LOW-RANK MATRIX COMPLETION VIA TRUST-REGIONS ON THE GRASSMANN MANIFOLD

NICOLAS BOUMAL* AND P.-A. ABSIL*

Abstract. We consider large matrices of low rank. We address the mathematical problem of recovering such matrices when most of the entries are unknown. We follow an approach that exploits the geometry of the low-rank constraint to recast the problem as an unconstrained optimization problem on the Grassmann manifold. We then apply first- and second-order Riemannian trust-region methods to solve it. The cost of each iteration is linear in the number of known entries. The proposed methods, RTRMC 1 and 2, outperform state-of-the-art algorithms on a wide range of problem instances. In particular, RTRMC performs very well on rectangular matrices and we note that second-order methods such as RTRMC 2 are well suited to solve badly conditioned or nonuniformly sampled matrix completion tasks.

This paper is an extension of a previously published conference paper [6]. It features more detailed mathematical developments and a number of new numerical experiments that help better outline the class of problems for which RTRMC competes with existing art.

Key words. Low-rank matrix completion, optimization on manifolds, Grassmann manifold, Riemannian trust-regions, second-order methods, fixed-rank geometry, RTRMC.

AMS subject classifications. 90C90, 53B21, 15A83

1. Introduction.

We address the problem of recovering a low-rank $m \times n$ matrix X of which a few entries are observed, possibly with noise. Throughout, we assume that $r = \operatorname{rank}(X) \ll m \leq n$ is known and note $\Omega \subset \{1 \dots m\} \times \{1 \dots n\}$ the set of indices of the observed entries of X, i.e., X_{ij} is known iff $(i, j) \in \Omega$.

It was shown in numerous publications referenced below that low-rank matrix completion is applicable to problems such as recommender systems for example. In this setting, the rows of the matrix may correspond to items and the columns may correspond to users. The known entries are the ratings given by users to some items. The aim is to predict the unobserved ratings to generate personalized recommendations. Such applications motivate the study of scalable algorithms given the size of practical instances (billions of entries to predict based on millions of observed entries). However, it is also clear by now, as exemplified by the winning entry of the Netflix prize [4], that low-rank matrix completion is not sufficient: effective recommendation systems require more application-specific insight than the sole, algorithmically motivated low-rank prior. This is why the focus of the present paper is the mathematical problem of low-rank matrix completion and not recommendation systems. We will expose the proposed algorithm and test it on synthetic data under different scenarios. Each scenario will challenge the proposed method and prior art on a different aspect of the problem, such as the size of the problem, the conditioning of the target matrix, the sampling process, etc.

1.1. Related work.

In the noiseless case, one could state the minimum rank matrix recovery problem as follows:

$$\min_{\hat{X} \in \mathbb{R}^{m \times n}} \operatorname{rank} \hat{X}, \text{ such that } \hat{X}_{ij} = X_{ij} \ \forall (i, j) \in \Omega. \tag{1.1}$$

This problem, however, is NP-hard [9]. A possible convex relaxation of (1.1) introduced by Candès and Recht [9] is to use the nuclear norm of $\hat{X}$ as objective function,

*ICTEAM Institute, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium. {nicolas.boumal, pa.absil}@uclouvain.be

i.e., the sum of its singular values, noted $\|\hat{X}\|_*$. The SVT method [8] attempts to solve such a convex problem using tools from compressed sensing. The ADMiRA method [18] does so using matching pursuit-like techniques.

Alternatively, one may minimize the discrepancy between $\hat{X}$ and X at entries Ω under the constraint that $\operatorname{rank}(\hat{X}) \leq r$ for some small constant r. Since any matrix $\hat{X}$ of rank at most r may be written in the form UW with $U \in \mathbb{R}^{m\times r}$ and $W \in \mathbb{R}^{r\times n}$, a reasonable formulation of the problem reads:

$$\min_{U\in\mathbb{R}^{m\times r}} \min_{W\in\mathbb{R}^{r\times n}} \sum_{(i,j)\in\Omega} \big((UW)_{ij} - X_{ij}\big)^2. \tag{1.2}$$

This is also NP-hard [13]. The LMaFit method [25] does a good job at solving this problem by alternatively fixing either of the variables and solving the resulting least-squares problem efficiently. The IRLS-M method [12] also proceeds by solving successive least-squares problems.

One drawback of the latter formulation is that the factorization of a matrix $\hat{X}$ into the product UW is not unique. Indeed, for any r-by-r invertible matrix M, we have $UW = (UM)(M^{-1}W)$. All the matrices UM share the same column space. Hence, the optimal value of the inner optimization problem in (1.2) is a function of $\operatorname{col}(U)$—the column space of U—rather than U specifically. Dai et al. [11, 10] exploit this to recast (1.2) on the Grassmann manifold $\operatorname{Gr}(m, r)$, i.e., the set of r-dimensional vector subspaces of $\mathbb{R}^m$ (see Section 2):

$$\min_{\mathscr{U}\in\operatorname{Gr}(m,r)} \min_{W\in\mathbb{R}^{r\times n}} \sum_{(i,j)\in\Omega} \big((UW)_{ij} - X_{ij}\big)^2, \tag{1.3}$$

where $U \in \mathbb{R}^{m\times r}$ is any matrix such that $\operatorname{col}(U) = \mathscr{U}$ and is often chosen to be orthonormal. Unfortunately, the objective function of the outer minimization in (1.3) may be discontinuous at points $\mathscr{U}$ for which the least-squares problem in W does not have a unique solution. Dai et al. proposed ingenious ways to deal with the discontinuity. Their focus, though, was on deriving theoretical performance guarantees rather than developing fast algorithms.

Keshavan et al. [14, 16] state the problem on the Grassmannian too, but propose to simultaneously optimize on the row and column spaces, yielding a smaller least-squares problem which is unlikely to not have a unique solution, resulting in a smooth objective function. In one of their recent papers [14], they solve:

$$\min_{\mathscr{U}\in\operatorname{Gr}(m,r),\mathscr{V}\in\operatorname{Gr}(n,r)} \min_{S\in\mathbb{R}^{r\times r}} \sum_{(i,j)\in\Omega} \big((USV^\top)_{ij} - X_{ij}\big)^2 + \lambda^2 \left\|USV^\top\right\|_{\mathrm{F}}^2, \tag{1.4}$$

where U and V are any orthonormal bases of $\mathscr{U}$ and $\mathscr{V}$, respectively, and λ is a regularization parameter. The authors propose an efficient SVD-based initial guess for $\mathscr{U}$ and $\mathscr{V}$ which they refine using a steepest descent method, along with strong theoretical guarantees.

Meyer et al. [20] proposed a Riemannian approach to linear regression on fixed-rank matrices. Their regression framework encompasses matrix completion problems. Likewise, Balzano et al. [3] introduced GROUSE for subspace identification on the Grassmannian, applicable to matrix completion. Finally, in [24], Vandereycken proposes an approach based on the submanifold geometry of the sets of fixed-rank matrices. Mishra et al. propose another geometric approach [21]. They address the problem

of low-rank trace norm minimization and propose an algorithm that alternates between fixed-rank optimization and rank-one updates, with applications to low-rank matrix completion. Tao and Yuan, among others, focus on an interesting variation on the theme of low-rank matrix completion, where one tries to identify a sum of low-rank and sparse matrices as the target matrix [22].

1.2. Our contribution and outline of the paper. Dai et al.'s initial formulation (1.3) has a discontinuous objective function on the Grassmannian. The OptSpace formulation (1.4) on the other hand has a continuous objective and comes with a smart initial guess, but optimizes on a higher-dimensional search space, while it is arguably preferable to keep the dimension of the manifold search space low, even at the expense of a larger least-squares problem. Furthermore, the OptSpace regularization term is efficiently computable since $\|USV^\top\|_{\mathrm{F}} = \|S\|_{\mathrm{F}}$, but it penalizes all entries instead of just the entries $(i,j) \notin \Omega$.

In an effort to combine the best of both worlds, we equip (1.3) with a regularization term weighted by $\lambda > 0$, which yields a smooth objective function defined over an appropriate search space:

$$\min_{\mathscr{U} \in \mathrm{Gr}(m,r)} \min_{W \in \mathbb{R}^{r\times n}} \frac{1}{2} \sum_{(i,j)\in\Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2 + \frac{\lambda^2}{2} \sum_{(i,j)\notin\Omega} (UW)_{ij}^2. \tag{1.5}$$

Here, we introduced a confidence index $C_{ij} > 0$ for each observation X_{ij}, which may be useful in applications. As we will see, introducing a regularization term is essential to ensure smoothness of the objective and hence obtain good convergence properties. It is not, however, critical for practical problem instances. As the solutions obtained are largely insensitive to the value of λ provided it is much smaller than the positive C_{ij}'s, in the numerical experiments, we set λ to a fixed value of 10^{-6}, with all positive C_{ij}'s set to 1.

We further innovate on previous work by using a Riemannian trust-region method, GenRTR, as optimization algorithm to minimize (1.5) on the Grassmannian [1]. GenRTR is readily available as a free Matlab package and comes with strong convergence results that are naturally inherited by the proposed algorithms.

In Section 2, we rapidly cover the essential useful tools on the Grassmann manifold. In Section 3, we derive expressions for the gradient and the Hessian of our objective function while paying special attention to complexity. Section 4 sums up the main properties of the Riemannian trust-region method. Section 5 shows a few results of numerical experiments demonstrating the effectiveness of the proposed approach.

2. Geometry of the Grassmann manifold. We tackle low-rank matrix completion as an optimization problem on the Grassmann manifold, hence our objective function (which we construct later on) f (3.15) is defined over said manifold $\mathrm{Gr}(m,r)$, i.e., the set of r-dimensional vector subspaces of $\mathbb{R}^m$. Absil et al. [2] give a computation-oriented description of the geometry of this manifold. Here, we only give a summary of the important tools we need. We assume some familiarity with standard differential geometric concepts. To acquire such familiarity if need be, the monographs [2, Ch. 3][5] are good starting points.

Each point $\mathscr{U} \in \mathrm{Gr}(m,r)$ is a vector subspace we may represent numerically as the column space of a full-rank matrix U:

$$\mathrm{Gr}(m,r) = \{\mathscr{U} = \mathrm{col}(U) : U \in \mathbb{R}_*^{m\times r}\}. \tag{2.1}$$

The notation $\mathbb{R}_*^{m\times r}$ stands for the set of full-rank $m \times r$ matrices. For numerical reasons, we will only use orthonormal matrices $U \in \mathrm{St}(m, r)$ to represent subspaces. The set $\mathrm{St}(m, r)$ is the (compact) Stiefel manifold:

$$\mathrm{St}(m, r) = \{U \in \mathbb{R}_*^{m\times r} : U^\top U = I_r\}. \tag{2.2}$$

We view $\mathrm{St}(m, r)$ as a Riemannian submanifold of the Euclidean space $\mathbb{R}^{m\times r}$, and we endow $\mathrm{Gr}(m, r)$ with the unique Riemannian metric such that $\mathrm{Gr}(m, r)$ is a Riemannian quotient manifold of $\mathrm{St}(m, r)$. Indeed, the mapping

$$\pi : \mathrm{St}(m, r) \to \mathrm{Gr}(m, r) : U \mapsto \pi(U) = \mathrm{col}(U) \tag{2.3}$$

is a Riemannian submersion. This Riemannian metric is the (essentially) unique one that is invariant by rotation of $\mathbb{R}^m$ [19]. The submersion π induces an equivalence relation such that U and U' are equivalent if $\pi(U) = \pi(U')$, that is, if U and U' represent the same column space. Let

$$\mathrm{O}(r) = \{Q \in \mathbb{R}_*^{r\times r} : Q^\top Q = I_r\} \tag{2.4}$$

denote the set of $r \times r$ orthogonal matrices. Since U and U' are equivalent if and only if there exists some $Q \in \mathrm{O}(r)$ such that $U' = UQ$, we say that $\mathrm{Gr}(m, r)$ is a quotient of $\mathrm{St}(m, r)$ by the action of $\mathrm{O}(r)$:

$$\mathrm{Gr}(m, r) = \mathrm{St}(m, r)/\mathrm{O}(r). \tag{2.5}$$

The Grassmannian is a manifold, and as such we may define a tangent space to $\mathrm{Gr}(m, r)$ at each point $\mathscr{U}$, noted $\mathrm{T}_{\mathscr{U}}\mathrm{Gr}(m, r)$. The latter is a vector space of dimension $\dim \mathrm{Gr}(m, r) = r(m - r)$. A tangent vector $\mathscr{H} \in \mathrm{T}_{\mathscr{U}}\mathrm{Gr}(m, r)$, where we represent $\mathscr{U}$ by the orthonormal matrix U, is represented by a matrix $H \in \mathbb{R}^{m\times r}$ verifying $\frac{\mathrm{d}}{\mathrm{d}t}\,\mathrm{col}(U + tH)\big|_{t=0} = \mathscr{H}$. This representation H of $\mathscr{H}$ is one-to-one if we further impose $U^\top H = 0$. For practical purposes, we often refer to $\mathscr{U}$ and $\mathscr{H}$ using their matrix counterparts U and H instead. This slight abuse of notation we often commit hereafter has the benefit of making it clearer how one can numerically work with the abstract objects $\mathscr{U}$ and $\mathscr{H}$. In simplified notation then, we have that the tangent space to $\mathrm{Gr}(m, r)$ at U is the set:

$$\mathrm{T}_U\mathrm{Gr}(m, r) = \{H \in \mathbb{R}^{m\times r} : U^\top H = 0\}. \tag{2.6}$$

Each tangent space is endowed with an inner product (the Riemannian metric) that varies smoothly from point to point. It is inherited from the embedding space $\mathbb{R}^{m\times r}$ of the matrix representation of tangent vectors:

$$\forall H_1, H_2 \in \mathrm{T}_U\mathrm{Gr}(m, r), \quad \langle H_1, H_2\rangle_U = \mathrm{Trace}(H_1^\top H_2). \tag{2.7}$$

The orthogonal projector from $\mathbb{R}^{m\times r}$ onto the tangent space $\mathrm{T}_U\mathrm{Gr}(m, r)$ is given by:

$$\mathrm{P}_U : \mathbb{R}^{m\times r} \to \mathrm{T}_U\mathrm{Gr}(m, r) : H \mapsto \mathrm{P}_U H = (I - UU^\top)H. \tag{2.8}$$

One can similarly define the tangent space at U to the Stiefel manifold:

$$\mathrm{T}_U\mathrm{St}(m, r) = \{H \in \mathbb{R}^{m\times r} : U^\top H + H^\top U = 0\}. \tag{2.9}$$

The projector from the ambient space $\mathbb{R}^{m\times r}$ onto the tangent space of the Stiefel manifold is given by:

$$\mathrm{P}_U^{\mathrm{St}} : \mathbb{R}^{m\times r} \to \mathrm{T}_U\mathrm{St}(m,r) : H \mapsto \mathrm{P}_U^{\mathrm{St}} H = (I - UU^\top)H + U\left\lceil U^\top H \right\rfloor, \tag{2.10}$$

where $\lceil A \rfloor = (A - A^\top)/2$ extracts the skew-symmetric part of A.

We now concern ourselves with the differentiation of functions defined on the Grassmannian. Let $\bar{f}$ be a suitably smooth mapping from $\mathbb{R}_*^{m\times r}$ to $\mathbb{R}$. Let $\bar{f}\big|_{\mathrm{St}}$ denote its restriction to the Stiefel manifold and let us further assume that

$$\forall U \in \mathrm{St}(m,r), Q \in \mathrm{O}(r), \quad \bar{f}\big|_{\mathrm{St}}(U) = \bar{f}\big|_{\mathrm{St}}(UQ). \tag{2.11}$$

Under this assumption, $\bar{f}\big|_{\mathrm{St}}$ is only a function of the column space of its argument, hence

$$f : \mathrm{Gr}(m,r) \to \mathbb{R} : \mathrm{col}(U) \mapsto f(\mathrm{col}(U)) = \bar{f}\big|_{\mathrm{St}}(U) \tag{2.12}$$

is well defined. The gradient of f at U is the unique tangent vector grad $f(U)$ in $\mathrm{T}_U\mathrm{Gr}(m,r)$ satisfying

$$\forall H \in \mathrm{T}_U\mathrm{Gr}(m,r), \quad \langle \mathrm{grad}\ f(U), H\rangle_U = \mathrm{D}f(U)[H], \tag{2.13}$$

where $\mathrm{D}f(U)[H]$ is the directional derivative of f at U along H,

$$\mathrm{D}f(U)[H] = \lim_{t\to 0} \frac{f(\mathrm{col}(U+tH)) - f(\mathrm{col}(U))}{t}. \tag{2.14}$$

Observe that grad $f(U)$ is an abuse of notation. In fact, grad $f(U)$ is the so-called horizontal lift of grad $f(\mathscr{U})$ at U, and the way we abuse notations is justified by the theory of Riemannian submersions, see [2, § 3.6.2, § 5.3.4]. A similar definition holds for grad $\bar{f}$ (the usual gradient) and grad $\bar{f}\big|_{\mathrm{St}}$. Since $\mathrm{St}(m,r)$ is a Riemannian submanifold of $\mathbb{R}_*^{m\times r}$, the reference [2, eq. (3.37)] tells us that

$$\mathrm{grad}\ \bar{f}\big|_{\mathrm{St}}(U) = \mathrm{P}_U^{\mathrm{St}}\,\mathrm{grad}\ \bar{f}(U). \tag{2.15}$$

That is, the gradient of the restricted function is obtained by computing the gradient of $\bar{f}$ in the usual way, then projecting the resulting vector onto the tangent space to the Stiefel manifold. Furthermore, since $\mathrm{Gr}(m,r)$ is a Riemannian quotient manifold of $\mathrm{St}(m,r)$, the same reference [2, eq. (3.39)] tells us that

$$\mathrm{grad}\ f(U) = \mathrm{grad}\ \bar{f}\big|_{\mathrm{St}}(U). \tag{2.16}$$

The notation grad $f(U)$ denotes the matrix representation of the abstract tangent vector grad $f(\mathrm{col}(U))$ with respect to the (arbitrary) choice of orthonormal basis U. They are related by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\,\mathrm{col}(U + t\ \mathrm{grad}\ f(U))\bigg|_{t=0} = \mathrm{grad}\ f(\mathrm{col}(U)). \tag{2.17}$$

Notice that $\mathrm{T}_U\mathrm{Gr}(m,r)$ is a vector subspace of $\mathrm{T}_U\mathrm{St}(m,r)$, so that $\mathrm{P}_U \circ \mathrm{P}_U^{\mathrm{St}} = \mathrm{P}_U$. Since grad $f(U)$ belongs to $\mathrm{T}_U\mathrm{Gr}(m,r)$, it is invariant under P_U. Combining (2.15) and (2.16) and applying P_U on both sides, we finally obtain a practical means of computing the gradient of f:

$$\mathrm{grad}\ f(U) = \mathrm{P}_U\,\mathrm{grad}\ \bar{f}(U) = (I - UU^\top)\,\mathrm{grad}\ \bar{f}(U). \tag{2.18}$$

In practice, this means that we need only compute the gradient of $\bar{f}$ in the usual way and then project accordingly.

We now apply similar techniques to derive the Hessian of f at U along $H \in \mathrm{T}_U\mathrm{Gr}(m,r)$. Let us define the vector field $\bar{F} : \mathbb{R}_*^{m\times r} \to \mathbb{R}^{m\times r}$:

$$\bar{F}(U) = (I - UU^\top)\,\mathrm{grad}\ \bar{f}(U). \tag{2.19}$$

Notice that the restriction of $\bar{F}$ to the Stiefel manifold, $\bar{F}\big|_{\mathrm{St}}$, is a tangent vector field, i.e., $\bar{F}(U) \in \mathrm{T}_U\mathrm{St}(m,r)$ for all $U \in \mathrm{St}(m,r)$. Then, the reference [2, eq. (5.14)] tells us that for all H in $\mathrm{T}_U\mathrm{Gr}(m,r) \subset \mathrm{T}_U\mathrm{St}(m,r)$,

$$\overline{\nabla}_H \bar{F}\big|_{\mathrm{St}}(U) = \mathrm{P}_U^{\mathrm{St}}\mathrm{D}\bar{F}(U)[H], \tag{2.20}$$

where $\mathrm{D}\bar{F}(U)[H]$ is the usual directional derivative of $\bar{F}$ at U along H and $\overline{\nabla}_H$ denotes the Levi-Civita connection on the Stiefel manifold w.r.t. any smooth tangent vector field X such that $X_U = H$. This is the analog on manifolds of directional derivatives of vector-valued functions. Furthermore, the reference [2, eq. (5.18)] tells us that

$$\mathrm{Hess}\, f(U)[H] = \mathrm{P}_U\overline{\nabla}_H \bar{F}\big|_{\mathrm{St}}(U), \tag{2.21}$$

where $\mathrm{Hess}\, f(U)[H]$ is the derivative at U along H (w.r.t. the Levi-Civita connection on the Grassmannian) of the gradient vector field grad f. Putting these two statements together and remembering that $\mathrm{P}_U \circ \mathrm{P}_U^{\mathrm{St}} = \mathrm{P}_U$, we find a simple expression for the Hessian of f at $\mathrm{col}(U)$ along H w.r.t. the (arbitrary) choice of orthonormal basis U:

$$\mathrm{Hess}\, f(U)[H] = \mathrm{P}_U\mathrm{D}\bar{F}(U)[H] = (I - UU^\top)\mathrm{D}\bar{F}(U)[H]. \tag{2.22}$$

In practice then, we simply need to differentiate the expression for grad $f(U)$ "as if it were defined on $\mathbb{R}_*^{m\times r}$" and project accordingly.

For iterative optimization algorithms, it is important to be able to move along the manifold from some initial point U in some prescribed direction specified by a tangent vector H: this is indeed the basic operation required by all descent methods. The theoretically appealing way of doing this would be to follow the geodesic originating at U and moving in the direction prescribed by H. This is often numerically expensive or difficult to compute. Fortunately, cheaper ways of moving on a manifold, known as *retractions*, can be used instead of geodesics without compromising the convergence properties of many optimization algorithms [2]. We use the following retraction on $\mathrm{Gr}(m,r)$:

$$\mathrm{R}_U(H) = \mathrm{qf}(U + H), \tag{2.23}$$

where $\mathrm{qf}(A) \in \mathrm{St}(m,r)$ designates the m-by-r Q-factor of the QR decomposition of $A \in \mathbb{R}^{m\times r}$. In abstract terms, this corresponds to having $\mathrm{col}(\mathrm{R}_U(H)) = \mathrm{col}(U + H)$.

3. The objective function and its derivatives. We seek an m-by-n matrix $\hat{X}$ of rank at most r (and usually exactly r) such that $\hat{X}$ agrees as much as possible with a matrix X whose entries at the observation set Ω are given. Furthermore, we are given a weight matrix $C \in \mathbb{R}^{m\times n}$ indicating the confidence we have in each observed entry of X. The matrix C is positive at entries in Ω and zero elsewhere. To this end,

we propose to minimize the following cost function w.r.t. $U \in \mathbb{R}_*^{m\times r}$ and $W \in \mathbb{R}^{r\times n}$, where $(X_\Omega)_{ij}$ equals X_{ij} if $(i,j) \in \Omega$ and is zero otherwise:

$$g : \mathbb{R}_*^{m\times r} \times \mathbb{R}^{r\times n} \to \mathbb{R} : (U,W) \mapsto g(U,W) = \frac{1}{2}\left\|C \odot (UW - X_\Omega)\right\|_\Omega^2 + \frac{\lambda^2}{2}\left\|UW\right\|_{\Omega_c}^2. \tag{3.1}$$

We write $\odot$ for the entry-wise product, $\lambda > 0$ is a regularization parameter, Ω_c is the complement of the set Ω and

$$\|M\|_\Omega^2 \triangleq \sum_{(i,j)\in\Omega} M_{ij}^2. \tag{3.2}$$

The interpretation is as follows: we are looking for an optimal matrix $\hat{X} = UW$ of rank at most r; we have confidence C_{ij} that $\hat{X}_{ij}$ should equal X_{ij} for $(i,j) \in \Omega$ and (very small) confidence λ that $\hat{X}_{ij}$ should equal 0 for $(i,j) \notin \Omega$.

For a fixed U, computing the matrix W that minimizes (3.1) is a least-squares problem. As we shall see in Section 3.3, the solution to that problem exists and is unique since we assume $\lambda > 0$. Let us note $g_U(W) \triangleq g(U,W)$. The mapping between U and this unique optimal W,

$$W_U = W(U) = \operatorname*{argmin}_{W\in\mathbb{R}^{r\times n}} g_U(W), \tag{3.3}$$

is smooth and easily computable—see Section 3.3. It is thus natural to consider the following cost function defined over the set of full-rank matrices $U \in \mathbb{R}_*^{m\times r}$:

$$\hat{f} : \mathbb{R}_*^{m\times r} \to \mathbb{R} : U \mapsto \hat{f}(U) = \frac{1}{2}\left\|C \odot (UW_U - X_\Omega)\right\|_\Omega^2 + \frac{\lambda^2}{2}\left\|UW_U\right\|_{\Omega_c}^2. \tag{3.4}$$

Notice that W is now W_U, the best W matrix one can pick given a prescribed matrix U. Because W_U is unique, this function is well defined.

By virtue of the discussion in Section 1, we expect that the function $\hat{f}$ be constant over sets of full-rank matrices U spanning the same column space. Indeed, let

$$\mathrm{GL}(r) = \{M \in \mathbb{R}^{r\times r} : M \text{ is invertible}\} \tag{3.5}$$

denote the general linear group. We have the following property:

$$\forall M \in \mathrm{GL}(r), \quad W_{UM} = M^{-1}W_U. \tag{3.6}$$

Indeed, since $g(U,W)$ merely depends on the product UW, for any $M \in \mathrm{GL}(r)$ we have that $g_U(W)$ and $g_{UM}(M^{-1}W)$ are two identical functions of W. Hence, since W_U is the unique minimizer of g_U, it holds that $M^{-1}W_U$ is the unique minimizer of g_{UM}, i.e., $W_{UM} = M^{-1}W_U$. As a consequence, $\hat{X} = UW_U = (UM)W_{UM}$ for all $M \in \mathrm{GL}(r)$. For such matrices M, it then follows as expected that:

$$\hat{f}(UM) = \hat{f}(U). \tag{3.7}$$

This induces an equivalence relation $\sim$ over the $m \times r$ matrices of full rank, $\mathbb{R}_*^{m\times r}$. Two such matrices are equivalent if and only if they have the same column space:

$$U \sim U' \Leftrightarrow \exists M \in \mathrm{GL}(r) \text{ s.t. } U' = UM \Leftrightarrow \mathrm{col}(U) = \mathrm{col}(U'). \tag{3.8}$$

This is equivalent to stating that U and U' are equivalent if they lead to the same reconstruction model $\hat{X} = UW_U = U'W_{U'}$, which certainly makes sense for our purpose. For each $U \in \mathbb{R}_*^{m\times r}$, we write

$$[U] = \{UM : M \in \mathrm{GL}(r)\} = \{U' \in \mathbb{R}_*^{m\times r} : \mathrm{col}(U') = \mathrm{col}(U)\} \tag{3.9}$$

for the equivalence class of U, and identify it with $\mathrm{col}(U)$, the column space of U. The set of all such equivalence classes is the Grassmann manifold $\mathrm{Gr}(m, r)$: the set of r-dimensional vector subspaces embedded in $\mathbb{R}^m$—see Section 2. Under this description, the Grassmannian is seen as the quotient space $\mathbb{R}_*^{m\times r}/\mathrm{GL}(r)$, which is an alternative to the quotient structure $\mathrm{St}(m, r)/\mathrm{O}(r)$ (2.5) developed in Section 2.

Consequently, $\hat{f}$ descends to a well-defined function over the Grassmann manifold. Our task is to minimize this function. This will single out a column space $\mathrm{col}(U)$. We may then pick any basis of that column space, say U, and compute W_U. The product $\hat{X} = UW_U$ (which is invariant w.r.t. the choice of basis U of $\mathrm{col}(U)$) is then our completion of the matrix X. In the next section, we rearrange the terms in $\hat{f}$ to make it easier to compute and give a slightly modified definition of our objective function.

3.1. Rearranging the objective function. Considering (3.4), it looks like evaluating $\hat{f}(U)$ will require the computation of the product UW_U at the entries in Ω *and* Ω_c, i.e., we would need to compute the whole matrix UW_U, which cannot cost much less than $\mathcal{O}(mnr)$. Since applications typically involve very large values of the product mn, this is not acceptable. Fortunately, the regularization term $\|UW_U\|_{\Omega_c}^2$ can be computed cheaply based on the computations that need to be executed for the principal term. Indeed, observe that:

$$\|UW_U\|_\Omega^2 + \|UW_U\|_{\Omega_c}^2 = \|UW_U\|_{\mathrm{F}}^2 = \mathrm{Trace}(U^\top UW_U W_U^\top). \tag{3.10}$$

The right-most quantity is computable in $\mathcal{O}((m+n)r^2)$ flops and since $(UW_U)_\Omega$ has to be computed for the first term in the objective function, $\|UW_U\|_\Omega^2$ turns out to be cheap to obtain. As a result, we see that computing $\hat{f}(U)$ as a whole only requires the computation of $(UW_U)_\Omega$ as opposed to the whole product UW_U, conferring to $\hat{f}$ a computational cost that is linear in the number of observed entries $k = |\Omega|$.

We have the freedom to represent a column space with any of its bases. From a numerical standpoint, it is sound to restrict our attention to orthonormal bases. The set of orthonormal bases U is termed the Stiefel manifold (2.2):

$$\mathrm{St}(m, r) = \{U \in \mathbb{R}_*^{m\times r} : U^\top U = I_r\}. \tag{3.11}$$

It is related to the Grassmannian via the quotient $\mathrm{Gr}(m, r) = \mathrm{St}(m, r)/\mathrm{O}(r)$ (2.5), since for any orthogonal matrix $Q \in \mathrm{O}(r)$, we have $UQ \in \mathrm{St}(m, r)$ and $\mathrm{col}(UQ) = \mathrm{col}(U)$. Assuming $U \in \mathrm{St}(m, r)$, equation (3.10) yields a simple expression for the regularization term:

$$\|UW_U\|_{\Omega_c}^2 = \|W_U\|_{\mathrm{F}}^2 - \|UW_U\|_\Omega^2 . \tag{3.12}$$

Based on this observation, we introduce the following function over $\mathbb{R}_*^{m\times r}$:

$$\bar{f} : \mathbb{R}_*^{m\times r} \to \mathbb{R} : U \mapsto \bar{f}(U) = \frac{1}{2}\|C \odot (UW_U - X_\Omega)\|_\Omega^2 + \frac{\lambda^2}{2}\left(\|W_U\|_{\mathrm{F}}^2 - \|UW_U\|_\Omega^2\right). \tag{3.13}$$

In particular, it is the restriction of $\bar{f}$ to $\mathrm{St}(m,r) \subset \mathbb{R}_*^{m\times r}$ that makes sense for our problem:

$$\bar{f}|_{\mathrm{St}} : \mathrm{St}(m,r) \to \mathbb{R} : U \mapsto \bar{f}|_{\mathrm{St}}(U) = \bar{f}(U). \tag{3.14}$$

Notice that this restriction coincides with the original cost function: $\hat{f}|_{\mathrm{St}} \equiv \bar{f}|_{\mathrm{St}}$. We then define our objective function f over the Grassmannian:

$$f : \mathrm{Gr}(m,r) \to \mathbb{R} : \mathrm{col}(U) \mapsto f(\mathrm{col}(U)) = \bar{f}|_{\mathrm{St}}(U), \tag{3.15}$$

where U is any orthonormal basis of the column space $\mathrm{col}(U)$. This is well-defined since $\bar{f}|_{\mathrm{St}}(U) = \bar{f}|_{\mathrm{St}}(UQ)$ for all orthogonal Q. On the other hand, notice that $\bar{f}$ does *not* reduce to a function on the Grassmannian (it does not have the invariance property $\bar{f}(UM) = \bar{f}(U)\ \forall M \in \mathrm{GL}(r)$), which explains why we had to first go through the Stiefel manifold.

Computing $f(\mathrm{col}(U))$ only requires the computation of UW_U at entries in Ω, at a cost of $\mathcal{O}(kr)$ flops, where $k = |\Omega|$ is the number of known entries. Computing $\|W_U\|_{\mathrm{F}}^2$ costs $\mathcal{O}(nr)$ flops, hence a total evaluation cost of $\mathcal{O}((k+n)r)$ flops, to which we will add the (dominating) cost of computing W_U in Section 3.3 to obtain the total complexity of evaluating f.

3.2. Gradient and Hessian of the objective function. We now derive formulas for the first- and second-order derivatives of f (3.15). As outlined in Section 2, grad $f(\mathrm{col}(U))$ is a tangent vector to the quotient manifold $\mathrm{Gr}(m,r)$. Because of the abstract nature of quotient manifolds, this vector is an abstract object too. In practice, we represent it with a concrete matrix grad $f(U)$ w.r.t. an (arbitrary) basis $U \in \mathrm{St}(m,r)$ of $\mathrm{col}(U)$. Equation (2.17) establishes the link between grad $f(\mathrm{col}(U))$ and grad $f(U)$. Following (2.18), we have a convenient expression for grad $f(U)$:

$$\text{grad } f(U) = (I - UU^\top) \text{ grad } \bar{f}(U). \tag{3.16}$$

We thus first set out to compute grad $\bar{f}(U)$, which is a usual gradient.

Let us introduce the function $h : \mathbb{R}_*^{m\times r} \times \mathbb{R}^{r\times n} \to \mathbb{R}$ as follows:

$$h(U,W) = \frac{1}{2}\|C \odot (UW - X_\Omega)\|_\Omega^2 + \frac{\lambda^2}{2}\left(\|W\|_{\mathrm{F}}^2 - \|UW\|_\Omega^2\right). \tag{3.17}$$

Obviously, h is related to $\bar{f}$ via

$$\bar{f}(U) = \min_W h(U,W) = h(U, W_U). \tag{3.18}$$

By definition of the classical gradient, we know that grad $\bar{f}(U) \in \mathbb{R}^{m\times r}$ is the unique vector that satisfies the following condition:

$$\forall H \in \mathbb{R}^{m\times r}, \quad \mathrm{D}\bar{f}(U)[H] = \left\langle H, \text{grad } \bar{f}(U)\right\rangle, \tag{3.19}$$

where $\mathrm{D}\bar{f}(U)[H]$ is the directional derivative of $\bar{f}$ at U along H and $\langle A, B\rangle = \mathrm{Trace}(A^\top B)$ is the usual inner product on $\mathbb{R}^{m\times r}$. We thus need to compute the directional derivatives of $\bar{f}$, which can be done in terms of the directional derivatives of h. Indeed, by the chain rule, it holds that:

$$\mathrm{D}\bar{f}(U)[H] = \mathrm{D}_1 h(U, W_U)[H] + \mathrm{D}_2 h(U, W_U)[W_{U,H}], \tag{3.20}$$

where D_i indicates differentiation w.r.t. the i^{th} argument and

$$W_{U,H} \triangleq \mathrm{D}W(U)[H] \tag{3.21}$$

is the directional derivative of the mapping $U \mapsto W_U$ at U along H. Since

$$W_U = \operatorname*{argmin}_W h(U, W), \tag{3.22}$$

W_U is a critical point of $h(U, \cdot)$ and it holds that $\mathrm{D}_2 h(U, W_U) = 0$. This substantially simplifies the computations as now $\mathrm{D}\bar{f}(U)[H] = \mathrm{D}_1 h(U, W_U)[H]$: we simply need to differentiate h w.r.t. U, considering W_U as constant. Let us define the mask $\Lambda \in \mathbb{R}^{m \times n}$ as:

$$\Lambda_{ij} = \begin{cases} \lambda & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \tag{3.23}$$

Using this notation, we may rewrite h in terms of Frobenius norms only:

$$h(U, W) = \frac{1}{2} \left\| C \odot (UW - X_\Omega) \right\|_{\mathrm{F}}^2 + \frac{\lambda^2}{2} \left\| W \right\|_{\mathrm{F}}^2 - \frac{1}{2} \left\| \Lambda \odot (UW) \right\|_{\mathrm{F}}^2 . \tag{3.24}$$

This is convenient for differentiation, since for suitably smooth mappings g,

$$\mathrm{D}\big(X \mapsto 1/2 \left\| g(X) \right\|_{\mathrm{F}}^2\big)(X)[H] = \langle \mathrm{D}g(X)[H], g(X) \rangle . \tag{3.25}$$

Observing that the following holds for all real matrices A, B, C of identical sizes:

$$\langle A \odot B, C \rangle = \langle B, A \odot C \rangle , \tag{3.26}$$

it then follows that:

$$\begin{aligned}
\mathrm{D}\bar{f}(U)[H] &= \mathrm{D}_1 h(U, W_U)[H] && (3.27)\\
&= \langle C \odot (HW_U), C \odot (UW_U - X_\Omega) \rangle - \langle \Lambda \odot (HW_U), \Lambda \odot (UW_U) \rangle && (3.28)\\
&= \left\langle H, \left[C^{(2)} \odot (UW_U - X_\Omega) \right] W_U^\top - \left[\Lambda^{(2)} \odot (UW_U) \right] W_U^\top \right\rangle && (3.29)\\
&= \left\langle H, \left[(C^{(2)} - \Lambda^{(2)}) \odot (UW_U - X_\Omega) \right] W_U^\top - \lambda^2 X_\Omega W_U^\top \right\rangle && (3.30)\\
&= \left\langle H, \operatorname{grad} \bar{f}(U) \right\rangle . && (3.31)
\end{aligned}$$

Throughout the text, we use the notation $M^{(n)}$ for entry-wise exponentiation, i.e.,

$$(M^{(n)})_{ij} \triangleq (M_{ij})^n. \tag{3.32}$$

For ease of notation, let us define the following m-by-n matrix with the sparsity structure induced by Ω:

$$\hat{C} = C^{(2)} - \Lambda^{(2)}. \tag{3.33}$$

We also introduce the sparse residue matrix R_U:

$$R_U = \hat{C} \odot (UW_U - X_\Omega) - \lambda^2 X_\Omega. \tag{3.34}$$

By identification in (3.31), we obtain a simple expression for the sought gradient:

$$\text{grad } \bar{f}(U) = R_U W_U^\top. \tag{3.35}$$

We pointed out that $\mathrm{D}_2 h(U, W_U) = 0$ because W_U is a critical point of $h(U, \cdot)$. This translates into the following matrix statement:

$$\begin{aligned} \forall H \in \mathbb{R}^{r\times n}, \quad 0 &= \mathrm{D}_2 h(U, W_U)[H] && (3.36)\\ &= \langle C \odot (UH), C \odot (UW_U - X_\Omega)\rangle + \lambda^2 \langle H, W_U\rangle \\ &\quad - \langle \Lambda \odot (UH), \Lambda \odot (UW_U)\rangle && (3.37)\\ &= \left\langle H, U^\top R_U + \lambda^2 W_U \right\rangle. && (3.38) \end{aligned}$$

Hence,

$$U^\top R_U + \lambda^2 W_U = 0. \tag{3.39}$$

Summing up, we obtain the gradient of f (3.15):

$$\text{grad } f(U) = (I - UU^\top) R_U W_U^\top = R_U W_U^\top + \lambda^2 U(W_U W_U^\top), \tag{3.40}$$

We now differentiate (3.40) according to the identity (2.22) for the Hessian of f. To this end, we introduce $\bar{F} : \mathbb{R}_*^{m\times r} \to \mathbb{R}^{m\times r}$:

$$\bar{F}(U) = R_U W_U^\top + \lambda^2 U(W_U W_U^\top). \tag{3.41}$$

We know that the Hessian of f will be given by:

$$\text{Hess}\, f(U)[H] = (I - UU^\top)\mathrm{D}\bar{F}(U)[H]. \tag{3.42}$$

Let us compute the differential of $\bar{F}$:

$$\begin{aligned} \mathrm{D}\bar{F}(U)[H] = [\hat{C} \odot (HW_U + UW_{U,H})]W_U^\top + R_U W_{U,H}^\top \\ + \lambda^2 H(W_U W_U^\top) + \lambda^2 U(W_{U,H} W_U^\top + W_U W_{U,H}^\top). \end{aligned} \tag{3.43}$$

Applying the projector $I - UU^\top$ to $\mathrm{D}\bar{F}(U)[H]$ will cancel out all terms of the form UM (since $(I - UU^\top)U = 0$) and leave all terms of the form HM unaffected (since $U^\top H = 0$). As a consequence of (3.39), applying the projector to $R_U W_{U,H}^\top$ yields:

$$(I - UU^\top) R_U W_{U,H}^\top = R_U W_{U,H}^\top + \lambda^2 U W_U W_{U,H}^\top. \tag{3.44}$$

Applying these observations to (3.42), we obtain an expression for the Hessian of our objective function on the Grassmann manifold:

$$\begin{aligned} \text{Hess}\, f(U)[H] = (I - UU^\top)\left[\hat{C} \odot (HW_U + UW_{U,H})\right] W_U^\top \\ + R_U W_{U,H}^\top + \lambda^2 H(W_U W_U^\top) + \lambda^2 U(W_U W_{U,H}^\top). \end{aligned} \tag{3.45}$$

Not surprisingly, the formula for the Hessian requires us to compute $W_{U,H}$, the differential of the mapping $U \mapsto W_U$ along H. We provide formulas for W_U and $W_{U,H}$ in the next section.

3.3. W_U and its derivative $W_{U,H}$. We still need to provide explicit formulas for W_U and $W_{U,H}$. We assume $U \in \mathrm{St}(m,r)$ since we use orthonormal matrices to represent points on the Grassmannian and $U^\top H = 0$ since $H \in \mathrm{T}_U\mathrm{Gr}(m,r)$.

The vectorization operator, vec, transforms matrices into vectors by stacking their columns—in Matlab notation, $\mathrm{vec}(A) = A(:)$. Denoting the Kronecker product of two matrices by $\otimes$, we will use the well-known identity for matrices A, Y, B of appropriate sizes [7]:

$$\mathrm{vec}(AYB) = (B^\top \otimes A)\mathrm{vec}(Y). \tag{3.46}$$

We also write I_Ω for the orthonormal $k \times mn$ matrix such that

$$\mathrm{vec}_\Omega(M) = I_\Omega \mathrm{vec}(M) \tag{3.47}$$

is a vector of length $k = |\Omega|$ corresponding to the entries M_{ij} for $(i,j) \in \Omega$, taken in order from $\mathrm{vec}(M)$.

Computing W_U comes down to minimizing the least-squares cost $h(U,W)$ (3.17) with respect to W. We manipulate h to reach a standard form for least-squares. To this end, we first define $S \in \mathbb{R}^{k \times mn}$:

$$S = I_\Omega \,\mathrm{diag}(\mathrm{vec}(C)). \tag{3.48}$$

This will come in handy through the identity

$$\begin{aligned}\|C \odot M\|_\Omega^2 = \|\mathrm{vec}_\Omega(C \odot M)\|_2^2 &= \|I_\Omega \mathrm{vec}(C \odot M)\|_2^2 \\ &= \|I_\Omega \,\mathrm{diag}(\mathrm{vec}(C))\mathrm{vec}(M)\|_2^2 = \|S\mathrm{vec}(M)\|_2^2 .\end{aligned} \tag{3.49}$$

We use this in the following transformation of h:

$$\begin{aligned}
h(U,W) &= \frac{1}{2}\|C \odot (UW - X_\Omega)\|_\Omega^2 + \frac{\lambda^2}{2}\|W\|_{\mathrm{F}}^2 - \frac{\lambda^2}{2}\|UW\|_\Omega^2 \\
&= \frac{1}{2}\|S\mathrm{vec}(UW) - \mathrm{vec}_\Omega(C \odot X_\Omega)\|_2^2 + \frac{\lambda^2}{2}\|\mathrm{vec}(W)\|_2^2 - \frac{\lambda^2}{2}\|\mathrm{vec}_\Omega(UW)\|_2^2 \\
&= \frac{1}{2}\|S(I_n \otimes U)\mathrm{vec}(W) - \mathrm{vec}_\Omega(C \odot X_\Omega)\|_2^2 + \frac{1}{2}\|\lambda I_{rn}\mathrm{vec}(W)\|_2^2 \\
&\quad - \frac{1}{2}\|\lambda I_\Omega(I_n \otimes U)\mathrm{vec}(W)\|_2^2 \\
&= \frac{1}{2}\left\|\begin{bmatrix} S(I_n \otimes U) \\ \lambda I_{rn}\end{bmatrix}\mathrm{vec}(W) - \begin{bmatrix}\mathrm{vec}_\Omega(C \odot X_\Omega) \\ 0_{rn}\end{bmatrix}\right\|_2^2 \\
&\quad - \frac{1}{2}\left\|\left[\lambda I_\Omega(I_n \otimes U)\right]\mathrm{vec}(W)\right\|_2^2 \\
&= \frac{1}{2}\|A_1 w - b_1\|_2^2 - \frac{1}{2}\|A_2 w\|_2^2 \\
&= \frac{1}{2}w^\top(A_1^\top A_1 - A_2^\top A_2)w - b_1^\top A_1 w + \frac{1}{2}b_1^\top b_1,
\end{aligned} \tag{3.50}$$

with $w = \mathrm{vec}(W) \in \mathbb{R}^{rn}$, $0_{rn} \in \mathbb{R}^{rn}$ is the zero-vector and the obvious definitions for A_1, A_2 and b_1. If $A_1^\top A_1 - A_2^\top A_2$ is positive-definite there is a unique minimizing vector $\mathrm{vec}(W_U)$, given by:

$$\mathrm{vec}(W_U) = (A_1^\top A_1 - A_2^\top A_2)^{-1} A_1^\top b_1. \tag{3.51}$$

It is easy to compute the following:

$$A_1^\top A_1 = (I_n \otimes U^\top)(S^\top S)(I_n \otimes U) + \lambda^2 I_{rn}, \tag{3.52}$$
$$A_2^\top A_2 = (I_n \otimes U^\top)(\lambda^2 I_\Omega^\top I_\Omega)(I_n \otimes U), \tag{3.53}$$
$$A_1^\top b_1 = (I_n \otimes U^\top) S^\top \mathrm{vec}_\Omega(C \odot X_\Omega) = (I_n \otimes U^\top)\mathrm{vec}(C^{(2)} \odot X_\Omega). \tag{3.54}$$

Note that $S^\top S - \lambda^2 I_\Omega^\top I_\Omega = \mathrm{diag}(\mathrm{vec}(\hat{C}))$. Let use call this matrix B:

$$B \triangleq S^\top S - \lambda^2 I_\Omega^\top I_\Omega = \mathrm{diag}(\mathrm{vec}(\hat{C})). \tag{3.55}$$

We then define $A \in \mathbb{R}^{rn \times rn}$ as:

$$A \triangleq A_1^\top A_1 - A_2^\top A_2 = (I_n \otimes U^\top) B (I_n \otimes U) + \lambda^2 I_{rn}. \tag{3.56}$$

Observe that the matrix A is block-diagonal, with n symmetric blocks of size r. This structure stems from the fact that each column of W_U can be computed separately from the others. Each block is indeed positive-definite provided $\lambda > 0$ (making A positive-definite too). Thanks to the sparsity of $\hat{C}$, we can compute these n blocks with $\mathcal{O}(kr^2)$ flops. To solve systems in A, we compute the Cholesky factorization of each block, at a total cost of $\mathcal{O}(nr^3)$ flops. Once these factorizations are computed, each system only costs $\mathcal{O}(nr^2)$ flops to solve [23]. Collecting all equations in this section, we obtain a closed-form formula for W_U:

$$\mathrm{vec}(W_U) = A^{-1}(I_n \otimes U^\top)\mathrm{vec}(C^{(2)} \odot X_\Omega) \tag{3.57}$$
$$= A^{-1}\mathrm{vec}\left(U^\top [C^{(2)} \odot X_\Omega]\right), \tag{3.58}$$

where A is a function of U and we have a fast way of applying A^{-1} to vectors.

We would like to differentiate W_U with respect to U, that is, compute

$$\begin{aligned}\mathrm{vec}(W_{U,H}) &= \mathrm{D}\left(U \mapsto \mathrm{vec}(W_U)\right)(U)[H] \\ &= \mathrm{D}\left(U \mapsto A^{-1}\right)(U)[H] \cdot \mathrm{vec}\left(U^\top [C^{(2)} \odot X_\Omega]\right) \\ &\quad + A^{-1}\mathrm{vec}\left(H^\top [C^{(2)} \odot X_\Omega]\right).\end{aligned} \tag{3.59}$$

Using the formula $\mathrm{D}(Y \mapsto Y^{-1})(X)[H] = -X^{-1}HX^{-1}$ [7] for the differential of the inverse of a matrix, we obtain

$$\begin{aligned}\mathrm{D}\left(U \mapsto A^{-1}\right)(U)[H] &= -A^{-1} \cdot \mathrm{D}\left(U \mapsto A\right)(U)[H] \cdot A^{-1} \\ &= -A^{-1}\Big[(I_n \otimes H^\top)B(I_n \otimes U) + (I_n \otimes U^\top)B(I_n \otimes H)\Big]A^{-1}.\end{aligned}$$

Let us plug this back in (3.59), recalling (3.58) for W_U:

$$\begin{aligned}
\operatorname{vec}(W_{U,H}) &= -A^{-1}\Big[(I_n \otimes H^\top)B(I_n \otimes U) + (I_n \otimes U^\top)B(I_n \otimes H)\Big]\operatorname{vec}(W_U) \\
&\quad + A^{-1}\operatorname{vec}\Big(H^\top[C^{(2)} \odot X_\Omega]\Big) \\
&= -A^{-1}\Big[(I_n \otimes H^\top)B\operatorname{vec}(UW_U) + (I_n \otimes U^\top)B\operatorname{vec}(HW_U)\Big] \\
&\quad + A^{-1}\operatorname{vec}\Big(H^\top[C^{(2)} \odot X_\Omega]\Big) \\
&= -A^{-1}\Big[(I_n \otimes H^\top)\operatorname{vec}(\hat{C} \odot UW_U) + (I_n \otimes U^\top)\operatorname{vec}(\hat{C} \odot HW_U)\Big] \\
&\quad + A^{-1}\operatorname{vec}\Big(H^\top[C^{(2)} \odot X_\Omega]\Big) \\
&= -A^{-1}\Big[\operatorname{vec}(H^\top[\hat{C} \odot UW_U]) + \operatorname{vec}(U^\top[\hat{C} \odot HW_U])\Big] \\
&\quad + A^{-1}\operatorname{vec}\Big(H^\top[C^{(2)} \odot X_\Omega]\Big).
\end{aligned} \tag{3.60}$$

Now recall the definition of R_U (3.34) and observe that

$$\hat{C} \odot UW_U - C^{(2)} \odot X_\Omega = \hat{C} \odot UW_U - \hat{C} \odot X_\Omega - \Lambda^{(2)} \odot X_\Omega = R_U. \tag{3.61}$$

Plugging the latter in (3.60) yields a compact expression for the directional derivative $W_{U,H}$:

$$\operatorname{vec}(W_{U,H}) = -A^{-1}\operatorname{vec}\Big(H^\top R_U + U^\top\big(\hat{C} \odot (HW_U)\big)\Big). \tag{3.62}$$

The most expensive operation involved in computing $W_{U,H}$ ought to be solving a linear system in A. Fortunately, we already factored the n small diagonal blocks of A in Cholesky form to compute W_U. Consequently, after computing W_U, computing $W_{U,H}$ is cheap. We summarize the complexities in the next section.

3.4. Numerical complexities. By exploiting the sparsity of many of the matrices involved and the special structure of the matrix A (3.56) appearing in the computation of W_U and $W_{U,H}$, it is possible to compute the objective function f (3.15) as well as its gradient (3.40) and its Hessian (3.45) on the Grassmannian in time linear in the size of the data $k = |\Omega|$. Memory complexities are also linear in k. We summarize the computational complexities in Table 3.1. Note that most computations are easily parallelizable, but we do not take advantage of it here.

The computational cost of W_U (3.58) is dominated by the computation of the n diagonal blocks of A of size $r \times r$—$\mathcal{O}(kr^2)$—and by the Cholesky factorization of these—$\mathcal{O}(nr^3)$—hence a total cost of $\mathcal{O}(kr^2 + nr^3)$ flops. The computation of $f(U)$ is dominated by the cost of computing W_U, hence they have the same complexity. Computing the gradient of f once W_U is known involves just a few supplementary matrix-matrix multiplications. Exploiting the sparsity of these matrices keeps the cost low: $\mathcal{O}(kr + (m+n)r^2)$ flops. Computing the Hessian of f along H requires (on top of W_U) the computation of $W_{U,H}$ and a few (structured) matrix-matrix products. Computing $W_{U,H}$ involves solving a linear system in A. Since we computed W_U already, we have a Cholesky-factored representation of A, hence solving a system in A is cheap: $\mathcal{O}(nr^2)$ flops. The total cost of computing $W_{U,H}$ and $\operatorname{Hess} f(U)[H]$ is $\mathcal{O}(kr + (m+n)r^2)$ flops.

Notice that computing the gradient and the Hessian is *cheaper* than computing f. This stems from the fact that once we have computed f at a certain point U, much of the work (such as computing and factoring the diagonal blocks of A) can be reused to compute higher-order information. This prompts us to investigate methods that exploit second-order information; cf. RTRMC 2 described below.

TABLE 3.1
All complexities are linear in $k = |\Omega|$, the number of observed entries.

Computation	Complexity	By-products	Formulas
W_U and $f(U)$	$\mathcal{O}(kr^2 + nr^3)$	Cholesky form of A	(3.13)–(3.15), (3.56), (3.58)
grad $f(U)$	$\mathcal{O}(kr + (m+n)r^2)$	R_U and $W_U W_U^\top$	(3.33), (3.34), (3.40)
$W_{U,H}$, Hess $f(U)[H]$	$\mathcal{O}(kr + (m+n)r^2)$		(3.45), (3.62)

4. The Riemannian trust-region method. We use a Riemannian trust-region (RTR) method [1] to minimize (3.15), via the freely available Matlab package GenRTR (version 0.3.0). The package is available at this address:
http://www.math.fsu.edu/~cbaker/GenRTR/?page=download.
GenRTR is a general purpose optimization package to minimize functions on Riemannian manifolds.

The RTR method is an iterative descent method. At the current iterate $\mathscr{U} = \mathrm{col}(U)$ represented by an arbitrary orthonormal basis U, the RTR method uses the retraction (2.23)

$$\mathrm{R}_U : \mathrm{T}_U\mathrm{Gr}(m,r) \to \mathrm{Gr}(m,r)$$

to build a quadratic model

$$m_U : \mathrm{T}_U\mathrm{Gr}(m,r) \to \mathbb{R}$$

of the lifted objective function (*lift*)

$$f \circ \mathrm{R}_U : \mathrm{T}_U\mathrm{Gr}(m,r) \to \mathbb{R}.$$

It then classically minimizes the model m_U inside a trust region on the vector space $\mathrm{T}_U\mathrm{Gr}(m,r)$ (*solve*), and retracts the resulting tangent vector H to a candidate $U^+ = \mathrm{R}_U(H)$ on the Grassmannian (*retract*). The quality of $\mathscr{U}^+ = \mathrm{col}(U^+)$ is assessed using f and the step is accepted or rejected accordingly. Likewise, the radius of the trust region is adapted based on the observed quality of the model. The procedure is iterated until a predefined stopping criterion is met.

The model m_U of $f \circ \mathrm{R}_U$ has the form:

$$m_U(H) = f(U) + \langle \mathrm{grad}\, f(U), H\rangle_U + \frac{1}{2}\langle A(U)[H], H\rangle_U ,$$

where $A(U)$ is some symmetric linear operator on $\mathrm{T}_U\mathrm{Gr}(m,r)$. A powerful property of the RTR method is that global convergence of the algorithm toward critical points—local minimizers in practice since it is a descent method—is guaranteed independently of $A(U)$ [1, Thm 4.4, Cor. 4.6]. Using the Hessian of f as the operator A yields the best (in some sense) quadratic model of the lifted cost function, but this strong property

of the RTR method tells us that other (cheaper) operators can be used without losing global convergence. It is thus worth trying.

We take advantage of this and first set A to be the identity. This yields a (first-order) steepest-descent algorithm we refer to as **RTRMC 1**. It differs from classical steepest-descent algorithms in that the line-search step is replaced by the trust-region paradigm.

Taking $A(U)$ to be the Hessian of f at U (3.45), the RTR method enjoys a quadratic convergence rate once we enter some neighborhood of a nondegenerate local minimizer. This remains true even if we only approximately minimize m_U within the trust region using a few steps of a well chosen iterative method [1, Thm 4.14] such as, for example, a truncated conjugate gradient method. This means that the RTR method only requires a few computations of the Hessian along specific directions at each iteration: there is no need for an exact solve of the Newton equations as would be required by the Newton method for example. We name the proposed (second-order) method using the Hessian **RTRMC 2**. From Section 3.4, we know that computing the Hessian is relatively cheap for our problem and hence we expect RTRMC 2 to be competitive.

To summarize, RTRMC 1 and 2 both enjoy global convergence toward critical points. RTRMC 1 merely enjoys a linear local convergence rate, while RTRMC 2 benefits from a quadratic local convergence rate, at the cost of somewhat more expensive iterations.

The RTR method requires an initial guess for the column space, $\mathrm{col}(U_0)$. To this end, we simply compute the r dominant left singular vectors of the masked matrix X_Ω. In Matlab, this is achieved by calling $[U_0, S_0, V_0] = \texttt{svds}(X_\Omega, r)$. Since X_Ω is sparse, this has a reasonable cost (linear in k).

We further set a few parameters for the RTR method. Among them, the maximum and initial trust-region radii, respectively $\bar{\Delta}$ and Δ_0. The trust-region radius at a given iterate is the upper bound on the stepsize at that iterate. Since the Grassmann manifold is compact, it makes sense to choose $\bar{\Delta}$ in proportion to the diameter of this manifold, i.e., the largest geodesic distance between any two points on $\mathrm{Gr}(m, r)$. The distance between two subspaces is $\sqrt{\theta_1^2 + \cdots + \theta_r^2}$, where the θ_i's are the principal angles between these spaces. Since these angles are bounded by $\pi/2$, we set $\bar{\Delta} = \pi\sqrt{r}/2$. We pick the initial trust-region radius as $\Delta_0 = \bar{\Delta}/8$. Two additional parameters are θ and κ (see [1]), which we set to 1 and 0.1 respectively (this is the default setting). For RTRMC 2, we also need to set a limit on the number of inner iterations. We usually set this to 40, that is: at each iterate, we allow up to 40 Hessian computations (unless otherwise stated). While this limit may jeopardize the quadratic convergence rate of RTRMC 2, we find that it constitutes a good trade-off between fast individual iterations and few overall iterations needed for convergence.

In interfacing GenRTR with code for the objective function and its derivatives, we add a caching layer that makes sure we reuse recently computed quantities if GenRTR asks for them multiple times. This increases the memory usage of the proposed algorithm in proportion to the data size k, in exchange for a significant speed up. A more careful implementation of GenRTR would provide the same speed up without the memory overhead.

Notice that RTRMC requires the user to specify the rank r of the target matrix X. If one over-estimates the rank, the factorization UW will result in a rank deficient factor W, which is detectable and the rank can then be reduced. If one under-estimates the rank, the algorithm will converge toward a lower-rank approximation of the target

TABLE 5.1

All Matlab implementations call subroutines in non-Matlab code to efficiently deal with the sparsity of the matrices involved. PROPACK [17] is a free package for large and sparse SVD computations.

Method	Software	Comment
RTRMC 1	Matlab + some C-Mex	The proposed method with "approximate Hessian" set to identity, i.e., no second-order information. $\lambda = 10^{-6}$.
RTRMC 2	Matlab + some C-Mex	The proposed method with exact Hessian.
OptSpace	C code	[16] with $\lambda = 0$. Gradient descent on two Grassmannians.
ADMiRA	Matlab with PROPACK	[18] Matching pursuit based.
LMaFit	Matlab + some C-Mex	[25] Alternating minimization.
Balanced Fact.	Matlab + some C-Mex	[20] One of their Riemannian regression methods.

matrix, which may be the desired behavior. This is certainly true for noisy problem instances.

5. Numerical experiments. We test the proposed algorithms on synthetic data and compare their performances against OptSpace, ADMiRA, LMaFit and Balanced Factorization in terms of accuracy and computation time. All algorithms are run sequentially by Matlab on the same personal computer[1] except for Scenario 5, which required more RAM. Table 5.1 specifies a few implementation details.

The proposed methods (RTRMC 1 and 2) and Balanced Factorization require knowledge of the target rank r. OptSpace, ADMiRA and LMaFit include a mechanism to guess the rank, but benefit from knowing it, hence we provide the target rank to these methods too. In the conference paper [6], we compared against SVT too [8]. The available code for this method does not allow to specify the rank. As a result, SVT suffered an unfair disadvantage and we decided to remove it from the benchmark.

We use the root mean square error (RMSE) criterion to assess the quality of reconstruction of X with $\hat{X}$:

$$\mathrm{RMSE}(X, \hat{X}) = \|X - \hat{X}\|_{\mathrm{F}} / \sqrt{mn}. \tag{5.1}$$

This quantity is cheap to compute when the target matrix is given in factored low-rank form $X = AB$ and $\hat{X}$ is (by construction) in the same form $\hat{X} = UW$. The RMSE may then be computed in $\mathcal{O}((m+n)r^2)$ flops using:

$$(mn)\mathrm{RMSE}(AB, UW)^2 = \mathrm{Trace}((A^\top A)(BB^\top)) + \mathrm{Trace}((U^\top U)(WW^\top)) - 2\mathrm{Trace}((U^\top A)(BW^\top)). \tag{5.2}$$

Be wary though that this formula is numerically inaccurate when the RMSE is much smaller than the norm of either AB or UW, owing to the computation of the difference of close large numbers. For the purpose of comparing the algorithms, we add code to all implementations so that the RMSE will be computed at each iterate. The time spent in this calculation is discounted from the reported timings.

[1] Intel Core i5 670 @ 3.60GHz (4), 8Go RAM, Matlab 7.10 (R2010a), Windows 7 (64 bits).

A number of factors intervene in the difficulty of a low-rank matrix completion task. Obviously, the *size* $m \times n$ of the matrix X to recover and its *rank* r are fundamental quantities. Among others, the presence or absence of *noise* is important. If the observations X_Ω are noisy, then X_Ω is not the masked version of a low-rank matrix X, but of a matrix which is close to being low-rank: X + noise. Of course, different noise distributions (with and without outliers etc.) can be investigated. The search space—the manifold of $m \times n$ matrices of rank r—has dimension $d = r(m + n - r)$. The *oversampling ratio* k/d is a crucial quantity: the larger it is, the easier the task is. The *sampling process* also plays some role in the difficulty of matrix completion. Under uniform sampling for example, all entries of the matrix X have identical probability of being observed. Uniform sampling prevents pathological cases (where some rows or columns have no observed entry at all for example) from happening with high probability. Real data sets often have nonuniform samplings. For example, some movies are particularly popular and some users rate particularly many movies. Finally, the *conditioning* of the low-rank matrix X (the ratio of its largest to its smallest nonzero singular values) may affect the difficulty of matrix completion too.

In the following numerical experiments, we illustrate the behavior of RTRMC against these various parameters, in comparison with prior art.

Scenario 1: low oversampling ratio. We first compare the convergence behavior of the different methods with square matrices $m = n = 10\,000$ and rank $r = 10$. We generate $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$ with i.i.d. normal entries of zero mean and unit variance. The target matrix is $X = AB$. We sample $2.5d$ entries uniformly at random without noise, which yields a sampling ratio of 0.5%. This is fairly low. Figure 5.1 is typical and shows the evolution of the RMSE as a function of time (left) and iteration count (right). LMaFit and RTRMC 2 both emerge as good contestants. We mention that, at this low sampling level, the task is not always completed.

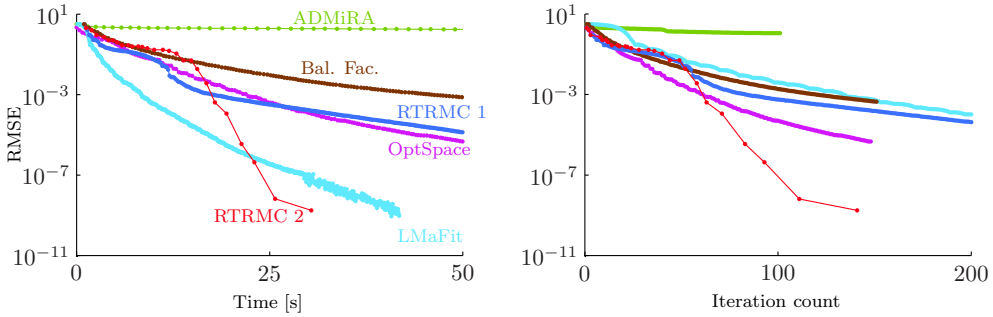

FIG. 5.1. *Evolution of the RMSE for the six methods under Scenario 1 ($m = n = 10\,000$, $r = 10$, $|\Omega|/(mn) = 0.5\%$, i.e., 99.5% of the entries are unknown). For RTRMC 2, we count the number of inner iterations, i.e., the number of parallelizable steps. ADMiRA stagnates. All other methods eventually find the exact solution. LMaFit and RTRMC 2 perform the best.*

Scenario 2: rectangular matrices. In this second test, we repeat the previous experiment with rectangular matrices: $m = 1\,000, n = 30\,000, r = 5$ and a sampling ratio of 2.6% ($5d$ known entries). We expect RTRMC to perform well on rectangular matrices since the dimension of the Grassmann manifold we optimize on only grows linearly with $\min(m, n)$, whereas it is the (simple) least-squares problem dimension that grows linearly in $\max(m, n)$. Figure 5.2 is typical and shows indeed that RTRMC is the fastest tested algorithm in this scenario.

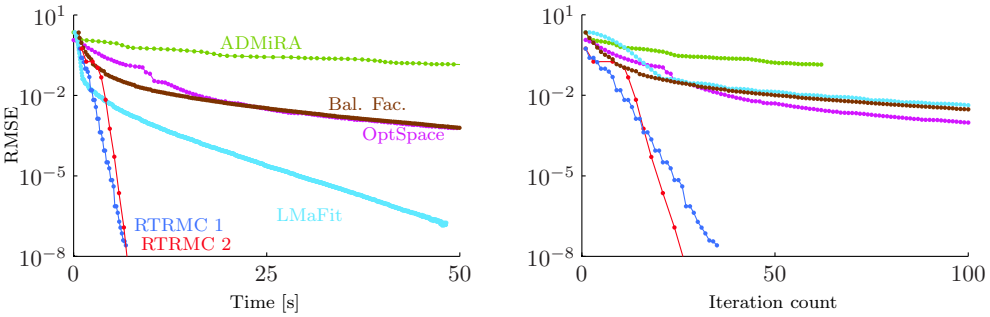

FIG. 5.2. *Evolution of the RMSE for the six methods under Scenario 2 ($m = 1\,000, n = 30\,000$, $r = 5$, $|\Omega|/(mn) = 2.6\%$). For rectangular matrices, RTRMC is especially efficient owing to the linear growth of the dimension of the search space in* $\min(m, n)$*, whereas for most methods the growth is linear in* $m + n$*.*

Scenario 3: bad conditioning. For this third test, we generate A and B as in Scenario 1 with $m = n = 1\,000, r = 10$. We then compute the thin SVD of the product $AB = USV^\top$, which can be done efficiently using economic QR factorizations of A and B separately. The diagonal $r \times r$ matrix S is replaced with a diagonal matrix S_+ whose diagonal entries decay exponentially as follows: $(S_+)_{ii} = S_{11} \exp(-5(i-1)/(r-1))$, for $i = 1 \dots r$. The product $X = US_+V^\top$ is then formed and this is the rank-r target matrix, of which we observe $5d$ entries uniformly at random (that is about 10%). Notice that X has a much worse conditioning ($e^5 \approx 148$) than the original product AB (typical conditioning below 2) without being unrealistically bad. From the numerical results in Figure 5.3, it appears that first-order methods have trouble solving this type of matrix completion tasks, whereas RTRMC 2, using second-order information more extensively than in other experiments, still manages to solve the problem to high accuracy. We raised the inner iteration limit of RTRMC 2 to 500, authorizing it to execute more Hessian computations than usually.

We venture an explanation of the better performance of the proposed second-order method here by studying the conditioning of the Hessian of the cost function f at the solution $\mathrm{col}(U)$. This Hessian is a symmetric linear operator on the vector space $\mathrm{T}_U\mathrm{Gr}(m, r)$ of dimension $r(m-r) = 9900$. For the present experiment (target $US_+V^\top$), we compute the 9900 associated eigenvalues. They are all positive. The condition number of the Hessian is on the order of 75 000. On the same experiment but without fiddling with the spectrum (target $USV^\top$), the condition number of the Hessian at U is on the order of 10–20. The fact that the bad conditioning of X seems to translate into even worse conditioning of the Hessian at the solution might explain why first-order methods perform poorly in this situation.

Scenario 4: nonuniform sampling. As a fourth test, we generate A and B as in Scenario 1 with $m = 1\,000, n = 10\,000$ and $r = 10$. The target matrix is $X = AB$, of which we observe entries with a nonuniform distribution. This artificial sampling mimics a situation where rows correspond to movies and columns correspond to raters. Some of the movies are much more often rated than others, and some of the raters rate many more movies than others. In a first experiment, each of the 100 first movies (they are the least popular ones) has a probability of being rated that is 5 times smaller than the 800 following movies. The 100 last movies are 5 times as likely to be rated as the 800 latter (they are the popular ones). Furthermore, each rater rates between 15 and 50 movies, uniformly at random, resulting in an oversampling ratio of

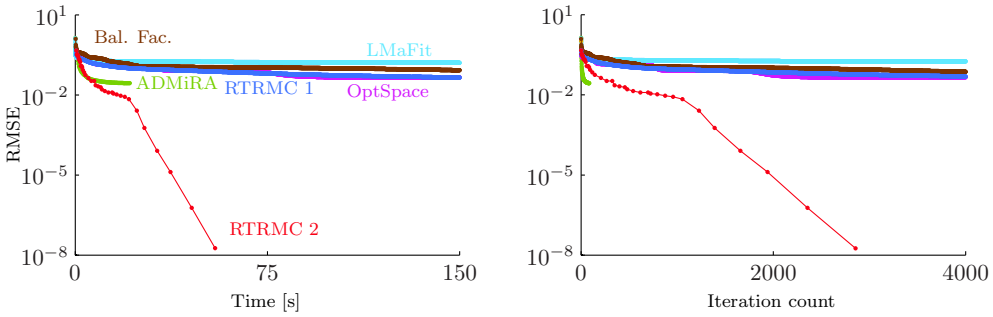

FIG. 5.3. *Evolution of the RMSE for the six methods under Scenario 3 ($m = n = 1\,000$, $r = 10$, $|\Omega|/(mn) = 10\%$ and condition number of about 150 for the target matrix). First-order methods exhibit bad behavior whereas RTRMC 2, using second-order information more extensively than on better conditioned problems, shows superior convergence quality.*

2.94 (3.2%). Figure 5.4 shows the associated mask probability, where raters (columns) have been sorted by number of given ratings.

Figure 5.5 (left panel) shows the behavior of the various methods tested on this instance of the problem. This scenario is particularly challenging and slows down first-order methods significantly. RTRMC 2 on the other hand still solves the completion task to high accuracy in relatively little time. Compare this with the right panel, where m, n, r, k are the same as in the left panel, but the sampling is uniform. This shows that it is indeed the heavily nonuniform sampling that makes this problem more challenging.

In Figure 5.6, we show the results for a similar test but this time the 100 first movies are only twice less likely to be rated that the 800 next ones, and the 100 last movies are only twice as likely to be rated as the latter 800. In this more uniform sampling scenario, we observe excellent behavior for both RTRMC 1 and RTRMC 2.

We conclude that RTRMC 2 can withstand non-uniformity in the sampling process.

FIG. 5.4. *Proposed nonuniform sampling density for Scenario 4. This image represents a $1\,000 \times 10\,000$ matrix. Each entry is colored on a grayscale. The lighter the color, the slimmer the chances that this entry will be observed. We see that entries in the top 100 rows are much less likely to be observed than in the bottom 100 rows. Columns on the right are also more densely sampled than columns on the left. This artificial sampling process mimics a situation where some objects are more popular than others (and hence more often rated) and some raters are more active than others.*

Scenario 5: larger instances. In this fifth test, we try out the various algorithms on a larger instance of matrix completion: $m = 10\,000, n = 100\,000, r = 20$ with oversampling ratio of 5, that is, 1.1% of the entries are observed, sampled uniformly at random. The target matrix $X = AB$ (formed as previously) has a billion entries. For this test, we use a computer with more RAM[2]. Figure 5.7 shows that RTRMC

[2]HP DL180 + Intel Xeon X5670 2.93 GHz (12 core), 144Go RAM, Matlab 7.10 (R2010a), Linux (64 bits).

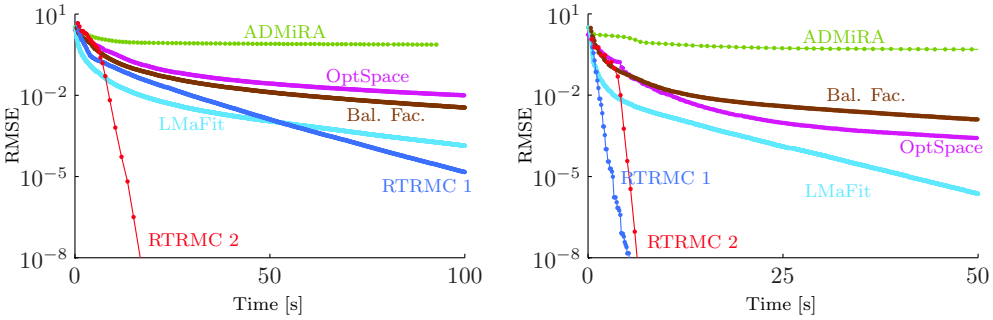

FIG. 5.5. *(Left) Evolution of the RMSE for the six methods under Scenario 4 ($m = 1\,000, n = 10\,000$, $r = 10$, $|\Omega|/(mn) = 3.2\%$ and nonuniform sampling as depicted in Figure 5.4). RTRMC 2 shows excellent behavior, whereas first-order methods are significantly slowed down. (Right) Same experiment with uniform sampling: the situation is comparable to that depicted in Figure 5.2, confirming that the non-uniformity is the root of the difficulty in this test.*

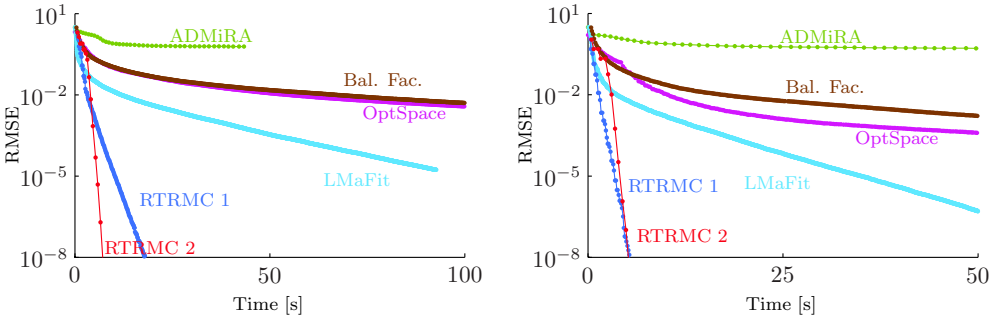

FIG. 5.6. *(Left) Same experiment as in Figure 5.5 (Left), with more uniform sampling (see description of Scenario 4). We see that RTRMC can withstand non-uniformity in the sampling process. (Right) Same experiment (same m, n, r, k) with uniform sampling. This is a different realization of the same experiment as in Figure 5.5 (Right).*

performs well on such instances, and so does LMaFit. The plateaus in the RTRMC 1 convergence (where two or three successive iterations reject the step and hence the RMSE stagnates) correspond to shrinking steps of the trust-region.

Scenario 6: noisy observations. As a sixth and last test, we try out RTRMC on a class of noisy instances of matrix completion with $m = n = 5\,000, r = 10$ and oversampling ratio of 4, that is, 1.6% of the entries are observed, sampled uniformly at random. The target matrix $X = AB$ is formed as before with $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{r\times n}$ whose entries are i.i.d. normal random variables. Notice that this implies the X_{ij}'s are also zero-mean Gaussian variables but with variance r and not independent. We then generate a noise matrix N_Ω, such that the $(N_\Omega)_{ij}$'s for (i, j) in Ω are i.i.d. normal random variables (Gaussian distribution with zero mean, unit variance), and the other entries of N are zero. The observed matrix is $X_\Omega + \sigma N_\Omega$, where σ^2 is the noise variance. The signal to noise ratio (SNR) is thus r/σ^2. This is the same setup as the standard scenario in [15].

All algorithms based on a least-squares strategy should perform rather well on this scenario, since least-squares are particularly well-suited to filter out Gaussian noise. We should however expect those same algorithms to perform suboptimally in the face of outliers. RTRMC makes no claim of being robust against outliers, hence we only

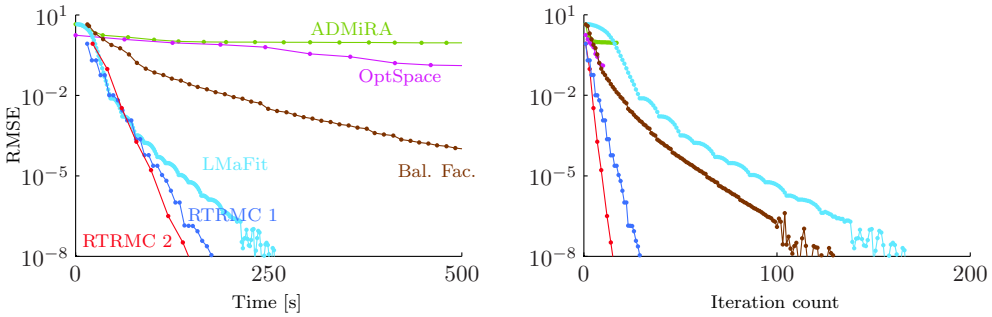

FIG. 5.7. *Evolution of the RMSE for the six methods under Scenario 5 ($m = 10\,000$, $n = 100\,000$, $r = 20$, $|\Omega|/(mn) = 1.1\%$). This larger test is run on a more powerful computer, with more RAM available. RTRMC and LMaFit exhibit a good behavior on problem instances of this larger size.*

test against Gaussian noise and show excellent behavior in that case on Figure 5.8. For comparison, we use the same oracle as in [15], that is: we compare the RMSE obtained by RTRMC with the RMSE we could obtain if we knew the column space $\mathrm{col}(X)$. This is known to be equal to $\mathrm{RMSE}_{\mathrm{oracle}} = \sigma\sqrt{(2nr - r^2)/k}$ (in expectation). Figure 5.8 illustrates the fact that, not surprisingly, RTRMC reaches almost the same RMSE as the oracle as soon as the SNR is large enough.

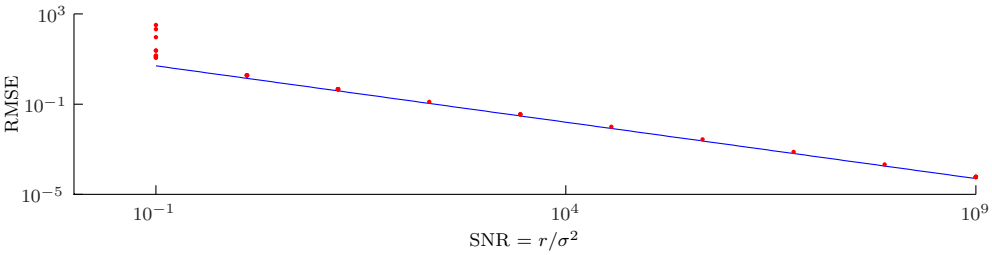

FIG. 5.8. *RTRMC is well suited to solve matrix completion tasks under Gaussian noise, owing to its least-squares objective function. The noise model is described in Scenario 6: $m = n = 5\,000, r = 10, |\Omega|/(mn) = 1.6\%$. The straight blue line indicates the RMSE an oracle who knows the column space of the target matrix X would reach. For different values of SNR, we generate 10 problem instances and solve them with RTRMC. The red dots report the RMSE's reached by RTRMC. For SNR's larger than 1, the dots are mostly indistinguishable and close to the oracle quality, which shows that Gaussian noise is easily filtered out.*

6. Conclusion. Our contribution is a set of two efficient numerical methods to solve large low-rank matrix completion problems: RTRMC 1 and 2. These are respectively first- and second-order Riemannian trust-region methods which minimize a smooth least-squares cost function on the Grassman manifold. RTRMC competes with the state-of-the-art and enjoys proven global and local convergence to local optima, with a quadratic convergence rate for RTRMC 2.

The methods we propose are particularly efficient on rectangular matrices. We believe this is because the dimension of the search space, $\mathrm{Gr}(m, r)$, grows linearly with $\min(m, n)$, whereas for most competing methods the growth is in $m+n$. We also observed that RTRMC 2 performs better than first-order methods when the matrix to complete is badly conditioned. We believe this is because the bad conditioning of the target matrix translates into an even worse conditioning of the Hessian of the

cost function at the solution, as seen in the numerical experiments. Furthermore, RTRMC 2 can withstand non-uniformity in the sampling process, i.e., in the way the observed entries of the target matrix are selected. We moreover observed that RTRMC is capable of solving matrix completion tasks at low oversampling ratios such as 2.5 and for large instances (on the order of a billion entries). RTRMC is effective against Gaussian noise, which is not surprising given its least-squares nature.

We conclude by mentioning a couple research directions that could possibly lead to improvements of the RTRMC method. First, the trust-region algorithm usually exhibits a slow start, during which it executes one Hessian computation at each iteration but eventually makes a gradient step anyway. Those first few steps are used by GenRTR to learn the scale of the problem and adapt its trust-region radius. It is only after this learning period that the trust-region scheme can benefit from second-order information. Possibly, one could try executing gradient steps at first without Hessian computations, while learning the trust-region radius with a yet-to-design procedure. After a few steps, one could then use second-order information with a (hopefully) good trust-region radius estimate. Second, the trust-region method can benefit from using a preconditioner for the Hessian. While the Hessian has a rather involved formulation, there may exist good and cheap approximations of it that could constitute good preconditioners. It is a well-known fact that good preconditioners can lead to dramatic accelerations in solving linear systems, so that this appears as a good research direction for future investigations.

Matlab code for RTRMC 1 and 2 is available at:
http://www.inma.ucl.ac.be/~absil/RTRMC/.

Acknowledgments. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. NB is an FNRS research fellow (*Aspirant*). The scientific responsibility rests with its authors.

REFERENCES

[1] P.-A. Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds*, Foundations of Computational Mathematics, 7 (2007), pp. 303–330.
[2] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008.
[3] L. Balzano, R. Nowak, and B. Recht, *Online identification and tracking of subspaces from highly incomplete information*, in Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on, IEEE, 2010, pp. 704–711.
[4] R.M. Bell, Y. Koren, and C. Volinsky, *The BellKor 2008 solution to the Netflix prize*, Statistics Research Department at AT&T Research, (2008).
[5] W.M. Boothby, *An introduction to differentiable manifolds and Riemannian geometry*, vol. 120 of Pure and Applied Mathematics, Elsevier, 1986.
[6] N. Boumal and P.-A. Absil, *RTRMC: A Riemannian trust-region method for low-rank matrix completion*, in Advances in Neural Information Processing Systems 24 (NIPS), J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, eds., 2011, pp. 406–414.
[7] M. Brookes, *The matrix reference manual*, Imperial College London, (2005).
[8] J.F. Cai, E.J. Candès, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on Optimization, 20 (2010), pp. 1956–1982.
[9] E.J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, 9 (2009), pp. 717–772.
[10] W. Dai, E. Kerman, and O. Milenkovic, *A geometric approach to low-rank matrix completion*, Information Theory, IEEE Transactions on, 58 (2012), pp. 237–247.

[11] W. Dai, O. Milenkovic, and E. Kerman, *Subspace evolution and transfer (SET) for low-rank matrix completion*, Signal Processing, IEEE Transactions on, PP (2011), p. 1.
[12] M. Fornasier, H. Rauhut, and R. Ward, *Low-rank matrix recovery via iteratively reweighted least squares minimization*, SIAM Journal on Optimization, 21 (2011), p. 1614.
[13] N. Gillis and F. Glineur, *Low-rank matrix approximation with weights or missing data is NP-hard*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), p. 1149.
[14] R.H. Keshavan and A. Montanari, *Regularization for matrix completion*, in Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on, IEEE, 2010, pp. 1503–1507.
[15] R.H. Keshavan, A. Montanari, and S. Oh, *Low-rank matrix completion with noisy observations: a quantitative comparison*, in Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on, IEEE, 2009, pp. 1216–1222.
[16] R.H. Keshavan and S. Oh, *OptSpace: A gradient descent algorithm on the Grassman manifold for matrix completion*, Arxiv preprint arXiv:0910.5260 v2, (2009).
[17] R.M. Larsen, *PROPACK–Software for large and sparse SVD calculations*, Available online. URL http://sun. stanford. edu/rmunk/PROPACK, (2005).
[18] K. Lee and Y. Bresler, *ADMiRA: Atomic decomposition for minimum rank approximation*, Information Theory, IEEE Transactions on, 56 (2010), pp. 4402–4416.
[19] Kurt Leichtweiss, *Zur Riemannschen Geometrie in Grassmannschen Mannigfaltigkeiten*, Math. Z., 76 (1961), pp. 334–366.
[20] G. Meyer, S. Bonnabel, and R. Sepulchre, *Linear regression under fixed-rank constraints: a Riemannian approach*, in 28th International Conference on Machine Learning, ICML, 2011.
[21] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre, *Low-rank optimization with trace norm penalty*, Arxiv preprint arXiv:1112.2318, (2011).
[22] M. Tao and X. Yuan, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM Journal on Optimization, 21 (2011), p. 57.
[23] L.N. Trefethen and D. Bau, *Numerical linear algebra*, Society for Industrial Mathematics, 1997.
[24] B. Vandereycken, *Low-rank matrix completion by riemannian optimization*, tech. report, ANCHP-MATHICSE, Mathematics Section, École Polytechnique Fédérale de Lausanne, 2011.
[25] Z. Wen, W. Yin, and Y. Zhang, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, tech. report, Rice University, 2010. CAAM Technical Report TR10-07.