AN EFFICIENT GAUSS-NEWTON ALGORITHM FOR SYMMETRIC LOW-RANK PRODUCT MATRIX APPROXIMATIONS

XIN LIU[†], ZAIWEN WEN[‡], AND YIN ZHANG[§]

Abstract. We derive and study a Gauss-Newton method for computing a symmetric low-rank product XX^{T}, where $X \in \mathbb{R}^{n\times k}$ for $k < n$, that is the closest to a given symmetric matrix $A \in \mathbb{R}^{n\times n}$ in Frobenius norm. When $A = B^{\mathrm{T}}B$ (or BB^{T}), this problem essentially reduces to finding a truncated singular value decomposition of B. Our Gauss-Newton method, which has a particularly simple form, shares the same order of iteration-complexity as a gradient method when $k \ll n$, but can be significantly faster on a wide range of problems. In this paper, we prove global convergence and a Q-linear convergence rate for this algorithm, and perform numerical experiments on various test problems, including those from recently active areas of matrix completion and robust principal component analysis. Numerical results show that the proposed algorithm is capable of providing considerable speed advantages over Krylov subspace methods on suitable application problems where high-accuracy solutions are not required. Moreover, the algorithm possesses a higher degree of concurrency than Krylov subspace methods, thus offering better scalability on modern multi/many-core computers.

Key words. Eigenvalue decomposition, singular value decomposition, low-rank product matrix approximation, Gauss-Newton methods

AMS subject classification. 15A18, 65F15, 65K05, 90C06

1. Introduction.

Low-rank matrix approximation techniques, such as dominant eigenvalue or singular value decompositions, appear in a wide variety of scientific and engineering applications. For example, principal component analyses (PCA) in statistics use them to compute a few directions of maximal variance. Many data dimensionality reduction methods in machine learning use them to perform low-dimensional embedding of high-dimensional data. More recently, identifying dominant eigenvalue or singular value decompositions of a sequence of closely related matrices has become an indispensable algorithmic component for many first-order optimization methods for various convex optimization problems, such as semidefinite programming, low-rank matrix completion, robust principal component analysis, sparse principal component analysis, sparse inverse covariance matrix estimation and nearest correlation matrix estimation (see [9] for a more detailed summary). More often than not, the computational cost of doing low-rank approximations forms a major bottleneck in the overall efficiency of solution processes.

For large-scale eigenvalue and singular value calculations, most state-of-the-art solvers are built on Krylov subspace methodologies, including Arnoldi methods for general matrices (e.g., [16, 15]) and Lanczos methods for symmetric (or Hermitian) matrices (e.g., [25, 14]). Jacobi-Davidson methods (e.g., [3, 26]), although built on a different framework, also rely on Krylov subspace methodologies in linear system solving at every iteration. By definition, the Krylov subspace of order p is $\mathcal{K}_p(A, v) = \mathrm{span}\{v, Av, A^2v, \ldots, A^{p-1}v\}$, for given matrix $A \in \mathbb{R}^{n\times n}$ and vector $v \in \mathbb{R}^n$. Both Arnoldi and Lanczos algorithms generate orthonormal bases for Krylov subspaces through a type of Gram-Schmidt process.

It is widely accepted that Krylov-subspace type methods are generally most efficient in terms of the number of matrix-vector multiplications. Indeed, they are fast and reliable for computing a few eigenpairs. A well-known limitation of these methods is the difficulty to warm-start them in an iterative setting where better and better approximate eigenspaces are available. So far, no procedure is known that can efficiently pack subspace information into a single starting vector v to produce $\mathcal{K}_p(A, v)$ that matches a given subspace.

[†]State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, CHINA (liuxin@lsec.cc.ac.cn). Research supported in part by NSFC grants 11101409, 11331012 and 91330115, and the National Center for Mathematics and Interdisciplinary Sciences, CAS.

[‡]Beijing International Center for Mathematical Research, Peking University, Beijing, CHINA (wenzw@math.pku.edu.cn). Research supported in part by NSFC grants 11322109 and 91330202 and by the National Basic Research Project under the grant 2015CB856000.

[§]Department of Computational and Applied Mathematics, Rice University, Houston, UNITED STATES (yzhang@rice.edu). Research supported in part by ONR Grant N00014-08-1-1101 and NSF Grant DMS-1115950.

Another limitation has to do with concurrency in algorithms. Krylov-subspace type methods incur a low degree of concurrency in the sequential process of generating orthonormal bases for $\mathcal{K}_p(A, v)$, especially when the dimension p is relatively large. To increase concurrency, there exist multiple-vector versions of these algorithms in which each single vector in matrix-vector multiplications is replaced by a small number of multiple vectors. Nevertheless, the amount of parallelism is still fundamentally limited by two computational bottlenecks: (i) the construction and maintenance of orthonormal bases for Krylov subspaces, and (ii) repeatedly solving p-dimensional dense eigenvalue problems to compute approximate eigenpairs.

There exists more concurrency in another class of algorithms in which every time p matrix-vector multiplications, say Av_i for $i = 1, 2, \cdots, p$, are carried out simultaneously as a block, i.e., $AV = [Av_1\ Av_2\ \cdots\ Av_p]$, where the number of vectors p is either equal to or slightly larger than k, and k is the dimension of the eigenspace to be computed. Such algorithms are sometimes called *block algorithms*. Clearly, block algorithms can easily take advantages of warm-starts in an iterative setting, requiring few iterations when starting from a good estimate of the desired eigenspace.

Classic block algorithms are simultaneous subspace iteration (SSI) methods that are simple extensions of power method for computing a single eigenpair (see [22, 23, 27, 28] for example). In general, these methods are useful only for special problems due to slow convergence. More recent block algorithms are mostly derived from optimization models. The commonly used optimization model is to maximize the Rayleigh-Ritz function subject to orthogonality constraints:

$$\max_{X\in\mathbb{R}^{n\times k}} \ \mathrm{tr}(X^{\mathrm{T}}AX), \ \mathbf{s.t.}\ X^{\mathrm{T}}X = I. \tag{1}$$

Block algorithms based on solving (1) include the locally optimal block preconditioned conjugate gradient method [12] and the limited memory block Krylov subspace optimization method [19]. The parallel scalability of these algorithms, although improved from that of Krylov subspace methods, is still limited by the extensive use of basis orthogonalizations and dense eigenvalue decompositions of $p \times p$ matrices where usually $p = 3k$. Recently, a trace-penalty minimization model

$$\min_{X\in\mathbb{R}^{n\times k}} \ \frac{1}{2}\mathrm{tr}(X^{\mathrm{T}}AX) + \frac{\mu}{4}\|X^{\mathrm{T}}X - I\|_{\mathrm{F}}^2 \tag{2}$$

is proposed in [31] for computing the eigenspace associated with the k smallest eigenvalues of A using a suitable parameter μ. It enables unconstrained optimization techniques to compute approximate eigenspaces and, consequently, reduces the use of basis orthogonalizations and dense eigenvalue decompositions. This algorithm has a higher parallel scalability on modern multi/many-core systems. However, relying on a gradient-based algorithm that often converges slowly, this algorithm is generally only suitable for computing solutions of low to moderate accuracies.

1.1. The SLRP Model. In this paper, we propose and study a new block algorithm derived from solving the nonlinear least squares model

$$\min_{X\in\mathbb{R}^{n\times k}} \ f(X) := \frac{1}{2}\|R(X)\|_{\mathrm{F}}^2, \tag{3}$$

where $\|\cdot\|_{\mathrm{F}}^2$ is the Frobenius norm squared, and

$$R(X) = XX^{\mathrm{T}} - A \tag{4}$$

is the residual of approximating $A = A^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ by *a symmetric low-rank product* (SLRP) XX^{T} for $X \in \mathbb{R}^{n\times k}$. It is well known that when A is positive semidefinite, a global minimizer of (3) spans a k-dimensional eigenspace

associated with k largest eigenvalues of A. More precisely, let $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ be the eigenvalues of $A \in \mathbb{R}^{n\times n}$ sorted in a descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, and $q_1, \ldots, q_n \in \mathbb{R}^n$ be their corresponding unit eigenvectors. Then a global minimizer of (3), $\hat{X} \in \mathbb{R}^{n\times k}$, takes the form

$$\hat{X} = Q_k \Lambda_k^{1/2} V^{\mathrm{T}}, \tag{5}$$

where $Q_k = [q_1,\ q_2,\ \ldots, q_k]$, $\Lambda_k = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$ and $V \in \mathbb{R}^{k\times k}$ is an arbitrary orthogonal matrix. When $A = B^{\mathrm{T}}B$ or BB^{T}, model (3) reduces essentially to the familiar problem of finding a truncated singular value decomposition (SVD) of B. Replacing A by a suitable polynomial function of A, one can also in principle apply the model to finding the smallest or interior eigenvalues as well.

1.2. Contributions. We propose a Gauss-Newton (GN) method (see [10, 17, 20] for three early references) for solving the nonlinear least squares model (3). At any full-rank iterate $X \in \mathbb{R}^{n\times k}$, the derived GN direction takes the simple form

$$S = \left(I - \frac{1}{2}X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\right)\left(AX(X^{\mathrm{T}}X)^{-1} - X\right),$$

that requires solving small $k \times k$ linear systems. The next iterate is $X^+ = X + \alpha S$, where $\alpha \in (0, 1]$ is a step size. With the help of properly chosen step sizes and a correction procedure to prevent possible rank deficiency, we show that the GN method converges globally to a stationary point. In addition, after a finite number of iterations, the fixed step size $\alpha = 1$ can always be taken with convergence guaranteed — a property not true in general for nonlinear least squares problems (see p.113 of [8]). Finally, we prove, under mild conditions, that our GN method possesses a Q-linear asymptotic rate of convergence.

We have tested a practical version of our GN method where a fixed $\alpha = 1$ is always taken, resulting in a particular simple and parameter-free algorithm. This algorithm has empirically demonstrated a highly reliable convergence behavior. Extensive numerical results show that this algorithm can offer significant speed advantages, especially on so-called small-residual problems, over not only gradient methods, but also current state-of-the-art methods in computing moderately accurate eigenvalue or singular-value decompositions. Computational tasks of this type have become increasingly important, as is commented by Halko, Martinsson and Tropp in [9]: *"In the information sciences, it is common that data are missing or inaccurate. Classical algorithms are designed to produce highly accurate matrix decompositions, but it seems profligate to spend extra computational resources when the imprecision of the data inherently limits the resolution of the output."* The warm-starting advantage of the GN method is also clearly exhibited in applications such as low-rank matrix completion and robust principal component analysis, where the dominant SVDs are computed on a converging sequence of matrices.

As a block algorithm where all main matrix operations are applied to n by k dense blocks (as opposed to single or a few vectors), the proposed GN method has a potential to be successfully mapped onto high performance parallel computers. In combination with other techniques, the GN method also has a potential to become a component of a general-purpose eigensolver. These subjects require more efforts beyond the scope of the present paper and will be studied in future research.

1.3. Organization and notation. The rest of this paper is organized as follows. We examine the properties the SLRP model (3) in Section 2, derive a Gauss-Newton algorithm for this model in Section 3, and analyze the algorithm's convergence properties in Section 4. Numerical results are presented in Section 5. Finally, we conclude the paper in Section 6.

We adopt the following notation. The Kronecker product of two matrices M_1 and M_2 is denoted by $M_1 \otimes M_2$.

The inner product between two matrices, M_1 and M_2, of the same size is defined as $\langle M_1, M_2\rangle = \mathrm{tr}(M_1^{\mathrm{T}} M_2)$. The trace of a matrix M is denoted by $\mathrm{tr}(M)$. The symbols $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ refer to the smallest and largest singular values of a matrix M, respectively. Similarly, $\lambda_{\min}(M)$, $\lambda_{\max}(M)$ and $\lambda_i(M)$ refer to the smallest, largest and the i-th largest eigenvalues of a real symmetric matrix $M \in \mathbb{R}^{n\times n}$, respectively. The notation $M \succ 0$ indicates that M is symmetric positive definite.

2. Properties of the SLRP Model. The gradient and the Hessian of $f(X)$ have well-known generic forms as, respectively,

$$\begin{aligned}\nabla f(X) &= J(X)^{\mathrm{T}}(R(X)),\\ \nabla^2 f(X) &= J(X)^{\mathrm{T}} J(X) + \sum_{i,j} R_{ij}(X)\nabla^2 R_{ij}(X),\end{aligned}$$

where $J(X) : \mathbb{R}^{n\times k} \to \mathbb{R}^{n\times n}$ is the Jacobian operator of $R(X)$ at X, $J(X)^{\mathrm{T}}$ is the adjoint operator of the Jacobian which is from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n\times k}$, and the Hessian $\nabla^2 f(X)$ is a linear operator from $\mathbb{R}^{n\times k}$ to $\mathbb{R}^{n\times k}$.

Since $R(X+S) = R(X) + SX^{\mathrm{T}} + XS^{\mathrm{T}} + SS^{\mathrm{T}}$ for all $S \in \mathbb{R}^{n\times k}$, $J(X)$ is clearly defined by

$$J(X)(S) = SX^{\mathrm{T}} + XS^{\mathrm{T}}. \tag{6}$$

We observe that $J(X)(S) = 0$ for $S = XC$ where $C \in \mathbb{R}^{k\times k}$ is any skew-symmetric matrix (i.e., $C^{\mathrm{T}} = -C$). Hence, the Jacobian $J(X)$ has a nontrivial null space at any X and is rank deficient. By straightforward calculations, we can derive the following formulas:

$$J(X)^{\mathrm{T}}(R) = (R + R^{\mathrm{T}})X, \tag{7}$$

$$J(X)^{\mathrm{T}} J(X)(S) = 2(SX^{\mathrm{T}}X + XS^{\mathrm{T}}X), \tag{8}$$

$$\nabla f(X) = 2(X(X^{\mathrm{T}}X) - AX), \tag{9}$$

$$\nabla^2 f(X)(S) = 2(SX^{\mathrm{T}}X + XS^{\mathrm{T}}X + R(X)S). \tag{10}$$

It follows from (9) that the first-order necessary condition of optimality for problem (3) is

$$AX = X(X^{\mathrm{T}}X). \tag{11}$$

Hence, every nonzero stationary point must span a nontrivial invariant subspace of A. This observation leads to the specific form of any stationary point of (3).

PROPOSITION 2.1. *Let $(U, D) \in \mathbb{R}^{n\times k} \times \mathbb{R}^{k\times k}$ denote k eigenpairs of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ so that $AU = UD$, $U^{\mathrm{T}}U = I$ and D is diagonal. A matrix $X \in \mathbb{R}^{n\times k}$ is a stationary point of (3) if and only if*

$$X = U(PD)^{1/2}V^{\mathrm{T}}, \tag{12}$$

where $V \in \mathbb{R}^{k\times k}$ is an arbitrary orthogonal matrix, and $P \in \mathbb{R}^{k\times k}$ is a diagonal projection matrix with diagonal entries

$$P_{ii} = \begin{cases} 0, & \text{if } D_{ii} < 0,\\ 0 \text{ or } 1, & \text{otherwise.}\end{cases}$$

In particular, X is a rank-k stationary point if and only if $P = I$ and $D_{ii} > 0$ for $i = 1, 2, \cdots, k$.

Proof. Obviously, any X defined by (12) satisfies (11). In the opposite direction, we only provide a proof for the case where X is of full-rank. The rank-deficient cases can be proved along the similar line, though more notationally involved and tedious.

Suppose that X is a full rank stationary point. It spans an invariant subspace of A according to the first-order optimality condition (11). Since every k-dimensional invariant subspace of A can be spanned by a set of k unit eigenvectors, we can write $X = UW$, where U consists of k unit eigenvectors of A, and $W \in \mathbb{R}^{k\times k}$ is nonsingular. Let $D \in \mathbb{R}^{k\times k}$ be a diagonal matrix with k eigenvalues of A on the diagonal corresponding to the eigenvectors in U so that $AU = UD$. Upon substituting $X = UW$ into (11), pre-multiplying both sides of it by U^{T} and noting $U^{\mathrm{T}}U = I$, we derive

$$DW = U^{\mathrm{T}}AUW = W(W^{\mathrm{T}}W) \Leftrightarrow D = WW^{\mathrm{T}} \Leftrightarrow W = D^{1/2}V^{\mathrm{T}},$$

for some orthogonal $V \in \mathbb{R}^{k\times k}$. □

We now state an equivalence between solving the SLRP model (3) and computing a dominant eigenspace of A associated with nonnegative eigenvalues. The proof is straightforward and therefore omitted.

PROPOSITION 2.2. *Assume that $A \in \mathbb{R}^{n\times n}$ is symmetric and has m positive eigenvalues. Then $\hat{X} \in \mathbb{R}^{n\times k}$ is a minimizer of (3) if and only if* (5) *holds, i.e.,*

$$\hat{X} = Q_k \left(\Lambda_k\right)_+^{1/2} V^{\mathrm{T}},$$

where $(t)_+ = \max(0, t)$, $V \in \mathbb{R}^{k\times k}$ is orthogonal and arbitrary. Consequently, $\mathrm{rank}(\hat{X}) = \min(k, m)$.

It follows from Proposition 2.2 that $X = 0$ if $m = 0$, and X has full rank if $m \geq k$. The next proposition states an equivalence between the trace-penalty model (2) and the SLRP model (3) with A replaced by $I - A/\mu$. Whenever $\mu > \max(0, \lambda_{n-k+1})$, the matrix $I - A/\mu$ has at least k positive eigenvalues corresponding to k smallest eigenvalues of A. Hence, the SLRP model can be used to find a correct optimal eigenspace.

PROPOSITION 2.3. *Let $\mu > 0$. The trace-penalty model* (2) *is equivalent to*

$$\min_{X\in\mathbb{R}^{n\times k}} \frac{1}{2}\|XX^{\mathrm{T}} - (I - A/\mu)\|_{\mathrm{F}}^2 \tag{13}$$

in the sense that any stationary point of one problem, after scaling and shifting, has a one-to-one correspondence with a stationary point of the other.

The above equivalence simply follows from the identity

$$\frac{1}{2}\mathrm{tr}(X^{\mathrm{T}}AX) + \frac{\mu}{4}\|X^{\mathrm{T}}X - I\|_{\mathrm{F}}^2 \equiv \frac{\mu}{4}\|XX^{\mathrm{T}} - (I - A/\mu)\|_{\mathrm{F}}^2 + \mathrm{constant}.$$

Theorem 2.2 in [31] shows that the trace-penalty minimization model (2) can have far fewer undesirable, full-rank stationary points than the trace minimization model (1). Due to the equivalence in Proposition 2.3, one could reasonably argue that from an optimization point of view the SLRP model is theoretically more desirable than trace minimization.

PROPOSITION 2.4. *Suppose $A \succeq 0$ and $\lambda_k > 0$. The SLRP model* (3) *has no local maxima, nor local minima other than the global minimum attained by $\hat{X}$ defined in* (5)*. Namely, all stationary points other than the global minimizers are saddle points.*

When $k = 1$ and $\lambda_1 > \lambda_2$, according to Proposition 2.2 there exists exactly two isolated global minimizers for (2) at which the Hessian of $f(X)$ is nonsingular. If $\lambda_1 = \lambda_2$, however, the Hessian of $f(X)$ becomes singular throughout

the solution set since the multiplicity is greater than one. In the case of $k > 1$, it follows from Proposition 2.2 that there is no isolated global minimizer. Hence, the Hessian at any solution is singular. Then, we cannot in general expect a linear rate of convergence from a first-order method for solving (3).

3. A Gauss-Newton Method For the SLRP Model. Proposition 2.2 indicates that the optimal solution of the SLRP model (3) is rank deficient if the number of positive eigenvalues of A is less than k. In this case, if a rank k approximation to A is desired, the SLRP model (3) can be applied to a shifted matrix $A + \mu I$, where μ is a constant so that $\lambda_k + \mu > 0$. For simplicity and without loss of generality, we will assume that $A \succ 0$ hereafter in our analysis.

The Gauss-Newton (GN) Method is arguably the most popular methodology for solving nonlinear least squares problems (e.g., see [8] for an introduction). To derive a GN method for the SLRP model (3), we linearize the residual function $R(X)$, defined in (4), at a given X and solve the resulting linear least squares problem,

$$\min_{S \in \mathbb{R}^{n\times k}} \frac{1}{2}\|R(X) + J(X)(S)\|_{\mathrm{F}}^2,$$

which leads to the normal equations $J(X)^{\mathrm{T}}J(X)(S) = -J(X)^{\mathrm{T}}(R(X))$. Using formulas (8) and (9), we reduce these normal equations to

$$SX^{\mathrm{T}}X + XS^{\mathrm{T}}X = AX - X(X^{\mathrm{T}}X). \tag{14}$$

Normal equations in (14) form a large linear system, with nk unknowns in S, that has infinitely many solutions due to the rank-deficiency of $J(X)$. Fortunately, the system has enough special structures to allow a close form solution, presented in the next proposition.

PROPOSITION 3.1. *Let $X \in \mathbb{R}^{n\times k}$ be full rank. $S(X) \in \mathbb{R}^{n\times k}$ is a solution to* (14) *if and only if*

$$S(X) = (I - \mathcal{P}_X)\left(AX(X^{\mathrm{T}}X)^{-1} - X\right) + XC \tag{15}$$

where $\mathcal{P}_X = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ is the orthogonal projector onto the range space of X, and

$$C \in \mathcal{C} := \{C \in \mathbb{R}^{k\times k} \mid C + C^{\mathrm{T}} = Z(X) := (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}AX(X^{\mathrm{T}}X)^{-1} - I\}. \tag{16}$$

In particular, the matrix $C_0 = \frac{1}{2}Z(X)$ has the minimum Frobenius norm in the set $\mathcal{C}$ that gives a GN direction

$$S_0(X) = \left(I - \frac{1}{2}\mathcal{P}_X\right)\left(AX(X^{\mathrm{T}}X)^{-1} - X\right). \tag{17}$$

Proof. Post-multiplying both sides of (14) by $(X^{\mathrm{T}}X)^{-1}$, we obtain an equivalent system

$$S + XS^{\mathrm{T}}X(X^{\mathrm{T}}X)^{-1} = AX(X^{\mathrm{T}}X)^{-1} - X. \tag{18}$$

The solution to (18) can be decomposed into a component from the range space of X plus one from its orthogonal complement, that is, $S = U + XC$, where $U \in \mathbb{R}^{n\times k}$ satisfies $U^{\mathrm{T}}X = 0$ and $C \in \mathbb{R}^{k\times k}$. Substituting $S = U + XC$ into (18) and using the orthogonality of U to X, we derive that

$$U + X(C + C^{\mathrm{T}}) = AX(X^{\mathrm{T}}X)^{-1} - X. \tag{19}$$

Pre-multiplying both sides of the above equation by $(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$, we obtain

$$(C + C^{\mathrm{T}}) = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}AX(X^{\mathrm{T}}X)^{-1} - I = Z(X). \tag{20}$$

By substituting (20) into (19), we obtain

$$U = AX(X^{\mathrm{T}}X)^{-1} - XZ(X) = (I - \mathcal{P}_X)\left(AX(X^{\mathrm{T}}X)^{-1} - X\right),$$

which, together with $S = U + XC$, gives the solution (15). Moreover, (20) implies that the symmetric part of C is $C_0 = \frac{1}{2}Z(X)$ and the skew-symmetric part is arbitrary. Clearly, C_0 has the minimum Frobenius norm in $\mathcal{C}$. Substituting C_0 into (15) yields (17), and completes the proof. □

We note that even though C_0 has the minimum Frobenius norm in $\mathcal{C}$, S_0 does not have the minimum Frobenius norm in $\mathbb{R}^{n\times k}$ since C appears in $S(X)$ in the form of XC (see (15)). It can be verified that S_0 is the minimum norm solution of (14) in a certain weighted Frobenius norm. To obtain the minimum unweighted Frobenius norm solution, one has to solve a rather complicated Sylvester equation at a considerable cost. Given the simplicity of $S_0(X)$, it appears to be the most natural choice for a Gauss-Newton direction in solving the SLRP model (3).

Now we have the following GN method: starting from a full rank initial point $X^0 \in \mathbb{R}^{n\times k}$, at iteration i,

$$X^{i+1} = X^i + \alpha^i S_0(X^i), \tag{21}$$

where $\alpha^i \in (0, 1]$. Although, the step size α^i could be selected using one of the standard line search procedures, our numerical experiments indicate that the algorithm works fine with a fixed step $\alpha^i = 1$ in almost all test problems (as long as A has enough positive eigenvalues). For the theoretical purpose of establishing global convergence, we will use the following simple step size rule,

$$\alpha^i = \min\left\{1, \sigma^3_{\min}(X^i)/\|\nabla f(X^i)\|_{\mathrm{F}}\right\}. \tag{22}$$

The assembly of $S_0(X^i)$, defined in (15), can be carried out in three steps: for $X = X^i$,

$$Y := X(X^{\mathrm{T}}X)^{-1}, \quad S_0 := AY - X, \quad S_0 := S_0 - X(Y^{\mathrm{T}}S_0)/2. \tag{23}$$

The computational cost at each iteration involves one block SpMV in AY, three dense matrix multiplications and solving one $k \times k$ linear systems with n columns of right hand sides. The main memory requirement is three $n \times k$ matrices for X, Y and S_0.

As will be shown in Lemma 4.4, the sequence $\{X^i\}$ generated by (21) using (22) will remain of full rank if X^0 is of full rank. However, this does not ensure the full rankness at the limit as $i \to \infty$. As a theoretical remedy, we devise a correction step that is invoked whenever $\sigma_{\min}(X^i) < \delta$ for a small positive number δ that will be defined later. If necessary, we correct X^i by adding a correction step P^i to it, i.e.,

$$X^i_c = X^i + P^i, \tag{24}$$

where $P^i \in \mathbb{R}^{n\times k}$ is perpendicular to X^i whose definition will be given in the next section. As will be shown in Lemma 4.5, this correction ensures that $\sigma_{\min}(X^i_c) \geq \delta > 0$, while the objective function value at X^i_c is decreased from that at X^i by a constant, which guarantees that this correction step can only be invoked a finite number of times.

A theoretical version of our GN method for solving the SLRP model is outlined in Algorithm 1.

Algorithm 1: A Theoretical GN Algorithm for SLRP (SLRPGN)

Input a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$.
Initialize $X^0 \in \mathbb{R}^{n\times k}$ to a rank-k matrix, set $\delta > 0$ (defined later).
for $i = 0, 1, 2, \cdots$, **do**
    If $\sigma_{\min}(X^i) < \delta$, set $X^i = X^i_c$ according to (24).
    Compute α^i using (22) and $S_0(X^i)$ using (23).
    Set $X^{i+1} = X^i + \alpha^i S_0(X^i)$.

In practice, extensive numerical results show that the correction step is not needed for $A \succ 0$, nor is the step size rule (22). When $\alpha^i \equiv 1$, the algorithm becomes parameter-free. In our numerical experiments, we use a practical implementation of the SLRPGN algorithm with $\alpha^i \equiv 1$ and without the correction step. Our parameter-free GN method is given as Algorithm 2 below, which is still dubbed as SLRPGN . A concrete termination rule for Algorithm 2 will be specified in Section 5. Unless it is in a warm-starting situation, we always start the algorithm from a randomly generated starting matrix.

Algorithm 2: A Practical GN Algorithm for SLRP (SLRPGN)

Input $A \in \mathbb{R}^{n\times n}$ and $k < n$.
Set $i = 0$ and initialize $X^0 \in \mathbb{R}^{n\times k}$ to a rank-k matrix.
while *"the termination criterion" is not met,* **do**
    Compute $Y = X^i \left((X^i)^{\mathrm{T}} X^i\right)^{-1}$ and $Z = AY$.
    Compute $X^{i+1} = Z - X^i(Y^{\mathrm{T}} Z - I)/2$.
    Increment i and continue.
If requested, compute Ritz pairs of A from the last X-iterate.

Ritz pairs, appearing in the last step of Algorithm SLRPGN , are approximate eigenpairs of A. Specifically, Ritz values are eigenvalues of the $k \times k$ matrix $U^T AU$ where $U \in \mathbb{R}^{n\times k}$ is from an orthonormalization of X, and Ritz vectors are eigenvectors of $U^T AU$ pre-multiplied by U. This procedure is the well known Rayleigh-Ritz procedure.

4. Convergence Analysis. In this section, we analyze the convergence properties of two version of GN algorithms. We first present our main results in Theorems 4.1, 4.2 and 4.3 without proofs. Theorem 4.1 gives the global convergence of Algorithm 1.

THEOREM 4.1. *Let $A \succ 0$ and $\{X^i\}$ be generated by Algorithm 1. Then there holds*

$$\lim_{i\to\infty} \nabla f(X^i) = 0, \tag{25}$$

and any limit point of $\{X^i\}$ has full rank. Moreover, there exists an integer $i_0 > 0$ such that for all $i \geq i_0$, $\alpha^i = 1$ and the correction step is not invoked.

The proof of the theorem will be developed in Subsections 4.1 to 4.3. In Subsection 4.1, we show that the full rankness is maintained by Algorithm 1 throughout iterations. Then we specify the correction step (24) in details, and show that it leads to a constant decrease in the objective function, while ensuring that the smallest singular values of the iterates are bounded away from zero. Once such a lower bound on singular values is established, we prove, in Subsection 4.2, that a sufficient reduction in the objective function along the GN direction can be guaranteed, which leads to the proof of Theorem 4.1 in Subsection 4.3.

Theorem 4.2 is about Algorithm 2, which is nothing but Algorithm 1 without the correction step and with a fixing

step size $\alpha^i \equiv 1$. This theorem does impose a restriction on starting points. Let us define the optimal solution set of the SLRP problem (3) as $\mathcal{L}(f^*)$ where f^* is the global minimum of (3) and $\mathcal{L}(\gamma) = \{X \mid f(X) \leq \gamma\}$ is a level set of $f(X)$.

THEOREM 4.2. *Let $A \succ 0$ and $\{X^i\}$ be generated by Algorithm 2 without termination. There exists a proper superset Ω of $\mathcal{L}(f^*)$, i.e., $\Omega \supset \mathcal{L}(f^*)$ strictly, so that if $X^0 \in \Omega$, then $\nabla f(X^i) \to 0$. Moreover, if $\lambda_k > \lambda_{k+1}$, then there exists a proper superset Ω_* of $\mathcal{L}(f^*)$ so that $X^0 \in \Omega_*$ guarantees $f(X^i) \to f^*$.*

We emphasize that although we have not been able to obtain a good estimate on the size of the convergence region Ω, an overwhelming amount of empirical evidence suggests that the region not be a small neighborhood, because convergence, to not only a stationary point but a minimum, appears to always occur when starting from randomly generated initial points. We will further comment on this region after proving our result.

Finally, we will show that under suitable conditions, either Algorithm 1 or 2 has a Q-linear convergence rate.

THEOREM 4.3. *Let $A \succ 0$ and $\{X^i\}$ be generated by either Algorithm 1 or Algorithm 2 without termination. If $f(X^i) \to f^*$ and $\lambda_{k+1} < \lambda_k$, then $\left\|\nabla f(X^i)\right\|_{\mathrm{F}} \to 0$ at a Q-linear rate no greater than $\lambda_{k+1}/\lambda_k < 1$.*

The proofs of Theorem 4.2 and Theorem 4.3 are left to Subsections 4.4 and 4.5, respectively.

4.1. Correction Step. As defined in (17), our GN directions require the full-rank property of iterates X^i. We first show that this full-rank property is guaranteed by Algorithm 1 as long as the initial point has full rank.

LEMMA 4.4. *Let $A \succ 0$ and X^{i+1} be generated by* (21) *with a step size α^i defined by* (22)*. Then*

$$\sigma_{\min}(X^{i+1}) \geq \frac{3}{4}\sigma_{\min}(X^i). \tag{26}$$

In particular, $\sigma_{\min}(X^{i+1}) > 0$ whenever $\sigma_{\min}(X^i) > 0$.

Proof. For notational brevity, we drop the superscripts and let $X = X^i$ and $X_+ = X^{i+1}$.

Define $Y = X(X^{\mathrm{T}}X)^{-1}$. Then we have $X_+^{\mathrm{T}}\left(\lambda_{\max}(YY^{\mathrm{T}})I - YY^{\mathrm{T}}\right)X_+ \succeq 0$, which yields

$$\sigma_{\min}^2(X_+)\sigma_{\max}^2(Y) \geq \sigma_{\min}^2(Y^{\mathrm{T}}X_+). \tag{27}$$

In view of the symmetry of $Y^{\mathrm{T}}X_+$ and formulas (17) and (9), we calculate that

$$\begin{aligned}\lambda_{\min}(Y^{\mathrm{T}}X_+) &= \lambda_{\min}\left(I - \frac{\alpha}{4}(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\nabla f(X)(X^{\mathrm{T}}X)^{-1}\right)\\ &= 1 - \frac{\alpha}{4}\lambda_{\max}\left((X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\nabla f(X)(X^{\mathrm{T}}X)^{-1}\right)\\ &\geq 1 - \frac{\alpha\|\nabla f(X)\|_{\mathrm{F}}}{4\sigma_{\min}^3(X)} \geq \frac{3}{4},\end{aligned} \tag{28}$$

where (22) is used in the last inequality. Noting $\sigma_{\max}(Y) = \sigma_{\min}^{-1}(X)$ and using (27) and (28), we obtain (26). □

Evidently, Lemma 4.4 does not exclude the possibility of $\lim_{i\to\infty}\sigma_{\min}(X^i) = 0$ (even though this has not been observed by us in practice). Proposition 2.1 indicates that any full-rank stationary point $\hat{X}$ satisfies $\sigma_{\min}(\hat{X}) \geq \sqrt{\lambda_n}$. If X^i has a singular value far smaller than $\sqrt{\lambda_n}$, then it cannot be close to any full-rank stationary point. Hence, there exist directions in the null space of $(X^i)^{\mathrm{T}}$ along which we can pull those small singular value back and reduce the function value at the same time. Based on this idea, we construct the correction step P^i introduced in (24).

Let δ be a positive number so that

$$\delta \leq \left(\frac{\lambda_n/\lambda_1}{4+\sqrt{20}}\right)\sqrt{\frac{\lambda_n}{k}}. \tag{29}$$

Again for brevity, we will drop the superscript i and let X denote the current iterate X^i (and similarly for X_c^i and P^i) in this Subsection and Subsection 4.2. The construction of the correction step is as follows.

Let $V\Pi V^{\mathrm{T}}$ be an eigenvalue decomposition of $X^{\mathrm{T}}X$, where the diagonal matrix $\Pi = \mathrm{Diag}(\pi_1, \ldots, \pi_k)$ consists of eigenvalues $0 \le \pi_1 \le \pi_2 \le \ldots \le \pi_k$ on its diagonal, and V are the corresponding eigenvectors. Assume that $\pi_1 < \delta^2$ (that is, $\sigma_{\min}(X) < \delta$). Let p be the largest integer in $[1, k]$ satisfying $\pi_p \le \delta^2$, $\Pi_p = \mathrm{Diag}(\pi_1, \ldots, \pi_p)$ and V_p be the matrix consisting of the first p columns of V. We construct the correction step P as follows: find any $U \in \mathbb{R}^{n\times p}$ such that

$$X^{\mathrm{T}}U = 0, \quad U^{\mathrm{T}}U = I, \quad P = \sqrt{\frac{\lambda_n}{p}}\, UV_p^{\mathrm{T}}. \tag{30}$$

LEMMA 4.5. *Let $A \in \mathbb{R}^{n\times n} \succ 0$, and $X \in \mathbb{R}^{n\times k}$ satisfy $\sigma_{\min}(X) < \delta$ where δ is defined in* (29)*. Let $X_c = X + P$ where P is defined in* (30)*. Then*

$$f(X_c) < f(X) - \frac{1}{4}\lambda_n^2. \tag{31}$$

Moreover, $\sigma_{\min}(X_c) \ge \delta$.

Proof. Let $\gamma = \lambda_n/p$. Noting that $X^{\mathrm{T}}P = 0$ and $\|XV_p\|_{\mathrm{F}}^2 = \mathrm{tr}(\Pi_p) < p\delta^2$, we calculate

$$\begin{aligned}
&2(f(X_c) - f(X)) = \|R(X+P)\|_{\mathrm{F}}^2 - \|R(X)\|_{\mathrm{F}}^2 \\
&= \|R(X) + XP^{\mathrm{T}} + PX^{\mathrm{T}} + PP^{\mathrm{T}}\|_{\mathrm{F}}^2 - \|R(X)\|_{\mathrm{F}}^2 \\
&= 2\|PX^{\mathrm{T}}\|_{\mathrm{F}}^2 + \|PP^{\mathrm{T}}\|_{\mathrm{F}}^2 - 2\mathrm{tr}(P^{\mathrm{T}}AP) - 4\mathrm{tr}(P^{\mathrm{T}}AX) \\
&= 2\gamma\|XV_p\|_{\mathrm{F}}^2 + \gamma^2 p - 2\gamma\,\mathrm{tr}(U^{\mathrm{T}}AU) - 4\sqrt{\gamma}\,\mathrm{tr}(U^{\mathrm{T}}AXV_p) \\
&< 2\gamma p\delta^2 + \gamma^2 p - 2\gamma\, p\lambda_n + 4\sqrt{\gamma}\,\|AU\|_{\mathrm{F}}\|XV_p\|_{\mathrm{F}} \\
&< 2\lambda_n\delta^2 + \gamma^2 p - 2\gamma\, p\lambda_n + 4\sqrt{\gamma}\,\lambda_1 p\delta \\
&= \left(\delta^2 + 2\lambda_1\sqrt{p/\lambda_n}\delta - (1 - 0.5/p)\,\lambda_n\right) 2\lambda_n \\
&\le \left(\delta^2 + 2\lambda_1\sqrt{k/\lambda_n}\delta - 0.5\lambda_n\right) 2\lambda_n \\
&\le (-0.25\lambda_n)2\lambda_n,
\end{aligned}$$

where the last inequality follows from the definition of δ in (29) where the right-hand side is no greater than the positive root of $t^2 + 2\lambda_1\sqrt{k/\lambda_n}t - 0.25\lambda_n = 0$; hence $\delta^2 + 2\lambda_1\sqrt{k/\lambda_n}\delta - 0.5\lambda_n \le -0.25\lambda_n$. This proves (31). Finally, the eigenvalue decomposition of $X^{\mathrm{T}}X$ and the construction of P yield

$$X_c^{\mathrm{T}}X_c = X^{\mathrm{T}}X + P^{\mathrm{T}}P = V\Pi V^{\mathrm{T}} + \frac{\lambda_n}{p}V_pV_p^{\mathrm{T}} \succeq \min\left(\pi_1 + \frac{\lambda_n}{p},\, \beta\right) I \succ \delta^2 I,$$

where $\beta = \pi_{p+1} > \delta^2$ if $p < k$, and $+\infty$ otherwise, which implies $\sigma_{\min}(X_c) > \delta$. This completes the proof. □

4.2. Sufficient Decrease in Residual. In this subsection, we first prove a sufficient decrease property for the objective function along the GN direction whenever the singular values of the iterates are bounded away from zero.

Let $\mathcal{P}_J(X)$ be the orthogonal projector in $\mathbb{R}^{n\times n}$ onto the range space of $J(X)$ which consists of low-rank symmetric matrices defined by $J(X)(S) = XS^{\mathrm{T}} + SX^{\mathrm{T}}$. Namely, $\mathcal{P}_J(X) := J(X)J(X)^{\dagger}$. Recall that $J(X)$ has a nontrivial null space $\{XC : C^{\mathrm{T}} = -C\}$.

PROPOSITION 4.6. *Let $X \in \mathbb{R}^{n\times k}$ be of full rank. The projector $\mathcal{P}_J(X)$ has the form*

$$\mathcal{P}_J(X) = I \otimes \mathcal{P}_X + \mathcal{P}_X \otimes I - \mathcal{P}_X \otimes \mathcal{P}_X. \tag{32}$$

Consequently, $I - \mathcal{P}_J(X) = (I - \mathcal{P}_X) \otimes (I - \mathcal{P}_X)$.

Proof. Note that any GN step $S(X)$ can be written as $S(X) = -J(X)^\dagger(R(X)) + XC$, where C is skew-symmetric, so that $J(X)(S(X)) = -J(X)J(X)^\dagger(R(X))$. In particular, for $S_0 = S_0(X)$ corresponding to $C = 0$,

$$S_0X^{\mathrm{T}} + XS_0^{\mathrm{T}} = -J(X)J(X)^\dagger(R(X)) = -\mathcal{P}_J(X)(R(X)).$$

On the other hand, it holds

$$S_0X^{\mathrm{T}} = \left(I - \frac{1}{2}\mathcal{P}_X\right)\left(A - XX^{\mathrm{T}}\right)X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}} = \left(I - \frac{1}{2}\mathcal{P}_X\right)(-R(X))\,\mathcal{P}_X,$$

which yields

$$\mathcal{P}_J(X)(R(X)) = -(S_0X^{\mathrm{T}} + XS_0^{\mathrm{T}}) = R(X)\mathcal{P}_X + \mathcal{P}_XR(X) - \mathcal{P}_XR(X)\mathcal{P}_X. \tag{33}$$

Therefore, the projector $\mathcal{P}_J(X)$ has the matrix representation (32). This completes the proof. □

The next lemma shows that our GN step $S_0(X)$ with a proper step size will ensure a sufficient reduction of the function value $f(X)$.

LEMMA 4.7. *Let $A \succ 0$, X be a non-stationary point, and α be defined by*

$$\alpha = \min\left\{1, \sigma_{\min}^3(X)/\|\nabla f(X)\|_{\mathrm{F}}\right\}. \tag{34}$$

Then it holds for the GN direction $S_0 = S_0(X)$ defined in (17) *that*

$$f(X + \alpha S_0) \le f(X) - \frac{\alpha}{128}\frac{\sigma_{\min}^2(X)}{\sigma_{\max}^4(X)}\|\nabla f(X)\|_{\mathrm{F}}^2. \tag{35}$$

Proof. Taking the norm square of $R(X + \alpha S_0) = (I - \alpha\mathcal{P}_J(X))\,(R(X)) + \alpha^2 S_0S_0^{\mathrm{T}}$ yields

$$\|R(X + \alpha S_0)\|_{\mathrm{F}}^2 = \|R(X)\|_{\mathrm{F}}^2 - \alpha(2-\alpha)\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 + \psi_1(X) + \psi_2(X) + \psi_3(X), \tag{36}$$

where

$$\begin{aligned}
\psi_1(X) &= -2\alpha^2\,\mathrm{tr}\left(S_0^{\mathrm{T}}(I - \mathcal{P}_X)A(I - \mathcal{P}_X)S_0\right),\\
\psi_2(X) &= 2\alpha^2(1-\alpha)\left\langle \mathcal{P}_J(X)(R(X)), S_0S_0^{\mathrm{T}}\right\rangle,\\
\psi_3(X) &= \alpha^4\|S_0^{\mathrm{T}}S_0\|_{\mathrm{F}}^2.
\end{aligned}$$

We introduce $Y = X(X^{\mathrm{T}}X)^{-1}$ and the quantities

$$G(X) = AY - X = -\frac{1}{2}\nabla f(X)(X^{\mathrm{T}}X)^{-1}, \tag{37}$$

$$\nu(X) = \|G_n(X)\|_{\mathrm{F}}^2 + \frac{1}{4}\|G_r(X)\|_{\mathrm{F}}^2, \tag{38}$$

where

$$G_n(X) = (I - \mathcal{P}_X)G(X), \quad G_r(X) = \mathcal{P}_X G(X).$$

From (37) and (38), clearly $\frac{1}{4}\|G(X)\|_{\mathrm{F}}^2 \le \nu(X) \le \|G(X)\|_{\mathrm{F}}^2$, and therefore,

$$\frac{1}{16}\frac{\|\nabla f(X)\|_{\mathrm{F}}^2}{\sigma_{\max}^4(X)} \le \nu(X) \le \frac{1}{4}\frac{\|\nabla f(X)\|_{\mathrm{F}}^2}{\sigma_{\min}^4(X)}. \tag{39}$$

Under the above notation, we have

$$S_0 = G_n(X) + \frac{1}{2}G_r(X), \quad \left\|S_0^{\mathrm{T}}S_0\right\|_{\mathrm{F}}^2 = \|G_r(X)\|_{\mathrm{F}}^2 + \frac{1}{16}\|G_n(X)\|_{\mathrm{F}}^2,$$

and we can rewrite or estimate

$$\psi_1(X) = -2\alpha^2\|A^{1/2}G_n(X)\|_{\mathrm{F}}^2, \tag{40}$$

$$\psi_2(X) \le 2\alpha^2\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}\left\|S_0S_0^{\mathrm{T}}\right\|_{\mathrm{F}}, \tag{41}$$

$$\psi_3(X) \le \alpha^4\,\nu(X)^2, \tag{42}$$

where in (41) we use the fact that $(1-\alpha) \in [0,1)$. Furthermore, since

$$R(X)\mathcal{P}_X = (X - AY)X^{\mathrm{T}} = -G(X)X^{\mathrm{T}},$$

it follows from (32) that

$$\begin{aligned}\mathcal{P}_J(X)(R(X)) &= -G(X)X^{\mathrm{T}} - XG(X)^{\mathrm{T}} + \mathcal{P}_X G(X)X^{\mathrm{T}}\\ &= -(I - \mathcal{P}_X)G(X)X^{\mathrm{T}} - XG(X)^{\mathrm{T}}\\ &= -G_n(X)X^{\mathrm{T}} - X(G_n(X) + G_r(X))^{\mathrm{T}},\end{aligned}$$

where the right-hand side has been decomposed into two orthogonal component in $\mathbb{R}^n$. Therefore,

$$\begin{aligned}\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 &= 2\|G_n(X)X^{\mathrm{T}}\|_{\mathrm{F}}^2 + \|G_r(X)X^{\mathrm{T}}\|_{\mathrm{F}}^2\\ &\ge 2\left(\|G_n(X)\|_{\mathrm{F}}^2 + \frac{1}{2}\|G_r(X)\|_{\mathrm{F}}^2\right)\sigma_{\min}^2(X),\end{aligned}$$

which implies

$$\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 \ge 2\nu(X)\sigma_{\min}^2(X) \tag{43}$$

and

$$-\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 \le -\frac{3}{4}\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 - \frac{1}{2}\nu(X)\sigma_{\min}^2(X). \tag{44}$$

Substituting the relationships in (40)-(42) and (44) into (36) and noting $2 - \alpha \ge 1$, we obtain

$$\|R(X + \alpha S_0)\|_{\mathrm{F}}^2 \le \|R(X)\|_{\mathrm{F}}^2 - \frac{3}{4}\alpha\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2 - \frac{1}{2}\alpha\nu(X)\sigma_{\min}^2(X)$$

$$\begin{aligned}&-2\alpha^2\|A^{1/2}G_n\|_{\mathrm{F}}^2+2\alpha^2\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}+\alpha^4\nu^2(X)\\ \le\ &\|R(X)\|_{\mathrm{F}}^2-\frac{1}{4}\alpha\nu(X)\sigma_{\min}^2(X)-\xi_1(X)-\xi_2(X),\end{aligned}$$

where we have dropped the term $2\alpha^2\|A^{1/2}G_n\|_{\mathrm{F}}^2$ in the last inequality, and set

$$\begin{aligned}\xi_1(X)&=\frac{1}{4}\alpha\nu(X)\sigma_{\min}^2(X)-\alpha^4\nu^2(X),\\ \xi_2(X)&=\frac{3}{4}\alpha\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2-2\alpha^2\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}.\end{aligned}$$

We will show that both $\xi_1(X)$ and $\xi_2(X)$ are nonnegative and thus can also be dropped, giving

$$\|R(X+\alpha S_0)\|_{\mathrm{F}}^2\le\|R(X)\|_{\mathrm{F}}^2-\frac{1}{4}\alpha\nu(X)\sigma_{\min}^2(X)$$

which, together with the left inequality in (39), implies (35).

We first show $\xi_1(X)\ge 0$. Using (34) and the right inequality in (39), we derive

$$\begin{aligned}\xi_1(X)&=\frac{1}{4}\alpha\nu(X)\left(\sigma_{\min}^2(X)-4\alpha^3\nu(X)\right)\\ &\ge\frac{1}{4}\alpha\nu(X)\left(\sigma_{\min}^2(X)-4\alpha^2\nu(X)\right)\\ &\ge\frac{1}{4}\alpha\nu(X)\left(\sigma_{\min}^2(X)-4\left(\frac{\sigma_{\min}^3(X)}{\|\nabla f(X)\|_{\mathrm{F}}}\right)^2\frac{1}{4}\frac{\|\nabla f(X)\|_{\mathrm{F}}^2}{\sigma_{\min}^4(X)}\right)=0.\end{aligned}$$

A byproduct of the above derivation is $\sigma_{\min}^2(X)\ge 4\alpha^2\nu(X)$, which, together with the inequality in (43) and the fact $\nu(X)\ge\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}^2$, implies $\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}^2\ge 8\alpha^2\nu(X)\ge 8\alpha^2\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}^2$. After taking square root on both sides, we have

$$\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}\ge\sqrt{8}\alpha\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}\ge\frac{8}{3}\alpha\|S_0S_0^{\mathrm{T}}\|_{\mathrm{F}}.\tag{45}$$

Finally, we show $\xi_2(X)\ge 0$ using (45),

$$\xi_2(X)=\frac{3}{4}\alpha\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}\left(\|\mathcal{P}_J(X)(R(X))\|_{\mathrm{F}}-\frac{8}{3}\alpha\|S_0^{\mathrm{T}}S_0\|_{\mathrm{F}}\right)\ge 0.$$

This completes the proof. □

4.3. Convergence of Algorithm 1. We are now ready to prove Theorem 4.1, which is restated below.

THEOREM 4.1. *Let $A\succ 0$ and $\{X^i\}$ be generated by Algorithm 1. Then there holds*

$$\lim_{i\to\infty}\nabla f(X^i)=0,$$

and any limit point of $\{X^i\}$ has full rank. Moreover, there exists an integer $i_0>0$ such that for all $i\ge i_0$, $\alpha^i=1$ and the correction step is not invoked.

Proof. It follows from Lemma 4.7 that

$$f(X^{i+1})\le f(X^i)-\frac{\alpha^i\sigma_{\min}^2(X^i)}{128\sigma_{\max}^4(X^i)}\|\nabla f(X^i)\|_{\mathrm{F}}^2<f(X^i).$$

The monotonicity in the decrease of $f(X^i)$ guarantees the boundedness of $\{X^i\}$; namely, there exists $\eta > 0$ such that $\sigma_{\max}(X^i) \leq \eta$ for all i. Hence,

$$f(X^{i+1}) \leq f(X^i) - \frac{\alpha^i \sigma_{\min}^2(X^i)}{128\eta^4} \|\nabla f(X^i)\|_{\mathrm{F}}^2.$$

By the step-size selection rule (22), we have

$$f(X^{i+1}) \leq f(X^i) - \min\left(\|\nabla f(X^i)\|_{\mathrm{F}}, \sigma_{\min}^3(X^i)\right) \frac{\sigma_{\min}^2(X^i)}{128\eta^4} \|\nabla f(X^i)\|_{\mathrm{F}}.$$

Since $\{f(X^i)\}$ is bounded from below by zero, Lemma 4.5 guarantees that the correction step (24) cannot be invoked infinitely many times. Eventually, $\sigma_{\min}(X^i) \geq \delta > 0$ for all sufficiently large i; thus

$$f(X^{i+1}) \leq f(X^i) - \min\left(\|\nabla f(X^i)\|_{\mathrm{F}}, \delta^3\right) \frac{\delta^2}{128\eta^4} \|\nabla f(X^i)\|_{\mathrm{F}}$$

for all sufficiently large i. The second term in the right-hand side must go to zero as $i \to \infty$, implying

$$\lim_{i\to\infty} \|\nabla f(X^i)\|_{\mathrm{F}} = 0.$$

In addition, the boundedness of $\{\sigma_{\min}(X^i)\}$ away from zero ensures that any limit point has full rank and

$$\alpha^i = \min\left(1, \frac{\sigma_{\min}^3(X^i)}{\|\nabla f(X^i)\|_{\mathrm{F}}}\right) \equiv 1$$

for all sufficiently large i. Finally, it follows from (31) that the correction step can be invoked at most a finite number of times. This completes the proof. □

4.4. Convergence of Algorithm 2. Now we analyze the convergence of Algorithm 2. It is known that a unit-step Gauss-Newton method in general does not converge even locally (see, for example, Section 6 of [8]) unless the residual at the solution is sufficiently small. Contrary to the general case, we show that there exist convergence regions for our unit-step Gauss-Newton Algorithm 2 regardless of residual sizes.

We continue to assume that $A \succ 0$. Recall that $\mathcal{L}(\gamma) = \{X \mid f(X) \leq \gamma\}$ is the level set of $f(X)$ associated with γ, and $\mathcal{L}(f^*)$ is the solution set of (3). We first introduce a set

$$\Omega_\tau := \left\{X \in \mathbb{R}^{n\times k} \mid \|\nabla f(X)\|_{\mathrm{F}} \leq \tau^3,\, \sigma_{\min}(X) \geq \tau\right\}, \tag{46}$$

where $\tau \in [\delta, \sqrt{\lambda_k})$, δ is the threshold for invoking correction used in Algorithm 1, see (29), and λ_k is the k-th largest eigenvalue of A. By continuity, Ω_τ contains at least a neighborhood of the solution set $\mathcal{L}(f^*)$ where $\nabla f(X) = 0$ and $\sigma_{\min}(X) = \sqrt{\lambda_k} > \delta$. Conceptually, we could choose τ within the prescribed interval to make Ω_τ as large as possible, which would potentially lead to a larger convergence region for Algorithm 2.

The following observation is important and will be used below in the proof of Theorem 4.2. That is, if Algorithm 1 is used, then for any $X^i \in \Omega_\tau$ we will have $\alpha^i = 1$ according to the step size rule (22), and the correction step will not be invoked because $\sigma_{\min}(X^i) \geq \delta$.

Next let γ_τ be the largest number so that $\mathcal{L}(\gamma_\tau) \subset \Omega_\tau$, i.e.,

$$\gamma_\tau = \sup_{\gamma \geq f^*} \{\gamma \mid \mathcal{L}(\gamma) \subset \Omega_\tau\}. \tag{47}$$

In other words, $\mathcal{L}(\gamma_\tau)$ is the largest level set of f that is contained in Ω_τ. The next lemma states that $\mathcal{L}(\gamma_\tau)$ is a proper superset of the solution set $\mathcal{L}(f^*)$.

LEMMA 4.7. *Let γ_τ be defined in* (47)*. Then $\gamma_\tau > f^*$.*

Proof. We construct a proof by contradiction. Suppose that for every $\gamma > f^*$, $\mathcal{L}(\gamma) \not\subset \Omega_\tau$. Then for every $\gamma > f^*$ there exists $X \in \mathcal{L}(\gamma)$ that is not in Ω_τ. Let $\{\gamma_j\}$ be a decreasing sequence such that $\gamma_j \to f^*$. There exists a sequence $\{X^j\}$ so that $X^j \in \mathcal{L}(\gamma_j)$ and all $X^j \notin \Omega_\tau$. As a result, $f(X^j) \to f^*$. Then there exists a cluster point, say $\hat{X}$, of $\{X^j\}$ that is a global minimizer of the SLRP model (3), due to $f(\hat{X}) = f^*$. Using Proposition 2.2, we have $\sigma_{\min}(\hat{X}) = \sqrt{\lambda_k}$. Hence, $\hat{X}$ is an interior point of the set Ω_τ, which contradicts the hypothesis that all $X^j \notin \Omega_\tau$. This completes the proof. □

Now we are ready to give a proof for Theorem 4.2 restated below.

THEOREM 4.2. *Let $A \succ 0$ and $\{X^i\}$ be generated by Algorithm 2 without termination. There exists a proper superset Ω of $\mathcal{L}(f^*)$, i.e., $\Omega \supset \mathcal{L}(f^*)$ strictly, so that if $X^0 \in \Omega$, then $\nabla f(X^i) \to 0$. Moreover, if $\lambda_k > \lambda_{k+1}$, then there exists a proper superset Ω_* of $\mathcal{L}(f^*)$ so that $X^0 \in \Omega_*$ guarantees $f(X^i) \to f^*$.*

Proof. Let $\Omega = \mathcal{L}(\gamma_\tau) \subset \Omega_\tau$. Starting from $X^0 \in \Omega$, which is clearly of full rank, we apply Algorithm 1 and obtain $\nabla f(X^i) \to 0$ by Theorem 4.1. We note that all iterates remain in the level set Ω, since $f(X^i)$ is monotonically decreasing. Hence, as has been observed above, $\alpha^i = 1$ is always taken according to the step size rule (22) and the correction step is never invoked due to $\sigma_{\min}(X^i) \geq \delta$. In this case, Algorithm 2 is identical to Algorithm 1.

To show the existence of Ω_*, let us choose τ in a smaller interval:

$$\tau \in \left(\sqrt{\lambda_{k+1}}, \sqrt{\lambda_k}\right),$$

then Ω_τ will contain no other stationary points except the global minima, since at any other stationary point we must have $\sigma_{\min}(X) \leq \sqrt{\lambda_{k+1}} < \tau$. For this τ value, we set $\Omega_* = \mathcal{L}(\gamma_\tau)$, and then the convergence of $\{f(X^i)\}$ must be to the global minimum f^*. This proves the theorem. □

4.5. Rate of Convergence. Finally, we establish a Q-linear convergence rate under suitable conditions for either Algorithm 1 or Algorithm 2, which are asymptotically identical.

THEOREM 4.3. *Let $A \succ 0$ and $\{X^i\}$ be generated by either Algorithm 1 or 2 without termination. If $f(X^i) \to f^*$ and $\lambda_{k+1} < \lambda_k$, then $\left\|\nabla f(X^i)\right\|_{\mathrm{F}} \to 0$ at a Q-linear rate no greater than $\lambda_{k+1}/\lambda_k < 1$.*

Proof. For brevity, we let $S = S_0(X)$ to denote our GN direction. Let us first examine $\nabla f(X+S) = 2R(X+S)(X+S)$. Noting that $J(X)(S) = -\mathcal{P}_J(R(X))$ and $(I-\mathcal{P}_J)R(X) = (I-\mathcal{P}_X)(-A)(I-\mathcal{P}_X)$, we have

$$\begin{aligned}
\nabla f(X+S)/2 &= [R(X) + J(X)(S) + SS^{\mathrm{T}}](X+S) \\
&= (I-\mathcal{P}_X)R(X)(I-\mathcal{P}_X)S + SS^{\mathrm{T}}(X+S) \\
&= (I-\mathcal{P}_X)(-A)(I-\mathcal{P}_X)S + SS^{\mathrm{T}}(X+S) \\
&= (I-\mathcal{P}_X)A(I-\mathcal{P}_X)\nabla f(X)(X^{\mathrm{T}}X)^{-1} + O(\|\nabla f(X)\|^2),
\end{aligned}$$

where the last equality due to $S = -\frac{1}{2}(I-\mathcal{P}_X/2)\nabla f(X)(X^{\mathrm{T}}X)^{-1} = O(\|\nabla f(X)\|)$. Hence, we obtain

$$\|\nabla f(X+S)\|_{\mathrm{F}} \leq \|T(X)\|_2 \, \|\nabla f(X)\|_{\mathrm{F}} + O(\|\nabla f(X)\|_{\mathrm{F}}^2), \tag{48}$$

where $T(X) = (X^{\mathrm{T}}X)^{-1} \otimes (I-\mathcal{P}_X)A(I-\mathcal{P}_X)$.

It is straightforward to see that $f(X^i) \to f^*$ implies that $\{X^i\}$ approaches the solution set $\mathcal{L}(f^*)$. In view of

(48), we have

$$\limsup_{i\to\infty} \frac{\|\nabla f(X^{i+1})\|_{\mathrm{F}}}{\|\nabla f(X^i)\|_{\mathrm{F}}} \le \limsup_{i\to\infty} \left(\|T(X^i)\|_2 + O(\|\nabla f(X^i)\|_{\mathrm{F}})\right) = \limsup_{i\to\infty} \|T(X^i)\|_2.$$

Recall that all global minimizers have the form $X^* = Q_k \Lambda_k^{1/2} V^{\mathrm{T}}$ where $V \in \mathbb{R}^{k\times k}$ can be arbitrary but must satisfy $V^{\mathrm{T}}V = I$. There must hold

$$\lambda_{\max}\left((X^*)^{\mathrm{T}}X^*\right)^{-1} = 1/\lambda_k, \ \text{ and } \ \lambda_{\max}\left(I - \mathcal{P}_{X^*}\right)A(I - \mathcal{P}_{X^*}) = \lambda_{k+1},$$

which gives $\|T(X^*)\|_2 = \lambda_{k+1}/\lambda_k$. Since $\{X^i\}$ must approach the set of minimizers (even if it does not converge to any particular one), it holds $\lim_{i\to\infty} \|T(X^i)\|_2 = \|T(X^*)\|_2$ by continuity. Therefore,

$$\limsup_{i\to\infty} \frac{\|\nabla f(X^{i+1})\|_{\mathrm{F}}}{\|\nabla f(X^i)\|_{\mathrm{F}}} \le \frac{\lambda_{k+1}}{\lambda_k} < 1,$$

which completes the proof. □

4.6. Discussions on Convergence Results. The global convergence of Algorithm 1 in Theorem 4.1 involves a few quantities, namely, the step size α, the correction tolerance δ and correction step P, that are computable but not without considerable overheads. This is why Algorithm 1 is designated as a theoretical algorithm.

Theorem 4.2 is an existence result for convergence regions of the unit-step GN method, i.e., our Algorithm 2 which is parameter-free and practical. We reiterate that such a convergence region does not exist in general for nonlinear least squares problems unless the optimal residual is sufficiently small. We establish the existence of convergence regions in Theorem 4.2 regardless of residual sizes, although the sizes of these convergence regions are still unclear.

The Q-linear rate obtained in Theorem 4.3 is typical for block algorithms such as the classic simultaneous subspace iterations (i.e., power method). It does give an indication that, when used as it is and under unfavorable conditions, the GN algorithm can become excessively slow in an asymptotic sense. For this reason, the pure GN algorithm should be used with caution when a high accuracy is required.

5. Numerical Experiments. In this section, we evaluate the numerical performance of the proposed GN method for computing principal eigenspaces or/and dominant singular value decompositions (SVD). The test problems include randomly generated matrices with specified singular values, and matrices from applications of low-rank matrix completion and robust principal component analysis. Most of these problems have the property that the optimal residual of the SLRP model is relatively small. In addition, for test problems from the two classes of applications, dominant SVDs need to be computed on a sequence of converging matrices, making our GN method a suitable choice because of its warm starting ability.

Given a matrix $A \in \mathbb{R}^{n\times m}$ and a positive integer $k \ll \min(m,n)$, a k-dimensional principal subspace for the range space of A can be computed from solving the SLRP model with the symmetric matrix AA^{T}; that is,

$$\min_{X\in\mathbb{R}^{n\times k}} \ \frac{1}{2}\|XX^{\mathrm{T}} - AA^{\mathrm{T}}\|_{\mathrm{F}}^2. \tag{49}$$

Once a solution X to the SLRP model (49) is computed along with $Y = X\left(X^{\mathrm{T}}X\right)^{-1}$, one can easily construct an approximately optimal rank-k approximation of A, that is, $\mathcal{P}_X A = X(Y^{\mathrm{T}}A)$. When it is required, a rank-k dominant SVD can also be readily computed from the solution X by performing a suitable version of the well-known Rayleigh-Ritz procedure.

We have implemented the practical version of SLRPGN , i.e., Algorithm 2, in MATLAB in which we make a single call to a Rayleigh-Ritz procedure at the end to calculate principal eigenpairs whenever required. Since the computations of the objective function value $f(X^i)$ and the gradient value $\nabla f(X^i)$ are not necessary in the execution of SLRPGN iterations, as the default termination rule we use the following criterion

$$\left|1 - \frac{\|X^{i-1}\|_{\mathrm{F}}}{\|X^i\|_{\mathrm{F}}}\right| < \mathrm{tol}, \tag{50}$$

for a prescribed tolerance $\mathrm{tol} > 0$, which measures the relative change in two consecutive iterates.

We adopt this practical termination rule because of two reasons: (i) it has a low computational cost (as is evidenced by Figure 5.1(a)), and (ii) it has been empirically proven to work well for SLRPGN when required solution accuracies are low or moderate. Therefore, termination rule (50) is used in our application-related experiments where warm-starting is used and low-accuracy solutions are sufficient.

A shortcoming of this rule is that the magnitude of the left-hand side in (50) does not always faithfully reflect the accuracy in function value $f(X^i)$ or gradient norm $\|\nabla f(X^i)\|_{\mathrm{F}}$ when seeking higher accuracies. As it is, SLRPGN is generally not effective in computing high-accuracy eigenpairs (after all, it only makes a single Rayleigh-Ritz call). However, combined with other techniques SLRPGN has demonstrated a potential to be useful in constructing new block eigensolvers capable of reaching higher accuracies. This potential will be illustrated in Section 5.4 where numerical experiments are performed on a set of commonly-used, sparse test matrices with a "standard" termination rule

$$\text{maxres} = \max_{i=1,\ldots,k} \left\{ \frac{\|Au_i - \theta_i u_i\|_2}{\max(1, |\theta_i|)} \right\} \leq tol, \tag{51}$$

where (θ_i, u_i), $i = 1, \cdots, k$, are computed approximate eigenpairs.

We compare the performance of SLRPGN with several state-of-the-art SVD solvers, including the Matlab built-in function EIGS which interfaces with the Fortran package ARPACK [16], a Matlab version of the solver LANSVD in the package PROPACK [14][1], and the Matlab solver LMSVD [19][2] which has been extensively compared with another block algorithm LOBPCG [12] and the classic subspace iteration method in [19]. The dedicated Matlab built-in interface SVDS for singular value decomposition is not used since numerical experiments indicate that the performance of EIGS (applied to AA^{T}) is generally much superior to that of SVDS. The Matlab version of LANSVD performs re-orthogonalization calculations by calling a Fortran subroutine via Matlab's MEX external interface. In all of our experiments, we provide A and A^{T} as linear operators to all solvers instead of forming the matrix product AA^{T}.

All the experiments are preformed on a workstation with two twelve-core Intel Xeon E5-2697 CPUs and 128GB of memory running Ubuntu 12.04 and MATLAB 2013b. The reported runtimes are wall-clock times. On such a multi-core computer, the memory access pattern and communication overhead already have a major impact on computing time. In Matlab 2013b, dense linear algebra operations have been generally well optimized by using BLAS and LAPACK tuned to the CPU processors in use. On the other hand, we have observed that some sparse linear algebra operations in Matlab 2013b seem to have not been as highly optimized. In particular, when doing multiplications between a sparse matrix and a dense vector or matrix (denoted by "SpMM") the performance of Matlab's own version of SPMM can differ significantly from that of the corresponding routine in Intel's Math Kernel Library (MKL)[3], which is named "`mkl_dcscmm`" and invoked via Matlab's MEX external interface in our experiments.

[1] Downloadable from `http://soi.stanford.edu/~rmunk/PROPACK`.
[2] Downloadable from `http://www.caam.rice.edu/~yzhang/LMSVD/lmsvd.html`.
[3] See `http://software.intel.com/en-us/intel-mkl` (version 11.0.2 on our workstation).

For brevity, we denote Matlab's SpMM by "SpMM-Matlab" and MKL's by "SpMM-MKL". Although SpMM-Matlab may be slightly faster than SpMM-MKL when a sparse matrix is multiplied by a single dense vector, it can become many times slower than SpMM-MKL when it is multiplied by sufficiently many dense columns altogether on multicore CPUs. For this reason, in our numerical experiments, SpMM-MKL is used as the default routine, although in section 5.2 we also include side-by-side comparisons between the results of SpMM-Matlab and those of SpMM-MKL to demonstrate their differences.

5.1. Comparison on Random Examples. The test examples in this subsection are generated as follows. Given dimension n, let

$$\hat{v}_i = \begin{cases} i^{-0.01}, & i \le 0.05n, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and} \quad v = \frac{\sqrt{n}}{\|\hat{v}\|_2}\hat{v}. \tag{52}$$

We first generate a random matrix A using the Matlab command $\texttt{A} = \texttt{gallery}(\texttt{'randcolu'}, \texttt{v}, \texttt{n}, 1)$ that has singular values given by the vector v defined in (52), then we add random noise to A by setting

$$A := A + \frac{\|A\|_{\rm F}}{10}\frac{B}{\|B\|_{\rm F}},$$

where B is produced by the Matlab command: $\texttt{sprandn}(\texttt{n}, \texttt{n}, \texttt{nnz}(\texttt{B})/\texttt{n}^2)$, representing sparse random noise. It is important to note that such an A has two clusters of singular values: one cluster of large singular values σ_k for $k \le 0.05n$, and one cluster of small singular values σ_k for $k > 0.05n$. Consequently, the eigenvalues of $AA^{\rm T}$ in the SLRP model (49), i.e., $\lambda_k = \sigma_k^2$ for all k, also form two distinct clusters, separated at boundary $k = 0.05n$.

We compare SLRPGN with a gradient method for solving (49) using the Barzilar-Borwein step size [1]. We call this gradient method SLRPBB, which is essentially a variant of the EigPen algorithm in [31] due to the equivalence in Proposition (2.3). Let $S^i = X^i - X^{i-1}$ and $Y^i = \nabla f(X^i) - \nabla f(X^{i-1})$. We use the following BB step-size formula: $\alpha^i_{\rm BB} = {\rm tr}((S^i)^{\rm T}Y^i)/\|Y^i\|_{\rm F}^2$. Then the next SLRPBB iterate is obtained by a back-tracking line search $X^{i+1} = X^i - \alpha^i \nabla f(X^i)$, where $\alpha^i = \alpha^i_{\rm BB}\beta^j$ for some $\beta \in (0,1)$ ($\beta = 0.25$ is used in our experiments), and j is the smallest nonnegative integer satisfying a loose back-tracking condition $f(X^i - \alpha^i_{\rm BB}\beta^j \nabla f(X^i)) \le 2f(X^i)$.

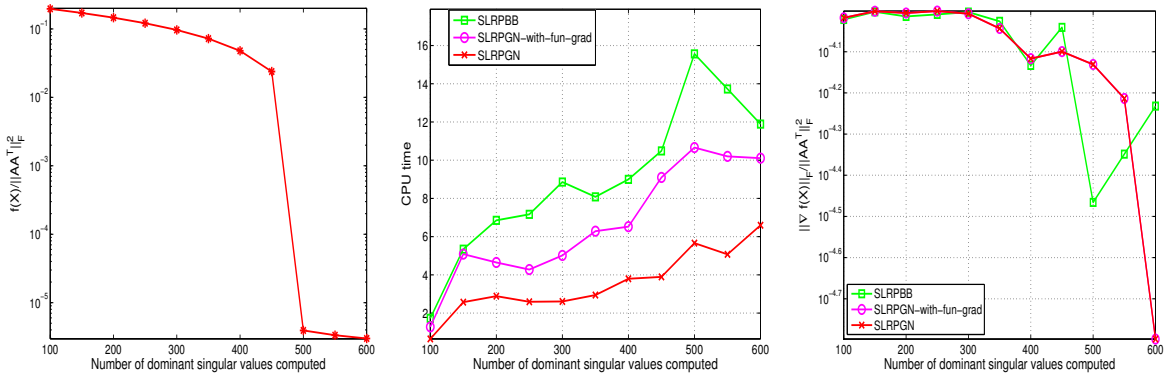
A main purpose of this comparison between SLRPGN and SLRPBB is to assess the effect of the least squares residual size on the convergence behavior of these two algorithms, for which we choose to examine the convergence history of the gradient norm. As is mentioned, the default termination rule does not give a precise control over the gradient norm. In order to avoid pre-mature or delayed terminations, we directly use a scaled gradient norm as the stopping criterion for the results in Figure 5.1 and Figure 5.2; that is,

$$\|\nabla \hat{f}(X^i)\|_{\rm F} = \frac{\|\nabla f(X^i)\|_{\rm F}}{\|AA^{\rm T}\|_{\rm F}^2} < {\rm tol}, \tag{53}$$

where $\hat{f}(X) = f(X)/\|AA^{\rm T}\|_{\rm F}^2$ is a scaled objective function so that $f(0) = 1/2$.

For $n = 10000$, we run SLRPBB and SLRPGN with $k \in \{100, 150, \ldots, 550, 600\}$, and apply the termination rule (53) to both algorithms with ${\rm tol} = 10^{-4}$. By construction, we know that the first 500 eigenvalues of $AA^{\rm T}$ are relatively large and the rest are much smaller. Consequently, the residual is relatively large for $k < 500$, and much smaller for $k \ge 500$, as can be clearly seen in Figure 5.1(a). The running-time results for this test are presented in Figure 5.1(b), where the line "SLRPGN -with-fun-grad" denotes a version of SLRPGN that evaluates both the objective function and gradient norm at each iteration (recall that by default SLRPGN does not compute these quantities). Figure 5.1(c) contains the scaled gradient norm values, $\|\nabla f(X)\|_{\rm F}/\|AA^{\rm T}\|_{\rm F}^2$, attained at the last iteration for all k. As can be seen

from Figure 5.1(b)-(c), (i) SLRPGN is several times faster than SLRPBB across the board while achieving the same level of accuracy in gradient norm; and (ii) not evaluating the function and gradient values does save a significant amount of time. However, by examining Figure 5.1(a) against (b), we do not see clear correlations between the residual size and the runtime at the given level of accuracy (i.e., $\mathrm{tol} = 10^{-4}$).



(a) Optimal residual (b) Runtime (c) gradient norm

FIG. 5.1. SLRPBB *and* SLRPGN *with varying number of computed singular values on the random example "randcolu"* $\left(\mathrm{tol} = 10^{-4}\right)$

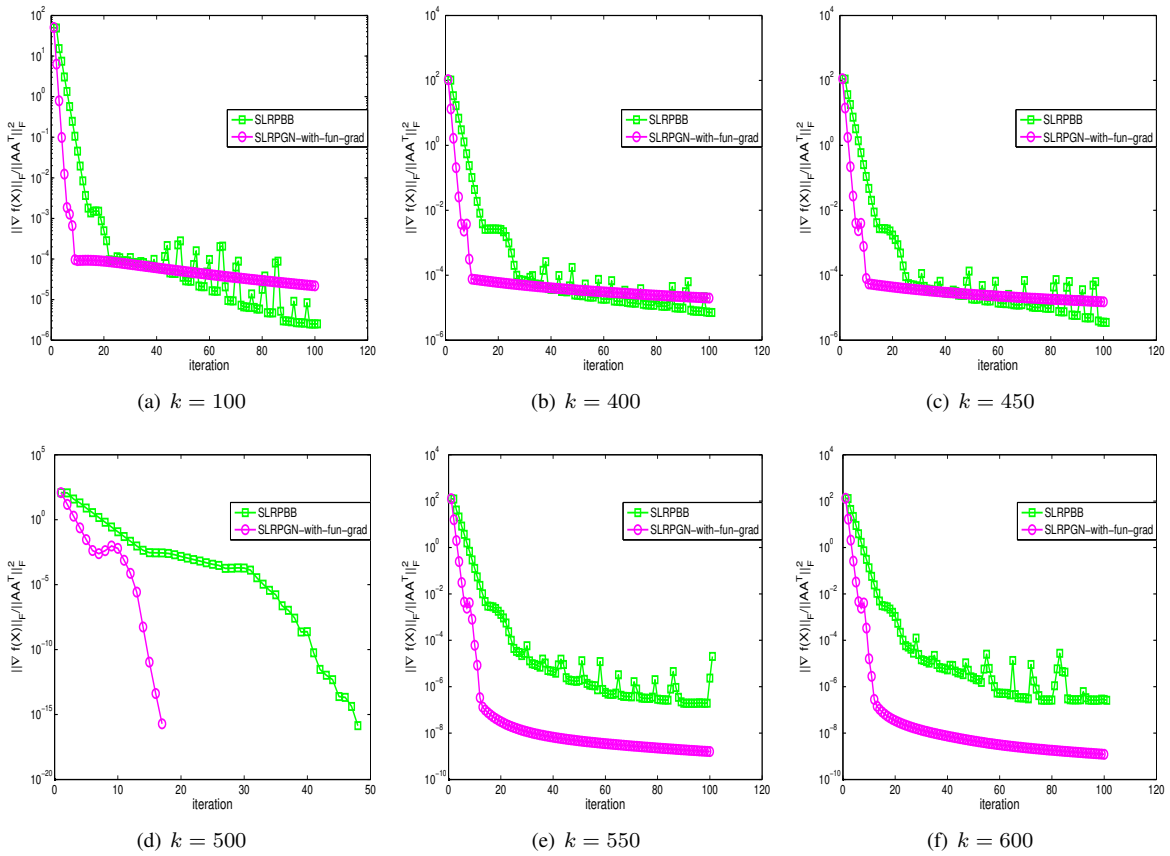
In order to clearly see the effect of the residual size on SLRPGN 's performance, we tighten the tolerance to $\mathrm{tol} = 10^{-15}$ in (53), set the maximum number of iterations to 100, and run the two codes SLRPGN and SLRPBB again for the given set of k values. Figure 5.2 presents the iteration history of the scaled gradient norm $\|\nabla f(X)\|_{\mathrm{F}}/\|AA^{\mathrm{T}}\|_{\mathrm{F}}^2$ for six different values of k, five of which are either at or around $k = 500$ where we know that the residual has a dramatic change in size, as is depicted in Figure 5.1(a).

The following observations can be drawn from Figure 5.2.

- SLRPGN exhibits a smoother and more predictable pattern of convergence than that of SLRPBB. With the exception for the case of $k = 500$, the pattern consists of two stages: an initial stage of fast convergence and an asymptotic stage of slower convergence.
- In all cases, SLRPGN converges notably faster than SLRPBB in the initial stage. For large residual problems ($k < 500$), this initial stage extends slightly beyond $\|\nabla f(X)\|_{\mathrm{F}}/\|AA^{\mathrm{T}}\|_{\mathrm{F}}^2 < 10^{-4}$; while for small residual problems ($k > 500$), it extends further to the level of $\|\nabla f(X)\|_{\mathrm{F}}/\|AA^{\mathrm{T}}\|_{\mathrm{F}}^2 < 10^{-7}$.
- For large residual problems, the asymptotic convergence rate of SLRPGN (with $\alpha = 1$), can be slightly slower that of SLRPBB. Therefore, the latter can eventually catch up if the codes are allowed to run long enough. However, for small residual problems, the asymptotic convergence rate of SLRPGN is either faster than or similar to that of SLRPBB. Therefore, SLRPBB does not have a chance to catch up with SLRPGN .
- The case of $k = 500$ is special, for which both codes reach the very tight tolerance of $\mathrm{tol} = 10^{-15}$ within far less than 100 iterations allowed, and both show rapid asymptotic convergence. Theorem 4.2 indicates that the asymptotic rate of convergence for SLRPGN is limited by the ratio λ_{k+1}/λ_k. On this given test problem set, this ratio is extremely close to one (> 0.995) in all cases except for $k = 500$ where $\lambda_{501}/\lambda_{500} = 0.0073$ is close to zero.

We add two more comments below to reiterate an important point and bring up a less obvious point.

- The effectiveness of SLRPGN on small residual problems should be evident. On large residual problems, due to the existence of a fast converging initial stage, SLRPGN can still be quite efficient as long as a high accuracy is not required.



(a) $k = 100$ (b) $k = 400$ (c) $k = 450$

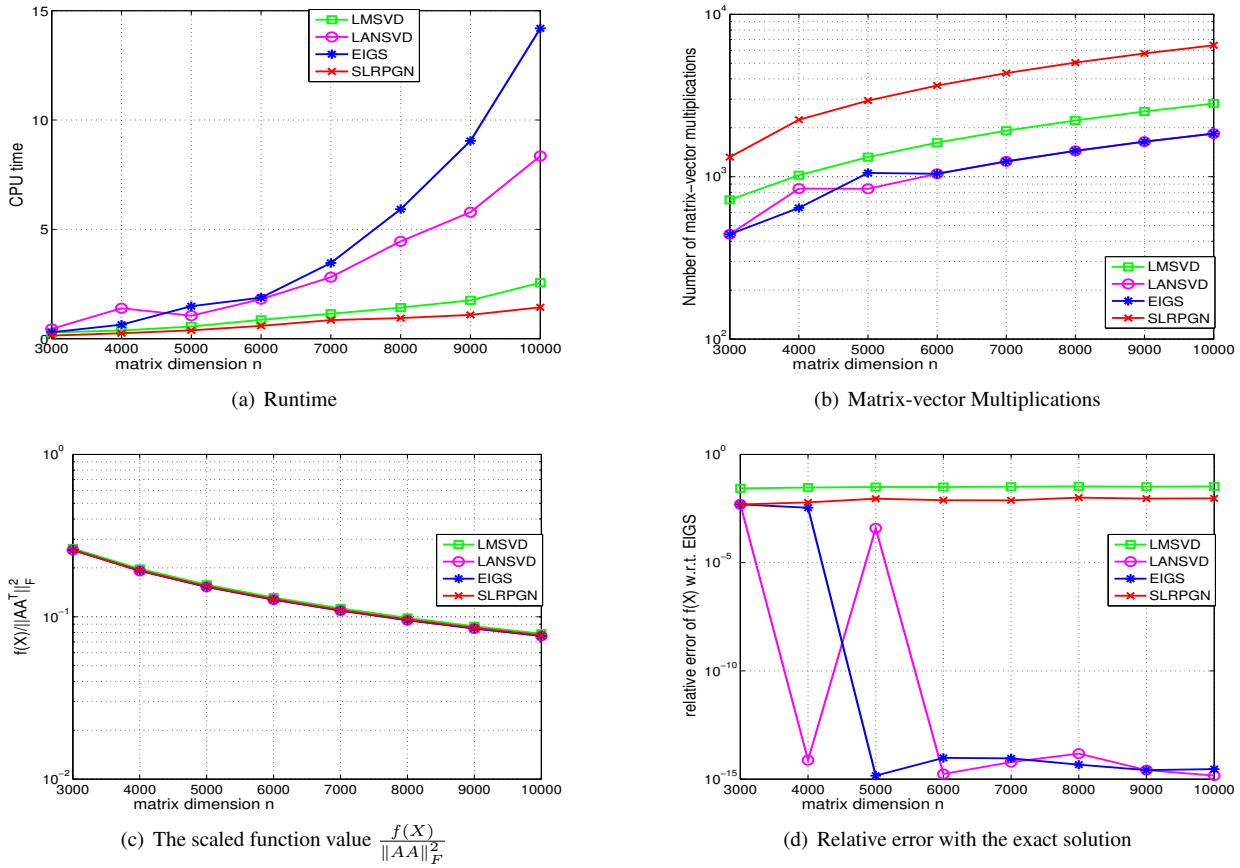
(d) $k = 500$ (e) $k = 550$ (f) $k = 600$

FIG. 5.2. *Gradient norm* $\|\nabla f(X)\|_{\mathrm{F}}/||AA^{\mathrm{T}}||_{\mathrm{F}}^2$ *versus iteration for the random example "randcolu"*

- For large residual problems, the step size $\alpha = 1$, although performing reliably in SLRPGN , is sub-optimal. For example, a simple experiment shows that $\alpha = 1.5$ gives better convergence for the case of $k = 100$. An investigation of various algorithmic options for large residual problems is worthwhile, but beyond the scope of the present study.

Our next experiment is to compare the performance of EIGS , LANSVD, LMSVD and SLRPGN on the same test problem set with $n = 3000, 4000, \ldots, 10000$ and $k = 0.05n - 40$. Since the selected k values are less than $0.05n$, the residuals for these problems are relatively large. In addition, the asymptotic convergence rate of SLRPGN is expected to be slow since, with $k < 0.05n$, both λ_k and λ_{k+1} belong to the large-value cluster and their ratio is close to 1. For such problems, we only require a moderate accuracy. Specifically, SLRPGN is terminated when (50) is satisfied with $\mathrm{tol} = 10^{-4}$, and all other solvers are stopped using their own termination rules with the tolerance 10^{-2}. The runtime, the total number of matrix-vector multiplications, the scaled objective function value $f(X)/\|AA\|_F^2$, and the relative error in the final objective value, with respect to an "exact" solution computed by EIGS with the default tolerance, are reported in Figure 5.3. We observe from Figure 5.3(a) that, with the given tolerance, SLRPGN is notably faster than EIGS and LANSVD, and also slightly faster than LMSVD. The speed gaps grows quickly as the dimension n increases. It is particularly worth noting that SLRPGN maintains its speed advantage in spite of taking considerably more matrix-vector multiplications, as can be seen from Figure 5.3(b), revealing a great impact of algorithmic concurrency on multi-core computing time. The scaled objective function values $f(X)/\|AA\|_F^2$ attained by all solvers are undistinguishable in Figure 5.3(c). Details on the obtained accuracy are given in Figure 5.3(d) where we see that both EIGS and LANSVD attain an accuracy close to machine precision for $n \geq 6000$ despite the

large tolerance 10^{-2} for termination. This phenomenon may be attributable to the fact that Krylov subspace methods need to maintain highly accurate orthonormal bases for these algorithms to succeed. As such, these Krylov subspace methods may not be well suited for computing approximate solutions of a low to moderate accuracy.



(a) Runtime

(b) Matrix-vector Multiplications

(c) The scaled function value $\frac{f(X)}{\|AA\|_F^2}$

(d) Relative error with the exact solution

FIG. 5.3. *Comparison between different eigensolvers with varying matrix dimensions on the random example "randcolu"*

5.2. Matrix Completion. In this subsection, we compare the performance of three solvers, LANSVD, SLRPGN and LMSVD, embedded in two nuclear-norm minimization algorithms for solving the matrix completion problem [5, 6, 21]. One is the singular value thresholding (SVT) algorithm [11] applying soft-thresholding operations on the singular values of a certain matrix at each iteration. Another is the accelerated proximal gradient (NNLS) algorithm [30] based on an extension to the iterative shrinkage-thresholding algorithm [2]. Both algorithms require computing a singular value decomposition at each iteration which dominates their computational costs as the size of the underlying matrices increases. The default SVD solver used in both SVT and NNLS is LANSVD. We do the comparison by changing LANSVD to LMSVD and SLRPGN . The terminating rules of each SVD solver are kept as they are, but the tolerance value of each rule is tuned to a more relaxed value than the default while maintaining the default accuracy of the SVT or NNLS. We do such a tuning to give all solvers the benefit of achieving highest efficiency possible for the given final solution accuracy.

The SVT algorithm [11] solves the problem

$$\min_{X\in\mathbb{R}^{m\times n}} \|X\|_*, \ \mathbf{s.t.}\ \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M),$$

where $\|X\|_*$, the nuclear or trace norm, is the sum of singular values of X, Ω is the set of indices for given entries of M, and $\mathcal{P}_\Omega$ is the projection onto the subspace of sparse matrices with nonzeros restricted to Ω. Starting from $Y^0 \in \mathbb{R}^{m\times n}$, and using a fixed $\tau > 0$ and a sequence of $\delta_k > 0$, the algorithm computes

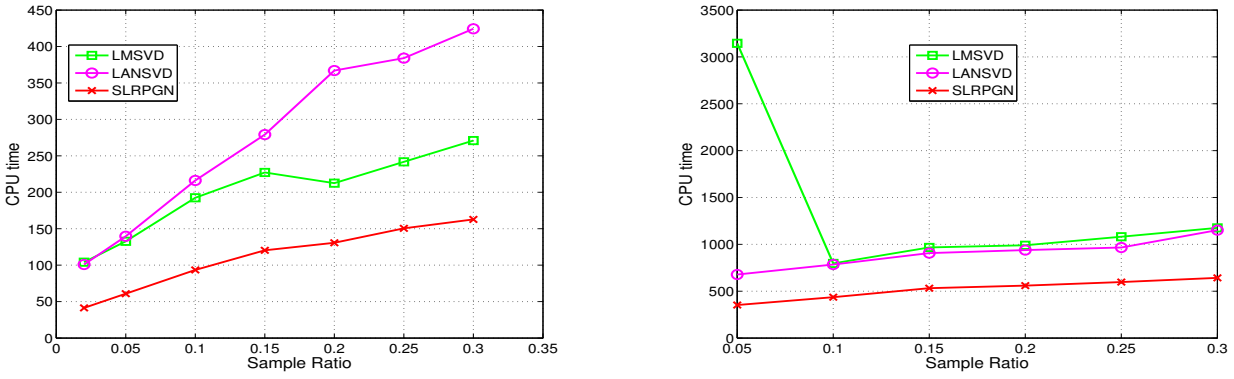
$$\begin{cases} X^k = \mathrm{shrink}(Y^{k-1}, \tau), \\ Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k), \end{cases}$$

where $\mathrm{shrink}(Y, \tau)$ is the matrix shrinkage operator [11] requiring the computation of the SVD of Y. The code SVT[4] always stores Y^k in the sparse matrix format. We tested both SpMM-Matlab and SpMM-MKL for matrix multiplications involving Y^k.

We run SVT code with each SVD solver embedded on a set of randomly generated problems. The test matrices $M \in \mathbb{R}^{m\times n}$ with rank r are created randomly by the following standard procedure: two random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ with i.i.d. standard Gaussian entries are first generated and then $M = M_L M_R^\top$ is assembled; then a subset Ω of p index pairs is sampled uniformly at random. The ratio $p/(mn)$ is denoted by "SR" (sampling ratio). A summary of the computational results using SpMM-Matlab and SpMM-MKL are presented in Tables 5.1 and 5.2, respectively. In these tables, "iter" denotes the number of iterations used, "svp" denotes the rank of the recovered solution, "time" denotes the runtime measured in seconds and mse $= \|X - M\|_{\mathrm{F}}/\|M\|_{\mathrm{F}}$ denotes the relative error between the true and the recovered matrices M and X. To further illustrate the performance of the four tested solvers, in Figures 5.4 and 5.5 we plot the runtime of SVT with fixed parameters $n = 10^4$, $r = 10$ and $r = 50$, and varying sample ratios "SR".

The following observations should be clear from the numerical results:

- All three solvers used by SVT give the same number of iterations, attain the same final rank, and achieve the same accuracy, no matter whether SpMM-Matlab or SpMM-MKL is used.
- LANSVD with SpMM-Matlab is faster than with SpMM-MKL.
- Both SLRPGN and LMSVD with SpMM-MKL are faster than with SpMM-Matlab.
- SLRPGN with SpMM-MKL is faster than LANSVD with either SpMM-Matlab or SpMM-MKL. The observed speedup factor varies approximately from one to ten.
- SLRPGN is faster than LMSVD when either SpMM-Matlab or SpMM-MKL is used. The observed speedup factor ranges from slightly more than one to over nine.



(a) Exact rank $r = 10$ (b) Exact rank $r = 50$

FIG. 5.4. *SVT: Comparison with varying sampling ratios on $n = 10^4$ by using SpMM-Matlab*

[4]Downloadable from `http://svt.stanford.edu/`

TABLE 5.1
Results of SVT *on randomly generated matrix completion problems by using SpMM-Matlab*

			LANSVD				SLRPGN				LMSVD			
m=n	SR (%)	r	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
1000	0.20	10	79	10	7.45	1.31e-04	79	10	2.97	1.31e-04	79	10	8.87	1.31e-04
1000	0.30	10	62	10	7.08	1.17e-04	62	10	3.17	1.17e-04	62	10	8.62	1.17e-04
1000	0.40	10	53	10	6.02	1.15e-04	53	10	3.49	1.14e-04	53	10	8.10	1.15e-04
1000	0.30	50	169	69	85.77	2.12e-04	171	68	24.01	2.12e-04	500	50	179.15	5.34e-03
1000	0.40	50	110	50	34.26	1.60e-04	110	50	16.79	1.60e-04	113	50	44.45	1.46e-04
1000	0.50	50	86	50	31.20	1.40e-04	86	50	16.86	1.40e-04	86	50	41.08	1.40e-04
5000	0.10	10	53	10	58.24	1.08e-04	53	10	25.73	1.08e-04	53	10	54.62	1.08e-04
5000	0.20	10	42	10	79.48	1.04e-04	42	10	35.08	1.04e-04	42	10	74.41	1.04e-04
5000	0.30	10	38	10	100.49	9.23e-05	38	10	43.26	9.23e-05	38	10	66.60	9.23e-05
5000	0.10	50	107	50	241.42	1.59e-04	107	50	140.63	1.59e-04	500	50	1294.60	9.58e-04
5000	0.20	50	67	50	256.46	1.23e-04	67	50	128.02	1.22e-04	67	50	272.74	1.23e-04
5000	0.30	50	54	50	295.23	1.21e-04	54	50	173.73	1.21e-04	54	50	304.87	1.21e-04
10000	0.01	10	500	10	898.52	8.93e-04	500	10	173.39	5.33e-04	500	10	331.98	2.74e-02
10000	0.05	10	54	10	139.10	1.10e-04	54	10	55.79	1.10e-04	54	10	132.89	1.10e-04
10000	0.10	10	43	10	239.42	1.07e-04	43	10	88.17	1.07e-04	43	10	184.85	1.07e-04
10000	0.05	50	109	50	650.78	1.66e-04	109	50	335.96	1.65e-04	500	50	3080.68	2.32e-03
10000	0.10	50	69	50	750.31	1.29e-04	69	50	411.60	1.29e-04	69	50	769.16	1.29e-04
10000	0.15	50	57	50	906.39	1.16e-04	57	50	524.13	1.16e-04	57	50	952.76	1.16e-04

TABLE 5.2
Results of SVT *on randomly generated matrix completion problems by using SpMM-MKL*

			LANSVD				SLRPGN				LMSVD			
m=n	SR (%)	r	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
1000	0.20	10	79	10	10.23	1.31e-04	79	10	2.53	1.31e-04	79	10	8.84	1.31e-04
1000	0.30	10	62	10	8.69	1.17e-04	62	10	2.42	1.17e-04	62	10	7.10	1.17e-04
1000	0.40	10	53	10	8.37	1.15e-04	53	10	2.56	1.14e-04	53	10	5.78	1.15e-04
1000	0.30	50	169	69	113.97	2.12e-04	171	68	10.80	2.12e-04	500	50	107.54	7.55e-03
1000	0.40	50	110	50	45.96	1.60e-04	110	50	7.71	1.60e-04	113	50	26.26	1.46e-04
1000	0.50	50	86	50	41.28	1.40e-04	86	50	7.35	1.40e-04	86	50	22.29	1.40e-04
5000	0.10	10	53	10	84.04	1.08e-04	53	10	21.52	1.08e-04	53	10	32.71	1.08e-04
5000	0.20	10	42	10	176.15	1.04e-04	42	10	25.04	1.04e-04	42	10	38.88	1.04e-04
5000	0.30	10	38	10	214.84	9.23e-05	38	10	33.68	9.23e-05	38	10	39.73	9.23e-05
5000	0.10	50	107	50	330.28	1.59e-04	107	50	78.51	1.59e-04	500	50	651.54	9.60e-04
5000	0.20	50	67	50	482.52	1.23e-04	67	50	56.82	1.22e-04	67	50	112.86	1.23e-04
5000	0.30	50	54	50	704.29	1.21e-04	54	50	68.07	1.21e-04	54	50	115.74	1.21e-04
10000	0.01	10	500	10	1120.59	6.57e-04	500	10	112.04	8.86e-04	500	10	222.52	2.66e-02
10000	0.05	10	54	10	279.99	1.10e-04	54	10	43.35	1.10e-04	54	10	74.76	1.10e-04
10000	0.10	10	43	10	458.86	1.07e-04	43	10	61.79	1.07e-04	43	10	94.25	1.07e-04
10000	0.05	50	109	50	1291.43	1.66e-04	109	50	171.14	1.65e-04	500	50	1315.04	2.32e-03
10000	0.10	50	69	50	1514.00	1.29e-04	69	50	199.85	1.29e-04	69	50	303.23	1.29e-04
10000	0.15	50	57	50	1738.79	1.16e-04	57	50	248.05	1.16e-04	57	50	350.56	1.16e-04

Now we turn to the NNLS algorithm[5] which solves the regularized linear least problem:

$$\min_{X\in\mathbb{R}^{m\times n}} \ \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(X-M)\|_{\mathrm{F}}^2.$$

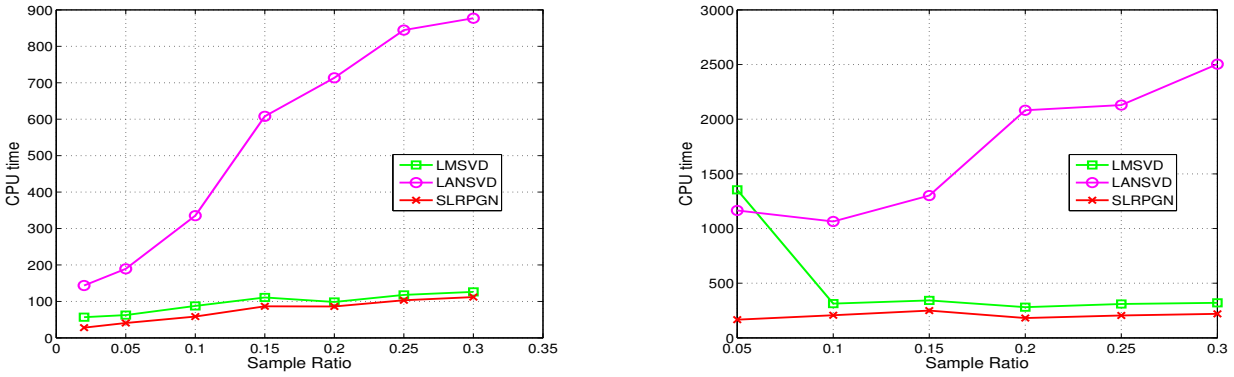
At the k-th iteration, it requires to compute the dominant SVD of a matrix of the form

$$A^k = \beta_1 U^k (V^k)^{\mathrm{T}} - \beta_2 U^{k-1}(V^{k-1})^{\mathrm{T}} - \beta_3 G^k,$$

where U^k, V^k, U^{k-1} and V^{k-1} are dense matrices, but G^k can be either dense (when SR $\geq 15\%$) or sparse (when SR $< 15\%$). The number of singular values to be computed by NNLS is set to one at the first iteration, and then can be increased or decreased in the subsequent iterations determined by certain rules. We examine the performance of NNLS separately according to whether G^k is sparse or dense. SpMM-Matlab and SpMM-MKL are tested only when G^k is sparse.

We first test NNLS with each SVD solver on random problems generated in the same way as in the case for SVT.

[5] http://www.math.nus.edu.sg/~mattohkc/NNLS.html



(a) Exact rank $r = 10$ (b) Exact rank $r = 50$

FIG. 5.5. *SVT: Comparison with varying sampling ratios on* $n = 10^4$ *by using SpMM-MKL*

A summary of computational results for the dense cases is presented in Table 5.3. The results of the sparse problems using SpMM-Matlab and SpMM-MKL are reported in Tables 5.4 and 5.5, respectively. We further present the runtime of NNLS using the three solvers in Figures 5.6 and 5.7 with parameters $n = 10^4$, $r = 10$ and $r = 50$, and varying sample ratios. We can make the following observations:

- On dense problems, SLRPGN is about one to three times faster than LANSVD.
- On sparse problems, all solvers give the same number of iterations, attain the same rank and achieve the same accuracy for either SpMM-Matlab or SpMM-MKL.
- LANSVD with SpMM-Matlab is faster than with SpMM-MKL.
- Both SLRPGN and LMSVD with SpMM-MKL are faster than with SpMM-Matlab.
- SLRPGN with SpMM-MKL is faster than LANSVD with either SpMM-Matlab or SpMM-MKL. The observed speedup factor varies approximately from one to four.
- The performances of SLRPGN and LMSVD are very similar. Being block algorithms, both have the advantage over LANSVD in their warm-start ability, which appears to be the main source of speedups in the NNLS experiments (also see the remark at the end of this subsection).

TABLE 5.3
Results of NNLS *on random dense examples*

			LANSVD				SLRPGN				LMSVD			
m=n	SR (%)	r	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
1000	25.0	50	55	50	9.66e+00	7.60e-04	55	50	3.35e+00	1.18e-03	50	50	3.48e+00	1.21e-03
1000	35.0	50	42	50	6.21e+00	5.42e-04	49	50	3.13e+00	4.40e-04	42	50	3.01e+00	5.41e-04
1000	49.9	50	39	50	5.76e+00	1.81e-04	39	50	2.61e+00	1.81e-04	39	50	3.33e+00	1.80e-04
1000	25.0	100	100	150	6.71e+01	3.55e-01	100	150	1.17e+01	3.87e-01	100	150	1.60e+01	3.54e-01
1000	35.0	100	64	100	2.08e+01	1.45e-03	64	100	5.96e+00	1.48e-03	64	100	7.38e+00	1.44e-03
1000	49.9	100	54	100	1.68e+01	4.70e-04	48	100	4.71e+00	1.05e-03	54	100	6.68e+00	4.70e-04
5000	25.0	50	36	50	5.38e+01	1.41e-04	36	50	2.75e+01	1.42e-04	36	50	2.81e+01	1.41e-04
5000	35.0	50	34	50	5.60e+01	1.60e-04	34	50	3.15e+01	1.61e-04	34	50	3.17e+01	1.60e-04
5000	50.0	50	32	50	6.08e+01	1.46e-04	32	50	3.68e+01	1.48e-04	32	50	3.72e+01	1.46e-04
5000	25.0	100	49	100	1.34e+02	2.83e-04	49	100	5.37e+01	2.88e-04	49	100	6.75e+01	2.83e-04
5000	35.0	100	45	100	1.27e+02	1.74e-04	45	100	5.87e+01	1.76e-04	45	100	6.82e+01	1.74e-04
5000	50.0	100	43	100	1.21e+02	1.43e-04	43	100	7.11e+01	1.44e-04	43	100	7.81e+01	1.43e-04
10000	25.0	50	32	50	1.53e+02	1.17e-04	32	50	8.88e+01	1.17e-04	32	50	8.97e+01	1.17e-04
10000	35.0	50	32	50	1.86e+02	1.16e-04	32	50	1.13e+02	1.16e-04	32	50	1.11e+02	1.16e-04
10000	50.0	50	31	50	2.08e+02	1.39e-04	32	50	1.44e+02	1.15e-04	31	50	1.36e+02	1.39e-04
10000	25.0	100	43	100	3.54e+02	1.59e-04	44	100	1.61e+02	1.74e-04	43	100	1.55e+02	1.59e-04
10000	35.0	100	42	100	3.63e+02	1.32e-04	42	100	1.96e+02	1.33e-04	42	100	1.93e+02	1.32e-04
10000	50.0	100	39	100	3.68e+02	3.70e-04	40	100	2.38e+02	1.18e-04	39	100	2.23e+02	3.70e-04

The next experiment using NNLS is on low-rank matrix approximation problems based on two "real" data sets: the

TABLE 5.4
Results of NNLS *on random sparse examples by using SpMM-Matlab*

			LANSVD				SLRPGN				LMSVD			
m=n	SR (%)	r	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
10000	2.0	10	45	10	1.72e+01	1.86e-04	45	10	1.58e+01	1.54e-04	45	10	2.03e+01	1.86e-04
10000	5.0	10	35	10	2.66e+01	1.30e-04	35	10	2.52e+01	1.30e-04	35	10	3.36e+01	1.30e-04
10000	10.0	10	30	10	4.33e+01	1.03e-04	30	10	4.35e+01	1.03e-04	30	10	5.14e+01	1.03e-04
10000	14.0	10	28	10	5.10e+01	1.07e-04	29	10	5.71e+01	1.09e-04	28	10	6.68e+01	1.07e-04
10000	2.0	50	100	77	1.84e+02	3.42e-01	100	50	1.47e+02	9.88e-03	100	50	1.35e+02	1.09e-01
10000	5.0	50	50	50	1.10e+02	2.50e-04	48	50	9.49e+01	2.52e-04	50	50	1.16e+02	2.50e-04
10000	10.0	50	37	50	1.32e+02	1.66e-04	37	50	1.35e+02	1.67e-04	37	50	1.34e+02	1.66e-04
10000	14.0	50	35	50	1.61e+02	1.29e-04	35	50	1.57e+02	1.30e-04	35	50	1.59e+02	1.29e-04
50000	2.0	10	38	10	3.67e+02	1.09e-04	38	10	3.57e+02	1.09e-04	38	10	4.29e+02	1.09e-04
50000	5.0	10	31	10	7.00e+02	9.62e-05	31	10	6.16e+02	9.62e-05	31	10	7.48e+02	9.62e-05
50000	10.0	10	28	10	1.26e+03	1.09e-04	28	10	1.18e+03	1.09e-04	28	10	1.44e+03	1.09e-04
50000	14.0	10	28	10	1.70e+03	1.03e-04	28	10	1.50e+03	1.03e-04	28	10	1.79e+03	1.03e-04
50000	2.0	50	49	50	1.06e+03	1.96e-04	49	50	1.42e+03	1.62e-04	49	50	1.25e+03	1.96e-04
50000	5.0	50	37	50	2.06e+03	1.74e-04	37	50	1.79e+03	1.77e-04	37	50	1.80e+03	1.74e-04
50000	10.0	50	32	50	3.39e+03	1.17e-04	32	50	3.26e+03	1.17e-04	32	50	3.14e+03	1.17e-04
50000	14.0	50	31	50	4.26e+03	1.10e-04	31	50	3.97e+03	1.10e-04	31	50	3.81e+03	1.10e-04

TABLE 5.5
Results of NNLS *on random sparse examples by using SpMM-MKL*

			LANSVD				SLRPGN				LMSVD			
m=n	SR (%)	r	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
10000	2.0	10	45	10	1.86e+01	1.86e-04	45	10	1.01e+01	1.54e-04	45	10	1.09e+01	1.86e-04
10000	5.0	10	35	10	3.25e+01	1.30e-04	35	10	1.69e+01	1.30e-04	35	10	1.72e+01	1.30e-04
10000	10.0	10	30	10	5.37e+01	1.03e-04	30	10	2.73e+01	1.03e-04	30	10	2.73e+01	1.04e-04
10000	14.0	10	28	10	6.58e+01	1.07e-04	29	10	3.48e+01	1.09e-04	28	10	3.26e+01	1.07e-04
10000	2.0	50	100	77	2.32e+02	3.42e-01	100	50	5.21e+01	9.88e-03	100	50	6.10e+01	7.09e-02
10000	5.0	50	50	50	1.45e+02	2.50e-04	48	50	4.17e+01	2.52e-04	50	50	5.26e+01	2.50e-04
10000	10.0	50	37	50	1.70e+02	1.66e-04	37	50	5.44e+01	1.67e-04	37	50	5.42e+01	1.66e-04
10000	14.0	50	35	50	2.11e+02	1.29e-04	35	50	6.68e+01	1.30e-04	35	50	6.54e+01	1.29e-04
50000	2.0	10	38	10	5.33e+02	1.09e-04	38	10	2.07e+02	1.09e-04	38	10	1.96e+02	1.09e-04
50000	5.0	10	31	10	1.02e+03	9.62e-05	31	10	4.15e+02	9.62e-05	31	10	3.55e+02	9.62e-05
50000	10.0	10	28	10	1.95e+03	1.09e-04	28	10	7.27e+02	1.09e-04	28	10	6.10e+02	1.09e-04
50000	14.0	10	28	10	2.40e+03	1.03e-04	28	10	1.00e+03	1.03e-04	28	10	8.43e+02	1.03e-04
50000	2.0	50	49	50	1.73e+03	1.96e-04	49	50	4.34e+02	1.62e-04	49	50	4.18e+02	1.96e-04
50000	5.0	50	37	50	3.21e+03	1.74e-04	37	50	7.11e+02	1.77e-04	37	50	6.54e+02	1.74e-04
50000	10.0	50	32	50	5.26e+03	1.17e-04	32	50	1.23e+03	1.17e-04	32	50	1.07e+03	1.17e-04
50000	14.0	50	31	50	6.86e+03	1.10e-04	31	50	1.55e+03	1.10e-04	31	50	1.36e+03	1.10e-04

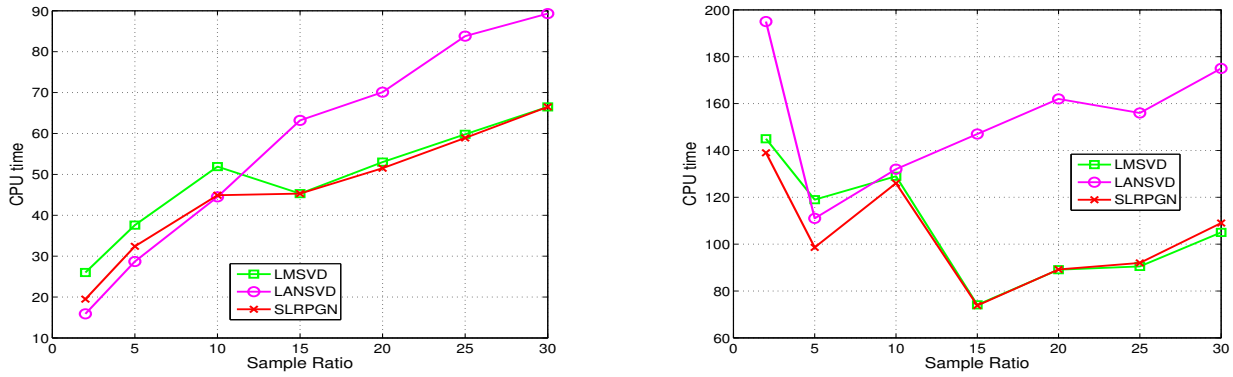
Jester joke data set and the MovieLens data set, which are also used in [30]. The Jester joke data set consists of four problems "jester-1", "jester-2", "jester-3" and "jester-all", where the last one is obtained by combining all of the first three. The MovieLens data set consists of three problems "movie-100K", "movie-1M" and "movie-10M". The results of using Matlab's own implementation and MKL are presented in Tables 5.6 and 5.7, respectively. The advantage of SLRPGN over LANSVD can be observed on the Movielens data set. Since the number of computed singular values can become larger than $\min(m,n)/3$ in the Jester joke data set, LMSVD is not suitable for these problems since it amounts to computing full singular value decompositions for these matrices.

TABLE 5.6
Results of NNLS *on real examples by using SpMM-Matlab*

		LANSVD				SLRPGN				LMSVD			
name	(m,n)	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
jester-1	(24983, 100)	26	93	1.01e+01	1.64e-01	27	69	9.01e+00	1.76e-01				
jester-2	(23500, 100)	26	93	9.93e+00	1.65e-01	26	79	1.08e+01	1.72e-01				
jester-3	(24938, 100)	24	83	7.43e+00	1.16e-01	27	74	7.83e+00	1.24e-01				
jester-all	(73421, 100)	26	93	2.74e+01	1.58e-01	26	82	3.32e+01	1.62e-01				
moive-100K	(943, 1682)	34	100	7.90e+00	1.28e-01	35	100	1.60e+00	1.26e-01	51	100	4.47e+00	3.56e-01
moive-1M	(6040, 3706)	50	100	4.04e+01	1.42e-01	50	100	2.56e+01	1.43e-01	50	100	6.11e+01	1.42e-01
moive-10M	(71567, 10677)	54	100	5.31e+02	1.26e-01	57	100	3.82e+02	1.27e-01	54	100	3.69e+02	1.26e-01

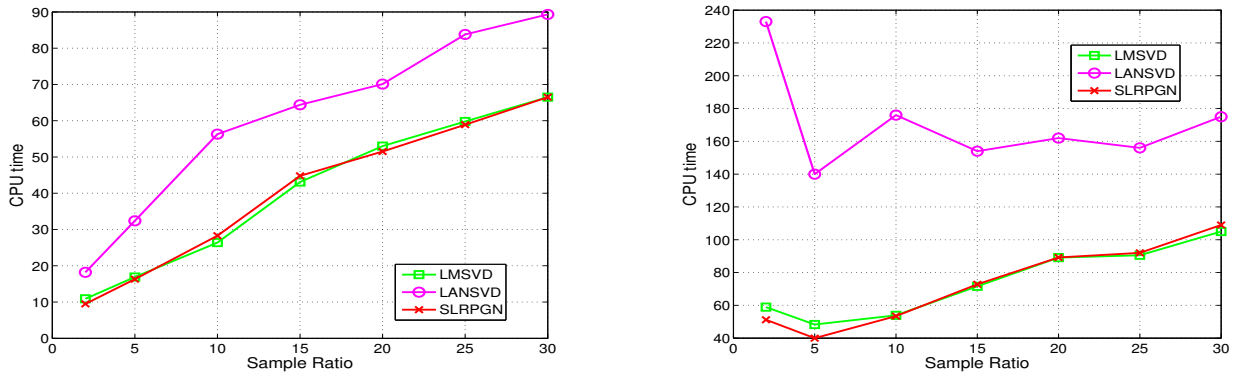
We conclude this subsection with the following remarks:

- The advantage of SLRPGN over LANSVD is obvious on either dense or sparse problems. On sparse problems,



(a) Exact rank $r = 10$ (b) Exact rank $r = 50$

FIG. 5.6. NNLS: *Comparison with varying sampling ratios by using SpMM-Matlab*



(a) Exact rank $r = 10$ (b) Exact rank $r = 50$

FIG. 5.7. NNLS: *Comparison with varying sampling ratios by using SpMM-MKL*

however, the magnitude of the advantage can vary with the efficiency of the code implementing sparse-dense matrix-matrix multiplications on multicore CPUs.

- The main computational cost of LMSVD lies in computing dense eigenvalue decompositions in subspace optimization and orthogonalization for stabilization. The cost of these operations is relatively low, compared with that of SpMM, if the number of singular pairs to be computed is relatively small. However, as the number of computed singular pairs grows these operations can become a bottleneck, making the advantage of SLRPGN more pronounced.
- Compared to the results with the SVT algorithm, the advantage of SLRPGN over LANSVD, although unmistakably present, appears less pronounced with the NNLS algorithm. This difference can be attributed to a strategy used by NNLS that gradually increases the number of computed singular pairs from one to a larger number. Therefore, the number of computed singular pairs only becomes sufficiently large near the end. This situation is not favorable to block algorithms like SLRPGN and LMSVD which derive their advantage over Krylov subspace methods from a high level of concurrency in SpMM when there are a relatively large number of dense columns involved in SpMM. Nevertheless, the two block methods still enjoy the advantage of the warm-start ability.

TABLE 5.7
Results of NNLS *on real examples by using SpMM-MKL*

		LANSVD				SLRPGN				LMSVD			
name	(m, n)	iter	svp	time	mse	iter	svp	time	mse	iter	svp	time	mse
jester-1	(24983, 100)	26	93	1.21e+01	1.64e-01	27	69	8.04e+00	1.76e-01				
jester-2	(23500, 100)	26	93	8.58e+00	1.65e-01	26	79	7.76e+00	1.72e-01				
jester-3	(24938, 100)	24	83	7.06e+00	1.16e-01	27	74	7.20e+00	1.24e-01				
jester-all	(73421, 100)	26	93	2.88e+01	1.58e-01	26	82	2.30e+01	1.62e-01				
moive-100K	(943, 1682)	34	100	8.45e+00	1.28e-01	35	100	2.20e+00	1.26e-01	53	100	5.31e+00	3.67e-01
moive-1M	(6040, 3706)	50	100	4.61e+01	1.42e-01	50	100	1.16e+01	1.43e-01	50	100	2.92e+01	1.42e-01
moive-10M	(71567, 10677)	54	100	5.90e+02	1.26e-01	57	100	1.78e+02	1.27e-01	54	100	2.14e+02	1.26e-01

5.3. Robust Principal Component Analysis. In this subsection, we test the performance of SLRPGN on robust principal component analysis, also called matrix separation. Given a matrix $D \in \mathbb{R}^{m\times n}$, the problem is to compute a low-rank matrix L_0 and a sparse matrix S_0 such that $D = L_0 + S_0$ (or $D \approx L_0 + S_0$). It has been shown in [4] that, under suitable conditions, the separation can be found by solving the convex optimization problem:

$$\min_{L,S\in\mathbb{R}^{m\times n}} \quad \|L\|_* + \mu\|S\|_1 \quad \mathbf{s.t.} \quad L + S = D, \tag{54}$$

where $\mu > 0$ is a proper balancing parameter, $\|L\|_*$ is the nuclear norm of L and $\|S\|_1 = \sum_{ij} |S_{ij}|$. Model (54) can be solved by the so-called inexact augmented Lagrange multiplier algorithms, and the accompanying Matlab code IALM, given in [18] (see also [33]). The main cost is again the computation of the dominant SVD of certain matrices related to the nuclear norm term. Hence, we study how the performance of the IALM solver changes as its default SVD solver LANSVD is replaced by other SVD solvers.

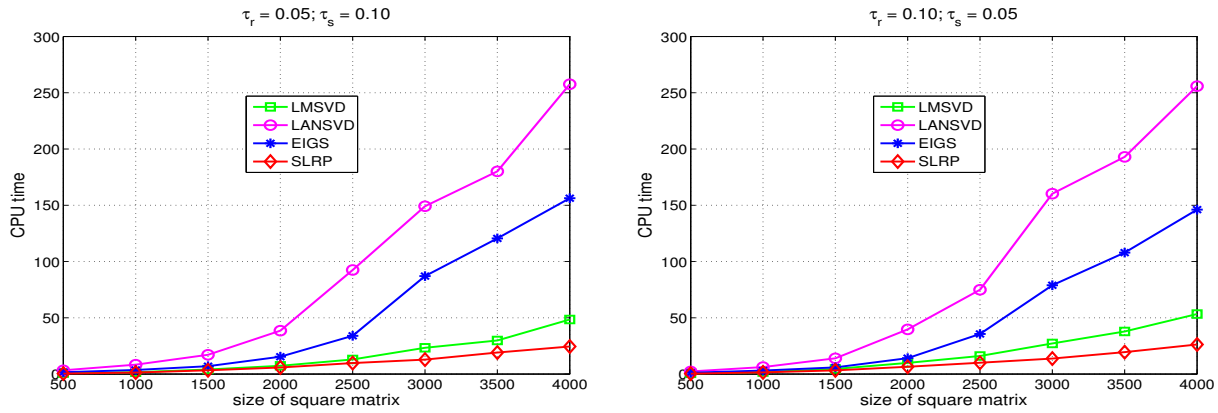
We first generate test matrices of the form $D = L_0 + S_0$, where L_0 is rank-r and S_0 is sparse. Specifically, $L_0 = XY^{\mathrm{T}}$, where X and Y are $n \times r$ random matrices with i.i.d. standard Gaussian entries; $S_0 = \tau\tilde{S}$, where $\tilde{S}$ is a sparse matrix whose nonzero positions are uniformly sampled and nonzero element values are independently chosen from the standard Gaussian distribution with variance $1/n$, and τ is a scalar which makes S_0 roughly the same magnitudes as L_0. The sizes of the test matrices are $m = n = 500, 1000, 1500, ..., 4000$. We test two groups of test problems, characterized by different parameter pairs $(\tau_r, \tau_s) = (0.05, 0.10)$ and $(\tau_r, \tau_s) = (0.10, 0.05)$, respectively, where $\tau_r = r/n$ is the rank ratio and $\tau_s = \mathtt{nnz}(S_0)/n^2$ is the density parameter. In all tests, the balancing parameter μ is fixed as $1/\sqrt{n}$.

We run IALM with four SVD solvers, LMSVD, LANSVD, EIGS and SLRPGN . Since all solvers achieve a very similar accuracy, we only summarize the runtime results in Figure 5.8. We observe that IALM using SLRPGN is about 10 times faster than using LANSVD, while it is about twice as fast as using LMSVD.

We next consider a video separation problem in [4] by applying IALM with different SVD solvers. It aims to separate a video into a static background and one with moving objects. The frames of a video are reshaped into long column vectors and then collected into a matrix. As such, the number of rows in the matrix equals to the number of pixels and the number of columns equals to the number of frames in a video. The information of nine test videos[6] is summarized in Table 5.8, for instance, the example "bootstrap" gives a 19200×3055 matrix separation problem. The average number of dominant SVD computed at each iteration (denoted by "av.sv") and the runtime are presented in Table 5.8. We observe that IALM with SLRPGN requires the least amount of runtime than with all other solvers.

5.4. Comparison on Achieving Higher Accuracies. There exists two main difficulties for SLRPGN to achieve high accuracies: a) it is often asymptotically slow, especially on large-residual problems; and b) a single Rayleigh-Ritz call is often not enough to reach a high accuracy. In this subsection, we will demonstrate that by adapting some

[6]Downloadable from `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`



(a) group 1 (b) group 2

FIG. 5.8. *Runtime results on randomly generated matrix separation problems.*

TABLE 5.8
Numerical results on video separation problems

		LMSVD		LANSVD		EIGS		SLRPGN	
name	(m, n)	time	av.sv	time	av.sv	time	av.sv	time	av.sv
bootstrap	(19200, 3055)	72.96	152.50	321.27	152.50	206.15	152.50	64.51	134.10
campus	(20480, 1439)	34.68	76.50	87.40	76.50	68.59	76.50	28.81	63.89
curtain	(20480, 2964)	76.04	143.67	341.30	143.67	226.21	143.67	64.11	119.55
escalator	(20800, 3417)	91.49	179.62	474.39	179.62	316.06	179.62	78.35	151.38
fountain	(20480, 523)	10.35	25.61	18.87	25.61	14.57	25.61	9.55	24.17
hall	(25344, 3584)	131.73	189.43	599.34	189.43	455.22	189.43	98.07	149.55
lobby	(20480, 1546)	32.11	61.16	129.54	61.16	71.20	61.16	23.75	39.32
shoppingmall	(81920, 1286)	124.83	67.00	283.86	67.00	224.42	67.00	93.19	50.65
watersurface	(20480, 633)	13.27	30.11	27.73	30.11	19.36	30.11	10.99	27.39

existing techniques from numerical linear algebra, it is possible to alleviate these difficulties.

Specifically, to compute the k largest eigenvalues of A, which has been made positive semidefinite, we apply SLRPGN to a suitable polynomial filter of A, say $\rho(A)$, to try to suppress the magnitudes of $\rho(\lambda_j)$ for $j > k$ where λ_j is the j-th largest eigenvalue of A (thus turning the least-squares problem into a small-residual one); and whenever necessary we make multiple Rayleigh-Ritz calls to progressively reach for high accuracies. Each call of the Rayleigh-Ritz procedure orthogonalizes the computed basis at hand, projects A onto the subspace spanned by the basis (whose dimension is greater than k), and computes the corresponding Ritz pairs by solving a small dense eigenvalue problem. The process is restarted, with a warm start, if a prescribed accuracy has not been reached. In addition, a deflation scheme is used to reduce unnecessary computation after some eigenpairs have "converged". We call this algorithm the Restarted SLRPGN or simply RSLRPGN. We terminate RSLRPGN when the maximum relative residual norm, as is defined in (51), is smaller than a given tolerance tol. The algorithm is also stopped if most of the Ritz pairs have relative residual norms much smaller than the tolerance tol and the remaining a few have residual norms not exceeding ten times tol. In our experiments, we set $tol = 10^{-6}$ (hence upon termination the worst eigenvector error is smaller than 10^{-5}). Since the algorithm checks the termination rule only after each Rayleigh-Ritz call, it often returns solutions of higher accuracies than what is prescribed by tol.

The aforementioned techniques used in RSLRPGN have been studied in numerical linear algebra over the years (e.g. [24, 34] on polynomial filters) and are relatively well understood. However, it is not a simple matter to integrate all of them together with the SLRPGN procedure to form an efficient and robust eigensolver. For example, effective use of a polynomial filter involves the choice of polynomials (types and degrees), the estimation of relevant eigenvalues, the updates of these eigenvalue estimates after each outer iteration, and so on. Beyond the simple sketch in the previous

paragraph, any meaningfully detailed description for such an algorithm would surely exceed the amount of space reasonable for this side topic of the paper. Therefore, we refer readers interested in further details to a forthcoming paper (which studies a broader framework than SLRPGN) currently under preparation [32].

We select a set of twelve symmetric sparse matrices from The University of Florida Sparse Matrix Collection[7]. The dimensionality n and the number of nonzeros nnz of these matrices are listed in Table 5.9. Many of these matrices are produced by PARSEC [13], a real space density functional theory (DFT) based code for electronic structure calculation in which the Hamiltonian is discretized by using finite difference. The number of eigenpairs to be computed is roughly 1% of the matrix dimension.

TABLE 5.9
Information of Test Matrices

matrix name	n	nnz	density
Andrews	60000	410077	0.01%
C60	17576	212390	0.07%
cfd1	70656	948118	0.02%
finance	74752	335872	0.01%
Ga10As10H30	113081	3114357	0.02%
Ga3As3H12	61349	3016148	0.08%
OPF3754	15435	82231	0.03%
shallow_water1	81920	204800	<0.01%
Si10H16	17077	446500	0.15%
Si5H12	19896	379247	0.10%
SiO	33401	675528	0.06%
wathen100	30401	251001	0.03%

Since it is impractical to carry out numerical experiments with a large number of solvers, we choose to compare RSLRPGN with the Matlab built-in solver EIGS (i.e., ARPACK) and a Matlab version of the solver LANEIG in the package PROPACK[8]. Two recently developed solvers, the filtered Lanczos algorithm in [7] and the FEAST algorithm in [29], are not included because they are designed to compute all eigenvalues (and eigenvectors) within a given interval. A commonly accepted interface for computing k extreme eigenpairs does not yet exist for either solver, to the best of our knowledge. In addition, the performance of these solvers is strongly affected by the quality of an estimated interval containing k extreme eigenvalues, making it difficult to accurately evaluate their performance if interval estimation is used to compute k extreme eigenpairs. The Chebyshev-Davidson algorithm in [34], another potential candidate, is not included in our experiments due to unavailability of a robust Matlab implementation.

Summaries of computational results on computing the k smallest and largest eigenpairs for the twelve test matrices are presented in Tables 5.10 and 5.11, respectively, where "maxres" denotes the maximum relative residual norm (51) at the final solutions, and "time" is the runtime measured in seconds. We observe that at this moderately high accuracy, RSLRPGN is still able to maintain a clear speed advantage over the other two solvers, and often returns smaller residual errors than $tol = 10^{-6}$.

5.5. Further Discussions. Theorem 4.2 does not guarantee that Algorithm 2 converges from any random starting points, but our numerical results strongly suggest that convergence occur at least with overwhelming probability. In addition, so far we have not noticed convergence to any non-optimal stationary point. Evidently, this phenomenon has to do with the fact that beside global minimizers all other full-rank stationary points of the function $f(X)$ are saddle points (see Proposition 2.4). These saddle points appear to have an extremely low probability to attract SLRPGN iterates (that contain components from all the eigenspaces when started from a random point, even though some components are gradually fading away as the iterates converge). How to rigorously verify and quantify such a behavior remains a subject of further study.

[7] Downloadable from `http://www.cise.ufl.edu/research/sparse/matrices`
[8] Downloadable from `http://soi.stanford.edu/~rmunk/PROPACK`.

TABLE 5.10
Comparison results on computing k smallest eigenpairs

Matrix Information		EIGS		LANEIG		RSLRPGN	
name	k	maxres	time	maxres	time	maxres	time
Andrews	600	4.90e-07	447.08	1.20e-05	552.48	1.74e-08	174.38
C60	200	7.86e-12	12.45	7.23e-06	20.70	1.14e-07	13.13
cfd1	700	4.45e-09	4354.05	1.93e-07	24532.30	1.31e-06	925.57
finance	700	7.44e-10	1644.88	6.01e-07	5849.13	2.52e-06	424.52
Ga10As10H30	1000	4.57e-12	3027.94	4.67e-07	15224.21	5.57e-06	4609.16
Ga3As3H12	600	1.85e-08	572.32	5.86e-07	2394.25	8.33e-06	1016.70
OPF3754	200	1.18e-14	5.07	9.14e-06	3.79	9.61e-13	13.66
hallow_water1s	800	2.67e-07	1626.18	2.12e-07	7374.38	2.18e-07	356.05
Si10H16	200	2.17e-12	18.85	2.46e-07	35.15	6.98e-09	18.17
Si5H12	200	4.51e-11	22.69	3.20e-06	42.00	2.45e-10	22.47
SiO	400	2.20e-10	121.24	5.62e-07	268.98	6.85e-09	62.12
wathen100	300	4.68e-09	144.99	1.78e-06	494.74	2.09e-06	92.02
Geometric Means		2.89e-10	200.74	1.13e-06	512.51	5.84e-08	138.04

TABLE 5.11
Comparison results on computing k largest eigenpairs

Matrix Information		EIGS		LANEIG		RSLRPGN	
name	k	maxres	time	maxres	time	maxres	time
Andrews	600	1.01e-07	224.77	3.17e-06	218.53	1.84e-08	104.60
C60	200	2.88e-07	14.18	1.68e-07	23.66	2.32e-06	9.86
cfd1	700	4.72e-14	295.23	9.32e-06	239.47	2.11e-07	110.71
finance	700	1.92e-14	282.55	2.13e-13	165.40	5.32e-12	69.68
Ga10As10H30	1000	4.15e-14	1531.19	6.67e-07	5050.30	3.15e-06	361.34
Ga3As3H12	600	1.37e-08	343.61	1.00e-06	848.46	1.92e-06	122.38
OPF3754	200	9.94e-07	5.11	1.42e-05	5.33	6.40e-09	11.44
shallow_water1s	800	4.39e-09	794.54	2.10e-06	1524.05	7.63e-07	393.44
Si10H16	200	2.76e-10	13.78	6.75e-07	16.75	3.78e-06	11.35
Si5H12	200	1.98e-11	15.65	2.40e-06	27.07	2.52e-06	11.80
SiO	400	3.00e-10	78.62	9.47e-07	103.04	2.67e-06	37.99
wathen100	300	9.61e-07	39.42	1.93e-06	41.63	1.35e-10	27.16
Geometric Means		4.65e-10	92.25	4.46e-07	124.50	1.14e-07	49.55

As is indicated by Theorem 4.3 and the numerical results in Figure 5.2, SLRPGN could become asymptotically slow on large residual problems when higher accuracies are is required. This is the main reason why we only require a moderate accuracy in our numerical experiments. Numerical results in Section 5.4 indicate that SLRPGN can be combined with polynomial filtering and multiple Rayleigh-Ritz (RR) calls to achieve higher accuracies on more general eigenvalue problems. This topic is currently under further investigations.

6. Concluding Remarks. The purpose of this paper is to study a Gauss-Newton method for computing dominant eigenspaces which forms the backbone of principal eigenvalue or singular-value decompositions. The proposed algorithm is a block algorithm that updates a basis matrix X using AX plus some additional dense matrix operations, as opposed to updating a basis piece by piece via a sequential process as in Krylov subspace methods. Moreover, the proposed algorithm does not require basis orthogonalization at each iteration or even periodically if high-accuracy solutions are not required. We list four main motivating factors behind the study of this Gauss-Newton algorithm.

(a) Block algorithms like SLRPGN can easily take the advantage of warm-starting in iterative settings, while Krylov subspace methods have difficulty to do so.

(b) Many low-rank approximation or principal component analysis problems are, by their very nature, small-residual problems for which the proposed Gauss-Newton methodology is known to be effective.

(c) The Gauss-Newton algorithm derived for the SLRP model has a low iteration-complexity. In particular, the unit-step version is parameter-free (though it may be sub-optimal for large residual problems).

We have analyzed convergence properties of the proposed Gauss-Newton method, including global convergence with correction and step-size control, convergence regions of the unit-step algorithm without correction, and the asymptotic rate of convergence, as are given in Theorems 4.1, 4.2 and 4.3. In particular, we recall that a convergence region does not exist for the unit-step Gauss-Newton method when applied to general nonlinear least squares

problems unless the residual is sufficiently small.

We have conducted rather extensive numerical experiments on both randomly generated matrices and matrices from recent applications in low-rank matrix completion and robust principal component analysis (PCA). Our numerical results show that, when a low or moderate accuracy is sufficient for large residual problems, our new Gauss-Newton algorithm either outperforms, or is competitive with, a number of existing algorithms. Most notably, in iterative settings such as solving matrix completion or robust PCA problems where (i) residuals tend to be small, (ii) warm-starts are used and (iii) high-accuracy solutions are unnecessary, the Gauss-Newton algorithm can provide multi-fold speedups over some state-of-the-art Krylov subspace algorithms which remain methods of choice at the present time.

So far in this paper, we have treated the Gauss-Newton algorithm almost entirely in an optimization framework. In particular, when highly accurate solutions are not necessary, as is the case in solving various application problems, we only make a single call to the Rayleigh-Ritz procedure to calculate principal eigenpairs after a basis matrix is returned by the Gauss-Newton algorithm. Such a "pure" optimization algorithm should not be directly taken as a general-purpose eigensolver for computing highly accurate eigenpairs of difficult matrices. As is suggested by the results in Section 5.4, however, it does have a potential to become a building component for a general-purpose eigensolver in combination with suitable techniques from numerical linear algebra.

Acknowledgements. The authors would like to thank Prof. William Hager and two anonymous referees for their detailed and valuable comments and suggestions.

REFERENCES

[1] JONATHAN BARZILAI AND JONATHAN M. BORWEIN, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[2] AMIR BECK AND MARC TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[3] MATTHIAS BOLLHÖFER AND YVAN NOTAY, *JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices*, Comput. Phys. Comm., 177 (2007), pp. 951–964.

[4] EMMANUEL J. CANDÈS, XIAODONG LI, YI MA, AND JOHN WRIGHT, *Robust principal component analysis?*, Journal of the ACM, 58 (2011), pp. 1– 37.

[5] EMMANUEL J. CANDÈS AND BENJAMIN RECHT, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, (2009).

[6] EMMANUEL J. CANDÈS AND TERENCE TAO, *The power of convex relaxation: near-optimal matrix completion*, IEEE Trans. Inform. Theory, 56 (2010), pp. 2053–2080.

[7] HAW-REN FANG AND YOUSEF SAAD, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comput., 34 (2012), pp. A2220–A2246.

[8] R. FLETCHER, *Practical methods of optimization*, A Wiley-Interscience Publication, John Wiley & Sons Ltd., Chichester, second ed., 1987.

[9] N. HALKO, P. G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[10] H. O. HARTLEY, *The modified guass-newton method for the fitting of nonlinear regression functions by least squares*, Technometrics, 3 (1961), pp. 269–280.

[11] CAI JIAN-FENG, EMMANUEL J. CANDES, AND SHEN ZUOWEI, *A singular value thresholding algorithm for matrix completion export*, SIAM J. Optim., 20 (2010), pp. 1956–1982.

[12] ANDREW V. KNYAZEV, *Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[13] L. KRONIK, A. MAKMAL, M. TIAGO, M. M. G. ALEMANY, X. HUANG, Y. SAAD, AND J. R. CHELIKOWSKY, *PARSEC – the pseudopotential algorithm for real-space electronic structure calculations: recent advances and novel applications to nanostructures*, Phys. Stat. Solidi. (b), 243 (2006), pp. 1063–1079.

[14] R. M. LARSEN, *Lanczos bidiagonalization with partial reorthogonalization*, Aarhus University, Technical report, DAIMI PB-357, September 1998.

[15] R. B. LEHOUCQ, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.

[16] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK users' guide: Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6 of Software, Environments, and Tools, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[17] KENNETH LEVENBERG, *A method for the solution of certain non-linear problems in least squares*, Quart. Appl. Math., 2 (1944), pp. 164–168.

[18] Z. LIN, M. CHEN, L. WU, AND Y. MA, *The augmented lagrange multiplier method for exact recovery of a corrupted low-rank matrices*. UIUC Technical Report UILU-ENG-09-2215, 2009.

[19] X. LIU, Z. WEN, AND Y. ZHANG, *Limited memory block krylov subspace optimization for computing dominant singular value decompositions*, SIAM Journal on Scientific Computing, 35-3 (2013), pp. A1641–A1668.

[20] DONALD W. MARQUARDT, *An algorithm for least-squares estimation of nonlinear parameters*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 431–441.

[21] BENJAMIN RECHT, MARYAM FAZEL, AND PABLO A. PARRILO, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.

[22] HEINZ RUTISHAUSER, *Computational aspects of F. L. Bauer's simultaneous iteration method*, Numer. Math., 13 (1969), pp. 4–13.

[23] H. RUTISHAUSER, *Simultaneous iteration method for symmetric matrices*, Numer. Math., 16 (1970), pp. 205–223.

[24] YOUCEF SAAD, *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems*, Mathematics of Computation, 42 (1984), pp. 567–588.

[25] DANNY C. SORENSEN, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, in Parallel numerical algorithms (Hampton, VA, 1994), vol. 4 of ICASE/LaRC Interdiscip. Ser. Sci. Eng., Kluwer Acad. Publ., 1996, pp. 119–165.

[26] A. STATHOPOULOS AND C. F. FISCHER, *A davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix*, Computer Physics Communications, 79 (1994), pp. 268–290.

[27] G. W. STEWART, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numer. Math., 25 (1975/76), pp. 123–136.

[28] WILLIAM J. STEWART AND ALAN JENNINGS, *A simultaneous iteration algorithm for real matrices*, ACM Trans. Math. Software, 7 (1981), pp. 184–198.

[29] PING TAK PETER TANG AND ERIC POLIZZI, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*. arXiv:1302.0432, 2014.

[30] KIM-CHUAN TOH AND SANGWOON YUN, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optimization, 6 (2010), pp. 615–640.

[31] ZAIWEN WEN, CHAO YANG, XIN LIU, AND YIN ZHANG, *Trace-penalty minimization for large-scale eigenspace computation*, Journal of Scientific Computing, (2015).

[32] Z. WEN AND Y. ZHANG, *Block algorithms with augmented rayleigh-ritz projections for large-scale eigenpair computation*, Tech. Report TR15-01, CAAM, Rice University, 2015.

[33] X. YUAN AND J. YANG, *Sparse and low-rank matrix decomposition via alternating direction methods*, Pacific Journal of Optimization, 9 (2013), pp. 167–180.

[34] Y. ZHOU AND Y. SAAD, *A Chebyshev–Davidson algorithm for large symmetric eigenproblems*, SIAM J. Matrix Anal. and Appl., 29 (2007), pp. 954–971.