



**HAL**  
open science

# Positive Semidefinite Matrix Factorization: A Connection with Phase Retrieval and Affine Rank Minimization

Dana Lahat, Yanbin Lang, Vincent Tan, Cédric Févotte

► **To cite this version:**

Dana Lahat, Yanbin Lang, Vincent Tan, Cédric Févotte. Positive Semidefinite Matrix Factorization: A Connection with Phase Retrieval and Affine Rank Minimization. *IEEE Transactions on Signal Processing*, 2021, 69, pp.3059-3074. 10.1109/TSP.2021.3071293 . hal-03431503

**HAL Id: hal-03431503**

**<https://hal.science/hal-03431503>**

Submitted on 16 Nov 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Positive Semidefinite Matrix Factorization: A Connection with Phase Retrieval and Affine Rank Minimization

Dana Lahat, *Member, IEEE*, Yanbin Lang, *Student Member, IEEE*, Vincent Y. F. Tan, *Senior Member, IEEE*,  
Cédric Févotte, *Senior Member, IEEE*

**Abstract**—Positive semidefinite matrix factorization (PSDMF) expresses each entry of a nonnegative matrix as the inner product of two positive semidefinite (psd) matrices. When all these psd matrices are constrained to be diagonal, this model is equivalent to nonnegative matrix factorization. Applications include combinatorial optimization, quantum-based statistical models, and recommender systems, among others. However, despite the increasing interest in PSDMF, only a few PSDMF algorithms were proposed in the literature. In this work, we provide a collection of tools for PSDMF, by showing that PSDMF algorithms can be designed based on phase retrieval (PR) and affine rank minimization (ARM) algorithms. This procedure allows a shortcut in designing new PSDMF algorithms, as it allows to leverage some of the useful numerical properties of existing PR and ARM methods to the PSDMF framework. Motivated by this idea, we introduce a new family of PSDMF algorithms based on iterative hard thresholding (IHT). This family subsumes previously-proposed projected gradient PSDMF methods. We show that there is high variability among PSDMF optimization problems that makes it beneficial to try a number of methods based on different principles to tackle difficult problems. In certain cases, our proposed methods are the only algorithms able to find a solution. In certain other cases, they converge faster. Our results support our claim that the PSDMF framework can inherit desired numerical properties from PR and ARM algorithms, leading to more efficient PSDMF algorithms, and motivate further study of the links between these models.

**Index Terms**—Positive semidefinite matrix factorization, phase retrieval, affine rank minimization, nonnegative matrix factorizations, iterative hard thresholding, singular value projection, low-rank approximations, low-rank matrix recovery.

## I. INTRODUCTION

**M**ATRIX factorization is a basic tool in numerous fields such as machine learning, engineering, and optimization. In this paper, we address positive semidefinite matrix factorization (PSDMF) [1], [2], a recently-proposed type of

factorization of nonnegative matrices. PSDMF expresses the  $(i, j)$ th entry  $x_{ij}$  of a nonnegative matrix  $\mathbf{X} \in \mathbb{R}^{I \times J}$  as an inner product of two  $K \times K$  symmetric positive semidefinite (psd) matrices  $\mathbf{A}_i$  and  $\mathbf{B}_j$ , indexed by  $i = 1, \dots, I$ ,  $j = 1, \dots, J$ :

$$x_{ij} \cong \langle \mathbf{A}_i, \mathbf{B}_j \rangle = \text{tr}\{\mathbf{A}_i \mathbf{B}_j\} \quad (1)$$

where  $\text{tr}\{\cdot\}$  denotes the trace of a matrix,  $\langle \mathbf{M}, \mathbf{N} \rangle = \text{tr}\{\mathbf{M}^T \mathbf{N}\}$  is the inner product between any two real-valued matrices  $\mathbf{M}$  and  $\mathbf{N}$  with compatible dimensions, and  $\cong$  stands for equality or approximation, depending on the context. In PSDMF literature, the minimal number  $K$  such that a nonnegative matrix  $\mathbf{X}$  admits an exact PSDMF is called the *psd rank* of  $\mathbf{X}$  [1]. Each psd matrix  $\mathbf{A}_i$  and  $\mathbf{B}_j$  may have a different rank, denoted as  $R_{A_i}$  and  $R_{B_j}$ , respectively. We shall sometimes refer to  $R_{A_i}$  and  $R_{B_j}$  as *inner ranks* [3]. Unlike the psd rank, the values of the inner ranks are not guaranteed to be unique, in general; see, e.g., [3], [4].

When  $\mathbf{A}_i$  and  $\mathbf{B}_j$  are constrained to be diagonal matrices for all  $i = 1, \dots, I$  and  $j = 1, \dots, J$ , the resulting model is equivalent to nonnegative matrix factorization (NMF) (e.g., [5]–[8]). In NMF, a nonnegative matrix  $\mathbf{X} \in \mathbb{R}^{I \times J}$  is modeled as a product of two nonnegative matrices:  $\mathbf{X} \cong \mathbf{W} \mathbf{H}^T$ , where  $\mathbf{W} \in \mathbb{R}^{I \times K_{\text{NMF}}}$ ,  $\mathbf{H} \in \mathbb{R}^{J \times K_{\text{NMF}}}$ , and  $K_{\text{NMF}} \leq \min(I, J)$ . In this context, the matrices  $\mathbf{W}$  and  $\mathbf{H}$  are sometimes referred to as *factors*. In NMF, the rows of each factor matrix  $\mathbf{W}$  and  $\mathbf{H}$  are in the nonnegative orthant  $\mathbb{R}_+^{K_{\text{NMF}}}$ , which is a closed convex cone. It is thus possible to express NMF using a PSDMF model by putting the  $i$ th row of  $\mathbf{W}$  as the diagonal of  $\mathbf{A}_i$ , the  $j$ th row of  $\mathbf{H}$  as the diagonal of  $\mathbf{B}_j$ , and setting all other entries of  $\mathbf{A}_i, \mathbf{B}_j$  to zero; in this case,  $K = K_{\text{NMF}}$ . The converse, however, does not hold in general, because the off-diagonal entries of  $\mathbf{A}_i$  and  $\mathbf{B}_j$  may take negative values. The relationship between PSDMF, NMF, and other matrix decompositions is further discussed in [Section II](#).

### A. Motivation

PSDMF was proposed as an extension of a well-known result [9] that links NMF with geometry and with linear constraints in linear programming. Yannakakis' [9] result is fundamental in combinatorial optimization (e.g., [10]), where problems can often be written as linear programs with constraints associated with the facets of a polytope. Yannakakis' result implies that if the nonnegative rank of a slack matrix

D. Lahat is with the School of Electrical Engineering, Tel Aviv University, 69978 Tel Aviv, Israel, and with the Department of CSEE, University of Maryland, Baltimore County, Baltimore, MD 21250, USA. Most of D. Lahat's work was carried out when she was with IRIT, Université de Toulouse, CNRS, Toulouse, France (email: Dana@Lahat.org.il). C. Févotte is with IRIT, Université de Toulouse, CNRS, Toulouse, France (email: Cedric.Fevotte@irit.fr). Y. Lang and V. Y. F. Tan are with Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (emails: e0004795@u.nus.edu, vtan@nus.edu.sg).

The work of D. Lahat and C. Févotte has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 681839 (project FACTORY). V. Y. F. Tan is supported by a Singapore National Research Foundation (NRF) Fellowship (R-263-000-D02-281) and a Singapore Ministry of Education Tier 2 grant (R-263-000-C83-112).

of the polytope associated with the linear constraints of the optimization problem is sufficiently small, one may find a simpler representation of the problem, with fewer constraints, in a higher dimension, and thus reduce the overall complexity of the problem. With PSDMF, this result extends to semidefinite programming [1], where now the psd rank is associated with the number of constraints in the optimization. Applications involving PSDMF include combinatorial optimization [1], [2], [4], quantum computing (e.g., [11]), quantum information theory and quantum communications [2], [4], [12], probabilistic modeling [13], and quantum-based models for recommender systems [14]. The relation to the quantum framework is due to the fact that quantum measurements, known as positive operator valued measure (POVM)s, are represented by a set of psd matrices whose sum is the identity matrix. Recently, it has been shown that PSDMF is a special case of a more general framework of tensor networks [13]. However, despite this broad range of timely applications, a surprisingly small number of PSDMF algorithms has been proposed in the literature, namely those in [3], [13], [14].

### B. Main Contributions

The three main contributions of this paper are as follows.

- We develop a large class of algorithms for PSDMF by relating the problem of PSDMF optimization to two important problems in the recent signal processing literature—affine rank minimization (ARM) (e.g., [15]–[22]) and phase retrieval (e.g., [23]–[28]). In particular, we show that in alternating algorithms, which are the most common framework used to address PSDMF optimization (e.g., in [3], [13], [14], [29]), each subproblem therein consists in approximately minimizing an objective function that is also used in ARM or phase retrieval.

- Based on this observation, we introduce a new family of PSDMF algorithms. These algorithms are based on *singular value projection (SVP)* [20], sometimes referred to as *iterative hard thresholding (IHT)* [30]. Our proposed SVP-based PSDMF algorithms subsume the projected gradient method (PGM) [3] by allowing the use of inner ranks smaller than  $K$ . We also show that SVP subsumes PRIME-Power [26], which is a majorization-minimization-based phase retrieval method. We further propose three variants to our basic SVP-based PSDMF—the first is fast singular value projection (FSVP), which is based on Nesterov’s accelerated gradient descent [31] and subsumes the fast projected gradient method (FPGM) [3]. Our second variant is based on normalized iterative hard thresholding (NIHT) [22], [32]. NIHT was proposed as a computationally efficient version of SVP with a specially-designed step size. The third variant is based on conjugate gradient iterative hard thresholding (CGIHT) [33], a variant of NIHT designed to have fast asymptotic convergence rate. FPGM, FSVP, and CGIHT-based PSDMF require an additional parameter that determines the number of inner iterations within each subproblem; the user has to fine-tune this parameter to achieve sufficient acceleration. Together with the two alternating block gradient descent (ABG) [34], [35] algorithms, recently proposed by two of the authors of this

paper, we provide a collection of tools for PSDMF optimization. These methods are based on different principles and thus can assist in addressing a variety of PSDMF problems.

- Finally, we carry out an extensive set of numerical experiments on random and geometric datasets to compare and contrast the proposed methods with the state-of-the-art coordinate descent (CD) [3]. We show that there exist cases in which NIHT and CGIHT dramatically outperform all other methods, being the *only* algorithms able to converge towards a solution with random initialization. We also show that our proposed projection-based methods generally have a smaller per-iteration computational complexity than CD; this trend is in agreement with our observation that our proposed projection-based methods generally need a smaller CPU time to reach a target model fit error than CD. We show that FSVP and NIHT generally achieve a desired model fit error with fewer iterations and faster than SVP, as predicted by theory. An advantage of NIHT over FSVP and CGIHT is that it does not require an extra parameter to control the number of acceleration steps, as is the case with FSVP and CGIHT. We exhibit some other test cases in which our ABG algorithms succeed in decomposing the matrix faster than the competing algorithms. Our results can serve as guidelines as to which methods might be preferred in different scenarios.

The main message of this paper is that with careful implementation, the PSDMF framework can inherit desirable numerical properties from the multitude of phase retrieval and ARM methods. This then allows for the design and analysis of a host of efficient algorithms for PSDMF starting from more basic signal processing primitives.

### C. Related Work

The first algorithms for PSDMF were developed independently in [3], [13], [14]. Stark’s [14] work is motivated by the predictive power of quantum-inspired recommender systems. Thus, his algorithm uses psd matrices normalized similarly to POVMs. Stark [14] uses a standard semidefinite programming solver to enforce the psd constraint and to minimize a quadratic objective function, in an alternating optimization approach. The framework in [14] does not take into account the values of the inner ranks.

Glasser et al. [13] show that PSDMF is a special case of a more general framework of tensor networks, in which the nonnegative matrix (or tensor) has a probabilistic interpretation. Their algorithm is based on maximum likelihood estimation of the tensor network parameters, and is implemented using a non-linear limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm. Due to their tensor network framework, the value of the inner ranks in their algorithm can be smaller than  $K$  but must be the same for all psd matrices. The focus in [13], [14] is on demonstrating the applicability of the algorithms to the tasks of recommendation and expressivity of certain probabilistic models, respectively, and not on numerical properties of the algorithms.

Vandaele et al.’s [3] work is the closest to ours in the sense that they design general-purpose PSDMF algorithms

that are not tailored to a specific application, and study their numerical properties on exact PSDMF tasks of matrices with geometric interpretation. Vandaele et al. [3] propose two families of alternating PSDMF algorithms that minimize a quadratic objective function: CD and PGM. PGM is limited to the case where all inner ranks are equal to  $K$ , whereas CD can handle any values of inner ranks—as is the case with the algorithms developed in this paper. Our SVP- (resp., FSVP-) based algorithm is a direct generalization of PGM (resp., FPGM) by allowing the inner ranks to take any value.

In these works [3], [13], [14], PSDMF has not yet been connected with phase retrieval or ARM; this idea was first described by two of the authors of this paper in [34]. Based on the connection with phase retrieval and ARM, two of the authors of this paper have recently proposed a new family of efficient PSDMF algorithms, based on ABG [34], [35]. The ABG algorithm in [34] is based on Wirtinger flow [25], which is a gradient descent approach for phase retrieval. The ABG variant in [35] differs from CD [3], from ABG [34], and from the other methods developed in this paper, in that it minimizes an objective function based on the generalized Kullback-Leibler divergence. The generalized Kullback-Leibler divergence is associated with the Poisson log-likelihood. However, as pointed out in [36], replacing a quadratic objective function with the generalized Kullback-Leibler divergence can significantly accelerate the convergence of Wirtinger flow even in the absence of noise. Our preliminary results in [35] indicate that this property can be inherited by PSDMF, in certain cases. Compared with the conference papers [34], [35], this paper provides a more detailed discussion of the connection of PSDMF with phase retrieval and ARM, as well as more extensive comparisons of the two recently-proposed variants of ABG, not only with CD [3], but also, for the first time, with the IHT-based methods that we introduce in this paper. Further details about PSDMF optimization, in the context of the methods developed in this paper, can be found in Section III.

#### D. Notations

We use font types  $a$  and  $A$  to denote scalars. Column vectors and matrices are denoted with  $\mathbf{a}$  and  $\mathbf{A}$ , respectively. Unless otherwise specified,  $\mathbf{a}_i$  is the  $i$ th column of  $\mathbf{A}$ ,  $a_{ij}$  is the  $(i, j)$ th entry of  $\mathbf{A}$ , and  $a_i$  is the  $i$ th entry of  $\mathbf{a}$ .  $\mathbf{I}$  denotes the identity matrix. The operator  $\text{vec}\{\cdot\}$  reshapes a matrix into a column vector, and  $\text{vec}^\top\{\cdot\}$  denotes the transpose of  $\text{vec}\{\cdot\}$ . The linear map  $\mathcal{A} : \mathbb{R}^{K_1 \times K_2} \mapsto \mathbb{R}^I$  is determined by  $I$  matrices  $\mathbf{A}_1, \dots, \mathbf{A}_I \in \mathbb{R}^{K_1 \times K_2}$  and is given by

$$\mathcal{A}(\mathbf{M}) = [\langle \mathbf{A}_1, \mathbf{M} \rangle \quad \dots \quad \langle \mathbf{A}_I, \mathbf{M} \rangle]^\top \in \mathbb{R}^I. \quad (2)$$

Let  $\mathcal{A}^\dagger : \mathbb{R}^I \mapsto \mathbb{R}^{K_1 \times K_2}$  denote the adjoint of  $\mathcal{A}$ ; then, for any  $\mathbf{y} \in \mathbb{R}^I$ , we have (e.g., [4])  $\mathcal{A}^\dagger(\mathbf{y}) = \sum_{i=1}^I y_i \mathbf{A}_i$ . In this paper, we focus on PSDMF over the real numbers  $\mathbb{R}$ . We do so for the sake of simplicity, but also because real-valued PSDMF is used in most applications (e.g., [1], [2]). However, PSDMF over other fields has been considered as well, e.g. [4]. The algorithms proposed in this paper, as well as those in [34], [35] and (F)PGM [3], work equally well over the complex

numbers  $\mathbb{C}$ : one only has to change the transpose operation to Hermitian in the appropriate locations. For the same reasons, and in order to simplify the transitions between the phase retrieval and ARM framework and PSDMF, we shall use real-valued notations (transpose) also for the phase retrieval and ARM equations.

#### E. Outline

In Section II, we provide theoretical background material about PSDMF. In Section III, we explain how PSDMF is related to phase retrieval and ARM, and how this link can be used to design new PSDMF algorithms. Based on this link, we present in Section IV three new projection-based alternating algorithms for PSDMF. Section V is dedicated to numerical issues and comparisons with state of the art. In Section VI, we discuss the impact of our results from a broader perspective.

## II. BACKGROUND

In this section, we provide theoretical preliminaries that are necessary for the exposition of our results in the next sections of this paper. In Section II-A, we explain how PSDMF is related to usual matrix factorization. Section II-B presents a factor-based representation. In Section II-C, we count the number of free variables in real-valued PSDMF. In Section II-D, we discuss implications of the presence of zero values in the input matrix  $\mathbf{X}$  on the factorization.

#### A. PSDMF as a Structured Matrix Factorization

In order to better see the link to usual matrix factorizations, let us vectorize the psd matrices and rearrange them as follows:

$$\mathfrak{A} \triangleq [\text{vec}\{\mathbf{A}_1\} \quad \dots \quad \text{vec}\{\mathbf{A}_I\}] \in \mathbb{R}^{K^2 \times I} \quad (3a)$$

$$\mathfrak{B} \triangleq [\text{vec}\{\mathbf{B}_1\} \quad \dots \quad \text{vec}\{\mathbf{B}_J\}] \in \mathbb{R}^{K^2 \times J}. \quad (3b)$$

Matrices  $\mathfrak{A}$  and  $\mathfrak{B}$  are *structured* because their columns, upon rearrangement as  $K \times K$  matrices, are in  $\mathbb{S}_+^K$ , the closed convex cone of  $K \times K$  psd matrices. With this notation, PSDMF can now be expressed as a structured matrix factorization [3], [4], [29], [37]:

$$\mathbf{X} \cong \mathfrak{A}^\top \mathfrak{B}. \quad (4)$$

The symmetry of the psd matrices implies that  $\text{rank}(\mathfrak{A}) \leq \min(\frac{K(K+1)}{2}, I)$  and  $\text{rank}(\mathfrak{B}) \leq \min(\frac{K(K+1)}{2}, J)$ . Hence,  $\text{rank}(\mathbf{X}) \leq \min(\frac{K(K+1)}{2}, I, J)$  [4]. In Section VII-A of the supplemental material, we demonstrate how the PSDMF model, in terms of psd matrices, can be written as a sum of rank-1 terms.

#### B. A Factor-Based Representation

In PSDMF, the psd matrices can be written as (e.g., [4])

$$\mathbf{A}_i \triangleq \mathbf{U}_i \mathbf{U}_i^\top \quad \text{and} \quad \mathbf{B}_j \triangleq \mathbf{V}_j \mathbf{V}_j^\top, \quad (5)$$

where  $\mathbf{U}_i \in \mathbb{R}^{K \times R_{A_i}}$  and  $\mathbf{V}_j \in \mathbb{R}^{K \times R_{B_j}}$  are referred to as *factor* matrices (factors, for short). This formulation

requires knowing (or guessing) the inner ranks  $R_{A_i}$  and  $R_{B_j}$  in advance. The change of variables in (5) implies that

$$\langle \mathbf{A}_i, \mathbf{B}_j \rangle = \text{tr}\{\mathbf{U}_i \mathbf{U}_i^\top \mathbf{V}_j \mathbf{V}_j^\top\} = \|\mathbf{U}_i^\top \mathbf{V}_j\|_F^2. \quad (6)$$

A change of variables as in (5) is common in semidefinite programming (e.g., [38]), especially when the psd matrices have low rank. Indeed, in PSDMF applications, the psd matrices are often of low rank, i.e.,  $R_{A_i}, R_{B_j} < K$  for some  $i$  and/or  $j$  (e.g., [4]; see also Section II-D). In Section VII-B of the supplemental material, we demonstrate how the PSDMF model, in the factor-based representation, can be written as a sum of rank-1 terms.

### C. Degrees of Freedom

We now use the factor-based formulation to calculate the effective number of degrees of freedom (d.o.f.) in a real-valued PSDMF model. Due to symmetry, a psd matrix  $\mathbf{A}_i$  of rank  $R_{A_i}$  has  $R_{A_i}K - R_{A_i}(R_{A_i} - 1)/2$  free variables. Due to the invariance of the trace operator to rotation of its variables, and given any arbitrary nonsingular  $K \times K$  matrix  $\mathbf{L}$ ,

$$\text{tr}\{\mathbf{A}_i \mathbf{B}_j\} = \text{tr}\{\mathbf{L}^\top \mathbf{A}_i \mathbf{L} \cdot \mathbf{L}^{-1} \mathbf{B}_j \mathbf{L}^{-\top}\} \quad (7)$$

which means that the matrices

$$\mathbf{L}^\top \mathbf{A}_1 \mathbf{L}, \dots, \mathbf{L}^\top \mathbf{A}_J \mathbf{L}, \mathbf{L}^{-1} \mathbf{B}_1 \mathbf{L}^{-\top}, \dots, \mathbf{L}^{-1} \mathbf{B}_J \mathbf{L}^{-\top} \quad (8)$$

also form a PSDMF of  $\mathbf{X}$  [4]. We thus have to subtract  $K^2$  from the number of variables in all factors. Hence, PSDMF with psd rank  $K$  has (at most)

$$N_{\text{model}} = \sum_{i=1}^I \left( R_{A_i} K - \frac{R_{A_i}(R_{A_i} - 1)}{2} \right) + \sum_{j=1}^J \left( R_{B_j} K - \frac{R_{B_j}(R_{B_j} - 1)}{2} \right) - K^2 \quad (9)$$

free variables that we have to learn from the  $N_{\text{data}} = IJ$  observations. In practice, this task is not always achievable, because the psd structure may result in additional constraints, e.g., due to zeros, see Section II-D, and because of the highly non-convex nature of the optimization problem. Besides, if  $\text{rank}(\mathbf{X}) > \frac{K(K+1)}{2}$ , an exact PSDMF does not exist even if  $N_{\text{model}} \gg N_{\text{data}}$ . Thus,  $N_{\text{model}} \geq N_{\text{data}}$  does not guarantee the existence of an exact factorization. If an exact factorization exists, intuitively, the more degrees of freedom we have with respect to (w.r.t.) the number of constraints imposed by the input matrix, the easier it is to satisfy all the constraints. For this reason, finding (bounds on) the psd rank of structured matrices is a major endeavor (e.g., [1], [3], [4], [13], [39], [40]). As demonstrated by [3], PSDMF algorithms can contribute to this effort by validating conjectures and finding new factorizations even in the absence of sufficient theory.

### D. How the Presence of Zeros Affects the Inner Ranks or Why PSDMF Differs from NMF in representing Nonnegative Data

In exact NMF, a zero observation  $x_{ij} = 0$  imposes zero values in the factors because the product of two nonnegative vectors can be zero only if each non-zero entry in one vector

has a zero counterpart in the other vector. Hence, if a given matrix  $\mathbf{X}$  contains zeros (or values relatively close to zero), the factors  $\mathbf{W}$  and  $\mathbf{H}$  will be sparse (or close to sparse). In applications, this turns out to enhance uniqueness and thus yield interpretable factors (e.g., [8], [41], [42]).

Consider now the factor-based formulation of PSDMF, as in (5), with  $\mathbf{u}_r^{(i)}$  and  $\mathbf{v}_s^{(j)}$  the  $r$ th and  $s$ th columns of  $\mathbf{U}_i$  and  $\mathbf{V}_j$ , respectively. A zero value in the observations implies  $x_{ij} = \|\mathbf{U}_i^\top \mathbf{V}_j\|_F^2 = 0$  and imposes  $\mathbf{u}_r^{(i)\top} \mathbf{v}_s^{(j)} = 0$  for all  $1 \leq r \leq R_{A_i}, 1 \leq s \leq R_{B_j}$ . That is, the subspace spanned by the  $R_{A_i}$  columns of  $\mathbf{U}_i$  must be orthogonal to the subspace spanned by the  $R_{B_j}$  columns of  $\mathbf{V}_j$ . Since the dimension of this subspace is  $K$ , this can hold only if  $R_{A_i} + R_{B_j} \leq K$  (e.g., [43], [4, Proposition 1]). We conclude that in PSDMF, the effect of zeros in the input matrix  $\mathbf{X}$  is different than that in NMF because for PSDMF, zeros in the input do not, in general, result in zero values in the factors, but only affect the inner ranks. When there are several zeros in  $\mathbf{X}$ , these orthogonality constraints must be satisfied simultaneously for all pairs of factors indexed by  $(i, j)$  for which  $x_{ij} = 0$ . These additional constraints imply that the balance of free model variables in Section II-C should be used with caution, as it provides only partial and limited information on the expressivity of the model. We shall validate this numerically in Section V-E.

## III. PSDMF OPTIMIZATION BASED ON A LINK WITH PHASE RETRIEVAL AND AFFINE RANK MINIMIZATION

This section presents in detail our concept of designing PSDMF algorithms based on phase retrieval and ARM methods. In Section III-A, we describe the alternating optimization framework for PSDMF, and review the relevant state of the art. In Section III-B, we revisit this alternating optimization framework and explain its relation to phase retrieval and ARM. In Section III-C, we discuss the caveats of this approach.

### A. Background: Alternating Optimization for PSDMF

The psd matrices can be estimated by minimizing the following quadratic objective function [3], [14], [29]:

$$f = f(\{\mathbf{A}_i\}_{i=1}^I, \{\mathbf{B}_j\}_{j=1}^J) = \frac{1}{2} \sum_{i=1}^I \sum_{j=1}^J (x_{ij} - \text{tr}\{\mathbf{A}_i \mathbf{B}_j\})^2 \quad (10a)$$

$$= f(\mathfrak{A}, \mathfrak{B}) = \frac{1}{2} \|\mathbf{X} - \mathfrak{A}^\top \mathfrak{B}\|_F^2. \quad (10b)$$

The formulation in (10b) is equivalent to matrix factorization (with structural constraints), a problem known to be non-convex in general (e.g., [3]). However, when one matrix is fixed, the objective function w.r.t. the other matrix variable is convex. This observation motivated [3] to propose optimizing PSDMF in a scheme that alternates between two subproblems, one to update  $\mathfrak{A}$ , the other to update  $\mathfrak{B}$ . Since the objective function in (10) is symmetric in  $\mathfrak{A}$  and  $\mathfrak{B}$ , one can use the same optimization procedure for the two subproblems. This alternating scheme was proposed independently also in [13], [14], [29]; however, it was not motivated by convexity

arguments. This alternating scheme is outlined in [Algorithm 1](#), based on [\[3, Algorithm 1\]](#) and [\[14, Algorithm 1\]](#).

**Algorithm 1** Alternating strategy for PSDMF [\[3\]](#), [\[14\]](#), [\[29\]](#)

**Input:**  $\mathbf{X} \in \mathbb{R}_+^{I \times J}$ .

**Output:**  $\mathbf{A}_1, \dots, \mathbf{A}_I, \mathbf{B}_1, \dots, \mathbf{B}_J$ .

- 1: **Initialize**  $\mathbf{A}_1, \dots, \mathbf{A}_I, \mathbf{B}_1, \dots, \mathbf{B}_J$
- 2: **while** Convergence criterion not satisfied **do**
- 3:    $\{\mathbf{B}_j\}_{j=1}^J \leftarrow \text{update\_psd}(\mathbf{X}, \{\mathbf{A}_i\}_{i=1}^I, \{\mathbf{B}_j\}_{j=1}^J)$
- 4:    $\{\mathbf{A}_i\}_{i=1}^I \leftarrow \text{update\_psd}(\mathbf{X}^\top, \{\mathbf{B}_j\}_{j=1}^J, \{\mathbf{A}_i\}_{i=1}^I)$
- 5: **end while**

As for implementing the subproblems in [Algorithm 1](#), Vandaele et al. [\[3\]](#) developed dedicated algorithms and showed that they outperform the use of general convex solvers. In this paper, we adopt the approach of [\[3\]](#). Stark [\[14\]](#) optimized each subproblem using an semidefinite programming solver. Motivated by their probabilistic framework, [\[13\]](#) proposed minimizing a Kullback-Leibler divergence objective function using a non-linear L-BFGS optimization algorithm.

One of the methods proposed by [\[3\]](#) consists in minimizing the objective function  $f = f(\mathfrak{A}, \mathfrak{B})$  alternately w.r.t. the matrix variables  $\mathfrak{A}$  and  $\mathfrak{B}$  using gradient descent, where each update is followed by projecting the columns of  $\mathfrak{A}$  or  $\mathfrak{B}$  on  $\mathbb{S}_+^K$  in order to guarantee the psd structure. This approach is termed PGM [\[3\]](#). This type of projection is mentioned also in [\[29\]](#) and implied in [\[14\]](#). Vandaele et al. [\[3\]](#) proposed also a variant of PGM based on Nesterov’s accelerated gradient descent [\[31\]](#), called FPGM. PGM and FPGM implicitly assume that  $R_{A_i} = R_{B_j} = K$  for all  $i = 1, \dots, I$  and  $j = 1, \dots, J$ . The methods we shall present in [Section IV](#) do not have this limitation, and coincide with PGM and FPGM when this special case holds.

Another optimization approach proposed in [\[3\]](#) is based on the factor-based representation in [Section II-B](#), where now the objective function in [\(10a\)](#) is written as

$$f = f(\{\mathbf{U}_i\}_{i=1}^I, \{\mathbf{V}_j\}_{j=1}^J) = \frac{1}{2} \sum_{i=1}^I \sum_{j=1}^J (x_{ij} - \|\mathbf{U}_i^\top \mathbf{V}_j\|_F^2)^2. \quad (11)$$

The idea of [\[3\]](#) is to minimize [\(11\)](#) using a CD method operating on the entries of the factors  $\{\mathbf{U}_i\}_{i=1}^I$  and  $\{\mathbf{V}_j\}_{j=1}^J$  alternately. Instead of working on each scalar entry of  $\mathbf{U}_i$  and  $\mathbf{V}_j$ , ABG [\[34\]](#) minimizes [\(11\)](#) in a gradient descent approach w.r.t. each factor matrix. In the ABG algorithm in [\[35\]](#), the quadratic objective function in [\(11\)](#) is replaced with the generalized Kullback-Leibler divergence. CD and ABG can handle any values of the inner ranks, as is the case with the methods proposed in this paper.

### B. How is PSDMF Related to Phase Retrieval and ARM?

Assume for a moment that we are given a system of quadratic equations, where

$$y_i \cong |\langle \mathbf{u}_i, \mathbf{v} \rangle|^2 = |\mathbf{u}_i^\top \mathbf{v}|^2, \quad i = 1, \dots, I \quad (12)$$

and  $\mathbf{v} \in \mathbb{R}^K$  is unknown. This is “almost” a system of linear equations, the difference being that the phase, or sign, of  $\mathbf{v}$ ,

is not available. The problem of recovering a signal  $\mathbf{v} \in \mathbb{R}^K$  from phaseless measurements as in [\(12\)](#) given *sensing vectors*  $\mathbf{u}_i \in \mathbb{R}^K$ ,  $i = 1, \dots, I$ , is known as (generalized) *phase retrieval* (e.g., [\[23\]–\[28\]](#)).

Instead of addressing the unknown  $\mathbf{v}$  directly, it is possible to “lift” the quadratic measurements into linear measurements in the rank-one matrix  $\mathbf{B} = \mathbf{v}\mathbf{v}^\top$  [\[24\]](#). In this case, we can write [\(12\)](#) as

$$y_i \cong |\langle \mathbf{u}_i, \mathbf{v} \rangle|^2 = \text{tr}\{\mathbf{v}^\top \mathbf{u}_i \mathbf{u}_i^\top \mathbf{v}\} = \text{tr}\{\mathbf{A}_i \mathbf{B}\} = [\mathcal{A}(\mathbf{B})]_i, \quad (13)$$

where  $\mathbf{A}_i \cong \mathbf{u}_i \mathbf{u}_i^\top \in \mathbb{R}^{K \times K}$ , and  $\mathcal{A} : \mathbb{R}^{K \times K} \mapsto \mathbb{R}^I$  is an affine transformation that maps matrices to vectors.

We point out that classical phase retrieval often deals with complex-valued entities, whence the use of the term “phase” instead of sign. However, in this work, we are dealing with real-valued entities in PSDMF, and thus we restrict ourselves to real-valued terminology.

Equations [\(12\)](#) and [\(13\)](#) can be generalized from the vector case to an unknown matrix  $\mathbf{V} \in \mathbb{R}^{K \times R_B}$ ,

$$y_i \cong \|\mathbf{U}_i^\top \mathbf{V}\|_F^2 = \text{tr}\{\mathbf{V}^\top \mathbf{A}_i \mathbf{V}\} = \text{tr}\{\mathbf{A}_i \mathbf{B}\}, \quad (14)$$

where  $\mathbf{U}_i \in \mathbb{R}^{K \times R_A}$  for all  $i$ . The problem of recovering a low-rank matrix  $\mathbf{B} \in \mathbb{R}^{K \times K}$  of rank  $R_B$  from a set of linear measurements as in [\(14\)](#), not necessarily with a psd constraint on  $\mathbf{B}$ , is known as ARM.

ARM (e.g., [\[15\]–\[22\]](#)) can be stated as:

$$\min_{\mathbf{B}} \text{rank}(\mathbf{B}) \quad \text{s.t.} \quad \mathcal{A}(\mathbf{B}) = \mathbf{y} \quad (15)$$

where  $\mathbf{B} \in \mathbb{R}^{K_1 \times K_2}$  is the unknown matrix (not necessarily psd),  $\mathbf{y} \in \mathbb{R}^I$  is the vector of observations, and  $\mathcal{A} : \mathbb{R}^{K_1 \times K_2} \mapsto \mathbb{R}^I$  is a known linear mapping. ARM underlies numerous problems in the signal processing literature, including low-rank matrix recovery, matrix completion, and compressed sensing, to name a few (e.g., [\[17\]](#), [\[24\]](#), [\[44\]](#)). ARM is NP-hard and hard to approximate (e.g., [\[16\]](#)). Hence, numerous relaxations and variants have been proposed in the literature, among which we mention relaxations to the equality  $\mathcal{A}(\mathbf{B}) = \mathbf{y}$  using a quadratic loss (e.g., [\[20\]](#)), and relaxations to the rank constraint by optimizing over  $\mathbf{V} \in \mathbb{R}^{K \times R}$ , where  $\mathbf{B} \triangleq \mathbf{V}\mathbf{V}^\top$  (e.g., [\[18\]](#), [\[19\]](#), [\[25\]](#)). With this in mind, we are ready to show how phase retrieval and ARM are related to PSDMF.

Without loss of generality (w.l.o.g.), the quadratic objective function in [\(10\)](#) can be written as a sum of  $J$  terms:

$$f = f(\{\mathbf{A}_i\}_{i=1}^I, \{\mathbf{B}_j\}_{j=1}^J) = \sum_{j=1}^J f_j, \quad (16)$$

where

$$f_j = f_j(\{\mathbf{A}_i\}_{i=1}^I, \mathbf{B}_j) = \frac{1}{2} \sum_{i=1}^I (x_{ij} - \text{tr}\{\mathbf{A}_i \mathbf{B}_j\})^2 \quad (17a)$$

$$= \frac{1}{2} \|\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j)\|_2^2 \quad (17b)$$

$$= \frac{1}{2} \|\mathbf{x}_j - \mathfrak{A}^\top \mathbf{b}_j\|_2^2. \quad (17c)$$

In [\(17\)](#),  $\|\cdot\|_2$  is the Euclidean norm,  $\mathbf{x}_j \in \mathbb{R}^I$  is the  $j$ th column vector of  $\mathbf{X}$ , and  $\mathbf{b}_j \triangleq \text{vec}\{\mathbf{B}_j\}$ . The formulation in [\(17b\)](#) is

implicit in [29]. It follows from (16) and (17) that minimizing the objective function  $f = f(\{\mathbf{A}_i\}_{i=1}^I, \{\mathbf{B}_j\}_{j=1}^J)$  w.r.t. the psd matrix  $\mathbf{B}_j$  is equivalent to minimizing the objective function  $f_j = f_j(\{\mathbf{A}_i\}_{i=1}^I, \mathbf{B}_j)$  in (17) w.r.t. the same variable. This optimization problem can be written as

$$\min_{\mathbf{B}_j} \|\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j)\|_2^2 \quad \text{s.t.} \quad \mathbf{B}_j \in \mathbb{S}_+^K, \quad (18)$$

possibly also subject to the rank constraint

$$\text{rank}(\mathbf{B}_j) \leq R_{B_j}. \quad (19)$$

For a specific value of  $j$ , the optimization problem in (18) can be associated with the problem of estimating a matrix  $\mathbf{B}_j$  from the vector of observations  $\mathbf{x}_j$  given the linear operator  $\mathcal{A}$  and the observation model [29]

$$\mathbf{x}_j \cong \mathcal{A}(\mathbf{B}_j), \quad (20)$$

subject to additional structural constraints, e.g., psd and low-rank, on the variable  $\mathbf{B}_j$ . The key point is that the optimization problems in (18) and (19), which we wrote down as subproblems in alternating PSDMF optimization, can be optimized using existing methods in the literature that were developed for phase retrieval and ARM. The reverse holds as well: methods for optimizing PSDMF subproblems in the alternating framework that we have just described may, essentially, be used for phase retrieval and ARM.

Algorithm 2 outlines a subproblem in Line 3 in Algorithm 1 when the update of the variable  $\mathbf{B}_j$  is carried out using the approach that we have just described. More specifically, in Algorithm 2, Line 2 stands for the update of the variable  $\mathbf{B}_j$  using a phase retrieval or ARM method. Concrete examples will be given in Section IV. The fact that each subproblem updates each  $\mathbf{B}_j$  independently of the others allows for parallelization of the computations within each subproblem.

---

**Algorithm 2** Subproblem to update  $\{\mathbf{B}_j\}_{j=1}^J$  given  $\mathcal{A}$

---

**Input:**  $\mathbf{X} \in \mathbb{R}_+^{I \times J}$ ,  $\mathcal{A}$ ,  $\{\mathbf{B}_j\}_{j=1}^J$ .

**Output:**  $\{\mathbf{B}_j\}_{j=1}^J$  (updated).

- 1: **for**  $j = 1 : J$  **do**
  - 2:      $\mathbf{B}_j \leftarrow \text{PR\_or\_ARM\_algorithm}(\mathcal{A}(\mathbf{B}_j), \mathbf{x}_j)$
  - 3: **end for**
- 

### C. Alternating PSDMF Versus phase retrieval and ARM

Before moving on to specific numerical methods, we point out some fundamental differences between alternating PSDMF and the phase retrieval or ARM methods they rely on that must be taken into account in the algorithm design process. Probably the most important difference is that in phase retrieval and ARM, the transformation  $\mathcal{A}$  is given and known. Based on this fact, various strategies for initializing phase retrieval and ARM have been proposed. These initialization methods are often critical to guarantee convergence of the phase retrieval and ARM method to the desired solution, and play an important part in the analysis of the minimal number of observations required for reliable reconstruction of the desired signal (see, e.g., [25]). In the alternating PSDMF framework, however, the affine operator  $\mathcal{A}$  in Algorithm 2 consists of psd matrices

that are unknowns themselves. Hence, initialization methods for phase retrieval and ARM that are based on knowing the *true*  $\mathcal{A}$  cannot be applied to PSDMF. Consequently, methods that promise high convergence speed subject to appropriate initialization in the phase retrieval and ARM setting may perform poorly in the alternating PSDMF framework. For similar reasons, it is not clear to which extent convergence guarantees that were derived for phase retrieval and ARM are relevant to the alternating PSDMF framework. Another issue that is of utmost importance in phase retrieval and ARM is finding bounds on the number of measurements that guarantee an exact reconstruction of the desired signal. However, this question does not apply naturally to PSDMF because this analysis relies on the prerequisite that the *true* sensing vectors or matrices are known. Due to these differences, our interest in phase retrieval and ARM is restricted to borrowing algorithms that update the variables in each PSDMF optimization step.

## IV. PROJECTION-BASED ALGORITHMS FOR PSDMF

In this section, we proceed from concept to practice. The new PSDMF algorithms that we introduce in this section are based on SVP [20], an ARM method that we describe in Section IV-A. Inspecting SVP from the perspective of the link of PSDMF to phase retrieval and ARM makes it natural to extend PGM [3] to a framework that can handle any value of inner ranks. Our SVP-based alternating PSDMF algorithm, and its accelerated variant FSVP, are described in Section IV-B. In Section IV-C, we present another PSDMF algorithm, based on NIHT [22]. In Section IV-D, we describe our CGIHT-based PSDMF algorithm. In Section IV-E, we show the link between SVP and a majorization-minimization-based algorithm for phase retrieval [26].

### A. Background: Singular Value Projection

Consider the objective function  $f_j(\mathbf{B}_j)$  in (17) for a given  $\mathcal{A}$  and a specific value of  $j$ . This function is convex in  $\mathbf{B}_j$ . In this section, we discuss minimizing  $f_j(\mathbf{B}_j)$  w.r.t.  $\mathbf{B}_j$  subject to  $\text{rank}(\mathbf{B}_j) \leq R_{B_j}$ . The set of low-rank matrices is not convex, and thus this optimization problem is non-convex, in general, when  $R_{B_j} < K$ . This problem was formulated by [20] as a robust variant of the ARM problem in (15). In [20], the matrix  $\mathbf{B}_j$  was not constrained to be psd. The first step in Jain et al.'s [20] approach involves gradient descent. To simplify our notation, we omit the index  $j$  and write  $\psi(\mathbf{B})$  instead of  $f_j(\mathbf{B}_j)$ . The gradient of  $\psi(\mathbf{B})$  w.r.t.  $\mathbf{B}$  can take any of the following forms:

$$\nabla \psi(\mathbf{B}) = \mathcal{A}^\dagger(\mathcal{A}(\mathbf{B}) - \mathbf{y}) = \sum_{i=1}^I [(\mathcal{A}(\mathbf{B}) - \mathbf{y})]_i \mathbf{A}_i \quad (21a)$$

$$= \sum_{i=1}^I (\text{tr}\{\mathbf{A}_i^\top \mathbf{B}\} - y_i) \mathbf{A}_i, \quad (21b)$$

where the conjugate map  $\mathcal{A}^\dagger$  was defined in Section I-D. Now, a gradient step based on (21) does not take into account the low-rank structure of  $\mathbf{B}_j$ . Instead, the rank constraint is imposed by orthogonal projection of the updated version of  $\mathbf{B}$

onto the set of low-rank matrices, an operation that consists of taking the truncated singular value decomposition (SVD) of  $\mathbf{B}$ . This procedure is termed SVP in [20].

The psd variant of SVP was addressed by [19], who used a few iterations of SVP to initialize another non-convex ARM algorithm. In the psd case, the projection of a matrix on the set of  $K \times K$  psd matrices of rank at most  $R$  is denoted by  $H_{\mathbb{S}_+^K, R}(\cdot)$ . This projection can be computed as  $H_{\mathbb{S}_+^K, R}(\mathbf{B}) = \mathbf{V}\mathbf{\Lambda}_R\mathbf{V}^\top$ , where  $\mathbf{\Lambda}_R \in \mathbb{R}^{R \times R}$  is a diagonal nonnegative matrix with the  $R$  largest nonnegative eigenvalues of  $\mathbf{B}$  on its main diagonal, and the columns of  $\mathbf{V} \in \mathbb{R}^{K \times R}$  are the eigenvectors of  $\mathbf{B}$  associated with these  $R$  largest nonnegative eigenvalues. If  $\mathbf{B}$  has less than  $R$  positive eigenvalues, some of the diagonal values of  $\mathbf{\Lambda}_R$  will be zero, and the corresponding columns of  $\mathbf{V}$  can be zero vectors as well. The letter ‘‘H’’ is reminiscent of the fact that  $H_{\mathbb{S}_+^K, R}(\cdot)$  is a hard thresholding operator (see also [30]). Note that in the psd case, we cannot use SVD or any other method that extracts the eigenvectors by the magnitude of the  $R$  leading eigenvalues because we must have access to the signs of the eigenvalues.

SVP with psd constraint on the unknown matrix  $\mathbf{B}$  is outlined in Algorithm 3, based on [20, Algorithm 1]. The update rule based on projected gradient descent is given in Line 3 of Algorithm 3, where  $\eta$  is the step size. The zero initialization in Line 1 of Algorithm 3 was proposed by [20]. Other initialization procedures, such as *spectral initialization* [25]:  $\mathbf{B} \leftarrow \mathcal{A}^\dagger(\mathbf{y})$ , are possible. Jain et al. [20] prove that SVP can converge to a desired low-rank solution if the step size  $\eta$  is smaller than a certain bound that depends on geometric properties of  $\mathcal{A}$  and on the rank of the desired solution.

---

**Algorithm 3** SVP Algorithm [20]—the psd case.

---

**Input:**  $\mathcal{A}$ ,  $\mathbf{y}$ ,  $R$ ,  $\eta$

**Output:**  $\mathbf{B}$  of  $\text{rank}(\mathbf{B}) \leq R$ .

- 1: **Initialize:**  $\mathbf{B} \leftarrow \mathbf{0}$
  - 2: **while** stopping criterion not satisfied **do**
  - 3:      $\mathbf{B} \leftarrow H_{\mathbb{S}_+^K, R}(\mathbf{B} - \eta\mathcal{A}^\dagger(\mathcal{A}(\mathbf{B}) - \mathbf{y}))$
  - 4: **end while**
- 

### B. A PSDMF Algorithm Based on SVP

Our proposed SVP-based PSDMF algorithm is constructed by using the update step of SVP to optimize the subproblems in Algorithm 2, along with the necessary adaptations to the alternating framework. Algorithm 4 outlines one subproblem to update  $\{\mathbf{B}_j\}_{j=1}^J$ , given  $\mathcal{A}$  and  $\{\mathbf{B}_j\}_{j=1}^J$  from the previous subproblem (see Algorithm 1).

---

**Algorithm 4** One subproblem of SVP-based PSDMF to update  $\{\mathbf{B}_j\}_{j=1}^J$

---

**Input:**  $\mathbf{X}$ ,  $\mathcal{A}$ ,  $\mathbf{B}_1, \dots, \mathbf{B}_J$ ,  $R_{B_1}, \dots, R_{B_J}$ ,  $D$ .

**Output:**  $\mathbf{B}_1, \dots, \mathbf{B}_J$ .

- 1:  $\eta \leftarrow (\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$
  - 2: **for**  $j = 1 : J$  **do**
  - 3:     **for**  $d = 1 : D$  **do**
  - 4:          $\mathbf{B}_j \leftarrow H_{\mathbb{S}_+^K, R_{B_j}}(\mathbf{B}_j - \eta\mathcal{A}^\dagger(\mathcal{A}(\mathbf{B}_j) - \mathbf{x}_j))$
  - 5:     **end for**
  - 6: **end for**
- 

The optional parameter  $D$  in Algorithm 4 determines the number of inner iterations [3]. In the special case that  $R_{A_j} = R_{B_j} = K$  for all  $i$  and  $j$ , the SVP-based PSDMF described in Algorithm 4 coincides with PGM [3]. In [3], Vandaele et al. proposed a variant to PGM in which the gradient step is replaced by  $D$  steps of Nesterov-based accelerated gradient descent [31]. It is thus natural to propose an accelerated variant of Algorithm 4 that will subsume FPGM [3]. In this case, we replace the loop on  $D$  in Algorithm 4 with:

- 1:  $\mathbf{B}_j^{\text{prev}} \leftarrow \mathbf{B}_j$
- 2: **for**  $d = 1 : D$  **do**
- 3:      $\mathbf{Y} \leftarrow \mathbf{B}_j + \frac{d-2}{d+1}(\mathbf{B}_j - \mathbf{B}_j^{\text{prev}})$
- 4:      $\mathbf{B}_j^{\text{prev}} \leftarrow \mathbf{B}_j$
- 5:      $\mathbf{B}_j \leftarrow H_{\mathbb{S}_+^K, R_{B_j}}(\mathbf{Y} - \eta\mathcal{A}^\dagger(\mathcal{A}(\mathbf{Y}) - \mathbf{x}_j))$
- 6: **end for**

We call this variant *FSVP* (henceforth, we omit the suffix ‘‘-PSDMF’’ when the context is clear). FSVP reduces to FPGM when all inner ranks are equal to  $K$ . When  $D = 1$ , FSVP is equivalent to SVP. Note that in a non-alternating framework,  $D$  is simply the number of iterations of the accelerated gradient descent algorithm until a stopping criterion is achieved. Thus, in order to benefit from the acceleration, one has to choose a sufficiently large value of  $D$ . However, due to the alternating framework,  $D$  should not be too large, otherwise the performance of FSVP degrades again, as demonstrated by [3]. The optimal value of  $D$  depends on many factors, including the values in  $\mathbf{X}$ , the factorization parameters, and the stopping criterion. Hence, in practice,  $D$  is chosen based on empirical evaluation [3]; see Section V-A3.

As for the step size in Line 1 of Algorithm 4, in (F)PGM [3] ( $R = K$ ), the step size within each subproblem is fixed and equal to  $\eta = 1/L$ , where  $L = \lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top)$  is the Lipschitz constant of the gradient  $\nabla\psi(\mathbf{B}_j)$  [see (21)],  $\mathfrak{A}\mathfrak{A}^\top = \sum_{i=1}^I \text{vec}\{\mathbf{A}_i\}\text{vec}^\top\{\mathbf{A}_i\}$ , and  $\lambda_{\max}(\cdot)$  is the leading eigenvalue of its operand. In numerical experiments, we observed that the objective function decreased monotonically also for SVP and FSVP, i.e.,  $R < K$ , with the same step size  $\eta = (\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$ . This empirical observation is not obvious, because this  $\eta$  is no longer guaranteed to be the optimal fixed step size when projecting on the non-convex set of low-rank matrices, as noted, e.g., in [20].

### C. A PSDMF Algorithm Based on NIHT

Since it was first proposed, several improvements to the basic SVP algorithm appeared, see, e.g., [20], [33], [45], [46] and references therein. It is thus natural to consider these variants as candidates for more efficient PSDMF methods, and to see if they can offer numerical advantages also in the alternating PSDMF framework. We now describe a PSDMF algorithm based on NIHT [22], [32]. One subproblem to update  $\mathbf{B}_j$ ,  $j = 1, \dots, J$ , based on NIHT, is outlined in Algorithm 5. Each update step of  $\mathbf{B}_j$  in Algorithm 5 is identical to an update step in the original NIHT algorithm.



---

**Algorithm 5** One subproblem of NIHT-based PSDMF to update  $\{\mathbf{B}_j\}_{j=1}^J$

---

**Input:**  $\mathbf{X} \in \mathbb{R}_+^{I \times J}$ ,  $\mathcal{A}$ ,  $\mathbf{B}_1, \dots, \mathbf{B}_J$ ,  $R_{B_1}, \dots, R_{B_J}$ .

**Output:**  $\mathbf{B}_1, \dots, \mathbf{B}_J$ .

- 1: **for**  $j = 1 : J$  **do**
  - 2:    $\mathbf{U} \leftarrow R_{B_j}$  eigenvectors of  $\mathbf{B}_j$  associated with the  $R_{B_j}$  largest nonnegative eigenvalues of  $\mathbf{B}_j$
  - 3:    $\mathbf{P}_U \leftarrow \mathbf{U}\mathbf{U}^\top$
  - 4:    $\eta \leftarrow \frac{\|\mathbf{P}_U \mathcal{A}^\dagger(\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j))\|_F^2}{\|\mathcal{A}(\mathbf{P}_U \mathcal{A}^\dagger(\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j)))\|_2^2}$
  - 5:    $\mathbf{B}_j \leftarrow \mathbb{H}_{\mathbb{S}_+^{K, R_{B_j}}}(\mathbf{B}_j - \eta \mathcal{A}^\dagger(\mathcal{A}(\mathbf{B}_j) - \mathbf{x}_j))$
  - 6: **end for**
- 

Our NIHT-based method in [Algorithm 5](#) differs from SVP ([Algorithm 4](#)) in the evaluation of the step size  $\eta$  for the gradient descent. Compared with SVP, which has a fixed step size, the step size in NIHT is adaptive. In each iteration, the current estimate of  $\mathbf{B}_j$  is updated along the gradient descent direction with the locally steepest descent stepsize, followed by thresholding to the manifold of rank- $R_{B_j}$  matrices [22], [47]. When  $R_{B_j} < K$ ,  $\eta$  in [Line 4](#) of [Algorithm 5](#) depends not only on  $\mathcal{A}$  but also on  $\mathbf{B}_j$ , and may be different for different  $\mathbf{B}_j$ . When  $R_{B_j} = K$ , the projection operator is the identity matrix:  $\mathbf{P}_U = \mathbf{U}\mathbf{U}^\top = \mathbf{I}$ , and thus  $\eta$  no longer depends on  $\mathbf{B}_j$ . However, for any choice of  $\mathbf{P}_U$ ,  $\mathcal{A}$ ,  $\mathbf{x}_j$ , and  $\mathbf{B}_j$ , the step size of NIHT is never smaller than that of (F)SVP:  $\eta^{\text{NIHT}} \geq \eta^{\text{(F)SVP}} = (\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$ . To see why this inequality always holds, let  $\mathbf{M} \neq \mathbf{0}$  be an arbitrary  $K \times K$  matrix. Then,

$$\frac{\|\mathcal{A}(\mathbf{M})\|_2^2}{\|\mathbf{M}\|_F^2} = \frac{\|\mathfrak{A}^\top \text{vec}\{\mathbf{M}\}\|_2^2}{\|\text{vec}\{\mathbf{M}\}\|_2^2} \leq \sqrt{\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top)}, \quad (22)$$

where the inequality follows from the definition of the spectral norm. Setting  $\mathbf{M} = \mathbf{P}_U \mathcal{A}^\dagger(\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j))$  in (22), the left-hand side of (22) is equal to  $(\eta^{\text{NIHT}})^{-1}$ . Hence,  $\eta^{\text{NIHT}} \geq \eta^{\text{(F)SVP}} = (\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$ . This inequality provides an intuitive explanation why our NIHT-based PSDMF algorithm can achieve the same model fit error with fewer iterations than our SVP-based PSDMF that has step size  $(\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$ , as we shall demonstrate in [Section V](#). Even more interesting is our observation that in general, our NIHT-based method also outperforms FSVP in terms of number of iterations required to achieve the same model fit error, as we show in [Section V](#). The latter fact is significant because NIHT does not require to estimate or adjust an additional acceleration parameter  $D$  as is the case with FPGM and FSVP in order to achieve its best performance. Another noteworthy property of NIHT is that its step size does not guarantee a monotonous decrease of the objective function [22], in contrast to ABG [34], [35] and CD [3]. As we shall demonstrate in [Section V-C1](#), this property may explain why in certain cases, our NIHT-based algorithm is the only method able to properly minimize the objective function.

#### D. A PSDMF Algorithm based on CGIHT

CGIHT [33] was proposed as an improvement to NIHT [22] by combining the fast asymptotic convergence rate of the conjugate gradient method with the low per-iteration complexity of hard thresholding methods. One subproblem to

update  $\mathbf{B}_j$ ,  $j = 1, \dots, J$ , based on CGIHT [33], is outlined in [Algorithm 6](#).

---

**Algorithm 6** One subproblem of CGIHT-based PSDMF to update  $\{\mathbf{B}_j\}_{j=1}^J$

---

**Input:**  $\mathbf{X} \in \mathbb{R}_+^{I \times J}$ ,  $\mathcal{A}$ ,  $\mathbf{B}_1, \dots, \mathbf{B}_J$ ,  $R_{B_1}, \dots, R_{B_J}$ ,  $D$ .

**Output:**  $\mathbf{B}_1, \dots, \mathbf{B}_J$ .

- 1: **for**  $j = 1 : J$  **do**
  - 2:    $\mathbf{Q} = \mathbf{0}$ ,  $d \leftarrow 0$ ,
  - 3:   **for**  $d = 1 : D$  **do**
  - 4:      $\mathbf{U} \leftarrow R_{B_j}$  eigenvectors of  $\mathbf{B}_j$  associated with the  $R_{B_j}$  largest nonnegative eigenvalues of  $\mathbf{B}_j$ ,
  - 5:      $\mathbf{G} \leftarrow \mathcal{A}^\dagger(\mathbf{y} - \mathcal{A}(\mathbf{B}_j))$
  - 6:      $\mathbf{P}_U \leftarrow \mathbf{U}\mathbf{U}^\top$
  - 7:     **if**  $d = 1$  **then**
  - 8:        $\beta \leftarrow 0$
  - 9:     **else**
  - 10:        $\beta \leftarrow \frac{\langle \mathcal{A}(\mathbf{P}_U \mathbf{G}), \mathcal{A}(\mathbf{P}_U \mathbf{Q}) \rangle}{\|\mathcal{A}(\mathbf{P}_U \mathbf{Q})\|_2^2}$
  - 11:     **end if**
  - 12:      $\mathbf{Q} \leftarrow \mathbf{G} + \beta \mathbf{Q}$
  - 13:      $\eta \leftarrow \frac{\langle \mathbf{P}_U \mathbf{G}, \mathbf{P}_U \mathbf{Q} \rangle}{\|\mathcal{A}(\mathbf{P}_U \mathbf{Q})\|_2^2}$
  - 14:      $\mathbf{B}_j \leftarrow \mathbb{H}_{\mathbb{S}_+^{K, R}}(\mathbf{B}_j - \eta \mathbf{Q})$
  - 15:   **end for**
  - 16: **end for**
- 

Each update step of  $\mathbf{B}_j$  in [Algorithm 6](#) is identical to an update step in the original CGIHT algorithm. If the orthogonalization weight  $\beta$  is zero, CGIHT becomes equivalent to NIHT. For  $\beta$  to be different from zero, we need at least one inner iteration,  $D \geq 2$ , in [Algorithm 6](#). Matrix  $\mathbf{Q}$  in [Line 12](#) defines the search direction, and  $\eta$  the step size.

Similarly to other phase retrieval and ARM methods, the stability and recovery guarantees of NIHT and CGIHT depend on satisfying conditions on the restricted isometry constants of the sensing operators. However, these conditions are generally not satisfied within the alternating PSDMF optimization framework. As our numerical experiments show, the error evolution trajectories of CGIHT are often irregular and erratic. Nevertheless, as we shall show in [Section V](#), such a behaviour can in fact turn out useful. In order to improve the stability of our algorithm, the following rules were implemented: if  $\beta$ ,  $\eta$ , or the norm of the gradient, become excessively large, we set them to zero.

#### E. A Link Between SVP, MM, and PRIME-Power

In [26, Sec. III.D], Qiu et al. proposed an algorithm for phase retrieval, called PRIME-Power, whose derivation is based purely on majorization-minimization considerations. The objective function minimized by PRIME-Power is equivalent to (17b) for a specific value of  $j$ . We establish a new connection between PRIME-Power and gradient descent. More specifically, we now show that PRIME-Power is equivalent to SVP [20] in the psd case ([Algorithm 3](#)) for  $R = 1$ , and that the same majorization-minimization procedure in [26] leads to SVP (in the psd case) for any  $R \leq K$ . Specifically, in [26], Qiu et al. use majorization-minimization considerations to construct a tight majorizer to (17b) that we denote  $h$ . Let

$\mathbf{C}$  denote the previous value of  $\mathbf{B}_j$ , and  $\mathbf{B}_j$  is the variable to update. By construction of  $h(\mathbf{B}_j | \mathbf{C})$  as a majorization function of  $f_j(\mathbf{B}_j)$  at the point  $\mathbf{C}$ ,  $h(\mathbf{B}_j | \mathbf{C})$  satisfies [26]:

$$h(\mathbf{B}_j | \mathbf{C}) \geq f_j(\mathbf{B}_j) \quad \text{for all } \mathbf{B}_j \in \mathbb{R}^{K \times K} \quad (23a)$$

$$h(\mathbf{C} | \mathbf{C}) = f_j(\mathbf{C}). \quad (23b)$$

A function  $h(\mathbf{B}_j | \mathbf{C})$  that majorizes the objective  $f_j(\mathbf{B}_j)$  in (17b) is given in [26, Eq. (28)]:

$$h(\mathbf{B}_j | \mathbf{C}) \triangleq \gamma \text{tr}\{\mathbf{B}_j \mathbf{B}_j\} + 2 \sum_{i=1}^I \text{tr}\{\mathbf{B}_j \mathbf{A}_i\} \text{tr}\{\mathbf{C} \mathbf{A}_i\} - 2\gamma \text{tr}\{\mathbf{B}_j \mathbf{C}\} - \sum_{i=1}^I 2x_{ij} \text{tr}\{\mathbf{A}_i \mathbf{B}_j\}, \quad (24)$$

where  $\gamma \geq \lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top)$ . Equation (24) and the bound on  $\gamma$  were derived in [26] using the psd matrices  $\mathbf{B}$  and  $\mathbf{A}_i$ , without making any assumption about their ranks. Our key observation is that the value of  $\mathbf{B}_j$  satisfying  $\frac{\partial h(\mathbf{B}_j | \mathbf{C})}{\partial \mathbf{B}_j} = 0$  is:

$$\mathbf{B}_j^{\text{opt}} = \mathbf{C} + \frac{1}{\gamma} \sum_{i=1}^I (y_i - \text{tr}\{\mathbf{A}_i \mathbf{C}\}) \mathbf{A}_i, \quad (25)$$

which, using (21), leads directly to the update step of SVP in Algorithm 3, where the rank constraint of PRIME-Power is imposed by the projection operator  $\mathbb{H}_{\mathbb{S}_+^K, R}(\cdot)$  with  $R = 1$ . Hence, the majorization-minimization update step of PRIME-Power is equivalent to the gradient-based update step of SVP in the  $R = 1$  psd case. Note also that the step size  $\frac{1}{\gamma} \leq (\lambda_{\max}(\mathfrak{A}\mathfrak{A}^\top))^{-1}$  in (25), which was obtained in [26] from majorization-minimization considerations, is the same step size arising from the Lipschitz constant of the gradient, see our discussion in Section IV-B. Next, given that the derivation of  $h$  and  $\gamma$  in [26] did not rely on the rank of the psd matrices  $\mathbf{A}_i$ , we conclude that SVP [20] (in the psd case) is equivalent to a straightforward extension of PRIME-Power to the recovery of full-column-rank  $K \times R$  matrices, instead of  $K \times 1$  vectors, from their phaseless measurements. The fact that SVP [20] coincides with a majorization-minimization-based method (in the psd case) can serve as a reminder that SVP-based PSDMF methods can equally be regarded as being derived based on majorization-minimization considerations. We mention that the fact that the gradient update step in SVP is related to majorization-minimization (regardless of the link to the PRIME-Power method) can be deduced directly by noting that the quadratic objective function in (10) is strongly convex in  $\mathfrak{A}$  (when  $\mathfrak{B}$  is fixed) with a Lipschitz continuous gradient.

## V. NUMERICAL EXPERIMENTS

In this section, we exhibit the potential of algorithms for PSDMF based on phase retrieval and ARM optimization methods. We also illustrate numerically some of the properties of PSDMF that we discussed theoretically in Section II. We consider the projection-based PSDMF algorithms introduced in Section IV: SVP, FSVP, NIHT and CGIHT. We consider also the ABG methods with their two types of objective functions: quadratic [34] and generalized Kullback-Leibler divergence [35]. The latter is denoted ABG-P in our plots,

where ‘P’ stands for Poisson log-likelihood, which is the likelihood function associated with the generalized Kullback-Leibler divergence. Among the methods that were not designed based on phase retrieval or ARM principles, we focus on CD [3] as the main competing method. We do not compare with the algorithms in [13], [14] because each of them has some restriction on the model or on the type of data addressed, as discussed in Section I-C. We consider both cyclic and greedy (also known as Gauss-Southwell (GS)) variants of CD [3], where we set the “greediness” coefficient to 0.5, as in [3]. We implement ABG as in [34], [35]. The backtracking line search parameters of ABG are set to  $\alpha = 0.1$  and  $\beta = 0.35$ . These values were chosen after verifying they provided satisfying performance in our experiments.

### A. Numerical Issues

1) *Initialization*: We initialize the algorithms with factors whose entries are drawn independently from the standard normal distribution  $\mathcal{N}(0, 1)$ . We normalize the input matrix to  $\|\mathbf{X}\|_F = 1$ . We then scale one set of initial factors:  $\mathbf{U}_i \leftarrow \sqrt{\lambda_*} \mathbf{U}_i$ ,  $i = 1, \dots, I$ , where  $\lambda_* = \arg \min_{\lambda} \|\mathbf{X} - \lambda \mathfrak{A}^\top \mathfrak{B}\|_F^2$ . This scaling procedure, suggested in [3], along with normalizing the input matrix, turned out to improve the convergence properties in our experiments. As explained in Section III-C, strategies that are useful for phase retrieval and ARM, such as spectral initialization, cannot be applied to PSDMF.

2) *Figure of Merit*: The relative model fit error (RMFE) is defined as  $\frac{\|\mathbf{X} - \hat{\mathbf{X}}\|_F}{\|\mathbf{X}\|_F}$ , where  $\hat{\mathbf{X}} \triangleq \hat{\mathfrak{A}}^\top \hat{\mathfrak{B}}$  denotes the approximation of  $\mathbf{X}$  based on the approximated model parameters  $\hat{\mathfrak{A}}$  and  $\hat{\mathfrak{B}}$  when a stopping criterion is achieved. As a stopping criterion, we use a tolerance on the RMFE:  $\frac{\|\mathbf{X} - \hat{\mathbf{X}}\|_F}{\|\mathbf{X}\|_F} \leq \text{To1RMFE}$ . We also use a tolerance on the relative change in the quadratic model fit error (QMFE):  $\frac{|f^{\ell+1} - f^\ell|}{f^\ell} < \text{To1Fun}$ , where the QMFE is defined as  $f^\ell \triangleq \frac{1}{2} \|\mathbf{X} - \mathfrak{A}^\ell \mathfrak{B}^\ell\|_F^2$ , and  $\mathfrak{A}^\ell, \mathfrak{B}^\ell$  are the estimates of  $\mathfrak{A}, \mathfrak{B}$  at iteration index  $\ell$ . By default,  $\text{To1Fun} = 0 = \text{To1RMFE}$ .

3) *Choosing  $D$* : In our experiments, we report on the number of *overall iterations*  $\ell \times D$ , where  $\ell$  is the (outer) iteration index. Together with the computational complexity in Table I and Section V-A6, the number of overall iterations gives an idea about the amount of computation needed to achieve a stopping criterion, regardless of a specific implementation or programming platform.

We set  $D = 1$  for SVP, NIHT, ABG, and ABG-P. For methods that rely on acceleration—FSVP and CGIHT: in some cases, we made preliminary tests on candidate values of  $D$ . However, it is not convenient nor practical to run a preliminary test of the optimal value of  $D$  whenever we wish to use an algorithms. Therefore, in the remainder of cases, we chose  $D_{\text{FSVP}}$  and  $D_{\text{CGIHT}}$  arbitrarily, with values similar to those that turned out useful in other experiments. In each experiment, we specify the values of  $D_{\text{FSVP}}$  and  $D_{\text{CGIHT}}$  that we use.

4) *CPU Time*: In our plots showing error evolution versus CPU time, all methods are coded in Matlab R2019a and run on a MacBook Pro with a 2.8GHz Intel Core i7 processor and 16 GB memory.

TABLE I  
COMPUTATIONAL COMPLEXITY PER SUBPROBLEM: UPDATING  $\{\mathbf{B}_j\}_{j=1}^J$

Method	Computational Complexity
SVP & FSVP	$\mathcal{O}(\min(IK^4, I^2K^2) + DIJK^2)$
NIHT & CGIHT	$\mathcal{O}(DIJK^2)$
CD	$\mathcal{O}(IJK^2 + IK^4R)$
ABG & ABG-P	$\mathcal{O}(IJK^2)$

5) *Code and Implementation:* We use the code in [48] to generate the geometric data matrices and run the CD algorithms, which are written in c and thus run faster (Matlab does not handle loops efficiently [3]), in our Monte Carlo (MC) trials. For runtime comparison, however, we implemented the CD algorithms of [3] in Matlab. Code for the algorithms proposed in this paper is available at <https://www.dana.lahat.org.il/psdmf.html>.

6) *Complexity:* In Table I, we compare the per-subproblem computational complexity of the different methods in updating  $\{\mathbf{B}_j\}_{j=1}^J$ . Deducing the results for the other set of variables is straightforward. The purpose is to show how each algorithm scales in each of the dimensions of the problem. The cost of the projection-based methods is dominated by: (i) computing the Lipschitz constant for the step size in SVP and FSVP, which involves an SVD of  $\mathfrak{A} \in \mathbb{R}^{K^2 \times I}$  and thus costs  $\mathcal{O}(\min(K^4I, K^2I^2))$ , (ii) computing the gradient  $\mathcal{A}^\dagger(\mathbf{x}_j - \mathcal{A}(\mathbf{B}_j))$ , which costs  $\mathcal{O}(IK^2)$  per  $j$ , (iii) projecting on  $\mathbb{S}_+^K$ , which requires an eigenvalue decomposition of  $\mathbf{B}_j$  for each  $j$ , and costs  $\mathcal{O}(K^3)$ , (iv) projecting  $\mathbf{B}_j$  on the set of rank- $R$  matrices, which costs  $\mathcal{O}(K^2R)$ . In NIHT, computing the matrix  $\mathbf{U}$  when  $R < K$  requires an additional eigenvalue decomposition of  $\mathcal{O}(K^3)$ , and computing the adjoint operator costs  $\mathcal{O}(IK^2)$ , per  $j$ . Since in PSDMF we always have  $K \leq I$  and  $R \leq K$ , the cost of the eigenvalue decomposition and the projection on rank- $R$  psd matrices are dominated by  $\mathcal{O}(IK^2)$ . Within each inner iteration, CGIHT requires more computations—in fact, approximately twice—compared to NIHT; however, these computations are of the same nature as those for NIHT and thus they scale the same in the dimensions of the problem. The computational complexity of one subproblem in CD is given in [3]. However, in contrast to [3], here we do not omit the term  $\mathcal{O}(IJK^2)$  because it might dominate  $\mathcal{O}(IK^4R)$  when  $I > K^3$ . The computational complexity of ABG is dominated by  $\mathcal{O}(IJK^2)$  [34]. Computing the denominator in the gradient in ABG-P has the same computational complexity as the computation of the gradient of ABG and thus the overall computational complexity does not change.

### B. Performance Comparison: Euclidean Distance Matrices

Consider an  $I \times I$  matrix  $\mathbf{D}_I$  whose  $(i, j)$ th entry is equal to  $d_{ij} = (\alpha_i - \alpha_j)^2$ , where  $\alpha_i$  are real numbers and  $\alpha_i \neq \alpha_j$  for any  $i, j = 1, \dots, I$ . Thus,  $d_{ii} = 0$  for all  $i$ . The psd rank of  $\mathbf{D}_I$  is equal to 2 and its inner ranks are all equal to 1, because  $\mathbf{D}_I$  admits a PSDMF with factors  $\mathbf{u}_i \triangleq [\alpha_i \ 1]^\top$  and  $\mathbf{v}_j \triangleq [1 \ -\alpha_j]^\top$  for all  $i, j$  [1]. The usual matrix rank of  $\mathbf{D}_I$  is 3 for  $I \geq 3$ , whereas its nonnegative rank becomes

TABLE II  
NUMBER OF SUCCESSFUL FACTORIZATIONS OF  $100 \times 100$  EUCLIDEAN DISTANCE MATRICES IN 100 MC TRIALS.

SVP	FSVP	NIHT	ABG	ABG-P	CD cyc	CD GS	CGIHT
1	2	37	0	0	1	9	91

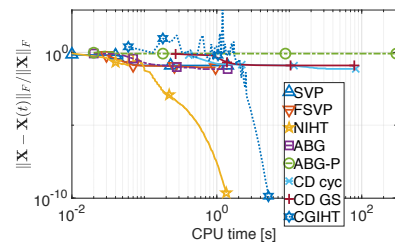


Fig. 1. Evolution of the relative model fit error in the PSDMF of a  $100 \times 100$  Euclidean distance matrix with  $K = 2$ ,  $R_A = 1 = R_B$ .

arbitrarily large as  $I \rightarrow \infty$  [40], [49]. This matrix is known as Euclidean distance matrix (EDM).

In this experiment, we compare the ability of the algorithms to factorize  $\mathbf{D}_I$ . We draw the values of  $\alpha_i$  independently from the standard uniform distribution:  $\alpha_i \sim \mathcal{U}[0, 1]$ . We set  $I = 100$ , and run 100 MC trials, each with a new initialization and a new draw of  $\alpha_i$  for all  $i$ . We fit  $\mathbf{D}_I$  to a PSDMF model with  $K = 2$  and  $R_A \triangleq R_{A_i} = 1$  for all  $i$ ,  $R_B \triangleq R_{B_j} = 1$  for all  $j$ . In this case,  $N_{\text{data}} = 10^4 > N_{\text{model}} = 396$ . Our stopping criterion is  $\text{ToIFun} = 10^{-15}$ .

In this setting, we obtain two distinct and clearly-separated clusters of results: one consists of trials that did not decrease the RMFE below  $\approx 10^{-2}$ , which is a rather large value. The other cluster consists of the successful trials, defined as achieving  $\text{RMFE} \leq 10^{-4}$ . For CGIHT and FSVP, we tested several candidate values of  $D$  and found that  $D_{\text{FSVP}} = 19$  and  $D_{\text{CGIHT}} = 14$  were associated with higher rates of successful factorizations than other values of  $D$  that we tried. Increasing  $D$  for other methods did not provide any substantial improvement in the rate of success, as expected. Fig. 1 exemplifies the trajectory of each algorithm in one trial and demonstrates the separation into the two categories of “success” and “failure”. Table II shows the number of successful factorizations per algorithm.

A possible explanation of the dominance of NIHT and its accelerated variant CGIHT can be seen from the irregular and non-monotonous error trajectories in Fig. 1. Their trajectories drop sharply (in the log-log scale) at a certain point. The other algorithms decrease the objective function monotonically until they reach a plateau. This is in agreement with the fact that NIHT and CGIHT are the only methods among the ones we compare that do not have a guarantee for monotone decrease of the objective function. We postulate that this property allows NIHT and CGIHT to escape certain local stationary points that the other methods get trapped in.

### C. Performance Comparison: Geometric Data

In this section, we compare our algorithms on slack matrices, which are matrices associated with the geometry of

polytopes. Finding the psd rank of such matrices, and sometimes also the inner ranks, is one of the central applications of PSDMF (e.g., [1], [3], [4]). In these applications, one is interested in exact PSDMF or at least in a good approximation thereof [37]. As demonstrated by [3], even the factorization of small slack matrices can be numerically challenging. The matrices considered in this section were studied also in [3].

1) *Submatrix of the slack matrix of the correlation polytope:* Consider a  $2^n \times 2^n$  binary matrix  $\mathbf{M}_n$  whose rows and columns are indexed by vectors  $\mathbf{c}, \mathbf{d} \in \{0, 1\}^n$  such that  $\mathbf{M}_n(\mathbf{c}, \mathbf{d}) = (1 - \mathbf{c}^\top \mathbf{d})^2$ . The psd rank of  $\mathbf{M}_n$  is equal to  $n + 1$  and its inner ranks are all equal to 1, because  $\mathbf{M}_n(\mathbf{c}, \mathbf{d})$  admits a PSDMF with factors  $\mathbf{u} \triangleq [1 \ -\mathbf{c}^\top]^\top$  and  $\mathbf{v} \triangleq [1 \ \mathbf{d}^\top]^\top$  [2]. This matrix is known as a submatrix of the slack matrix of the correlation polytope [2]. As an example,  $\mathbf{M}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$ .

In this experiment, we compare the ability of the algorithms to factorize  $\mathbf{M}_n$ . For each value of  $n = 2, \dots, 7$ , we run 100 MC trials, each with a new initialization. We fit  $\mathbf{M}_n$  to a PSDMF model with  $K = n + 1$  and  $R_A \triangleq R_{A_i} = 1$  for all  $i$ ,  $R_B \triangleq R_{B_j} = 1$  for all  $j$ . Our stopping criterion is  $\text{ToIFun} = 10^{-12}$ .

In this setting, similarly to the EDM example in Section V-B, for each value of  $n$ , we obtain two distinct and clearly-separated clusters of results: one cluster consists of a significant number of trials that did not decrease the RMFE below  $\approx 10^{-2}$ , which is a relatively large value. The other cluster consists of the remaining trials, which managed to decrease the error to  $\approx 10^{-4}$  or less. For the trials in the latter cluster, using a smaller  $\text{ToIFun}$  can further decrease the RMFE by several orders of magnitude, whereas for trials in the former cluster, their error remains on the same scale. For each algorithm, we define as “success” any trial with a sufficiently small RMFE to place it in the second cluster.

We set  $D_{\text{FSVP}} = 14$  for all  $n$  for acceleration. We did not observe, in our preliminary tests, any particular influence of the value of  $D_{\text{FSVP}}$  on the rate of success. However, for  $n = 2, 3, 4$ , we set  $D_{\text{CGIHT}} = 14, 110, 9$ , respectively, as we observed that these values allow to increase the rate of successful factorizations significantly. For  $n = 5, 6, 7$  we did not observe any influence of  $D_{\text{CGIHT}}$  on the rate of success and thus for these three matrices, we set  $D_{\text{CGIHT}} = 1$ . Since CGIHT with  $D_{\text{CGIHT}} = 1$  is equivalent to NIHT, we do not show CGIHT in the plots for  $\mathbf{M}_5, \mathbf{M}_6$ , and  $\mathbf{M}_7$ . Note that here,  $D_{\text{CGIHT}}$  is not used in its original role as an acceleration parameter but to control the rate of success.

Fig. 2 exemplifies the results for  $\mathbf{M}_4 \in \mathbb{R}^{16 \times 16}$  and  $\mathbf{M}_7 \in \mathbb{R}^{128 \times 128}$ . Fig. 2a and 2c show the histogram (with 10 bins) of the final error. Fig. 2b and 2d exemplify the trajectory of each algorithm in one trial, where here the stopping criterion is  $\text{ToIFun} = 10^{-13}$ . Indeed, for  $\mathbf{M}_7$ , NIHT is the only algorithm that succeeded in properly decreasing the objective function, in 65 of 100 MC trials, whereas CGIHT dominates for  $\mathbf{M}_4$ , as shown in Table III.

Table III summarizes the number of successful factorizations per algorithm for each value of  $n$ . Table III shows that the ability of each algorithm to factorize  $\mathbf{M}_n$  varies greatly with  $n$ . While all methods factorize  $\mathbf{M}_2 \in \mathbb{R}^{4 \times 4}$  with 38%–

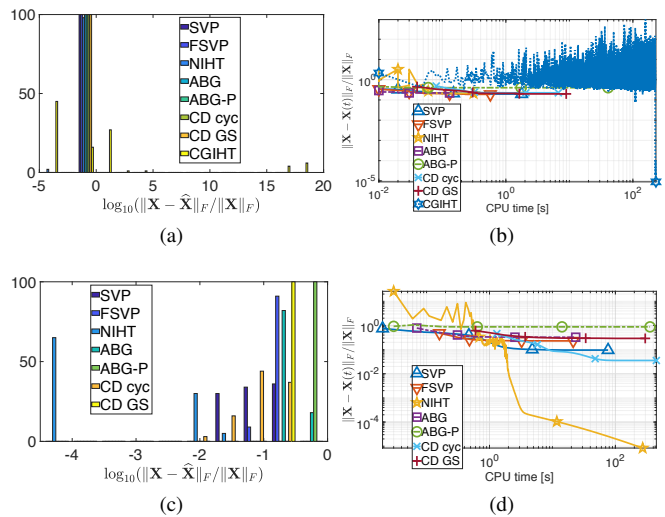


Fig. 2. PSDMF of  $\mathbf{M}_4$  (top) and  $\mathbf{M}_7$  (bottom). (a) and (c) Histogram of final error in 100 MC trials, (b) and (d) error evolution of one trial

96% success rate, none of ABG’s two variants succeed in factorizing  $\mathbf{M}_3 \in \mathbb{R}^{8 \times 8}$ , and the remaining algorithms have only 1% success rate—except for CGIHT with 55% success. At  $n = 4$ , our proposed NIHT-based PSDMF algorithm has modest success of 2% whereas its CGIHT variant succeeds in 45% of the attempts. For  $n = 5, 6, 7$ , NIHT is the only one to factorize  $\mathbf{M}_n$ , with success rates rising from 30% in  $n = 5$  to 65% with  $n = 7$ . While the drop in the general success rate as  $n$  increases may be explained by the non-convex nature of this problem and the smaller number of free variables versus constraints (for  $n = 2$ ,  $N_{\text{data}} = 16 > N_{\text{model}} = 15$ , and the ratio  $N_{\text{model}}/N_{\text{data}}$  only decreases with  $n$ ), the remarkable success rate of NIHT and CGIHT necessitates another set of arguments. These results resemble those in Section V-B for linear EDM with random distances. A possible explanation may arise from Fig. 2b, which shows that the error trajectory of NIHT starts very irregular and non-monotonous, until it drops sharply (in the log-log scale). CGIHT inherits the erratic behaviour from NIHT, and its trajectory seems to become more erratic as  $D_{\text{CGIHT}}$  increases. All other algorithms exhibit a monotonously decreasing trajectory that reaches a plateau. As mentioned in Sections IV-C and IV-D, NIHT and CGIHT are the only methods among the ones we compare that do not have a guarantee for monotonous decrease of the objective function. It is possible that this property allows them to escape certain local stationary points that the other methods get trapped in.

2) *Slack matrix of a regular  $n$ -gon:* We consider a slack matrix of a regular  $n$ -gon. An  $n$ -gon is a polygon with  $n$  sides. Regular slack matrices, denoted  $S_n$ , are determined up to scaling and transposition, and have size  $n \times n$ . The usual matrix rank of  $S_n$  is 3 for all  $n$ . For most  $n$ -gons, neither the psd rank nor the values of the inner ranks are known.

In this experiment, we factorize  $S_{32} \in \mathbb{R}^{32 \times 32}$ , visualized in Fig. 5a. The psd rank of  $S_{32}$  is not known; we shall use  $K = 1 + \lceil \log_2(n) \rceil = 6$  based on [3, Conjecture 1]. There is no conjecture about the values of the inner ranks,

TABLE III

NUMBER OF SUCCESSFUL FACTORIZATIONS OF  $M_n$  IN 100 MC TRIALS. For  $n = 5, 6, 7$ , THE BEST RESULTS FOR CGIHT WERE OBTAINED WHEN IT WAS EQUIVALENT TO NIHT.

$n$	SVP	FSVP	NIHT	ABG	ABG-P	CD cyc	CD GS	CGIHT
2	38	43	45	43	51	63	75	96
3	1	1	1	0	0	1	0	55
4	0	0	2	0	0	0	0	45
5	0	0	30	0	0	0	0	NIHT
6	0	0	61	0	0	0	0	NIHT
7	0	0	65	0	0	0	0	NIHT

but due to the presence of zeros in  $S_{32}$ , it is likely that  $R_{A_i} + R_{B_j} \leq K$  for any pair of  $(i, j)$ . We set  $R_A = 3$  and  $R_B = 3$ . In this case,  $N_{\text{data}} = 1024 > N_{\text{model}} = 924$ . This is a numerically challenging setting even in the absence of zeros. We set  $D_{\text{FSVP}} = 216$  and  $D_{\text{CGIHT}} = 21$ . We run 30 MC trials, where at each trial we use a new initialization. Our stopping criterion is  $\text{To1RMFE} = 0.0011$ . The CPU time allotted for each trial is 600[s]. Fig. 5c and 5e summarize our results. The boxplots in Fig. 5c show the overall number of iterations  $\ell \times D$  used by each algorithm to achieve the stopping criteria. Fig. 5e exemplifies the error evolution of each algorithm in one randomly-chosen run, now stopping after 20 minutes. We add that none of the trials of ABG-P achieved the designated RMFE in the allotted time; this is consistent with its behaviour in Fig. 5e.

We observe in Fig. 5c is that our choice of  $D_{\text{FSVP}}$  and  $D_{\text{CGIHT}}$  was good: the number of overall iterations of FSVP is the smallest on average among all methods, and the number of iterations of CGIHT is often smaller than its non-acceleration counterpart, NIHT. However, as we see in Fig. 5e, after a sufficiently large number of iterations, all methods eventually reach a point in which the decrease of the objective function is very slow. One possible explanation to our results is that the inner ranks were not chosen correctly: however, we observed the same trend also with other choices of inner ranks, e.g.,  $R_A = 4$  and  $R_B = 2$ , and larger. The difficulty to factorize  $S_n$  as  $n$  increases is in agreement with the results in [3].

For  $S_{12} \in \mathbb{R}^{12 \times 12}$ , the psd rank is conjectured to be  $K = 5$  [3]. The inner ranks are not known. The purpose of the following experiment is to provide evidence that the inner ranks can be equal to  $R_A = 3$  and  $R_B = 1$ . We tested 20 MC trials with stopping criteria CPU time 120[s] (24[s] for the CD algorithms running in c). We set  $D_{\text{FSVP}} = 10$  and  $D_{\text{CGIHT}} = 3$ . Fig. 3 shows our results. Fig. 3a shows the final error when the stopping criterion is achieved, in 20 MC trials. Fig. 3b shows the error evolution of one such trial, stopping at 600[s] CPU time. In this experiment, ABG-P did not manage to decrease the objective function properly in all trials, whereas this happened to ABG only occasionally. The other methods decreased the error reasonably. We note the fast decrease of the error of CGIHT in Fig. 3b, which means that the acceleration works properly in this case. These results provide evidence that this choice of inner ranks may indeed lead to exact factorization; however, experiments with longer run-time are required to determine this. We mention that also for  $S_{11}$  (not shown), we observed that a small reconstruction

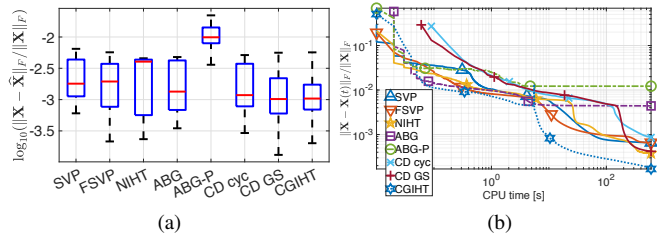


Fig. 3. PSDMF of (a)  $S_{12}$  with  $K = 5$ ,  $R_A = 3$ ,  $R_B = 1$ . (a) Final error in 20 MC trials. (b) Error evolution.

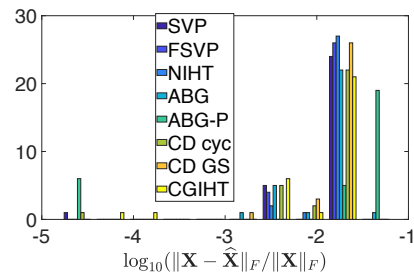


Fig. 4. PSDMF of  $S_8$  with  $K = 4$ ,  $R_A = 2$ ,  $R_B = 1$ . Histogram of final error in 30 MC trials.

error can be achieved with  $K = 5$ ,  $R_A = 3$  and  $R_B = 1$ .

Fig. 4 shows results for factorizing  $S_8 \in \mathbb{R}^{8 \times 8}$  with  $K = 4$ ,  $R_A = 2$ ,  $R_B = 1$ . These inner ranks yield an exact factorization [3]. We set  $D_{\text{FSVP}} = 3$  and  $D_{\text{CGIHT}} = 9$ . The stopping criteria are  $\text{To1Fun} = 10^{-14}$  and  $\text{To1RMFE} = 1.9 \cdot 10^{-5}$ . The histogram (with 10 bins) in Fig. 4 shows the error when the stopping criterion is achieved. The successful trials, with  $\text{RMFE} \leq 10^{-4}$ , are: 6 successful trials for ABG-P, 2 for CGIHT, and 1 for SVP and CD cyclic each, out of 30 MC trials.

3) *Subset Matrices*: Fig. 5b, 5d and 5f show our results for another (generalized) slack matrix of interest, the *subset matrix*  $P_n$ , described, e.g., in [3, Sec. 4.2], [4, Problem 2–3]. Here, we consider  $P_7 \in \mathbb{R}^{35 \times 35}$ , visualized in Fig. 5b. We factorize  $P_7$  with  $K = 6$ , which is conjectured to be its psd rank (e.g., [4, Sec. 8.1]). There are no conjectures about its inner ranks. However, based on the effect of zeros discussed in Section II-D, and the symmetry of  $P_n$ , we choose  $R_A = 3 = R_B$  such that  $R_A + R_B = K$ . With these ranks,  $N_{\text{model}} = 1014 < N_{\text{data}} = 1225$ , which is a challenging setting due to the smaller number of model parameters versus number of constraints and the non-convexity of the objective function. We used  $D_{\text{FSVP}} = 24$  and  $D_{\text{CGIHT}} = 3$ . Fig. 5d shows our results for MC = 30 trials with different initializations. Our stopping criteria are  $\text{To1RMFE} = 10^{-3}$  and 900[s] in CPU time. Fig. 5f shows the error evolution for these settings, where each algorithm was stopped after 15 minutes of CPU time.

In this experiment, only once the ABG-P algorithm managed to decrease the objective function properly. It is likely that ABG-P converges differently due to the different objective function, which changes the optimization landscape. In terms of the overall number of iterations in Fig. 5d, we observe small average values for the CD methods, NIHT and CGIHT, the latter with the smallest average. However, we should keep

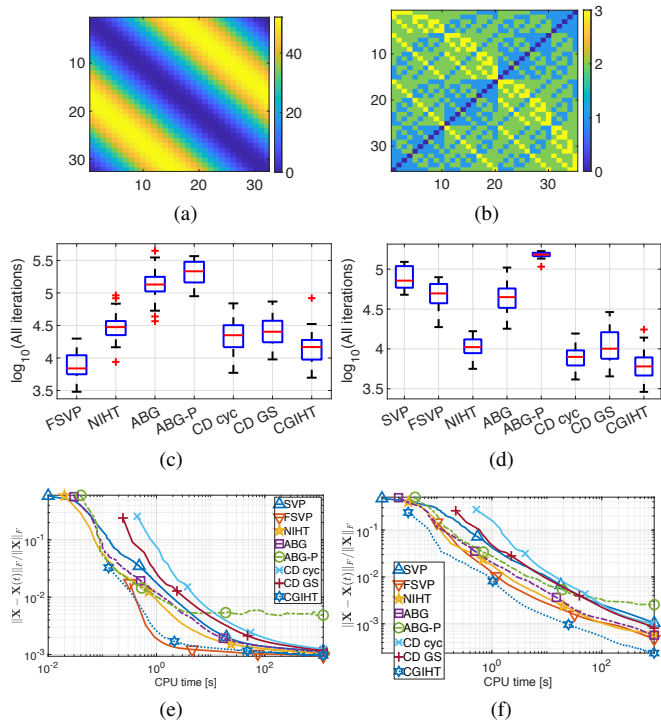


Fig. 5. PSDMF of (left)  $S_{32}$  and (right)  $P_7$ . (a) and (b) Visualization of  $S_{32}$  and  $P_7$ , respectively. (c) and (d) Number of overall iterations in 30 MC trials. (e) and (f) Error evolution in one trial.

in mind that the update steps of ABG are generally lighter, and that the computational complexity per update step of CD is generally larger. This (at least partly) explains why, consistently in all our numerical experiments, CD converged to the same target error with the highest CPU time, as shown also in Fig. 5e and 5f. The relations among the proposed projection-based methods are also in agreement with the theory: FSVP has on average fewer overall iterations than SVP, whereas NIHT has fewer iterations than both, which means the property of NIHT as a more numerically efficient method than SVP can be inherited by the PSDMF framework. We also see that with a good choice of  $D_{\text{CGIHT}}$ , CGIHT can indeed perform more efficiently than NIHT. The CPU time differences between NIHT and FSVP are generally not so pronounced. However, it is clear that CGIHT is faster. ABG is a relatively lightweight method in terms of computational complexity and number of operations per iteration. Hence, although its number of iterations can be higher than that of SVP, its speed of convergence is generally closer to that of NIHT and FSVP.

In Fig. 6, we show results for  $P_5 \in \mathbb{R}^{10 \times 10}$ . In this experiment, our stopping criterion is  $\text{To1RMFE} = 10^{-4}$ . We use  $D_{\text{FSVP}} = 9$  and  $D_{\text{CGIHT}} = 2$ . We set  $K = 4$  as conjectured by [3]. We choose  $R_A = 2 = R_B$  due to the symmetry of  $P_5$  and to satisfy  $R_A + R_B \leq K$ . In Fig. 6a, we observe that the number of overall iterations to achieve the RMFE is smallest, on average, for CGIHT. We remark that ABG-P failed to decrease its objective function properly in 5 out of the 30 MC trials. In this example, we observe that our choice of  $D_{\text{FSVP}}$  did not provide much improvement compared

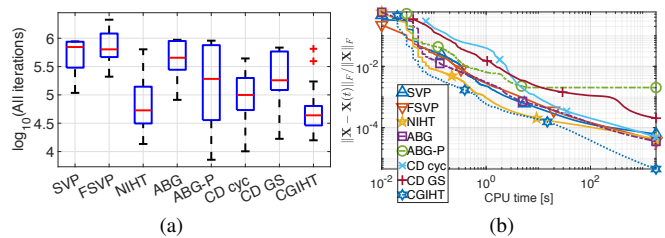


Fig. 6. PSDMF of  $P_5$ . (a) Number of overall iterations in 30 MC trials. (b) Error evolution.

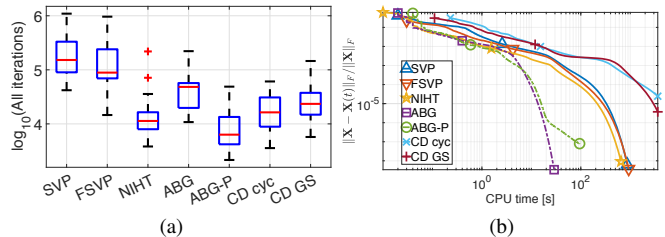


Fig. 7. PSDMF of a  $20 \times 20$  random matrix. (a) Number of overall iterations in 30 MC trials. (b) Error evolution in one run.

with its non-accelerated counterpart. These results provide supporting evidence that  $P_5$  may have an exact factorization with these ranks. For  $P_6$  (not shown) we achieved, in certain trials, and after a sufficient number of iterations, small error with  $K = 5$  and  $R_A = 2 = R_B$ . Combined with our results for  $P_7$  in Fig. 5f, we conjecture that for  $P_n$ ,  $K = n - 1$  and  $R_A = R_B = \lfloor K/2 \rfloor$  provide exact factorization.

#### D. Performance Comparison: Factorization of Dense Random Matrices

In this numerical experiment, our input is a  $20 \times 20$  matrix whose entries are drawn independently from the standard uniform distribution  $\mathcal{U}[0, 1]$ . When normalized such that  $\|\mathbf{X}\|_1 \triangleq \sum_{i,j} x_{ij} = 1$ ,  $\mathbf{X}$  can be interpreted as a probability mass function (PMF) of two discrete random variables, where one random variable takes  $I$  values and the other  $J$  values, such that each row or column of  $\mathbf{X}$  sums up to the marginal probability of each value given the other random variable. This setup was addressed in [13], in the context of expressive power of PSDMF (and its higher-order tensor network generalizations) in probabilistic modeling.

We fit  $\mathbf{X}$  to a PSDMF model with  $K = 7$  and inner ranks all equal to 2, as in [13, Sec. 6.1]. In this setting, there are more free model variables than constraints:  $N_{\text{data}} = 400 < N_{\text{model}} = 471$  for  $K = 7$ . We run 30 MC trials. In each trial, we generate a new random matrix  $\mathbf{X}$  and a new initialization. Our stopping criterion is  $\text{To1RMFE} = 10^{-4}$ . We use  $D_{\text{FSVP}} = 14$ . We did not find any value of  $D_{\text{CGIHT}}$  with which CGIHT outperformed NIHT, in this scenario. Fig. 7a shows the number of overall iterations  $\ell \times D$ . The error evolution as a function of CPU time is shown in Fig. 7b, for a randomly-chosen run, stopping at  $\text{To1Fun} = 10^{-20}$ .

In agreement with the results in [13], and our predictions from the model parameters, all trials achieved the target

RMFE. In terms of number of iterations, NIHT has the smallest number of iterations among the projection-based methods, whereas SVP has the largest, as expected from the theory. In agreement with the preceding experiments on geometric data and our computational complexity results, we observe that in general, CD methods take the longest CPU time to reach the same RMFE despite their relatively small number of iterations. Fig. 7 shows that in certain cases, ABG and ABG-P converges particularly fast in this setting. Fig. 7b shows a case in which the two ABG methods were the fastest to reach the designated error. Among the projection-based methods, NIHT is the fastest, in agreement with its smaller number of overall iterations. This random matrix setting differs from the geometric data in Section V-C by (i) not containing zero entries in  $\mathbf{X}$  and (ii) having an excess of free model parameters versus number of constraints, which might explain the difference in convergence behavior compared with those in Fig. 5. We mention that for  $K = 6$  and  $R_A = 2 = R_B$ , none of our methods managed to decrease the RMFE below a rather large value, in agreement with [13, Sec. 6.1], and although  $N_{\text{data}} = 400 < N_{\text{model}} = 404$ .

### E. Performance versus Level of Sparsity

In Section II-D, we showed how the presence of zeros in the input matrix  $\mathbf{X}$  imposes orthogonality constraints on the factor matrices. In this experiment, we numerically illustrate the consequences of these constraints. The following example demonstrates that even a small percentage of zeros can result in a difference between a model that fits the data well relatively easy and another in which it is difficult to find a fit that is approximately exact. Indeed, this is so even if there is an excess of free variables compared with the number of entries in  $\mathbf{X}$ . We generate a  $10 \times 10$  matrix  $\mathbf{X}$  whose  $(i, j)$ th entry is drawn independently from the standard uniform distribution, i.e.,  $x_{ij} \sim \mathcal{U}[0, 1]$ . This matrix  $\mathbf{X}$  is fixed throughout the experiment except for the number and locations of the zeros we add to it that vary. We determine the number of zeros in  $\mathbf{X}$  using the parameter  $0 \leq p \leq 1$ , where  $p$  is the ratio of zeros and  $p = 0$  means no zeros in  $\mathbf{X}$ . The number of zeros in  $\mathbf{X}$  is  $pIJ$ . We choose their  $pIJ$  locations randomly, independently and uniformly, with the following caveat: if a row or column of  $\mathbf{X}$  contains only zeros, we draw new locations for all the  $pIJ$  zeros. We repeat this procedure if necessary. The reason is that if the  $i$ th row of  $\mathbf{X}$  contains only zeros, it is impossible to obtain any constraints on  $\mathbf{A}_i$  and  $\mathbf{U}_i$ , and similarly for  $j$ , and thus they can be chosen completely arbitrarily—without needing any optimization. For each value of  $p$  and at each MC trial we draw new locations for the zeros. In this experiment,  $p = 0, \frac{1}{I}, \frac{2}{I}, \dots, \frac{8}{I}$ . We fit  $\mathbf{X}$  to a PSDMF model with  $K = 5$ ,  $R_A \triangleq R_{A_i} = 3$  for all  $i$ ,  $R_B \triangleq R_{B_j} = 1$  for all  $j$ . This setup has  $N_{\text{model}} = 145$  free variables in the model that we try to fit to the  $N_{\text{data}} = 100$  observations (see Section II-C). We also note that with probability 1,  $\text{rank}(\mathbf{X}) = 10$ , which is smaller than  $K(K+1)/2 = 15$ , which is the maximal rank of an arbitrary matrix modeled with psd rank  $K = 5$  (see Sections II-A and II-B). We remind that if  $\text{rank}(\mathbf{X}) > K(K+1)/2$ , it is impossible to find an exact

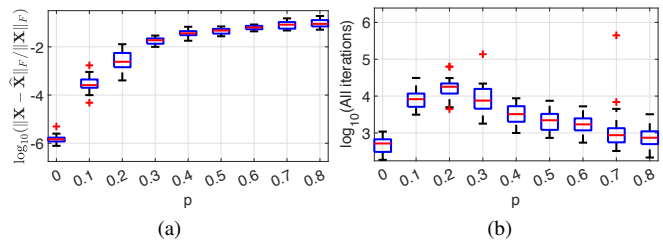


Fig. 8. Influence of the proportion of zeros  $p$  in a  $10 \times 10$  random matrix on the (a) model fit error, and (b) number of iterations.

PSDMF for  $\mathbf{X}$  with this value of  $K$ . These facts hint that fitting this model to  $\mathbf{X}$  should be easy—before we introduce the orthogonality constraints. We run 20 MC trials for each value of  $p$ , each with a new initialization. The stopping criterion is  $\text{ToIFun} = 10^{-13}$ .

Fig. 8 shows the results obtained using our NIHT-based PSDMF algorithm. Similar trends were observed with all other methods; we defer the remaining plots to Section VIII in the supplemental material. The error in Fig. 8 is calculated w.r.t. the value of  $\mathbf{X}$  at each iteration, including the zeros. Fig. 8a and 8b show the RMFE and number of iterations, respectively, when the stopping criterion is achieved. In the dense ( $p = 0$ ) case, the RMFE is  $\approx 10^{-6}$ , which means the algorithm achieved a very good model fit, in accordance with the surplus of free model variables versus number of observations, and “excess of rank”. This case also needed, in most MC trials, the smallest number of iterations to achieve the stopping criterion. Fig. 8a shows that even with as few as 10% of zeros, the algorithm does not manage to reduce the model fit error below a certain value, and this value increases with the proportion of zeros. This observation is in accordance with the theory. The number of iterations in Fig. 8b can be regarded as reflecting the “effort” the algorithm makes to fit the free model variables to the orthogonality constraints: with up to 20% of zeros, the algorithm manages to satisfy the orthogonality constraints to a certain extent, due to the excess of free model variables over the number of observations. But, as the number of zeros increases, this task fails faster because there are too many orthogonality constraints to satisfy.

## VI. DISCUSSION

The main contribution of this paper is a connection between the new problem of PSDMF and some canonical primitives in the recent signal processing literature such as phase retrieval and ARM. Based on this connection, we showed that families of PSDMF algorithms can be readily derived from their phase retrieval and ARM counterparts. Extensive numerical experiments compared and contrasted the performance of the proposed algorithms on benchmark datasets, showing that our proposed methods can outperform state-of-the-art algorithms [3] in terms of their convergence rates, ability to avoid local stationary points, and computational complexities, in various cases.

From a practical point of view, we presented a collection of algorithms for PSDMF. We showed that there is high variability among PSDMF problems such that the same algorithm

can behave differently on data of similar nature, for example, matrices generated by the same model. Our results show that there is no single algorithm that can achieve satisfying results on all data. Our fast method for prototyping new algorithms showed successful in that different algorithms that we designed based on different phase retrieval methods achieved remarkable success on different matrices. Therefore, we advise trying a number of algorithms from different families when addressing difficult PSDMF problems.

As a practical advice which algorithms to choose and which algorithms are more likely to be successful in PSDMF: Methods of particularly high success rate on difficult problems, in our experiments, are NIHT and ABG-P: these algorithms distinguish themselves from the others by not having guarantees for a monotone decrease of the objective function (NIHT) and a non-quadratic objective function (ABG-P). It is possible that the fact that these methods do not “play by the rules” allows them to overcome local stationary points that other methods get stuck in more often, in certain cases, while this same property also explains why in other cases (other matrices or other initializations for the same matrix) these methods perform less satisfactorily than their competitors. As for CGIHT, it has the disadvantage of needing a parameter  $D$  that has to be fine-tuned. However, as we demonstrated, the effort sometimes pays as there exist scenarios in which CGIHT can achieve high rates of success. Regarding FSVP, we have shown that there are cases in which the acceleration is significant. However, one has to make the balance between the effort in computing an optimal  $D$  and using a sub-optimal one, or another method.

These observations, and connection between PSDMF and other signal processing primitives (phase retrieval and ARM), have, however, their limitations, as PSDMF is a more challenging problem than phase retrieval and ARM, in certain respects, as detailed in Section III-C. In particular, PSDMF is highly non-convex, a challenge encountered in most, if not all, matrix factorization problems. This results in the problem of finding a good initialization for PSDMF algorithms. For phase retrieval and ARM, good initialization methods—such as spectral initialization—have been found based on knowing the true “dictionary”  $\mathcal{A}$ . The analogue of the dictionary is unknown for PSDMF problems. Indeed, we have discovered that methods that are fast and effective for phase retrieval and ARM are not always useful in the PSDMF framework, as shown in some of our numerical experiments.

Nevertheless, revealing the connection between PSDMF, phase retrieval and ARM opens the door to new algorithms and, more importantly, analyses. Like NMF, alternating methods are almost always used to solve PSDMF problems. Hence, we envision that some of the techniques used to analyze NMF may also be of utility.

Another potential impact of this work is in applications—especially those related to nonnegative and phaseless data. Our enrichment of the numerical tools available for PSDMF optimization motivates considering these tools in existing and new application. One natural extension of our work is applying our algorithms in quantum-based analysis of signal processing problems: for example, recommender systems [14] or proba-

bilistic models [13]. Some of these applications may require additional constraints, e.g., normalization as in POVMs. Another natural extension of our work is extending our algorithms from matrices to tensors, similarly to the algorithms for the tensor networks proposed by [13]. We leave these issues for future work.

## REFERENCES

- [1] J. Gouveia, P. A. Parrilo, and R. R. Thomas, “Lifts of convex sets and cone factorizations,” *Mathematics of Operations Research*, vol. 38, no. 2, pp. 248–264, May 2013.
- [2] S. Fiorini, S. Massar, S. Pokutta, H. R. Tiwary, and R. De Wolf, “Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds,” in *Proc. STOC*, May 2012, pp. 95–106.
- [3] A. Vandaele, F. Glineur, and N. Gillis, “Algorithms for positive semidefinite factorization,” *Computational Optimization and Applications*, vol. 71, no. 1, pp. 193–219, Sep 2018.
- [4] H. Fawzi, J. Gouveia, P. A. Parrilo, R. Z. Robinson, and R. R. Thomas, “Positive semidefinite rank,” *Mathematical Programming*, vol. 153, no. 1, pp. 133–177, Oct 2015.
- [5] L. B. Thomas, “Rank factorization of nonnegative matrices (A. Berman),” *SIAM Rev.*, vol. 16, no. 3, pp. 393–394, 1974.
- [6] J. E. Cohen and U. G. Rothblum, “Nonnegative ranks, decompositions, and factorizations of nonnegative matrices,” *Linear Algebra and its Applications*, vol. 190, pp. 149–168, September 1993.
- [7] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, no. 2, pp. 111–126, Jun. 1994.
- [8] D. Donoho and V. Stodden, “When does non-negative matrix factorization give a correct decomposition into parts?” in *Proc. NeurIPS*, 2004, pp. 1141–1148.
- [9] M. Yannakakis, “Expressing combinatorial optimization problems by linear programs,” *Journal of Computer and System Sciences*, vol. 43, no. 3, pp. 441–466, Dec. 1991.
- [10] V. Kaibel, “Extended formulations in combinatorial optimization,” *arXiv:1104.1023 [math.CO]*, Apr. 2011.
- [11] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, “Quantum SDP-Solvers: Better upper and lower bounds,” *Quantum*, vol. 4, p. 230, Feb. 2020.
- [12] R. Jain, Y. Shi, Z. Wei, and S. Zhang, “Efficient protocols for generating bipartite classical distributions and quantum states,” *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5171–5178, Aug 2013.
- [13] I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and I. Cirac, “Expressive power of tensor-network factorizations for probabilistic modeling,” in *Proc. NeurIPS*. Curran Associates, Inc., 2019, pp. 1496–1508.
- [14] C. J. Stark, “Recommender systems inspired by the structure of quantum theory,” *arXiv:1601.06035 [cs.LG]*, 2016.
- [15] M. Fazel, “Matrix rank minimization with applications,” Ph.D. dissertation, Stanford University, Stanford, CA, USA, Mar. 2002.
- [16] R. Meka, P. Jain, C. Caramanis, and I. S. Dhillon, “Rank minimization via online learning,” in *Proc. ICML*, ser. ICML ’08. New York, NY, USA: ACM, 2008, pp. 656–663.
- [17] B. Recht, M. Fazel, and P. Parrilo, “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization,” *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.
- [18] Q. Zheng and J. Lafferty, “A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements,” in *Proc. NeurIPS*, 2015, pp. 109–117.
- [19] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht, “Low-rank solutions of linear matrix equations via Procrustes flow,” in *Proc. ICML*, ser. Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, Jun. 2016, pp. 964–973.
- [20] P. Jain, R. Meka, and I. S. Dhillon, “Guaranteed rank minimization via singular value projection,” in *Proc. NeurIPS*, 2010, pp. 937–945.
- [21] D. Goldfarb and S. Ma, “Convergence of fixed-point continuation algorithms for matrix rank minimization,” *Foundations of Computational Mathematics*, vol. 11, no. 2, pp. 183–210, 2011.
- [22] J. Tanner and K. Wei, “Normalized iterative hard thresholding for matrix completion,” *SIAM J. Sci. Comput.*, vol. 35, no. 5, pp. S104–S125, 2013.
- [23] E. J. Candès, T. Strohmer, and V. Voroninski, “Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming,” *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1241–1274, 2013.



- [24] E. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski, "Phase retrieval via matrix completion," *SIAM Rev.*, vol. 57, no. 2, pp. 225–251, 2015.
- [25] E. J. Candès, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1985–2007, April 2015.
- [26] T. Qiu, P. Babu, and D. P. Palomar, "PRIME: Phase retrieval via majorization-minimization," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5174–5186, Oct 2016.
- [27] Y. Chen and E. Candès, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," in *Proc. NeurIPS*. Curran Associates, Inc., 2015, pp. 739–747.
- [28] R. Chandra, Z. Zhong, J. Hontz, V. McCulloch, C. Studer, and T. Goldstein, "PhasePack: A phase retrieval library," in *Proc. ACSSC*, Pacific Grove, CA, USA, Oct 2017, pp. 1617–1621.
- [29] A. Basu, M. Dinitz, and X. Li, "Computing approximate PSD factorizations," in *Proc. APPROX/RANDOM*, vol. 60, Dagstuhl, Germany, Sep. 2016, pp. 2:1–2:12.
- [30] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.
- [31] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate  $o(\frac{1}{k^2})$ ," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983.
- [32] T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 298–309, Apr. 2010.
- [33] J. D. Blanchard, J. Tanner, and K. Wei, "CGIHT: conjugate gradient iterative hard thresholding for compressed sensing and matrix completion," *Information and Inference: A Journal of the IMA*, vol. 4, no. 4, pp. 289–327, Dec 2015.
- [34] D. Lahat and C. Févotte, "Positive semidefinite matrix factorization: A link to phase retrieval and a block gradient algorithm," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.
- [35] D. Lahat and C. Févotte, "Positive semidefinite matrix factorization based on truncated Wirtinger flow," in *Proc. EUSIPCO*, Amsterdam, Netherlands, Jan. 2021.
- [36] Y. Chen and E. J. Candès, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," *Communications on Pure and Applied Mathematics*, vol. 70, no. 5, pp. 822–883, 2017.
- [37] J. Gouveia, P. A. Parrilo, and R. R. Thomas, "Approximate cone factorizations and lifts of polytopes," *Mathematical Programming*, vol. 151, no. 2, pp. 613–637, Jul. 2015.
- [38] S. Burer and R. D. C. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, Feb 2003.
- [39] S. Gröbbling, D. de Laat, and M. Laurent, "Lower bounds on matrix factorization ranks via noncommutative polynomial optimization," *Foundations of Computational Mathematics*, Jan 2019.
- [40] Y. Shitov, "Euclidean distance matrices and separations in communication complexity theory," *Discrete & Computational Geometry*, vol. 61, no. 3, pp. 653–660, Apr 2019.
- [41] P. Anttila, P. Paatero, U. Tapper, and O. Järvinen, "Source identification of bulk wet deposition in finland by positive matrix factorization," *Atmospheric Environment*, vol. 29, no. 14, pp. 1705–1718, 1995.
- [42] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.
- [43] T. Lee and D. O. Theis, "Support-based lower bounds for the positive semidefinite rank of a nonnegative matrix," *arXiv:1203.3961 [math.CO]*, 2013.
- [44] M. Fazel, E. Candès, B. Recht, and P. Parrilo, "Compressed sensing and robust recovery of low rank matrices," in *Proc. ACSSC*, Pacific Grove, CA, USA, Oct. 2008, pp. 1043–1047.
- [45] Z. Xue, X. Yuan, J. Ma, and Y. Ma, "TARM: A turbo-type algorithm for affine rank minimization," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5730–5745, Nov 2019.
- [46] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1214–1236, 2013.
- [47] K. Wei, J.-F. Cai, T. F. Chan, and S. Leung, "Guarantees of Riemannian optimization for low rank matrix recovery," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 3, pp. 1198–1222, 2016.
- [48] A. Vandaele, F. Glineur, and N. Gillis. Code for PSD factorization. [Online]. Available: <https://sites.google.com/site/exactnmf/psd-factorization>
- [49] L. B. Beasley and T. J. Laffey, "Real rank versus nonnegative rank," *Linear Algebra and its Applications*, vol. 431, no. 12, pp. 2330–2335, Dec. 2009, special Issue in honor of Shmuel Friedland.

Supplemental Material for “Positive Semidefinite Matrix Factorization: A Connection with Phase Retrieval and Affine Rank Minimization”, Dana Lahat, Yanbin Lang, Vincent Y. F. Tan, Cédric Févotte.

## VII. PSDMF AS A SUM OF RANK-1 TERMS

In this section, we demonstrate how PSDMF can be written as a sum of rank-1 terms.

### A. PSDMF As A Sum of Rank-1 Terms: The Positive Semidefinite Case

Next, we rearrange the rows of  $\mathfrak{A}$  and  $\mathfrak{B}$  such that their first  $K$  rows contain the entries on the diagonals of the psd matrices, which are nonnegative numbers. To do so, let  $a_{kl}^{(i)}$  and  $b_{kl}^{(j)}$  denote the  $(k, l)$ th entry of  $\mathbf{A}_i$  and  $\mathbf{B}_j$ , resp., for  $k, l = 1, \dots, K$ , and collect the terms indexed by  $l = k$  in the vectors:

$$\mathbf{a}_{kk} = \begin{bmatrix} a_{kk}^{(1)} & \cdots & a_{kk}^{(I)} \end{bmatrix}^T, \quad \mathbf{b}_{kk} = \begin{bmatrix} b_{kk}^{(1)} & \cdots & b_{kk}^{(J)} \end{bmatrix}^T. \quad (26)$$

Let  $\mathfrak{A}_+ \in \mathbb{R}_+^{K \times I}$  and  $\mathfrak{B}_+ \in \mathbb{R}_+^{K \times J}$  be the nonnegative matrices whose  $k$ th rows are  $\mathbf{a}_{kk}^T$  and  $\mathbf{b}_{kk}^T$ , resp.. Similarly, for  $l < k$ , define the vectors

$$\mathbf{a}_{kl} \triangleq \sqrt{2} \begin{bmatrix} a_{kl}^{(1)} & \cdots & a_{kl}^{(I)} \end{bmatrix}^T, \quad \mathbf{b}_{kl} \triangleq \sqrt{2} \begin{bmatrix} b_{kl}^{(1)} & \cdots & b_{kl}^{(J)} \end{bmatrix}^T$$

and let  $\mathfrak{A}_\pm \in \mathbb{R}^{((K-1)K/2) \times I}$  and  $\mathfrak{B}_\pm \in \mathbb{R}^{((K-1)K/2) \times J}$  denote the matrices whose rows are  $\mathbf{a}_{kl}^T$  and  $\mathbf{b}_{kl}^T$ , respectively, for  $l < k$ . The subscript  $\pm$  reminds that these matrices generally contain both negative and positive values. However, they are structured in the sense that they must satisfy that  $\mathbf{A}_i$  and  $\mathbf{B}_j$  remain psd. Using the notations that we have just introduced, we can express  $\mathbf{X}$  as:

$$\mathbf{X} \cong \mathfrak{A}^T \mathfrak{B} = \begin{bmatrix} \mathfrak{A}_+^T & \mathfrak{A}_\pm^T \end{bmatrix} \begin{bmatrix} \mathfrak{B}_+ \\ \mathfrak{B}_\pm \end{bmatrix} \quad (27a)$$

$$= \mathfrak{A}_+ \mathfrak{B}_+^T + \mathfrak{A}_\pm \mathfrak{B}_\pm^T \quad (27b)$$

$$= \underbrace{\mathbf{a}_{11} \mathbf{b}_{11}^T + \cdots + \mathbf{a}_{KK} \mathbf{b}_{KK}^T}_{\text{sum of } K \text{ nonnegative rank-1 terms}} \quad (27c)$$

$$+ \underbrace{\mathbf{a}_{12} \mathbf{b}_{12}^T + \cdots + \mathbf{a}_{(K-1),K} \mathbf{b}_{(K-1),K}^T}_{\text{sum of } K(K-1)/2 \text{ rank-1 terms}} \quad (27d)$$

$$= \underbrace{\mathbf{X}^{\text{“NMF”}}}_{\text{nonnegative}} + \mathbf{X}^{\text{“structured”}}. \quad (27e)$$

Equation (27) demonstrates that a PSDMF of a matrix  $\mathbf{X}$  with psd rank  $K$  can always be expressed as a sum of  $K$  nonnegative rank-1 terms and up to  $K(K-1)/2$  rank-1 terms

that are allowed to take negative values. Thus, if a given matrix  $\mathbf{X}$  can be expressed using a PSDMF model with a certain value  $K$ , the usual matrix rank of  $\mathbf{X}$  is at most  $K(K+1)/2$ .

### B. PSDMF As A Sum of Rank-1 Terms: The Factor-Based Case

In analogy to the psd-based representation in (27), we can write  $\mathbf{X}$  as a sum of rank-1 terms using the factor-based representation. In order to have a simpler formulation, we temporarily assume that  $R_{A_i} = R_A$  for all  $i$  and  $R_{B_j} = R_B$  for all  $j$ . We obtain:

$$\mathbf{X} = \underbrace{\sum_{k=l} \alpha_{kk} \beta_{kk}^T}_{\text{sum of } K \text{ nonnegative rank-1 terms}} + \underbrace{\sum_{k=1}^K \sum_{l \neq k} \alpha_{kl} \beta_{kl}^T}_{\text{sum of } K(K-1)/2 \text{ rank-1 terms}} = \underbrace{\mathbf{X}^{\text{“NMF”}}}_{\text{nonnegative}} + \mathbf{X}^{\text{“structured”}} \quad (28)$$

where

$$\alpha_{kl} \triangleq \sum_{r=1}^{R_A} \mathbf{u}_{kr} * \mathbf{u}_{lr} \in \mathbb{R}^I \quad (29a)$$

$$\beta_{kl} \triangleq \sum_{s=1}^{R_B} \mathbf{v}_{ks} * \mathbf{v}_{ls} \in \mathbb{R}^J, \quad (29b)$$

\* denotes the Hadamard (elementwise) product, and

$$\mathbf{u}_{kr} = \begin{bmatrix} u_{kr}^{(1)} & \cdots & u_{kr}^{(i)} & \cdots & u_{kr}^{(I)} \end{bmatrix}^T \in \mathbb{R}^I \quad (30a)$$

$$\mathbf{v}_{ks} = \begin{bmatrix} v_{ks}^{(1)} & \cdots & v_{ks}^{(j)} & \cdots & v_{ks}^{(J)} \end{bmatrix}^T \in \mathbb{R}^J, \quad (30b)$$

where  $u_{kr}^{(i)}$ ,  $r = 1, \dots, R_A$ , and  $v_{ks}^{(j)}$ ,  $s = 1, \dots, R_B$ , denote the  $(k, r)$ th and  $(k, s)$ th entry of  $\mathbf{U}_i$  and  $\mathbf{V}_j$ , respectively. We note that  $\alpha_{kk}$  and  $\beta_{kk}$  consist only of nonnegative values.

Similarly to (27), the model in (28) shows that when  $\mathbf{X}$  can be expressed as a sum of  $K$  or more nonnegative rank-1 terms only, the factorization is equivalent to NMF, and there is no need for the PSDMF framework. Equations (27) and (28) provide further evidence that the usual matrix rank of  $\mathbf{X}$  is at most  $K(K+1)/2$ . In fact, except for very special cases (see, e.g., [4]), the usual matrix rank of  $\mathbf{X}$  with psd rank  $K$  is  $K(K+1)/2$ .

## VIII. PERFORMANCE VERSUS LEVEL OF SPARSITY—ADDITIONAL RESULTS

Fig. 9 summarizes the plots generated using the same model and optimization parameters as in Fig. 8 for all methods concerned in our numerical experiments.

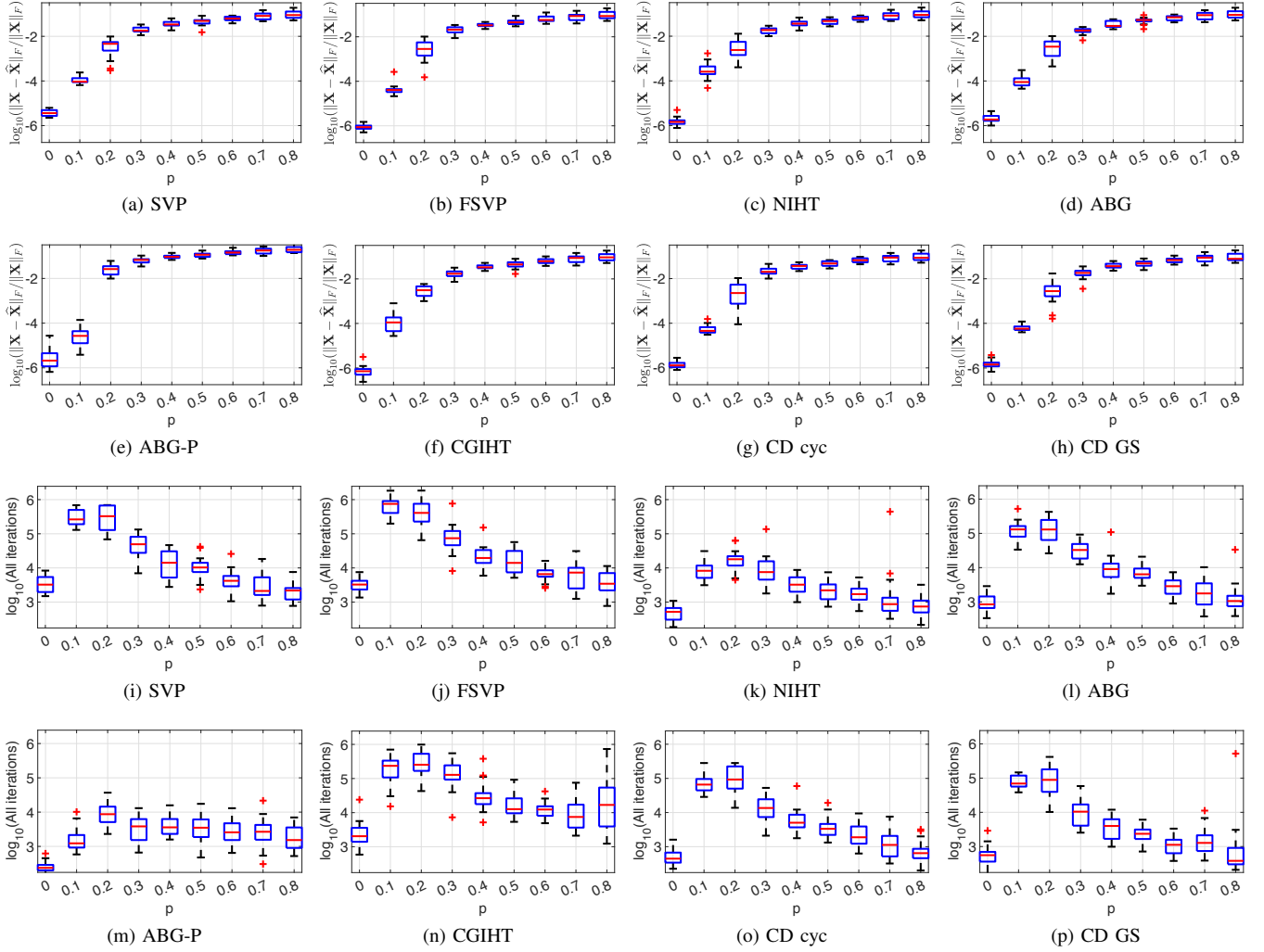


Fig. 9. Influence of the proportion of zeros  $p$  in a  $10 \times 10$  random matrix on the model fit error (Fig. 9a to 9h) and on the overall number of iterations (Fig. 9i to 9p). Factorized with  $K = 5$ ,  $R_A = 3$ ,  $R_B = 1$ . Stopping criterion:  $\text{ToIFun} = 10^{-13}$ .  $D_{\text{FSVP}} = 14$ ,  $D_{\text{CGIHT}} = 14$ . All plots have the same Y-axis for number of iterations, and similarly for the final RMFE. We observe the same trend for all methods: as soon as  $p > 0$ , the RMFE cannot be reduced below a certain value. However, for  $p = 0$ , further decreasing the stopping criterion  $\text{ToIFun}$  will yield smaller RMFE by several orders of magnitude. This matrix size and values of inner and outer ranks were chosen because they are the same as for  $S_{10}$ , a matrix that we also factorize in Section V.