^k) \leq 10^{-8}$ or if the number of iterations exceeds 4×10^5. Fig. 8 and Table 5 show the MSE and runtime performance, respectively, for $(M,N,T) = (20,10)$, $T = 1,000$; the simulation settings are the same as the previous. There, “Pr-SISAL”, “Pr-SISAL, PG” and “Pr-SISAL, EPG” refer to the inexact BCD algorithm in Algorithm 5.1, the proximal gradient algorithm and the extrapolated proximal gradient algorithm, all for Formulation 3. We see that all the implementations yield similar MSE performance, but the proximal gradient implementations are very slow.

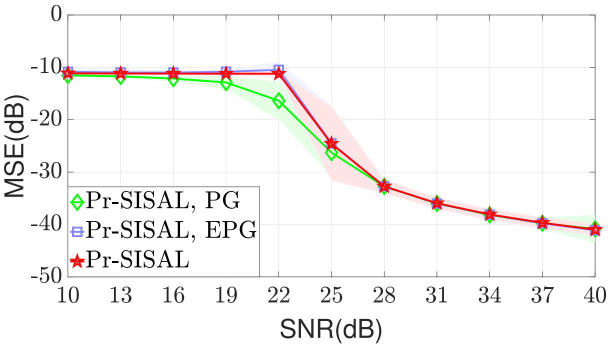


Figure 8. *Performance comparison of several Pr-SISAL implementations.*

Table 5
Average runtime (in sec.) for several Pr-SISAL implementations.

Algorithms	Runtimes
Pr-SISAL, PG	198.814
Pr-SISAL, EPG	243.307
Pr-SISAL	21.542

REFERENCES

[1] H. Attias, *Independent factor analysis*, Neural Comput., 11 (1999), pp. 803–851.

[2] A. Beck, *First-Order Methods in Optimization*, vol. 25, SIAM, Philadelphia, PA, USA, 2017.

[3] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[4] J. Bioucas-Dias, *A variable splitting augmented Lagrangian approach to linear spectral unmixing*, in Procedings of the First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, IEEE, 2009.

[5] J. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, *Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches*, IEEE J. Sel. Topics Appl. Earth Observ., 5 (2012), pp. 354–379.

[6] S. Bonettini, I. Loris, F. Porta, and M. Prato, *Variable metric inexact line-search-based methods for nonsmooth optimization*, SIAM J. Optim., 26 (2016), pp. 891–921.

[7] N. Boumal, *Nonconvex phase synchronization*, SIAM J. Optim., 26 (2016), pp. 2355–2377.

[8] 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 Trans. Signal Process., 57 (2009), pp. 4418–4432.

[9] R. N. Clark, G. A. Swayze, R. Wise, K. E. Livo, T. Hoefen, R. F. Kokaly, and S. J. Sutley, *USGS digital spectral library splib06a*, U.S. Geological Survey, Digital Data Series 231, (2007).

[10] N. Dobigeon, S. Moussaoui, M. Coulon, J.-Y. Tourneret, and A. O. Hero, *Joint Bayesian endmember extraction and linear unmixing for hyperspectral imagery*, IEEE Trans. Signal Process., 57 (2009), pp. 4355–4368.

[11] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma, *Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications*, IEEE Signal Process. Mag., 36 (2019), pp. 59–80.

[12] 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 Trans. Signal Process., 64 (2016), pp. 6254–6268.

[13] X. Fu, W.-K. Ma, K. Huang, and N. D. Sidiropoulos, *Blind separation of quasi-stationary sources: Exploiting convex geometry in covariance domain*, IEEE Trans. Signal Process., 63 (2015), pp. 2306–2320.

[14] S. Ghadimi and G. Lan, *Accelerated gradient methods for nonconvex nonlinear and stochastic programming*, Math. Program., 156 (2016), pp. 59–99.

[15] N. Gillis, *Nonnegative Matrix Factorization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2021.

[16] P. Gritzmann, V. Klee, and D. Larman, *Largest j-simplices in n-polytopes*, Discrete Comput. Geom.,

13 (1995), pp. 477–515.

[17] I. Khemakhem, D. Kingma, R. Monti, and A. Hyvarinen, *Variational autoencoders and nonlinear ICA: A unifying framework*, in Proceedings of the 23th International Conference on Artificial Intelligence and Statistics, vol. 108, PMLR, 2020, pp. 2207–2217.

[18] J. Li, A. Agathos, D. Zaharie, J. M. Bioucas-Dias, A. Plaza, and X. Li, *Minimum volume simplex analysis: A fast algorithm for linear hyperspectral unmixing*, IEEE Trans. Geosci. Remote Sens., 53 (2015), pp. 5067–5082.

[19] C.-H. Lin, W.-K. Ma, W.-C. Li, C.-Y. Chi, and A. Ambikapathi, *Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case*, IEEE Trans. Geosci. Remote Sens., 53 (2015), pp. 5530–5546.

[20] W.-K. Ma, *On hyperspectral unmixing*, in Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, 2021. online available: https://arxiv.org/pdf/2106.14177.pdf.

[21] 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*, IEEE Signal Process. Mag., 31 (2014), pp. 67–81.

[22] J. Nascimento and J. Bioucas-Dias, *Vertex component analysis: A fast algorithm to unmix hyperspectral data*, IEEE Trans. Geosci. Remote Sens., 43 (2005), pp. 898–910.

[23] J. Nascimento and J. Bioucas-Dias, *Hyperspectral unmixing based on mixtures of Dirichlet components*, IEEE Trans. Geosci. Remote Sens., 50 (2012), pp. 863–878.

[24] J. M. Nascimento and J. M. Bioucas-Dias, *Learning dependent sources using mixtures of Dirichlet: Applications on hyperspectral unmixing*, in Procedings of the First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, IEEE, 2009.

[25] D. T. Pham and P. Garat, *Blind separation of mixture of independent sources through a quasi-maximum likelihood approach*, IEEE Trans. Signal Process., 45 (1997), pp. 1712–1725.

[26] M. Shao, Q. Li, W.-K. Ma, and A. M.-C. So, *Minimum symbol error rate-based constant envelope precoding for multiuser massive MISO downlink*, in Procedings of Statistical Signal Processing Workshop (SSP), IEEE, 2018.

[27] M. Shao, Q. Li, W.-K. Ma, and A. M.-C. So, *A framework for one-bit and constant-envelope precoding over multiuser massive MISO channels*, IEEE Trans. Signal Process., 67 (2019), pp. 5309–5324.

[28] M. Shao and W.-K. Ma, *Divide and conquer: One-bit MIMO-OFDM detection by inexact expectation maximization*, in Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 4890–4894.

[29] P. Stoica and R. L. Moses, *Spectral Analysis of Signals*, Prentice Hall, Inc., New Jersey, US, 2005.

[30] M. E. Tipping and C. M. Bishop, *Probabilistic principal component analysis*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 61 (1999), pp. 611–622.

[31] J. Tranter, N. D. Sidiropoulos, X. Fu, and A. Swami, *Fast unit-modulus least squares with applications in beamforming*, IEEE Trans. Signal Process., 65 (2017), pp. 2875–2887.

[32] G. Vane, R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter, *The airborne visible/infrared imaging spectrometer (AVIRIS)*, Remote Sensing of Environment, 44 (1993), pp. 127–143.

[33] S. Verdu, *Multiuser Detection*, Cambridge University Press, 1998.

[34] R. Wu, W.-K. Ma, Y. Li, A. M.-C. So, and N. D. Sidiropoulos, *Probabilistic simplex component analysis*, IEEE Trans. Signal Process., 70 (2022), pp. 582–599.

[35] R. Wu, H.-T. Wai, and W.-K. Ma, *Hybrid inexact BCD for coupled structured matrix factorization in hyperspectral super-resolution*, IEEE Trans. Signal Process., 68 (2020), pp. 1728–1743.

[36] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion*, SIAM J. Imaging Sci., 6 (2013), pp. 1758–1789.

[37] Y. Xu and W. Yin, *A globally convergent algorithm for nonconvex optimization based on block coordinate update*, J. Sci. Comput., 72 (2017), pp. 700–734.

[38] F. Zhu, *Hyperspectral unmixing: ground truth labeling, datasets, benchmark performances and survey*, arXiv preprint arXiv:1708.05125, (2017).