Generalized Unitarily Invariant Gauge Regularization for Fast Low-Rank Matrix Recovery

Xixi Jia, Xiangchu Feng, Weiwei Wang and Lei Zhang *Fellow, IEEE*

***Abstract*—Spectral regularization is a widely used approach for low-rank matrix recovery (LRMR) by regularizing matrix singular values. Most of the existing LRMR solvers iteratively compute the singular values via applying singular value decomposition (SVD) on a dense matrix, which is computationally expensive and severely limits their applications to large-scale problems. To address this issue, we present a generalized unitarily invariant gauge (GUIG) function for LRMR. The proposed GUIG function does not act on the singular values, however, we show that it generalizes the well-known spectral functions, including the rank function, Schatten-*p* quasi norm and logsum of singular values. The proposed GUIG regularization model can be formulated as a bi-linear variational problem, which can be efficiently solved without computing SVD. Such a property makes it well suited for large-scale LRMR problems. We apply the proposed GUIG model to matrix completion and robust principal component analysis and prove the convergence of the algorithms. Experimental results demonstrate that the proposed GUIG method is not only more accurate but also much faster than the state-of-the-art algorithms, especially on large-scale problems.**

***Index Terms*—Low rank matrix recovery, Spectral regularization, Bi-linear matrix factorization, Robust principal component analysis.**

I. Introduction

Matrix rank minimization is an effective approach for finding a low-rank approximation of the given data matrix in myriad applications of machine learning and computer vision [1]–[7]. For example, in matrix completion (MC) [5], [8]–[10] the inherent low-rank matrix can be recovered with only partial observation data. In recommendation systems [11], [12], the missing data of customers can be estimated from the available data by assuming that the whole data matrix is low-rank. In robust principal component analysis (RPCA) [7], [13], the target matrix is assumed to be a low rank matrix corrupted by large but sparse noise. A variety of rank-minimization algorithms have been developed to address different low-rank matrix recovery (LRMR) problems in past decades [10], [14], [15].

The rank of a matrix is equal to the number of its nonzero singular values, and the general rank minimization problem is NP-hard [9]. Spectral functions, as tractable surrogates of the rank function, have been investigated by imposing sparsity regularizations on the singular values. Popular spectral functions include the nuclear norm [9], the weighted nuclear norm [2], the Schatten-p quasi norm [16], the logsum of singular values [17], [18] and the partial sum of singular values [3]. Mathematically, the matrix spectral regularization problem can be formulated as:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \mathcal{F}(\boldsymbol{X}) = f(\boldsymbol{X}) + \mathfrak{g}\circ\sigma(\boldsymbol{X}) \tag{1}$$

where f is a bounded smooth function with L_f Lipschitz continuous gradient in $\mathbb{R}^{m\times n}$. $\sigma(\boldsymbol{X}) : \mathbb{R}^{m\times n} \to \mathbb{R}^d$ extracts the singular values of the matrix $\boldsymbol{X}$. $\mathfrak{g} : \mathbb{R}^d \to \mathbb{R}$ is a sparse regularization function such as the l_1-norm, the $l_p(0 < p < 1)$ quasi norm and the logsum function. Symbol "$\circ$" means that the function $\mathfrak{g}$ operates on $\sigma(\boldsymbol{X})$. Usually, the function $\mathfrak{g}$ is separable such that $\mathfrak{g}\circ\sigma(\boldsymbol{X}) = \sum_{i=1}^{r} g(\sigma_i(\boldsymbol{X}))$, where $g : \mathbb{R} \to \mathbb{R}$ operates on $\sigma_i(\boldsymbol{X})$.

Among the existing spectral functions, the nuclear norm has proven to be the tightest convex envelop of the rank function [19], and nuclear norm minimization has been widely studied for LRMR and representation in recent years [9], [13], [20], [21]. However, as indicated in [16], [22], the nuclear norm tends to produce a biased low-rank solution since it penalizes the singular values uniformly. The non-convex Schatten-p quasi norm provides a more accurate rank approximation, and the Schatten-p quasi-norm minimization with a small p needs much less measurements than the nuclear norm for matrix completion [23]. The logsum of matrix singular values has been considered as a smooth approximation of the rank function [18]. Lu *et al.* [24] presented more non-convex spectral functions and demonstrated that the non-convex ones are generally superior to the convex nuclear norm.

Given the spectral functions, fast numerical solution to the spectral regularization problem are highly desired. Cai *et al.* [5] presented the well-known singular value thresholding algorithm for the nuclear norm minimization problem with *Frobenius*-norm loss. Based on the work of [5], Toh *et al.* [25] proposed an accelerated proximal gradient algorithm (APG) for nuclear norm minimization. Lin *et al.* [26] proposed to use the augmented lagrange multiplier method for nuclear norm minimization. For the general non-convex spectral regularization problem, Lu *et al.* [24] designed an iteratively reweighted nuclear norm (IRNN) algorithm. Lai *et al.* [27] proposed an iteratively reweighted least square (IRucLq) algorithm for Schatten-p quasi norm minimization. Mohan *et al.* [28] further employed the iterative reweighted algorithm for matrix rank minimization. Most existing spectral regularization solvers

This work was supported by the National Natural Science Foundation of China (Grants 69972036, 61772389, 61906144) and China Postdoctoral Science Foundation under Grant 2019M663639. Corresponding author: Xiangchu Feng (E-mail: xcfeng@mail.xidian.edu.cn).

X. Jia is with the School of Mathematics and Statistics, Xidian University, xi'an, 710126, China, and also with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: xxjia@xidian.edu.cn

X. Feng and W. Wang are with the School of Mathematics and Statistics, Xidian University, xi'an, 710126, China. E-mail: xcfeng@mail.xidian.edu.cn; vivianwang919@gmail.com

L. Zhang is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: cslzhang@comp.polyu.edu.hk

employ singular value decomposition (SVD) to compute the singular values in each iteration. The computational complexity of SVD is $O(mn\min(m,n))$ for a matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, which is rather time consuming and therefore hinders the application of spectral functions to large-scale problems. In addition, the memory requirement for storing the matrix $\boldsymbol{X}$ is $O(mn)$ which is unaffordable when the dimension of the matrix becomes large.

To reduce the computational complexity and memory request, the bi-linear matrix factorization has been successfully used for nuclear norm based LRMR [9], [29], [30], in which a matrix is factorized into two smaller factor matrices. More recently, Xu *et al.* [16] proved that the Schatten-p quasi norm can be represented by bi-linear matrix factorization. Shang *et al.* [22], [31]–[33] introduced a series of representative bi-linear matrix norms, such as: double nuclear norm, Frobenius/nuclear norm, to approximate the Schatten quasi-p norm for different p values, and these bi-linear matrix norms have been successfully used in solving the LRMR and RPCA problems. However, the works in [16], [22] still need to iteratively compute SVD of the factor matrices, which is costly especially for large scale problems.

In this paper, we propose a generalized unitarily invariant gauge (GUIG) function for the LRMR problem. The proposed GUIG function is partially inspired by the convex gauge function (which is also called atom norm) analyzed in [34], [35], while we extend it to the general non-convex case. Our proposed method scales well to large-scale problems. It does not directly act on the singular values, and can be easily solved without computing SVD. The main contributions of our work are summarized as follows.

First, we prove that the GUIG function is a new feasible spectral function. It generalizes the commonly used spectral functions such as the rank function, Schatten-p quasi norm and logsum of singular values.

Second, a flexible bi-linear factorization model is constructed for the GUIG function. Our model generalizes the bi-linear nuclear norm factorization models in [9], [29], [30] as special cases, and it is simpler than the Bi-Schatten-p model proposed in [16], [22], [31]–[33].

Third, we use GUIG function as a low-rank regularizer for MC and RPCA. The proposed models enjoy the merit of fast computation in an SVD-free manner. The per-iteration complexity is significantly reduced from $O(mn\min(m,n))$ to $O(mnd)$, where $d \ll \min(m,n)$. An alternating proximal algorithm with an adaptive dimension estimation method is developed for our model optimization, and the global convergence of our algorithm is proved.

Fourth, we propose a simple yet effective initialization strategy for the alternating proximal algorithm. Our initialization strategy not only speeds up the convergence of our algorithm, but also yields a good solution especially for the non-convex regularization problem.

We evaluate the performance of our method on MC and RPCA applications. Extensive experiments are performed on both synthetic and real data sets such as *MovieLens* and *Netflix*. Results show that the proposed method can be several orders faster than the existing spectral function models. It is also faster than the state-of-the-art LRMR algorithms while maintaining competitive accuracy.

Notations: In the whole paper, vectors are denoted by lowercase boldface letters such as $\boldsymbol{x}$, and matrices are denoted by uppercase boldface letters such as $\boldsymbol{X}$. For a matrix $\boldsymbol{X}$ in $\mathbb{R}^{m\times n}$, its nuclear norm is $\|\boldsymbol{X}\|_* = \sum_{i=1}^r \sigma_i(\boldsymbol{X})$ and Schatten-p quasi norm is $\|\boldsymbol{X}\|_{S_p} = \left(\sum_{i=1}^r \sigma_i^p(\boldsymbol{X})\right)^{1/p}$. We define a column structure-p function of $\boldsymbol{X}$ as $\|\boldsymbol{X}\|_{2,p}^p = \sum_{j=1}^n \|\boldsymbol{X}_{\cdot j}\|_2^p$, where $\|\boldsymbol{X}_{\cdot j}\|_2^p = \left(\sqrt{\sum_{i=1}^m \boldsymbol{X}_{ij}^2}\right)^p$. Similarly, we define the structure-log function as $\|\boldsymbol{X}\|_{2,\log} = \sum_{j=1}^d \log\left(\|\boldsymbol{X}_{\cdot j}\|_2\right)$. We denote by $\mathfrak{U}$ the unitary matrix space : $\mathfrak{U} = \{\boldsymbol{X}|\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}\}$ where $\boldsymbol{I}$ is the identity matrix.

II. Related Work

A. Gauge Function

Let $\mathcal{A}$ be a collection of atoms that is a compact subset of $\mathbb{R}^{m\times n}$, the gauge of $\mathcal{A}$ is defined as [34]:

$$\|\boldsymbol{X}\|_{\mathcal{A}} = \inf\left\{\sum_{\boldsymbol{a}\in\mathcal{A}} |c_{\boldsymbol{a}}| : \boldsymbol{X} = \sum_{\boldsymbol{a}\in\mathcal{A}} c_{\boldsymbol{a}}\boldsymbol{a}, \forall \boldsymbol{a}\in\mathcal{A}\right\} \tag{2}$$

where $\mathcal{A}$ is centrally symmetric w.r.t. the origin, i.e., $\boldsymbol{a}\in\mathcal{A}$ if and only if $-\boldsymbol{a}\in\mathcal{A}$. The gauge $\|\boldsymbol{X}\|_{\mathcal{A}}$ is a norm, called *atom norm* induced by $\mathcal{A}$. As an appealing instance, when $\mathcal{A}$ is the set of rank-one matrices of unit-Euclidean-norm, the gauge function is the well-known nuclear norm [34], [35], [36].

With the gauge function, fast optimization algorithms for the nuclear norm regularization problem can be readily designed, such as the rank-one matrix pursuit algorithms proposed in [37]. The algorithm include two main steps in each iteration. The first step computes the top singular vector pair from the residual to form the rank-one matrices $\boldsymbol{a}$, and the second step refines the weights $c_{\boldsymbol{a}}$. Yet, these greedy algorithms may suffer from the suboptimal solutions. Bach [34] showed that bi-linear factorization of the nuclear norm can be constructed from the gauge function. Alternating minimization or gradient descent method can be utilized to solve the optimization problem as reported in [9], [29], [30].

The gauge function is convex by its definition (the absolute sum of the combination coefficients) [34]. While for the non-convex case, e.g., sum of non-convex functions applied to the combination coefficients, the gauge function may not be a norm. To the best of our knowledge, little work on the non-convex gauge function has been reported. To move one step forward, in this work we propose a generalized unitarily invariant gauge function (GUIG) to study the non-convex gauge function and its correlation to the spectral function of a matrix.

B. Alternating Minimization of Bi-linear Model

Consider the following optimization problem

$$\min_{\boldsymbol{U},\boldsymbol{V}} F(\boldsymbol{U},\boldsymbol{V}) = f(\boldsymbol{U}\boldsymbol{V}^T) + \psi(\boldsymbol{U}) + \phi(\boldsymbol{V}) \tag{3}$$

where $\psi(\cdot)$ and $\phi(\cdot)$ are regularization functions. The bi-linear factorization models of the nuclear norm [9], [29], [30] and the Schatten-p quasi norm regularizers [16], [22] are special cases of the above model. Alternating-minimization strategies are popularly used to solve the problem (3), where $\boldsymbol{U}$ and $\boldsymbol{V}$ are updated alternatively by fixing one and updating another [10], [16], [22], [38], [39]. Jain *et al.* [10] optimized only the smooth function $f(\boldsymbol{U}\boldsymbol{V}^T)$ to solve the low-rank matrix completion and compressive sensing problems. Bolte *et al.* [39] proved that if the objective function in (3) satisfies the Kurdyka-Łojasiewicz property, then it can be solved by the proximal alternating linearized minimization (PALM) algorithm with global convergence. The Kurdyka-Łojasiewicz condition is satisfied by our proposed GUIG function.

Yin et al. [40] proposed an accelerated block prox-linear algorithm for problem (3), while most of these first order algorithms can only converge with sublinear convergence rate. It has been recently reported in [10], [38] that with an SVD based initialization, the low rank factorization model can be solved with linear convergence rate. In this work, we show that for our GUIG model, a simple initialization strategy will greatly improve the convergence speed of alternating minimization.

III. Generalized Unitarily Invariant Gauge (GUIG) Function

As shown in Eq. (2), the gauge function is defined as the l_1-norm of the combination coefficients. While many works have been done on how to set the constraint set $\mathcal{A}$ [34], [36], few efforts have been devoted to study the main function other than the l_1-norm. Here we propose a generalized formulation of the gauge function, which not only generalizes the previous convex gauge functions but also leads to new non-convex gauge functions.

A. The Proposed GUIG Function

For ease of our presentation, we first introduce some definitions and lemmas.

Definition 1. *[41] A function $\Phi : \mathbb{R}^{m\times n} \rightarrow [-\infty, +\infty]$ is unitarily invariant if $\Phi(\boldsymbol{V}\boldsymbol{X}\boldsymbol{U}) = \Phi(\boldsymbol{X})$ for all $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ where $\boldsymbol{U}, \boldsymbol{V} \in \mathfrak{U}$.*

Given any vector γ in $\mathbb{R}^q$, we denote by $\widehat{\gamma}$ the vector with components $|\gamma_i|$ arranged in non-ascending order.

Definition 2. *[41] A function $f : \mathbb{R}^q \rightarrow [-\infty, +\infty]$ is absolutely symmetric if $f(\gamma) = f(\widehat{\gamma})$ for any vector γ in $\mathbb{R}^q$.*

It can be easily verified that the widely used sparse biased vector norms such as the l_1-norm and l_p quasi norm $\|\boldsymbol{x}\|_p$ $(0 < p < 1)$ are all absolutely symmetric functions.

Lemma 1 (Proposition 2.2 [41])**.** *If the function $\Phi : \mathbb{R}^{m\times n} \rightarrow [-\infty, +\infty]$ is unitarily invariant, then there exists an absolutely symmetric function $\phi : \mathbb{R}^q \rightarrow [-\infty, +\infty]$ such that $\Phi(\boldsymbol{X}) = \phi \circ \sigma(\boldsymbol{X})$.*

We are now ready to give the definition of our generalized unitarily invariant gauge (GUIG) function:

Definition 3. *The GUIG function of a matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, denoted by $G_g(\cdot) : \mathbb{R}^{m\times n} \rightarrow \mathbb{R}$, is defined as:*

$$G_g(\boldsymbol{X}) = \inf\left\{\sum_{i=1}^{d} g(|\lambda_i|) : \boldsymbol{X} = \sum_{i=1}^{d} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \|\boldsymbol{u}_i\|_2 = \|\boldsymbol{v}_i\|_2 = 1\right\} \tag{4}$$

where d is set as rank $(\boldsymbol{X}) \leq d \leq min\{m, n\}$ and $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$.

$G_g(\boldsymbol{X})$ extends the l_1-norm on the coefficients (refer to Eq. (2)) to a more general function g. The rank one matrix decomposition $\boldsymbol{X} = \sum_{i=1}^d \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ is not unique. SVD leads to a decomposition which imposes orthogonal constraint on the factors $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$. Note that in Eq. (4) we do not enforce the factors $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ to be orthogonal. Interestingly, we find that the proposed GUIG function $G_g(\boldsymbol{X})$ is a spectral function, as shown in the following theorem.

Theorem 1. *For any given $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ with $rank(\boldsymbol{X}) \leq d$, there exists an absolutely symmetric function ϕ, such that:*

$$G_g(\boldsymbol{X}) = \phi \circ \sigma(\boldsymbol{X}) \tag{5}$$

Proof. It can be easily verified that $G_g(\boldsymbol{X})$ is a unitarily invariant function. According to Lemma 1, Eq. (5) holds. □

Theorem 1 guarantees that there exists an absolutely symmetric spectral function $\phi \circ \sigma(\boldsymbol{X})$ in correspondence to our proposed GUIG function G_g. However, the function is not explicitly expressed, and it remains unknown how the function relates to the absolutely symmetric function $\mathfrak{g}(\boldsymbol{\lambda}) = \sum_{i=1}^d g(|\lambda_i|)$ in G_g. It has been proved [34], [36] that when $g(|\lambda_i|) = |\lambda_i|$, there exists $G_g(\boldsymbol{X}) = \sum_{i=1}^r |\sigma_i(\boldsymbol{X})| = \|\boldsymbol{X}\|_*$, and when $g(|\lambda_i|) = |\lambda_i|^0$, $G_g(\boldsymbol{X}) = \sum_i |\sigma_i(\boldsymbol{X})|^0 = \text{rank}(\boldsymbol{X})$ [1], where r is the rank of matrix $\boldsymbol{X}$. One may naturally ask whether the equality $G_g(\boldsymbol{X}) = \mathfrak{g} \circ \sigma(\boldsymbol{X})$ holds for any function g. We give a simple example to show that the answer is negative.

Let $g(|x|) = |x|^2$ and given $\boldsymbol{X} = \begin{bmatrix} \frac{3}{4} & \frac{2+\sqrt{3}}{4} \\ \frac{2+\sqrt{3}}{4} & \frac{5}{4} \end{bmatrix}$ and the decomposition $\boldsymbol{X} = \sum_{i=1}^2 \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ with $\boldsymbol{u}_1 = \boldsymbol{v}_1 = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]^T$, $\boldsymbol{u}_2 = \boldsymbol{v}_2 = [\frac{1}{2}, \frac{\sqrt{3}}{2}]^T$ and $\lambda_1 = \lambda_2 = 1$, it can be easily verified that $G_g(\boldsymbol{X}) \leq 2 < \sum_{i=1}^d \sigma_i^2(\boldsymbol{X}) = 3.866$ and the equality does not hold. Then it is interesting and important to study when and under what conditions, the equality $G_g(\boldsymbol{X}) = \mathfrak{g} \circ \sigma(\boldsymbol{X})$ will hold. We have the following theorem to answer this question.

Theorem 2. *Given matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ with $rank(\boldsymbol{X}) \leq d$ and $\mathfrak{g} \circ \sigma(\boldsymbol{X}) = \sum_{i=1}^r g(\sigma_i(\boldsymbol{X}))$, the equality*

$$G_g(\boldsymbol{X}) = \mathfrak{g} \circ \sigma(\boldsymbol{X}) \tag{6}$$

holds if the function g satisfies:

1) *g is concave and monotonically ascending in $(0, +\infty)$;*
2) *the function $\vartheta(t) \equiv g(e^t)$ is convex.*

Proof. The proof can be found in Appendix A-A. □

[1]To avoid ambiguity, in this paper the zeroth-power of a nonzero number is 1, and the zeroth-power of 0 is 0.

Theorem 2 elegantly connects the explicit spectral function with the proposed GUIG function. It should be noted that the conditions in Theorem 2 can be easily satisfied by many functions. For example, it can be easily verified that the functions $g(|x|) = |x|$ and $g(|x_i|) = |x|^0$ satisfy these conditions. Besides, the widely used non-convex functions $|x|^p$ $(0 < p < 1)$ and $\log(|x|)$ also satisfy these conditions.

B. Bilinear Representation of the GUIG Function

Given the GUIG function G_g, flexible bi-linear factorization models can be readily constructed if the function g satisfies the condition specified in the following theorem.

Theorem 3. *Given the GUIG function $G_g(\boldsymbol{X})$, if there exist functions g_1 and g_2 such that $g(z) = \min\limits_{z=ab} g_1(a) + g_2(b)$, then $G_g(\boldsymbol{X})$ can be represented as:*

$$G_g(\boldsymbol{X}) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \sum\nolimits_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2) + \sum\nolimits_{i=1}^{d} g_2(\|\boldsymbol{V}_{\cdot i}\|_2) \quad (7)$$

where $\boldsymbol{U}_{\cdot i}$ and $\boldsymbol{V}_{\cdot i}$ are the ith column of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively.

Proof. The proof can be found in Appendix A-B. □

Let's show some concrete examples where the functions g_1 and g_2 can be easily constructed from the corresponding function g. The examples cover the widely used Schatten-p quasi norm [16], the rank function and the logsum function on singular values [18].

Example 1. *Let the function g in G_g be $g(|\cdot|) = |\cdot|^p$ with $p \in (0,1]$, we have*

$$G_{|\cdot|^p}(\boldsymbol{X}) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \frac{p}{p_1}\|\boldsymbol{U}\|_{2,p_1}^{p_1} + \frac{p}{p_2}\|\boldsymbol{V}\|_{2,p_2}^{p_2} \quad (8)$$

where $\frac{1}{p} = \frac{1}{p_1} + \frac{1}{p_2}$ and $0 < p_1, p_2 \le 2$.

In Example 1, the corresponding g_1 and g_2 to $g(z) = |z|^p$ are $g_1(a) = \frac{p}{p_1}|a|^{p_1}$ and $g_2(b) = \frac{p}{p_2}|b|^{p_2}$, respectively, and we have $g(z) = \min\limits_{z=ab} g_1(a) + g_2(b)$.

Compared with the works in [16] and [22], where the Schatten-p quasi norm is factorized as:

$$\|\boldsymbol{X}\|_{S_p}^p = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \frac{p}{p_1}\|\boldsymbol{U}\|_{S_{p_1}}^{p_1} + \frac{p}{p_2}\|\boldsymbol{V}\|_{S_{p_2}}^{p_2} \quad (9)$$

one can see that our bi-linear representation model in Eq. (8) is simpler than Eq. (9). In specific, $\|\boldsymbol{U}\|_{S_{p_1}}^{p_1} = \sum_{i=1}^{r} \sigma_i(\boldsymbol{U})^{p_1}$, which needs to compute the singular values of the matrix $\boldsymbol{U}$, and likewise for matrix $\boldsymbol{V}$. In contrast, $\|\boldsymbol{U}\|_{2,p_1}^{p_1} = \sum_{j=1}^{n} \sqrt[p_1]{\sum_{i=1}^{m} \boldsymbol{U}_{ij}^2}$, which can be easily calculated and is SVD free. Thus, solving the problem (8) can be computational easier than solving the problem (9).

Apart from the Schatten-p quasi norm, the rank function and the logsum function on matrix singular values can also be represented in a bi-linear factorization form.

Example 2. *Let g in G_g be $g(|x|) = |x|^0$, we have*

$$G_{|x|^0}(\boldsymbol{X}) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \frac{\|\boldsymbol{U}\|_{2,0} + \|\boldsymbol{V}\|_{2,0}}{2} \quad (10)$$

In Example 2, we have correspondingly $g_1(a) = \frac{1}{2}|a|^0$ and $g_2(b) = \frac{1}{2}|b|^0$.

Example 3. *Let g in G_g be $g(|\cdot|) = \log(|\cdot|)$, we have*

$$G_{\log}(\boldsymbol{X}) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \|\boldsymbol{U}\|_{2,\log} + \|\boldsymbol{V}\|_{2,\log} \quad (11)$$

Similarly, we have $g_1(a) = \log(|a|)$ and $g_2(b) = \log(|b|)$ for Example 3. Note that we are the first to present the bi-linear factorization model of the logsum function on singular values in Eq. (11).

According to Theorem 2, it can be seen that spectral function $\mathrm{g} \circ \sigma(\boldsymbol{x})$ can be equivalently formulated as a GUIG function under some easy conditions. This verifies that the proposed GUIG function is a good rank surrogate. Moreover, flexible bi-linear factorization models can be readily constructed from the GUIG function according to Theorem 3. The overwhelming superiority in memory requirement of $\boldsymbol{U} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times d}$ to $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ $(d \ll \min(m,n))$ significantly enlarges the applicability of GUIG to very large scale low rank matrix recovery (LRMR) problems.

C. GUIG Regularization

In the spectral function regularization problem (1), when the function g in $\mathrm{g}\circ\sigma(\boldsymbol{X})$ satisfies the condition in Theorem 2, we have $\mathrm{g} \circ \sigma(\boldsymbol{X}) = G_g(\boldsymbol{X})$, and thus the problem (1) becomes the following optimization problem

$$\min_{\boldsymbol{X}} F(\boldsymbol{X}) = f(\boldsymbol{X}) + G_g(\boldsymbol{X}) \quad (12)$$

Furthermore, if the function g satisfies the conditions given in Theorem 3, we have the bi-linear representation of $G_g(\boldsymbol{X})$ in Eq. (7). Therefore the problem (12) becomes

$$\min_{\boldsymbol{X}} F(\boldsymbol{X}) = f(\boldsymbol{X}) + \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \sum\nolimits_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2) + \sum\nolimits_{i=1}^{d} g_2(\|\boldsymbol{V}_{\cdot i}\|_2) \quad (13)$$

The problem (13) is an inner constraint optimization problem. We relax it to a bi-linear optimization problem:

$$\min_{\boldsymbol{U},\boldsymbol{V}} F(\boldsymbol{U},\boldsymbol{V}) = f\left(\boldsymbol{U}\boldsymbol{V}^T\right) + \sum\nolimits_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2) + \sum\nolimits_{i=1}^{d} g_2(\|\boldsymbol{V}_{\cdot i}\|_2) \quad (14)$$

It can be shown that the optimal value of objective function of (14) is equal to that of (1), and the solution of the two problems are equivalent. We have the following theorem.

Theorem 4. *Suppose that $\boldsymbol{X}^*$ is a solution of problem (1) in which the function g satisfies the conditions in Theorem 2 and Theorem 3. Let $\boldsymbol{U}^*, \boldsymbol{V}^*$ be a solution of the problem (14). Then we have*

$$\mathcal{F}(\boldsymbol{X}^*) = F(\boldsymbol{U}^*, \boldsymbol{V}^*) \quad (15)$$

Further more, we have $\overline{\boldsymbol{X}} = \boldsymbol{U}^*\boldsymbol{V}^{*T}$ is also a solution of (1) and there exists a decomposition $\boldsymbol{X}^* = \overline{\boldsymbol{U}}\overline{\boldsymbol{V}}^T$ such that $\overline{\boldsymbol{U}}, \overline{\boldsymbol{V}}$ is the solution of (14).

Proof. The proof can be found in Appendix A-C. □

Theorem 4 reveals that the spectral function regularization problem can be equivalently expressed as a corresponding bi-linear matrix factorization problem and they have the same optimal value in objective function. Instead of solving the problem (1), one can easily solve the problem (14). There are two main advantages of (14). First, it is optimized on the factors directly, achieving a structured decomposition of $\boldsymbol{X}$ in (1). Second, the number of variables to be optimized scales linearly w.r.t. $m+n$, ensuring its applicability to large scale problems.

IV. GUIG Regularized Fast Matrix Recovery

In this section, we apply the proposed model (14) to representative LRMR problems, including matrix completion (MC) [5], [9], [37] and robust principal component analysis (RPCA) [2], [3], [7], [13].

A. Matrix Completion

MC aims to complete a low rank matrix from its partial observation [5], [8], [21], [25]. One primary application area of MC is recommendation system, where the task is to estimate the attributes of many users using only a small number of available data. With (14), the problem of MC can be formulated as:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V}^T-\boldsymbol{M})\|_F^2+\sum\nolimits_{i=1}^{d} g_1(\|\boldsymbol{U}^*_{\cdot i}\|_2)+g_2(\|\boldsymbol{V}^*_{\cdot i}\|_2) \quad (16)$$

where $\mathcal{P}_\Omega(\boldsymbol{S})_{i,j}=\begin{cases}\boldsymbol{S}_{i,j} & \text{if } (i,j)\in\Omega\\ 0 & \text{otherwise}\end{cases}$.

1) Optimization: We use the accelerated block prox-linear (ABPL) algorithm [40] to solve the problem (16) with an adaptive dimension estimation technique. We will prove in Section IV-C that the algorithm converges globally to the critical point, which is stronger than the subsequence convergence of the existing methods such as IRucLp [27] and IRNN [24].

The variables $\boldsymbol{U}$ and $\boldsymbol{V}$ are alternatively updated by linearizing the smooth part $\frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V}^T-\boldsymbol{M})\|_F^2$ with an additional proximal term. Denote by $f(\boldsymbol{U},\boldsymbol{V})=\frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V}^T-\boldsymbol{M})\|_F^2$, we have

$$\begin{aligned}\boldsymbol{U}^{k+1} &= \arg\min_{\boldsymbol{U}} \sum\nolimits_{i=1}^d g_1(\|\boldsymbol{U}_{\cdot i}\|_2)+\\ &\left\langle \nabla f_{\boldsymbol{U}^k}(\boldsymbol{U}^k,\boldsymbol{V}^k),\boldsymbol{U}-\boldsymbol{U}^k\right\rangle+\frac{\tau_{f_u}^k}{2}\|\boldsymbol{U}-\boldsymbol{U}^k\|_F^2\end{aligned} \quad (17)$$

and

$$\begin{aligned}\boldsymbol{V}^{k+1} &= \arg\min_{\boldsymbol{V}} \sum\nolimits_{i=1}^d g_2(\|\boldsymbol{V}_{\cdot i}\|_2)+\\ &\left\langle \nabla f_{\boldsymbol{V}^k}(\boldsymbol{U}^{k+1},\boldsymbol{V}^k),\boldsymbol{V}-\boldsymbol{V}^k\right\rangle+\frac{\tau_{f_v}^k}{2}\|\boldsymbol{V}-\boldsymbol{V}^k\|_F^2\end{aligned} \quad (18)$$

The Lipschitz constants $\tau_{f_u^k}$ and $\tau_{f_v^k}$ are set as:

$$\tau_{f_u^k}=\max\left\{\|\boldsymbol{V}^T\boldsymbol{V}\|_2,\epsilon\right\},\tau_{f_v^k}=\max\left\{\|\boldsymbol{U}^T\boldsymbol{U}\|_2,\epsilon\right\} \quad (19)$$

To efficiently solve (17) and (18), we introduce a proximal function of g_1 for matrix $\boldsymbol{X}$:

$$\mathbf{Prox}_{g_1}(\boldsymbol{X})=\arg\min_{\boldsymbol{Z}}\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Z}\|_F^2+\sum\nolimits_{i=1}^d g_1(\|\boldsymbol{Z}_{\cdot i}\|_2) \quad (20)$$

Proposition 1. *The proximal function defined in Eq. (20) has a simple solution as*

$$\boldsymbol{Prox}_{g_1}(\boldsymbol{X})=\boldsymbol{X}\boldsymbol{W} \quad (21)$$

where $\boldsymbol{W}$ *is a diagonal matrix with diagonal elements* $\boldsymbol{W}_{ii}=\frac{\varpi_i}{\|\boldsymbol{X}_{\cdot i}\|_2}$ *and* $\varpi_i=\arg\min_\alpha \frac{1}{2}(\|\boldsymbol{X}_{\cdot i}\|_2-\alpha)^2+g_1(\alpha)$.

Algorithm 1 GUIG for matrix completion (16)

```
Input: Incomplete observation $M$, functions $g_1$ and $g_2$.
Output: The optimal factor matrices $U^*$ and $V^*$
 1: Initialize $U^{-1}=U^0$, $V^{-1}=V^0$, $k=0$;
 2: repeat
 3:         Set $k=k+1$ and compute $t_k$;
 4:         Compute $\tau^k_{f_u}$ by (19) and $w^k_u$ by (24);
 5:         Update $\widehat{U}^k$ by (23), update $U^{k+1}$ by solving (17);
 6:         Compute $\tau^k_{f_v}$ by (19) and $w^k_v$ by (24);
 7:         Update $\widehat{V}^k$ by (23), update $V^{k+1}$ by solving (18);
 8:         Find the valid index set $\mathcal{S}=\mathcal{I}\bigcap\mathcal{J}$, where
 9:           $\mathcal{I}=\{i|U^{k+1}_{\cdot i}\neq \mathbf{0}\}$ and $\mathcal{J}=\{j|V^{k+1}_{\cdot j}\neq \mathbf{0}\}$.
10:         Set $U^{k+1}=U^{k+1}_{\cdot\mathcal{S}}$, $V^{k+1}=V^{k+1}_{\cdot\mathcal{S}}$.
11:     if $\mathcal{F}(U^{k+1},V^{k+1})\geq\mathcal{F}(U^k,V^k)$
12:         Renew the updated variables by setting
13:         $\widehat{U}^k=U^k$ and $\widehat{V}^k=V^k$ ;
14:     end
15: until Converge.
```

Proof. The optimization problem in (20) can be separated columnwise as $\min_{\boldsymbol{z}} \frac{1}{2}\|\boldsymbol{X}_{\cdot i}-\boldsymbol{z}\|_2^2+g_1(\|\boldsymbol{z}\|_2)$, which can be further transformed to

$$\begin{aligned}&\min_{\boldsymbol{z}} \tfrac{1}{2}\|\boldsymbol{X}_{\cdot i}\|_2^2+\tfrac{1}{2}\|\boldsymbol{z}\|_2^2-\langle\boldsymbol{X}_{\cdot i},\boldsymbol{z}\rangle+g_1(\|\boldsymbol{z}\|_2)\\ &=\min_{\boldsymbol{z}} \frac{\|\boldsymbol{X}_{\cdot i}\|_2}{2}+\frac{\|\boldsymbol{z}\|_2^2}{2\|\boldsymbol{X}_{\cdot i}\|_2}-\left\langle\frac{\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2},\boldsymbol{z}\right\rangle+\frac{g_1(\|\boldsymbol{z}\|_2)}{\|\boldsymbol{X}_{\cdot i}\|_2}\\ &=\min_{\boldsymbol{z}} \frac{\|\boldsymbol{X}_{\cdot i}\|_2}{2}+\frac{\|\boldsymbol{z}\|_2^2}{2\|\boldsymbol{X}_{\cdot i}\|_2}-\|\boldsymbol{z}\|_2\left\langle\frac{\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2},\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}\right\rangle+\frac{g_1(\|\boldsymbol{z}\|_2)}{\|\boldsymbol{X}_{\cdot i}\|_2}\end{aligned} \quad (22)$$

Since $-1\leq\left\langle\frac{\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2},\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}\right\rangle\leq 1$, to minimize Eq. (22) it holds $\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}=\frac{\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2}$. As a result, the optimal $\boldsymbol{z}$ is equal to $\|\boldsymbol{z}\|_2\frac{\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2}$. Let $\alpha=\|\boldsymbol{z}\|_2$, the column wise minimization problem becomes $\min_\alpha \frac{1}{2}(\|\boldsymbol{X}_{\cdot i}\|_2-\alpha)^2+g_1(\alpha)$, then each column of the matrix $\boldsymbol{Z}$ is $\frac{\varpi_i\boldsymbol{X}_{\cdot i}}{\|\boldsymbol{X}_{\cdot i}\|_2}$, and thus the optimal solution of the problem (20) is $\boldsymbol{Z}=\boldsymbol{X}\boldsymbol{W}$. This completes the proof. □

Similar result holds for the function g_2 and $\mathbf{Prox}_{g_2}(\boldsymbol{X})$. Due to the sparse nature of the functions g_1 and g_2, $\mathbf{Prox}_{g_1}(\boldsymbol{X})$ and $\mathbf{Prox}_{g_2}(\boldsymbol{X})$ will have many zero columns. Consequently, many columns of the temporal variables $\boldsymbol{U}^k$ and $\boldsymbol{V}^k$ become zero. For the purpose of robust rank estimation and fast computation, we delete the resulted zero columns and resize the matrices $\boldsymbol{U}^k$ and $\boldsymbol{V}^k$ to a matched lower dimension. We call such a process adaptive dimension estimation (lines 8 and 9 in Algorithm 1).

In addition, the acceleration technique proposed in [40] is adopted:

$$\widehat{\boldsymbol{U}}^k=\boldsymbol{U}^k+w_u^k(\boldsymbol{U}^k-\boldsymbol{U}^{k-1}),\widehat{\boldsymbol{V}}^k=\boldsymbol{V}^k+w_v^k(\boldsymbol{V}^k-\boldsymbol{V}^{k-1}) \quad (23)$$

where w_u^k and w_v^k is defined as

$$w_u^k=\min\left\{\frac{t_k-1}{t_{k+1}},0.99\sqrt{\frac{\tau_{fu}^{k-1}}{\tau_{fu}^k}}\right\},w_v^k=\min\left\{\frac{t_k-1}{t_{k+1}},0.99\sqrt{\frac{\tau_{fv}^{k-1}}{\tau_{fv}^k}}\right\} \quad (24)$$

with $t_0=1$ and $t_k=\left(1+\sqrt{1+4t_{k-1}^2}\right)/2$.

Algorithm 2 Initialization using power method

```
Input: Observed matrix $M \in \mathbb{R}^{m\times n}$, the estimated rank $k$, random matrix $R \in \mathbb{R}^{n\times k}$ and the number of power iteration $J = 3$.
Output: The initialized variables $U_0$ and $V_0$.
1: $Y_1 = MR$, $k = 0$;
2: repeat
3:     Set $k = k + 1$;
4:     $Q_k = QR(Y_k)$; // QR decompositon
5:     $Y_{k+1} = M(M^T Q_k)$
6: until Converge.
7: $[U, S, V] = \text{svd}(Q^T M)$;
8: $U_0 = QUS^{\frac{1}{2}}$, $V_0 = VS^{\frac{1}{2}}$;
9: $U_0 = U_0(:, 1:k)$, $V_0 = V_0(:, 1:k)$.
```

We summarize our algorithm in Algorithm 1. Like MSS in [16], the backtracking continuation technique is adopted to find a proper local Lipshitz constant. Specifically, we initially underestimate τ_{fu}^k and τ_{fv}^k by multiplying a factor $\rho < 1$, then gradually increase ρ with the iteration until it reaches the value in Eq. (19). As indicated in [16], such a technique can improve the solution for non-convex optimization. Benefit from the adaptive dimension estimation technique, the dimension of the factor matrices decreases with the iteration until it reaches the rank of the optimal solution, saving much the computational time.

2) Initialization: It has been shown [10], [14], [38], [42] that SVD based initialization provides a good solution for the bi-linear low rank matrix factorization model (16). That is, we can set $U^0 = U\Sigma^{\frac{1}{2}}$ and $V^0 = V\Sigma^{\frac{1}{2}}$, where U and V are the left and right singular vector matrices of the matrix M, and Σ is the singular value matrix. In our model, the function g_1 and g_2 may not be equivalent. To balance the function g_1 and g_2 and generalize the initialization, we set $U^0 = U\Sigma_1$ and $V^0 = V\Sigma_2$ such that $\Sigma = \Sigma_1\Sigma_2$ and the following equation is satisfied

$$\sum_{i=1}^{r} g(\Sigma_{ii}) = \sum_{i=1}^{r} g_1(\Sigma_{1_{ii}}) + \sum_{i=1}^{r} g_2(\Sigma_{2_{ii}}).$$

Rather than using the exact SVD for initialization, we use the power method to initialize U and V, as summarized in Algorithm 2, which is computationally more efficient than SVD for large scale problems. It is proved in [42] that the elaborated initial point is close to the optimal solution, thus the algorithm converges in a few iterations. We show in Fig. 1 the convergence of our algorithm using random initialization and the proposed initialization. It can be seen that our algorithm converges very fast (nearly 20 iterations) to a good solution with the proposed initialization, while it converges much slower (about 70 iterations) using random initialization.

B. Robust Principal Component Analysis

RPCA targets on recovering a low rank matrix from the observation with sparse outliers. It has been widely used in applications such as foreground background separation, face recognition and latent semantic indexing [13]. Mathematically, RPCA aims to minimize the following energy function:

$$\min_{X,S} \frac{1}{2}\|Y - X - S\|_F^2 + \lambda_1\|X\|_* + \lambda_2\|S\|_1 \tag{25}$$

Algorithm 3 GUIG based algorithm for RPCA (26)

```
Input: Observed matrix $M$, functions $g_1$, $g_2$ and $g$.
Output: The optimal factor matrices $U^*$, $V^*$ and $S^*$
1: Initialize $U^{-1} = U^0$, $V^{-1} = V^0$, $k = 0$;
2: repeat
3:     Set $k = k + 1$ and compute $t_k$;
4:     Compute $\tau_{f_u}^k$ by (19) and $w_u^k$ by (24);
5:     Update $\widehat{U}^k$ by (23), update $U^{k+1}$ by solving (27);
6:     Compute $\tau_{f_v}^k$ by (19) and $w_v^k$ by (24);
7:     Update $\widehat{V}^k$ by (23), update $V^{k+1}$ by solving (28);
8:     Find the valid index set $\mathcal{S} = \mathcal{I} \bigcap \mathcal{J}$, where
9:       $\mathcal{I} = \{i|U_{\cdot i}^{k+1} \neq \mathbf{0}\}$ and $\mathcal{J} = \{j|V_{\cdot j}^{k+1} \neq \mathbf{0}\}$.
10:    Set $U^{k+1} = U_{\cdot\mathcal{S}}^{k+1}$, $V^{k+1} = V_{\cdot\mathcal{S}}^{k+1}$.
11:    Update $\widehat{S}^k$ by (30), update $S^k$ by solving (29).
12:   if Objective function does not decrease
13:      Renew the updated variables by setting
14:      $\widehat{U}^k = U^k$, $\widehat{V}^k = V^k$ and $\widehat{S}^k = S^k$ ;
15:   end
16: until Converge.
```

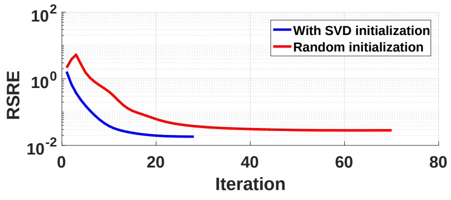


Fig. 1. Illustration of the proposed initialization. One can see that compared with random initialization, the proposed initialization enables the Algorithm 1 to converge very fast to a good solution (RSRE is defined in Section V-A).

By incorporating the proposed GUIG function into the low rank matrix X, the minimization problem (25) can be converted into

$$\begin{aligned}&\min_{U,V,S} \tfrac{1}{2}\|Y - UV^T - S\|_F^2\\ &+\lambda_1 \textstyle\sum_{i=1}^{d} g_1(\|U_{\cdot i}\|_2) + g_2(\|V_{\cdot i}\|_2) + \lambda_2\|S\|_g\end{aligned} \tag{26}$$

where $\|S\|_g = \sum_{i,j} g(|S_{ij}|)$ and g is the sparse biased function. The problem (26) is slightly different from the problem (14) in that it introduces an extra variable S, while the optimization of problem (26) is similar to that of (16). The variables U and V are alternatively updated by linearizing the smooth part $\frac{1}{2}\|Y - UV^T - S\|_F^2$ with an additional proximal term. Fixing

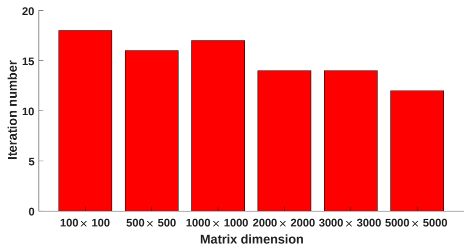
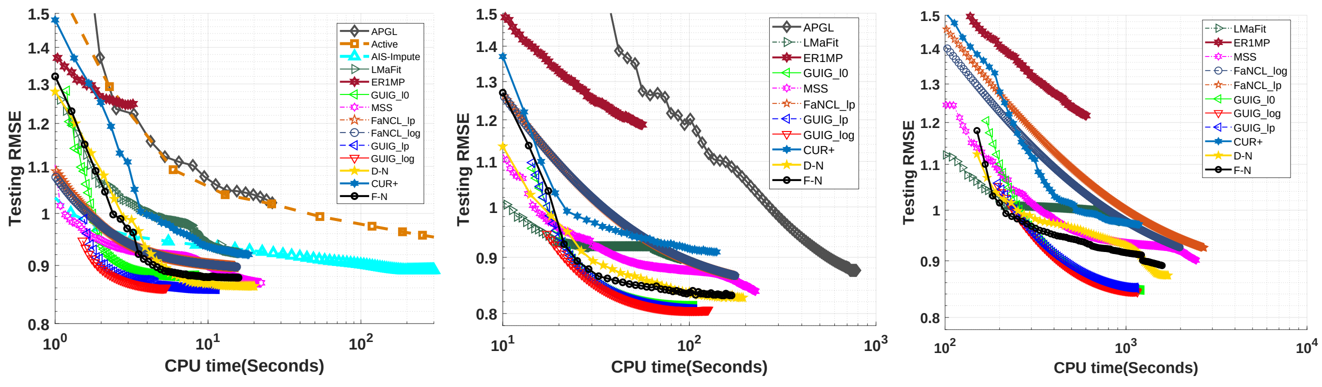


Fig. 2. Total iteration number for Algorithm 3 to converge. One can see that the proposed RPCA algorithm converges very fast from small scale case to large scale case.

TABLE I
MATRIX COMPLETION PERFORMANCE ON THE SYNTHETIC DATA. RSRE IS SCALED BY $\times 10^{-2}$ AND THE CPU TIME IS IN SECONDS. THE TESTED MATRICES ARE OF SIZE $m \times m$ CORRUPTED BY GAUSSIAN NOISE OF VARIANCE 0.1 ON THE OBSERVED ENTRIES.

		m = 500 observed: 5%, rank = 5			m = 2000 observed: 3.5%, rank = 10			m = 5000 observed: 2.5%, rank = 15			m = 10000 observed: 2%, rank = 20		
		RSRE	iter	time	RSRE	iter	time	RSRE	iter	time	RSRE	iter	time
Nuclear Norm	APG [25]	4.75 ± 0.02	100	2.51	2.07 ± 0.02	48	1.88	1.51 ± 0.02	48	4.97	1.19 ± 0.02	61	15.8
	AIS-Impute [43]	4.92 ± 0.02	145	0.369	2.58 ± 0.02	139	2.12	1.69 ± 0.02	218	26.45	1.38 ± 0.02	354	205
	Active [44]	4.83 ± 0.03	150	3.93	2.64 ± 0.03	120	20.7	1.77 ± 0.03	156	150	1.43 ± 0.03	166	238
Rank Model	LMaFit [12]	4.07 ± 0.01	92	0.142	2.05 ± 0.01	44	0.828	1.47 ± 0.01	35	3.76	1.14 ± 0.01	29	12.2
	CUR+ [46]	4.25 ± 0.03	487	0.616	2.12 ± 0.02	352	3.537	1.56 ± 0.05	238	17.6	1.27 ± 0.03	226	30.5
	ER1MP [37]	33.7 ± 0.05	15	0.043	28.6 ± 0.05	30	0.392	21.7 ± 0.05	45	1.94	20.7 ± 0.05	60	8.02
	GUIG_l_0	$\mathbf{3.98 \pm 0.01}$	62	**0.052**	$\mathbf{2.03 \pm 0.01}$	26	**0.332**	$\mathbf{1.47 \pm 0.01}$	23	**1.55**	$\mathbf{1.12 \pm 0.01}$	17	**4.69**
Schatten-p	IRNN [24]	4.37 ± 0.02	820	73.3	–	–	$> 10^3$	–	–	$> 10^4$	–	–	$> 10^5$
	IRucLq [27]	4.17 ± 0.01	652	48.1	–	–	$> 10^3$	–	–	$> 10^4$	–	–	$> 10^5$
	D-N [31]	4.15 ± 0.01	720	0.968	2.06 ± 0.01	308	3.22	1.46 ± 0.01	209	10.9	1.15 ± 0.01	190	28.1
	F-N [32]	4.16 ± 0.01	678	0.873	2.07 ± 0.01	286	3.05	1.47 ± 0.01	223	11.8	1.16 ± 0.01	182	27.1
	MSS [16]	4.19 ± 0.01	877	0.782	2.05 ± 0.01	257	3.09	1.46 ± 0.01	180	10.2	1.16 ± 0.01	169	26.4
	FaNCL [45]	4.02 ± 0.01	989	0.742	2.03 ± 0.01	888	7.59	1.44 ± 0.01	812	40.2	1.13 ± 0.01	702	160
	GUIG_l_p	$\mathbf{4.00 \pm 0.01}$	47	**0.038**	$\mathbf{2.03 \pm 0.01}$	31	**0.407**	$\mathbf{1.45 \pm 0.01}$	37	**2.58**	$\mathbf{1.14 \pm 0.01}$	28	**7.34**
Logdet	IRNN [24]	4.05 ± 0.01	750	37.7	–	–	$> 10^3$	–	–	$> 10^4$	–	–	$> 10^5$
	FaNCL [45]	4.00 ± 0.01	923	0.754	2.03 ± 0.01	858	7.52	1.45 ± 0.01	792	34.6	1.12 ± 0.01	684	155
	GUIG_log	$\mathbf{3.96 \pm 0.01}$	74	**0.068**	$\mathbf{2.02 \pm 0.01}$	25	**0.377**	$\mathbf{1.43 \pm 0.01}$	23	**1.65**	$\mathbf{1.12 \pm 0.01}$	21	**6.02**



(a) *MovieLens*-1M (b) *MovieLens*-10M (c) *Netflix*

Fig. 3. RMSE vs. CPU time for matrix completion on recommendation system data sets.

TABLE II
RECOMMENDATION DATA SETS USED IN OUR EXPERIMENTS.

		#users	#movies	#ratings
MovieLens	100K	943	1,682	100,000
	1M	6,040	3,449	999,714
	10M	480,189	17,770	100,480,507
Netflix		249,012	296,111	62,551,438

$\boldsymbol{U}$ and $\boldsymbol{V}$, the variable $\boldsymbol{S}$ can be easily solved. Specifically, we have

$$\boldsymbol{U}^{k+1} = \arg\min_{\boldsymbol{U}} \sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2) + \left\langle \nabla f_{\boldsymbol{U}^k}(\boldsymbol{U}^k, \boldsymbol{V}^k, \boldsymbol{S}^k), \boldsymbol{U} - \boldsymbol{U}^k \right\rangle + \frac{\tau_{fu}^k}{2}\|\boldsymbol{U} - \boldsymbol{U}^k\|_F^2 \tag{27}$$

and

$$\boldsymbol{V}^{k+1} = \arg\min_{\boldsymbol{V}} \sum_{i=1}^{d} g_2(\|\boldsymbol{V}_{\cdot i}\|_2) + \left\langle \nabla f_{\boldsymbol{V}^k}(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k, \boldsymbol{S}^k), \boldsymbol{V} - \boldsymbol{V}^k \right\rangle + \frac{\tau_{fv}^k}{2}\|\boldsymbol{V} - \boldsymbol{V}^k\|_F^2 \tag{28}$$

and $\boldsymbol{S}$ is obtained by a proximal function as

$$\boldsymbol{S}_{i,j}^{k+1} = \mathbf{Prox}_g\left(\boldsymbol{Y}_{i,j} - (\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1^T})_{i,j}\right) \tag{29}$$

where $f(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{S}) = \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{V}^T - \boldsymbol{S}\|_F^2$ and $\mathbf{Prox}_g(x) = \arg\min_y \frac{1}{2}(y-x)^2 + g(y)$ can be easily solved in our case. Our algorithm for RPCA is given in Algorithm 3. The same accelerate technique and adaptive dimension estimation method used in Algorithm 1 are utilized for the RPCA problem. And the variable $\boldsymbol{S}$ is accelerated as

$$\widehat{\boldsymbol{S}}^k = \boldsymbol{S}^k + w^k(\boldsymbol{S}^k - \boldsymbol{S}^{k-1}) \tag{30}$$

where $w^k = \frac{t_k - 1}{t_{k+1}}$ and t_k is defined as in Section IV-A1. The same initialization strategy as in MC (Algorithm 2) is used here and it makes our RPCA converge very fast, as shown in Fig. 2.

C. Convergence Analysis

Let's first analyze the convergence of Algorithm 1 for the MC problem (16). Since the function g such as the l_p quasi norm and the log function used in our algorithm satisfy the K-Ł property [39], the global convergence of the Algorithm 1 can be guaranteed, as shown in the following theorem.

Theorem 5. *The sequence* $\{(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ *generated by Algorithm 1 is a Cauchy sequence and converges to a critical point to the problem (16)* $[\boldsymbol{U}^*, \boldsymbol{V}^*]^T$.

Proof. The proof can be found in Appendix A-D. $\square$

Since Algorithm 3 is similar to Algorithm 1, they share the same convergence result.

TABLE III
MATRIX COMPLETION PERFORMANCE ON REAL DATA SETS. WE REPORT THE RMSE RESULTS AND CPU TIME (IN SECONDS).

		MovieLens-100K						*MovieLens*-1M					
		TR = 50%		TR = 70%		TR = 80%		TR = 50%		TR=70%		TR = 80%	
		RMSE	time	RMSE	time	RMSE	time	RMSE	time	RMSE	time	RMSE	time
Nuclear Norm	APG [25]	1.21	6.43	1.11	4.71	1.08	5.00	1.08	24.6	1.01	27.5	0.985	21.4
	AIS-Impute [43]	1.03	8.14	0.987	8.17	0.970	12.6	0.919	272	0.892	437	0.882	549
	Active [44]	1.06	40.9	0.992	68.5	0.978	82.3	0.932	3365	0.902	4814	0.892	5441
Rank Model	LMaFit [12]	0.984	0.827	0.958	1.16	0.955	1.11	0.924	16.5	0.915	15.8	0.910	17.1
	CUR+ [46]	1.05	0.985	0.982	1.23	0.976	1.20	0.938	19.1	0.920	18.8	0.922	22.1
	ER1MP [37]	1.46	0.322	1.41	0.305	1.37	0.413	1.26	2.81	1.25	4.07	1.14	4.52
	GUIG_l_0	**0.968**	**0.401**	**0.952**	**0.583**	**0.942**	**0.710**	**0.897**	**6.01**	**0.878**	**8.20**	**0.872**	**9.11**
Schatten-p	IRNN [24]	1.01	259	0.971	263	0.960	271	0.935	4328	0.928	4578	0.9013	4429
	IRucLq [27]	0.987	189	0.962	175	0.945	210	0.910	2786	0.892	2698	0.875	2801
	D-N [31]	0.965	1.62	0.947	2.13	0.940	2.28	0.880	17.6	0.863	19.5	0.850	22.0
	F-N [32]	0.969	1.52	0.955	1.86	0.943	2.07	0.894	15.3	0.878	16.1	0.862	18.9
	MSS [16]	0.968	1.87	0.951	2.37	0.948	2.49	0.887	19.4	0.868	22.2	0.855	23.54
	FaNCL [45]	1.00	1.54	0.965	1.61	0.955	1.64	0.908	10.8	0.899	12.7	0.893	12.8
	GUIG_l_p	**0.954**	**0.473**	**0.931**	**0.522**	**0.924**	**0.469**	**0.873**	**7.15**	**0.855**	**10.8**	**0.844**	**11.3**
Logsum	IRNN [24]	1.00	258	0.966	261	0.956	265	0.930	4007	0.921	4392	0.8974	4415
	FaNCL [45]	1.00	1.48	0.966	1.59	0.955	1.63	0.912	12.7	0.897	14.5	0.891	15.3
	GUIG_log	**0.956**	**0.438**	**0.933**	**0.495**	**0.924**	**0.664**	**0.875**	**2.69**	**0.858**	**4.60**	**0.849**	**8.07**
		MovieLens-10M						*Netflix*					
		TR = 50%		TR = 70%		TR = 80%		TR = 50%		TR=70%		TR = 80%	
		RMSE	time	RMSE	time	RMSE	time	RMSE	time	RMSE	time	RMSE	time
Nuclear Norm	APG [25]	1.00	361	0.877	786	0.874	783	–	–	–	–	–	–
	AIS-Impute [43]	0.853	26259	0.826	32926	0.817	43139	–	–	–	–	–	–
Rank Model	LMaFit [12]	0.899	107	0.892	120	0.889	117	0.962	1125	0.959	1045	0.955	1108
	CUR+ [46]	0.925	138	0.913	143	0.910	176	0.983	1258	0.971	1196	0.970	1209
	ER1MP [37]	1.25	51	1.18	55.4	1.15	61.9	1.29	550	1.217	602	1.20	645
	GUIG_l_0	**0.830**	**83.3**	**0.813**	**106**	**0.809**	**118**	**0.862**	**922**	**0.847**	**1010**	**0.843**	**1057**
Schatten-p	D-N [31]	0.834	206	0.827	198	0.826	219	0.883	1874	0.873	1722	0.869	1531
	F-N [32]	0.840	183	0.833	172	0.830	196	0.895	1583	0.880	1608	0.878	1385
	MSS [16]	0.848	198	0.838	226	0.835	241	0.908	2101	0.900	2443	0.894	2613
	FaNCL [45]	0.902	163	0.865	176	0.860	188	0.965	2389	0.925	1977	0.911	2898
	GUIG_l_p	**0.824**	**79.8**	**0.807**	**117**	**0.800**	**129**	**0.859**	**735**	**0.847**	**815**	**0.840**	**1012**
Logsum	FaNCL [45]	0.903	184	0.866	175	0.856	210	0.966	2357	9.25	2701	0.912	2870
	GUIG_log	**0.820**	**80.5**	**0.806**	**128**	**0.799**	**141**	**0.858**	**939**	**0.843**	**1003**	**0.837**	**1175**

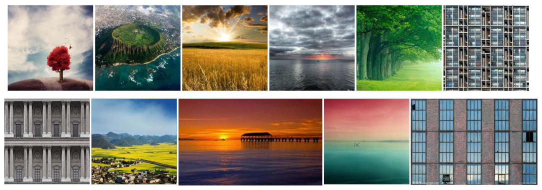


Fig. 4. Images used in the inpainting experiments.

D. Complexity Analysis

zui The computational cost of the existing algorithms for the spectral function regularization is dominated by the computation of SVD in each iteration. The complexity of SVD for a matrix of size $m \times n$ is $O(mn \min(m, n))$. In contrast, our algorithms for both MC and RPCA are SVD free, and their computational cost is mainly spent on matrix multiplication for updating $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{S}$ with complexity $O(mnd + mn)$. Given that $d \ll \min(m, n)$, our algorithms have much lower computational complexity than the existing algorithms. Furthermore, by utilizing the proposed initialization strategy, our algorithms converge very fast, greatly reducing the computational time. In addition, the memory requirement for a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ in the spectral function regularization problem is $O(mn)$, while the memory requirement of our proposed method is only $O(md + nd)$ since it only needs to store $\boldsymbol{U} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times d}$.

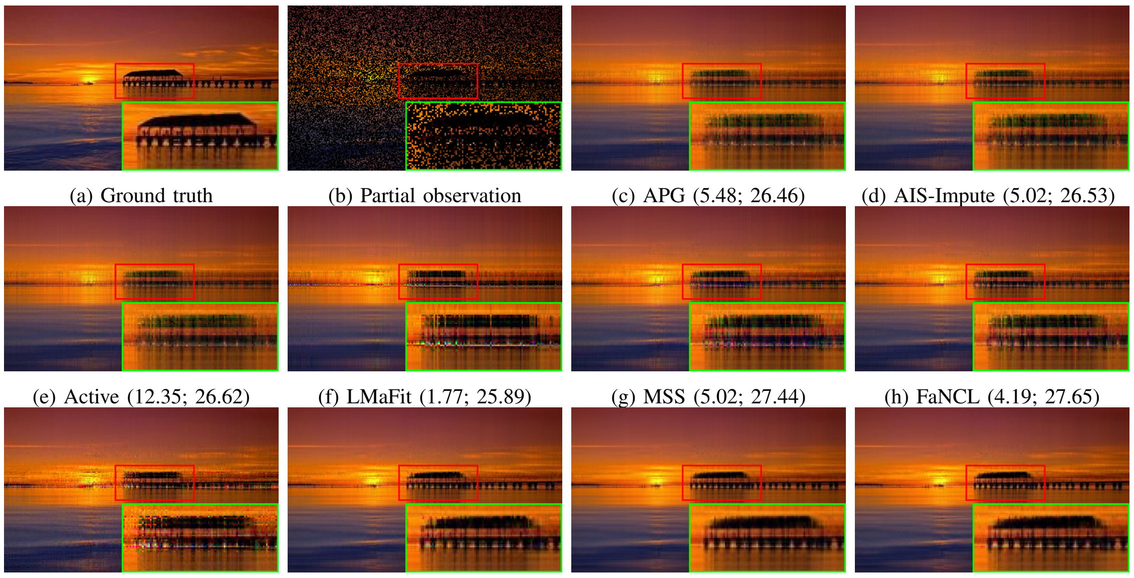
V. EXPERIMENTAL RESULTS

In this section, we perform extensive experiments on MC (Section V-A) and RPCA (Section V-B) to evaluate the effectiveness and efficiency of our method. The experiments are performed on both synthetic data and real-word data. Three instantiations of our GUIG model, i.e., GUIG_l_0, GUIG_l_p (we set $p = 0.5$ in all the experiments) and GUIG_log are adopted. Experiments are performed on a PC with Windows 10 OS, Intel(R) Core(TM) i7-7700K CPU (4.20 GHz), and 16 G RAM. The code is written in Matlab 2016b.

A. Matrix Completion

We compare our method with state-of-the-art spectral function based LRMR methods and bi-linear factorization based LRMR methods. All the executable codes are obtained from the original author's homepage.

The compared nuclear norm based LRMR methods include: Accelerated proximal gradient (APG) [25], Active subspace selection [44] denoted by "Active", Accelerated inexact Soft-Impute algorithm (AIS-Impute) [43]. The compared Schatten-p quasi norm based LRMR methods are: IRNN [24], IRucLq [27], MSS [40], D-N [31], F-N [32], FaNCL [45]. We also compare our method with bi-linear low rank matrix factorization model LMaFit [12], rank one matrix pursuit algorithm ER1MP [37] and the CUR matrix decomposition method CUR+ [46]. Both of them use a low rank matrix to



(a) Ground truth (b) Partial observation (c) APG (5.48; 26.46) (d) AIS-Impute (5.02; 26.53)

(e) Active (12.35; 26.62) (f) LMaFit (1.77; 25.89) (g) MSS (5.02; 27.44) (h) FaNCL (4.19; 27.65)

(i) TNNR (12.54; 27.06) (j) GUIG_l_0 (**1.21; 28.25**) (k) GUIG_l_p (**1.04**; **28.62**) (l) GUIG_log (**1.11; 28.52**)

Fig. 5. Image inpainting results on an image with 65% random missing pixels. We compare the visual results of algorithms APG [25], AIS-Impute [43], Active [44], LMaFit [12], MSS [16], FaNCL [45], TNNR [15] with our algorithms GUIG_l_0, GUIG_l_p and GUIG_log (best viewed in colors). We also report their CPU time (in seconds) and PSNR values (dB) in the parentheses as: (Time; PSNR).

TABLE IV
Average PSNR values (dB) and CPU time (seconds) of the compared methods for image inpainting on 11 test images.

	APG	AIS-Impute	Active	LMaFit	MSS
PSNR	26.07	26.15	26.20	24.91	28.10
Time	6.8	4.5	15.1	1.4	4.9
	FaNCL	TNNR	GUIG_l_0	GUIG_l_p	GUIG_log
PSNR	27.82	28.21	**28.44**	**28.50**	**28.72**
Time	4.7	16.3	**1.2**	**1.1**	**1.3**

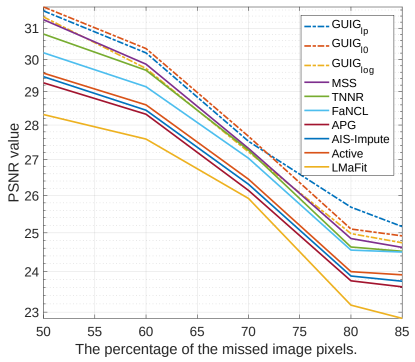


Fig. 6. Average PSNR values of the compared methods for inpainting on the 11 tested images with 50% – 85% missing pixels.

approximate the incomplete observation without regularization on the singular values.

1) Synthetic Data: We first generate a synthetic low rank matrix $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ and $r \ll \min(m, n)$. The entries of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are i.i.d. samples from a standard Gaussian distribution $\mathcal{N}(0, 1)$. A small portion (s%) of the matrix $\boldsymbol{M}$ is uniformly selected as observation and the $\mathcal{N}(0, \sigma)$ Gaussian noise is added to these observations. We conduct experiments by varying the matrix dimension m, n, the rank r, and the ratio of observation s%. In all the experiments, we fix the noise level $\sigma = 0.1$ and set the dimension of the matrix larger than the ground-truth matrix rank as $d = 1.25\times r$ [12]. We repeat the experiments 20 times, and calculate the average accuracy and CPU time.

We adopt $g(x) = |x|^0$, $g(x) = |x|^p$ ($p = 0.5$) and $g(x) = \log(|x|)$ in our GUIG function $G_g(\boldsymbol{X})$, resulting in the proposed GUIG_l_0, GUIG_l_p and GUIG_log models. The relative square root error (RSRE), defined as $\frac{\|\boldsymbol{M}-\boldsymbol{U}^*\boldsymbol{V}^{*T}\|_F}{\|\boldsymbol{U}^*\boldsymbol{V}^{*T}\|_F}$, is used to evaluate the LRMR performance. The RSRE, total iteration number and the CPU time are reported in Table I. With the increase of the matrix dimension, we decrease the observation ratio and increase the matrix rank.

From Table I, we can see that our methods consistently outperform the compared algorithms in both RMSE and CPU time. In particular, the proposed models are several orders faster than the SVD based algorithms APG [25], AIS-Impute [43], IRNN [24] and IRucLq [27], and they are also considerably faster than the state-of-the-art methods LMafit [12], MSS [40], D-N [31], F-N [32] and FaNCL [45]. For matrices with dimension $m, n > 2 \times 10^3$, the CPU time of IRNN [24] and IRucLq [27] exceeds 10^3, which is not comparable to other algorithms. Therefore, we do not report the results of IRNN [24] and IRucLq [27] for larger scale matrix since they require much more computational time and memory.

In terms of RMSE, one can see that the non-convex regularization models (Schatten-p quasi norm and Logsum) result in lower RMSE than the convex nuclear norm model. Compared with the Schatten-p quasi norm solvers such as IRNN [24], IRucLq [27], D-N [31], F-N [32] and MSS [40], our method GUIG_l_p provides a better solution with lower RSRE.

2) MovieLens and Netflix: In this part, we perform experiments on the popular recommendation system data sets *MovieLens*[2] and *Netflix*, whose statistics are listed in Table

[2]http://www.grouplens.org/node/73

TABLE V
RPCA PERFORMANCE ON THE SYNTHETIC DATA. RSRE IS SCALED BY $\times 10^{-2}$ AND TIME IS IN SECONDS.

	m = 500 Support: 10%, rank = 5			m = 2000 Support: 10%, rank = 10			m = 5000 Support: 15%, rank = 15			m = 10000 Support: 20%, rank = 20		
	RSRE	rank	time	RSRE	rank	time	RSRE	rank	time	RSRE	rank	time
MoG-RPCA [47]	2.07	5	1.27	0.997	10	26.6	0.68	15	185	–	–	NaN
ALM-RPCA [26]	5.19	5	0.737	2.53	10	23.4	1.52	15	192	–	–	NaN
SpLq [48]	3.88	25	175	1.93	26	3192	–	–	NaN	–	–	NaN
RegL1-ALM [49]	3.5	5	30.7	1.72	10	594	1.11	15	4449	–	–	NaN
WNNM-RPCA [2]	2.68	5	5.20	1.31	10	174	0.908	15	1101	–	–	NaN
D-N [31]	2.12	5	0.580	1.08	10	8.9	0.698	15	78.2	0.541	20	305
F-N [32]	2.15	5	0.51	1.16	10	7.3	0.707	15	69	0.553	20	273
GUIG_l_p-RPCA	**2.06**	5	**0.284**	**1.01**	10	**3.27**	**0.680**	15	**32.3**	**0.513**	20	**140**
GUIG_l_0-RPCA	**2.03**	5	**0.235**	**1.00**	10	**3.12**	**0.680**	15	**21.8**	**0.530**	20	**103**
GUIG_log-RPCA	**2.03**	5	**0.265**	**1.00**	10	**2.80**	**0.670**	15	**26.8**	**0.502**	20	**158**

TABLE VI
QUANTITATIVE RESULTS AND CPU TIME (SECONDS) OF THE COMPARED METHODS ON FOREGROUND BACKGROUND SEPARATION.

Methods	*Watersurface*		*Fountain*		*Airport*		*Curtain*	
	S value	CPU time	S value	CPU time	S value	CPU time	S value	CPU time
MoG-RPCA [47]	0.4758	57.3	0.5243	37.5	0.4831	468.5	0.6139	241.1
ALM-RPCA [26]	0.7782	72.7	0.5864	54.3	0.4015	517.7	0.5951	288.9
SpLq [48]	0.7903	128.2	0.6219	108.2	0.5252	1124.1	0.7804	754.5
RegL1 [49]	0.7883	58.6	0.6023	49.6	0.5156	551.8	0.7741	272.8
WNNM [2]	0.8015	85.2	0.6354	72.4	0.5382	720.2	0.7995	352.2
D-N [31]	0.8025	47.8	0.6377	43.2	0.5409	335.4	0.8018	208.1
F-N [32]	0.8010	42.3	0.6395	36.9	0.5395	317.3	0.8004	185.2
GUIG_l_0	0.8010	21.3	0.6321	18.8	0.5379	152.3	0.7869	75.1
GUIG_l_p	0.8021	15.6	0.6388	13.8	0.5441	137.1	0.8010	62.3
GUIG_log	**0.8125**	22.8	**0.6401**	19.3	**0.5487**	162.5	**0.8174**	83.1

II. We randomly pick out 50%, 70% and 80% of the observed entries as the training data and use the remaining for testing. The root mean square error (RMSE), defined as $\sqrt{\sum_{(i,j)\in\mathcal{T}}(\boldsymbol{X}_{i,j}-\boldsymbol{M}_{i,j})^2/|\mathcal{T}|}$ on the test set $\mathcal{T}$, is used as the quantitative measure.

The results of compared methods are shown in Table III. Our GUIG models (GUIG_l_0, GUIG_l_p, GUIG_log) show distinct advantages over the convex nuclear norm based methods, and they also outperform the state-of-the-art Schatten-p quasi norm regularized models. The methods LMaFit [12] and ER1MP [37] impose no regularization on the singular values of the target matrix, and they have comparable CPU time with our models; however, their RMSE results are very poor. Compared with the non-convex spectral function regularization models MSS [16], D-N [31], F-N [32] and FaNCL [45], our methods achieve better RMSE results. On *MovieLens-10M* and *Netflix*, some of the SVD based algorithms are non-executable due to the large size of data, and we do not report their results on the two data sets.

We plot the CPU time vs. RMSE curves in Fig. 3. One can see that there are obvious gaps between our methods (GUIG_l_0, GUIG_l_p, GUIG_log) and the compared ones, especially on large scale data sets. For *MovieLens-1M* (Fig. 3 (a)), our algorithms converge in less than 10 seconds and achieve the lowest RMSE, while the compared algorithms converge slower, and some methods converge in more than 100 seconds. Similar observations can be found in Fig. 3 (b) for *MovieLens-10M* and Fig. 3 (c) for *Netfilx*. Specifically, our algorithms can solve the *Netflix* problem in less than 20 minutes with the best accuracies. Please note that since some of the compared algorithms are non-executable on *MovieLens-10M* and *Netfilx*, we do not report their results.

3) Image Inpainting: Natural images usually have many repeated patterns, which can be approximated by a low rank matrix. Following the setup in [15], we apply our proposed models to the image inpainting problem. For a given image, we randomly pick up 35% of the image pixels as observation data, and estimate the remaining pixels.

In Table IV, we present the average PSNR results and CPU time of all competing methods on 11 widely used test images in [15], whose snapshots are shown in Fig. 4. One can see that our methods achieve the highest PSNR measures and the fastest speeds among all the compared methods. Fig. 5 shows the inpainting results of an image. One can see that all the competing models are able to deliver a satisfactory result, while our models can preserve much better the details and structures of the image. Note that the compared methods may create some visually unpleasing artifacts, as shown in the zoom-in regions. In Fig. 6, we illustrate the inpainting results of all the compared methods on images with 50%−85% missing pixels. It can be seen from Fig. 6 that the proposed GUIG methods uniformly outperform all the competing methods in all cases, which verifies the effectiveness of the proposed methods.

B. RPCA

In this part, we evaluate the effectiveness and efficiency of our models on the RPCA problem. We compare our methods with state-of-the-art RPCA algorithms, including MoG-RPCA [47], ALM-RPCA [26], SpLq [48], RegL1-ALM [49], D-N [31], F-N [32] and WNNM-RPCA [2].

1) Synthetic Data: The synthetic matrix $\boldsymbol{Y}$ is generated by $\boldsymbol{Y} = \boldsymbol{U}_0\boldsymbol{V}_0^T + \boldsymbol{S} + \boldsymbol{N}$, where $\boldsymbol{U}_0 \in \mathbb{R}^{m\times r}$, $\boldsymbol{V}_0 \in \mathbb{R}^{n\times r}$ are

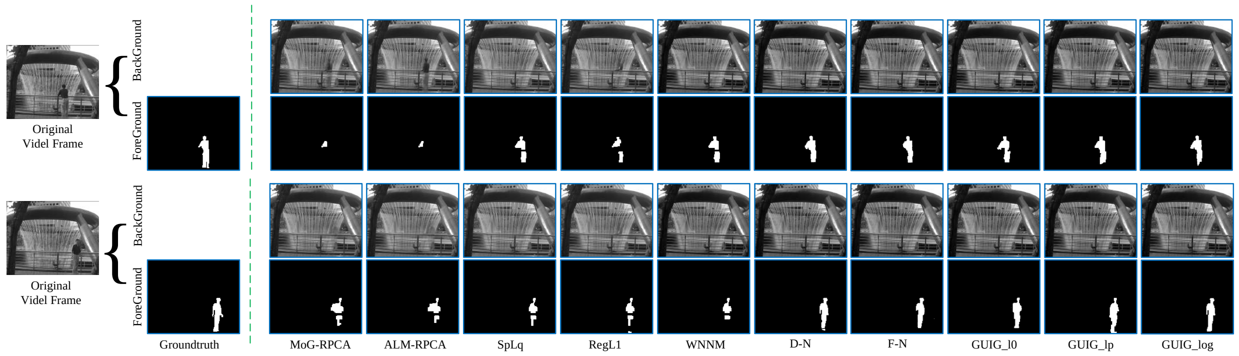


Fig. 7. From left to right: typical frames from the *Fountain* sequence, groundtruth foreground objects, foreground and background separation results by all competing methods.

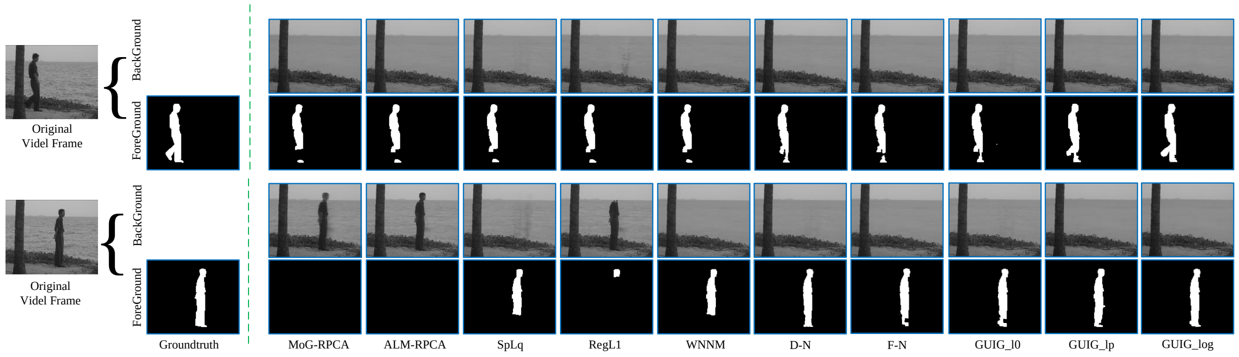


Fig. 8. From left to right: typical frames from the *Watersurface* sequence, groundtruth foreground objects, foreground and background separation results by all competing methods.

matrices with i.i.d. entries sampled from standard Gaussian distributions $\mathcal{N}(0,1)$ and $r \ll \min(m,n)$. The matrix $\boldsymbol{S}$ is a sparse matrix with $s\%$ nonzero entries, whose locations are uniformly selected and values follow uniform distribution in interval $[-50,50]$. $\boldsymbol{N}$ is Gaussian noise with zero mean and standard deviation 0.3. In all the experiments, we set $d = 1.25 \times r$ and vary rank r and ratio $s\%$. We repeat the experiments 30 times and report the average results.

The experimental results are given in Table V, where we show the RSRE (RSRE $= \|\boldsymbol{U}\boldsymbol{V}^T - \boldsymbol{U}_0\boldsymbol{V}_0^T\|_F / \|\boldsymbol{U}_0\boldsymbol{V}_0^T\|_F$), estimated rank and the CPU time. NaN in Table V means out of memory or the CPU time exceeds 10^4 seconds. It can be seen that our proposed GUIG methods demonstrate clear advantage over the compared methods in terms of RSRE and CPU time. They are much faster than existing methods while achieving the lowest RSRE in almost all tests. The method MoG-RPCA [47] shows very close RSRE to our methods, but it is much slower. The bilinear norm methods D-N [31] and F-N [32] achieve comparable RSRE results to our method but cost more computational time. When the matrix size reaches 5000×5000, the method SpLq [48] is too slow to run, and RegL1-ALM [49] and WNNM-RPCA [2] take thousands of seconds. When the matrix size reaches $10,000 \times 10,000$, only our methods can run due to their low memory requirement and low complexity.

2) Video Foreground Background Separation: We apply the proposed methods to video foreground and background separation, where the background of the video sequence is assumed to have a low rank structure and the foreground is sparsely distributed.

In the experiments, four benchmark video sequences provided in [50], including *Fountain*, *Watersurface*, *Curtain* and *Airport*, are adopted. For each sequence, ground-truth foreground regions of 20 frames are provided [50] for quantitatively evaluating the testing results. $S(A,B) = \frac{A \bigcap B}{A \bigcup B}$ is used to measure the similarity between the estimated foreground and the ground-truth. Follow the same setting as in [2], the Markov random field (MRF) is used to generate the binary foreground map. The quantitative results of the S value and CPU time by the compared methods are shown in Table VI. One can see that on all the video sequences, the proposed GUIG models have better S values than the other compared methods. Meanwhile, the GUIG models cost the least amount of CPU time.

Visual comparison results are given in Fig. 7 and Fig. 8. It can be seen that GUIG models are capable of extracting clear background scenes and present more accurate foreground objects. In contrast, the compared methods will generate some ghost shadows in the background, and extract less accurate foreground objects.

VI. Conclusion

We proposed a generalized unitarily invariant gauge (GUIG) function for low rank matrix recovery (LRMR). We proved that the GUIG function generalizes the conventional spectral functions such as the Schatten-p quasi norm and logsum function on singular values, while it does not act on the singular values of a matrix. We further showed that flexible bi-linear factorization representation can be easily constructed from the GUIG function, which is much less memory demanding and can be solved in an SVD free manner. An advanced initialization strategy was proposed, with which our GUIG algorithms can converge quickly in a few iterations. Extensive experiments on MC and RPCA problems were conducted, and the results verified that GUIG is an effective and efficient approach for LRMR, especially for large scale problems. For example, it takes less than 20 minutes to solve the well-known Netflix problem with leading accuracy.

References

[1] Y. Liu, S. Yang, P. Wu, C. Li, and M. Yang, "$l\_\{1\}$-norm low-rank matrix decomposition by neural networks and mollifiers," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 2, pp. 273–283, 2016. 1

[2] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *International journal of computer vision*, vol. 121, no. 2, pp. 183–208, 2017. 1, 5, 10, 11

[3] T.-H. Oh, H. Kim, Y.-W. Tai, J.-C. Bazin, and I. So Kweon, "Partial sum minimization of singular values in rpca for low-level vision," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 145–152. 1, 5

[4] X. Shen and Y. Wu, "A unified approach to salient object detection via low rank matrix recovery," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 853–860. 1

[5] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010. 1, 5

[6] X. Jia, X. Feng, and W. Wang, "Rank constrained nuclear norm minimization with application to image denoising," *Signal Processing*, vol. 129, pp. 1–11, 2016. 1

[7] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in neural information processing systems*, 2009, pp. 2080–2088. 1, 5

[8] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE transactions on neural networks and learning systems*, pp. 1–13, 2016. 1, 5

[9] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010. 1, 2, 5

[10] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 665–674. 1, 3, 6

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009. 1

[12] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, pp. 1–29, 2012. 1, 7, 8, 9, 10

[13] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011. 1, 5, 6

[14] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank svd via fast alternating least squares." *Journal of Machine Learning Research*, vol. 16, pp. 3367–3402, 2015. 1, 6

[15] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 9, pp. 2117–2130, 2013. 1, 9, 10

[16] C. Xu, Z. Lin, and H. Zha, "A unified convex surrogate for the schatten-p norm." in *AAAI*, 2017, pp. 926–932. 1, 2, 3, 4, 6, 7, 8, 9, 10, 14

[17] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l_1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5, pp. 877–905, 2008. 1

[18] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3. IEEE, 2003, pp. 2156–2162. 1, 4

[19] X. Peng, C. Lu, Z. Yi, and H. Tang, "Connections between nuclear-norm and frobenius-norm-based representations," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 218–224, 2018. 1

[20] F. Zhang, J. Yang, J. Qian, and Y. Xu, "Nuclear norm-based 2-dpca for extracting features from images," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 10, pp. 2247–2260, 2015. 1

[21] W. Hu, D. Tao, W. Zhang, Y. Xie, and Y. Yang, "The twist tensor nuclear norm for video completion," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 12, pp. 2961–2973, 2017. 1, 5

[22] F. Shang, Y. Liu, and J. Cheng, "Tractable and scalable schatten quasi-norm approximations for rank minimization," in *Artificial Intelligence and Statistics*, 2016, pp. 620–629. 1, 2, 3, 4

[23] M. Zhang, Z.-H. Huang, and Y. Zhang, "Restricted p-isometry properties of nonconvex matrix recovery," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4316–4323, 2013. 1

[24] C. Lu, J. Tang, S. Yan, and Z. Lin, "Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 829–839, 2016. 1, 5, 7, 8, 9

[25] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, 2010. 1, 5, 7, 8, 9

[26] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv preprint arXiv:1009.5055*, 2010. 1, 10

[27] M.-J. Lai, Y. Xu, and W. Yin, "Improved iteratively reweighted least squares for unconstrained smoothed l_q minimization," *SIAM Journal on Numerical Analysis*, vol. 51, no. 2, pp. 927–957, 2013. 1, 5, 7, 8, 9

[28] K. Mohan and M. Fazel, "Iterative reweighted algorithms for matrix rank minimization," *Journal of Machine Learning Research*, vol. 13, no. Nov, pp. 3441–3473, 2012. 1

[29] R. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino, "Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2488–2495. 2

[30] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Advances in neural information processing systems*, 2005, pp. 1329–1336. 2

[31] F. Shang, J. Cheng, Y. Liu, Z.-Q. Luo, and Z. Lin, "Bilinear factor matrix norm minimization for robust pca: Algorithms and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 9, pp. 2066–2080, 2018. 2, 7, 8, 9, 10, 11

[32] F. Shang, Y. Liu, and J. Cheng, "Scalable algorithms for tractable schatten quasi-norm minimization," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2, 7, 8, 9, 10, 11

[33] ——, "Unified scalable equivalent formulations for schatten quasi-norms," *arXiv preprint arXiv:1606.00668*, 2016. 2

[34] F. Bach, "Convex relaxations of structured matrix factorizations," *Technical Report 00861118, HAL*, 2013. 2, 3

[35] K. Lee and Y. Bresler, "Admira: Atomic decomposition for minimum rank approximation," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4402–4416, 2010. 2

[36] J. D. Lee, B. Recht, N. Srebro, J. Tropp, and R. R. Salakhutdinov, "Practical large-scale optimization for max-norm regularization," in *Advances in Neural Information Processing Systems*, 2010, pp. 1297–1305. 2, 3

[37] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye, "Orthogonal rank-one matrix pursuit for low rank matrix completion," *SIAM Journal on Scientific Computing*, vol. 37, no. 1, pp. A488–A514, 2015. 2, 5, 7, 8, 10

[38] T. Zhao, Z. Wang, and H. Liu, "A nonconvex optimization framework for low rank matrix estimation," in *Advances in Neural Information Processing Systems*, 2015, pp. 559–567. 3, 6

[39] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014. 3, 7, 14

[40] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, pp. 1–35, 2017. 3, 5, 8, 9, 14

[41] A. S. Lewis, "The convex analysis of unitarily invariant matrix functions," *Journal of Convex Analysis*, vol. 2, no. 1, pp. 173–183, 1995. 3

[42] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," in *Advances in neural information processing systems*, 2016, pp. 4152–4160. 6

[43] Q. Y. J. T. Kwok, "Accelerated inexact soft-impute for fast large-scale matrix completion," 2015. 7, 8, 9

[44] C.-J. Hsieh and P. Olsen, "Nuclear norm minimization via active subspace selection," in *International Conference on Machine Learning*, 2014, pp. 575–583. 7, 8, 9

[45] Y. Quanming, J. T. Kwok, T. Wang, and T.-Y. Liu, "Large-scale low-rank matrix learning with nonconvex regularizers," *arXiv preprint arXiv:1708.00146*, 2017. 7, 8, 9, 10

[46] M. Xu, R. Jin, and Z.-H. Zhou, "Cur algorithm for partially observed matrices," in *International Conference on Machine Learning*, 2015, pp. 1412–1421. 7, 8

[47] Q. Zhao, D. Meng, Z. Xu, W. Zuo, and L. Zhang, "Robust principal component analysis with complex noise," in *International Conference on Machine Learning*, 2014, pp. 55–63. 10, 11

[48] F. Nie, H. Wang, X. Cai, H. Huang, and C. Ding, "Robust matrix completion via joint schatten p-norm and lp-norm minimization," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 566–574. 10, 11

[49] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi, "Practical low-rank matrix approximation under robust l 1-norm," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1410–1417. 10, 11

[50] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1459–1472, 2004. 11

[51] R. Bhatia, *Matrix analysis*. Springer Science & Business Media, 2013, vol. 169. 13

[52] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990. 13

Appendix A
Appendix of the Paper

A. *Proof of Theorem 2*

To prove Theorem 2, we first introduce the definition of majorisation and some key lemmas.

Definition 4. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. We say that $\boldsymbol{x}$ is majorised by $\boldsymbol{y}$, denoted by $\boldsymbol{x} \prec \boldsymbol{y}$, if for $1 \le k \le n$ the following holds*

$$\sum_{j=1}^{k} \boldsymbol{x}_j^{\downarrow} \le \sum_{j=1}^{k} \boldsymbol{y}_j^{\downarrow}, \ \sum_{j=1}^{n} \boldsymbol{x}_j^{\downarrow} = \sum_{j=1}^{n} \boldsymbol{y}_j^{\downarrow} \tag{31}$$

where $\boldsymbol{x}_j^{\downarrow}$ is the vector obtained by rearranging the coordinates of $\boldsymbol{x}$ in descending order.

Lemma 2. *[51] [**Schur's Theorem**] Let $\boldsymbol{A}$ be an $n\times n$ Hermitian matrix. Let $diag(\boldsymbol{A})$ denote the vector whose coordinates are the diagonal entries of $\boldsymbol{A}$ and let $\lambda(\boldsymbol{A})$ denote the vector whose coordinates are the eigenvalues of $\boldsymbol{A}$ specified in any order, the following holds*

$$diag(\boldsymbol{A}) \prec \lambda(\boldsymbol{A}) \tag{32}$$

Lemma 3. *[51] [**Theorem II.3.1**] Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. The following two conditions are equivalent:*

- $\boldsymbol{x} \prec \boldsymbol{y}$
- $\sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \le \sum_{i=1}^{n} \phi(\boldsymbol{y}_i)$ *for all convex functions ϕ from $\mathbb{R}$ to $\mathbb{R}$.*

Lemma 4. *[52] [**Theorem 3.3.14 (c)**] Given any matrices $\boldsymbol{A} \in \mathbb{R}^{m\times l}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times l}$, and denote by $\{\sigma(\cdot)\}$ the singular values in descending order. Then for any real-valued function f such that $\varphi(t) \equiv f(e^t)$ is increasing and convex, we have*

$$\sum_{i=1}^{k} f(\sigma_i(\boldsymbol{A}\boldsymbol{B}^T)) \le \sum_{i=1}^{k} f(\sigma_i(\boldsymbol{A})\sigma_i(\boldsymbol{B}^T)). \tag{33}$$

where $1 \le k \le q$ and $q = min\{m, n, l\}$.

We are now ready to prove Theorem 2.

Proof. Given the matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ and a decomposition $\boldsymbol{X} = \boldsymbol{U}\Lambda\boldsymbol{V}^T = \widehat{\boldsymbol{U}}|\Lambda|\widehat{\boldsymbol{V}}^T$, where $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix with diagonal elements $\lambda_1, \lambda_2, \cdots, \lambda_n$ such that $|\lambda_1| \ge |\lambda_2| \ge, \cdots, \ge |\lambda_n|$. The matrices $\boldsymbol{U}, \widehat{\boldsymbol{U}} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{V}, \widehat{\boldsymbol{V}} \in \mathbb{R}^{n\times d}$ are of unit l_2-norm column length. Let $\boldsymbol{A} = \widehat{\boldsymbol{U}}|\Lambda|^{\frac{1}{2}}$ and $\boldsymbol{B} = |\Lambda|^{\frac{1}{2}}\widehat{\boldsymbol{V}}^T$. Denote by $\overline{\boldsymbol{A}} = \boldsymbol{A}^T\boldsymbol{A}$ and $\overline{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{B}^T$, from Lemma 2 we have

$$|\Lambda| = diag(\overline{\boldsymbol{A}}) \prec \ \sigma(\overline{\boldsymbol{A}}) = \sigma^2(\boldsymbol{A})$$
$$|\Lambda| = diag(\overline{\boldsymbol{B}}) \prec \ \sigma(\overline{\boldsymbol{B}}) = \sigma^2(\boldsymbol{B})$$

Since $|\lambda_1| \ge \cdots \ge |\lambda_n|$, we have $|\Lambda| \prec \frac{\sigma^2(\boldsymbol{A})+\sigma^2(\boldsymbol{B})}{2}$. For concave function g in Theorem 2, we know $-g(|x|)$ is convex with respect to $|x|$ and according to Lemma 3 we have

$$\sum_{i=1}^{d} -g(|\lambda_i|) \le \sum_{i=1}^{d} -g\left(\frac{\sigma_i^2(\boldsymbol{A}) + \sigma_i^2(\boldsymbol{B})}{2}\right).$$

Then the following equation holds

$$\begin{aligned}&\textstyle\sum_{i=1}^{d} g(|\lambda_i|) \ge \sum_{i=1}^{d} g(\frac{\sigma_i^2(\boldsymbol{A})+\sigma_i^2(\boldsymbol{B})}{2})\\ &\ge \textstyle\sum_{i=1}^{d} g(\sigma_i(\boldsymbol{A})\sigma_i(\boldsymbol{B}))\text{①}\\ &\ge \textstyle\sum_{i=1}^{d} g(\sigma_i(\boldsymbol{A}\boldsymbol{B}))\text{②} = \sum_{i=1}^{d} g(\sigma_i(\boldsymbol{X}))\end{aligned}$$

The inequality ① holds due to that the function g is increasing and the fact that $a^2+b^2 \ge 2ab$. The inequality ② holds based on Lemma 4. The equality in ② holds if and only if $\boldsymbol{X} = \boldsymbol{U}\Lambda\boldsymbol{V}^T$ is the SVD and Λ is the singular value matrix. Since $G_g(\boldsymbol{X})$ achieves the minimum of $\sum_{i=1}^{d} g(|\lambda_i|)$ among all the possible decompositions, we have the result that

$$G_g(\boldsymbol{X}) = \sum_{i=1}^{d} g(\sigma_i(\boldsymbol{X}))$$

The proof is completed. □

B. *Proof of Theorem 3*

Proof. Let $\lambda_i = \lambda_i^1\lambda_i^2$ and $g(\lambda_i) = \min_{\lambda_i=\lambda_i^1\lambda_i^2} g_1(\lambda_i^1) + g_2(\lambda_i^2)$, there holds

$$\inf_{\lambda_i} g(\lambda_i) = \inf_{\lambda_i^1,\lambda_i^2} g_1(\lambda_i^1) + g_2(\lambda_i^2) \tag{34}$$

for $\lambda_i, \lambda_i^1, \lambda_i^2 \in \mathbb{R}$ such that $\boldsymbol{X} = \sum_{i=1}^{d} \lambda_i\boldsymbol{u}_i\boldsymbol{v}_i^T$ and $\boldsymbol{X} = \sum\limits_{i=1}^{d} \lambda_i^1\lambda_i^2\boldsymbol{u}_i\boldsymbol{v}_i^T$. Equality in (34) can be easily verified since the infimum is equal to the minimum for bounded functions.

Given $\lambda_i = \lambda_i^1\lambda_i^2$, we have

$$\boldsymbol{X} = \sum_{i=1}^{d} \lambda_i\boldsymbol{u}_i\boldsymbol{v}_i^T = \sum_{i=1}^{d} \underbrace{\lambda_i^1\boldsymbol{u}_i}\ \underbrace{\lambda_i^2\boldsymbol{v}_i^T}$$

Let $\widetilde{\boldsymbol{u}}_i = \lambda_i^1\boldsymbol{u}_i$ and $\widetilde{\boldsymbol{v}}_i = \lambda_i^1\boldsymbol{v}_i$, since $\|\boldsymbol{u}_i\|_2 = \|\boldsymbol{v}\|_2 = 1$, we have $|\lambda_i^1| = \|\widetilde{\boldsymbol{u}}_i\|_2$ and $|\lambda_i^2| = \|\widetilde{\boldsymbol{v}}_i\|_2$. According to Eq. (34), we have

$$\min_{\lambda_i} g(|\lambda_i|) = \min_{\widetilde{\boldsymbol{u}}_i,\widetilde{\boldsymbol{v}}_i} g_1(\|\widetilde{\boldsymbol{u}}_i\|_2) + g_2(\|\widetilde{\boldsymbol{v}}_i\|_2) \tag{35}$$

Denote by $\boldsymbol{U}$ the matrix with $\widetilde{\boldsymbol{u}}_i$ as its ith column and the same to $\boldsymbol{V}$. Add up Eq. (35) from $i = 1$ to d, we immediately have

$$\min_{\boldsymbol{\lambda}} \sum_{i=1}^{d} g(|\lambda_i|) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2)+\sum_{i=1}^{d} g_2(\|V_{\cdot i}\|_2)$$

The proof is completed. □

C. Proof of Theorem 4

Proof. Since $\mathfrak{g} \circ \sigma(\boldsymbol{X}) = \sum_{i=1}^{r} g(\sigma_i(\boldsymbol{X}))$, according to Theorem 3, for each $g(\sigma_i(\boldsymbol{X}))$ we have $g(\sigma_i(\boldsymbol{X})) \le g_1(\alpha_i) + g_2(\beta_i)$ for any α_i, β_i where $\sigma_i(\boldsymbol{X}) = \alpha_i\beta_i$. Assume that there exists α_i^*, β_i^* such that $\sigma_i(\boldsymbol{X}) = \alpha_i^*\beta_i^*$ and $g(\sigma_i(\boldsymbol{X})) = g_1(\alpha_i^*) + g_2(\beta_i^*)$ (the equality is attainable according to Theorem 3). Let $\boldsymbol{X}^* = \boldsymbol{U}\Sigma\boldsymbol{V}^T$ be the SVD of the matrix $\boldsymbol{X}^*$, $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \cdots, \alpha_d^*]^T$ and $\boldsymbol{\beta}^* = [\beta_1^*, \beta_2^*, \cdots, \beta_d^*]^T$ such that $\Sigma_{ii} = \alpha_i^*\beta_i^*$ for $i \le r$ and $\alpha_i^* = 0$, $\beta_i^* = 0$ for $r < i \le d$. Denote by $\overline{\boldsymbol{U}} = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\alpha}^*)$ and $\overline{\boldsymbol{V}} = \boldsymbol{V}\mathrm{diag}(\boldsymbol{\beta}^*)$, we have $\sum_{i=1}^{r} g(\sigma_i(\boldsymbol{X})) = \sum_{i=1}^{d} g_1(\|\overline{\boldsymbol{U}}_{\cdot i}\|_2) + \sum_{i=1}^{d} g_2(\|\overline{\boldsymbol{V}}_{\cdot i}\|_2)$. According to Eq. (14), we have:

$$\begin{aligned}\mathcal{F}(\boldsymbol{X}^*) &= f\left(\overline{\boldsymbol{U}}\overline{\boldsymbol{V}}^T\right) + \textstyle\sum_{i=1}^{d} g_1(\|\overline{\boldsymbol{U}}_{\cdot i}\|_2) + g_2(\|\overline{\boldsymbol{V}}_{\cdot i}\|_2)\\ &\ge f\left(\boldsymbol{U}^*\boldsymbol{V}^{*T}\right) + \textstyle\sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}^*\|_2) + g_2(\|\boldsymbol{V}_{\cdot i}^*\|_2)\\ &= F\left(\boldsymbol{U}^*, \boldsymbol{V}^{*T}\right)\end{aligned} \tag{36}$$

Denoted by $\overline{\boldsymbol{X}} = \boldsymbol{U}^*\boldsymbol{V}^{*T}$. According to Theorem 2 and Theorem 3, we have $\sum_{i=1}^{r} g(\sigma_i(\boldsymbol{X})) = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}\|_2) + g_2(\|\boldsymbol{V}_{\cdot i}\|_2)$, and thus $\sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}^*\|_2) + g_2(\|\boldsymbol{V}_{\cdot i}^*\|_2) \ge \sum_{i=1}^{r} g(\sigma_i(\overline{\boldsymbol{X}}))$ holds. Consequently, the following equation holds:

$$\begin{aligned}F(\boldsymbol{U}^*, \boldsymbol{V}^*) &= f\left(\boldsymbol{U}^*\boldsymbol{V}^{*T}\right) + \textstyle\sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}^*\|_2) + g_2(\|\boldsymbol{V}_{\cdot i}^*\|_2)\\ &\ge f\left(\overline{\boldsymbol{X}}\right) + \textstyle\sum_{i=1}^{r} g(\sigma_i(\overline{\boldsymbol{X}}))\\ &\ge f(\boldsymbol{X}^*) + \textstyle\sum_{i=1}^{r} g(\sigma_i(\boldsymbol{X}^*)) = \mathcal{F}(\boldsymbol{X}^*)\end{aligned} \tag{37}$$

Thus we have the equality $F(\boldsymbol{U}^*, \boldsymbol{V}^*) = \mathcal{F}(\boldsymbol{X}^*)$. According to Eq. (36), we have $F(\boldsymbol{U}^*, \boldsymbol{V}^*) = F\left(\overline{\boldsymbol{U}}, \overline{\boldsymbol{V}}\right)$; therefore $(\overline{\boldsymbol{U}}, \overline{\boldsymbol{V}})$ is also a solution of problem (14). According to Eq. (37), we have $\mathcal{F}(\boldsymbol{X}^*) = \mathcal{F}\left(\overline{\boldsymbol{X}}\right)$, and thus $\overline{\boldsymbol{X}}$ is a solution of problem (1). The proof is completed. □

D. Proof of Theorem 5

The convergence analysis of the proposed algorithm is analogous to that of [40] and [16], please refer to [16], [40]and [39] for more about the definition and properties of the semi-algebraic sets and functions.

Let's first present some propositions before giving the proof.

Proposition 6. *The matrix operators $\|\boldsymbol{X}\|_{2,p}^p$ $(0 \le p \le 1)$ and $\|\boldsymbol{X}\|_{2,log}$ are semi-algebraic functions.*

Proof. For a given $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, the l_2-norm $\|\cdot\|_2$ on the column of $\boldsymbol{X}$ is semi-algebraic. The power function $|x|^p$ for $0 \le p \le 1$ and the finite sum of the log function are semi-algebraic, and it is natural to conclude that their compositions are also semi-algebraic. The proof is completed. □

As Algorithm 1 is a special case of the general optimization algorithm in [40], to prove the convergence in Theorem 5, we only need to show that the conditions ensuring the sequence convergence of [40] [Theorem 2] are satisfied.

Proposition 7. *The sequence generated by Algorithm 1 is a Cauchy sequence and converges to a critical point if the following conditions hold:*

1) *The sequence of $F(\boldsymbol{U}^k, \boldsymbol{V}^k)$ is non-ascending.*
2) *Within any bounded consecutive iterations, every block $\boldsymbol{U}$ or $\boldsymbol{V}$ is updated at least once.*
3) *$\{(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ is a bounded sequence.*
4) *F is a semi-algebraic function.*
5) *$\nabla_{\boldsymbol{U}}^k f(\boldsymbol{U})$ and $\nabla_{\boldsymbol{V}}^k f(\boldsymbol{V})$ is Lipschitz continuous.*
6) *$\nabla f(\boldsymbol{U}, \boldsymbol{V})$ has Lipschitz constant on any bounded set.*

Now we present our proof of Theorem 5 by verifying that all the conditions list above are satisfied in our Algorithm 1.

Proof. Conditions 1 and 2 naturally hold in our algorithm by the updating rule. According to the non-ascending property, $F(\boldsymbol{U}^k, \boldsymbol{V}^k)$ is bounded. Meanwhile, we have

$$F(\boldsymbol{U}^k, \boldsymbol{V}^k) \ge \sum_{i=1}^{d} g_1(\|\boldsymbol{U}_{\cdot i}^k\|_2) + \sum_{i=1}^{d} g_2(\|\boldsymbol{V}_{\cdot i}^k\|_2)$$

Therefore Condition 3 holds for l_p $(0 \le p \le 1)$ and log function. For MC in our Algorithm 1, $f(\boldsymbol{U}, \boldsymbol{V})$ is a semi-algebraic function. Based on Proposition 6, Condition 4 holds. As $f(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2}\|P_\Omega(\boldsymbol{U}\boldsymbol{V}^T) - M\|_F^2$ where P_Ω is a bounded linear operator, f is naturally gradient Lipschitz continuous and Conditions 5 and 6 hold. The proof is completed. □