Low-rank matrix completion by Riemannian optimization

Bart Vandereycken*

August 24, 2011

Abstract

The matrix completion problem consists of finding or approximating a low-rank matrix based on a few samples of this matrix. We propose a novel algorithm for matrix completion that minimizes the least square distance on the sampling set over the Riemannian manifold of fixed-rank matrices. The algorithm is an adaptation of classical non-linear conjugate gradients, developed within the framework of retraction-based optimization on manifolds. We describe all the necessary objects from differential geometry necessary to perform optimization over this low-rank matrix manifold, seen as a submanifold embedded in the space of matrices. In particular, we describe how metric projection can be used as retraction and how vector transport lets us obtain the conjugate search directions. Additionally, we derive second-order models that can be used in Newton's method based on approximating the exponential map on this manifold to second order. Finally, we prove convergence of a regularized version of our algorithm under the assumption that the restricted isometry property holds for incoherent matrices throughout the iterations. The numerical experiments indicate that our approach scales very well for large-scale problems and compares favorable with the state-of-the-art, while outperforming most existing solvers.

1 Introduction

Let $A \in \mathbb{R}^{m\times n}$ be an $m \times n$ matrix that is only known on a subset Ω of the complete set of entries $\{1,\dots,m\} \times \{1,\dots,n\}$. The low-rank matrix completion problem [CR09] consists of finding the matrix with lowest rank that agrees with M on Ω:

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \operatorname{rank}(X), \\ \text{subject to} & X \in \mathbb{R}^{m\times n},\ \mathrm{P}_\Omega(X) = \mathrm{P}_\Omega(A), \end{array} \quad . \tag{1}$$

where

$$\mathrm{P}_\Omega \colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n} \colon X_{i,j} \mapsto \begin{cases} X_{i,j}, & \text{if } (i,j), \in \Omega, \\ 0, & \text{if } (i,j) \notin \Omega, \end{cases} \tag{2}$$

denotes the orthogonal projection onto Ω. Without loss of generality, we can assume $m \leq n$.

Due to the presence of noise, it is advisable to relax the equality constraint in (1) to allow for some misfit. Then, given a tolerance $\epsilon \geq 0$, a more robust version of the rank minimization problem

*Chair of Numerical Algorithms and HPC, MATHICSE, École Polytechnique Fédérale de Lausanne, Station 8, CH-1015 Lausanne, Switzerland (bart.vandereycken@epfl.ch)

becomes

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \operatorname{rank}(X), \\ \text{subject to} & X \in \mathbb{R}^{m\times n},\ \|\operatorname{P}_\Omega(X) - \operatorname{P}_\Omega(A)\|_{\mathrm{F}} < \epsilon, \end{array} . \tag{3}$$

where $\|X\|_{\mathrm{F}}$ denotes the Frobenius norm of X.

Matrix completion has a number of interesting applications such as collaborative filtering, system identification, and global positioning, but unfortunately it is NP-hard in general. Recently, there has been a considerably body of work devoted to the identification of large classes of matrices for which (1) has a unique solution that can be recovered in polynomial time. In [CT09], for example, the authors show that when Ω is sampled uniformly at random, the nuclear norm relaxation

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \|X\|_*, \\ \text{subject to} & X \in \mathbb{R}^{m\times n},\ \operatorname{P}_\Omega(X) = \operatorname{P}_\Omega(A), \end{array} \tag{4}$$

can recover any matrix M of rank k that has so-called low incoherence, provided that the number of samples is large enough, $|\Omega| > Cnk \operatorname{polylog}(n)$. Related work has been done by [CR09], [KMO10b], [CP10], [KMO10a], which, in particular, also establishes similar recovery results for the robust formulation (3).

Many of the potential applications for matrix completion involve very large data sets; the Netflix matrix, for example, has more than 10^8 entries [ACM07]. It is therefore crucial to develop algorithms than can cope with such a large-scale setting, but, unfortunately, solving (4) by off-the-shelf methods for convex optimization scales very badly in the matrix dimension. This has spurred a considerable amount of algorithms that aim to solve the nuclear norm relaxation by specifically designed methods that try to exploit the low-rank structure of the solution; see, e.g., [CCS10], [MGC11], [MJD10], [LCWM09], [GM11], [TY10], [LST10], [CO10]. Other approaches include optimization on the Grassmann manifold [KMO10b], [BNR10], [DM10]; atomic decompositions [LB10], and non-linear SOR [WYZ10].

1.1 The proposed method: optimization on manifolds

We present a new method for low-rank matrix completion based on a direct optimization over the set of all fixed-rank matrices. By prescribing the rank of the global minimizer of (3), say k, the robust matrix completion problem is equivalent to

$$\begin{array}{ll} \underset{X}{\text{minimize}} & f(X) := \frac{1}{2}\|\operatorname{P}_\Omega(X - A)\|_{\mathrm{F}}^2, \\ \text{subject to} & X \in \mathcal{M}_k := \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = k\}. \end{array} \tag{5}$$

It is well known that $\mathcal{M}_k$ is a smooth (C^∞) manifold; it can, for example, be identified as the smooth part of the determinantal variety of matrices of rank at most k [BV88, Proposition 1.1]. Since the objective function is smooth also, problem (5) is a smooth, Riemannian optimization problem, which turns out to be significantly easier to solve than (3) by methods from Riemannian optimization, as introduced, amongst others, by [EAS99], [ADM$^+$02], [AMS08]. Simply put, Riemannian optimization is the generalization of standard unconstrained optimization, where the search space is $\mathbb{R}^n$, to optimization of a smooth objective function on a Riemannian manifold.

For solving the optimization problem (5), in principle any method from optimization on Riemannian manifolds could be used. In this paper, we construct an algorithm based on the framework of retraction-based optimization in [AMS08]. The numerical algorithm heavily exploits the smoothness of $\mathcal{M}_k$ and is the generalization of classical non-linear conjugate gradients (CG) on Euclidean

space to perform optimization on manifolds; see, e.g., [Smi94], [EAS99], [AMS08]. The skeleton of the proposed method is listed in Algorithm 1. Although such a method can be used to solve any problem of the form

$$\begin{aligned} &\text{minimize} && f(X), \\ &\text{subject to} && X \in \mathcal{M}, \end{aligned} \tag{6}$$

where $\mathcal{M}$ is a smooth manifold and $f : \mathcal{M}_k \to \mathbb{R}$ a smooth objective function, this paper focuses on using Algorithm 1 for solving (5), the low-rank matrix completion problem.

Algorithm 1 is derived using concepts from differential geometry, yet it closely resembles a typical non-linear CG algorithm with Armijo line-search for unconstrained optimization. It is schematically visualized in Figure 1 for iteration number i and relies on the following crucial ingredients which will be explained in more detail for (5) in Section 2.

1. The *Riemannian gradient*, denoted $\operatorname{grad} f(X_i)$, is a specific tangent vector ξ_i which corresponds to the direction of steepest ascent of $f(X_i)$, but restricted to only directions in the tangent space $T_{X_i}\mathcal{M}_k$.

2. The search direction $\eta_i \in T_{X_i}\mathcal{M}_k$ is conjugate to the gradient and is computed by a variant of the classical Polak–Ribière updating rule in non-linear CG. This requires taking a linear combination of the Riemannian gradient with the previous search direction η_{i-1}. Since η_{i-1} does not lie in $T_{X_i}\mathcal{M}_k$, it needs to be transported to $T_{X_i}\mathcal{M}_k$. This is done by a mapping $\mathcal{T}_{X_{i-1}\to X_i} : T_{X_{i-1}}\mathcal{M}_k \to T_{X_i}\mathcal{M}_k$, the so-called *vector transport*.

3. As a tangent vector only gives a direction but not the line search itself on the manifold, a smooth mapping $R_{X_i} : T_{X_i}\mathcal{M}_k \to \mathcal{M}_k$, called the *retraction*, is needed to map tangent vectors to the manifold. Using the conjugate direction η_i, a line-search for t can then be performed along the curve $t \mapsto R_{X_i}(t\,\eta_i)$. Step 6 uses a standard backtracking procedure where we have chosen to fix the constants. More judiciously chosen constants can sometimes improve the line search, but our numerical experiments indicate that this is not necessary in the setting under consideration.

1.2 Relation to existing work

Manifold identification An obvious shortcoming of solving (5) instead of (3) is that the rank k has to be specified. For some classes of problems, such as Euclidean distance matrix completion [AKW99] or rank aggregation [GL11], the optimization problem (5) is directly formulated over a set of matrices for which the rank is known a priori. Some approximation problems, where the matrix to compute is actually of full rank with decaying singular values, tend to be rather insensitive to the rank chosen and a crude guess is likely to work well. However, for the typical completion problems in machine learning, like recommender systems, there are only a few good choices for the rank. Hence, in order to apply (5) on these kind of problems, we need to determine this rank.

Fortunately, there are many existing algorithms for low-rank matrix completion that do not need to know the rank, most of them based on solving the nuclear norm relaxation (4). For some of the algorithms, like SVT of [CCS10], the rank of the iterates is even observed to converge monotonically. One can therefore think about using such a method to detect the rank and then switch over to solving (5). Such an approach can be interpreted as manifold identification, which is advocated in [HL07], [Har09], [LW11] as a promising strategy for the optimization of partial

Algorithm 1 Geometric CG for (6)

Require: smooth objective function $f : \mathcal{M}_k \to \mathbb{R}$, initial iterate $X_1 \in \mathcal{M}_k$, tolerance $\tau > 0$, tangent vector $\eta_0 = 0$

1: **for** $i = 1, 2, \ldots$ **do**
2: Compute the gradient:
$\xi_i := \operatorname{grad} f(X_i)$ {see Algorithm 2}
3: Check convergence:
if $\|\xi_i\| \leq \tau$, break
4: Compute a conjugate direction by PR+:
$\eta_i := -\xi_i + \beta_i \mathcal{T}_{X_{i-1} \to X_i}(\eta_{i-1})$ {see Algorithm 4}
5: Determine an initial step t_i from a linearized problem:
$t_i = \arg\min_t f(X_i + t\,\eta_i)$ {see Algorithm 5}
6: Perform Armijo backtracking to find the smallest integer $m \geq 0$ such that
$f(X_i) - f(R_{X_i}(0.5^m\, t_i\, \eta_i)) \geq -0.0001\, \langle\, \eta_i, 0.5^m\, t_i\, \eta_i \,\rangle$
and obtain the new iterate:
$X_{i+1} := R_{X_i}(0.5^m\, t_i\, \eta_i)$ {see Algorithm 6}
7: **end for**

smooth functions. The rationale for switching between two methods for these class of problems (as well as in our case) is that the active constraints around the optimizer usually form a smooth manifold, while the original problem, considering all the constraints, is non-smooth. A slower, first-order method for non-smooth optimization is used to identify the active manifold. After that, a Newton-type method can exploit second-order information for the smooth subproblems to speed up the convergence considerably.

In our setting, we advocate taking $\mathcal{M}_k$, the manifold of fixed-rank matrices, as the active manifold since all non-smoothness is eliminated by simply fixing the rank. Compared to the work of [HL07], [Har09], [LW11], manifold identification for (5) becomes rather straightforward: Once the rank is known, one does not need to form a set of equations that define the manifold since the differential geometric structure of this manifold is available explicitly. Furthermore, a Newton-type optimization routine that exploits second-order information can be easily derived from this description, as we demonstrate by constructing second-order models for (5).

The possibility of using second-order information is likely to be significant for many type of rank-constrained optimization problems. In [JBAS10] and [VV10], for example, the authors use a similar geometric description for the set of symmetric matrices where a Newton-type method did effectively speed up convergence compared to any first-order method. In the current setting, however, Newton's method could not compete with non-linear CG, although the effectiveness of our non-linear CG acceleration scheme can most likely be attributed to the smoothness of the active manifold as well.

Debiasing Exploiting smoothness is one advantage of solving (3) instead of (5). Another major advantage is the possibility for debiasing the solution after minimization of a relaxed version, such as (4). In the context of compressed sensing and matrix completion, it is well-known that such a debiasing can significantly enhance the accuracy of the solution. For compressed sensing, debiasing

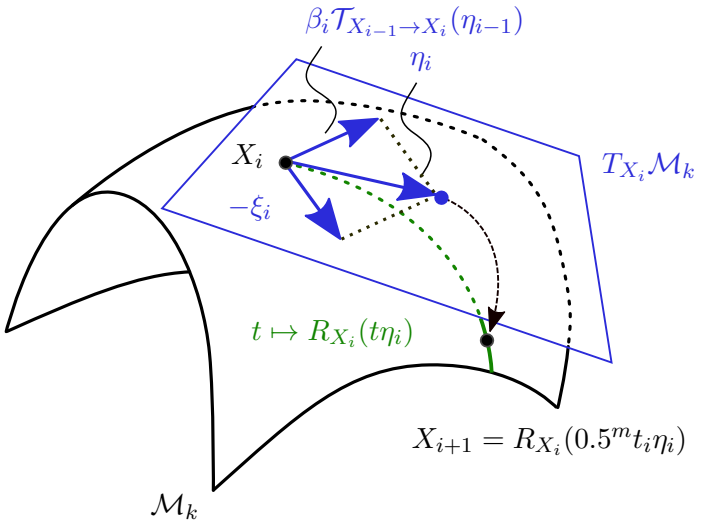

Figure 1: Visualization of Algorithm 1: non-linear CG on a Riemannian manifold.

is the procedure of solving the linear least-square problem

$$\min_{x_I} \|A_I x_I - b\|_2, \tag{7}$$

restricted to a putative set of active constraints, i.e., a discrete set of row indices I. Given that I indeed forms the correct active set of constraints, solving (7) has the advantage that the original least-square error is minimized and not some relaxed problem.

In [MGC11, Section 4.3], the authors advocate debiasing for matrix completion as well. However, the approach as done in compressed sensing is not directly applicable to matrix completion since the rank function is not easily parameterized as a set of active constraints. In order to become a linear least-square problem, [MGC11] makes the following choice: Given the SVD of an iterate $X_i = U_i \Sigma_i V^T$, fix the subspaces U_i and V_i and optimize the singular values:

$$\min_{\Sigma} \| \mathrm{P}_\Omega(U_i \Sigma V_i^T - A)\|_{\mathrm{F}}. \tag{8}$$

Obviously, this is a linear least-squares problem (the matrix Σ can be taken diagonal or full, this does not change the point much). The authors in [MJD10] also perform such a procedure in their SVP algorithm and call it SVP-Newton since solving the linear least-squares problem (8) explicitly is identical to performing one Newton step for (8).

Contrary to the previous approaches that fix the left and right singular subspaces of an iterate, our approach is essentially debiasing over the whole set of rank-k matrices, i.e., the manifold $\mathcal{M}_k$. Since this is a smooth, embedded manifold, we will show that such a debiasing can be performed efficiently despite that $\mathcal{M}_k$ is not a vector space.

1.3 Outline of the paper

The plan of the paper is as follows. In the next section, the necessary concepts of differential geometry are explained to turn Algorithm 1 into a concrete method. The implementation of each step is explained in Section 3. We also prove convergence of this method in Section 4 under some

assumptions which are reasonable for matrix completion. Numerical experiments and comparisons to the state-of-the art are carried out in Section 5. The last section is devoted to conclusions.

2 Differential geometry for low-rank matrix manifolds

In this section, we explain the differential geometry concepts used in Algorithm 1 applied to our particular matrix completion problem (5).

2.1 The Riemannian manifold $\mathcal{M}_k$

Let

$$\mathcal{M}_k = \{X \in \mathbb{R}^{m\times n} \colon \operatorname{rank}(X) = k\}$$

denote the manifold of fixed-rank matrices. Using the SVD, one has the equivalent characterization

$$\mathcal{M}_k = \{U\Sigma V^T \colon U \in \operatorname{St}_k^m,\, V \in \operatorname{St}_k^n,\, \Sigma = \operatorname{diag}(\sigma_i), \sigma_1 \geq \cdots \geq \sigma_k > 0\}, \tag{9}$$

where St_k^m is the Stiefel manifold of $m \times k$ real, orthonormal matrices, and $\operatorname{diag}(\sigma_i)$ denotes a diagonal matrix with σ_i on the diagonal. Whenever we use the notation $U\Sigma V^T$ in the rest of the paper, we always mean matrices that satisfy (9), unless explicitly noted otherwise. Furthermore, the constants k, m, n are always used to denote dimensions of matrices and, for simplicity, we assume $1 \leq k < m \leq n$.

The following proposition shows that $\mathcal{M}_k$ is indeed a smooth manifold. While the existence of such a smooth manifold structure, together with its tangent space, is well known (see, e.g., [HM94] and [KL07] for applications of gradient flows on $\mathcal{M}_k$), more advanced concepts like retraction-based optimization on $\mathcal{M}_k$ have only very recently been investigated; see [SWC10]. In contrast, the case of optimization on *symmetric* fixed-rank matrices has been studied in more detail in [OHM06], [JBAS10], [VV10], and [MBS11].

Proposition 2.1 *The set $\mathcal{M}_k$ is a smooth submanifold of dimension $(m+n-k)k$ embedded in $\mathbb{R}^{m\times n}$. Its tangent space $T_X\mathcal{M}_k$ at $X = U\Sigma V^T \in \mathcal{M}_k$ is given by*

$$T_X\mathcal{M}_k = \left\{ \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} \mathbb{R}^{k\times k} & \mathbb{R}^{k\times(n-k)} \\ \mathbb{R}^{(m-k)\times k} & 0_{(m-k)\times(n-k)} \end{bmatrix} \begin{bmatrix} V & V_\perp \end{bmatrix}^T \right\} \tag{10}$$

$$= \{UMV^T + U_pV^T + UV_p^T \colon M \in \mathbb{R}^{k\times k}, \tag{11}$$
$$U_p \in \mathbb{R}^{m\times k}, U_p^TU = 0, V_p \in \mathbb{R}^{n\times k}, V_p^TV = 0\}.$$

Proof. See [Lee03, Example 8.14] for a proof that uses only elementary differential geometry based on local submersions. The tangent space is obtained by the first-order perturbation of the SVD and counting dimensions. □

The *tangent bundle* is defined as the disjoint union of all tangent spaces,

$$T\mathcal{M}_k := \bigcup_{X\in\mathcal{M}_k} \{X\} \times T_X\mathcal{M}_k = \{(X,\xi) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \colon X \in \mathcal{M}_k, \xi \in T_X\mathcal{M}_k\}.$$

By restricting the Euclidean inner product on $\mathbb{R}^{m\times n}$,

$$\langle A, B\rangle = \operatorname{tr}(A^TB), \quad A, B \in \mathbb{R}^{m\times n}$$

to the tangent bundle, we turn $\mathcal{M}_k$ into a Riemannian manifold with Riemannian metric

$$g_X(\xi,\eta) := \langle \xi,\, \eta \rangle = \operatorname{tr}(\xi^T \eta), \quad X \in \mathcal{M}_k,\, \xi, \eta \in T_X\mathcal{M}_k, \tag{12}$$

where the tangent vectors ξ, η are seen as matrices in $\mathbb{R}^{m\times n}$.

Once the metric is fixed, the notion of the gradient of an objective function can be introduced. For a Riemannian manifold, the *Riemannian gradient* of a smooth function $f : \mathcal{M}_k \to \mathbb{R}$ at $X \in \mathcal{M}_k$ is defined as the unique tangent vector $\operatorname{grad} f(X)$ in $T_X\mathcal{M}_k$ such that

$$\langle \operatorname{grad} f(X), \xi \rangle = \mathrm{D}\, f(X)[\xi], \quad \text{for all } \xi \in T_X\mathcal{M}_k,$$

where we denoted directional derivatives by $\mathrm{D}\, f$. Since $\mathcal{M}_k$ is embedded in $\mathbb{R}^{m\times n}$, the Riemannian gradient is given as the orthogonal projection onto the tangent space of the gradient of f seen as a function on $\mathbb{R}^{m\times n}$; see, e.g., [AMS08, (3.37)]. Defining $\mathrm{P}_U := UU^T$ and $\mathrm{P}_U^\perp := I - \mathrm{P}_U$ for any $U \in \mathrm{St}_k^m$, we denote the orthogonal projection onto the tangent space at X as

$$\mathrm{P}_{T_X\mathcal{M}_k} : \mathbb{R}^{m\times n} \to T_X\mathcal{M}_k,\, Z \mapsto \mathrm{P}_U\, Z\, \mathrm{P}_V + \mathrm{P}_U^\perp\, Z\, \mathrm{P}_V + \mathrm{P}_U\, Z\, \mathrm{P}_V^\perp\,. \tag{13}$$

Then, using $\mathrm{P}_\Omega(X - A)$ as the (Euclidean) gradient of $f(X) = \| \mathrm{P}_\Omega(X - A)\|_{\mathrm{F}}^2/2$, we obtain

$$\operatorname{grad} f(X) := \mathrm{P}_{T_X\mathcal{M}_k}(\mathrm{P}_\Omega(X - A)). \tag{14}$$

2.2 Metric projection as retraction

As explained in Section 1.1, we need a so-called retraction mapping to go back from an element in the tangent space to the manifold. In order to prove convergence within the framework of [AMS08], this mapping has to satisfy certain properties such that it becomes a first-order approximation of the exponential mapping on $\mathcal{M}_k$.

Definition 2.2 (Retraction [AM10, Definition 1]) *A mapping $R\colon T\mathcal{M}_k \to \mathcal{M}_k$ is said to be a* retraction on $\mathcal{M}_k$ *if, for every $\overline{X} \in \mathcal{M}_k$, there exists a neighborhood $\mathcal{U}$ around $(\overline{X}, 0) \subset T\mathcal{M}_k$ such that:*

(1) $\mathcal{U} \subseteq \operatorname{dom}(R)$ and $R|_{\mathcal{U}}\colon \mathcal{U} \to \mathcal{M}_k$ is smooth;

(2) $R(X, 0) = X$ for all $(X, 0) \in \mathcal{U}$;

(3) $\mathrm{D}\, R(X, 0)[0, \xi] = \xi$ for all $(X, \xi) \in \mathcal{U}$.

We also introduce the following shorthand notation:

$$R_X : T_X\mathcal{M}_k \to \mathcal{M}_k\colon \xi \mapsto R(X, \xi).$$

Property (3) is called *local rigidity*. It is equivalent to

(3') $\mathrm{D}\, R_X(0)[\xi] = \xi$ for all $X \in \mathcal{M}_k$ and $\xi \in T_X\mathcal{M}_k$.

In our setting, we have chosen metric projection as retraction:

$$R_X : \mathcal{U}_X \to \mathcal{M}_k, \quad \xi \mapsto \mathrm{P}_{\mathcal{M}_k}(X+\xi) := \operatorname*{arg\,min}_{Z \in \mathcal{M}_k} \|X + \xi - Z\|_{\mathrm{F}}, \tag{15}$$

where $\mathcal{U}_X \subset T_X\mathcal{M}_k$ is a suitable neighborhood around the zero vector, and $\mathrm{P}_{\mathcal{M}_k}$ is the orthogonal projection onto $\mathcal{M}_k$. Based on [LM08, Lemma 2.1], it can be easily shown that this map satisfies the conditions in Definition 2.2.

In order not to overload notation, the notation R_X is used here (and in the following) for any retraction satisfying Definition 2.2 and for (15) simultaneously. It will be clear from context which retraction is meant.

The metric projection can be computed in closed-form by the SVD: Let $X = U\Sigma V^T$ be given, then for sufficiently small $\xi \in \mathcal{U}_X$, we have

$$R_X(\xi) = \mathrm{P}_{\mathcal{M}_k}(X+\xi) = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \tag{16}$$

where u_i and v_i are left and right singular vectors, respectively. It is obvious that when $\sigma_k = 0$ there is no best rank-k solution for (15), and additionally, when $\sigma_{k-1} = \sigma_k$ the minimization in (15) does not have a unique solution. In fact, since the manifold is non-convex [LM08], metric projections can never be well defined on the whole tangent bundle. Fortunately, as indicated in Definition 2.2, the retraction has to be defined only locally because this is sufficient to establish convergence of the Riemannian algorithms.

2.3 Newton's method on $\mathcal{M}_k$ and second-order retractions

The application of any Riemannian Newton-type method to minimize (6), requires the knowledge of the Riemannian Hessian of the objective function f. Although this operator is usually defined in terms of the Levi-Civita connection, in case of an embedded submanifold, there is an elegant way to define this Hessian by means of so-called second-order retractions. Although we do not exploit second-order information in Algorithm 1, we derive in this section such a second-order retraction (and, in the next section, the second-order model of f) since it may be useful for the sake of completeness.

Second-order retractions are defined as second-order approximations of the exponential map [AM10, Proposition 3]. In case of embedded submanifolds, it is shown in [AM10] that they can be identified as retractions that additionally satisfy

$$\mathrm{P}_{T_X\mathcal{M}_k}\left(\frac{\mathrm{d}}{\mathrm{d}\,t^2} R_X(t\xi)\Big|_{t=0}\right) = 0, \quad \text{for all } \xi \in T_X\mathcal{M}_k. \tag{17}$$

Now, the Riemannian Hessian operator, a symmetric and linear operator

$$\operatorname{Hess} f(X) : T_X\mathcal{M}_k \to T_X\mathcal{M}_k,$$

can be determined from the classical Hessian of the function $f \circ R_X : T_X\mathcal{M}_k \to \mathbb{R}$ defined on a vector space since

$$\operatorname{Hess} f(X) = \operatorname{Hess}(f \circ R_X)(0),$$

where R_X is a second-order retraction, see [AMS08, Proposition 5.5.5].

The next proposition gives a second-order retraction and the plan of this construction is as follows: first, we construct a smooth mapping $T_X\mathcal{M}_k \to \mathbb{R}^{m\times n}$; then, we prove that it satisfies the requirements in Definition 2.2 and (17); and finally, we expand in series. Since the metric projection (15) can be shown to satisfy (17) also, we have in addition obtained a second-order Taylor series for (15).

Proposition 2.3 *For any $X = U\Sigma V^T \in \mathcal{M}_k$ satisfying (9) and $\xi \in T_X\mathcal{M}_k$ satisfying (11), the mapping $R_X^{(2)}$, given by*

$$R_X^{(2)} : T_X\mathcal{M}_k \to \mathcal{M}_k, \quad \xi = UMV^T + U_pV^T + UV_p^T \mapsto Z_UZ_V^T, \tag{18}$$

where

$$Z_U := U(\Sigma + \tfrac{1}{2}M - \tfrac{1}{8}M\Sigma^{-1}M) + U_p(I_k - \tfrac{1}{2}\Sigma^{-1}M), \tag{19}$$
$$Z_V := V(I_k + \tfrac{1}{2}M^T\Sigma^{-T} - \tfrac{1}{8}M^T\Sigma^{-T}M^T\Sigma^{-T}) + V_p(\Sigma^{-T} - \tfrac{1}{2}\Sigma^{-T}M^T\Sigma^{-T}), \tag{20}$$

is a second-order approximation of the exponential map on $(g_X, \mathcal{M}_k)$. Furthermore, we have

$$R_X^{(2)}(\xi) = X + \xi + U_p\Sigma^{-1}V_p^T + O(\|\xi\|^3), \quad \|\xi\| \to 0, \tag{21}$$

and

$$R_X^{(2)}(\xi) - R_X(\xi) = O(\|\xi\|^3), \quad \|\xi\| \to 0, \tag{22}$$

with R_X the metric projection (15).

Before proving Proposition 2.3, we first state the following lemma that shows smoothness of $R_X^{(2)}$ in ξ for *fixed* X.

Lemma 2.4 *Using the notation of Proposition 2.3, we have*

$$R_X^{(2)}(\xi) = WX^\dagger W, \tag{23}$$

with

$$\begin{aligned} W := X &+ \tfrac{1}{2}\,\mathrm{P}_X^s(\xi) + \mathrm{P}_X^p(\xi) \\ &- \tfrac{1}{8}\,\mathrm{P}_X^s(\xi)X^\dagger\,\mathrm{P}_X^s(\xi) - \tfrac{1}{2}\,\mathrm{P}_X^p(\xi)X^\dagger\,\mathrm{P}_X^s(\xi) - \tfrac{1}{2}\,\mathrm{P}_X^s(\xi)X^\dagger\,\mathrm{P}_X^p(\xi), \end{aligned}$$

and

$$\mathrm{P}_X^s \colon \mathbb{R}^{n\times m} \to T_X\mathcal{M}_k \colon Z \mapsto \mathrm{P}_U\, Z\, \mathrm{P}_V, \tag{24}$$
$$\mathrm{P}_X^p \colon \mathbb{R}^{n\times m} \to T_X\mathcal{M}_k \colon Z \mapsto \mathrm{P}_U^\perp\, Z\, \mathrm{P}_V + \mathrm{P}_U\, Z\, \mathrm{P}_V^\perp\,. \tag{25}$$

Furthermore, for fixed $X \in \mathcal{M}_k$, the mapping $R_X^{(2)}(\xi)$ is smooth in ξ as a mapping from $T_X\mathcal{M}_k$ to $\mathbb{R}^{m\times n}$.

Proof. Substituting $X^\dagger = V\Sigma^{-1}U^T$ and the definitions (24),(25) for $\mathrm{P}^{\mathrm{s}}_X, \mathrm{P}^{\mathrm{p}}_X$ in W gives

$$\begin{aligned} W &= U\Sigma V^T + \tfrac{1}{2}UMV^T + U_pV^T + VV_p^T - \tfrac{1}{8}UMV^TV\Sigma^{-1}U^TUMV^T \\ &\quad - \tfrac{1}{2}(U_pV^T + UV_p^T)V\Sigma^{-1}U^TUMV^T - \tfrac{1}{2}UMV^TV\Sigma^{-1}U^TU_pV^T + UV_p^T, \\ &= U\Sigma V^T + \tfrac{1}{2}UMV^T + U_pV^T + UV_p^T - \tfrac{1}{8}UM\Sigma^{-1}MV^T \\ &\quad - \tfrac{1}{2}U_p\Sigma^{-1}MV^T - \tfrac{1}{2}UM\Sigma^{-1}V_p^T. \end{aligned}$$

This shows the equivalence of (23) to (18) after some tedious but straightforward algebraic manipulations.

Observe that (23) consists of only smooth operations involving X: taking transposes, multiplying matrices and inverting full rank matrices can all be expressed as rational functions without singularities in the matrix entries. From this, we have immediately smoothness of $R^{(2)}_X(\xi)$ in ξ. □

The previous lemma shows smoothness of $R^{(2)}_X : T_X\mathcal{M}_k \to \mathbb{R}^{m\times n}$. For Proposition 2.3, we also need to prove local smoothness of $R^{(2)}$ as a function from the tangent bundle onto $\mathcal{M}_k$. We will do this by relying on Boman's theorem [Bom67, Theorem 1], which states that a function $f : \mathbb{R}^d \to \mathbb{R}$ is smooth if $f \circ \gamma$ is smooth along all smooth curves $\gamma : \mathbb{R} \to \mathbb{R}^d$.

In addition, for the proof below we will allow the matrix Σ in definition (18) to be any full-rank matrix. This is possible because none of the derivations above relied on Σ being diagonal and definition (18) for $R^{(2)}_X$ is independent of the choice of specific matrices U, V, Σ in the factorization of X. Indeed, suppose that we would have chosen $\widetilde{U} = UQ_U$, $\widetilde{V} = VQ_V$ and $\widetilde{\Sigma} = Q_U^T\Sigma Q_V$ as factorization matrices. Then the coefficient matrices of the tangent vector ξ would have been $\widetilde{M} = Q_U^TMQ_V$, $\widetilde{U}_p = U_pQ_V$ and $\widetilde{V} = V_pQ_U$. Now, it is straightforward to check that the corresponding expressions for (19)–(20) become $Z_{\widetilde{U}} = Z_UQ_V$ and $Z_{\widetilde{V}} = Z_VQ_V$. Hence, $R^{(2)}_X(\xi) = Z_{\widetilde{U}}Z_{\widetilde{V}}^T = Z_UZ_V^T$ stays the same.

Proof of Proposition 2.3. First, we show (21). Elementary manipulation using (19)–(20) reveals that

$$R^{(2)}_X(\xi) = \begin{bmatrix} U & U_p \end{bmatrix} \begin{bmatrix} \Sigma + M + O(\|M\|^3) & I + O(\|M\|^2) \\ I + O(\|M\|^2) & \Sigma^{-1} + O(\|M\|) \end{bmatrix} \begin{bmatrix} V & V_p \end{bmatrix}^T, \quad \|M\| \to 0.$$

For fixed X, the norms $\|M\|, \|U_p\|, \|V_p\|$ are all $O(\|\xi\|)$, and so

$$R^{(2)}_X(\xi) = U(\Sigma + M)V^T + UV_p^T + U_pV^T + U_p\Sigma^{-1}V_p^T + O(\|\xi\|^3), \quad \|\xi\| \to 0,$$

which can be rewritten as (21). Next, we show that $R^{(2)}_X$ is a retraction according to Definition 2.2. By construction, the image of R_X is a matrix $Z_UZ_V^T$ of rank at most k. Since $\mathrm{rank}(X)$ is continuous in X, one can take $\delta_t > 0$ sufficiently small such that the image of $t \mapsto R^{(2)}_X(t\xi)$ is always of rank k for all $t \in (-\delta_t, \delta_t)$.

Finally, we establish smoothness of $R^{(2)}_X$ in a neighborhood of $(X, 0) \in T\mathcal{M}_k$. Take any $(Y, \xi) \in T\mathcal{M}_k$ sufficiently close to $(X, 0)$ and consider any smooth curve $\gamma : \mathbb{R} \to T\mathcal{M}_k$ that connects $(X, 0)$ to (Y, ξ). Consider the partitioning $\gamma(t) = (\alpha(t), \beta(t))$ with α a smooth curve on $\mathcal{M}_k$ and β on $T_{\alpha(t)}\mathcal{M}_k$. By the existence of a smooth SVD for fixed-rank matrices [CD00, Theorem 2.4], there exists smooth $U(t), V(t), \Sigma(t)$ (like mentioned above, $\Sigma(s)$ is now a full-rank matrix) such that

$\alpha(t) = U(t)\Sigma(t)V(t)^T$. Furthermore by Lemma 2.4 map $R^{(2)}_{\alpha(t)}(\xi)$ is smooth in ξ around zero, so it will be smooth along $R^{(2)}_{\alpha(t)}(\beta(t))$. Then, the expression $R_{\alpha(t)}(\beta(t))$ consists of smooth matrix operations that depend smoothly on t. Since the curve $\gamma(t) = (\alpha(t), \beta(t))$ was chosen arbitrarily, we have shown smoothness of $R_X(\xi)$ on the tangent bundle in a neighborhood of $(X, 0)$. To conclude, observe from (21) that $R^{(2)}_X$ obeys (17) since $U_p\Sigma^{-1}V_p^T$ is orthogonal to the tangent space. This makes $R^{(2)}_X$ a second-order retraction. □

Although parts of the previous results can also be found in [SWC10, Theorem 1], we prove additionally that $R^{(2)}_X$ satisfies Definition 2.2 by showing smoothness on the whole tangent bundle (and not only for each tangent space separately, as in [SWC10]).

2.4 Second-order model of f on $\mathcal{M}_k$

Based on the Taylor series of $R^{(2)}_X$, it is now straightforward to derive the second-order model

$$m(X) = f(X) + \langle \operatorname{grad} f(X),\, \xi\rangle + \frac{1}{2}\langle \operatorname{Hess} f(X)[\xi],\, \xi\rangle$$

of any smooth objective function $f : \mathcal{M}_k \to \mathbb{R}$. In this section, we will illustrate this with our objective function $f(X) = \frac{1}{2}\| \operatorname{P}_\Omega(X - A)\|^2_{\mathrm{F}}$.

Expanding $f(R^{(2)}_X(\xi))$ using (21) gives for $\|\xi\| \to 0$

$$\begin{aligned} f(R^{(2)}_X(\xi)) &= \frac{1}{2}\| \operatorname{P}_\Omega(R^{(2)}_X(\xi) - A)\|^2_{\mathrm{F}} \\ &= \frac{1}{2}\| \operatorname{P}_\Omega(X + \xi + U_p\Sigma^{-1}V_p^T - A)\|^2_{\mathrm{F}} + O(\|\xi\|^3) \\ &= \frac{1}{2}\operatorname{tr}[\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(X - A) + 2\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(\xi) \\ &\quad + 2\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(U_p\Sigma^{-1}V_p^T) + \operatorname{P}_\Omega(\xi)^T \operatorname{P}_\Omega(\xi)] + O(\|\xi\|^3). \end{aligned}$$

We can recognize the constant term

$$f(X) = \frac{1}{2}\operatorname{tr}[\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(X - A)],$$

the linear term

$$\langle \operatorname{grad} f(X),\, \xi\rangle = \operatorname{tr}[\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(\xi)], \tag{26}$$

and the quadratic term

$$\langle \operatorname{Hess} f(X)[\xi],\, \xi\rangle = \operatorname{tr}[2\operatorname{P}_\Omega(X - A)^T \operatorname{P}_\Omega(U_p\Sigma^{-1}V_p^T) + \operatorname{P}_\Omega(\xi)^T \operatorname{P}_\Omega(\xi)]. \tag{27}$$

Using (14), we see that our expression for the Riemannian gradient, $\operatorname{grad} f(x) = \operatorname{P}_{T_X\mathcal{M}_k}(\operatorname{P}_\Omega(X - A))$, indeed satisfies (26). This only leaves determining the Riemannian Hessian as the symmetric and linear mapping

$$\operatorname{Hess} f(X) : T_X\mathcal{M}_k \to T_X\mathcal{M}_k, \quad \xi \mapsto \operatorname{Hess} f(X)[\xi],$$

that satisfies the inner product (27). First note that $U_p\Sigma^{-1}V_p^T$ is quadratic in ξ since

$$U_p\Sigma^{-1}V_p^T = (U_pV^T + UV_p^T)(V\Sigma^{-1}U^T)(U_pV^T + UV_p^T) = \operatorname{P}^{\mathrm{p}}_X(\xi)X^\dagger \operatorname{P}^{\mathrm{p}}_X(\xi).$$

Now we get

$$\begin{aligned}\langle \operatorname{Hess} f(X)[\xi],\, \xi\rangle &= 2\langle \mathrm{P}_\Omega(X-A),\, \mathrm{P}_\Omega(\mathrm{P}^{\mathrm{p}}_X(\xi)X^\dagger\,\mathrm{P}^{\mathrm{p}}_X(\xi))\rangle + \langle \mathrm{P}_\Omega(\xi),\, \mathrm{P}_\Omega(\xi)\rangle \\ &= \underbrace{2\langle \mathrm{P}_\Omega(X-A),\, \mathrm{P}^{\mathrm{p}}_X(\xi)X^\dagger\,\mathrm{P}^{\mathrm{p}}_X(\xi)\rangle}_{f_1 := \langle \mathcal{H}_1(\xi),\,\xi\rangle} + \underbrace{\langle \mathrm{P}_\Omega(\xi),\, \xi\rangle}_{f_2 := \langle \mathcal{H}_2(\xi),\,\xi\rangle}.\end{aligned}$$

So, the inner product with the Hessian consists of two parts, f_1 and f_2, and the Hessian itself is the sum of two operators, $\mathcal{H}_1$ and $\mathcal{H}_2$. The second contribution to the Hessian is readily available from f_2:

$$f_2 = \langle \mathrm{P}_\Omega(\xi),\, \xi\rangle = \langle \mathrm{P}_\Omega(\xi),\, \mathrm{P}_{T_X\mathcal{M}_k}(\xi)\rangle = \langle \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}\mathrm{P}_\Omega(\xi),\, \xi\rangle,$$

and so

$$\mathcal{H}_2(\xi) := \mathrm{P}_{T_X\mathcal{M}_k}\mathrm{P}_\Omega(\xi) = \mathrm{P}_U\,\mathrm{P}_\Omega(\xi)\,\mathrm{P}_V + \mathrm{P}^\perp_U\,\mathrm{P}_\Omega(\xi)\,\mathrm{P}_V + \mathrm{P}_U\,\mathrm{P}_\Omega(\xi)\,\mathrm{P}^\perp_V\,.$$

In f_1 we still need to separate ξ to one side of the inner product. Since we can choose whether we bring over the first $\mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)$ or the second, we introduce a constant $c \in \mathbb{R}$,

$$\begin{aligned} f_1 &= 2c\,\langle \mathrm{P}_\Omega(X-M)\cdot(X^\dagger \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi))^T,\, \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)\rangle \\ &\quad + 2(1-c)\,\langle (\mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)X^\dagger)^T\cdot \mathrm{P}_\Omega(X-A),\, \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)\rangle \\ &= 2c\,\langle \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}[\mathrm{P}_\Omega(X-A)\cdot \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)^T\cdot (X^\dagger)^T],\, \xi\rangle \\ &\quad + 2(1-c)\,\langle \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}[(X^\dagger)^T\cdot \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)^T\cdot \mathrm{P}_\Omega(X-A)],\, \xi\rangle. \end{aligned}$$

The operator $\mathcal{H}_1$ is clearly linear. Imposing the symmetry requirement $\langle \mathcal{H}_1(\xi),\, \nu\rangle = \langle \xi,\, \mathcal{H}_1(\nu)\rangle$ for any arbitrary tangent vector ν, we get that $c = 1/2$, and so

$$\begin{aligned}\mathcal{H}_1(\xi) &= \mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}[\mathrm{P}_\Omega(X-A)\mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)^T(X^\dagger)^T + (X^\dagger)^T\mathrm{P}^{\mathrm{p}}_{T_X\mathcal{M}_k}(\xi)^T\,\mathrm{P}_\Omega(X-A)] \\ &= \mathrm{P}^\perp_U\,\mathrm{P}_\Omega(X-A)V_p\Sigma^{-1}V^T + U\Sigma^{-1}U_p^T\,\mathrm{P}_\Omega(X-A)\,\mathrm{P}^\perp_V\,.\end{aligned}$$

Finally, we can put together the whole Hessian

$$\begin{aligned}\operatorname{Hess} f(X)[\xi] = \mathrm{P}_U\,\mathrm{P}_\Omega(\xi)\,\mathrm{P}_V + \mathrm{P}^\perp_U(\mathrm{P}_\Omega(\xi) + \mathrm{P}_\Omega(X-A)V_p\Sigma^{-1}V^T)\,\mathrm{P}_V \\ + \mathrm{P}_U(\mathrm{P}_\Omega(\xi) + U\Sigma^{-1}U_p^T\,\mathrm{P}_\Omega(X-A))\,\mathrm{P}^\perp_V\,.\end{aligned}$$

2.5 Vector transport

Vector transport was introduced in [AMS08] and [QGA10] as a means to transport tangent vectors from one tangent space to another. In a similar way as retractions are approximations of the exponential mapping, vector transport is the first-order approximation of parallel transport, another important concept in differential geometry.

The definition below makes use of the Whitney sum, which is the vector bundle over some space obtained as the direct sum of two vector bundles over that same space. In our setting, we can define it as

$$\begin{aligned}T\mathcal{M}_k \oplus T\mathcal{M}_k &= \bigcup_{X\in\mathcal{M}_k} \{X\}\times T_X\mathcal{M}_k \times T_X\mathcal{M}_k \\ &= \{(X,\eta,\xi)\colon X\in\mathcal{M}_k, \xi\in T_X\mathcal{M}_k, \eta\in T_X\mathcal{M}_k\}.\end{aligned}$$

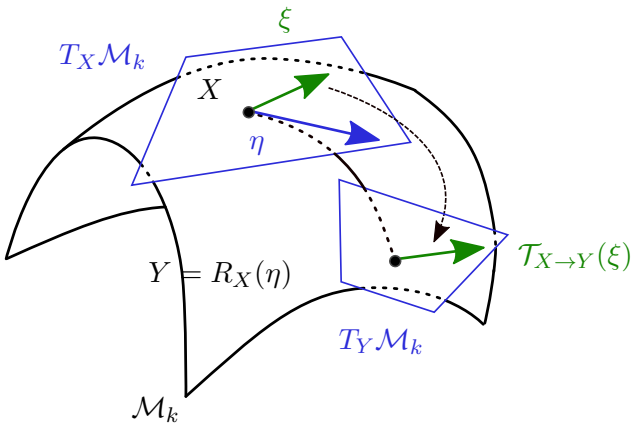

Figure 2: Vector transport on a Riemannian manifold.

Definition 2.5 (Vector transport [QGA10, Definition 2]) *A vector transport on the manifold $\mathcal{M}_k$ is a smooth mapping: $T\mathcal{M}_k \oplus T\mathcal{M}_k \to T\mathcal{M}_k, (X, \eta, \xi) \mapsto \mathcal{T}_\eta(\xi)$ satisfying the following properties for all $X \in \mathcal{M}_k$:*

(1) There exists a retraction R such that, for all $(X, \eta, \xi) \in T\mathcal{M}_k \oplus T\mathcal{M}_k$, it holds that $\mathcal{T}_\eta(\xi) \in T_{R_X(\eta)}\mathcal{M}_k$.

(2) $\mathcal{T}_0(\xi) = \xi$ for all $\xi \in T_X\mathcal{M}_k$.

(3) For all $(X, \eta) \in T\mathcal{M}_k$, the mapping $\mathcal{T}_\eta : T_X\mathcal{M}_k \to T_{R_X(\eta)}\mathcal{M}_k$, $\xi \mapsto \mathcal{T}_\eta(\xi)$ is linear.

By slight abuse of notation, we will use the following shorthand notation that emphasizes the two tangent spaces involved:

$$\mathcal{T}_{X\to Y} : T_X\mathcal{M}_k \to T_Y\mathcal{M}_k,\ \xi \mapsto \mathcal{T}_{R_X^{-1}(Y)}(\xi),$$

where $\mathcal{T}_{R_X^{-1}(Y)}$ uses the notation of Definition 2.5. See also Figure 2 for an illustration of this definition. By the implicit function theorem, R_X is locally invertible for every $X \in \mathcal{M}_k$ but $R_X^{-1}(Y)$ does not need to exist for all $Y \in \mathcal{M}_k$. Hence, $\mathcal{T}_{X\to Y}$ has to be understood locally, i.e., for Y sufficiently close to X.

Since $\mathcal{M}_k$ is an embedded submanifold of $\mathbb{R}^{m\times n}$, orthogonally projecting the translated tangent vector in $\mathbb{R}^{m\times n}$ onto the new tangent space constitutes a vector transport, see [AMS08, Section 8.1.2]. In other words, we get

$$\mathcal{T}_{X\to Y} : T_X\mathcal{M}_k \to T_Y\mathcal{M}_k,\ \xi \mapsto \mathrm{P}_{T_Y\mathcal{M}_k}(\xi), \tag{28}$$

with $\mathrm{P}_{T_Y\mathcal{M}_k}$ as defined in (13).

As explained in Section 1.1, the conjugate search direction η_i in Algorithm 1 is computed as a linear combination of the gradient and the previous direction:

$$\eta_i := -\operatorname{grad} f(X_i) + \beta_i\, \mathcal{T}_{X_{i-1}\to X_i}(\eta_{i-1}),$$

where we transported η_{i-1}. For β_i, we have chosen the geometrical variant of Polak–Ribière (PR+), as introduced in [AMS08, Chapter 8.2]. Again using vector transport, this becomes

$$\beta_i = \frac{\langle \operatorname{grad} f(X_i),\, \operatorname{grad} f(X_i) - \mathcal{T}_{X_{i-1}\to X_i}(\operatorname{grad} f(X_{i-1}))\,\rangle}{\langle \operatorname{grad} f(X_{i-1}), \operatorname{grad} f(X_{i-1})\,\rangle}. \tag{29}$$

3 Implementation details

This section is devoted to the implementation of Algorithm 1.

Low-rank matrices Since every element $X \in \mathcal{M}_k$ is a rank k matrix, we store it as the result of a compact SVD:

$$X = U\Sigma V^T,$$

with orthonormal matrices $U \in \mathrm{St}_k^m$ and $V \in \mathrm{St}_k^n$, and a diagonal matrix $\Sigma \in \mathbb{R}^{k\times k}$ with decreasing positive entries on the diagonal. For the computation of the objective function and the gradient, we also precompute the sparse matrix $X_\Omega := \mathrm{P}_\Omega(X)$. As explained in the next paragraph, computing X_Ω costs roughly $2(m + |\Omega|)k$ flops.

Projection operator P_Ω During the course of Algorithm 1, we require the application of P_Ω, as defined in (2), to certain low-rank matrices. By exploiting the low-rank form, this can be done efficiently as follows: Let Z be a rank-k_Z matrix with factorization $Z := Y_1 Y_2^T$ (k_Z is possibly different from k, the rank of the matrices in $\mathcal{M}_k$). Define $Z_\Omega := \mathrm{P}_\Omega(Z)$. Then element (i,j) of Z_Ω is given by

$$(Z_\Omega)_{i,j} = \begin{cases} \sum_{l=1}^{k_Z} Y_1(i,l)Y_2(j,l), & \text{if } (i,j) \in \Omega, \\ 0, & \text{if } (i,j) \notin \Omega. \end{cases} \tag{30}$$

The cost for computing Z_Ω is roughly $2|\Omega|k_Z$ flops.

Tangent vectors A tangent vector $\eta \in T_X\mathcal{M}_k$ at $X = U\Sigma V^T \in \mathcal{M}_k$ will be represented as

$$\eta = UMV^T + U_pV^T + UV_p^T, \tag{31}$$

where $M \in \mathbb{R}^{k\times k}$, $U_p \in \mathbb{R}^{m\times k}$ with $U^TU_p = 0$, and $V_p \in \mathbb{R}^{n\times k}$ with $V^TV_p = 0$.

Since X is available as an SVD, the orthogonal projection onto the tangent space $T_X\mathcal{M}_k$ becomes

$$\mathrm{P}_{T_X\mathcal{M}_k}(Z) := \mathrm{P}_U\, Z\, \mathrm{P}_V + (Z - \mathrm{P}_U\, Z)\, \mathrm{P}_V + \mathrm{P}_U(Z^T - \mathrm{P}_V\, Z^T)^T,$$

where $\mathrm{P}_U := UU^T$ and $\mathrm{P}_V := VV^T$. If Z can be efficiently applied to a vector, evaluating and storing the result of $\mathrm{P}_{T_X\mathcal{M}_k}(Z)$ as a tangent vector in the format (31) can also be performed efficiently.

Riemannian gradient From (14), the Riemannian gradient at $X = U\Sigma V^T$ is the orthogonal projection of $X_\Omega - A_\Omega$ onto the tangent space at X. Since the sparse matrix $A_\Omega := \mathrm{P}_\Omega(A)$ is given and X_Ω was already precomputed, the gradient can be computed by Algorithm 2. The total cost is roughly $2(n + 2m)k^2 + 4|\Omega|k$ flops.

Algorithm 2 Calculate gradient $\operatorname{grad} f(X)$

Require: matrix $X = U\Sigma V^T \in \mathcal{M}_k$, sparse matrix $R = X_\Omega - A_\Omega \in \mathbb{R}^{m\times n}$.
Ensure: $\operatorname{grad} f(X) = UMV^T + U_pV^T + UV_p^T \in T_X\mathcal{M}_k$
1: $R_u \leftarrow R^TU, \quad R_v \leftarrow RV$ $\{4|\Omega|k$ flops$\}$
2: $M \leftarrow U^TR_v$ $\{2mk^2$ flops$\}$
3: $U_p \leftarrow R_v - UM, \quad V_p \leftarrow R_u - VM^T$ $\{2(m+n)k^2$ flops$\}$

Vector transport Let $\nu = UMV^T + U_pV^T + UV_p^T$ be a tangent vector at $X = U\Sigma V^T \in \mathcal{M}_k$. By (28), the vector

$$\nu_+ := \mathcal{T}_{X\to X_+}(\nu) = \mathrm{P}_{T_{X_+}\mathcal{M}_k}(\nu)$$

is the transport of ν to the tangent space at some $X_+ = U_+\Sigma_+V_+^T$. It is computed by Algorithm 3 with a total cost of approximately $12(m+n)k^2 + 12k^3$ flops.

Algorithm 3 Calculate vector transport $\mathcal{T}_{X\to X_+}(\nu_X)$

Require: matrices $X = U\Sigma V^T \in \mathcal{M}_k$ and $X_+ = U_+\Sigma_+V_+^T \in \mathcal{M}_k$,
tangent vector $\nu_X = UMV^T + U_pV^T + UV_p^T$.
Ensure: $\mathcal{T}_{X\to X_+}(\nu_X) = U_+M_+V_+^T + U_{p_+}V_+^T + U_+V_{p_+}^T \in T_{X_+}\mathcal{M}_k$
1: $A_v \leftarrow V^TV_+, \quad A_u \leftarrow U^TU_+$ $\{2(m+n)k^2\}$
2: $B_v \leftarrow V_p^TV_+, \quad B_u \leftarrow U_p^TU_+$ $\{2(m+n)k^2\}$
3: $M_+^{(1)} \leftarrow A_u^TMA_v, \quad U_+^{(1)} \leftarrow U(MA_v), \quad V_+^{(1)} \leftarrow V(M^TA_u)$ $\{8k^3 + 2(m+n)k^2\}$
4: $M_+^{(2)} \leftarrow B_u^TA_v, \quad U_+^{(2)} \leftarrow U_pA_v, \quad V_+^{(2)} \leftarrow VB_u$ $\{2k^3 + 2(m+n)k^2\}$
5: $M_+^{(3)} \leftarrow A_u^TB_v, \quad U_+^{(3)} \leftarrow UB_v, \quad V_+^{(3)} \leftarrow V_pA_u$ $\{2k^3 + 2(m+n)k^2\}$
6: $M_+ \leftarrow M_+^{(1)} + M_+^{(2)} + M_+^{(3)}$ $\{2k^2\}$
7: $U_{+_p} \leftarrow U_+^{(1)} + U_+^{(2)} + U_+^{(3)}, \quad U_{p_+} \leftarrow U_{p_+} - U_+(U_+^TU_{p_+})$ $\{2mk^2\}$
8: $V_{+_p} \leftarrow V_+^{(1)} + V_+^{(2)} + V_+^{(3)}, \quad V_{p_+} \leftarrow V_{p_+} - V_+(V_+^TV_{p_+})$ $\{2nk^2\}$

Non-linear CG The geometric variant of Polak–Ribière is implemented as Algorithm 4 costing about $30(m+n)k^2 + 30k^3$ flops. In order to improve robustness, we use the PR+ variant and a restart when two consecutive gradients are almost collinear; see, e.g., [NW06, (5.52)].

Initial guess for line search Although Algorithm 1 uses line search, a good initial guess can greatly enhance performance. We observed in our numerical experiments that an exact minimization on the tangent space alone (so, neglecting the retraction),

$$\min_t f(X + t\,\eta) = \frac{1}{2}\min_t \|\,\mathrm{P}_\Omega(X) + t\,\mathrm{P}_\Omega(\eta) - \mathrm{P}_\Omega(A)\|_{\mathrm{F}}^2, \tag{32}$$

performed extremely well as initial guess, in the sense that backtracking is almost never necessary.

Equation (32) is essentially a one-dimensional least-square fit for t on Ω. The closed-form solution satisfies

$$\langle\,\mathrm{P}_\Omega(\eta), \mathrm{P}_\Omega(A - X)\,\rangle \,/\, \langle\,\mathrm{P}_\Omega(\eta), \mathrm{P}_\Omega(\eta)\,\rangle$$

Algorithm 4 Compute the conjugate direction by PR+

Require: previous iterate X_{i-1}, previous gradient ξ_{i-1}, previous direction η_{i-1}
current iterate X_i, current gradient ξ_i

Ensure: conjugate direction $\eta_i \in T_{X_i}\mathcal{M}_k$

1: Transport previous gradient and direction to current tangent space:
$\overline{\xi}_i \leftarrow \mathcal{T}_{X_{i-1}\to X_i}(\xi_{i-1})$ {apply Algorithm 3}
$\overline{\eta}_i \leftarrow \mathcal{T}_{X_{i-1}\to X_i}(\eta_{i-1})$ {apply Algorithm 3}
2: Compute orthogonality between previous and current gradient:
$\theta \leftarrow \langle \overline{\xi}_i, \xi_i\rangle / \langle \xi_i, \xi_i\rangle$
3: Compute the conjugate direction:
$\delta_i \leftarrow \xi_i - \overline{\xi}_i$
if $\theta \geq 0.1$ then $\beta \leftarrow 0$
else $\beta \leftarrow \max(0, \langle \delta_i, \xi_i\rangle / \langle \xi_{i-1}, \xi_{i-1}\rangle)$
$\eta_i \leftarrow -\xi_i + \beta\, \overline{\eta}_i$

and is unique when $\eta \neq 0$. As long as Algorithm 1 has not converged, η will always be non-zero since it is the direction of search. The solution to (32) is performed by Algorithm 5. In the actual implementation, the sparse matrices N and B are stored as vectors on Ω alone. Hence, the total cost is about $4(k+1)|\Omega| + 2mk^2$ flops.

Algorithm 5 Compute the initial guess for line search $t_* = \arg\min_t f(X + t\,\nu)$

Require: iterate $X = U\Sigma V^T$ and projection X_Ω, tangent vector $\nu = UMV^T + U_pV^T + UV_p^T$, sparse matrix $R = X_\Omega - A_\Omega \in \mathbb{R}^{m\times n}$

Ensure: step length t_*

1: $N \leftarrow \mathrm{P}_\Omega(\begin{bmatrix}UM + U_p & U\end{bmatrix}\begin{bmatrix}V & V_p\end{bmatrix}^T)$ {See (30)}
2: $t_* \leftarrow -\operatorname{tr}(N^TR)/\operatorname{tr}(R^TR)$ {$4|\Omega|$ flops}

Retraction As shown in (16), the retraction (15) can be directly computed by the SVD of $X+\xi$. A full SVD of $X+\xi$ is, however, prohibitively expensive since it costs $O(n^3)$ flops. Fortunately, the matrix to retract has the particular form

$$X + \xi = \begin{bmatrix}U & U_p\end{bmatrix}\begin{bmatrix}\Sigma + M & I_k\\ I_k & 0\end{bmatrix}\begin{bmatrix}V & V_p\end{bmatrix}^T,$$

with $X = U\Sigma V^T \in \mathcal{M}_k$ and $\xi = UMV^T + U_pV^T + UV_p^T \in T_X\mathcal{M}_k$. Algorithm 6 now performs an economy-sized QR and an SVD of a small $2k$-by-$2k$ matrix to reduce the naive flop count of $O(n^3)$ to $10(m+n)k^2$ when $k \ll \min(m,n)$.

Observe that the listing of Algorithm 6 uses Matlab notation to denote common matrix operations. In addition, the flop count of computing the SDV in Step 3 is given as $O(k^3)$ since it is difficult to estimate beforehand for all matrices. In practice, the constant of $O(k^3)$ is modest, say, less than 100. Furthermore, in Step 4, we have added ϵ_{mach} to Σ_s so that, in the unlucky event of zero singular values, the retracted matrix is perturbed to a rank-k matrix in $\mathcal{M}_k$.

Algorithm 6 Compute the retraction by metric projection

Require: iterate $X = U\Sigma V^T$, tangent vector $\xi = UMV^T + U_pV^T + UV_p^T$

Ensure: retracted element $R_X(\xi) = \mathrm{P}_{\mathcal{M}_k}(X + \xi) = U_+\Sigma_+V_+^T$

1: $Q_u, R_u \leftarrow \mathrm{qr}(U_p, 0), \quad Q_v, R_v \leftarrow \mathrm{qr}(V_p, 0)$ $\{6(m+n)k^2$ flops$\}$

2: $S \leftarrow \begin{bmatrix} \Sigma + M & R_v^T \\ R_u & 0 \end{bmatrix}$

3: $U_s, \Sigma_s, V_s \leftarrow \mathrm{svd}(S)$ $\{O(k^3)$ flops$\}$

4: $\Sigma_+ \leftarrow \Sigma_s(1:k, 1:k) + \epsilon_{\mathrm{mach}}$

5: $U_+ \leftarrow \begin{bmatrix} U & Q_u \end{bmatrix} U_s(:, 1:k), \quad V_+ \leftarrow \begin{bmatrix} V & Q_v \end{bmatrix} V_s(:, 1:k)$ $\{4(m+n)k^2$ flops$\}$

Computational cost Summing all the flop counts for Algorithm 1, we arrive at a cost per iteration of approximately

$$(58 + 52)k^2 + 14|\Omega|k + \mathrm{nb}_{\mathrm{Armijo}}\, 14(m+n)k^2,$$

where $\mathrm{nb}_{\mathrm{Armijo}}$ denotes the average number of Armijo backtracking steps. It is remarkable, that in all experiments below we have observed that $\mathrm{nb}_{\mathrm{Armijo}} = 0$, that is, backtracking was never needed.

For our typical problems in Section 5, the size of the sampling set satisfies $|\Omega| = \mathrm{OS}\,(m+n-k)k$ with $\mathrm{OS} > 2$ the oversampling factor. When $m = n$ and $\mathrm{nb}_{\mathrm{Armijo}} = 0$, this brings the total cost per iteration to

$$t_{\mathrm{theoretical}} = (110 + 28\,\mathrm{OS})nk^2. \tag{33}$$

Comparing this theoretical flop count with experimental results, we observe that sparse matrix-vector multiplications and applying P_Ω require significantly more time then predicted by (33). This can be explained due to the lack of data locality for these sparse matrix operations, whereas the majority of the remaining time is spent by dense linear algebra, rich in level 3 BLAS.

A small experiment reveals that in practice P_Ω is about 5 to 10 times slower then QR for an equivalent theoretical flop count. Similarly, computing a sparse matrix-vector product is about 2 to 5 times slower then QR. Larger problems tend to be faster. Normalizing the theoretical estimate (33) to the equivalent flop count of QR, we obtain the following practical estimate:

$$t_{\mathrm{practical}} = (110 + C_{\mathrm{sparse}}\,\mathrm{OS})nk^2, \quad C_{\mathrm{sparse}} \simeq 80, \ldots, 200. \tag{34}$$

For a sensible value of $\mathrm{OS} = 3$, we can expect that about 70 to 85 percent of the computational time will be spent on operations with sparse matrices.

Comparison to existing methods The vast majority of specialized solvers for matrix completion require computing the dominant singular vectors of a sparse matrix in each step of the algorithm. This is, for example, the case for approaches using soft- and hard-thresholding, like in [CCS10], [MGC11], [MJD10], [LCWM09], [GM11], [TY10], [LST10], [CO10]. Typically, PROPACK from [Lar04] is used for computing such a truncated SVD. The use of sparse matrix-vector products and the potential convergence issues, frequently makes this the computationally most expensive part of all these algorithms.

On the other hand, our algorithm first projects any sparse matrix onto the tangent space, and then computes the dominant singular vectors for a small $2k$-by-$2k$ matrix. Apart from the

application of P_Ω and *a few* sparse matrix-vector products, the rest of the algorithm solely consists of dense linear algebra operations. This is more robust and significantly faster than computing the SVD of a sparse matrix in each step. To the best of our knowledge, only one competitive low-rank matrix completion algorithm, namely LMAFit of [WYZ10], suffices with only dense algebra together with the application of P_Ω and a few sparse matrix-vector products.

Due to the difficulty of estimating practical flop counts for algorithms based on computing a sparse SVD, we will only compare the computational complexity of our method with that of LMAFit. An estimate similar to the one of above, reveals that LMAFit has a practical flop count of

$$t^{\text{LMAFit}}_{\text{practical}} = (10 + C_{\text{sparse}}\,\text{OS})nk^2, \quad C^{\text{LMAFit}}_{\text{sparse}} \simeq 30, \ldots, 50. \tag{35}$$

Comparing this estimate to (34), LMAFit should be about 3.5 to 4.5 times faster per iteration when $\text{OS} = 3$. As we will see later, this is almost what we observe experimentally, namely a factor of 3. Since LRGeomCG is more costly per iteration, it will have to converge much faster in order to compete with LMAFit. This is studied in detail in Section 5.

4 Convergence

In this section, we will show that Algorithm 1 converges to a critical point of f under standard matrix completion assumptions. Using (14), this means

$$\|\operatorname{grad} f(X_*)\|_{\mathrm{F}} = 0 \quad \Longleftrightarrow \quad \mathrm{P}_{T_{X_*}\mathcal{M}_k}\,\mathrm{P}_\Omega(X_*) = \mathrm{P}_{T_{X_*}\mathcal{M}_k}\,\mathrm{P}_\Omega(A).$$

In other words, we will have fitted X to the data A on the image of $\mathrm{P}_{T_{X_*}\mathcal{M}_k}\,\mathrm{P}_\Omega$ and, hopefully, this is sufficient to have $X_* = A$ on the whole space $\mathbb{R}^{m\times n}$. Without further assumptions on X_* and A, however, it is not reasonable to expect that X_* equals A since P_Ω can have a very large null space. In the following section, we define such a suitable assumption.

4.1 RIP for incoherent matrices

Let us first split the error $E := X - A$ into the three quantities

$$\begin{aligned} E_1 &:= \mathrm{P}_{T_X\mathcal{M}_k}\,\mathrm{P}_\Omega(X - A), \\ E_2 &:= \mathrm{P}_{N_X\mathcal{M}_k}\,\mathrm{P}_\Omega(X - A), \\ E_3 &:= \mathrm{P}^{\perp}_{\Omega}(X - A), \end{aligned}$$

where $N_X\mathcal{M}_k$ is the normal space of X, which is the subspace of all matrices $Z \in \mathbb{R}^{m\times n}$ orthogonal to $T_X\mathcal{M}_k$.

Upon convergence of Algorithm 1, $E_1 \to 0$, but how big can E_2 be? In general, E_2 can be arbitrary large since it is the Lagrange multiplier of the fixed rank constraint of X. In fact, when the rank of X is smaller than the exact rank of A, E_2 typically never vanishes, even though E_1 converges to zero. On the other hand, when the ranks of X and A are the same, one typically observes that $E_1 \to 0$ implies $E_2 \to 0$. This can be easily checked during the course of the algorithm, since $\mathrm{P}_\Omega(X - A) = E_1 + E_2$ is computed explicitly for the computation of the gradient in Algorithm 2.

Suppose now that $\|E_1\|_{\mathrm{F}} \le \epsilon_1$ and $\|E_2\|_{\mathrm{F}} \le \epsilon_2$, which means that X is in good agreement with A on Ω. The hope is that X and A agree on the complement of Ω too. This is of course

not always the case, but it is exactly the crucial assumption of low-rank matrix completion that the observations on Ω are sufficient to complete A. To quantify this assumption, one can make use of standard theory in matrix completion literature; see, e.g., [CT09], [CR09], [KMO10b]. For our purpose, the modified restricted isometry property (RIP) as introduced in [MJD10] is most appropriate.

Define $\delta_k := \delta_k(\mathcal{L})$ to be the constant belonging to the RIP of the linear map $\mathcal{L} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$. It is the smallest number $0 < \delta_k < 1$ such that

$$(1 - \delta_k)\|X\|_{\mathrm{F}}^2 \leq \|\mathcal{L}(X)\|_{\mathrm{F}}^2 \leq (1 + \delta_k)\|X\|_{\mathrm{F}}^2 \tag{36}$$

holds for all matrices X of rank at most k. Ideally, δ_k should be as small as possible since then $\mathcal{L}$ is almost an isometry after restriction to $\mathcal{M}_k$. In addition, when $\delta_{2k} < 1$, recovering any rank-k matrix via $\mathcal{L}$ is well-posed since rank minimization can only have one rank-k solution (Theorem 3.2 in [RFP10]).

Despite its elegant formulation, it is well-known that RIP does not hold for matrix completion, that is, when $\mathcal{L} = \mathrm{P}_\Omega$, since it is trivial to construct nonzero matrices in the null space of P_Ω. In [MJD10], however, it has been shown that for so-called *incoherent* matrices, RIP does hold with very high probability for certain random sampling operators P_Ω.

Definition 4.1 *A matrix $X \in \mathbb{R}^{m\times n}$ with $X = U\Sigma V^T$ as SVD is μ-incoherent if $\max_{i,j} |U_{i,j}| \leq \sqrt{\mu}/\sqrt{m}$ and $\max_{i,j} |V_{i,j}| \leq \sqrt{\mu}/\sqrt{n}$.*

Incoherent matrices now satisfy a modified RIP.

Theorem 4.2 (Theorem 2.2 of [MJD10]) *There exists a constant $C > 0$ such that the following holds for all $0 < \delta_k < 1, \mu \geq 0, n \geq m \geq 3$. Let Ω be sampled according to the Bernoulli model with each pair $(i, j) \in \Omega$ chosen independently with probability $p \geq C\mu^2k^2 \log n/\delta_k^2 m$. Then with probability at least $1 - \exp(n \log n)$, the following restricted isometry property holds for all μ-incoherent matrices $X \in \mathbb{R}^{m\times n}$ of rank at most k:*

$$(1 - \delta_k)p\|X\|_F^2 \leq \|\,\mathrm{P}_\Omega(X)\|_F^2 \leq (1 + \delta_k)p\|X\|_F^2. \tag{37}$$

Theorem 4.2 only deals with sampling sets constructed by the Bernouilli model. Furthermore, the constant C is unknown which makes it impossible to bound the necessary sampling set a priori. Since it is not the purpose of this paper to analyze the convergence properties of Algorithm 1 for various, rather theoretical random models, we will not rely on this theorem to construct the matrix completion problem in the numerical experiments later on. However, in order to perform any analysis we have to make some assumptions on the data. We therefore assume that (37) holds for all matrices that appear in the course of our algorithm, even if the random sampling sets Ω do not satisfy Theorem 4.2. A similar approach was also taken in [MJD10] and it seems to be supported by our numerical experiments as well.

Assumption 1 *The map P_Ω defined in* (3) *satisfies the modified RIP* (37) *for the unknown matrix A in* (3) *and for all iterates X_i of Algorithm 1.*

Returning back to the original question of determining the size of the error $X - A$ based on the knowledge of $\|\,\mathrm{P}_\Omega(X - A)\|_{\mathrm{F}}$, the modified RIP gives immediately that, for rank-k matrices A and X, we have

$$\|X - A\|_{\mathrm{F}} \leq \frac{\|\,\mathrm{P}_\Omega(X - A)\|_{\mathrm{F}}}{\sqrt{(1 - \delta_{2k})p}}.$$

4.2 Convergence to critical points

For establishing the convergence of Algorithm 1, we rely on the general convergence theory for Riemannian optimization in [AMS08]. In particular, since R_X is a smooth retraction, the Armijo-type line search together with the search directions of non-linear CG, allows us to conclude that any limit point of Algorithm 1 is a critical point.

Proposition 4.3 *Let X_i be an infinite sequence of iterates generated by Algorithm 1. Then, every accumulation point X_* of X_i satisfies $\mathrm{P}_{T_{X_*}\mathcal{M}_k}\mathrm{P}_\Omega(X_*) = \mathrm{P}_{T_{X_*}\mathcal{M}_k}\mathrm{P}_\Omega(A)$.*

Proof. From Theorem 4.3.1 in [AMS08], every accumulation is a a critical point of the cost function f of Algorithm 1. These critical points X_* are determined by $\operatorname{grad} f(X) = 0$. By (14) this gives $\mathrm{P}_{T_{X_*}\mathcal{M}_k}\mathrm{P}_\Omega(X_* - A) = 0$. □

Since $\mathcal{M}_k$ is open, it is however not guaranteed that any infinite sequence has a limit point in $\mathcal{M}_k$. To guarantee that there exists at least one limit point in the sequence of iterates of Algorithm 1, we want to exploit that the iterates stay in a compact subset of $\mathcal{M}_k$.

It does not seem possible to show this for the objective function f of Algorithm 1 without modifying the algorithm or making further assumptions. We therefore show that this property holds when Algorithm 1 optimizes a regularized version of f, namely

$$g : \mathcal{M} \to \mathbb{R}, X \mapsto f(X) + \mu\|X^\dagger\|_{\mathrm{F}}^2, \quad \mu > 0.$$

Since every $X \in \mathcal{M}_k$ is of rank k, the pseudo-inverse is smooth on $\mathcal{M}_k$ and hence $g(X)$ is also smooth. In theory, μ can be very small, so one can choose it arbitrarily small, for example, as small as $\epsilon_{\text{mach}} \simeq 10^{-16}$. This means that as long as $\sigma_1(X_i) = O(1)$ and $\sigma_k(X_i) > \epsilon_{\text{mach}}$, the regularization has no effect numerically and one might as well have optimized the original objective function f, or g with $\mu = 0$. This is what we observe in practice when $\operatorname{rank}(A) \geq k$. In case $\operatorname{rank}(A) < k$, it is obvious that now $\sigma_k(X_i) \to 0$ as $i \to \infty$. Theoretically one can still optimize g, although in practice it makes more sense to transfer the original optimization problem to the manifold of rank $k-1$ matrices. Since these rank drops do not occur in our experiments, proving convergence of such a method is beyond the scope of this paper. A similar argument appears in the analysis of other methods for rank constrained optimization too. For example, the proof of convergence in [BM05] for SDPLR [BM03] uses a regularization by $\mu \det(X)$, yet in practice, ones optimizes without this regularization using the observation that μ can be made arbitrarily small and in practice one always observes convergence.

Proposition 4.4 *Let X_i be an infinite sequence of iterates generated by Algorithm 1 but with the objective function $g(X) = f(X) + \mu\|X^\dagger\|_F^2$, for some $\mu > 0$. Assume that according Assumption 1 the modified RIP* (37) *holds with constant $\delta_{2k} < 1$. Then, $\lim_{i\to\infty} \mathrm{P}_{T_{X_i}\mathcal{M}_k}\mathrm{P}_\Omega(X_i + \mu X_i^\dagger) = \mathrm{P}_{T_{X_\infty}\mathcal{M}_k}\mathrm{P}_\Omega(A)$.*

Proof. We will first show that the iterates stay in a closed and bounded subset of $\mathcal{M}_k$. Let $\mathcal{L} = \{X \in \mathcal{M}_k : g(X) \leq g(X_0)\}$ be the level set at X_0. By construction of the line search, all X_i stay inside $\mathcal{L}$ and we get

$$\|\mathrm{P}_\Omega(X_i - A)\|_{\mathrm{F}}^2 \leq C_0^2 - \mu\|X_i^\dagger\|_{\mathrm{F}}^2 \leq C_0^2$$

where $C_0^2 := g(X_0)$. Together with the RIP (37), this implies

$$\|X_i - A\|_{\mathrm{F}} \leq C_0/\sqrt{p(1-\delta_{2k})},$$

and we obtain an upper bound on the largest singular value of each X_i:

$$\sigma_1(X_i) \leq \|X_i\|_{\mathrm{F}} \leq \|X_i - A\|_{\mathrm{F}} + \|A\|_{\mathrm{F}} \leq C_0/\sqrt{p(1-\delta_{2k})} + \|A\|_{\mathrm{F}} =: C^\sigma.$$

Similarly, we get a lower bound on the smallest singular value:

$$\mu\|X_i^\dagger\|_{\mathrm{F}}^2 = \mu \sum_{j=1}^{k} \frac{1}{\sigma_j(X_i)^2} \leq C_0^2 - \|\,\mathrm{P}_\Omega(X_i - A)\|_{\mathrm{F}}^2 \leq C_0^2,$$

which implies that

$$\sigma_k(X_i) \geq \sqrt{\mu}/C_0 =: C_\sigma.$$

It is clear that all X_i stay inside the set

$$\mathcal{B} = \{X \in \mathcal{M}_k : \sigma_1(X) \leq C^\sigma,\ \sigma_k(X) \geq C_\sigma\}.$$

This set is closed and bounded, hence compact.

Now suppose that $\lim_{i\to\infty} \|\operatorname{grad} g(X_i)\|_{\mathrm{F}} \neq 0$. Then there is a subsequence in $\{X_i\}_{i\in\mathcal{K}}$ such $\|\operatorname{grad} g(X)\|_{\mathrm{F}} \geq \epsilon > 0$ for all $i \in \mathcal{K}$. Since $X_i \in \mathcal{B}$, this subsequence $\{X_i\}_{i\in\mathcal{K}}$ has a limit point X_* in $\mathcal{B}$. By continuity of $\operatorname{grad} g$, this implies that $\|\operatorname{grad} g(X_*)\|_{\mathrm{F}} \geq \epsilon$ which contradicts Theorem 4.3.1 in [AMS08] that every accumulation point is a critical point of g. Hence, $\lim_{i\to\infty} \|\operatorname{grad} g(X_i)\|_{\mathrm{F}} = 0$. □

5 Numerical experiments

The implementation of LRGeomCG was done in Matlab R2008 on a desktop computer with a 2.67 GHz CPU and 8 GB of memory. All reported times are wall-clock time that include the setup phase of the solvers (if needed) but exclude setting up the (random) problem. For simplicity, we will only deal with square matrices, i.e., $m = n$.

In the first set of experiments, we will complete random low-rank matrices which are generated as proposed in [CCS10]. First, generate two random matrices $A_L, A_R \in \mathbb{R}^{n\times k}$ with i.i.d. standard Gaussian entries; then, assemble $A := A_L A_R^T$; finally, the set of observations Ω is sampled uniformly at random among all sets of cardinality $|\Omega|$. The resulting observed matrix to complete is now $A_\Omega := \mathrm{P}_\Omega(A)$. Standard random matrix theory asserts that $\|A\|_{\mathrm{F}} \simeq n\sqrt{k}$ and $\|A_\Omega\|_{\mathrm{F}} \simeq \sqrt{|\Omega|k}$. In the later experiments, the test matrices are different and their construction will be explained there.

When we report on the relative error of an approximation X, we mean the error

$$\text{relative error} = \|X - A\|_{\mathrm{F}}/\|A\|_{\mathrm{F}} \tag{38}$$

computed on all the entries of A. On the other hand, the algorithms will compute a relative residual on Ω only:

$$\text{relative residual} = \|\,\mathrm{P}_\Omega(X - A)\|_{\mathrm{F}}/\|\,\mathrm{P}_\Omega(A)\|_{\mathrm{F}}. \tag{39}$$

Unless stated otherwise, we set as tolerance for the solvers a relative residual of 10^{-12}. Such a high precision is necessary to study the asymptotic convergence rate of the solvers, but where appropriate we will draw conclusions for moderate precisions too.

As initial guess X_1 for Algorithm 1, we construct a random rank-k matrix following the same procedure as above for M. Such a matrix is incoherent such that Assumption 1 is guaranteed for X_1.

The oversampling factor (OS) for a rank-k matrix is defined as the ratio of the number of samples to the degrees of freedom in a non-symmetric matrix of rank k,

$$\text{OS} = |\Omega|/(k(2n-k)). \tag{40}$$

Obviously one needs at least $\text{OS} \geq 1$ to uniquely complete any rank-k matrix. In the experiments below, we observe that, regardless of the method, $\text{OS} > 2$ is needed to reliably recover an incoherent matrix of rank k after uniform sampling. In addition, each row and each column needs to be sampled at least once. By the coupon's collector problem, this requires that $|\Omega| > Cn\log(n)$ but in all experiments below, Ω is always sufficiently large such that each row and column is sampled at least once. Hence, we will only focus on the quantity OS.

In our numerical experiments below we only compare with LMAFit of [WYZ10]. Extensive comparison with other approaches based in [CCS10], [MGC11], [MJD10], [GM11], [LCWM09], [TY10], [LST10], [CO10] revealed that LMAFit is overall the fastest method. We refer to [Mic11] for these results.

5.1 Influence of size and rank for fixed oversampling

As first test, we complete random matrices A of exact rank k with LRGeomCG and LMAFit and we explicitly exploit the knowledge of the rank in the algorithms. The purpose of this test is to investigate the dependence of the algorithms on the size and the rank of the matrices. In Section 5.3, we will study the influence of OS, but for now we fix the oversampling to $\text{OS} = 3$. In Table 1, we report on the mean run time and the number of iterations taking over 10 random instances. The run time and the iteration count of all these instances are visualized in Figures 3 and 4, respectively.

Overall, LRGeomCG needs less iterations than LMAFit for all problems. In Figure 3, we see that this is because LRGeomCG convergences faster asymptotically although the convergence of LRGeomCG exhibits a slower transient behavior in the beginning. This phase is, however, only limited to the first 20 iterations. Since the cost per iteration for LMAFit is cheaper than for LRGeomCG (see the estimates (34) and (35)), this transient phase leads to a trade-off between runtime and precision. This is clearly visible in the timings of Figure 4: when sufficiently high accuracy is requested, LRGeomCG will always be faster, whereas for low precision, LMAFit is faster.

Let us now check in more detail the dependence on the size n. It is clear from the left sides of Table 1 and Figure 3 that the iteration count of LMAFit grows quickly with n, whereas LRGeomCG shows much less growth. Even though each iteration of LRGeomCG is about 2.3 times more expensive than one iteration of LMAFit, LRGeomCG will eventually always be faster than LMAFit for any precision as long as n is sufficiently large. In Figure 4 on the left, one can, for example, see that this trade-off point happens at lower precision as n grows.

For growing rank k, the conclusion of the same analysis is different. Now, the ratio of the number of iterations between LMAFit and LRGeomCG goes to three as k grows. As we will see next, the amount of oversampling OS determines this ratio. But for difficult problems ($\text{OS} = 3$ is near the lower limit of 2), we can already observe that LRGeomCG is slightly faster than LMAFit.

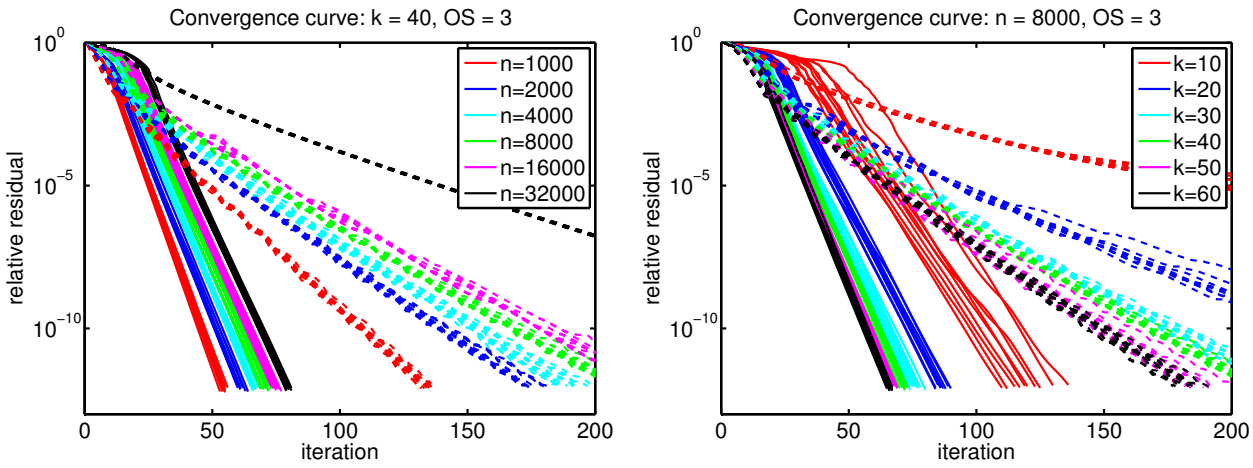

Figure 3: Convergence curves for LMAFit (dashed line) and LRGeomCg (full line) for fixed oversampling of OS = 3. Left: variable size n and fixed rank $k = 40$; right: variable ranks k and fixed size $n = 8000$.

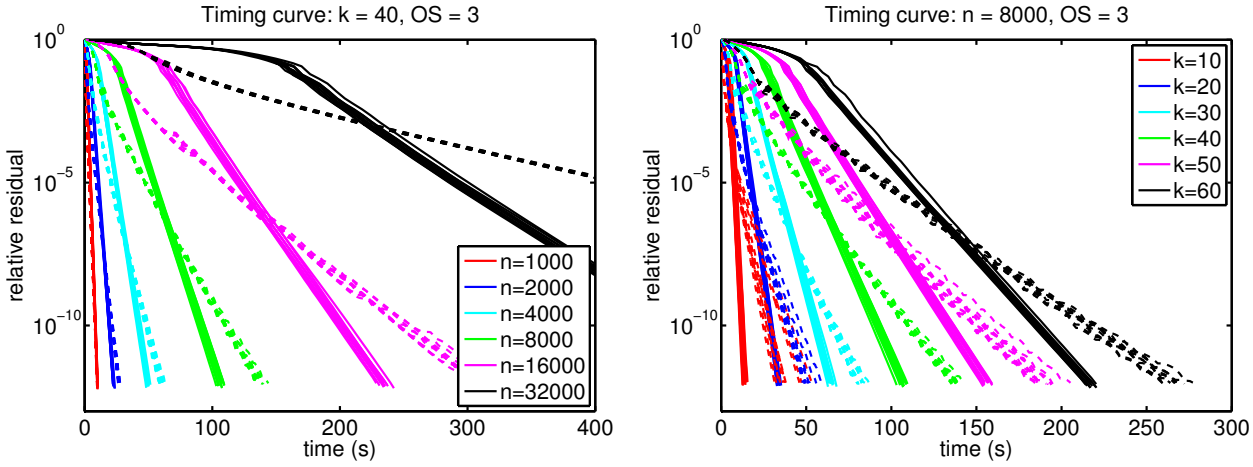

Figure 4: Timing curves for LMAFit (dashed line) and LRGeomCg (full line) for fixed oversampling of OS = 3. Left: variable size n and fixed rank $k = 40$; right: variable ranks k and fixed size $n = 8000$.

Table 1: The mean of the computational results for Figure 3–4 for solving with a tolerance of 10^{-12}.

Fixed rank $k = 40$					Fixed size $n = 8000$				
n	LMAFit		LRGeomCG		k	LMAFit		LRGeomCG	
	time(s.)	#its.	time(s.)	#its.		time(s.)	#its.	time(s.)	#its.
1000	9.93	133	10.8	54.5	10	39.1	816	13.8	121
2000	26.9	175	22.7	61.3	20	48.7	289	33.8	86.5
4000	60.2	191	49.7	66.7	30	81.7	221	64.6	76.1
8000	139	211	107	71.7	40	138	211	107	71.7
16000	312	226	235	74.9	50	191	188	156	67.7
32000	1220	404	542	79.7	60	267	182	218	66.1

5.2 Hybrid strategy

The experimental results from above clearly show that the transient behavior of LRGeomCG's convergence is detrimental if only modest accuracy is required. The reason for this slow phase is the lack of non-linear CG acceleration (β in Algorithm 4 is almost always zero) which essentially reduces LRGeomCG to a steepest descent algorithm. On the other hand, LMAFit is much less afflicted by slow convergence during the initial phase.

A simple heuristic to overcome this bad phase is a hybrid solver: for the first I iterations, we use LMAFit; after that, we hope that the non-linear CG acceleration kicks in immediately and we can efficiently solve with LRGeomCG. In Figure 5, we have tested this strategy for different values of I on a problem of the previous section with $n = 16000$ and $k = 40$.

From the left panel of Figure 5, we get that the wall-clock time to solve for a tolerance of 10^{-12} is reduced from 230 sec. (LRGeomCG) and 306 sec. (LMAFit) to about 190 sec. for the choice $I = 10, 20, 30, 40$. Performing only one iteration of LMAFit is less effective. For $I = 20$, the hybrid strategy has almost no transient behavior anymore and it is always faster than LRGeomCG or LMAFit alone. Furthermore, the choice of I is not very sensitive to the performance as long as it is between 10 and 40, and already for $I = 10$, there is a significant speedup and of LRGeomCG noticeable for all tolerances.

The quantity β_i of PR+ in Algorithm 4 is plotted in the right panel of Figure 5. Until the 20th iteration, plain LRGeomCG shows no meaningful CG acceleration. For the hybrid strategies with $I > 10$, the acceleration kicks in almost immediately. Observe that all approaches converge to $\beta \simeq 0.4$.

While this experiment shows that there is potential to speed up LRGeomCG in the early phase, we do not wish to claim that the present strategy is very robust nor directly applicable. The main point that we wish to make, however, is that the slow phase can be avoided in a relatively straightforward way, and that LRGeomCG can be warm started by any other method. In particular, this shows that LRGeomCG can be efficiently employed once the active manifold is identified by some other method (in these experiments, LMAFit). As explained in the introduction, there are numerous other methods for low-rank matrix completion that can reliably identity the active manifold, but have a slow asymptotic convergence. Combining them with LRGeomCG seems like a promising way forward.

5.3 Influence of oversampling

Next, we investigate the influence of oversampling on the convergence speed. We took 10 random problems with fixed rank and size, but now we vary the oversampling factor OS $= 1, \ldots, 15$. In Figure 6 on the left, the asymptotic convergence factor ρ is visible for LMAFit and LRGeomCG for three different combinations of the size and rank. Since the convergence is linear and we want to filter out any transient behavior, this factor was computed as

$$\rho = \left(\frac{\| \mathrm{P}_\Omega(X_{10} - A)\|_\mathrm{F}}{\| \mathrm{P}_\Omega(X_{i_\mathrm{end}} - A)\|_\mathrm{F}} \right)^{1/(i_\mathrm{end}-10)}$$

where i_end indicates the last iteration. A factor of one indicates failure to converge within 3000 steps. One can observe that both methods become slower as OS $\to 2$. Further, the convergence factor of LMAFit seems to stagnate for large values of OS while LRGeomCG's becomes smaller.

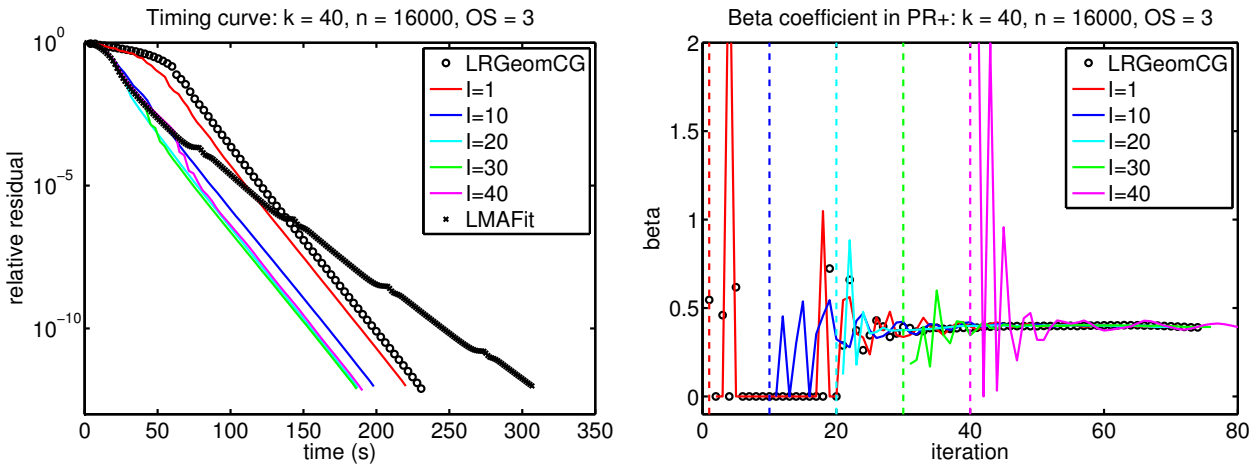

Figure 5: Timing curve and β coefficient of the hybrid strategy for different values of I.

Since for growing OS the completion problems become easier as more entries are available, only LRGeomCG shows the expected behavior.

In the right panel of Figure 6, the mean time to decrease the relative residual by a factor of 10 is displayed. These timings were again determined by neglecting the first 10 iterations and then interpolating the time needed for a reduction of the residual by 10. Similarly as before, we can observe that since the cost per iteration is cheaper for LMAFit, there is a range for OS around $6 \ldots 8$ where LMAFit and LRGeomCG are comparably fast. However, for smaller and larger values of OS, LRGeomCG is always faster by a significant margin (observe the logarithmic scale).

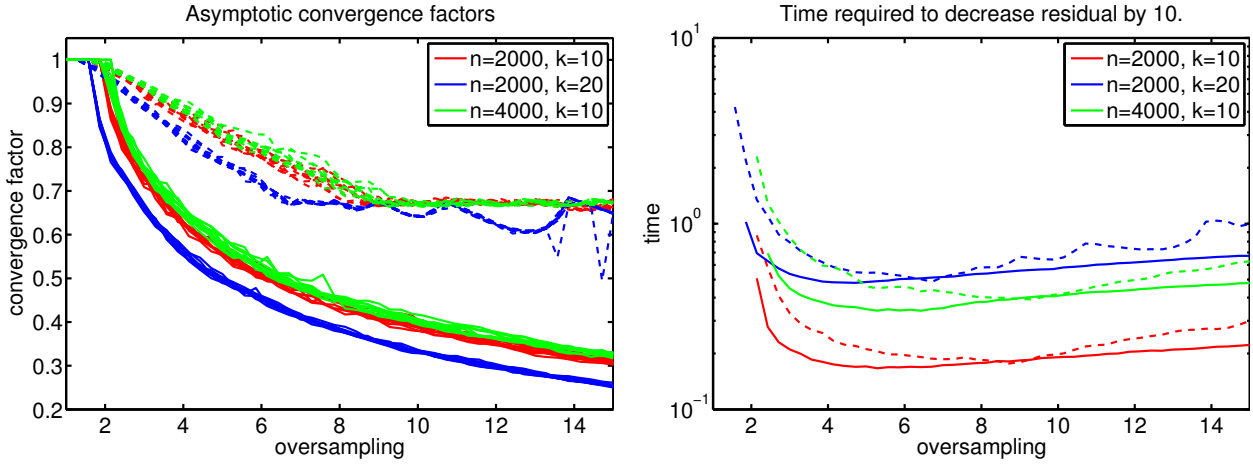

Figure 6: Asymptotic convergence factor ρ (left) and mean time to decrease the error by a factor of 10 (right) for LMAFit (dashed line) and LRGeomCG (full line) in function of the oversampling rate $\mathrm{OS} = 3$.

5.4 Influence of noise

Next, we investigate the influence of noise by adding random perturbations to the rank k matrix A. We define the noisy matrix $A^{(\epsilon)}$ with noise level ϵ as

$$A^{(\epsilon)} := A + \epsilon \frac{\|A_\Omega\|_{\mathrm{F}}}{\|N_\Omega\|_{\mathrm{F}}} N,$$

where N is a standard Gaussian matrix and Ω is the usual random sampling set (see also [TY10, WYZ10] for a similar setup). The reason for defining $A^{(\epsilon)}$ in this way is that we have

$$\|\,\mathrm{P}_\Omega(A - A^{(\epsilon)})\|_{\mathrm{F}} = \epsilon \frac{\|A_\Omega\|_{\mathrm{F}}}{\|N_\Omega\|_{\mathrm{F}}} \|\,\mathrm{P}_\Omega(N)\|_{\mathrm{F}} \simeq \epsilon \sqrt{|\Omega| k}.$$

So, the best relative residual we may expect from an approximation $X^{\mathrm{opt}} \simeq A$ is

$$\|\,\mathrm{P}_\Omega(X^{\mathrm{opt}} - A^{(\epsilon)})\|_{\mathrm{F}} / \|\,\mathrm{P}_\Omega(A)\|_{\mathrm{F}} \simeq \|\,\mathrm{P}_\Omega(A - A^{(\epsilon)})\|_{\mathrm{F}} / \|\,\mathrm{P}_\Omega(A)\|_{\mathrm{F}} \simeq \epsilon. \tag{41}$$

Similarly, the best relative error of X^{opt} should be on the order of the noise ratio too since

$$\|A - A^{(\epsilon)}\|_{\mathrm{F}} / \|A\|_{\mathrm{F}} \simeq \epsilon. \tag{42}$$

Due to the presence of noise, the relative residual (38) cannot go to zero but the Riemannian gradient of our optimization problem can. In principle, this suffices to detect convergence of LR-GeomCG. In practice, however, we have noticed that this wastes a lot of iterations in case the iteration stagnates. A simply remedy is to monitor

$$\text{relative change gradient at step } i = \left|1 - \sqrt{f(X_i)/f(X_{i-1})}\right| \tag{43}$$

and stop the iteration when this value drops below a certain threshold. After some experimentation, we fixed this threshold to 10^{-3}. Such a stagnation detection is applied by most other low-rank completion solvers [TY10, WYZ10], although its specific form varies. Our choice coincides with the one from [WYZ10], but we have lowered the threshold.

After equipping LMAFit and LRGeomCG with this stagnation detection, we can display the experimental results on Figure 7 and Table 2. We have only reported on one choice for the rank, size and oversampling since the same conclusions can be reached for any other choice. Based on Figure 7, it is clear that the stagnation is effectively detected by both methods, except for the very noisy case of $\epsilon = 1$. (Recall that we changed the original stagnation detection procedure in LMARank to ours of (43), to be able to draw a fair comparison.)

Comparing the results with those of the previous section, the only difference is that the iteration stagnates when the error reaches the noise level. In particular, the iterations are undisturbed by the presence of the noise up until the very last iterations. In Table 2, one can observe that both methods solve all problems up to the noise level, which is the best that can be expected from (41) and (42). Again, LRGeomCG is faster when a higher accuracy is required, which, in this case, corresponds to small noise levels. Since the iteration is unaffected by the noise until the very end, the hybrid strategy of the previous section should be effective here too.

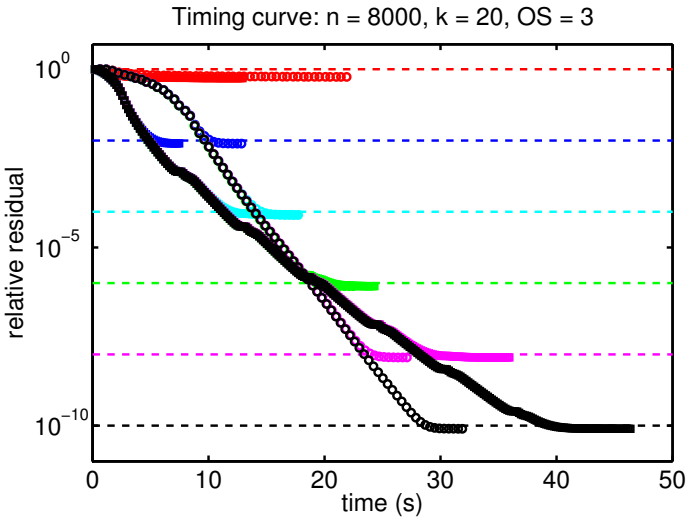

Figure 7: Timing curves for LMAFit (squares) and LRGeomCG (circles) for different noise levels with size $n = 8000$, rank $k = 20$, fixed oversampling of OS $= 3$. The noise levels ϵ are indicated by dashed lines in different color.

Table 2: Computational results for Figure 7.

ϵ	LMAFit				LRGeomCG			
	time(s.)	#its.	error	residual	time(s.)	#its.	error	residual
10^{-0}	13	76	$0.99\,\epsilon$	$0.56\,\epsilon$	22	46	$1.38\,\epsilon$	$0.61\,\epsilon$
10^{-2}	7.4	44	$0.76\,\epsilon$	$0.83\,\epsilon$	13	28	$0.72\,\epsilon$	$0.82\,\epsilon$
10^{-4}	18	104	$0.72\,\epsilon$	$0.82\,\epsilon$	18	40	$0.72\,\epsilon$	$0.82\,\epsilon$
10^{-6}	24	144	$0.73\,\epsilon$	$0.82\,\epsilon$	22	52	$0.72\,\epsilon$	$0.82\,\epsilon$
10^{-8}	36	212	$0.72\,\epsilon$	$0.82\,\epsilon$	27	63	$0.72\,\epsilon$	$0.82\,\epsilon$
10^{-10}	46	275	$0.72\,\epsilon$	$0.82\,\epsilon$	32	75	$0.72\,\epsilon$	$0.82\,\epsilon$

5.5 Exponentially decaying singular values

In the previous problems, the rank of the matrices could be unambiguously defined. Even in the noisy case, there was still a sufficient gap between the originally nonzero singular values and the ones affected by noise. In this section, we will complete a matrix for which the singular values decay, but do not become exactly zero.

Although a different setting than the previous problems, this type of matrices occurs frequently in the context of approximation of discretized functions or solutions of PDEs on tensorized domains; see, e.g., [KT10]. In this case, the motivation of approximating discretized functions by low-rank matrices is primarily motivated by reducing the amount of data to store. On a continuos level, low-rank approximation corresponds in this setting to approximation by sums of separable functions. As such, one is typically interested in approximation the matrix up to a certain tolerance using the lowest rank possible.

The (exact) rank of a matrix is now better replaced by the numerical rank, or ϵ-rank [GL96, Chapter 2.5.5]. Dependent on the required accuracy ϵ, the ϵ-rank is the quantity

$$k_\epsilon(A) = \min_{\|A-B\|_2 \leq \epsilon} \operatorname{rank}(B). \tag{44}$$

Obviously, when $\epsilon = 0$, one recovers the (exact) rank of a matrix. It is well known that the ϵ-rank of $A \in \mathbb{R}^{m\times n}$ can be determined from the singular values σ_i of A as follows

$$\sigma_1 \geq \sigma_{k_\epsilon} > \epsilon \geq \sigma_{k_\epsilon+1} \geq \cdots \geq \sigma_p, \quad p = \min(m, n).$$

In the context of the approximation of bivariate functions, the relation (44) is very useful in the following way. Let $f(x, y)$ be a function defined on a tensorized domain $(x, y) \in \mathcal{X} \times \mathcal{Y}$. After discretization of x and y, we arrive at a matrix A. If we allow for an absolute error in the spectral norm of the size ϵ, then (44) tells us that we can approximate M by a rank k_ϵ matrix. As example, we take the following simple bivariate function to complete:

$$f(x, y) = \frac{1}{\sigma + \|x - y\|_2^2}, \tag{45}$$

for some $\sigma > 0$. The parameter σ controls the decay of the singular values. Such functions occur as two-point correlations related to second-order elliptic problems with stochastic source terms; see [HPS10].

We ran LRGeomCG and LMAFit after discretization of (45) on a matrix of size $n = 8000$, with an oversampling factor of 3 and 11, and a decay of $\sigma = 1$ and $\sigma = 0.5$. The iteration was stopped when the norm of the Riemannian gradient was below 10^{-12} (for LRGeomCG) or after 2000 iterations (for both methods). In Figure 8, the final accuracy for different choices of the rank is visible for these four problem sets.

One can clearly see that although the accuracy grows when the rank increases (as it should be), both methods cannot achieve very high accuracy. Nevertheless, LRGeomCG is much more accurate than LMAFit since it reaches an accuracy in the order of 10^{-6}; on the other hand, LMAFit cannot go below 10^{-2} which is most likely not accurate enough for applications.

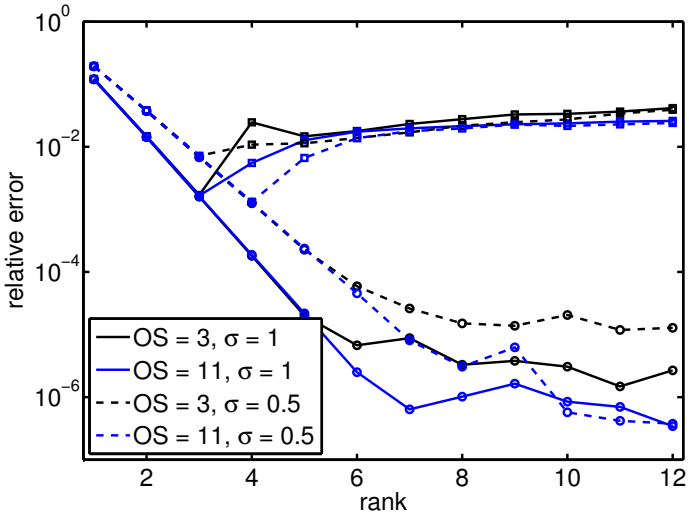

Figure 8: Relative error of LMAFit (squares) and LRGeomCG (circles) after completion of a discretization of (45) in function of the rank for two oversampling (OS) factors and two values for the decay σ of the singular values.

6 Conclusions

The matrix completion consists of recovering a low-rank matrix based on a very sparse set of entries of this matrix. In this paper, we have presented a new method based on optimization on manifolds to solve large-scale matrix completion problems. Compared to most other existing methods in the literature, our approach consists of directly minimizing the least-square error of the fit directly over $\mathcal{M}_k$, the space of matrices of rank k.

The main contribution of this paper is to show that the lack of vector space structure of $\mathcal{M}_k$ does not need be an issue when optimizing over this set, and illustrating this for low-rank matrix completion. Using the framework of retraction-based optimization, the necessary expressions were derived in order to minimize any smooth objection function over the Riemannian manifold $\mathcal{M}_k$. This included the construction of second-order models, suitable when applying Newton's method on the manifold. The geometry chosen for $\mathcal{M}_k$, namely a submanifold embedded in the space of matrices, allowed for an efficient implementation of non-linear CG for matrix completion. Indeed, the numerical experiments illustrated that this approach is very competitive with state-of-the-art solvers for matrix completion. In particular, the method achieved very fast asymptotic convergence factors without any tuning of parameters, thanks to a simple computation of the initial guess for the line search.

A drawback of the proposed approach is however that the rank of the manifold is fixed. Although there are several problems were choosing the rank is straightforward, for example in the context of manifold identification, an integrated manifold-based solution, like in [CAV11], is desirable. This is currently a topic of further research.

Acknowledgments

Parts of this paper has been prepared while the author was affiliated with the Seminar of Applied Mathematics, ETH Zürich. In addition, he gratefully acknowledges the helpful discussions with Daniel Kressner regarding low-rank matrix completion.

References

[ACM07] ACM SIGKDD and Netflix, editor. *Proceedings of KDD Cup and Workshop*, 2007.

[ADM+02] R. L. Adler, J.-P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub. Newton's method on Riemannian manifolds and a geometric model for the human spine. *IMA J. Numer. Anal.*, 22(3), 2002.

[AKW99] A. Y. Alfakih, A. Khandani, and H. Wolkowicz. Solving Euclidean distance matrix completion problems via semidefinite programming. *Comp. Optim. Appl.*, 12(1–3):13–30, 1999.

[AM10] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. Technical report, Department of Mathematical Engineering, Université catholique de Louvain, 2010.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[BM03] S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95(2):329–357, 2003.

[BM05] S. Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Math. Program.*, 103(3):427–444, 2005.

[BNR10] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *Proceedings of Allerto*, 2010.

[Bom67] J. Boman. Differentiability of a function and of its compositions with functions of one variable. *Math. Scand.*, 20:249—268, 1967.

[BV88] W. Bruns and U. Vetter. *Determinantal rings*, volume 1327 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1988.

[CAV11] T. P. Cason, P.-A. Absil, and P. Van Dooren. Iterative methods for low rank approximation of graph similarity matrices. *Lin. Alg. Appl.*, 2011.

[CCS10] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, 20(4):1956–1982, 2010.

[CD00] J. Chern and L. Dieci. Smoothness and periodicity of some matrix decompositions. *SIAM J. Matrix Anal. Appl.*, 22(3), 2000.

[CO10] J. Cai and S. Osher. Fast singular value thresholding without singular value decomposition. Technical report, Rice University, 2010.

[CP10] E. J. Candés and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[CR09] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.

[CT09] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory*, 56(5):2053–2080, 2009.

[DM10] W. Dai and O. Milenkovic. SET: an algorithm for consistent matrix completion. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[EAS99] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1999.

[GL96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in Mathematical Sciences, 3rd edition, 1996.

[GL11] D. F. Gleich and L.-H. Lim. Rank aggregation via nuclear norm minimization. *arxiv.org/abs/1102.4821*, 2011.

[GM11] D. Goldfarb and S. Ma. Convergence of fixed point continuation algorithms for matrix rank minimization. *Found. Comput. Math.*, 11(2):183–210, 2011.

[Har09] W. L. Hare. A proximal method for identifying active manifolds. *Comput. Optim. Appl.*, 43(3):295–306, 2009.

[HL07] W. L. Hare and A. S. Lewis. Identifying active manifolds. *Algorithmic Oper. Res.*, 2(2):75–82, 2007.

[HM94] U. Helmke and J. B. Moore. *Optimization and Dynamical Systems*. Springer-Verlag London Ltd., 1994.

[HPS10] H. Harbrecht, M. Peters, and R. Schneider. On the low-rank approximation by the pivoted Cholesky decomposition. Technical Report 2010-32, SimTech Cluster of Excellence, Universität Stuttgart, Germany, 2010.

[JBAS10] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM J. Optim.*, 20(5):2327–2351, 2010.

[KL07] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM. J. Matrix Anal.*, 29(2):434–454, 2007.

[KMO10a] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *JMLR*, 11:2057–2078, 2010.

[KMO10b] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Trans. Inform. Theory*, 56(6):2980–2998, 2010.

[KT10] D. Kressner and C. Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM J. Matrix Anal. Appl.*, 31(4):1688–1714, 2010.

[Lar04] R. M. Larsen. PROPACK—software for large and sparse SVD calculations. `http://soi.stanford.edu/~rmunk/PROPACK`, 2004.

[LB10] K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Trans. Inform. Theory*, 56(9):4402–4416, 2010.

[LCWM09] Z. Lin, M. Chen, L. Wu, and Yi Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,. Technical Report UILU-ENG-09-2215, University of Illinois, Urbana, Department of Electrical and Computer Engineering, 2009.

[Lee03] John M. Lee. *Introduction to smooth manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2003.

[LM08] A. S. Lewis and J. Malick. Alternating projections on manifolds. *Math. Oper. Res.*, 33:216–234, 2008.

[LST10] Y.-J. Liu, D. Sun, and K.-C. Toh. An implementable proximal point algorithmic framework for nuclear norm minimization. Technical report, National University of Singapore, 2010.

[LW11] A. S. Lewis and S. J. Wright. Identifying activity. *SIAM J. Optim.*, 21(2):597–614, 2011.

[MBS11] G. Meyer, S. Bonnabel, and R. Sepulchre. Linear regression under fixed-rank constraints: a riemannian approach. In *Proc. of the 28th International Conference on Machine Learning (ICML2011), Bellevue (USA)*, 2011.

[MGC11] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Math. Progr.*, 128(1–2):321–353, 2011.

[Mic11] M. Michenková. Numerical algorithms for low-rank matrix completion problems. `http://www.math.ethz.ch/~kressner/students/michenkova.pdf`, 2011.

[MJD10] R. Meka, P. Jain, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, 2010.

[NW06] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Verlag,, second edition, 2006.

[OHM06] R. Orsi, U. Helmke, and J. B. Moore. A Newton-like method for solving rank constrained linear matrix inequalities. *Automatica*, 42(11):1875–1882, 2006.

[QGA10] C. Qi, K. A. Gallivan, and P.-A. Absil. Riemannian BFGS algorithm with applications. In *Recent Advances in Optimization and its Applications in Engineering*, 2010.

[RFP10] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[Smi94] S. T. Smith. Optimization techniques on Riemannian manifold. In *Hamiltonian and Gradient Flows, Algorithms and Control*, volume 3, pages 113—136. Amer. Math. Soc., Providence, RI, 1994.

[SWC10] U. Shalit, D. Weinshall, and G. Chechik. Online learning in the manifold of low-rank matrices. In *Neural Information Processing Systems (NIPS spotlight)*, 2010.

[TY10] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific J. Optimization*, (615–640), 2010.

[VV10] B. Vandereycken and S. Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 31(5):2553–2579, 2010.

[WYZ10] Z. Wen, W. Yin, and Yin Zhang. Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. Technical Report TR10-07, CAAM Rice, 2010.