J. Appl. Numer. Optim. 3 (2021), No. 1, pp. 115-150
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.3.2021.1.08

EXTENDED GAUSS-NEWTON AND ADMM-GAUSS-NEWTON ALGORITHMS FOR LOW-RANK MATRIX OPTIMIZATION

QUOC TRAN-DINH

Department of Statistics and Operations Research
The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

Abstract. In this paper, we develop a variant of the well-known Gauss-Newton (GN) method to solve a class of nonconvex optimization problems involving low-rank matrix variables. As opposed to standard GN method, our algorithm allows one to handle general smooth convex objective function. We show, under mild conditions, that the proposed algorithm globally and locally converges to a stationary point of the original problem. We also show empirically that our GN algorithm achieves higher accurate solutions than the alternating minimization algorithm (AMA). Then, we specify our GN scheme to handle the symmetric case and prove its convergence, where AMA is not applicable. Next, we incorporate our GN scheme into an alternating direction method of multipliers (ADMM) to develop a new variant, called ADMM-GN. We prove that, under mild conditions and a proper choice of the penalty parameter, our ADMM-GN globally converges to a stationary point of the original problem. Finally, we provide several numerical experiments to illustrate the proposed algorithms. Our results show that the new algorithms have encouraging performance compared to existing state-of-the-art methods.

Keywords. Low-rank approximation, Gauss-Newton method, nonconvex alternating direction method of multipliers, linear and quadratic convergence rates.

1. INTRODUCTION

Problem statement. In this paper, we consider the following class of low-rank matrix nonconvex optimization problems:

$$\Phi^\star := \min_{U,V}\left\{\Phi(U,V) := \phi\left(\mathscr{A}(UV^\top) - B\right) + \mathscr{R}(U,V) \ : \ U \in \mathbb{R}^{m\times r},\ V \in \mathbb{R}^{n\times r}\right\}, \tag{1.1}$$

where $\mathscr{A}(Z) := [\mathrm{trace}\left(A_1^\top Z\right), \mathrm{trace}\left(A_2^\top Z\right), \cdots, \mathrm{trace}\left(A_l^\top Z\right)]$ for l matrices $A_1, \cdots, A_l$ in $\mathbb{R}^{m\times n}$ is a linear operator; $\phi : \mathbb{R}^l \to \mathbb{R}\cup\{+\infty\}$ is a proper, closed, and convex function; and $B \in \mathbb{R}^l$ is a vector of observations. The function $\mathscr{R}$ is often referred to as a regularizer, which can be chosen as $\mathscr{R}(U,V) := \frac{1}{4}\|U^\top U - V^\top V\|_F^2$ as suggested in [1]. Clearly, (1.1) is nonconvex due to the bilinear term $UV^\top$. Hence, it is NP-hard [2], and numerical methods for solving (1.1) often aim at obtaining a local optimum or a stationary point of (1.1). In this paper, we are interested in the low-rank case, where $r \ll \min\{m,n\}$.

E-mail address: quoctd@email.unc.edu.

Received November 9, 2020; Accepted February 27, 2021.

Problem (1.1) covers various practical models in low-rank embedded problems, function learning, matrix completion in recommender systems, inpainting and compression in image processing, robust principal component analysis in statistics, and semidefinite programming relaxations in combinatorial optimization, see, e.g., [3, 4, 5, 6, 7, 8, 9]. Among these applications, the following problems have been recently attracted a great attention. The most common case is when $\phi(\cdot) := (1/2)\|\cdot\|_2^2$, where (1.1) becomes a least-squares low-rank approximation problem in compressive sensing (see, e.g., [7]):

$$\min_{U,V}\left\{(1/2)\|\mathscr{A}(UV^\top)-B\|_2^2 \ : \ U\in\mathbb{R}^{m\times r},\ V\in\mathbb{R}^{n\times r}\right\}. \tag{1.2}$$

Here, the linear operator $\mathscr{A}$ is often assumed to satisfy a restricted isometric property (RIP) [10] that allows us to recover an exact solution from a few number of observations in B. In particular, if $\mathscr{A}=\mathscr{P}_\Omega$, the projection on a given index subset $\Omega\subset\{1,2,\cdots,m\}\times\{1,2,\cdots,n\}$, then (1.2) covers the matrix completion model:

$$\min_{U,V}\left\{(1/2)\|\mathscr{P}_\Omega(UV^\top)-B_\Omega\|_F^2 \ : \ U\in\mathbb{R}^{m\times r},\ V\in\mathbb{R}^{n\times r}\right\}, \tag{1.3}$$

where B_Ω is the observed entries in Ω. If $\mathscr{A}$ is an identity operator and $B\in\mathbb{R}^{m\times n}$ is given, then (1.2) becomes a low-rank matrix factorization problem

$$\Phi^\star := \min_{U,V}\left\{\Phi(U,V)=(1/2)\|UV^\top-B\|_F^2 \ : \ U\in\mathbb{R}^{m\times r},\ V\in\mathbb{R}^{n\times r}\right\}. \tag{1.4}$$

Especially, if $U=V$ and B is symmetric positive definite, then (1.4) reduces to

$$\Phi^\star := \min_{U}\left\{\Phi(U) := (1/2)\|UU^\top-B\|_F^2 \ : \ U\in\mathbb{R}^{n\times r}\right\}, \tag{1.5}$$

which is studied in [8]. Alternatively, if we choose $\Phi(U) := (1/2)\|\mathscr{A}(UU^\top)-B\|_F^2$ in (1.2), then (1.1) reduces to the case investigated in [11]. While both special cases (1.4) and (1.5) possess a closed form solution via a truncated SVD and an eigenvalue decomposition, respectively, GN methods can also be applied to solve these problems. The authors in [8] demonstrate the advantages of a GN method for solving (1.5) with significantly encouraging performance.

Related work. The low-rank structure is key to recast many existing problems into new frameworks or to design new models by means of regularizers to promote solution structures in various applications such as matrix completion (MC) [3], robust principal component analysis (RPCA) [12], and their variants. Hitherto, extensions to group structured sparsity, low-rankness, tree models, and tensor representations have attracted a great attention in recent years, see, e.g., [13, 14, 15, 16, 17, 18, 19]. A majority of research for low-rank models focuses on estimating sample complexity results for specific instances of (1.1), while numerous recent papers revolve around the RPCA setting, MC, and their extensions [3, 12, 20].

Together with modeling, solution methods have also been extensively developed for solving concrete instances of (1.1) in low-rank matrix completion and recovery applications. Among various approaches, convex optimization is perhaps one of the most powerful tools to solve several instances of (1.1), including MC, RPCA, their variants, and extensions. Unfortunately, convex models only provide an approximation to the low-rank model (1.1) by convex relaxations using, e.g., nuclear or max norms, which may not adequately approximate the desired rank. Alternatively, nonconvex as well as discrete optimization methods have been also considered for solving (1.1), see, e.g., [7, 9, 21, 22, 23, 24]. While these approaches work directly on the original problem (1.1), they can only find a local optimum or a critical point, and strongly depend on the priori knowledge of problems, the initial point of algorithm, and predicted ranks. However, recent empirical evidence has been provided to support these approaches, and surprisingly, in many cases, they outperform the convex optimization approach in terms of "accuracy" to the original model, and the overall computational time [7, 9, 22]. Other approaches such as stochastic gradient descent, Riemannian manifold-based, greedy methods, parallel and distributed algorithms have also recently been studied for solving (1.1), see, e.g., [20, 24, 25, 26, 27].

Motivation. Gauss-Newton (GN) methods work extremely well for nonlinear least-squares problems [28]. When ϕ is quadratic and the residual term $\mathscr{A}(UV^\top) - B$ in (1.1) is small or zero at solutions, they can achieve a local superlinear and even a quadratic convergence rate. With a "good" initial point (i.e., close to the set of stationary points), GN methods often reach a highly accurate stationary point within a few iterations [29]. Such a "good" initial point can be obtained using priori knowledge of the problem and the underlying algorithm (e.g., steady states of dynamical systems, or previous iterations of the algorithm) as a warm-start strategy.

As in classical GN methods, we develop an iterative scheme for solving (1.1) by using a linearization of $\mathscr{A}(UV^\top) - B$ and a quadratic surrogate of ϕ. At each iteration, it requires to solve a simple convex problem to form a GN direction and then incorporates it with a globalization strategy to update the next iteration. In our setting, computing a GN direction reduces to solving a linear least-squares problem. Compared to the alternating minimization method (AMA) [9] that alternatively solves for each U and V, GN simultaneously solves for U and V using the linearization of $UV^\top$. We have observed that (*cf.* Subsection 6.2) GN uses a linearization of $UV^\top$ to provide a good local approximation to $UV^\top$ compared to the alternating form $U\bar{V}^\top$ (or $\bar{U}V^\top$), when $U - \bar{U}$ (or $V - \bar{V}$) is relatively large. This makes AMA saturated and does not significantly improve the objective values. In addition, without regularization, AMA may fail to converge as indicated by a counterexample in [30]. Moreover, AMA is not applicable to solving the symmetric case of (1.1) as shown in Section 5, but our GN method is.

While GN methods are often applied to solve nonlinear least-squares problems [31], they have not been widely exploited for matrix optimization. Our aim in this paper is to extend the GN method for solving a class of problems (1.1) with a general smooth convex objective function ϕ and low-rank matrix variables. This paper is also inspired by a recent work [8], where the authors proposed a simple symmetric GN scheme to solve (1.5) and demonstrated its encouraging performance on various numerical examples.

Contribution. Our contribution in this paper can be summarized as follows:

(a) We extend the classical GN method to solve the low-rank matrix optimization problem (1.1) with smooth convex objective function ϕ. We prove the existence of a GN direction and provide a closed form formulation to compute it. We empirically show that our GN method can achieve higher accurate solutions than the well-known AMA scheme within the same number of iterations in certain cases.

(b) We show that there exists an explicit step-size to guarantee a descent property of the GN direction, which allows us to perform a backtracking linesearch procedure to find an appropriate step-size. We specify our framework to the symmetric case. Under mild conditions, we prove a global convergence of the proposed methods.

(c) We prove local linear and quadratic convergence rates of the full-step GN variant under standard assumptions imposed on (1.1) at its solution set.

(d) We combine an alternating direction method of multipliers (ADMM) and the proposed GN method to obtain a new algorithm for handling (1.1). Under standard assumptions on (1.1), we prove a global convergence of the new algorithm.

Unlike AMA which only achieves a sublinear convergence rate even with a good initial point, GN methods may require additional computation for GN directions, but they can achieve a fast local linear or quadratic convergence rate, which is key for online and real-time implementations by using warm-start. Alternatively, gradient descent-based methods can achieve local linear convergence but often require much stronger assumptions imposed on (1.1). In contrast, GN methods work with the "small residual" setting under mild assumptions, and can easily achieve high accuracy solutions within a small number of iterations.

Paper outline. The rest of this paper is organized as follows. We first review basic concepts related to problem (1.1) in Section 2. Section 3 presents a linesearch GN method for solving (1.1) and its convergence guarantees. Section 4 develops an ADMM-GN algorithm to solve (1.1) and investigates its global convergence. Section 5 specifies the GN algorithm to the symmetric case and proves its convergence. Section 6.1 discusses some implementation aspects of our algorithms and their extension to the nonsmooth objective function setting. Numerical experiments are conducted in Section 6 with several examples from different fields. For the sake of presentation, we defer all the technical proofs in the main text to the appendix.

2. Background and optimality condition

We briefly describe basic notation, the optimality condition of (1.1), and our assumptions.

2.1. Basic notation and concepts.

For a matrix X, $\sigma_{\min}(X)$ and $\sigma_{\max}(X)$ denote its positive smallest and largest singular values, respectively. If X is symmetric, then $\lambda_{\min}(X)$ and $\lambda_{\max}(X)$ denote its smallest and largest eigenvalues, respectively. We use $X = P\Sigma Q^{\top}$ for singular value decomposition (SVD) and $X = U\Lambda U^{-1}$ for eigenvalue decomposition. $X^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of X. When X is full-column rank, $X^{\dagger} = (X^{\top}X)^{-1}X^{\top}$. We define $P_X := XX^{\dagger}$ the projection onto the range space of X, and $P_X^{\perp} := \mathbb{I} - P_X$ the orthogonal projection of P_X, i.e., $P_X P_X^{\perp} = P_X^{\perp}P_X = 0$, where $\mathbb{I}$ is the identity matrix. Clearly, $P_X^{\perp}X = 0$. We define $\mathrm{vec}\,(X) = (X_{11}, \cdots, X_{m1}, \cdots, X_{1n}, \cdots, X_{mn})^{\top}$ the vectorization of X, and mat the inverse mapping of vec, i.e., $\mathrm{mat}\,(\mathrm{vec}\,(X)) = X$. $X \otimes Y$ denotes the Kronecker product of X and Y. We have $\mathrm{vec}\,(AXB) = (B^{\top} \otimes A)\mathrm{vec}\,(X)$ and $(A \otimes B)(C \otimes D) = AC \otimes BD$. $\mathscr{A}^*$ denotes the adjoint of a linear operator $\mathscr{A}$. We say that a continuously differentiable function f is L_f-smooth if there exists a constant $L_f \in [0, +\infty)$ such that $\|\nabla f(x) - \nabla f(y)\|_2 \leq L_f\|x - y\|_2$ for all $x, y \in \mathrm{dom}(f)$. Here, L_f is called the Lipschitz constant of f. A function f is said to be μ_f-strongly convex if $f(\cdot) - \frac{\mu_f}{2}\|\cdot\|_2^2$ remains convex, where $\mu_f \geq 0$. If $\mu_f = 0$, then f is just convex. For a convex function f, we use $\mathrm{dom}(f)$ to denote its domain and ∂f to denote its subdifferential.

2.2. Optimality condition and basic assumptions.

We define $X := [U, V]$ as the joint variable of U and V. We assume that ϕ in (1.1) is smooth. The *optimality condition* of (1.1) can be written as follows:

$$\begin{cases} U_{\star}^{\top}\mathscr{A}^*\left(\nabla\phi(\mathscr{A}(U_{\star}V_{\star}^{\top}) - B)\right) &= 0,\\ \mathscr{A}^*\left(\nabla\phi(\mathscr{A}(U_{\star}V_{\star}^{\top}) - B)\right)V_{\star} &= 0. \end{cases} \tag{2.1}$$

Any $X_\star = [U_\star, V_\star]$ satisfying (2.1) is called a *stationary point* of (1.1). We denote by $\mathscr{X}_\star$ the set of stationary points of (1.1). Since $r \leq \min\{m,n\}$, the solution of (2.1) is generally nonunique. Our aim is to design algorithms for generating a sequence $\{X_k\}$ converging to $X_\star \in \mathscr{X}_\star$ under the following assumptions.

Assumption 2.1. Problem (1.1) satisfies the following conditions:

(a) the objective function Φ of (1.1) is coercive, i.e., $\lim_{\|(U,V)\| \to +\infty} \Phi(U,V) = +\infty$;
(b) ϕ is L_ϕ-smooth and μ_ϕ-convex with $0 \leq \mu_\phi \leq L_\phi < +\infty$.

Under Assumption A.2.1(a), the set of optimal solutions, and therefore the set $\mathscr{X}_\star$ of stationary points, of (1.1) is nonempty and bounded. Moreover, $\Phi^\star > -\infty$ and the sub-level set $\mathscr{F}_\Phi(\beta) := \{[U,V] \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} : \Phi(U,V) \leq \beta\}$ of Φ is nonempty and bounded for $\beta \geq \Phi^\star$.

We allow $\mu_\phi = 0$, which also covers the non-strongly convex case. Since ϕ is smooth and $\mathscr{A}$ is linear, Φ in (1.1) is also smooth. Moreover, as shown in [32], ϕ satisfies

$$\frac{\mu_\phi}{2}\|y-x\|_2^2 \leq \phi(y) - \phi(x) - \langle \nabla\phi(x), y-x\rangle \leq \frac{L_\phi}{2}\|y-x\|_2^2, \quad \forall x,y \in \mathrm{dom}(\phi). \tag{2.2}$$

Note that Assumption A.2.1(b) covers a wide range of applications, including logistic loss, Huber loss, and entropy function in statistics and machine learning [33]. We can also extend our results in this paper to the case ϕ is nonconvex, but μ_ϕ-weakly convex with $\mu_\phi < 0$. If ϕ is L_ϕ-smooth and nonconvex, then ϕ is already weakly convex with $\mu_\phi = -L_\phi$. However, to avoid overloading this paper, we omit this extension in this paper.

3. Linesearch Gauss-Newton method

In this section, we develop a linesearch Gauss-Newton (Ls-GN) algorithm for solving (1.1).

3.1. **Forming a surrogate of the objective.** By Assumption A.2.1, we can upper bound the objective function $\Phi(U,V) := \phi(\mathscr{A}(UV^\top) - B)$ of (1.1) by a quadratic surrogate as follows.

Lemma 3.1. *Under Assumption A.2.1(b), for any U, V, $\hat{U}$, and $\hat{V}$, we have*

$$\Phi(\hat{U},\hat{V}) \leq \Phi(U,V) + \frac{L_\Phi}{2}\|\hat{U}\hat{V}^\top - (UV^\top - L_\Phi^{-1}\Phi'(UV^\top))\|_F^2 - \frac{1}{2L_\Phi}\|\Phi'(UV^\top)\|_F^2, \tag{3.1}$$

where $\Phi'(\cdot) := \mathscr{A}^*(\nabla\phi(\mathscr{A}(\cdot) - B))$ is the gradient of the composition $\phi(\mathscr{A}(\cdot) - B)$, and $L_\Phi := L_\phi\|\mathscr{A}\|^2$ is the Lipschitz constant of the gradient of $\phi(\mathscr{A}(\cdot) - B)$.

Proof. Since $\mathscr{A}$ is linear, using (2.2) with $y := \mathscr{A}(\hat{U}\hat{V}^\top) - B$ and $x := \mathscr{A}(UV^\top) - B$, we have

$$\begin{aligned}
\Phi(\hat{U},\hat{V}) &= \phi(\mathscr{A}(\hat{U}\hat{V}^\top) - B) \\
&\overset{(2.2)}{\leq} \phi(\mathscr{A}(UV^\top) - B) + \langle \nabla\phi(\mathscr{A}(UV^\top) - B), \mathscr{A}(\hat{U}\hat{V}^\top - UV^\top)\rangle \\
&\quad + \tfrac{L_\phi}{2}\|\mathscr{A}(\hat{U}\hat{V}^\top - UV^\top)\|_2^2 \\
&\leq \Phi(U,V) + \langle \Phi'(UV^\top), \hat{U}\hat{V}^\top - UV^\top\rangle + \tfrac{L_\Phi}{2}\|\hat{U}\hat{V}^\top - UV^\top\|_F^2 \\
&= \Phi(U,V) + \tfrac{L_\Phi}{2}\|\hat{U}\hat{V}^\top - (UV^\top - L_\Phi^{-1}\Phi'(UV^\top))\|_F^2 - \tfrac{1}{2L_\Phi}\|\Phi'(UV^\top)\|_F^2,
\end{aligned}$$

which proves (3.1). Here, we have used $\|\mathscr{A}(\hat{U}\hat{V}^\top - UV^\top)\|_2^2 \leq \|\mathscr{A}\|^2\|\hat{U}\hat{V}^\top - UV^\top\|_F^2$ in the second inequality, and $\langle u,v\rangle + \frac{L_\Phi}{2}\|v\|^2 = \frac{L_\Phi}{2}\|u-v\|^2 - \frac{1}{2L_\Phi}\|u\|^2$ in the last equality. □

Gradient descent-type methods rely on finding a descent direction of Φ by approximately minimizing the right-hand side surrogate of Φ in (3.1). Unfortunately, this surrogate remains nonconvex due to the bilinear term $\hat{U}\hat{V}^{\top}$. Our next step is to linearize this term around a given point $[U,V]$ as follows:

$$\hat{U}\hat{V}^{\top} \approx UV^{\top} + U(\hat{V}-V)^{\top} + (\hat{U}-U)V^{\top}. \tag{3.2}$$

Then, the minimization of the right-hand side of (3.1) is approximated by

$$\min_{\hat{U},\hat{V}} \left\{ (1/2)\|U(\hat{V}-V)^{\top} + (\hat{U}-U)V^{\top} + L_{\Phi}^{-1}\Phi'(UV^{\top})\|_F^2 \right\}. \tag{3.3}$$

This is a linear least-squares problem, and can be solved by standard linear algebra routines.

3.2. **Computing Gauss-Newton direction.** Let us recall the three derivatives: $\nabla\phi(\cdot)$ is the gradient of ϕ, $\Phi'(\cdot) := \mathscr{A}^*(\nabla\phi(\mathscr{A}(\cdot)-B))$ is the gradient of the composition function $\phi(\mathscr{A}(\cdot)-B)$, and $\nabla\Phi(U,V) = [\Phi'(UV^{\top})V,\ U^{\top}\Phi'(UV^{\top})]$ is the gradient of Φ w.r.t. $[U,V]$, which will be frequently used in the sequel. We also define

$$D_U := \hat{U}-U, \quad D_V := \hat{V}-V, \quad \text{and} \quad Z := -L_{\Phi}^{-1}\mathscr{A}^*\left(\nabla\phi(\mathscr{A}(UV^{\top})-B)\right).$$

Then, we rewrite (3.3) as

$$\min_{D_U,D_V} \left\{ (1/2)\|UD_V^{\top} + D_UV^{\top} - Z\|_F^2 \ :\ D_U \in \mathbb{R}^{m\times r}, D_V \in \mathbb{R}^{m\times r} \right\}. \tag{3.4}$$

The optimality condition of (3.4) becomes

$$\begin{cases} U^{\top}UD_V^{\top} + U^{\top}D_UV^{\top} &= U^{\top}Z, \\ UD_V^{\top}V + D_UV^{\top}V &= ZV. \end{cases} \tag{3.5}$$

As usual, we refer to (3.5) as the normal equation of (3.4). We will construct a closed form solution of (3.5) in Lemma 3.2, whose proof is given in Appendix A.1.

Lemma 3.2. *The rank of the square linear system* (3.5) *does not exceed* $r(m+n-r)$*. In addition,* (3.5) *has a solution. If* $\mathrm{rank}(U) = \mathrm{rank}(V) = r \le \min\{m,n\}$*, then the solution of* (3.5) *is given explicitly by*

$$\begin{cases} D_U &= P_U^{\perp}Z(V^{\dagger})^{\top} + U\hat{D}_r, \\ D_V^{\top} &= U^{\dagger}Z - \hat{D}_rV^{\top}, \end{cases} \tag{3.6}$$

which forms a linear subspace in $\mathbb{R}^{r\times r}$*, and* $\hat{D}_r \in \mathbb{R}^{r\times r}$ *is an arbitrary matrix.*

In particular, if we choose $\hat{D}_r := 0.5U^{\dagger}Z(V^{\dagger})^{\top} \in \mathbb{R}^{r\times r}$*, then*

$$D_U = \left(\mathbb{I}_m - 0.5P_U\right)Z(V^{\dagger})^{\top} \quad \text{and} \quad D_V^{\top} = U^{\dagger}Z\left(\mathbb{I}_n - 0.5P_V\right). \tag{3.7}$$

Moreover, the optimal value of (3.4) *is* $(1/2)\|P_U^{\perp}ZP_V^{\perp}\|_F^2$.

Lemma 3.2 also shows that if either Z is in the null space of P_U or $Z^{\top}$ is in the null space of P_V, then $\|P_U^{\perp}ZP_V^{\perp}\|_F^2 = 0$. Since (3.7) only gives us one choice for $D_X := [D_U, D_V]$, if $\hat{D}_r = 0^r$, we obtain another simple GN search direction.

Remark 3.1. Let $m = n$. If we assume that $U = V$, then $D_U = D_V$ and

$$D_U = P_U^{\perp}Z(U^{\dagger})^{\top} + U\hat{D}_r, \quad \text{where} \quad \hat{D}_r \in \mathscr{S}_r := \left\{\hat{D}_r \in \mathbb{R}^{r\times r} \ :\ \hat{D}_r + \hat{D}_r^{\top} = U^{\dagger}Z(U^{\dagger})^{\top}\right\}.$$

Clearly, $\mathscr{S}_r$ is a linear subspace, and its dimension is $r(r+1)/2$.

3.3. **The damped-step Gauss-Newton scheme.** Using Lemma 3.2, we can form a damped step GN scheme as follows:

$$\begin{cases} U_+ & := & U + \alpha D_U, \\ V_+ & := & V + \alpha D_V, \end{cases} \tag{3.8}$$

where D_U and D_V defined in (3.7) is a GN direction, and $\alpha > 0$ is a given step-size determined in the next lemma.

Since the GN direction computed from (3.4) is not unique, we need to choose an appropriate D_X such that it is a descent direction of Φ at X. We prove in Lemma 3.3 that (3.7) indeed gives a descent direction of Φ at X. The proof of this lemma is deferred to Appendix A.2.

Lemma 3.3. *Let $X := [U,V]$ be a non-stationary point of* (1.1) *and $D_X := [D_U, D_V]$ be given by* (3.7)*. If $D_X \neq 0$ and α is chosen in $0 < \alpha \leq \underline{\alpha}$ where*

$$\underline{\alpha} := \min\left\{1, \frac{L_\Phi \sigma_{\min}^3}{2\|\nabla\Phi(U,V)\|_F}, \frac{3\sigma_{\min}^4}{32\sigma_{\max}^2\|\Phi'(UV^\top)\|_F}\right\} \in (0,1], \tag{3.9}$$

then we have

$$\Phi(U_+,V_+) \leq \Phi(U,V) - \frac{\alpha\sigma_{\min}^2}{128 L_\Phi \sigma_{\max}^4}\|\nabla\Phi(U,V)\|^2, \tag{3.10}$$

where $\Phi'(\cdot) = \mathscr{A}^*(\nabla\phi(\mathscr{A}(\cdot) - B))$, $L_\Phi := L_\phi\|\mathscr{A}\|^2$, $\sigma_{\min} := \min\{\sigma_{\min}(U), \sigma_{\min}(V)\}$, *and* $\sigma_{\max} := \max\{\sigma_{\max}(U), \sigma_{\max}(V)\}$. *Hence, D_X is a descent direction of Φ.*

Lemma 3.3 shows that if the residual term $\mathscr{A}^*(\nabla\phi(\mathscr{A}(UV^\top) - B))$ is sufficient small near $\mathscr{X}_\star$, then we obtain a full-step size $\alpha = 1$.

The existence of the GN direction in Lemma 3.2 requires U and V to be full-rank. We prove in Appendix A.3 the following lemma.

Lemma 3.4. *If* $\mathrm{rank}(U) = \mathrm{rank}(V) = r$*, then $X_+ := [U_+, V_+]$ updated by* (3.8) *using the step-size $\underline{\alpha}$ in* (3.9) *satisfies*

$$\sigma_{\min}(U_+) \geq 0.5\sigma_{\min}(U) \quad \textit{and} \quad \sigma_{\min}(V_+) \geq 0.5\sigma_{\min}(V). \tag{3.11}$$

Hence, (3.8) *preserves the rank of U_+ and V_+, i.e.,* $\mathrm{rank}(U_+) = \mathrm{rank}(V_+) = r$.

3.4. **The algorithmic template and its global convergence.** Theoretically, we can use the step-size $\underline{\alpha}$ in Lemma 3.3 for (3.8). However, in practice, computing $\underline{\alpha}$ requires a high computational cost. We instead incorporate the GN scheme (3.8) with an Armijo's backtracking linesearch strategy to find an appropriate step-size $\alpha \geq \beta\underline{\alpha}$ for a given $\beta \in (0,1)$.

Find the smallest integer number $i_k \geq 0$ such that $\alpha := \beta^{i_k}\alpha_0 \geq \underline{\alpha}$ and

$$\Phi(U + \alpha D_U, V + \alpha D_V) \leq \Phi(U,V) - 0.5c_1\alpha\|\nabla\Phi(U,V)\|_F^2, \tag{3.12}$$

where $\alpha_0 > 0$, $c_1 > 0$, and $\beta \in (0,1)$ are given (e.g., $c_1 := 0.5$ and $\beta := \frac{\sqrt{5}-1}{\sqrt{5}+1}$).

By Lemma 3.3, this procedure is terminated after a finite number of iterations i_k such that

$$0 \leq i_k \leq \lfloor \log_\beta(\underline{\alpha}/\alpha_0) \rfloor + 1, \tag{3.13}$$

where $\underline{\alpha}$ is given by (3.9). Now, we describe the complete linesearch GN algorithm for approximating a stationary point of (1.1) as in Algorithm 1.

Algorithm 1 (*Linesearch Gauss-Newton Algorithm* (Ls-GN))

```
1: Initialization: Given a tolerance $\varepsilon > 0$. Choose $X_0 := [U_0, V_0]$. Set $c_1 := 0.5$ and $\alpha_0 := 1$.
2: for $k = 0$ to $k_{\max}$ do
3:     GN direction: Let $Z_k := -L_{\Phi}^{-1}\Phi'(U_kV_k^{\top})$. Compute $D_{X_k} := [D_{U_k}, D_{V_k}]$:
           $D_{U_k} := (\mathbb{I}_m - 0.5P_{U_k})Z_k(V_k^{\dagger})^{\top}$ and $D_{V_k} = (\mathbb{I}_n - 0.5P_{V_k})Z_k^{\top}(U_k^{\dagger})^{\top}$.
4:     Stopping criterion: If stopping_criterion, then TERMINATE.
5:     Backtracking linesearch: Find the smallest integer number $i_k \geq 0$ such that
           $\Phi(U_k + \alpha_k D_{U_k}, V_k + \alpha_k D_{V_k}) \leq \Phi(U_k, V_k) - 0.5c_1\alpha_k\|\nabla\Phi(U_k, V_k)\|_F^2$,
       where $\alpha_k := \alpha_0\beta^{i_k}$.
6:     Update $X_{k+1} := [U_{k+1}, V_{k+1}]$ as $U_{k+1} := U_k + \alpha_k D_{U_k}$ and $V_{k+1} := V_k + \alpha_k D_{V_k}$.
7: end for
```

Per-iteration complexity. The main steps of Algorithm 1 are Steps 3 and 5, i.e. computing D_{X_k} and performing the linesearch routine, respectively.

(a) Computing D_{X_k} requires two inverses $(U^{\top}U)^{-1}$ and $(V^{\top}V)^{-1}$ of the size $r \times r$, and two matrix-matrix multiplications (of the size $m \times r$ or $n \times r$).

(b) Evaluating $\Phi'(\cdot)$ requires one matrix-matrix multiplication $UV^{\top}$ and one evaluation of the form $\mathscr{A}^*(\nabla\phi(\mathscr{A}(\cdot) - B))$. When $\mathscr{A}$ is a subset projection $\mathscr{P}_{\Omega}$ (e.g., in matrix completion), we can compute $(UV^{\top})_{(i,j)\in\Omega}$ instead of the full matrix $UV^{\top}$.

(c) Each step of the linesearch needs one matrix-matrix multiplication $UV^{\top}$ and one evaluation of Φ. It requires at most $\lfloor \log_{\beta}(\underline{\alpha}/\alpha_0)\rfloor + 1$ linesearch iterations. However, we observe that i_k often varies from 1 to 2 on average in our experiments in Section 6.

Global convergence. Since (1.1) is nonconvex, we only expect $\{X_k\}$ generated by Algorithm 1 to converge to a stationary point $X_{\star} \in \mathscr{X}_{\star}$. However, Lemma 3.4 only guarantees the full-rankness of U_k and V_k at each iteration, but we may have $\lim_{k\to\infty}\sigma_{\min}(U_k) = 0$ or $\lim_{k\to\infty}\sigma_{\min}(V_k) = 0$. In order to prove a global convergence of Algorithm 1, we require one additional condition: There exists $\underline{\sigma} > 0$ such that:

$$\sigma_{\min}(U_k) \geq \underline{\sigma} \quad \text{and} \quad \sigma_{\min}(V_k) \geq \underline{\sigma} \quad \text{for all } k \geq 0. \tag{3.14}$$

Under Assumption A.2.1(a), the following sublevel set of Φ:

$$\mathscr{F}_{\Phi}(\beta) := \left\{[U,V] \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} \ : \ \Phi(U,V) \leq \beta\right\}$$

is bounded for a given $\gamma > 0$. We prove in Appendix A.4 a global convergence of Algorithm 1 stated in the following theorem.

Theorem 3.1. *Let $\{X_k\}$ with $X_k := [U_k, V_k]$ be generated by Algorithm 1. Then, under Assumption A.2.1, we have*

$$\sum_{k=0}^{\infty}\alpha_k\|\nabla\Phi(U_k,V_k)\|_F^2 < +\infty, \ \textit{and} \ \lim_{k\to\infty}\alpha_k\|\nabla\Phi(U_k,V_k)\|_F = 0. \tag{3.15}$$

If, in addition, the condition (3.14) holds, then

$$\lim_{k\to\infty}\|\nabla\Phi(U_k,V_k)\|_F = 0. \tag{3.16}$$

There exists a limit point $X_\star$ of $\{X_k\}$, and any limit point $X_\star$ is in $\mathscr{X}_\star$.

3.5. **Local linear convergence without strong convexity.** We prove a local convergence of the full-step Gauss-Newton scheme (3.8) when $\alpha = 1$. Generally, problem (1.1) does not satisfy the regularity assumption: the Jacobian $J_R(X) = A[V \otimes \mathbb{I}_m, \mathbb{I}_n \otimes U] \in \mathbb{R}^{l \times (m+n)r}$ of the objective residual $R(X) := \mathscr{A}(UV^\top) - B$ in (1.1) is not full-column rank, where A is the matrix form of the linear operator $\mathscr{A}$. However, we can still guarantee a fast local convergence under the following conditions:

Assumption 3.1. Problem (1.1) satisfies the following conditions:

(a) ϕ is twice continuously differentiable on a neighborhood $\mathscr{N}(Z_\star)$ of $Z_\star = \mathscr{A}(U_\star V_\star^\top) - B$, and its Hessian $\nabla^2\phi$ is Lipschitz continuous in $\mathscr{N}(Z_\star)$ with the constant $L_{\phi''}$.

(b) The Hessian $\nabla^2\Phi(X_\star)$ of $\Phi(X) := \phi(\mathscr{A}(UV^\top) - B)$ at $X_\star \in \mathscr{X}_\star$ satisfies

$$\left\| \left[\mathbb{I} - L_\Phi^{-1} H(X)^\dagger \nabla^2\Phi(X_\star)\right](X - X_\star)\right\|_F \le \kappa(X_\star)\|X - X_\star\|_F, \quad \forall X \in \mathscr{N}(X_\star), \tag{3.17}$$

where $H(X) := \begin{bmatrix} V^\top \otimes U & V^\top V \otimes \mathbb{I}_m \\ \mathbb{I}_n \otimes U^\top U & V \otimes U^\top \end{bmatrix}$, $L := L_\phi \|\mathscr{A}\|^2$, and $0 \le \kappa(X_\star) \le \bar{\kappa} < 1$.

Assumption A.3.1(b) looks technical, but relates to a "small residual condition" in standard GN methods, see, e.g., [31]. For instance, if $\phi(\cdot) = (1/2)\|\cdot\|_2^2$, and $\mathscr{A} = \mathbb{I}$, the identity operator, then the residual term becomes $R(X) = UV^\top - B$, and $\Phi(X) = (1/2)\|R(X)\|_F^2$. In this case, condition (3.17) holds if $\|R(X_\star)\|_F \le \kappa(X_\star) < 1$ (i.e., we have a "small residual" case).

Now, we prove in Appendix A.7 a local convergence of the full-step GN variant.

Theorem 3.2. *Let $\{X_k\}$ be generated by (3.8) with a full step-size $\alpha_k = 1$, and $X_\star := [U_\star, V_\star] \in \mathscr{X}_\star$ be a given stationary point of (1.1) such that* $\mathrm{rank}(U_\star) = \mathrm{rank}(V_\star) = r$*. Assume that Assumptions A.2.1 and A.3.1 hold. Then, there exists a neighborhood $\mathscr{N}(X_\star)$ of $X_\star$ and a constant $K_1 > 0$ independent of X_k such that*

$$\|X_{k+1} - X_\star\|_F \le \left(\bar{\kappa} + 0.5K_1\|X_k - X_\star\|_F\right)\|X_k - X_\star\|_F, \quad \forall X_k \in \mathscr{N}(X_\star). \tag{3.18}$$

Consequently, if $H(X_\star)^\dagger\nabla^2\Phi(X_\star) = L_\Phi\mathbb{I}$ in (3.17) (i.e., zero residual), then there exists a constant $K_2 > K_1$ such that the sequence $\{X_k\}$ generated by our full-step GN algorithm starting from $X_0 \in \mathscr{N}(X_\star)$ with $\|X_0 - X_\star\|_F < 2K_2^{-1}$ quadratically converges to $X_\star \in \mathscr{X}_\star$.

If $\kappa(X_\star) \in (0,1)$ in (3.17) (i.e., small residual), then, for any $X_0 \in \mathscr{N}(X_\star)$ such that $\|X_0 - X_\star\|_F \le \bar{r}_0 < 2K_1^{-1}(1 - \bar{\kappa})$, $\{X_k\}$ linearly converges to $X_\star$.

4. ADMM-Gauss-Newton Algorithm

The GN method only works well and has a fast local convergence for the "small residual" case. In general, it may converge very slowly or even fails to converge. In this section, we propose to combine the GN scheme (3.8) and the alternating direction method of multipliers (ADMM) to develop a new algorithm for solving (1.1) called ADMM-GN. The ADMM can be viewed as a variant of augmented Lagrangian-based methods in nonlinear optimization [34, 35, 36]. It can also be derived from Douglas-Rachford's method in convex optimization.

4.1. **The augmented Lagrangian function and ADMM scheme.** We introduce $W = \mathscr{A}(UV^\top) - B$ and rewrite (1.1) as the following problem:

$$\Phi_\star := \min_{U,V,W} \left\{ \phi(W) \ : \ \mathscr{A}(UV^\top) - W = B \right\}. \tag{4.1}$$

We can define the augmented Lagrangian function associated with (4.1) as

$$\begin{aligned} \mathscr{L}_\rho(U,V,W,\Lambda) \ &:= \ \phi(W) + \langle \Lambda, \mathscr{A}(UV^\top) - W - B \rangle + \tfrac{\rho}{2}\|\mathscr{A}(UV^\top) - W - B\|_2^2 \\ &= \ \phi(W) + \tfrac{\rho}{2}\|\mathscr{A}(UV^\top) - W - B + \rho^{-1}\Lambda\|_2^2 - \tfrac{1}{2\rho}\|\Lambda\|_2^2, \end{aligned} \tag{4.2}$$

where $\rho > 0$ is a penalty parameter and Λ is a Lagrange multiplier.

Next, we apply the standard ADMM scheme to (4.1) which leads to the following 3 steps:

$$(U_{k+1}, V_{k+1}) := \arg\min_{U,V} \left\{ \|\mathscr{A}(UV^\top) - W_k - B + \rho^{-1}\Lambda_k\|_2^2 \right\}, \tag{4.3a}$$

$$W_{k+1} := \arg\min_{W} \left\{ \phi(W) + (\rho/2)\|W - \big(\mathscr{A}(U_{k+1}V_{k+1}^\top) - B + \rho^{-1}\Lambda_k\big)\|_2^2 \right\}, \tag{4.3b}$$

$$\Lambda_{k+1} := \Lambda_k + \rho(\mathscr{A}(U_{k+1}V_{k+1}^\top) - W_{k+1} - B). \tag{4.3c}$$

Obviously, both subproblems (4.3a) and (4.3b) remain computationally expensive. While (4.3a) is nonconvex, (4.3b) is smooth and convex. Without any further step applying to (4.3), convergence theory for this nonconvex ADMM scheme can be found in several recent papers including [9, 37, 38]. However, (4.3) remains impractical since (4.3a) and (4.3b) cannot be solved with a closed form or a highly accurate solution. We approximately solve these subproblems.

4.2. **Approximation of the alternating steps.** We apply the GN scheme to approximate (4.3a) and a linearization to approximate (4.3b) in our ADMM scheme above.

Gauss-Newton step for the UV-subproblem (4.3a). We first apply on step of (3.8) to solve (4.3a) as follows. We first approximate $\|\mathscr{A}(UV^\top) - W_k - B + \rho^{-1}\Lambda_k\|_2^2$ by using the quadratic surrogate of $\mathscr{A}(\cdot)$ and the linearization $U_kV_k^\top + U_kD_V^\top + D_UV_k^\top$ of $UV^\top$ with $D_U := U - U_k$ and $D_V := V - V_k$ as new variables. By letting $Z_k := -L_{\mathscr{A}}^{-1}\mathscr{A}^*\left(\mathscr{A}(U_kV_k^\top) - W_k - B + \rho^{-1}\Lambda_k\right)$ with $L_{\mathscr{A}} := \|\mathscr{A}\|^2$, we solve

$$[D_{U_k}, D_{V_k}] := \arg\min_{D_U, D_V} \left\{ \mathscr{Q}_k(D_U, D_V) := \frac{1}{2}\|U_kD_V^\top + D_UV_k^\top - Z_k\|_F^2 \right\}. \tag{4.4}$$

Here, the Lipschitz constant $L_{\mathscr{A}} := \|\mathscr{A}\|^2$ can be computed by a power method [39]. Using Lemma 3.2, we can compute $[D_{U_k}, D_{V_k}]$ as

$$\begin{cases} D_{U_k} \ := \ \big(\mathbb{I}_m - 0.5P_{U_k}\big)Z_k(V_k^\dagger)^\top \\ D_{V_k}^\top \ := \ U_k^\dagger Z_k\big(\mathbb{I}_n - 0.5P_{V_k}\big). \end{cases} \tag{4.5}$$

The corresponding objective value is $\mathscr{Q}_k(D_{U_k}, D_{V_k}) := (1/2)\|P_{u_k}^\perp Z_k P_{V_k}^\perp\|_F^2$. Then, we update $X_{k+1} := [U_{k+1}, V_{k+1}]$ as

$$U_{k+1} := U_k + \alpha_k D_{U_k} \quad \text{and} \quad V_{k+1} := V_k + \alpha_k D_{V_k}, \tag{4.6}$$

where $\alpha_k > 0$ is a step-size computed by a linesearch procedure as in (3.12).

Gradient step for the W-subproblem (4.3b)**.** If ϕ does not have a tractably proximal operator (i.e., its proximal operator cannot be computed in a closed form, or with a low-order polynomial-time algorithm), we approximate (4.3b) by using one gradient step as

$$W_{k+1} := \arg\min_{W} \left\{ \frac{L_\phi}{2} \| W - (W_k - L_\phi^{-1} \nabla \phi(W_k)) \|_2^2 + \frac{\rho}{2} \| W - E_k \|_2^2 \right\}, \tag{4.7}$$

where $E_k := \mathscr{A}(U_{k+1}V_{k+1}^\top) - B + \rho^{-1}\Lambda_k$. Solve (4.7) directly, we get

$$W_{k+1} := (\rho + L_\phi)^{-1} \Big(L_\phi W_k - \nabla \phi(W_k) + (\Lambda_k + \rho(\mathscr{A}(U_{k+1}^\top V_{k+1}) - B) \Big). \tag{4.8}$$

4.3. **The ADMM-Gauss-Newton algorithm and its global convergence.** Putting (4.6), (4.3c), and (4.3b) or (4.8) together, we obtain the following ADMM-GN scheme with two options:

$$\begin{cases} Z_k & := -L_{\mathscr{A}}^{-1} \mathscr{A}^* \left(\mathscr{A}(U_k V_k^\top) - W_k - B + \rho^{-1}\Lambda_k \right), \\ U_{k+1} & := U_k + \alpha_k \left(\mathbb{I}_m - 0.5 P_{U_k} \right) Z_k (V_k^\dagger)^\top, \\ V_{k+1} & := V_k + \alpha_k \left(\mathbb{I}_n - 0.5 P_{V_k} \right) Z_k (U_k^\dagger)^\top, \\ W_{k+1} & \text{is computed by (4.3b) for } \textbf{Option 1}\text{, or by (4.7) for } \textbf{Option 2}, \\ \Lambda_{k+1} & := \Lambda_k + \rho(\mathscr{A}(U_{k+1}V_{k+1}^\top) - W_{k+1} - B). \end{cases} \tag{4.9}$$

Clearly, computing $[U_{k+1}, V_{k+1}]$ in (4.9) using the step-size in Lemma 3.3 is impractical. Similar to Algorithm 1, we find an appropriate α_k by a backtracking linesearch on $\mathcal{Q}_k(U,V) := (1/2) \left\| \mathscr{A}(UV^\top) - W_k - B + \rho^{-1}\Lambda_k \right\|_2^2$ as

$$\mathcal{Q}(U_k + \alpha D_{U_k}, V_k + \alpha_k V_k) \le \mathcal{Q}(U_k, V_k) - 0.5 c_1 \alpha_k \Delta_k^2, \tag{4.10}$$

where $\Delta_k^2 := \|U_k^\top \mathscr{A}^*(E_k - W_k)\|_F^2 + \|\mathscr{A}^*(E_k - W_k)V_k\|_F^2$ and $\alpha_k := \beta^{i_k}\alpha_0$ with $\alpha_0 > 0$ and $\beta := (\sqrt{5}-1)/(\sqrt{5}+1) \in (0,1)$ given a priori. Obviously, by Lemma 3.3, this procedure terminates after a finite number of linesearch steps i_k satisfying (3.13). In addition, $D_{X_k} := [D_{U_k}, D_{V_k}]$ is a descent direction of the quadratic objective $\mathcal{Q}_k$ at X_k.

Now, we expand (4.9) algorithmically as in Algorithm 2.

Algorithm 2 (*ADMM-Gauss-Newton Algorithm* (ADMM-GN))

```
Initialization: Given $\varepsilon > 0$, choose $\rho > 0$ and $X_0 := [U_0, V_0]$.
    Set $W_0 := U_0 V_0^\top$ and $\Lambda_0 := 0^{m \times n}$.
for $k = 0$ to $k_{\max}$ do
    Gauss-Newton step: Compute a GN direction $D_{X_k} := [D_{U_k}, D_{V_k}]$ by (4.5).
    Linesearch step: Find $\alpha_k > 0$ from the linesearch condition (4.10) and update
        $U_{k+1} := U_k + \alpha_k D_{U_k}$   and   $V_{k+1} := V_k + \alpha_k D_{V_k}$.
    Gradient step: Evaluate $Y_{k+1} := \mathscr{A}(U_{k+1}V_{k+1}^\top) - B$, and $\Phi'(W_k)$, and
        Option 1: update $W_{k+1}$ by (4.3b)   or   Option 2: update $W_{k+1}$ by (4.7)
    If stopping_criterion, then TERMINATE.
    Update $\Lambda_{k+1} := \Lambda_k + \rho(Y_{k+1} - W_{k+1})$.
end for
```

Per-iteration complexity. The main steps of Algorithm 2 remain at Steps 4 and 5, where they require to compute $D_{X_k} := [D_{U_k}, D_{V_k}]$ and to perform a linesearch procedure, respectively. Steps 6 and 8 only require matrix-matrix additions which have the complexity of $\mathcal{O}(m \times n)$. Overall, the per-iteration complexity of Algorithm 2 is higher than of Algorithm 1, but as we can see from Section 6 that we can simply use the full-step GN scheme at Step 4 without linesearch, and Algorithm 2 often requires a fewer number of iterations than Algorithm 1. Moreover, Algorithm 2 seems working well for the "large residual" case, i.e., $\mathcal{A}^*(\nabla\phi(\mathcal{A}(U_\star V_\star^\top) - B))$ is large.

Global convergence analysis. We first write the optimality condition (or the KKT condition) for (4.1) as follows:

$$\nabla\phi(W_\star) - \mathcal{A}^*(\Lambda_\star) = 0,\ U_\star^\top\mathcal{A}^*(\Lambda_\star) = 0,\ \mathcal{A}^*(\Lambda_\star)V_\star = 0, \text{ and } \mathcal{A}(U_\star V_\star^\top) - W_\star = B. \quad (4.11)$$

This condition can be rewritten as (2.1) by eliminating $W_\star$ and the multiplier $\Lambda_\star$. Hence, if $[U_\star, V_\star, W_\star, \Lambda_\star]$ satisfies (4.11), then $X_\star := [U_\star, V_\star] \in \mathcal{X}_\star$.

The following lemma provides a key step to prove the convergence of Algorithm 2, whose proof is given in Appendix A.5.

Lemma 4.1. *Let $\{[U_k, V_k, W_k, \Lambda_k]\}$ be generated by Algorithm 2. Then, under Assumption A.2.1, the following statements hold:*

(a) *The sequence $\{(W_k, \Lambda_k)\}$ is bounded. In addition, for $k \geq 1$, we have*

$$\begin{array}{lll} & \|\Lambda_{k+1} - \Lambda_k\|_2 \leq L_\phi\|W_{k+1} - W_k\|_2 & \textit{for } \textbf{\textit{Option 1}}, \\ \textit{or} & \|\Lambda_{k+1} - \Lambda_k\|_2 \leq L_\phi\left(\|W_k - W_{k-1}\|_2 + \|W_{k+1} - W_{k-1}\|_2\right) & \textit{for } \textbf{\textit{Option 2}}. \end{array} \quad (4.12)$$

(b) *Let $\mathcal{L}_\rho$ be defined by (4.2). Then, for any $\rho > 0$, we have*

$$\begin{array}{l} \mathcal{L}_\rho(U_{k+1}, V_{k+1}, W_{k+1}, \Lambda_{k+1}) \leq \mathcal{L}_\rho(U_k, V_k, W_k, \Lambda_k) - \frac{\eta_1}{2}\|W_{k+1} - W_k\|_2^2 \\ \quad + \frac{\eta_0}{2}\|W_k - W_{k-1}\|_2^2 - \frac{c_1\rho\alpha_k}{2}\left[\|U_k^\top\mathcal{A}^*(E_k - W_k)\|_F^2 + \|\mathcal{A}^*(E_k - W_k)V_k\|_F^2\right], \end{array} \quad (4.13)$$

where $E_k := \mathcal{A}(U_k V_k^\top) - B + \rho^{-1}\Lambda_k$, and

$$\begin{array}{llll} & \eta_1 := \rho^{-1}\left(\rho^2 + \mu_\phi\rho - 2L_\phi^2\right) & \textit{and} \quad \eta_0 := 0 & \textit{for } \textbf{\textit{Option 1}}, \\ \textit{or} & \eta_1 := \rho^{-1}\left(\rho^2 + L_\phi\rho - 4L_\phi^2\right) & \textit{and} \quad \eta_0 := 8\rho^{-1}L_\phi^2 & \textit{for } \textbf{\textit{Option 2}}. \end{array} \quad (4.14)$$

Similar to Algorithm 1, we prove a global convergence of Algorithm 2 in the following theorem, whose proof is deferred to Appendix A.6.

Theorem 4.1. *Under Assumption A.2.1 and condition (3.14), let $\{[U_k, V_k]\}$ be generated by Algorithm 2. Then, if we choose ρ such that*

$$\begin{cases} \rho > 0.5\left(\left(\mu_\phi + 8L_\phi^2\right)^{1/2} + \mu_\phi\right) & \textit{for } \textbf{\textit{Option 1}}, \\ \rho > 3L_\phi & \textit{for } \textbf{\textit{Option 2}}, \end{cases} \quad (4.15)$$

then

$$\lim_{k\to\infty}\|\nabla\Phi(U_k, V_k)\|_F = 0. \quad (4.16)$$

Consequently, there exists a limit point $X_\star := [U_\star, V_\star]$ of $\{[U_k, V_k]\}$ and $X_\star \in \mathcal{X}_\star$.

5. Symmetric low-rank matrix optimization

In this section, we develop a symmetric GN variant of Algorithm 1 for solving the following special symmetric setting of (1.1) when $U = V$:

$$\Phi^\star := \min_U \left\{ \Phi(U) := \phi\left(\mathscr{A}(UU^\top) - B\right) \ : \ U \in \mathbb{R}^{m\times r} \right\}. \tag{5.1}$$

Clearly, (3.14) is a generalization of the least-squares problem in [8]. In addition, we cannot directly apply alternating scheme to solve (5.1) without reformulating it into other form. The optimality condition of (5.1) is written as

$$U^\top \mathscr{A}^*\left(\nabla\phi(\mathscr{A}(UU^\top) - B)\right) = 0. \tag{5.2}$$

Any $U_\star$ satisfying this condition is called a *stationary point* of (5.1). We again assume that the set of stationary points $\mathscr{U}_\star$ of (5.1) is nonempty.

We now customize Algorithm 1 to find a stationary point of (5.1). Since $U = V$, the symmetric GN direction can be computed from Remark 3.1 as

$$D_U = (\mathbb{I} - 0.5P_U)Z(U^\dagger)^\top, \quad \text{where} \quad Z = -L_\Phi^{-1}\mathscr{A}^*\left(\nabla\phi(\mathscr{A}(UU^\top) - B)\right).$$

Combining this step and modifying the linesearch procedure (3.12), we can describe a new variant of Algorithm 1 for solving (5.1) as in Algorithm 3.

Algorithm 3 (*Symmetric linesearch Gauss-Newton algorithm* (SLs-GN))

```
1: Initialization: Given a tolerance $\varepsilon > 0$. Choose $U_0 \in \mathbb{R}^{m\times r}$. Set $\alpha_0 := 1$ and $c_1 := 0.5$.
2: for $k = 0$ to $k_{\max}$ do
3:     Gauss-Newton direction: Evaluate $Z_k := -L_\Phi^{-1}\mathscr{A}^*\nabla\phi(\mathscr{A}(U_kU_k^\top) - B)$ and compute
           $D_{U_k} := (\mathbb{I}_m - 0.5P_{U_k})Z_k(U_k^\dagger)^\top.$
4:     If $\|D_{U_k}\|_F \le \varepsilon\max\{1, \|U_k\|_F\}$, then TERMINATE.
5:     Linesearch: Find the smallest number $i_k \ge 0$ such that $\alpha_{i_k} := \beta^{i_k}\alpha_0$ and
           $\Phi(U_k + \alpha_{i_k}D_{U_k}) \le \Phi(U_k) - 0.5c_1\alpha_{i_k}\|\nabla\Phi(U_k)\|_F^2.$
6:     Update $U_{k+1} := U_k + \alpha_k D_{U_k}$.
7: end for
```

Per-iteration complexity. Computing $U^\dagger$ requires one QR-factorization of an $m \times r$ matrix to get $[Q, R] = \mathrm{qr}(U)$. Then, we form $U^\dagger = R^\dagger Q^T$, where $R^\dagger$ is obtained by solving an upper triangle linear system. P_{U_k} is computed by $P_{U_k} = U_kU_k^\dagger$. Computing Z_k at Step 3 requires $U_kU_k^\top$, one linear operator $\mathscr{A}$ and one adjoint $\mathscr{A}^*$. The linesearch routine at Step 5 requires i_k function evaluations as indicated in (3.13). Each linesearch step needs one $U_kU_k^\top$ and one $\mathscr{A}(\cdot)$.

The following corollary summarizes the convergence properties of Algorithm 3, which is a direct consequence of Lemma 3.3 and Theorem 3.1.

Corollary 5.1. *Let $\{U_k\}$ be generated by Algorithm 3. Then, under Assumption A.2.1:*

(a) *There exists* $\underline{\alpha}_k := \min\left\{1, \frac{L_\Phi\sigma_{\min}^3(U_k)}{2\|\nabla\Phi(U_k)\|_F}, \frac{3\sigma_{\min}(U_k)^4}{32\sigma_{\max}(U_k)^2\Phi'(U_k)}\right\} \in (0, 1]$ *such that*

$$\Phi(U_k + \alpha_k D_{U_k}) \le \Phi(U_k) - \frac{\alpha_k}{128L_\Phi}\frac{\sigma_{\min}^2(U_k)}{\sigma_{\max}^4(U_k)}\|\nabla\Phi(U_k)\|_F^2, \quad \forall\alpha_k \in (0, \underline{\alpha}_k]. \tag{5.3}$$

Consequently, the linesearch procedure at Step 5 is well-defined (i.e., it terminates after a finite number of iterations i_k).

(b) *If there exists $\underline{\sigma} > 0$ such that $\sigma_{\min}(U_k) \geq \underline{\sigma}$ for all $k \geq 0$, then $\lim_{k\to\infty} \|\nabla\Phi(U_k)\|_F = 0$, and any limit point of $\{U_k\}$ is in $\mathscr{U}_\star$.*

The results in Corollary 5.1 is fundamentally different from [8], even when $\phi(\cdot) := (1/2)\|\cdot\|_2^2$ and $\mathscr{A}$ is identical, since B is not positive definite. We note that Algorithm 2 can be specified to handle the symmetric case (5.1) by substituting Steps 4 and 5 by Steps 3 and 5 in Algorithm 3, respectively. We omit the details of this specification.

6. Numerical experiments

In this section, we first discuss some implementation aspects. Next, we compare the full-step GN scheme and AMA. Then, we test Algorithm 1 on a low-rank matrix approximation problem and compare it with standard SVDs. Finally, we apply Algorithms 1, 2 and 3 to solve three problems: matrix completion, matrix recovery, and robust low-rank matrix recovery.

6.1. Implementation remarks.

The following aspects are implemented in our experiments.

Computing initial point. Since (1.1) is nonconvex, the performance of the above algorithms strongly depends on an initial point. Principally, these algorithms still converge from any initial point. However, we propose to use the following simple procedure for finding a "good" initial point. We first form a matrix $M \in \mathbb{R}^{m\times n}$ such that $\mathscr{A}(M) = B$. Then, we compute the r-truncated SVD of M as $[U_f, \Sigma_f, V_f]$ and form

$$U_0 := U_f(:, 1:r)\Sigma_f(1:r)^{1/2} \text{ and } V_0 := V_f(:, 1:r)\Sigma_f(1:r)^{1/2}.$$

In Algorithm 2, given $[U_0, V_0]$, we set $W_0 := \mathscr{A}(U_0V_0^\top) - B$ and $\Lambda_0 := 0^l$.

Stopping criterions. We can implement different stopping criterions for Algorithms 1 and 2. The first criterion is based on the optimality condition (2.1) as

$$\max\left\{\|U_k^\top\Phi'(U_kV_k^\top)\|_F, \|\Phi'(U_kV_k^\top)V_k\|_F\right\} \leq \varepsilon_1 \max\{1, \|B\|_F\}, \tag{6.1}$$

where $\Phi'(UV^\top) := \mathscr{A}^*\left(\nabla\phi(\mathscr{A}(UV^\top) - B)\right)$. We can terminate Algorithm 1 if

$$\max\{\|D_{U_k}\|_F, \|D_{V_k}\|_F\} \leq \varepsilon_1 \max\{1, \|B\|_F\}. \tag{6.2}$$

We can add to Algorithm 2 the following condition for feasibility in (4.1):

$$\|U_kV_k^\top - W_k\|_F \leq \varepsilon_1 \max\{1, \|B\|_F\}. \tag{6.3}$$

When $\phi(\cdot) := (1/2)\|\cdot\|_F^2$ and the optimal value is zero, we also use

$$\|\mathscr{A}(U_kV_k^\top) - B\|_F \leq \varepsilon_2 \max\{1, \|B\|_F\}. \tag{6.4}$$

Similar stopping criterions are applied to Algorithm 3.

Penalty parameter update. Theoretically, we can fix any parameter ρ as indicated in (4.15). However, in Section 6, we follow the update rule used in [23] but with different parameters. We also use the full-step GN scheme at Step 4.

6.2. **Comparison of Gauss-Newton and Alternating Minimization Algorithm.** In order to observe the advantage of the GN scheme over AMA (also called alternating direction method) for solving (1.1), we compare these algorithms on the following special case of (1.1):

$$\Phi^{\star} := \min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}} \left\{\Phi(U,V) := (1/2)\|\mathscr{A}(UV^{\top}) - B\|_2^2\right\}. \tag{6.5}$$

Since $\mathscr{A}$ is nonidentical, we upper bound $(1/2)\|\mathscr{A}(\cdot) - B\|_2^2$ as

$$\begin{aligned}\tfrac{1}{2}\|\mathscr{A}(UV^{\top}) - B\|_2^2 \;\leq\; & \tfrac{1}{2}\|\mathscr{A}(U_kV_k^{\top}) - B\|_2^2 + \tfrac{1}{2}\|UV^{\top} - (U_kV_k^{\top} - L^{-1}\mathscr{A}^{*}(\mathscr{A}(U_kV_k^{\top}) - B)))\|_2^2 \\ & - \tfrac{1}{2L}\|\mathscr{A}^{*}(\mathscr{A}(U_kV_k^{\top}) - B))\|_2^2,\end{aligned}$$

where $L := \|\mathscr{A}\|^2$ is the Lipschitz constant of the gradient of $(1/2)\|\mathscr{A}(\cdot) - B\|_2^2$.

Let $Z_k := L^{-1}\mathscr{A}^{*}(\mathscr{A}(U_kV_k^{\top}) - B))$. We can write AMA as

$$\begin{cases} U_{k+1} := \arg\min_{U}\left\{(1/2)\|UV_k^{\top} - (U_kV_k^{\top} - Z_k)\|_2^2\right\}, \\ V_{k+1} := \arg\min_{V}\left\{(1/2)\|U_{k+1}V^{\top} - (U_kV_k^{\top} - Z_k)\|_2^2\right\}. \end{cases} \tag{AMA}$$

We compare this algorithm and the following full-step GN scheme of (3.8):

$$(U_{k+1}, V_{k+1}) := \arg\min_{U,V}\left\{(1/2)\|U_kV^{\top} + UV_k^{\top} - (U_kV_k^{\top} + Z_k)\|_2^2\right\}. \tag{FsGN}$$

Clearly, AMA alternates between U and V and solves for them separately, while FsGN linearizes $UV^{\top}$ and solves for U_{k+1} and V_{k+1} simultaneously.

We implement these schemes in Matlab and running on a MacBook laptop with a 2.6 GHz Intel Core i7 processor and 16GB memory. The input data is generated as follows. For $\mathscr{A}$, we generate an $(mn \times mn)$-matrix from either a fast Fourier transform (fft) or a standard Gaussian distribution, and take l random sub-samples from the rows of this matrix to form $\mathscr{A}$, where $l \leq mn$. We generate $B = \mathscr{A}(U^{\natural}(V^{\natural})^{\top}) + \mathscr{N}(0, \sigma^2\mathbb{I})$, where $U^{\natural} \in \mathbb{R}^{m\times r}$ and $V^{\natural} \in \mathbb{R}^{n\times r}$ are given matrices, and $\mathscr{N}(0, \sigma^2\mathbb{I})$ is i.i.d. Gaussian noise of variance σ^2. We consider two cases: the underdetermined case with $l < r(m+n)$, and the overdetermined case with $l > r(m+n)$. In the first case, problem (6.5) always has a solution with zero residual. We choose $[U_0, V_0]$ randomly, which may not be in the local convergence region of the GN method.

Figure 1 shows the convergence behavior of the two algorithms. The right plot is $l = 2r(m+n)$, and the left one is $l = 0.5r(m+n)$, where $m = n = 512$ and $r = 32$.

We can see from Figure 1 that both algorithms perform very similarly in early iterations, but then FsGN gives better result in terms of accuracy (terminated around 10^{-9} in the overdetermined case due to the nonzero objective residual), while AMA is saturated at a certain level, and does not improve the objective values. In addition, Figures 1 and 2 show that the full-step Gauss-Newton scheme has a local linear convergence rate for the underdetermined case. However, as a compensation, FsGN requires one $(r \times m)$-matrix multiplication $U^{\top}U$ and one $(r \times r)$-inverse compared to AMA. This suggests that we can perform AMA in early iterations and switch to FsGN if AMA does not make significant progress to improve the objective values.

We test the underdetermined case by choosing a Gaussian operator $\mathscr{A}$ generated as $\mathscr{A} = \frac{1}{\sqrt{l}}\texttt{sprandn}(l, mn, 0.05)$. The convergence of two algorithms on this dataset is plotted in Figure 2 (left). Finally, we consider the effect of noise to both algorithms by adding a Gaussian noise with $\sigma^2 = 10^{-3}$. The performance of these algorithms is plotted in Figure 2 (right).

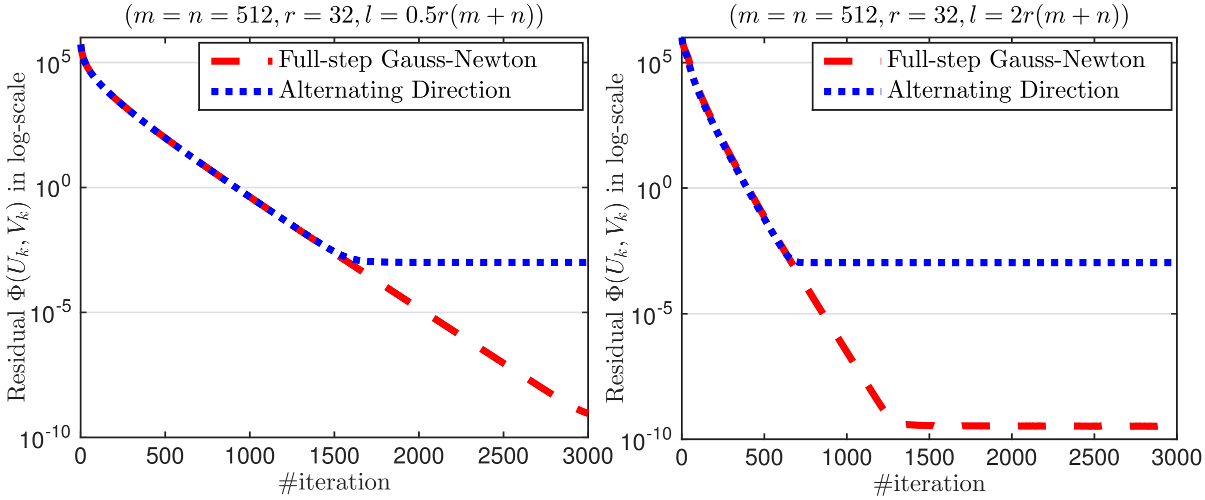

FIGURE 1. A comparison between FsGN (*Legend*: Full-step Gauss-Newton) and AMA (*Legend:* Alternating Direction). Left: The underdetermined case – $l = 0.5r(m+n)$. Right: The overdetermined case – $l = 2r(m+n)$.

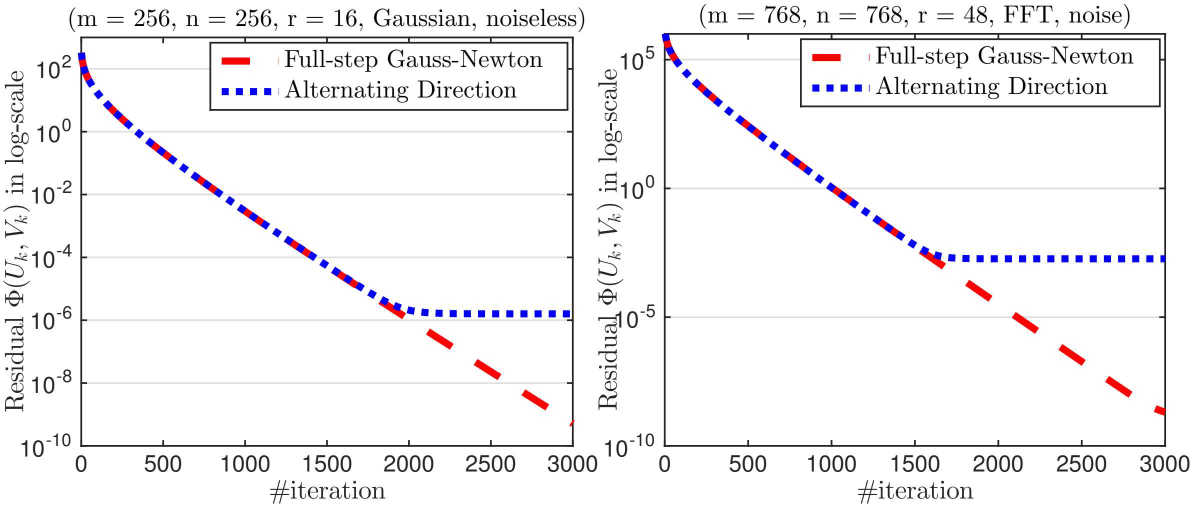

FIGURE 2. A comparison between FsGN and AMA. *Left:* sparse Gaussian operator without noise. *Right:* subsampling FFT linear operator with noise.

We can observe from Figure 2 the same behavior as in the previous test. Our FsGN still maintains a local linear convergence even with noise, while AMA is saturated at a certain level of the objective values.

6.3. **Low-rank matrix factorization and linear subspace selection.** We consider a special case of (1.1) by taking $\phi(\cdot) := (1/2)\|\cdot\|_F^2$ and $\mathscr{A} = \mathbb{I}$ as

$$\Phi^\star := \min_{U,V}\left\{\Phi(U,V) := (1/2)\|UV^\top - B\|_F^2 \ : \ U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}\right\}. \tag{6.6}$$

Although this problem has a closed form solution by truncated SVD, our objective is to compare the full-step GN variant of Algorithm 1 with standard Matlab singular value decomposition routines: `svds` and `lansvd`. The full-step GN scheme for (6.6) is presented as

$$V_{k+1}^{\top} := U_k^{\dagger} B \quad \text{and} \quad U_{k+1} := U_k + (B - U_k^{\top} V_{k+1})(V_k^{\dagger})^{\top}. \tag{6.7}$$

At each iteration, (6.7) requires two $(r \times r)$-matrix inverses $U_k^{\top} U_k$ and $V_k^{\top} V_k$, and three $(m \times r)$- or $(n \times r)$- matrix - $(r \times r)$-matrix multiplications. We compute these two inverses by Cholesky decomposition. We note that we do not form the $(m \times n)$-matrix $U_k V_k^{\top}$ at each iteration, but we can occasionally compute it to check the objective value if required. We choose $U_0 := [\mathbb{I}_r, 0_{(m-r)\times r}^{\top}]^{\top}$ and $V_0 := [0_{(n-r)\times r}^{\top}, \mathbb{I}_r]^{\top}$ as a starting point, where $\mathbb{I}_r$ is the identity matrix.

Scheme (6.7) generates two low-rank matrices U_k and V_k so that $U_k V_k^{\top} \approx B$. We can perform a Rayleigh–Ritz (RR) routine to orthonormalize U_k and V_k,

- Compute $[Q_u, R_u] = \mathtt{qr}(U_k, 0)$ and $[Q_v, R_v] = \mathtt{qr}(V_k, 0)$, the two economic QR-factorizations of size r.
- Compute $[U_r, \Sigma_r, V_r] = \mathtt{svd}(Q_u^{\top} B Q_v)$ the singular value decomposition of the $r \times r$ matrix $Q_u^{\top} B Q_v$.
- Then form $U = Q_u U_r$ and $V = Q_v V_r$ to obtain two orthogonal matrices U and V of the size $m \times r$ and $n \times r$ so that $[U, \Sigma, V] = \mathtt{svds}(B, r)$.

Here, (6.7) works on a symmetric positive definite matrix compared to [8].

Now, we test (6.7) in combining with the Rayleigh–Ritz procedure, and compare it with `svds` and `lansvd`. We generate an input matrix B of size $m \times n$ with rank r. Once m is chosen, we set $n = m$ and either $r = 0.01 \times m$ or $r = 0.05 \times m$ (which is either 1% or 5% of the problem size, respectively). Then, we generate $B \in \mathbb{R}^{m \times n}$ using the following Matlab code:

```
min_mn      = min(m, n);
nnz_sig_vec = [1:1:r].^(-0.01);
sig_vec     = [nnz_sig_vec(:); zeros(min_mn-r, 1)];
n_sig_vec   = sqrt(length(sig_vec))/norm(sig_vec(:), 2)*sig_vec;
B           = gallery(’randcolu’, n_sig_vec, max(m, n), 1);
G           = sprandn(m, n, nnz(B)/(m*n));
M_mat       = B + 0.1*norm(B, ’fro’)*G/norm(G, ’fro’);
```

Clearly, the singular values σ_i of B are clustered into two parts: $\sigma_i = i^{-0.01}$ for $i = 1, \cdots, r$, and $\sigma_i = 0$ for $i = r+1, \cdots, \min\{m, n\}$. In addition, an i.i.d. Gaussian noise $\frac{\|B\|_F}{10\|G\|_F} G$ is added to B, where $G = \mathcal{N}(0, \sigma\mathbb{I})$, with σ being the sparsity of B. We terminate (6.7) using either (6.1) or (6.2) with $\varepsilon_1 = 10^{-6}$ or $\varepsilon_2 = 10^{-4}$, respectively. We also terminate `svds` and `lansvd` using $\mathtt{tol} = 10^{-4}$, which is a moderate accuracy.

The performance of three algorithms in terms of computational time vs. problem size is plotted in Figure 3 for 10 problems from $m = n = 1{,}000$ to $m = n = 10{,}000$, carried out on a MacBook laptop with a 2.6 GHz Intel Core i7 processor and 16GB memory. We run each problem size 10 times and compute the averaging computational time. The abbreviation `Full-step Gauss-Newton` indicates the time of both scheme (6.7) and Rayleigh-Ritz procedure, while `Full-step Gauss-Newton without RR` only counts for the time of (6.7). Figure 3 (left) shows the performance with $r = 0.01 \times m$, while Figure 3 (right) reveals the case $r = 0.05 \times m$.

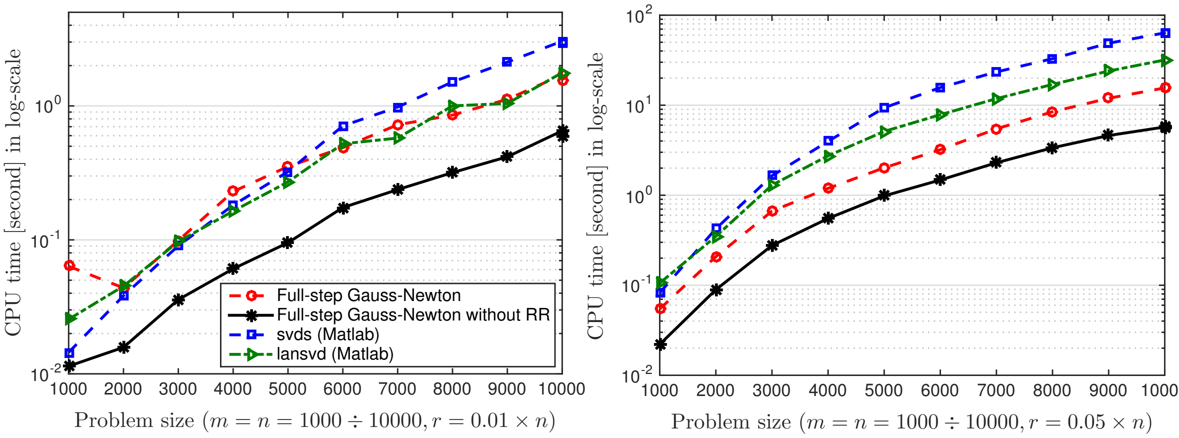

FIGURE 3. A comparison between the full-step GN scheme, `Matlab SVDS`, and `Matlab LanSVD` on 10 problem sizes (from 1,000 to 10,000), and two different ranks. The result is on the average of 10 random runs for each problem size.

When the rank r is about 1% of the problem size, (6.7) is comparable to `lansvd` while it is slightly better than `svds`. However, when the rank r is increased to 5% of the problem size, (6.7) clearly outperforms both `lansvd` and `svds`.

6.4. **Recovery with Pauli measurements in quantum tomography.** We consider a d spin-1/2 system with unknown state S as described in [40]. A d-qubit Pauli matrix is given by the form $w = \otimes_{i=1}^{d} w_i$, where $w_i \in \{1, \sigma^x, \sigma^y, \sigma^z\}$ is a given set of elements. There are n^2, $n = 2^d$, such matrices denoted by $w(s)$ with $s \in \{1, \cdots, n^2\}$. A compressive sensing procedure takes m integer numbers $s_1, \cdots, s_m \in \{1, \cdots, n^2\}$ randomly and measures the expected values $\mathrm{trace}\,(Sw(s_i))$. Then, it solves the following convex problem to construct the unknown states:

$$\mathrm{trace}\,(X) = 1, \quad \mathrm{trace}\,(Xw(s_i)) = \mathrm{trace}\,(w_i S) \quad (i = 1, \cdots, m). \tag{6.8}$$

From [40], the number of measurement m to reconstruct the quantum states can be estimated as $m = cnr\log^2 n \ll n^2$ for some constant c and the rank r.

Given that X characterizes a density matrix, which is positive semidefinite Hermitian, we instead consider the following least-squares formulation of (6.8):

$$\min_{X \in \mathscr{H}_+^n} \left\{(1/2)\|B - \mathscr{A}(X)\|_F^2 \ : \ \mathrm{trace}(X) = 1\right\}, \tag{6.9}$$

where $\mathscr{H}_+^n$ is the set of positive semidefinite Hermitian matrices of size n, and $\mathscr{A}$ and B are the measurement operator and observed measurements obtained from (6.8), respectively. Assume that $X = UU^\top$, where $U \in \mathbb{C}^{n\times 1}$, we can write (6.9) into

$$\min_{U \in \mathbb{C}^{n\times 1}} \left\{(1/2)\|B - \mathscr{A}(UU^\top)\|_F^2\right\}, \tag{6.10}$$

where $\mathbb{C}^{n\times 1}$ is the set of $(n \times 1)$ - complex matrices. Clearly, problem (6.10) falls into the special form (5.1) of (1.1) which can be solved by Algorithm 3.

We test Algorithm 3 and compared it with Frank-Wolfe's method proposed in [41]. We use both the standard Frank-Wolfe and its linesearch variant. We generate $U_0 := [\mathbb{I}_r, 0_{(n-r)\times r}^\top]^\top$ and terminate Algorithm 3 using either (6.1), (6.2), or (6.4) with $\varepsilon_1 = 10^{-9}$ and $\varepsilon_2 = 10^{-6}$,

TABLE 1. Numerical results of three algorithms on noiseless and noisy data

			Algorithm 3			Frank-Wolfe without LS			Frank-Wolfe with LS		
#qubits	m	n	iter	time[s]	$\frac{\|B-\mathscr{A}(X)\|_F}{\|B\|_F}$	iter	time[s]	$\frac{\|B-\mathscr{A}(X)\|_F}{\|B\|_F}$	iter	time[s]	$\frac{\|B-\mathscr{A}(X)\|_F}{\|B\|_F}$
The noiseless case											
10	14196	1024	26	12.25	3.21e-06	1707	664.90	1.65e-03	322	129.35	1.62e-03
11	31231	2048	25	71.60	2.64e-06	1654	2803.18	1.61e-03	370	593.56	1.54e-03
12	68140	4096	25	696.27	1.78e-06	1577	17990.98	1.56e-03	254	1741.19	1.54e-03
13	147635	8192	27	1516.97	1.73e-06	648	20574.13	3.68e-03	303	9654.69	1.52e-03
The depolarizing noisy case (1%)											
10	14196	1024	24	16.07	8.99e-06	1711	692.16	1.66e-03	238	78.22	1.66e-03
11	31231	2048	23	94.98	8.80e-06	1663	2683.90	1.62e-03	258	423.27	1.61e-03
12	68140	4096	23	589.73	6.03e-06	1585	12146.01	1.56e-03	247	1892.66	1.56e-03
13	147635	8192	24	3684.57	8.76e-06	648	20537.15	3.70e-03	292	8691.90	1.53e-03

respectively. We generate $\mathscr{A}$ and B using the procedures in [40]. We perform two cases: noise and noiseless. In the noisy case, we set S to be $0.99S + 0.01\mathbb{I}_n/n$ before computing the observed measurement B. Since Frank-Wolfe's algorithms take long time to reach a high accuracy, we terminate them if $\|\mathscr{A}(X) - B\|_F \leq 10^{-3}\sqrt{2}\|B\|_F$ which is different from Algorithm 3.

We test on 4 problems of the size d with $d \in \{10, 11, 12, 13\}$ being the number of qubits running one a single node of an Intel(R) Xeon(R) 2.67GHz cluster with 4GB memory, but can share up to 320GB RAM. The results and performance of three algorithms are reported in Table 1, where m is the number of measurements, $n = 2^d$, iter is the number of iterations, time[s] is the computational time in seconds. The convergence behavior of three algorithms for both noiseless and noisy cases with $d = 13$ is also plotted in Figure 4.

We can observe from our results that Algorithm 3 highly outperforms the two Frank-Wolfe variants. It also reaches a highly accurate solution after a few iterations. However, each iteration of Algorithm 3 is more expensive than that of Frank-Wolfe's algorithms. As can be seen from Figure 4, Algorithm 3 behaves like super-linearly convergent.

6.5. **Matrix completion.** Our next experiment is solve the well-known matrix completion (MC) widely used in recommender systems [3, 6, 9]. This problem is a special case of (1.1) and can be written as follows:

$$\min_{U,V}\left\{\|(1/2)\mathscr{P}_{\Omega}(UV^{\top}) - B\|_F^2 \ : \ U \in \mathbb{R}^{r\times m}, V \in \mathbb{R}^{r\times n}\right\}, \tag{6.11}$$

where $\mathscr{P}_{\Omega}$ is a selection operator on an index subset Ω, and B is the set of observed entries.

There are two major approaches to solve (6.11). The first one is using a convex relaxation for the rank constraint via nuclear or max norms. Methods based on this approach have been widely developed, including SVT [42], and [accelerated] gradient descent [6, 43]. The second approach is using nonconvex optimization, including, e.g., OpenSpace [26] and LMaFit [9, 22].

In this experiment, we select the most efficient algorithms for our comparison: the over-relaxation alternating direction method (LMaFit) in [9], and the accelerated proximal gradient method (APGL) in [43]. We will test the four algorithms on synthetic datasets and the three first algorithms on some real datasets.

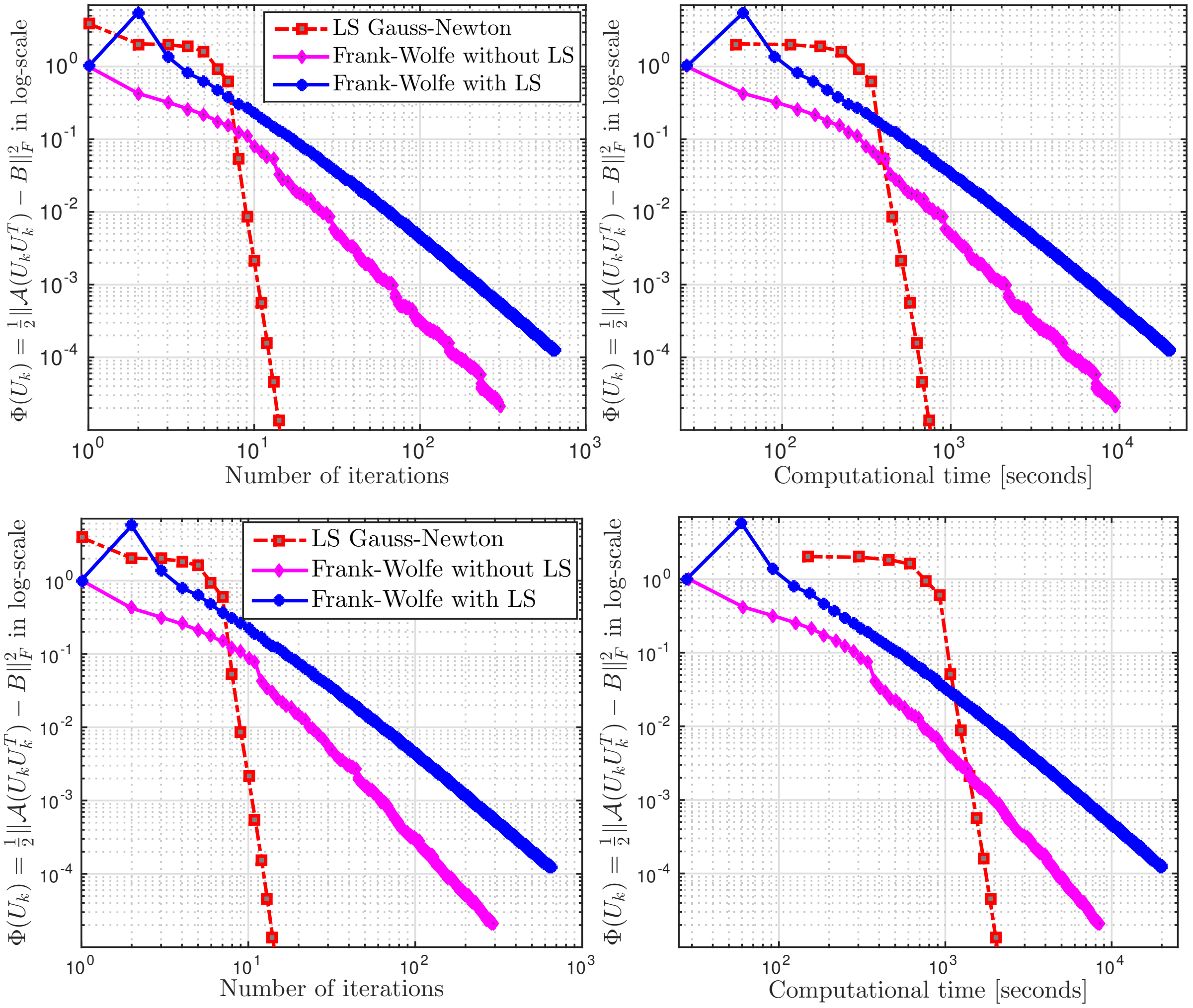

FIGURE 4. A comparison of three algorithms for the noiseless case (first row) and for the 0.01 depolarizing noisy case (second row).

Synthetic datasets. Since data in rating systems is often integer, our synthetic dataset is generated as follows. We randomly generate two integer matrices U and V whose entries are in $\{1,\cdots,5\}$ of the size $m \times r$ and $n \times r$, respectively. Then, we form $M = UV^{\top}$. Finally, we randomly take either 50% or 30% entries of M as an output matrix B. We can also add a standard Gaussian noise to B if necessary. A Matlab script for generating such a dataset is given below.

```
U_org     = randi(5, m, r);
V_org     = randi(5, n, r);
M_org     = U_org*V_org';
s         = round(0.5*m*n);
Omega     = randsample(m*n, s);
M_omega   = M_org(Omega);
B         = M_omega + sigma*randn(size(M_omega));
```

We first test these algorithms with a fixed rank r and 50% randomly observed entries, which is relative dense. We terminate Algorithms 1 and 2 using the conditions given in Section 6.1 with $\varepsilon_1 = 10^{-6}$ and $\varepsilon_2 = 10^{-4}$, respectively. We also terminate LMaFit and APGL with the same tolerance `tol` $= 10^{-4}$. The initial point is computed by a truncated SVD as in Section 6.1.

TABLE 2. Comparison of four algorithms on synthetic integer data without noise

			Algorithm 1					Algorithm 2				
m	n	r	iter	time[s]	δf_k	NMAE	rank	iter	time[s]	δf_k	NMAE	rank
1000	2000	10	15.9	2.11	4.15e-05	1.39e-05	10	30.0	2.07	8.34e-05	3.55e-05	10
1000	2000	50	20.8	4.34	6.91e-05	5.78e-05	50	37.6	4.43	9.61e-05	7.83e-05	50
2500	2500	25	14.3	8.05	5.31e-05	2.97e-05	25	31.1	9.18	1.04e-04	6.18e-05	25
2500	2500	125	26.3	28.94	7.35e-05	8.18e-05	125	35.2	23.68	1.05e-04	1.32e-04	125
5000	5000	50	15.7	53.40	5.42e-05	3.90e-05	50	32.0	56.71	9.87e-05	7.80e-05	50
5000	5000	250	23.6	180.70	7.94e-05	1.35e-04	250	35.0	165.89	1.10e-04	1.94e-04	250
5000	7500	50	14.9	76.93	5.10e-05	3.64e-05	50	32.0	85.72	8.51e-05	6.65e-05	50
5000	7500	250	23.7	273.30	7.61e-05	1.22e-04	250	35.0	245.93	9.97e-05	1.68e-04	250
10000	10000	100	16.2	289.14	5.99e-05	5.86e-05	100	32.2	319.92	1.10e-04	1.16e-04	100
10000	10000	500	24.8	1303.01	8.02e-05	1.75e-04	500	35.0	1173.38	1.14e-04	2.60e-04	500
			LMaFit [9]					APGL [43]				
m	n	r	iter	time[s]	δf_k	NMAE	rank	iter	time[s]	δf_k	NMAE	rank
1000	2000	10	13.3	0.94	4.74e-05	1.43e-05	10	28.0	4.29	3.31e-04	1.40e-04	10
1000	2000	50	109.8	12.13	2.71e-04	1.51e-04	50	28.6	6.79	1.06e-02	9.02e-03	41.6
2500	2500	25	10.0	3.10	6.98e-05	3.86e-05	25	29.0	16.64	5.06e-04	3.07e-04	25
2500	2500	125	135.3	85.78	2.99e-04	2.59e-04	125	31.6	20.00	1.92e-02	2.43e-02	5
5000	5000	50	10.0	18.43	5.57e-05	4.30e-05	50	30.6	81.69	8.48e-04	6.73e-04	50.2
5000	5000	250	140.9	631.58	6.70e-05	8.58e-05	250	30.4	60.75	1.38e-02	2.44e-02	5
5000	7500	50	10.1	28.21	4.38e-05	2.92e-05	50	30.8	122.47	7.35e-04	5.76e-04	50
5000	7500	250	126.8	845.90	7.02e-05	7.30e-05	250	30.5	90.74	1.38e-02	2.33e-02	5
10000	10000	100	11.0	112.82	3.10e-05	3.24e-05	100	32.7	266.16	2.15e-02	2.28e-02	5
10000	10000	500	120.4	3818.05	6.25e-05	8.63e-05	500	30.3	206.86	9.85e-03	2.26e-02	5

The test is conducted on 10 problems of different sizes running on a single node of an Intel(R) Xeon(R) 2.67GHz cluster with 4GB memory, but can share up to 100GB RAM. We run each problem size 10 times and compute the average result and performance. The problem sizes and results are reported in Table 2 for two different ranks. The rank r is chosen as $r = 0.01 \times m$, and $r = 0.05 \times m$, which correspond to 1%, and 5% of the problem size. Here, iter and time[s] are the number of iterations and the computational time in seconds, respectively; rank is the rank of $U_k V_k^\top$ given by the algorithms; and

$$\delta f_k := \|\mathscr{P}_\Omega(U_k V_k^\top) - B\|_F / \|B\|_F \quad \text{and} \quad \mathrm{NMAE} := C^{-1} \sum_{(i,j)\in\Omega} \left|(U_k V_k^\top)_{ij} - B_{ij}\right|,$$

are the relative objective residual; and the Normalized Mean Absolute Error, respectively, where $C := (\max_{i,j} B_{ij} - \min_{ij} B_{ij})|\Omega|$.

The results in Table 2 show that both Algorithms 1 and 2 produce similar results as LMaFit in terms of the relative objective residual and NMAE. When the rank is small (i.e., 1% of problem size), Algorithm 1 and LMaFit have similar number of iterations, but LMaFit has better computational time. When the rank is increasing up to 5% of the problem size, both Algorithm 1 and Algorithm 2 require a fewer iterations than LMaFit, and outperform this solver in terms of computational time. In this experiment, the number of iterations in Algorithm 2 is very similar in all the test cases, from 30 to 38 iterations, and similar to APGL. Note that we fix the rank in the first three algorithms, since APGL uses a convex approach, it cannot predict well an approximate rank if it is 5% of the problem size, or when the problem size is increasing.

Now, we add i.i.d. Gaussian noise $\mathcal{N}(0,\sigma\mathbb{I})$ with $\sigma = 0.01$ to B as $B := B^\natural + 5 \times \mathcal{N}(0,\sigma\mathbb{I})$, and only randomly take 30% observed entries. The convergence behavior of three algorithms for one problem instance with $m = n = 5000$ is plotted in Figure 5. When the rank $r = 0.01m$ (i.e., 1% of the problem size), LMaFit outperforms Algorithms 1 and 2 in terms of iterations, but when the rank $r = 0.025m$ (i.e., 2.5% of the problem size), Algorithms 1 and 2 are much better than LMaFit. Algorithm 1 works really well in the second case, and takes only 22 iterations. We also observe the monotone decrease in Algorithm 1 as guaranteed by our theory, but not in Algorithm 2.

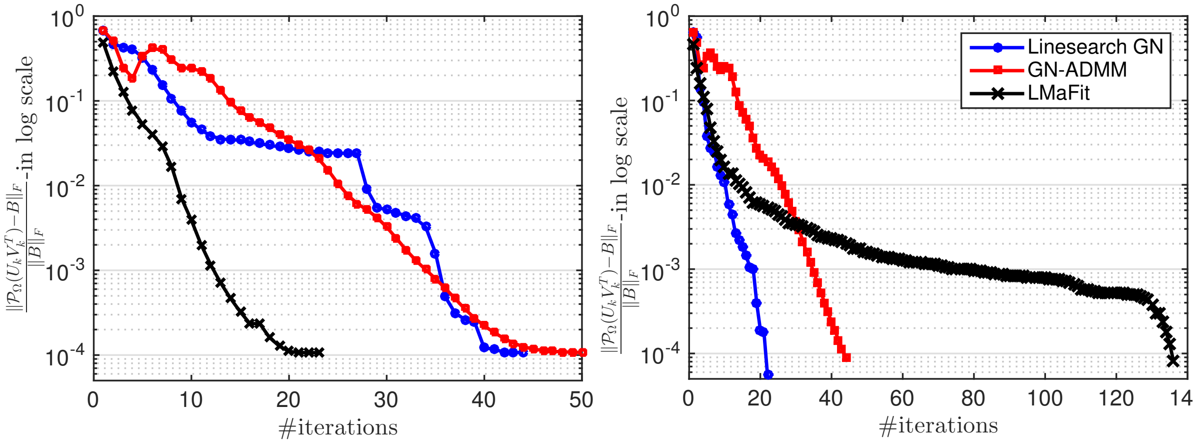

FIGURE 5. The convergence behavior of three algorithms ($m = n = 5000$) with noise ($\sigma = 0.01$) and 30% known entries (Left: $r = 0.01m$, Right: $r = 0.025m$).

Finally, we test three first algorithms on two problem instances with 30% observed entries in B and with i.i.d. Gaussian noise $\mathcal{N}(0,0.01\mathbb{I})$). The results of this test is reported in Table 3. LMaFit remains working well for then low-rank cases, while getting slower when the rank r increases. Algorithms 1 and 2 have similar performance in this case.

Real datasets. Now, we test three algorithms: Algorithms 1 and 2, and LMaFit on MovieLens and Jester jokes datasets available on `http://grouplens.org/datasets/movielens/`. For the MovieLens dataset, we test our algorithms on 5 problems: "movie-lens-latest (small)", "movie-lens" 100k, 1M, 10M, and 20M, which we abbreviate by "movie(s)", and "moviexM" in Table 4, respectively. We also test all problems in Jester joke dataset: "jester-1", "jester-2", "jester-3", and "jester-all".

In this test, since the data in "movie10M" and "movie20M" is sparse, we run the three algorithms on a MacBook laptop with a 2.6 GHz Intel Core i7 processor and 16GB memory. We

TABLE 3. Comparison of the four algorithms on synthetic integer datasets with noise.

			Algorithm 1					Algorithm 2				
m	n	r	iter	time[s]	δf_k	NMAE	rank	iter	time[s]	δf_k	NMAE	rank
1000	2000	10	40.8	3.56	5.33e-04	2.27e-04	10	40.5	2.24	5.32e-04	2.26e-04	10
1000	2000	25	49.1	5.50	6.05e-04	2.76e-04	25	62.0	4.62	2.09e-04	1.31e-04	25
5000	5000	50	19.5	39.82	1.10e-04	8.71e-05	50	45.0	67.70	1.09e-04	8.63e-05	50
5000	5000	125	16.6	56.70	7.90e-05	9.58e-05	125	40.0	103.45	9.50e-05	1.19e-04	125
			LMaFit [9]					APGL [43]				
m	m	r	iter	time[s]	δf_k	NMAE	rank	iter	time[s]	δf_k	NMAE	rank
1000	2000	10	31.6	1.71	5.31e-04	2.26e-04	10	28.0	3.08	6.34e-04	2.70e-04	10
1000	2000	25	121.0	8.55	2.08e-04	1.31e-04	25	30.7	5.86	9.87e-04	5.93e-04	25
5000	5000	50	20.0	30.39	1.07e-04	8.49e-05	50	28.5	52.96	4.97e-03	3.82e-03	48.2
5000	5000	125	48.0	121.11	7.19e-05	7.34e-05	125	31.3	46.99	1.92e-02	2.42e-02	5

TABLE 4. Summary of results of four the algorithms for MC on "real" datasets

			Algorithm 1			Algorithm 2			LMaFit [9]		
Name	m	n	iter	time[s]	δf_k	iter	time[s]	δf_k	iter	time[s]	δf_k
jester-1	24983	100	45	11.60	1.75e-01	59	11.35	1.75e-01	36	4.78	1.75e-01
jester-2	23500	100	41	9.93	1.77e-01	57	11.07	1.77e-01	34	5.51	1.77e-01
jester-3	24938	100	30	5.15	9.04e-04	32	4.94	9.66e-04	25	2.13	9.26e-04
jester-all	73421	100	48	35.12	1.65e-01	57	30.12	1.65e-01	36	12.21	1.65e-01
movie(s)	668	10325	200	16.69	1.64e-03	87	7.14	1.58e-03	200	44.18	1.58e-03
movie100k	943	1682	200	9.66	1.03e-02	84	4.87	1.00e-02	200	15.63	1.00e-02
movie1M	6040	3706	79	41.71	1.18e-01	42	21.96	1.19e-01	70	49.84	1.18e-01
movie10M	69878	10677	69	109.02	2.14e-01	33	38.32	2.15e-01	61	40.48	2.14e-01
movie20M	138493	26744	89	307.22	2.30e-01	37	117.23	2.30e-01	87	133.86	2.30e-01
			Algorithm 1			Algorithm 2			LMaFit [9]		
Name	m	n	rank	δx_k	NMAE	rank	δx_k	NMAE	rank	δx_k	NMAE
jester-1	24983	100	80	4.71e-01	2.29e-02	80	4.71e-01	2.36e-02	80	4.59e-01	2.30e-02
jester-2	23500	100	80	4.82e-01	2.35e-02	80	4.82e-01	2.42e-02	80	4.78e-01	2.39e-02
jester-3	24938	100	80	8.78e-04	1.08e-05	80	8.78e-04	7.22e-05	80	9.87e-04	2.08e-05
jester-all	73421	100	80	4.09e-01	1.95e-02	80	4.09e+01	2.05e-02	80	3.95e-01	1.98e-02
movie(s)	668	10325	100	1.36e-01	3.76e-04	100	1.36e-01	6.91e-04	100	1.36e-01	3.97e-04
movie100k	943	1682	100	1.10e-04	5.24e-03	100	1.10e-04	4.87e-03	100	1.00e-04	4.64e-03
movie1M	6040	3706	100	2.34e-01	8.28e-02	100	2.34e-01	8.44e-02	100	2.32e-01	8.32e-02
movie10M	69878	10677	20	5.86e-01	1.34e-01	20	5.86e-01	1.35e-01	20	5.81e-01	1.34e-01
movie20M	138493	26744	10	6.29e-01	1.42e-01	10	6.29e-01	1.42e-01	10	6.30e-01	1.42e-01

use `C-mex` routines in Matlab to compute $\mathscr{P}_\Omega(UV^\top)$ in three algorithms to avoid forming $U^\top V$. We terminates our algorithms based on the objective obtained from LMaFit such that the three algorithms have similar objective values.

The result is summarized in Table 4, where we add a new measurement defined by $\delta x_k := \frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\left|\lfloor (U_kV_k^\top)_{ij}\rfloor - B_{ij}\right|$ to measure the agreement ratio between the recovered matrix $M_k := U_kV_k^\top$ and the observed data B projected onto Ω. Due to our stopping criterion, three algorithms produce similar results in terms of the objective residuals, solution agreement, and NMAE. LMaFit works well on the Jester jokes dataset, but the computational time on these problems is relatively small. Algorithm 2 works well on Movielen dataset, especially for Movie 10MB and Movie 20MB. As mentioned previously, Algorithm 1 often achieve better solution in terms of accuracy if we run it long enough, while LMaFit and Algorithm 2 can be used to achieve a low or medium accurate solution for matrix completion.

6.6. **Robust low-rank matrix recovery.** We consider the following nonsmooth problem in low-rank matrix recovery:

$$\min_{U,V}\left\{\|UV^\top - B\|_1 \ : \ U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}\right\}, \tag{6.12}$$

where $\|Z\|_1 := \sum_{ij}|Z_{ij}|$ is the ℓ_1-norm of Z. This a low-rank matrix recovery problem with the ℓ_1-norm, which can be referred to as a robust recovery as opposed to the standard square loss. This formulation is often used in background extraction, see, e.g., [23].

Clearly, we can solve (6.12) using our ADMM-GN scheme above, which can be written as

$$\begin{cases} V_{k+1}^\top & := U_k^\dagger(B + W_k - \Lambda_k),\\ U_{k+1} & := U_k + \left(B + W_k - \Lambda_k - U_kV_{k+1}^\top\right)(V_k^\dagger)^\top,\\ W_{k+1} & := \mathrm{prox}_{\rho^{-1}\|\cdot\|_1}\left(U_{k+1}V_{k+1}^\top + \Lambda_k - B\right)\\ \Lambda_{k+1} & := \Lambda_k + (U_{k+1}V_{k+1}^\top - W_{k+1}). \end{cases} \tag{6.13}$$

We apply this scheme to solve the (6.12) using video surveillance datasets at http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html. We implement (6.13) in Algorithm 2 and compare it with the augmented Lagrangian method proposed in [23], which we denote by L1-LMaFit. We use the same strategy as in L1-LMaFit to update the penalty parameter ρ, while using $U_0 := [\mathbb{I}_r, 0_{(m-r)\times r}]$ and $V_0 := [\mathbb{I}_r, 0_{(n-r)\times r}]$ as an initial point. As suggested in [23], we choose the rank r to be $r = 1$ when testing gray-scale video data. As experienced, L1-LMaFit was based on alternating minimization idea, which can be saturated. Hence, we run both algorithm up to 100 iterations to observe the outcome. The computational time and the relative objective value $\|U_kV_k^\top - B\|_1/\|B\|_1$ of these two algorithms are reported in Table 5.

TABLE 5. Summary of results of the two algorithms for video background extraction.

Video data			Algorithm 2		L1-LMaFit [23]	
Video	Resolution	#Frames	Time	$\|U_kV_k^\top - B\|_1/\|B\|_1$	Time	$\|U_kV_k^\top - B\|_1/\|B\|_1$
Escalator	130×160	200	12.48	9.434063×10^{-2}	13.30	9.435117×10^{-2}
Fountain	128×160	200	13.27	4.197912×10^{-2}	13.71	4.198963×10^{-2}
Bootstrap	120×160	250	15.76	13.103802×10^{-2}	16.91	13.107209×10^{-2}
Curtain	128×160	250	18.43	2.965992×10^{-2}	25.45	2.969248×10^{-2}
Campus	128×160	300	24.83	9.315523×10^{-2}	30.10	9.316343×10^{-2}
Hall	144×176	300	31.63	5.708911×10^{-2}	39.05	5.709121×10^{-2}
ShoppingMall	256×320	350	82.22	4.442732×10^{-2}	85.48	4.442907×10^{-2}
WaterSurface	128×160	350	35.92	3.607625×10^{-2}	40.25	3.607747×10^{-2}

We can observe from Table 5 that the computational time in both algorithms is almost the same. This is consistent with our theoretical result, since the per-iteration complexity of the two algorithms is almost the same when we choose $r = 1$. However, Algorithm 2 provides a slightly better objective value since it still improves the objective when running further compared to L1-LMaFit. Here, we use the full-step variant of Algorithm 2, a fast convergence guarantee can be achieved when a good initial point is provided. This remains unclear in L1-LMaFit [23]. Unfortunately, global convergence of our variant as well as L1-LMaFit has not been known yet.

7. Conclusions

We have proposed a new Gauss-Newton scheme to approximate a stationary point of a class of low-rank matrix nonconvex optimization problems. Our method features several advantages from classical Gauss-Newton (GN) method such as fast local convergence, achieving high accuracy solutions compared to the well-known alternating minimization algorithm (AMA). We have proposed a linesearch GN algorithm and established its global and local convergence under standard assumptions. We have also specified this algorithm to the symmetric case, where AMA is not applicable. Then, we have combined our GN scheme with the alternating direction method of multipliers (ADMM) to design a new ADMM-GN algorithm that has global convergence guarantee and low per-iteration complexity. Several numerical experiments have been presented to demonstrate the theory and show the advantages of nonconvex optimization approaches. The theory and algorithms presented in this paper can be extended to different directions, including constrained low-rank matrix/tensor optimization.

Acknowledgments. The author was partly supported by the Office of Naval Research under Grant No. ONR-N00014-20-1-2088. The author would like to thank Dr. Zheqi Zhang for proving some Matlab codes to conduct the experiments in Subsection 6.2.

Appendix A. The proof of technical results

We provide the full proofs of all the technical results in the main text.

A.1. **The proof of Lemma 3.2: Closed form of Gauss-Newton direction.** Le us define $x := [\mathrm{vec}\left(D_V^\top\right), \mathrm{vec}\left(D_U\right)]$ and $b := [\mathrm{vec}(U^\top B), \mathrm{vec}(BV)]$. Then, we can write (3.5) as $\mathscr{B}x = b$, where $\mathscr{B} := \begin{bmatrix} \mathbb{I}_n \otimes U^\top U & V \otimes U^\top \\ V^\top \otimes U & V^\top V \otimes \mathbb{I}_m \end{bmatrix}$. We can show that

$$\mathscr{B} = \begin{bmatrix} (\mathbb{I}_n \otimes U^\top)(\mathbb{I}_n \otimes U) & (\mathbb{I}_n \otimes U^\top)(V \otimes \mathbb{I}_m) \\ (V^\top \otimes \mathbb{I}_m)(\mathbb{I}_n \otimes U) & (V^\top \otimes \mathbb{I}_m)(V \otimes \mathbb{I}_m) \end{bmatrix} = \begin{bmatrix} \mathbb{I}_n \otimes U^\top \\ V^\top \otimes \mathbb{I}_m \end{bmatrix} \begin{bmatrix} \mathbb{I}_n \otimes U & V \otimes \mathbb{I}_m \end{bmatrix}.$$

By [44, Fact. 7.4.24], we have $\mathrm{rank}\left([\mathbb{I}_n \otimes U, V \otimes \mathbb{I}_m]\right) \leq (m+n-r)r$. Hence, $\mathrm{rank}\left(\mathscr{B}\right)$ in (3.5) does not exceed $r(m+n-r) < r(m+n)$.

Next, we can rewrite $b = [(\mathbb{I}_n \otimes U^\top)\mathrm{vec}(B); (V^\top \otimes \mathbb{I}_m)\mathrm{vec}(B)]$. If we consider the extended matrix $\bar{\mathscr{B}} := [\mathscr{B}, b]$, then we can express it as

$$\bar{\mathscr{B}} = \begin{bmatrix} \mathbb{I}_n \otimes U^\top \\ V^\top \otimes \mathbb{I}_m \end{bmatrix} \begin{bmatrix} \mathbb{I}_n \otimes U & V \otimes \mathbb{I}_m & \mathrm{vec}(B) \end{bmatrix}.$$

This shows that $\mathrm{rank}\left(\bar{\mathscr{B}}\right) = \mathrm{rank}\left(\mathscr{B}\right)$. Hence, by the well-known consistency Rouché–Capelli theorem, (3.5) has a solution.

Now, we find the closed form (3.7). Since $\mathrm{rank}(U) = \mathrm{rank}(V) = r$, both $U^\top U$ and $V^\top V$ are invertible. Pre-multiplying the first equation of (3.5) by $(U^\top U)^{-1}$ and rearranging the result, we have

$$D_V^\top = (U^\top U)^{-1}U^\top(Z - D_U V^\top). \tag{A.1}$$

Substituting this expression into the second equation of (3.5) we get

$$(\mathbb{I} - U(U^\top U)^{-1}U^\top)D_V V^\top V = (\mathbb{I} - U(U^\top U)^{-1}U^\top)ZV. \tag{A.2}$$

Using the definition of the projections P_U, P_V, $P_U^\perp$ and $P_V^\perp$, we have from (A.2) that $P_U^\perp D_V V^\top V = P_U^\perp ZV$. Post-multiplying this expression by $(V^\top V)^{-1}$, we obtain

$$P_U^\perp D_V = P_U^\perp ZV(V^\top V)^{-1}. \tag{A.3}$$

Assume that $D_U := D_U^0 + U\hat{D}_r$, where D_U^0 is a given vector in the null space of $U^\top$, i.e., $U^\top D_U^0 = 0$, and $\hat{D}_r \in \mathbb{R}^{r\times r}$ is an arbitrary matrix. Substituting this expression into (A.3) and noting that $P_U^\perp U = 0$, we obtain

$$D_U^0 = D_U^0 - U(U^\top U)^{-1}U^\top D_U^0 + P_U^\perp U\hat{D}_r = P_U^\perp ZV(V^\top V)^{-1}.$$

Hence, we finally get

$$D_U = P_U^\perp ZV(V^\top V)^{-1} + U\hat{D}_r, \text{ for any } \hat{D}_r \in \mathbb{R}^{r\times r},$$

which is exactly the first term in (3.6). Substituting this D_U into (A.1) to yield the second term of (3.6) as

$$\begin{aligned} D_V^\top &= (U^\top U)^{-1}U^\top\left(Z - P_U^\perp ZV(V^\top V)^{-1}V^\top - U\hat{D}_r V^\top\right) \\ &= (U^\top U)^{-1}U^\top Z - U^\top U)^{-1}U^\top P_U^\perp ZV(V^\top V)^{-1}V^\top - \hat{D}_r V^\top \\ &= (U^\top U)^{-1}U^\top Z - \hat{D}_r V^\top. \end{aligned}$$

Since $\hat{D}_r$ is arbitrary in $\mathbb{R}^{r\times r}$, we choose $\hat{D}_r := \frac{1}{2}(U^\top U)^{-1}U^\top ZV(V^\top V)^{-1} \in \mathbb{R}^{r\times r}$. Substituting this choice into (3.6), we obtain

$$D_U = \left(\mathbb{I}_m - (1/2)P_U\right)ZV(V^\top V)^{-1} \quad \text{and} \quad D_V^\top = (U^\top U)^{-1}U^\top Z\left(\mathbb{I}_n - (1/2)P_V\right),$$

which is (3.7). Hence, the solution set of (3.5) forms an $(r\times r)$-linear subspace.

Finally, let us denote the residual term in the objective of (3.4) by $R(D_U, D_V) := UD_V^\top + D_U V^\top - Z$. Then, using the expression (3.6) we can easily show that

$$\begin{aligned} R(D_U, D_V) &= U((U^\top U)^{-1}U^\top Z - \hat{D}_r V^\top) + (P_U^\perp ZV(V^\top V)^{-1} + U\hat{D}_r)V^\top - Z \\ &= P_U Z + P_U^\perp Z P_V - P_U Z - P_U^\perp Z = P_U^\perp Z P_V - P_U^\perp Z. \end{aligned}$$

Hence, we can write

$$R(D_U, D_V) = -P_U^\perp Z P_V^\perp \quad \text{and} \quad (1/2)\|R(D_U, D_V)\|_F^2 = (1/2)\|P_U^\perp Z P_V^\perp\|_F^2.$$

The last term $(1/2)\|P_U^\perp Z P_V^\perp\|_F^2$ is the optimal value of (3.4). □

A.2. **The proof of Lemma 3.3: Descent property of GN algorithm.** Let us define $U(\alpha) := U + \alpha D_U$ and $V(\alpha) := V + \alpha D_V$ for $\alpha > 0$. Then

$$U(\alpha)V(\alpha)^\top = UV^\top + \alpha(UD_V^\top + D_U V^\top) + \alpha^2 D_U D_V^\top. \tag{A.4}$$

Let $W := UD_V^\top + D_U V^\top$ and $r(\alpha) := \|U(\alpha)V(\alpha)^\top - UV^\top - Z\|_F^2$. Using (A.4) we have

$$\begin{aligned} r(\alpha) &= \|\alpha(UD_V^\top + D_U V^\top) + \alpha^2 D_U D_V^\top - Z\|_F^2 \\ &= \|Z\|_F^2 + \alpha^2\|W\|_F^2 + \alpha^4\|D_U D_V^\top\|_F^2 + 2\alpha^3\langle W, D_U D_V^\top\rangle - 2\alpha\langle W, Z\rangle - 2\alpha^2\langle Z, D_U D_V^\top\rangle. \end{aligned}$$

Now, using the fact that

$$\langle W, Z - W\rangle = \langle UD_V^\top + D_U V^\top, P_U^\perp Z P_V^\perp\rangle = \mathrm{trace}\left((D_V U^\top + V D_U^\top) P_U^\perp Z P_V^\perp\right) = 0,$$

we can further expand $r(\alpha)$ as

$$\begin{aligned} r(\alpha) = {} & \|Z\|_F^2 - \alpha(2-\alpha)\|W\|_F^2 + \alpha^4\|D_U D_V^\top\|_F^2 \\ & - 2\alpha^2(1-\alpha)\langle W, D_U D_V^\top\rangle + 2\alpha^2\langle W - Z, D_U D_V^\top\rangle. \end{aligned} \tag{A.5}$$

Using the pseudo-inverse of U and V and $(V^\top)^\dagger U^\dagger = (UV^\top)^\dagger$, we can show that

$$D_U D_V^\top = \left(\mathbb{I}_m - 0.5P_U\right) Z (UV^\top)^\dagger Z \left(\mathbb{I}_n - 0.5P_V\right).$$

From the optimality condition (2.1) and the definition of $Z = -L_\Phi^{-1}\mathscr{A}^*\nabla\phi(\mathscr{A}(UV^\top) - B))$, we can show that $\nabla_U\Phi(U,V) = -L_\Phi U^\top Z$ and $\nabla_V\Phi(U,V) = -L_\Phi ZV$. However, since D_U and D_V are given by (3.7), we express

$$\begin{cases} D_U &= -\frac{1}{L_\Phi}\left(P_U^\perp + \frac{1}{2}P_U\right)\nabla_U\Phi(U,V)(V^\top V)^{-1}, \\ D_V^\top &= -\frac{1}{L_\Phi}(U^\top U)^{-1}\nabla_V\Phi(U,V)\left(P_V^\perp + \frac{1}{2}P_V\right). \end{cases}$$

Using this expression, we can write $\nu := \|D_U\|_F^2 + \|D_V\|_F^2$ as

$$\nu = \frac{1}{L_\Phi^2}\|\left(P_U^\perp + \frac{1}{2}P_U\right)\nabla_U\Phi(U,V)(V^\top V)^{-1}\|_F^2 + \frac{1}{L_\Phi^2}\|(U^\top U)^{-1}\nabla_V\Phi(U,V)\left(P_V^\perp + \frac{1}{2}P_V\right)\|_F^2.$$

Hence, we can estimate

$$\frac{\|\nabla_U\Phi(U,V)\|_F^2}{4L_\Phi^2(\sigma_{\max}(V))^4} + \frac{\|\nabla_V\Phi(U,V)\|_F^2}{4L_\Phi^2(\sigma_{\max}(U))^4} \le \nu \le \frac{\|\nabla_U\Phi(U,V)\|_F^2}{L_\Phi^2(\sigma_{\min}(U))^4} + \frac{\|\nabla_V\Phi(U,V)\|_F^2}{L_\Phi^2(\sigma_{\min}(V))^4}, \tag{A.6}$$

where $\sigma_{\min}(\cdot)$ and $\sigma_{\max}(\cdot)$ are the smallest and largest singular values of $(\cdot)$, respectively. Let $\sigma_{\max} := \max\{\sigma_{\max}(U), \sigma_{\max}(V)\}$ and $\sigma_{\min} := \min\{\sigma_{\min}(U), \sigma_{\min}(V)\}$. Using $\|\nabla\Phi(U,V)\|_F^2 = \|\nabla_U\Phi(U,V)\|_F^2 + \|\nabla_V\Phi(U,V)\|_F^2$, (A.6) leads to

$$\frac{\|\nabla\Phi(U,V)\|_F^2}{4L_\Phi^2\sigma_{\max}^4} \le \nu = \|D_U\|_F^2 + \|D_V\|_F^2 \le \frac{\|\nabla\Phi(U,V)\|_F^2}{L_\Phi^2\sigma_{\min}^4}. \tag{A.7}$$

Next, using the orthonormality, we estimate $\|W\|_F^2$ as follows:

$$\begin{aligned} \|W\|_F^2 &= \|UD_V^\top + D_U V^\top\|_F^2 = \|UD_V^\top\|_F^2 + \|D_U V^\top\|_F^2 \\ &= \mathrm{trace}\left(D_V(U^\top U)D_V^\top\right) + \mathrm{trace}\left(D_U(V^\top V)D_U^\top\right) \\ &\ge (\sigma_{\min}(U))^2\|D_U\|_F^2 + (\sigma_{\min}(V))^2\|D_V\|_F^2 \\ &\ge \sigma_{\min}^2\nu. \end{aligned} \tag{A.8}$$

On the one hand, we estimate individually each term of the expression (A.5) as follows:

$$\|D_UD_V^\top\|_F^2 = \operatorname{trace}\left((D_UD_V^\top)^\top(D_UD_V)\right) \le \frac{1}{4}\left(\|D_U\|_F^2 + \|D_V\|_F^2\right)^2 = \frac{\nu^2}{4}.$$

On the other hand, since $W - Z = -P_U^\perp Z P_V^\perp$ by Lemma 3.2, we can show that

$$\langle W - Z, D_UD_V^\top\rangle = \langle P_U^\perp Z P_V^\perp, D_UD_V^\top\rangle = \operatorname{trace}\left(D_V^\top P_V^\perp Z^\top P_U^\perp D_U\right) \le \|Z\|_F\|D_UD_V^\top\|_F.$$

In addition, $-\langle W, D_UD_V\rangle \le \|W\|_F\|D_UD_V^\top\|_F$. Substituting these estimates into (A.5) and using the fact that $2-\alpha \ge 1$ and $1-\alpha \le 1$ we obtain

$$\begin{aligned} r(\alpha) \ \le\ & \|Z\|_F^2 - \alpha\|W\|_F^2 + \tfrac{\nu^2\alpha^4}{4} + 2\alpha^2\|W\|_F\|D_UD_V^\top\|_F + 2\alpha^2\|Z\|_F\|D_UD_V^\top\|_F \\ \le\ & \|Z\|_F^2 - \tfrac{\alpha}{16}\|W\|_F^2 - \tfrac{\alpha\nu\sigma_{\min}^2}{16} + \tfrac{\nu^2\alpha^4}{4} - \tfrac{\alpha}{2}\|W\|_F^2 + 2\alpha^2\|W\|_F\|D_UD_V^\top\|_F \\ & - \tfrac{3\alpha}{8}\|W\|_F^2 + 2\alpha^2\|Z\|_F\|D_UD_V^\top\|_F \\ =\ & \|Z\|_F^2 - \tfrac{\alpha\nu\sigma_{\min}^2}{16} - \tfrac{\alpha\nu}{16}\left(\sigma_{\min}^2 - 4\alpha^3\nu\right)_{[b_1]} - \tfrac{\alpha}{2}\|W\|_F\left(\|W\|_F - 4\alpha\|D_UD_V^\top\|_F\right)_{[b_2]} \\ & - \tfrac{\alpha}{8}\left(3\|W\|_F^2 - 16\alpha\,\|Z\|_F\,\left\|D_UD_V^\top\right\|_F\right)_{[b_3]} \\ =\ & \|Z\|_F^2 - \tfrac{\alpha\nu\sigma_{\min}^2}{16} - \tfrac{\alpha\nu}{16}b_1 - \tfrac{\alpha}{2}\|W\|_F b_2 - \tfrac{\alpha}{8}b_3. \end{aligned} \tag{A.9}$$

We estimate each term in (A.9). From (A.8), we can see that $W = 0$ implies $D_U = 0$ and $D_V = 0$, which is contradict to our assumption. Hence, $W \ne 0$. First, we choose $\alpha \in (0,1]$ such that

$$\sigma_{\min}^2 \ge \left(\frac{16\|Z\|_F}{3\|W\|_F}\right)^2 \nu\alpha^2 \quad \text{and} \quad \sigma_{\min}^2 \ge 4\nu\alpha^2. \tag{A.10}$$

Since $W \ne 0$, this condition allows us to compute α as

$$0 < \alpha \le \frac{\sigma_{\min}}{2\sqrt{\nu}}\min\left\{1, \frac{3\|W\|_F}{8\|Z\|_F}\right\}. \tag{A.11}$$

Under the second condition of (A.10) and $\alpha \in (0,1]$, we have $b_1 = \sigma_{\min}^2 - 4\nu\alpha^3 \ge \sigma_{\min}^2 - 4\nu\alpha^2 \ge 0$. Next, since (A.8) and the first condition in (A.10) we have $\sigma_{\min}^2 \ge 4\alpha^2\nu$. Using (A.8) we have $\|W\|_F^2 \ge \sigma_{\min}^2\nu \ge 4\alpha^2\nu^2 = 4\alpha^2\left(\|D_U\|_F^2 + \|D_V\|_F^2\right)^2 \ge 16\alpha^2\left\|D_UD_V^\top\right\|_F^2$. Hence, $\|W\|_F \ge 4\alpha\left\|D_UD_V^\top\right\|_F$. This inequality leads to $b_2 = \|W\|_F - 4\alpha\|D_UD_V^\top\|_F \ge 0$.

Now, using $\|W\|_F^2 \ge \sigma_{\min}^2\nu \ge \left(\frac{16\|Z\|_F}{3\|W\|_F}\right)^2\alpha^2\nu^2$, we have $\|W\|_F \ge \frac{16\|Z\|_F}{3\|W\|_F}\nu\alpha$. Therefore, we can estimate

$$b_3 = 3\|W\|_F^2 - 16\alpha\|Z\|_F\|D_UD_V^\top\|_F \ge 3\alpha\|W\|_F\frac{16\|Z\|_F}{3\|W\|_F} - 16\alpha\|Z\|_F\|D_UD_V^\top\|_F = 0.$$

From (A.7) we have $\sqrt{\nu} \le \frac{\|\nabla\Phi(U,V)\|_F}{L_\Phi\sigma_{\min}^2}$, while from (A.8) we have

$$\|W\|_F \ge \sqrt{\nu}\sigma_{\min} \ge \frac{\sigma_{\min}\|\nabla\Phi(U,V)\|_F}{2L_\Phi\sigma_{\max}^2}.$$

Substituting these estimates into (A.11) of α and using $\|Z\|_F = \frac{1}{L_\Phi}\left\|\Phi'(UV^\top)\right\|_F$ we can lower estimate α as

$$0 < \alpha \le \frac{\sigma_{\min}^3 L_\Phi}{2\|\nabla\Phi(U,V)\|_F}\min\left\{1, \frac{3\sigma_{\min}\|\nabla\Phi(U,V)\|_F}{16L_\Phi\|\Phi'(UV^\top)\|_F\sigma_{\max}^2}\right\}. \tag{A.12}$$

Note that $\alpha \in (0,1]$, we obtain from (A.12) the update rule (3.9).

We finally estimate (3.10). Since α satisfies (3.9), it follows from (A.9) that

$$r(\alpha) \le \|Z\|_F^2 - \frac{\alpha\nu\sigma_{\min}^2}{16} \overset{(A.7)}{\le} \|Z\|_F^2 - \frac{\alpha\sigma_{\min}^2}{64L_\Phi^2\sigma_{\max}^4}\|\nabla\Phi(U,V)\|^2$$

Substituting this inequality into (3.1) we obtain (3.10). □

A.3. **The proof of Lemma 3.4: Full-rankness of iterates.** Since $\operatorname{rank}(U) = \operatorname{rank}(V) = r$ by assumption, we have $\lambda_{\min}(U^\top U) > 0$ and $\lambda_{\min}(V^\top V) > 0$. We consider $Q := (U^\dagger)^\top = U(U^\top U)^{-1}$ and $S := (V^\dagger)^\top = V(V^\top V)^{-1}$. We always have $U_+^\top(\lambda_{\max}(QQ^\top)\mathbb{I} - QQ^\top)U_+ \succeq 0$. This implies that

$$\lambda_{\min}(U_+^\top U_+)\lambda_{\max}(QQ^\top) \ge \lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+)).$$

Clearly, since $Q^\top = U^\dagger$, we have $\lambda_{\max}(QQ^\top) = \lambda_{\min}^{-1}(U^\top U)$. Using this relation into the last inequality, we get

$$\frac{\lambda_{\min}(U_+^\top U_+)}{\lambda_{\min}(U^\top U)} \ge \lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+)). \tag{A.13}$$

Hence, it is sufficient to show that $\lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+)) > 0$. By Lemma 3.2, we have $U_+ = U + \alpha D_U = U + \alpha(P_U^\perp + 0.5P_U)ZV(V^\top V)^{-1}$. Therefore, we can compute $Q^\top U_+ = \mathbb{I}_m + 0.5\alpha H$, where $H := (U^\top U)^{-1}U^\top ZV(V^\top V)^{-1}$. Then, we estimate $\lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+))$ as follows:

$$\begin{aligned}
\lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+)) &= \lambda_{\min}\left(\mathbb{I} + 0.5\alpha(H^\top + H) + \alpha^2 H^\top H\right) \\
&\overset{[44,\, 9.13.6.]}{\ge} 1 - 0.5\alpha\lambda_{\max}(H^\top + H) \overset{[44,\, 5.11.25]}{\ge} 1 - \alpha\sigma_{\max}(H) \\
&= 1 - \alpha\sigma_{\max}\left((U^\top U)^{-1}U^\top ZV(V^\top V)^{-1}\right) \\
&\ge 1 - \frac{\alpha\sigma_{\max}(U^\top Z)}{\sigma_{\min}(U)^2\sigma_{\min}(V)} \ge 1 - \frac{\alpha\|U^\top Z\|_F}{\sigma_{\min}^3},
\end{aligned}$$

where $\sigma_{\min} = \min\{\sigma_{\min}(U), \sigma_{\min}(V)\}$ and $\|U^\top Z\|_F \ge \sigma_{\max}(U^\top Z)$. We note that $\|\Phi(U,V)\|_F \ge \|\nabla_U\Phi(U,V)\|_F = L_\Phi\|U^\top Z\|$. Substituting this estimate into the last inequality and noting from (3.9) that $\alpha \le \frac{\sigma_{\min}^3 L_\Phi}{2\|\nabla\Phi(U,V)\|_F}$, we obtain

$$\lambda_{\min}((Q^\top U_+)^\top(Q^\top U_+)) \ge 1 - \alpha\frac{\|\nabla\Phi(U,V)\|_F}{L_\Phi\sigma_{\min}^3} \ge 1 - \frac{L_\Phi}{2L_\Phi} = \frac{1}{2} > 0.$$

Combining this estimate and (A.13) we have $\lambda_{\min}(U_+^\top U_+) \ge 0.5\lambda_{\min}(U^\top U)$. Hence, we conclude that $\operatorname{rank}(U_+) = r$. With a similar proof, we can show that $\operatorname{rank}(V_+) = r$. □

A.4. **The proof of Theorem 3.1: Global convergence of GN method.** By Lemma 3.3, we can see that the backtracking linesearch step at Step 5 of Algorithm 1 is finite and $\alpha_k > 0$. The inequality (3.12) guarantees that $\Phi(U_{k+1}, V_{k+1}) < \Phi(U_k, V_k)$. Hence, the sequence $\{\Phi(U_k, V_k)\}$ is decreasing and bounded from below by $\Phi^\star$. It converges to a limit point Φ^*. Now, using (3.12) we obtain

$$\sum_{k=0}^{n} \alpha_k \|\nabla\Phi(U_k, V_k)\|_F^2 \le \Phi(U_0, V_0) - \Phi(U_{n+1}, V_{n+1}) \le \Phi(U_0, V_0) - \Phi^\star < +\infty.$$

Taking the limit in this inequality as $n \to \infty$, we obtain $\sum_{k=0}^{\infty} \alpha_k \|\nabla\Phi(U_k, V_k)\|_F^2 < +\infty$. Consequently, $\lim_{k\to\infty} \alpha_k \|\nabla\Phi(U_k, V_k)\|_F^2 = 0$. This proves the first part (3.15).

In order to prove the second part, we need to show that $\alpha_k \ge \alpha > 0$ for all k sufficiently large. Indeed, by our Assumption A.2.1(a), the sublevel set $\mathscr{F}_\Phi(\Phi(U_0, V_0))$ of Φ is bounded. By the descent inequality (3.12), it is clear that $\{[U_k, V_k]\}$ is in $\mathscr{F}_\Phi(\Phi(U_0, V_0))$ and therefore bounded. Hence, $\|\Phi'(U_k, V_k)\|_F \le K_1 < +\infty$. Similarly, $\|\nabla\Phi(U_k, V_k)\|_F \le K_2 < +\infty$ and $\max\{\sigma_{\max}(U_k), \sigma_{\max}(V_k)\} \le K_3 < +\infty$. Using these arguments and condition (3.14) into (3.9), we obtain

$$2\alpha_k \ge \underline{\alpha} \overset{(3.9)}{\ge} 2\alpha := \min\left\{1, \frac{L_\Phi \underline{\sigma}^3}{2K_2}, \frac{3\underline{\sigma}^4}{32K_1K_3^2}\right\} > 0.$$

Using this lower bound into (3.15) we have $\lim\limits_{k\to\infty} \|\nabla\Phi(U_k, V_k)\|_F^2 \le \alpha^{-1} \lim\limits_{k\to\infty} \alpha_k \|\nabla\Phi(U_k, V_k)\|_F^2 = 0$, which implies (3.16).

As shown above, $\{X_k\}$ generated by Algorithm 1 is bounded. Hence, there exists a limit point $X_\star := [U_\star, V_\star]$. Passing through the limit (3.16) via subsequence, we can see that $\nabla\Phi(U_\star, V_\star) = 0$, and hence, $X_\star$ satisfies the optimality condition (2.1). □

A.5. **The proof of Lemma 4.1: Descent property of $\mathscr{L}_\rho$.** We first prove part (a). Since $\{[U_k, V_k]\}$ is bounded by the boundedness of the sublevel set $\mathscr{F}_\Phi(\Phi(U_0, V_0))$ as in the proof of Theorem 3.1, and since $\lim\limits_{k\to\infty} \|W_k - \mathscr{A}(U_k V_k^\top) + B\|_F = 0$ due to part (b), $\{W_k\}$ is also bounded.

Now, we prove part (b) for **Option 1**. First, since $[U_{k+1}, V_{k+1}]$ is updated by Step 5 of Algorithm 2 that satisfies the backtracking linesearch condition (4.10), we have

$$\mathscr{Q}_k(U_{k+1}, V_{k+1}) \le \mathscr{Q}_k(U_k, V_k) - 0.5c_1\alpha_k\Delta_k^2, \tag{A.14}$$

where $\mathscr{Q}_k$ is defined by (4.10), $E_k := \mathscr{A}(U_k V_k^\top) - B + \rho^{-1}\Lambda_k$, and Δ_k^2 is

$$\Delta_k^2 := \|U_k^\top \mathscr{A}^*(E_k - W_k)\|_F^2 + \|\mathscr{A}^*(E_k - W_k)V_k\|_F^2. \tag{A.15}$$

This condition implies

$$\mathscr{L}_\rho(U_{k+1}, V_{k+1}, W_k, \Lambda_k) \le \mathscr{L}_\rho(U_k, V_k, W_k, \Lambda_k) - \frac{c_1\rho\alpha_k}{2}\Delta_k^2. \tag{A.16}$$

Second, we consider the objective function $h(W) := \phi(W) + (\rho/2)\|W - C_k\|_F^2$ of (4.3b), where $C_k := \mathscr{A}(U_{k+1}V_{k+1}) - B + \rho^{-1}\Lambda_k$. Since $h(\cdot)$ is strongly convex with the strong convexity parameter $\rho + \mu_\phi$, and W_{k+1} is the optimal solution of h, we have

$$h(W_{k+1}) \le h(W_k) - ((\rho + \mu_\phi)/2)\|W_{k+1} - W_k\|_F^2.$$

Using this inequality, and the definition of h and $\mathscr{L}_\rho$, we can show that

$$\mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_k) \le \mathscr{L}_\rho(U_{k+1},V_{k+1},W_k,\Lambda_k) - \frac{(\rho+\mu_\phi)}{2}\|W_{k+1}-W_k\|_F^2. \tag{A.17}$$

In addition, since ϕ is L_ϕ-smooth, we can write down the optimality condition of (4.3b) as $\nabla\phi(W_{k+1}) + \rho(W_{k+1}-C_k) = 0$. Using the definition of C_k and (4.3c) we get $\Lambda_{k+1} = \nabla\phi(W_{k+1})$. Hence, we can derive

$$\|\Lambda_{k+1}-\Lambda_k\|_F = \|\nabla\phi(W_{k+1}) - \nabla\phi(W_k)\|_F \le L_\phi\|W_{k+1}-W_k\|_F, \tag{A.18}$$

which is the first inequality in (4.12). The boundedness of $\{\Lambda_k\}$ also follows from the relation $\Lambda_{k+1} = \nabla\phi(W_{k+1})$ and the boundedness of $\{W_k\}$.

Third, since Λ_k is updated by (4.3c), using the definition of $\mathscr{L}_\rho$, it is easy to show that

$$\begin{aligned}\mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_{k+1}) &= \mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_k) + \rho^{-1}\|\Lambda_{k+1}-\Lambda_k\|_F^2 \\ &\overset{(A.18)}{\le} \mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_k) + \rho^{-1}L_\phi^2\|W_{k+1}-W_k\|_F^2.\end{aligned} \tag{A.19}$$

Summing up (A.16), (A.17) and (A.19) we get (4.13).

Finally, we prove (b) for **Option 2**. We consider the gradient step (4.8) instead of (4.3b). Using the optimality condition of (4.7) and (4.3c), we can derive $\Lambda_{k+1} = \nabla\phi(W_k) + L_\phi(W_{k+1} - W_k)$. Using this relation and the Lipschitz continuity of $\nabla\phi$, we have

$$\begin{aligned}\|\Lambda_{k+1}-\Lambda_k\|_F &= \|L_\phi(W_{k+1}-W_{k-1}) + \nabla\phi(W_k) - \nabla\phi(W_{k-1})\|_F \\ &\le L_\phi\left[\|W_{k+1}-W_{k-1}\|_F + \|W_k - W_{k-1}\|_F\right],\end{aligned} \tag{A.20}$$

which is exactly the second expression of (4.12). Using $\Lambda_{k+1} = \nabla\phi(W_k) + L_\phi(W_{k+1}-W_k)$, similar above, we can also show the boundedness of $\{\Lambda_k\}$.

Now, since we apply the gradient step to solve (4.3b), with h defined as in (A.17), it is well-known that

$$h(W_{k+1}) \le h(W_k) - ((L_\phi+\rho)/2)\|W_{k+1}-W_k\|_F^2,$$

which implies

$$\mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_k) \le \mathscr{L}_\rho(U_{k+1},V_{k+1},W_k,\Lambda_k) - \frac{(\rho+L_\phi)}{2}\|W_{k+1}-W_k\|_F^2. \tag{A.21}$$

Summing up (A.16), (A.21) and the first equality of (A.19) we obtain

$$\mathscr{L}_\rho(U_{k+1},V_{k+1},W_{k+1},\Lambda_{k+1}) = \mathscr{L}_\rho(U_k,V_k,W_k,\Lambda_k) - (1/2)c_1\rho\alpha_k\Delta_k^2 - T_k, \tag{A.22}$$

where $T_k := \frac{(\rho+L_\phi)}{2}\|W_{k+1}-W_k\|_F^2 - \rho^{-1}\|\Lambda_{k+1}-\Lambda_k\|_F^2$. Finally, using (A.20), we can estimate $\|\Lambda_{k+1}-\Lambda_k\|_F$ as follows:

$$\begin{aligned}\|\Lambda_{k+1}-\Lambda_k\|_F^2 &\le L_\phi^2\left[\|W_{k+1}-W_{k-1}\|_F + \|W_k-W_{k-1}\|_F\right]^2 \\ &\le 2L_\phi^2\|W_{k+1}-W_k\|_F^2 + 4L_\phi^2\|W_k-W_{k-1}\|_F^2.\end{aligned}$$

Hence, $T_k \ge (0.5(\rho+L_\phi) - 2\rho^{-1}L_\phi^2)\|W_{k+1}-W_k\|_F^2 - 4\rho^{-1}L_\phi^2\|W_k-W_{k-1}\|_F^2$. Substituting this estimate of T_k into (A.22) we obtain (4.13). □

A.6. **The proof of Theorem 4.1: Global convergence of ADMM-GN.** We first prove for **Option 1**. Let us define $\eta := \rho^{-1}(\rho^2 + \mu_\phi\rho - 2L_\phi^2)$. Then, $\eta > 0$ if we choose $\rho > 0.5((\mu_\phi + 8L_\phi^2)^{1/2} + \mu_\phi)$ as given by (4.14) in Lemma 4.1. Hence, the sequence $\left\{\mathscr{L}_\rho(U_k, V_k, W_k, \Lambda_k)\right\}$ is strictly decreasing, it is bounded from bellow due to Assumption A.2.1 and the boundedness of $\{(U_k, V_k, W_k, \Lambda_k]\}$. It converges to a finite value $\mathscr{L}_\rho^\star$. In addition, (4.13) implies

$$\begin{aligned}
&\lim_{k\to\infty} \|W_{k+1} - W_k\|_F = 0,\\
&\lim_{k\to\infty} \alpha_k\big\|U_k^\top \mathscr{A}^*\big(\rho^{-1}\Lambda_k + \mathscr{A}(U_kV_k^\top) - B - W_k\big)\big\|_F^2 = 0, \quad \text{and}\\
&\lim_{k\to\infty} \alpha_k\big\|\mathscr{A}^*\big(\rho^{-1}\Lambda_k + \mathscr{A}(U_kV_k^\top) - B - W_k\big)V_k\big\|_F^2 = 0.
\end{aligned} \tag{A.23}$$

Under condition (3.14), similar to the proof of Theorem 3.1 we can show that $\alpha_k \geq 0.5\underline{\alpha} > 0$ for k sufficiently large. Hence, the two last limits of (A.23) imply

$$\begin{aligned}
&\lim_{k\to\infty} \big\|U_k^\top \mathscr{A}^*\big(\Lambda_k + \rho\big(\mathscr{A}(U_kV_k^\top) - B - W_k\big)\big)\big\|_F = 0 \quad \text{and}\\
&\lim_{k\to\infty} \big\|\mathscr{A}^*\big(\Lambda_k + \rho\big(\mathscr{A}(U_kV_k^\top) - B - W_k\big)V_k\big\|_F = 0.
\end{aligned} \tag{A.24}$$

On the other hand, using Lemma 4.1(a), (4.3c), and the first limit in (A.23), we obtain

$$\begin{aligned}
\lim_{k\to\infty}\|\mathscr{A}(U_{k+1}V_{k+1}^\top) - W_{k+1} - B\|_F &\overset{(4.3c)}{=} \rho^{-1}\lim_{k\to\infty}\|\Lambda_{k+1} - \Lambda_k\|_F\\
&\overset{\text{Lemma A.5(a)}}{\leq} \rho^{-1}L_\phi \lim_{k\to\infty}\|W_{k+1} - W_k\|_F \overset{(A.23)}{=} 0.
\end{aligned} \tag{A.25}$$

We consider a convergent subsequence $\{[U_{k_i}, V_{k_i}]\}_{i\in\mathbb{N}}$ with the limit $[U_*, V_*]$. Then, the limit (A.25) shows that the corresponding subsequence $\{W_{k_i}\}$ also converges to W_* such that $W_* = \mathscr{A}(U_*V_*^\top) - B$, which is the last condition in (4.11).

Now, using the limit in (A.25) and combining with the triangle inequality, we get

$$\begin{aligned}
\|U_k^\top\mathscr{A}^*(\Lambda_k)\|_F &\leq \big\|U_k^\top\mathscr{A}^*\big(\Lambda_k + \rho\big(\mathscr{A}(U_kV_k^\top) - B - W_k\big)\big)\big\|_F\\
&\quad + \rho\big\|U_k^\top\mathscr{A}^*\big(\mathscr{A}(U_kV_k^\top) - B - W_k\big)\big\|_F \overset{(A.24),(A.25)}{\rightarrow} 0 \quad \text{as} \quad k_i \to \infty.
\end{aligned}$$

This implies $U_*^\top\mathscr{A}^*(\Lambda_*) = 0$ via subsequence. Similarly, we can also show that $\mathscr{A}^*(\Lambda_*)V_* = 0$. These are the second and the third conditions in (4.11). Finally, the first condition of (4.11) follows directly from the relation $\Lambda_k = \nabla\phi(W_k)$ as the optimality condition of (4.3b) by taking the limit via subsequence.

We have shown in the above steps that the limit point $(U_*, V_*, W_*, \Lambda_*)$ satisfies the optimality condition (4.11) of (4.1). By eliminating Λ_* and W_* in (4.11) we obtain (2.1), which shows that any limit point $[U_*, V_*]$ of $\{[U_k, V_k]\}$ is a stationary point of (1.1). The proof of (4.16) can be done as in Theorem 3.1.

We prove for **Option 2**. We note that if $\rho > 3L_\phi$, then we can examine from (4.14) that $\eta_1 > \eta_0$. If we denote by $\mathscr{L}_k := \mathscr{L}_\rho(U_k, V_k, W_k, \Lambda_k)$ and $r_k := \|W_k - W_{k-1}\|_F$ for $k \geq 1$, then we can write (4.13) as

$$\mathscr{L}_{k+1} + \frac{\eta_0}{2}r_{k+1}^2 \leq \mathscr{L}_k + \frac{\eta_0}{2}r_k^2 - \frac{c_1\rho}{2}\Delta_k^2 - \frac{(\eta_1 - \eta_0)}{2}r_{k+1}^2.$$

By induction, we can show from this inequality that $\sum_{k=0}^\infty\left(c_1\rho\Delta_k^2 + (\eta_1 - \eta_0)r_{k+1}^2\right) = 0$, which implies (A.23). With the same proof as in **Option 1** we obtain the same conclusions of the theorem as in **Option 1**. □

A.7. **The proof of Theorem 3.2: Local convergence of GN method.** Let us define $x := [\mathrm{vec}\,(U), \mathrm{vec}\,(V^\top)] \in \mathbb{R}^{(m+n)r}$ the vecterization of U and V, and $R(x) := \mathscr{A}(UV^\top) - B$ the residual term. We can compute the Jacobian $J_R(x)$ of R at x as $J_R(x) = A[V \otimes \mathbb{I}_m, \mathbb{I}_n \otimes U^\top] \in \mathbb{R}^{l\times(m+n)r}$, where A is the matrix form of the linear operator $\mathscr{A}$. The objective function $\Phi(U,V)$ can be written as $\Phi(x) = \phi(R(x))$. Its gradient and Hessian are given by

$$\begin{cases} \nabla\Phi(x) &= J_R(x)^\top \nabla\phi(R(x)) \quad \text{and} \\ \nabla^2\Phi(x) &= J_R(x)^\top \nabla^2\phi(R(x)) J_R(x) + \displaystyle\sum_{i=1}^{l} \frac{\partial\phi(R(x))}{\partial R_i} \nabla^2 R_i(x). \end{cases} \tag{A.26}$$

First, we show that under Assumption A.3.1(a), $\nabla^2\Phi$ is also Lipschitz continuous in $\mathcal{N}(x_\star)$ of $x_\star \in \mathscr{X}_\star$. Indeed, $\nabla^2 R(x)$ is bounded in $\mathcal{N}(x_\star)$ by $M_{R_i''}$, and $\nabla^2 R_i(\cdot)$ is Lipschitz continuous with the Lipschitz constant $L_{R_i''}$. In addition, $J_R(\cdot)$ is also bounded in $\mathcal{N}(x_\star)$ by $M_{R'}$, and $R(\cdot)$ is also Lipschitz continuous with the Lipschitz constant L_R. Since $\nabla^2\phi$ is Lipschitz continuous in $\mathcal{N}(R(x_\star))$, $\frac{\partial\phi(R(x)}{\partial R_i}$ is also bounded by $M^i_{\phi'}$, and Lipschitz continuous in $\mathcal{N}(R(x_\star))$ with the Lipschitz constant $L^i_{\phi'}$. Combining these statements and (A.26), we can show that for any $x, \hat{x} \in \mathcal{N}(x_\star)$, the following estimate holds:

$$\begin{aligned} \|\nabla^2\Phi(x) - \nabla^2\Phi(\hat{x})\| &\leq \left\| J_R(x)^\top \nabla^2\phi(R(x)) J_R(x) - J_R(\hat{x})^\top \nabla^2\phi(R(\hat{x})) J_R(\hat{x}) \right\| \\ &\quad + \left\| \textstyle\sum_{i=1}^{l} \left[\frac{\partial\phi(R(x))}{\partial R_i} \nabla^2 R_i(x) - \frac{\partial\phi(R(\hat{x}))}{\partial R_i} \nabla^2 R_i(\hat{x}) \right] \right\| \\ &\leq \left\| J_R(x)^\top \nabla^2\phi(R(x)) \left(J_R(x) - J_R(\hat{x})\right) \right\| \\ &\quad + \left\| J_R(x)^\top \left(\nabla^2\phi(R(x)) - \nabla^2\phi(R(\hat{x})) \right) J_R(\hat{x}) \right\| \\ &\quad + \left\| \left(J_R(x) - J_R(\hat{x})\right)^\top \nabla^2\phi(R(\hat{x})) J_R(\hat{x}) \right\| \\ &\quad + \textstyle\sum_{i=1}^{l} \left[\left\| \frac{\partial\phi(R(x))}{\partial R_i} \left(\nabla^2 R_i(x) - \nabla^2 R_i(\hat{x}) \right) \right\| \right. \\ &\quad + \left. \left\| \left(\frac{\partial\phi(R(x))}{\partial R_i} - \frac{\partial\phi(R(\hat{x}))}{\partial R_i} \right) \nabla^2 R_i(\hat{x}) \right\| \right] \\ &\leq \left(2M_{R'}M_{\phi''}L_{R'} + M_{R'}^2 L_{\phi''} + \textstyle\sum_{i=1}^{l} (M_{R_i''} L^i_{\phi'} + L_{R_i''} M^i_{\phi'}) \right) \|x - \hat{x}\|. \end{aligned}$$

This inequality shows that the Hessian $\nabla^2\Phi$ is Lipschitz continuous in $\mathcal{N}(x_\star)$ with the Lipschitz constant $L_{\Phi''} := 2M_{R'}M_{\phi''}L_{R'} + M_{R'}^2 L_{\phi''} + \sum_{i=1}^{l}(M_{R_i''} L^i_{\phi'} + L_{R_i''} M^i_{\phi'}) > 0$.

Next, we consider the GN direction D_{X_k} in (3.4). Let $d := [\mathrm{vec}\,(D_U), \mathrm{vec}\,(D_V^\top)]$ and $H_0(x) := \begin{bmatrix} V^\top \otimes U & V^\top V \otimes \mathbb{I}_m \\ \mathbb{I}_n \otimes U^\top U & V \otimes U^\top \end{bmatrix}$. Due to the full-rankness of U and V, by using the result in [45] we can show that $H_0(x)^\dagger$ is bounded by M_h, i.e.:

$$\|H_0(x)^\dagger\| \leq M_h < +\infty, \quad \forall x \in \mathcal{N}(x_\star), \tag{A.27}$$

Moreover, we can see from (3.5) that $[\mathrm{vec}\,(ZV), \mathrm{vec}\,(U^\top Z)] = -L_\Phi^{-1} J_R(x)^\top \nabla\phi(R(x))$. Hence, (3.5) can be written as $H_0(x)d = -L_\Phi^{-1} J_R(x)^\top \nabla\phi(R(x))$, which implies $d = -L_\Phi^{-1} H_0(x)^\dagger \nabla\Phi(x)$. The full-step GN scheme becomes

$$x_+ = x + d = x - L_\Phi^{-1} H_0(x)^\dagger \nabla\Phi(x). \tag{A.28}$$

We consider the residual term $r = x - x_\star$, where $x_\star := \left[\mathrm{vec}\left(U_\star\right), \mathrm{vec}\left(V_\star^\top\right)\right] \in \mathscr{X}_\star$ is a given stationary point of (1.1). From (A.28) we can write

$$\begin{aligned} r_+ &= x_+ - x_\star = r - L_\Phi^{-1} H_0(x)^\dagger \nabla\Phi(x) \\ &= r - L_\Phi^{-1} H_0(x)^\dagger \left[\nabla\Phi(x) - \nabla\Phi(x_\star)\right] \\ &= \left[\mathbb{I} - L_\Phi^{-1} H_0(x)^\dagger \nabla^2\Phi(x_\star)\right] r \\ &\quad - L_\Phi^{-1} H_0(x)^\dagger \left[\int_0^1 \left(\nabla^2\Phi(x_\star + \tau(x - x_\star)) - \nabla^2\Phi(x_\star)\right)(x - x_\star) d\tau\right]. \end{aligned}$$

Using condition (3.17) and the Lipschitz continuity of the Hessian $\nabla^2\Phi$, the last expression can be upper bounded as follows:

$$\begin{aligned} \|r_+\| &\le \|\left(\mathbb{I} - L_\Phi^{-1} H_0(x)^\dagger \nabla^2\Phi(x_\star)\right) r\| \\ &\quad + L_\Phi^{-1} \|H_0(x)^\dagger\| \int_0^1 \|\nabla^2\Phi(x_\star + \tau(x - x_\star)) - \nabla^2\Phi(x_\star)\| \|x - x_\star\| d\tau \\ &\le \kappa(x_\star)\|r\| + \tfrac{1}{2} L^{-1} L_{\Phi''} \|H_0(x)^\dagger\| \|r\|^2 \\ &\le \left(\bar{\kappa} + 0.5 L_\Phi^{-1} L_{\Phi''} K_h \|r\|\right) \|r\|. \end{aligned} \tag{A.29}$$

Since $r = x - x_\star = \mathrm{vec}\left(X - X_\star\right)$, we can write (A.29) as

$$\|X_+ - X_\star\|_F \le \left(\bar{\kappa} + 0.5 L_\Phi^{-1} L_{\Phi''} K_h \|X - X_\star\|_F\right) \|X - X_\star\|_F,$$

which is exactly (3.18) with $K_1 := L_\Phi^{-1} L_{\Phi''} K_h > 0$.

Next, we prove quadratic convergence of the full-step GN scheme. Under Assumption 2.1, it follows from [46] that there exists a neighborhood $\mathscr{N}(x_\star)$ of $x_\star$ such that $H_0(\cdot)^\dagger$ is Lipschitz continuous in $\mathscr{N}(x_\star$ with the Lipschitz constant $L_H > 0$. Here, we use the same $\mathscr{N}(x_\star$ as in Assumption 3.1. Otherwise, we can shrink it if necessary. We consider the condition $H(X_\star)^\dagger \nabla^2\Phi(X_\star) = L_\Phi \mathbb{I}$. Reforming this condition into vector form, we have $H_0(x_\star)^\dagger \nabla^2\Phi(x_\star) = L_\Phi \mathbb{I}$, which is equivalent to $\mathbb{I} - L_\Phi^{-1} H(X_\star)^\dagger \nabla^2\Phi(X_\star) = 0$. Using the last condition, and the Lipschitz continuity of $H_0^\dagger(\cdot)$, we can show that

$$\begin{aligned} S(x_\star) &:= \|\left[\mathbb{I} - L_\Phi^{-1} H_0(x)^\dagger \nabla^2\Phi(x_\star)\right](x - x_\star)\| \\ &\le \|\left[\mathbb{I} - L_\Phi^{-1} H_0(x_\star)^\dagger \nabla^2\Phi(x_\star)\right](x - x_\star)\| + L_\Phi^{-1} \|\left(H_0(x)^\dagger - H_0(x_\star)^\dagger\right)(x - x_\star)\| \\ &\le L_\Phi^{-1} \|H_0(x)^\dagger - H_0(x_\star)^\dagger\| \|x - x_\star\| \\ &\le L_\Phi^{-1} L_H \|x - x_\star\|^2, \ \forall x \in \mathscr{N}(x_\star). \end{aligned}$$

Substituting this $S(x_\star)$ estimate into (A.29) we get $\|r_+\| \le L_\Phi^{-1}(L_H + 0.5 L_{\Phi''} K_h)\|r\|^2$, which is reformed into the matrix form as

$$\|X_+ - X_\star\|_F \le 0.5 K_2 \|X - X_\star\|_F^2, \ \forall X \in \mathscr{N}(X_\star), \text{ where } K_2 := L_\Phi^{-1}\left(2L_H + L_{\Phi''} K_h\right).$$

In order to guarantee the monotonicity of $\{\|X - X_\star\|_F\}$, we require $\|X_+ - X_\star\|_F \le 0.5 K_1 \|X - X_\star\|_F^2 < \|X - X_\star\|_F$, which implies $\|X - X_\star\|_F < 2K_2^{-1}$. Hence, if we choose $X_0 \in \mathscr{N}(X_\star)$ such that $\|X_0 - X_\star\|_F < 2K_2^{-1}$, then $\|X_k - X_\star\|_F < 2K_2^{-1}$ for all $k \ge 0$ and $\{\|X_k - X_\star\|_F\}$ is monotone. Moreover, $\|X_{k+1} - X_\star\|_F \le 0.5 K_2 \|X_k - X_\star\|_F^2$ shows that this sequence converges quadratically to zero. Hence, $\{X_k\}$ converges to $X_\star$ at a quadratic rate. Here, we can easily check that $K_2 > K_1$.

Finally, if $\bar{\kappa} \in (0,1)$, then for all ≥ 0, the estimate (3.18) implies that

$$\|X_{k+1} - X_\star\|_F \le \left(\bar{\kappa} + 0.5 K_1 \|X_k - X_\star\|_F\right) \|X_k - X_\star\|_F.$$

In order to guarantee $\|X_{k+1}-X_{\star}\|_F < \|X_k - X_{\star}\|_F$, we require $\bar{\kappa}+0.5K_1\|X_k-X_{\star}\|_F < 1$, which leads to $\|X_k - X_{\star}\|_F < 2K_1^{-1}(1-\bar{\kappa})$. Hence, if we take $\bar{r}_0 < 2K_1^{-1}(1-\bar{\kappa})$, and choose $X_0 \in \mathcal{N}(X_{\star})$ such that $\|X_0 - X_{\star}\|_F \leq \bar{r}_0$, then $\|X_k - X_{\star}\|_F \leq \bar{r}_0$ for all $k \geq 0$. In addition, we have $\|X_{k+1}-X_{\star}\|_F \leq (\bar{\kappa}+0.5K_1\|X_k-X_{\star}\|_F)\|X_k-X_{\star}\|_F \leq (\bar{\kappa}+0.5K_1\bar{r}_0)\|X_k-X_{\star}\|_F$, which shows that $\{\|X_k-X_{\star}\|_F\}$ converges to zero at a linear rate with the contraction factor $\omega := \bar{\kappa}+0.5K_1\bar{r}_0 < 1$. □

References

[1] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, B. Recht, Low-rank solutions of linear matrix equations via procrustes flow, 33rd International Conference on Machine Learning, ICML 2016, pp. 1520-1529, New York, 2016,

[2] B. K. Natarajan, Sparse approximate solutions to linear systems, SIAM J. Comput. 24 (1995), 227-234.

[3] E. Candès, B. Recht, Exact matrix completion via convex optimization, Communications of the ACM, 55 (2012), 111-119.

[4] E. J. Candes, Y. Eldar, T. Strohmer, V. Voroninski, Phase retrieval via matrix completion, SIAM J. Imaging Sci. 6 (2011), 199-225.

[5] J. E. Esser, Primal-dual algorithm for convex models and applications to image restoration, registration and nonlocal inpainting, PhD Thesis, University of California, Los Angeles, Los Angeles, USA, 2010.

[6] D. Goldfarb, S. Ma, Convergence of fixed-point continuation algorithms for matrix rank minimization, Found. Comput. Math. 11 (2011), 183-210.

[7] A. Kyrillidis, V. Cevher, Matrix recipes for hard thresholding methods, J. Math. Imaging Vis. 48 (2014), 235-265.

[8] X. Liu, Z. Wen, Y. Zhang, An efficient Gauss-Newton algorithm for symmetric low-rank product matrix approximations, SIAM J. Optim. 25 (2015), 1571-1608.

[9] Z. Wen, W. Yin, Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Math. Program. Comput. 4 (2012), 333-361.

[10] E. Candes, J. Romberg, T. Tao, Stable signal recovery from incomplete and inaccurate measurements, Commun. Pure Appl. Math. 8 (2006), 1207-1223.

[11] S. Bhojanapalli, A. Kyrillidis, S. Sanghavi, Dropping convexity for faster semi-definite optimization, Arxiv preprint:1509.03917, 2015.

[12] E.J. Candés, X. Li, Y. Ma, J. Wright, Robust principal component analysis? Journal of the ACM, 58 (2011), 1-37.

[13] M. Fazel, Matrix rank minimization with applications, PhD thesis, Stanford University, 2002.

[14] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen, 36 (2013), 53-78.

[15] J. Huang, T. Zhang, D. Metaxas, Learning with structured sparsity, J. Mach. Learn. Res. 12 (2011), 3371-3412.

[16] A. Kyrillidis, L. Baldassarre, M. El-Halabi, Q. Tran-Dinh, V. Cevher, Structured sparsity: Discrete and convex approaches, In Compressed Sensing and its Applications, pp. 341-387, Springer, 2015.

[17] B. Recht, M. Fazel, P.A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, SIAM Rev. 52 (2010), 471-501.

[18] M. Signoretto, Q. Tran-Dinh, L. De-Lathauwer, J.A.K. Suykens, Learning with Tensors: a framework based on convex optimization and spectral regularization, Machine Learning, 94 (2014), 303-351.

[19] Y. Yu, Fast gradient algorithms for structured sparsity, PhD thesis, University of Alberta, 2014.

[20] C.R Johnson, Matrix completion problems: a survey, In Matrix theory and applications, vol. 40, pp. 171-198, Providence, RI, 1990.

[21] S. Burer, R. DC. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization, Math. Program. 95 (2003), 329-357.

[22] Z. Lin, M. Chen, L. Wu, Y. Ma, The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices, UIUC Technical Report UILU-ENG-09-2215, 2009.

[23] Y. Shen, Z. Wen, Y. Zhang, Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization, Optim. Method Softw. 29 (2014), 239-263.

[24] H.F. Yu, C.J. Hsieh, S. Si, and I.S. Dhillon, Parallel matrix factorization for recommender systems, Knowledge Info. Sys. 41 (2014), 793-819.

[25] L. Bottou, Large-scale machine learning with stochastic gradient descent, Proceedings of COMPSTAT'2010, pp. 177-186. Springer, 2010.

[26] R.H. Keshavan, S. Oh, A gradient descent algorithm on the Grassman manifold for matrix completion, Arxiv preprint:0910.5260, 2009.

[27] B. Vandereycken, Low-rank matrix completion by Riemannian optimization, SIAM J. Optim. 23 (2013), 1214-1236.

[28] A. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.

[29] P. Deuflhard, Newton Methods for Nonlinear Problems – Affine Invariance and Adaptative Algorithms, vol.35, Springer Series in Computational Mathematics, 2nd edition, Springer, 2006.

[30] E.F. Gonzalez, Y. Zhang, Accelerating the Lee-Seung algorithm for non-negative matrix factorization, Dept. Comput. & Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR-05-02, 2005.

[31] J. Nocedal, S.J. Wright, Numerical Optimization, Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[32] Y. Nesterov, Introductory lectures on convex optimization: A basic course, volume 87 of Applied Optimization, Kluwer Academic Publishers, 2004.

[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2011), 1-122.

[34] D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Athena Scientific, 1996.

[35] M. R. Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. 4 (1969), 303-320.

[36] R. A. Polyak, On the local quadratic convergence of the primal–dual augmented lagrangian method, Optim. Methods Softw. 24 (2009), 369-379.

[37] G. Li, T.-K. Pong, Global convergence of splitting methods for nonconvex composite optimization, SIAM J. Optim. 25 (2015), 2434-2460.

[38] Y. Wang, W. Yin, J. Zeng, Global convergence of ADMM in nonconvex nonsmooth optimization, Arxiv preprint:1511.06324, 2015.

[39] G.H. Golub, C.F. van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[40] D. Gross, Y.-K. Liu, S. Flammia, S. Becker, J. Eisert, Quantum state tomography via compressed sensing, Phys. Rev. Lett. 105 (2010), 150401.

[41] M. Jaggi, Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization, 30th International Conference on Machine Learning, ICML 2013, pp. 427-435, Atlanta, 2013.

[42] J.-F. Cai, E. J. Candes, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (1956), 1956-1982.

[43] K.-C. Toh, S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, Pacific J. Optim. 6 (2010), 615-640.

[44] D.S. Bernstein, Matrix Mathematics, Princeton University Press, 2005.

[45] S.L. Campbell, C.D. Meyer, Generalized inverses of linear transformations, vo. 56, SIAM, 2009.

[46] G. H. Golub, V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SIAM J. Numer. Anal. 10 (1973), 413-432.