The Geometry of Weighted Low-Rank Approximations

Jonathan H. Manton, *Member, IEEE*, Robert Mahony, and Yingbo Hua, *Fellow, IEEE*

***Abstract*—The low-rank approximation problem is to approximate optimally, with respect to some norm, a matrix by one of the same dimension but smaller rank. It is known that under the Frobenius norm, the best low-rank approximation can be found by using the singular value decomposition (SVD). Although this is no longer true under weighted norms in general, it is demonstrated here that the weighted low-rank approximation problem can be solved by finding the subspace that minimizes a particular cost function. A number of advantages of this parameterization over the traditional parameterization are elucidated. Finding the minimizing subspace is equivalent to minimizing a cost function on the Grassmann manifold. A general framework for constructing optimization algorithms on manifolds is presented and it is shown that existing algorithms in the literature are special cases of this framework. Within this framework, two novel algorithms (a steepest descent algorithm and a Newton-like algorithm) are derived for solving the weighted low-rank approximation problem. They are compared with other algorithms for low-rank approximation as well as with other algorithms for minimizing a cost function on a Grassmann manifold.**

***Index Terms*—Grassman manifold, low-rank approximations, optimization on manifolds, reduced rank signal processing.**

I. Introduction

THE weighted low-rank approximation problem is to compute

$$\arg \min_{\substack{R \\ \text{rank}\{R\} \le r}} \|X-R\|_Q^2, \qquad \|X-R\|_Q^2 = \text{vec}\{X-R\}^T Q\, \text{vec}\{X-R\} \quad (1)$$

for a given data matrix $X \in \mathbb{R}^{n\times m}$ and positive definite symmetric weighting matrix $Q \in \mathbb{R}^{mn\times mn}$. Here, $\text{vec}\{\cdot\}$ denotes the vec operator [18], and it is important to note that the norm $\|\cdot\|_Q$ is more general than the usual weighted norm $\|X\|_Q^2 = \text{tr}\{X^TQX\}$, where $\text{tr}\{\cdot\}$ is the trace operator. The minimizing R in (1) is the best rank r approximation of X under the norm $\|\cdot\|_Q$. If Q is the identity matrix, denoted $Q = I$, then (1) reduces to the well-studied unweighted low-rank approximation problem. This paper analyzes the geometry of the low-rank approximation problem, drawing connections between the weighted and unweighted cases. It then uses this analysis to construct efficient algorithms for locally minimizing (1).

The weighted low-rank approximation problem has received less attention in the literature than the unweighted low-rank approximation problem [16]. Presumably, this is because a closed-form solution does not exist for the weighted low-rank approximation problem in general. Furthermore, existing algorithms for the weighted case only converge to a local minimum of (1) in general. Despite this though, the following applications illustrate that it is still beneficial to consider the weighted low-rank approximation problem.

A. Applications

One application that benefits from the use of a weighted low rank matrix approximation is the two-dimensional (2-D) filter design problem. The approach in [17] and [30] to the 2-D filter design problem is to start with a matrix X whose elements correspond to samples of the desired frequency response and then decompose the 2-D design task into a set of simpler one-dimensional design tasks by applying the singular value decomposition (SVD) to X. A disadvantage of using the SVD to decompose the desired frequency response X is that it treats all entries of X equally, which in some cases leads to degraded designs. In order to discriminate between the important and unimportant elements of X, the idea of replacing the SVD with a weighted low-rank approximation was proposed in [16], [27]. (See [16] for a design example.)

Although finding the global minimum of (1) is ideal, it may still be the case that a filter design resulting from finding a local minimum of (1) outperforms the unweighted filter design. Furthermore, since the performance of the resulting filter is readily measurable, if it so happens that the weighted design is worse than the unweighted design, the weighted low-rank approximation can be recomputed with a different initial condition in the hopes that a better local minimum of (1) will be found.

A second application requiring the solution of a weighted low-rank approximation problem is the convolutive reduced-rank Wiener filtering problem [20]. Motivated by the same idea of using a double minimization (2) to reformulate the original optimization problem (1), it was shown in [20] that the convolutive reduced-rank Wiener filter can be found by solving a weighted low-rank approximation problem. As in the 2-D filter design problem, because the mean-square error of the resulting convolutive reduced-rank Wiener filter can be readily

Manuscript received August 31, 2000; revised October 3, 2002. This work was supported by the Australian Research Council. The first author is associated with the ARC Special Research Centre for Ultra-Broadband Information Networks (CUBIN). The associate editor coordinating the review of this paper and approving it for publication was Prof. Jian Li.

J. H. Manton is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria, Australia (e-mail: j.manton@ee.mu.oz.au).

R. Mahony is with the Department of Engineering, Australian National University, Canberra, Australia (e-mail: mahony@ieee.org).

Y. Hua is with the Department of Electrical Engineering, University of California, Riverside, CA, 92521 USA (e-mail: yhua@ee.ucr.edu).

Digital Object Identifier 10.1109/TSP.2002.807002

1053-587X/03$17.00 © 2003 IEEE

calculated and compared with that of the standard (nonconvolutive) reduced-rank Wiener filter [3], [14], even a suboptimal solution of (1) can lead to a convolutive reduced-rank Wiener filter whose performance is verifiably better than that of the standard reduced-rank Wiener filter.

B. Known Properties and Related Work

In (1), if $Q = I$, then $\|\cdot\|_Q$ is the Frobenius norm. The low-rank approximation problem with respect to the Frobenius norm was first studied by Eckart and Young [7]. They proved that if $X = U\Sigma V^T$ is the compact SVD [12] of X, then the best rank r approximation of X is $R = U\Sigma_r V^T$, where Σ_r is obtained from Σ by setting all but the first r singular values to zero. This result is commonly referred to as the Eckart–Young–Mirsky Theorem (Mirsky [21] proved the result also holds under the 2-norm).

The best unweighted rank r approximation R is also readily computed from the eigenvector decomposition (EVD) of $X^T X$. If $V \in \mathbb{R}^{m\times r}$ contains the normalized (so that $V^T V = I$) eigenvectors associated with the r largest eigenvalues of $X^T X$, then $R = XVV^T$. This follows almost immediately from the Eckart–Young–Mirsky Theorem and the fact that the eigenvectors of $X^T X$ are the right singular vectors of X. In a sense to be made precise later, this result is generalized in the present paper to the weighted case.

An alternative to performing an SVD or EVD and one that immediately extends to the weighted case, is to first over-parameterize the problem to remove the rank constraint and then apply an alternating projection algorithm [16]. Specifically, the algorithm proposed in [16] works as follows. Replace R in (1) with the matrix product AB, where $A \in \mathbb{R}^{n\times r}$, and $B \in \mathbb{R}^{r\times m}$. Fix a value for A and minimize over B, then fix B, minimize over A, and repeat until the product AB converges. It can be shown that, in general, $R = AB$ converges to a local minimum of (1).

Remark : If $Q = I$ in (1), then it is known [13] that (1) has no local minimum other than the global minimum. It does, however, have saddle points.

Copious works deal with the unweighted low-rank approximation problem and applications thereof. This is because appropriate usage of reduced-rank approximations can result in increased computational efficiency and robustness against noise and model errors. Fundamental results on optimal reduced-rank estimators and filters can be found in [3], [9], [10], [15], [23]–[26], [28], and [29]. Other algorithms for solving the (adaptive) unweighted low-rank approximation problem include [6], [14]. However, the only algorithm the authors are aware of for solving the weighted low-rank approximation problem is the alternating projection algorithm presented in [16].

A variant of the low-rank approximation problem appears in [4] and references therein. Specifically, [4] uses a modified inverse power method to solve (1) under the extra constraints

i) $r = m - 1$;
ii) R is restricted to have some affine structure.
iii) Q is diagonal. This variant is discussed further in Section II.

C. Contributions

The main contributions of this paper can be summarized as follows.

- We introduce a novel reformulation of the weighted low-rank approximation problem, which is more natural than the $R = AB$ reformulation traditionally used in rank-reduced problems [16], [28].
- We determine conditions on the weighting matrix Q for a closed-form solution of (1) to exist.
- We derive efficient numerical algorithms that converge to a local minimum of (1).
- We compare the existing alternating projection algorithm with the novel algorithms proposed here for solving (1).

The other contributions, arising from the reformulation of the weighted low-rank approximation problem as a constrained optimization problem on the Grassmann manifold, are the following.

- We derive a general framework for minimizing a cost function on a Grassmann manifold.
- We prove that the algorithms in [8] are a special case of this framework.
- We discuss the advantages this framework has over the narrower Riemannian framework in [8] and, in particular, why it is misleading to interpret the algorithms proposed here as approximations of the Riemannian-based algorithms in [8].

These contributions are now discussed in relation to existing results in the literature.

As already mentioned, the traditional approach to reduced-rank problems is to write the rank r matrix R as the product $R = AB$ of two matrices, where A has r columns, and B has r rows. The potential disadvantage of this approach is that the decomposition $R = AB$ is not unique, or equivalently, too many parameters are used to represent rank r matrices. The novel idea in this paper is to use a parameterization that is one-to-one, thus reducing the number of parameters and accordingly reducing the dimension of the optimization problem. This is achieved by reformulating (1) as an unconstrained optimization problem on a Grassmann manifold. (A Grassmann manifold is the collection of all subspaces of a given dimension [8], [13].) The authors believe this reformulation to be more natural than the $R = AB$ reformulation not only because the parameterization is one-to-one but because conditions for (1) to have a closed-form solution become readily apparent as well.

The reformulated problem of minimizing a cost function on a Grassmann manifold can be solved numerically using the recent algorithms in [8], and indeed, it is candidly stated that such an approach leads to algorithms that perform comparably with the proposed algorithms in this paper. Nevertheless, for reasons given in Section III, the approach taken here is to derive first a more general framework for optimizing a cost function on a manifold and then specialize it to the weighted low-rank approximation problem. Although the resulting algorithms might be interpreted by some as "flat space approximations" of the algorithms in [8], Section III explains why this interpretation is misleading; the algorithms in [8] can just as well be interpreted as approximations of the algorithms proposed here.

Remark : If $Q = I$, then the resulting cost function on the Grassmann manifold takes a special form (specifically, it is a generalized Rayleigh quotient cost function) for which dedicated minimization algorithms have been proposed in [1] and [5]. This specific case has also been studied in detail in [8].

The algorithms proposed for solving (1), which are a modified steepest descent method and a modified Newton method, are shown to have the following advantages over the alternating projection algorithm in [16]. The alternating projection algorithm asymptotically has a linear rate of convergence,[1] meaning that often, a significant number of iterations are required to achieve an acceptable accuracy. The modified Newton method presented here overcomes this problem since, asymptotically, it has a quadratic rate of convergence in general and a cubic rate of convergence if $Q = I$. Furthermore, simulations show that closely spaced eigenvalues of X^TX adversely affect the convergence rate of the alternating projection algorithm, whereas they do not seem to affect the algorithms in this paper. [This observation is mathematically substantiated in [19] for the special case of $Q = I$ and $r = 1$ or $r = m - 1$ in (1).] Yet another advantage is that here the optimization algorithms work over an $r(m - r)$ dimensional space (assuming $n \geq m$), whereas the alternating projection algorithm works over an $nr+mr$-dimensional space [see (1) for the definitions of m, n, r]. When n is much larger than m and r is small, the reduction in dimension is very significant. For instance, if $n = 100$, $m = 10$, and $r = 3$, then $nr + mr = 330$, yet $r(m - r) = 21$.

D. Organization of Paper

The rest of this paper is organized as follows. Section II shows how the low-rank approximation problem can be solved by first computing the minimizing subspace of a certain cost function. It also derives conditions for (1) to have a closed form solution. Section III derives a general framework for finding a minimizing subspace of a cost function. It highlights the advantages of this more general framework over the Riemannian-based framework presented recently in [8]. This framework is used to derive novel steepest descent algorithms in Section IV and Newton methods in Section V for solving the weighted low-rank approximation problem. These algorithms are not standard steepest descent and Newton algorithms; the cost function changes at each iteration. A numerical study in Section VI demonstrates that the algorithms are superior to the classical alternating projection algorithm, which is the only other algorithm the authors' are aware of for solving the weighted low-rank approximation problem. All proofs are relegated to Appendix A.

II. Weighted Low-Rank Approximation

This section derives a novel reformulation of the low-rank approximation problem (1). This reformulation is used in this section to derive conditions for (1) to have a closed-form solution, and it is used in subsequent sections to derive efficient algorithms for converging to a local minimum of (1). Connections with the Riemannian SVD [4] are also discussed.

Without loss of generality, it is assumed throughout that $n \geq m$, where n and m are the number of rows and columns of the data matrix X. [If $n < m$, simply replace X by X^T and adjust Q in (1) accordingly.]

The underlying idea in this paper is to reformulate (1) as the double-minimization

$$\min_{\substack{N\in\mathbb{R}^{m\times(m-r)}\\ N^TN=I}} \left(\min_{\substack{R\in\mathbb{R}^{n\times m}\\ RN=0}} \|X-R\|_Q^2 \right). \tag{2}$$

Close inspection shows that if N and R are the minimizing arguments of the two minimizations in (2), then R is the solution of the low-rank approximation problem (1); the restriction $RN = 0$ enforces the constraint $\text{rank}\{R\} \leq r$ since every column of N must belong to the null space of R. Theorem 1 below shows that the inner minimization has a closed-form solution. Moreover, because the inner minimization depends only on the span of the columns of N and not on the individual elements of N, it will be shown in subsequent sections that the outer minimization reduces to one of dimension $r(m - r)$.

Theorem 1: For any given data matrix $X \in \mathbb{R}^{n\times m}$ and positive definite symmetric weighting matrix $Q \in \mathbb{R}^{nm\times nm}$, define

$$f(N) = \min_{\substack{R\in\mathbb{R}^{n\times m}\\ RN=0}} \|X-R\|_Q^2 \tag{3}$$

where $N \in \mathbb{R}^{m\times(m-r)}$ and $\|\cdot\|_Q^2$ is defined in (1). Then, the minimizing R is given by

$$\begin{aligned}\text{vec}\{R\} = {} & \text{vec}\{X\} - Q^{-1}(N\otimes I_n) \\ & \cdot \left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(N\otimes I_n)^T\text{vec}\{X\}\end{aligned} \tag{4}$$

where $\otimes$ is Kronecker's product [18]. Furthermore, $f(N)$ is given by

$$\begin{aligned}f(N) = {} & \text{vec}\{X\}^T(N\otimes I_n) \\ & \cdot \left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(N\otimes I_n)^T\text{vec}\{X\}\end{aligned} \tag{5}$$

and depends only on the range space of N; for any invertible matrix $S \in \mathbb{R}^{(m-r)\times(m-r)}$, $f(NS) = f(N)$.

By considering the unweighted case $Q = I$, it will be seen that (2) is the generalization of the EVD approach, which is defined in Section I-B, to the weighted case.

Corollary 2: Define $f(N)$ as in (3). If $Q = I$ in (3), then (5) becomes

$$f(N) = \text{tr}\left\{N^TX^TXN\left(N^TN\right)^{-1}\right\}. \tag{6}$$

If $Q = I$ and $N^TN = I$, then (4) and (5) become

$$R = X - XNN^T \tag{7}$$

$$f(N) = \text{tr}\left\{N^TX^TXN\right\}. \tag{8}$$

[1] A linear rate of convergence means that the logarithm of the error decreases linearly or, equivalently, that the number of correct digits in the answer increases by approximately a fixed amount per iteration. Similarly, a quadratic rate of convergence means that the logarithm of the error decreases quadratically, implying that the number of correct digits approximately doubles each iteration. It is a standard result that steepest descent methods asymptotically have a linear rate of convergence, whereas Newton methods asymptotically have a quadratic rate of convergence [22].

The cost function (6) is called the *generalized Rayleigh quotient* [13] since it is a generalization of the usual Rayleigh quotient [12]. Furthermore, it is known that the minimum of (6), or of (8) subject to $N^TN = I$, occurs when the columns of N correspond to the $m - r$ smallest eigenvectors of X^TX. That is, (2) is precisely the EVD approach; if the columns of $V \in \mathbb{R}^{m\times r}$ are the r largest eigenvectors of X^TX and if the columns of N are the $m - r$ smallest eigenvectors, then $I - NN^T = VV^T$; therefore, (7) becomes $R = XVV^T$.

The traditional approach [16], [28] to reduced-rank problems is to enforce the rank r constraint on R by replacing R by AB, where A has r columns and B has r rows. The interpretation is that matrices with rank at most r are being parameterised by the many-to-one map $p(A, B) = AB$. Reasons for believing the reformulation (2) to be more natural than the traditional reformulation are now given.

1) The implicit mapping in (2) from a null space represented by N to a matrix $R = \arg\min_{\substack{R\in\mathbb{R}^{n\times m}\\ RN=0}} \|X-R\|_Q^2$ of rank at most r is a one-to-one mapping.[2]
2) As shown above, if $Q = I$, then (2) is equivalent to the EVD approach for computing a low-rank approximation.
3) The fact that the SVD or EVD can be used to solve (1) for certain weighting matrices Q is not apparent from the $R = AB$ reformulation. However, as is shown below, conditions on Q for (2) to have a solution in terms of an SVD or EVD are easily found.

If Q is chosen so that (2) is equivalent to the minimization of a generalized Rayleigh quotient, then (1) has a closed-form solution in terms of an SVD or EVD. Such a Q must make (5) a quadratic function when N is appropriately restricted, cf., (8). For (5) to be quadratic, it is necessary to "remove" the $[\cdot]^{-1}$ term. This can be done whenever Q is of the form $Q = Q_1 \otimes Q_2$. Note that if $Q = Q_1 \otimes Q_2$, then $\|X\|_Q^2 = \text{tr}\left\{X^TQ_2XQ_1^T\right\}$.

Theorem 3: In (1), if $Q = Q_1 \otimes Q_2$, where $Q_1 \in \mathbb{R}^{m\times m}$ and $Q_2 \in \mathbb{R}^{n\times n}$ are both positive definite and symmetric, then the solution R of (1) is given by the following closed-form expression. Let $Q_2^{1/2}XQ_1^{1/2} = U\Sigma V^T$ be the compact SVD [12] of $Q_2^{1/2}XQ_1^{1/2}$, where $Q_1^{1/2}$ is the unique positive definite symmetric matrix such that $Q_1^{1/2}Q_1^{1/2} = Q_1$ and similarly for $Q_2^{1/2}$. Then, $R = Q_2^{-1/2}U\Sigma_rV^TQ_1^{-1/2}$, where Σ_r is obtained from Σ by setting all but the first r singular values to zero.

For completeness, connections with the Riemannian SVD [4] are discussed briefly. The Riemannian SVD can be used to solve (1) only in the special case of a rank one reduction ($r = m - 1$). (It also requires Q to be diagonal.) If $Q = I$, then the algorithm in [4] for computing the Riemannian SVD reduces to the standard inverse power method, whereas the steepest descent algorithm in Section IV-B specializes to the algorithm in [19]. As shown in [19], the steepest descent algorithm is preferable to the inverse power method since it is not sensitive to closely spaced eigenvalues. Furthermore, the Newton method in Section V asymptotically has a cubic rate of convergence when $Q = I$, whereas the inverse power method only has a linear rate of convergence asymptotically.

[2]Specifically, it induces a one-to-one mapping from points on the Grassmann manifold to matrices with rank at most r. Since the mapping is not onto, it uses even fewer parameters than the traditional $R = AB$ parameterization.

More detailed comparisons with the Riemannian SVD have not been made because the Riemannian SVD is not used in practice to solve (1). Instead, it is used to solve (1) subject to the extra constraint that the rank-reduced matrix R has a particular affine structure [4]. Although not pursued here, it may be possible to incorporate this structural constraint into (2), leading to modified steepest descent and Newton methods having superior performance to the modified Inverse Power method in [4].

III. Optimization Algorithms on Grassmann Manifolds

The previous section showed that (1) can be solved by first finding the matrix N, which minimizes the cost function (5). Directly minimizing $f(N)$ is an $m(m - r)$ dimensional optimization problem because N is m by $m - r$. However, $f(N)$ only depends on the range space of N and not on the individual elements of N. As recognized in [8], this symmetry can be exploited to reduce the dimension of the optimization problem to $r(m-r)$ parameters (see Section III-A for an elementary proof). Moreover, the algorithms in [8] can be used to minimize $f(N)$ (once the necessary derivatives have been calculated), thus resulting in efficient algorithms for solving the weighted low-rank approximation problem.

For reasons given later though, this paper prefers to use the algorithm in Section III-A for minimizing $f(N)$. The motivation for considering alternatives to the algorithms in [8] is that [8] introduces an artificial structure, namely, a Riemannian structure, into the optimization problem that, depending on the actual function to be minimized, may or may not be detrimental. Specifically, unless $f(N)$ possesses properties that make it natural or desirable to introduce a Riemannian structure, there is no compelling reason to do so; see Section III-B. Therefore, the algorithm in Section III-A takes the liberty of introducing a different artificial structure into the constraint surface that serendipitously appears to be better suited to the specific cost function (5). It is candidly stated, however, that the improvement over the algorithms in [8] for the specific cost function (5) appears to be relatively minor and has not been rigorously established; the reasons then for presenting this alternative approach are that this approach is more accessible to readers since it does not require knowledge of differential geometry, and moreover, the secondary aim of this paper is to correct the possible misconception that only geodesic-based optimization algorithms are "natural" or "correct" algorithms.

A. Elementary Optimization Algorithm

This section derives an algorithm for the constrained minimization of a function $f(N)$ subject to $N^TN = I$ and under the assumption that the value of f at any point N depends only on the range space of N. The algorithm itself is not new but its interpretation is; previously, the algorithm was thought to be a "flat space approximation" of the geodesic-based algorithms in [8], whereas Section III-B shows that it is just as valid as the algorithms in [8].

Henceforth, $N_\perp$ is used to denote the orthogonal complement of N, that is, $N_\perp \in \mathbb{R}^{m\times r}$ is any full column rank matrix satisfying $N^TN_\perp = 0$. Since $N_\perp$ is not uniquely defined, implicit in any statement involving $N_\perp$ is that the statement holds for any fixed choice of $N_\perp$.

Take an arbitrary matrix $N \in \mathbb{R}^{m\times(m-r)}$ satisfying $N^TN = I$, and consider a perturbation matrix $Z \in \mathbb{R}^{m\times(m-r)}$. For certain Z, the range space of $N+Z$ is the same as the range space of N (written range$\{N+Z\}$ = range$\{N\}$), and in this case, $f(N+Z) = f(N)$. It is therefore not necessary to consider all $m(m-r)$ search directions when trying to minimize $f(N)$.

For fixed N and $N_\perp$, a perturbation $Z \in \mathbb{R}^{m\times(m-r)}$ uniquely decomposes as $Z = NL + N_\perp K$, where $L \in \mathbb{R}^{(m-r)\times(m-r)}$ and $K \in \mathbb{R}^{r\times(m-r)}$. Since range$\{N+NL\} \subset$ range$\{N\}$, it suffices to consider only search directions $Z = N_\perp K$. It is necessary to consider these directions because range$\{N + N_\perp K_1\}$ = range$\{N+N_\perp K_2\}$ implies $K_1 = K_2$. Since K has $r(m-r)$ elements, minimizing $f(N)$ is an $r(m-r)$-dimensional problem.

The above suggests the following iterative minimization scheme.

Algorithm 4
Let $f(N)$ be a function that only depends on the range of N. A locally minimizing N, subject to $N^TN = I$, can be found as follows.
1) Choose a starting position N satisfying $N^TN = I$.
2) Choose $N_\perp$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$. Use the local parameterization from $K \in \mathbb{R}^{r\times(m-r)}$ into $\mathbb{R}^{m\times(m-r)}$ defined by

$$\phi(K) = N + N_\perp K \tag{9}$$

to form the local cost function

$$g(K) = f(\phi(K)) = f(N + N_\perp K). \tag{10}$$

3) By applying a standard optimization technique (such as steepest descent or Newton's method) to $g(K)$ at the point $K = 0$, compute a descent step ΔK.
4) Set N to any matrix such that range$\{N\}$ = range$\{\phi(\Delta K)\}$ and $N^TN = I$. (Gram–Schmidt orthogonalization or the QR algorithm [12] can be used to compute such an N.)
5. Repeat steps 2–4 until convergence.

The descent step ΔK typically is a function of the first and second derivatives of $g(K)$. The following proposition gives formulae for the first and second differentials of (10). (For the definition of differentials, see [18]. See also Example 7 in Section III-B.)

Proposition 5: Fix N and $N_\perp$, and define $g(K)$ as in (10). If $df(dN)$ and $d^2f(dN, dN)$ are the first and second differentials of f about the point N, then the first and second differentials of $g(K)$ about $K = 0$ are given by

$$dg(dK) = df(N_\perp dK) \tag{11}$$

$$d^2g(dK, dK) = d^2f(N_\perp dK, N_\perp dK). \tag{12}$$

Remark: A consequence of Proposition 5 is that $N_\perp$ will never appear on its own but always in the form $N_\perp N_\perp^T$ in Algorithm 4. This fact can be exploited for a more efficient implementation (cf., [8]) but is not done here for clarity of presentation and because computing the derivatives of $f(N)$ in (5) and not computing $N_\perp$ is the most complex computation per iteration.

B. Discussion

This section first states a general framework for deriving optimization on manifold algorithms and then shows that the Riemannian framework in [8] is a special case of this more general framework. This general framework is used to explain the similarities and differences between Algorithm 4 and the algorithms in [8]. Readers only interested in the low-rank approximation problem are advised to skip this section.

Minimizing a function $f(N)$ whose value only depends on the range of N can be posed as an optimization problem on a Grassmann manifold [8]. There is no unique way of generalizing Newton's method in Euclidean space to a Newton method on a manifold. One way is to continue to use Newton's formula by treating the first and second derivatives in Newton's formula as the gradient and Hessian of the cost function on the manifold; this necessitates endowing the manifold with a Riemannian structure and is the approach taken in [8]. Another way is to use the manifold structure to form a local cost function at each iteration and apply Newton's method to this local cost function; this is the approach taken here and is discussed in detail below. Yet another way is to generalize the property that a Newton method approximates the cost function by a quadratic function at each iteration and then moves to the minimum of this quadratic approximation; this generalization is different from the above two generalizations and is a topic for future research.

The general framework (but not the only possible framework) proposed here for minimizing a function f on an n dimensional manifold M is the following. (For this section only, the symbols n and p have a new meaning.) For every point p on the manifold M, choose a particular local parameterization[3] $\phi_p : \mathbb{R}^n \to M$ centred on p, that is, ϕ_p is a diffeomorphism, and $\phi_p(0) = p$. Different choices of local parameterizations lead to different optimization algorithms in general. Given the current iterate $p^{(i)}$, the next iterate $p^{(i+1)}$ is obtained as follows. Define the local cost function $g(z) = f(\phi_{p^{(i)}}(z))$ that maps $\mathbb{R}^n$ to $\mathbb{R}$. Apply to g a single iteration of an ordinary optimization algorithm (such as Newton's method in Euclidean space) at the origin (recall $\phi_{p^{(i)}}(0) = p^{(i)}$) to obtain a z such that $g(z) < g(0)$. Finally, set $p^{(i+1)} = \phi_{p^{(i)}}(z)$.

For brevity, any algorithm which can be written in the above form (with a Newton method used to find z) is called a *true Newton method*. Clearly, Algorithm 4 (with Step 3 a Newton step) is a true Newton method.

An interesting and nontrivial observation is that the Newton algorithm in [8] is also a true Newton method. Specifically, the Newton algorithm in [8] is obtained from Algorithm 4 by

[3]In more general cases, the domain of ϕ_p can be chosen to be an open subset of $\mathbb{R}^n$ rather than the whole of $\mathbb{R}^n$.

making Step 3 a Newton step and by replacing the local parameterization (9) with the alternative local parameterization

$$\phi(K) = NV\cos(\Sigma) + U\sin(\Sigma), \qquad N_\perp K = U\Sigma V^T \quad (13)$$

where $U \in \mathbb{R}^{m\times(m-r)}$, $\Sigma \in \mathbb{R}^{(m-r)\times(m-r)}$, and $V \in \mathbb{R}^{(m-r)\times(m-r)}$ are obtained from the compact SVD of $N_\perp K$ (that is, $U^TU = V^TV = I$, and Σ is diagonal). A proof of this follows from the facts stated in the proof of Proposition 6.

Under the local parameterization (13), the local cost function (10) becomes

$$g(K) = f(NV\cos(\Sigma) + U\sin(\Sigma)), \qquad N_\perp K = U\Sigma V^T. \quad (14)$$

Before arguing that Algorithm 4 is just as valid an algorithm as those in [8], the derivatives of (14) are calculated.

Proposition 6: Let $f(N)$ be a cost function such that $f(N_1) = f(N_2)$ if range$\{N_1\}$ = range$\{N_2\}$ and $N_1^TN_1 = N_2^TN_2 = I$. Choose N and $N_\perp$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$ and define $g(K)$, as in (14). If $df(dN)$ and $d^2f(dN, dN)$ are the first and second differentials of f about the point N, then the first and second differentials of $g(K)$ about $K = 0$ are given by

$$dg(dK) = df\,(N_\perp dK) \quad (15)$$

$$d^2g(dK, dK) = d^2f\,(N_\perp dK, N_\perp dK) - df\left(NdK^TdK\right). \quad (16)$$

The following example clarifies the notation in Proposition 6.

Example 7: The first and second differentials of $g(K)$ will be computed when $f(N)$ is as defined in (3) and $Q = I$. Since Proposition 6 assumes $N^TN = I$ and, furthermore, only requires $f(NS) = f(N)$ to hold for orthogonal matrices S and not for invertible matrices S, (8) can be used instead of (6). Thus, define $f(N) = \text{tr}\{N^TX^TXN\}$. Its first differential about N is $df(dN) = 2\text{tr}\{N^TX^TXdN\}$. Its second differential is $d^2f(dN, dN) = 2\text{tr}\{dN^TX^TXdN\}$. Applying Proposition 6 shows that the first and second differentials of (14) about $K = 0$ are

$$dg(dK) = 2\text{tr}\left\{N^TX^TXN_\perp dK\right\} \quad (17)$$

$$d^2g(dK, dK) = 2\text{tr}\left\{dK^TN_\perp^TX^TXN_\perp dK\right\} - 2\text{tr}\left\{N^TX^TXNdK^TdK\right\} \quad (18)$$

$$= 2\text{tr}\left\{dK^TN_\perp^TX^TXN_\perp dK - dKN^TX^TXNdK^T\right\}. \quad (19)$$

Theorem 8 proves that the first and second derivatives of (10) and (14) about $K = 0$ are the same.

Theorem 8: Let $f(N)$ be a cost function such that $f(N_1) = f(N_2)$ if range$\{N_1\}$ = range$\{N_2\}$ and N_1, N_2 have full column rank. Choose N and $N_\perp$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$. Then, the first and second differentials about $K = 0$ of $g(K)$ defined in (10) are identical to the first and second differentials about $K = 0$ of $g(K)$ defined in (14).

Theorem 8 shows that the step ΔK will be the same for both Algorithm 4, with Step 3 a Newton step and the Newton algorithm in [8]. The only difference between the two algorithms is that Algorithm 4 moves along the straight line $N + N_\perp\Delta K$ rather than along the geodesic $NV\cos(\Sigma) + U\sin(\Sigma)$, where $U\Sigma V^T = N_\perp\Delta K$. Therefore, it is possible to derive Algorithm 4 by starting with the algorithm in [8] and approximating geodesics by straight lines and moreover; this makes Algorithm 4 appear to be a "flat space approximation" of the algorithms in [8].

However, the algorithms in [8] can equally well be thought of as approximations of Algorithm 4; replace the straight line parameterization (10) by the geodesic approximation (14). The key point though is that thinking of either algorithm as an approximation of the other is misleading because the term "approximation" has the connotation of inferiority, yet both algorithms are true Newton methods and neither can claim superiority in general; for some cost functions, the algorithms in [8] may converge more quickly,[4] whereas for others, Algorithm 4 may be faster. The following simplified example in Euclidean space is used to explain this phenomenon.

Consider the two cost functions $f_1(x, y) = (x-2)^2+(y-1)^2$ and $f_2(r, \theta) = (r\cos\theta-2)^2+(r\sin\theta-1)^2$. Newton's method applied to f_1 finds the exact solution after a single iteration. However, it requires an infinite number of iterations to converge to the exact solution if it is applied to f_2. This is because the standard Newton method assumes that the cost function is approximately quadratic in Cartesian coordinates. Conversely, a Newton method in polar coordinates converges in one iteration when applied to f_2. Clearly, the Newton algorithm in Cartesian coordinates and the Newton algorithm in polar coordinates are equally valid Newton algorithms, and neither can claim superiority.

The difference between Algorithm 4 and the algorithms in [8] is analogous to the above example; they merely use different coordinate systems (cf., (10) and (14)). Which algorithm is better depends on the particular cost function to minimize. (Roughly speaking, for a given cost function f, if the local cost function (10) centred at the minimum of f more closely resembles a quadratic function than (14) does, then (10) should be used instead of (14).)

Last, to refute any claim that the algorithms in [8] are superior because they correctly exploit the geometry of the Grassmann manifold, it is emphasised that the "geometry" in [8] is an *artificial* geometry. In the original constrained optimization problem, only the constraint set $M = \{N : N^TN = I\}$ is given. Making M into a manifold is already adding an artificial structure (a topology and an atlas), yet there is a clear choice here; making M a Stiefel or Grassmann manifold means that if f is smooth as a function in Euclidean space, then it remains smooth as a function on the Stiefel or Grassmann manifold. *However, if nothing else is known about f, then there is no compelling reason to go further and endow the constraint set M with a metric, making it a Riemannian manifold.* In other words, using the artificial Riemannian structure is conceptually no better or no worse than using the artificial local parameterization (10).

[4]Note that the asymptotic rate of convergence will be the same for both algorithms (e.g., quadratic for Newton methods) but the constant of proportionality will in general be different; one algorithm may require twice as many iterations as the other to achieve the same level of accuracy, for instance.

Remark: Note that (10) is a canonical parameterization of the Grassmann manifold known as homogeneous coordinates in the literature; if the cost function is not specified in advance, then an arbitrary choice must be made, and the choice (10) is a natural one, as is (14).

IV. First-Order Descent Methods

This section presents algorithms for solving the weighted low-rank approximation problem (1). The classical alternating projection algorithm is derived in Section IV-A, whereas novel steepest descent algorithms are proposed in Section IV-B. It is important to note that these steepest descent algorithms are not standard descent algorithms; the cost function changes at each iteration (see Algorithm 4). The performance of these algorithms is discussed in Section VI.

The computational complexity of each algorithm is calculated for an arbitrary weighting matrix, a diagonal weighting matrix, and the identity weighting matrix (unweighted case). It is expected that in many applications the weighting matrix Q will be diagonal. Indeed, taking Q to be diagonal corresponds to considering the weighted norm in [16].

A. Alternating Projections

An alternating projection algorithm was proposed in [16] for finding the weighted low-rank approximation of a matrix. Since [16] used a different[5] (and less general) weighting function, their notation was somewhat cumbersome. Proposition 9 derives a compact form of the alternating projection algorithm.

Proposition 9: Let $X \in \mathbb{R}^{n\times m}$ be an arbitrary matrix. Then for a fixed $A \in \mathbb{R}^{n\times r}$, the B which minimizes $\|X - AB\|_Q^2$ is given by

$$\text{vec}\{B\} = \left[(I_m \otimes A)^T Q (I_m \otimes A)\right]^{-1} (I_m \otimes A)^T Q \,\text{vec}\{X\}. \tag{20}$$

Similarly, for a fixed $B \in \mathbb{R}^{r\times m}$, the A, which minimizes $\|X - AB\|_Q^2$ is given by

$$\text{vec}\{A\} = \left[(B \otimes I_n) Q (B \otimes I_n)^T\right]^{-1} (B \otimes I_n) Q \,\text{vec}\{X\}. \tag{21}$$

Based on Proposition 9, the alternating projection algorithm is as follows. Initialize A randomly. Use (20) to compute B. Use (21) to compute a new A. Repeat until convergence. The (locally) best rank r approximation of X is then $R = AB$.

When $Q = I$, (20) and (21) reduce to $B = (A^TA)^{-1}A^TX$ and $A = XB^T(BB^T)^{-1}$, respectively.

Complexity: One iteration of the alternating projection algorithm requires $O(n^2mr^2 + n^3r^3)$ flops. If Q is diagonal, only $O(nmr^2)$ flops are required. (These flop counts are obtained by exploiting the block structure introduced by the Kronecker product.) If $Q = I$, there are $O(nmr)$ flops per iteration.

B. Steepest Descent

This section first derives an expression for the steepest descent direction of the local cost function $g(K)$ in Algorithm 4. It then uses this expression to derive steepest descent algorithms for solving the low-rank approximation problem (1).

Theorem 10 (Steepest Descent): Define $f(N)$ as in (5) and, having fixed $N \in \mathbb{R}^{m\times(m-r)}$ and $N_\perp \in \mathbb{R}^{m\times r}$, define $g(K)$ as in (10). Then, the gradient of $g(K)$ about $K = 0$ is

$$\text{grad } g = 2N_\perp^T (X - B)^T A \tag{22}$$

where $A \in \mathbb{R}^{n\times(m-r)}$ and $B \in \mathbb{R}^{n\times m}$ are the unique matrices that satisfy

$$\begin{aligned}\text{vec}\,\{A\} &= \left[(N \otimes I_n)^T Q^{-1} (N \otimes I_n)\right]^{-1} \text{vec}\{XN\} \\ \text{vec}\,\{B\} &= Q^{-1}\,\text{vec}\left\{AN^T\right\}.\end{aligned} \tag{23}$$

If $g(K)$ is instead defined as in (14), then under the extra condition that $[N\ N_\perp]^T[N\ N_\perp] = I$, the gradient of $g(K)$ about $K = 0$ is also given by (22).

Note that if $Q = I$ and $[N\ N_\perp]^T[N\ N_\perp] = I$, then grad $g = 2N_\perp^T X^T X N$, agreeing with (17).

Complexity: Computing grad g requires $O(n^2m^2(m - r) + n^3(m - r)^3)$ flops. If Q is diagonal, this reduces to $O(nm(m - r)^2 + nmr)$ flops. If $Q = I$ and X^TX is precomputed, then $O(m^2r)$ flops are required. If $r \ll m$, then the flop counts for these three cases are $O(n^3m^3)$, $O(nm^3)$, and $O(m^2r)$, respectively.

Remark: Evaluating the cost function $f(N)$ requires $O(n^2m^2(m - r) + n^3(m - r)^3)$ flops for arbitrary Q. It requires $O(nm(m - r)^2)$ flops if Q is diagonal. If $Q = I$, it will be seen later that minimizing $f(N)$ is equivalent to maximizing $f(N_\perp)$, and the latter requires only $O(m^2r)$ flops to be evaluated, provided X^TX is precomputed.

Theorem 10 combined with Algorithm 4 leads to four different steepest descent algorithms, depending on which local parameterization [(9) or (13)] is used and on whether or not $Q = I$. The two algorithms based on (9) are presented below. Their counterparts, based on (13), are presented in Appendix B. They all use Armijo's step-size rule [22, Sec. 1.2.3], and they are all tailored for the case when $r \le m/2$.

Notation: The norm $\|\cdot\|$ appearing in the algorithms is the Frobenius norm. The "Q-Factor" operator qf$\{\cdot\}$ is defined to be the orthogonal part qf$\{X\} = Q$ of the QR decomposition $X = QR$.

Algorithm 11 (Steepest Descent Along Straight Lines)

1) Choose $N \in \mathbb{R}^{m\times(m-r)}$ and $N_\perp \in \mathbb{R}^{m\times r}$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$. Set step size $\lambda := 1$.

2) Evaluate $f(N) = \text{vec}\ \{XN\}^T[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)]^{-1}\text{vec}\{XN\}$.

3) Compute descent direction $K = -2N_\perp^T(X - B)^TA$, where A and B are defined in (23).

4) Evaluate $f(N + 2\lambda N_\perp K)$. If $f(N) - f(N + 2\lambda N_\perp K) \ge \lambda\|K\|^2$, then set $\lambda := 2\lambda$, and repeat Step 4.

5) Evaluate $f(N + \lambda N_\perp K)$. If $f(N) - f(N + \lambda N_\perp K) < 1/2\lambda\|K\|^2$, then set $\lambda := 1/2\lambda$, and repeat Step 5.

6) Set $N := N+\lambda N_\perp K$. Renormalize $[N\ N_\perp]$ by setting $[N\ N_\perp] := \text{qf}\{N\}$. Go to Step 2.

[5]The weighted norm used in [16] takes the form $\|X - R\|^2 = \sum_{i,j} W_{ij}^2(X - R)_{ij}^2$ for some weighting matrix W. This is equivalent to restricting Q in (1) to be diagonal.

Complexity: Each iteration of Algorithm 11 requires $O(n^2m^2(m-r)+n^3(m-r)^3)$ flops in general and $O(nm(m-r)^2+nmr+m^3)$ flops if Q is diagonal. If $r \ll m$, then these flop counts approach $O(n^3m^3)$ and $O(nm^3)$, respectively.

Remarks:

1) In any given iteration of Algorithm 11, if Step 4 is repeated at least once, then the test in step 5 becomes redundant. This holds for all the steepest descent algorithms.
2) In practice, the algorithms must include a test for convergence. One possibility is to test to see if the magnitude of the gradient $||K||$ is sufficiently close to zero [22]. Once the algorithm is terminated, the low-rank approximation R is found by evaluating (4).
3) Renormalizing $[N\ N_\perp]$ in Step 6 serves the purpose of computing an $N_\perp$ orthogonal to N.

The disadvantage of Algorithm 11 is its computational complexity; many flops are required to evaluate the cost function. The following algorithm overcomes this in the unweighted case by optimizing over $f(N_\perp)$ rather than over $f(N)$. Specifically, if $f(N)=\text{tr}\{N^TX^TXN\}$, then

$$\begin{aligned} f(N)+f(N_\perp) =& \text{tr}\left\{N^TX^TXN+N_\perp^TX^TXN_\perp\right\} \\ =& \text{tr}\left\{[N\ N_\perp]^T X^TX\,[N\ N_\perp]\right\} \\ =& \text{tr}\left\{X^TX\right\}. \end{aligned} \tag{24}$$

Thus, performing steepest descent on $f(N)$ is identical to performing steepest ascent on $f(N_\perp)$. When $r<(m/2)$, it is computationally more efficient to maximize $f(N_\perp)$ rather than minimize $f(N)$.

Algorithm 12 (Steepest Descent along Straight Lines, Unweighted Case)
1) Choose $N_\perp \in \mathbb{R}^{m\times r}$ such that $N_\perp^TN_\perp = I$. Set step size $\lambda:=1$. Precompute X^TX.
2) Evaluate $f(N_\perp)=\text{tr}\{N_\perp^TX^TXN_\perp\}$.
3) Compute ascent direction $Z = 2(I - N_\perp N_\perp^T)X^TXN_\perp$.
4) Evaluate $f(N_\perp + 2\lambda Z)$, where $f(Y) = \text{tr}\{Y^TX^TXY(Y^TY)^{-1}\}$. If $f(N_\perp+2\lambda Z)-f(N_\perp)\geq \lambda\|Z\|^2$, then set $\lambda:=2\lambda$, and repeat Step 4.
5) Evaluate $f(N_\perp + \lambda Z)$, where $f(Y) = \text{tr}\{Y^TX^TXY(Y^TY)^{-1}\}$. If $f(N_\perp+\lambda Z)-f(N_\perp) < 1/2\lambda\|Z\|^2$, then set $\lambda := 1/2\lambda$, and repeat Step 5.
6) Set $N_\perp:=N_\perp+\lambda Z$. Renormalize $N_\perp$. Go to Step 2.

Complexity: Each iteration of Algorithm 12 requires $O(m^2r)$ flops.

Remarks:

1) Algorithm 11 with $Q=I$ and Algorithm 12 are equivalent in that they both produce the same sequence of points $N_\perp$. However, Algorithm 12 requires fewer flops per iteration.
2) It is not necessary to renormalize $N_\perp$ in Step 6 at every iteration. However, $N_\perp$ can become ill-conditioned if it is not renormalized regularly.
3) The low-rank approximation R is given by $R=XN_\perp N_\perp^T$ (provided $N_\perp^TN_\perp=I$).
4) If $r=1$ or $r=m-1$, then the optimal step size rule can be used instead of Armijo's rule [19].

In Algorithm 12, modest computational savings can be made by first reducing X^TX to tridiagonal form. Specifically, if $S\in\mathbb{R}^{m\times m}$ is an orthonormal matrix such that S^TX^TXS is tridiagonal and if $N_\perp$ maximizes $\text{tr}\{N_\perp^TS^TX^TXSN_\perp\}$, then $SN_\perp$ maximizes $f(N_\perp)$, and thus, the best rank r approximation of X is $R=XSN_\perp N_\perp^TS^T$.

V. Second-Order Descent Methods

This section presents quadratically (and, in the unweighted case, cubically) convergent algorithms for solving the low-rank approximation problem (1). At each iteration, the algorithms perform a Newton step in local coordinates.

The following theorem derives an expression for the Hessian of $g(K)$ in Algorithm 4. Its statement requires the commutation matrix [18] $C\in\mathbb{R}^{r(m-r)\times r(m-r)}$, which is the unique matrix for which

$$\text{vec}\left\{K^T\right\} = C\,\text{vec}\{K\} \tag{25}$$

holds for all $K\in\mathbb{R}^{r\times(m-r)}$.

Theorem 13 (Quadratic Approximation): Define $f(N)$ as in (5) and, having fixed $N\in\mathbb{R}^{m\times(m-r)}$ and $N_\perp\in\mathbb{R}^{m\times r}$, define $g(K)$ as in (10). Then, the second-order Taylor series approximation of $g(K)$ about $K=0$ is

$$\tilde{g}(K)=g(0)+\text{vec}\{\text{grad}\,g\}^T\text{vec}\{K\}+\frac{1}{2}\text{vec}\{K\}^TH\,\text{vec}\{K\} \tag{26}$$

where grad g is defined in (22), and $H\in\mathbb{R}^{r(m-r)\times r(m-r)}$ is the symmetric matrix in (27), shown at the bottom of the page.

The commutation matrix C in (27) is defined in (25).

$$\begin{aligned} H =& 2\Big\{ (I_{m-r}\otimes(X-B)N_\perp)^T\left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(I_{m-r}\otimes(X-B)N_\perp) \\ & -(I_{m-r}\otimes(X-B)N_\perp)^T\left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(N\otimes I_n)^TQ^{-1}(N_\perp\otimes A)C \\ & -C^T(N_\perp\otimes A)^TQ^{-1}(N\otimes I_n)\left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(I_{m-r}\otimes(X-B)N_\perp) \\ & -C^T(N_\perp\otimes A)^T\left(Q^{-1}-Q^{-1}(N\otimes I_n)\left[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)\right]^{-1}(N\otimes I_n)^TQ^{-1}\right)(N_\perp\otimes A)C\Big\}. \end{aligned} \tag{27}$$

If $g(K)$ is instead defined as in (14), then under the extra condition that $[N\ N_\perp]^T[N\ N_\perp] = I$, the second-order Taylor series approximation of $g(K)$ about $K = 0$ is also given by (26).

If $Q = I$ and $[N\ N_\perp]^T[N\ N_\perp] = I$, then the Hessian (27) simplifies to

$$H = 2\left(I_{m-r} \otimes N_\perp^T X^T X N_\perp\right) - 2\left(N^T X^T X N \otimes I_r\right). \tag{28}$$

Complexity: Computing the Hessian (27) requires $O(n^3(m-r)^3 + n^2m^2(m-r)r)$ flops in general and $O\big(nm(m-r)^2r^2\big)$ flops if Q is diagonal. If $Q = I$, then $O\big(m^3\big)$ flops are required if X^TX is precomputed. If $r \ll m$, then these flop counts are $O(n^3m^3)$, $O(nm^3r^2)$, and $O(m^3)$, respectively.

It is now straightforward to derive the Newton step K for which $d\tilde{g}(K; dK) = 0$ for all dK. Since $d\tilde{g}(K; dK) = [\mathrm{vec}\{\mathrm{grad}\ g\}^T + \mathrm{vec}\{K\}^T H]\mathrm{vec}\{dK\}$, the Newton step K is obtained by solving the linear equation

$$H\mathrm{vec}\{K\} = -\mathrm{vec}\{\mathrm{grad}\ g\}. \tag{29}$$

It requires $O\big((m-r)^3r^3\big)$ flops to solve (29), which is fewer than it takes to compute the Hessian (27) in the weighted case.

A Newton step is not guaranteed to decrease the cost function. It is standard [22, Sec. 1.4.4] to include a test to ensure that the Newton step significantly decreases the cost function. If the test fails, an alternative descent step, such as one iteration of Algorithm 11, should be used.

Algorithm 14 (Newton Step)
1) Choose $N \in \mathbb{R}^{m\times(m-r)}$ and $N_\perp \in \mathbb{R}^{m\times r}$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$.
2) Compute the negative of the gradient $G = -2N_\perp^T(X-B)^TA$, where A and B are defined in (23).
3) Compute the Hessian H as defined in (27).
4) Solve the linear equation $H\mathrm{vec}\{K\} = \mathrm{vec}\{G\}$ for the matrix $K \in \mathbb{R}^{r\times(m-r)}$.
5) Evaluate $f(N) = \mathrm{vec}\{XN\}^T[(N\otimes I_n)^TQ^{-1}(N\otimes I_n)]^{-1}\mathrm{vec}\{XN\}$.
6) Evaluate $f(N + N_\perp K)$. If $\mathrm{tr}\{K^TG\} < 0$ or $f(N) - f(N+N_\perp K) < 1/4\,\mathrm{tr}\{K^TG\}$, then abort Newton step.
7) Set $N := N + N_\perp K$. Renormalize $[N\ N_\perp]$ by setting $[N\ N_\perp] := \mathrm{qf}\{N\}$. Go to Step 2.

Complexity: One iteration of Algorithm 14 requires $O\big(n^3(m-r)^3 + n^2m^2(m-r)r\big)$ flops in general and $O\big(nm(m-r)^2r^2\big)$ flops if Q is diagonal. If $r \ll m$, then these flop counts are $O(n^3m^3)$ and $O(nm^3r^2)$, respectively.

Remarks:

1) It is straightforward to modify Algorithm 14 to move along geodesics rather than straight lines; refer to Algorithm 16 to see how. Such a modification does not alter the order of the computational complexity of the algorithm. The same goes for Algorithm 15 as well.
2) The constant 1/4 in Step 6 can be replaced by any constant strictly between 0 and 1/2; see [22].
3) Analogous to the algorithms in Section IV-B, Algorithms 14 and 15 are deemed to have converged (and hence should be terminated) if $\|G\|$ is sufficiently small.

In the unweighted case, (29) can be written in the form

$$N_\perp^T X^T X N_\perp K - K N^T X^T X N = -N_\perp^T X^T X N. \tag{30}$$

Thus, the step size K is found by solving the Sylvester equation (30); efficient algorithms to do so appear in [2] and [11]. They require $O(m^3)$ flops.

Algorithm 15 (Newton Step, Unweighted Case)
1) Choose $N \in \mathbb{R}^{m\times(m-r)}$ and $N_\perp \in \mathbb{R}^{m\times r}$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$. Precompute X^TX.
2) Compute one half times the negative of the gradient $G = -N_\perp^TX^TXN$.
3) Compute $A = N_\perp^TX^TXN_\perp$, $B = N^TX^TXN$, and solve the Sylvester equation $AK - KB = G$ for $K \in \mathbb{R}^{r\times(m-r)}$.
4) Evaluate $f(N) = \mathrm{tr}\{N^TX^TXN\}$.
5) Evaluate $f(N + N_\perp K)$, where $f(Y) = \mathrm{tr}\{Y^TX^TXY(Y^TY)^{-1}\}$. If $\mathrm{tr}\{K^TG\} < 0$ or $f(N) - f(N+N_\perp K) < 1/2\ \mathrm{tr}\{K^TG\}$; then, abort Newton step.
6) Set $N := N + N_\perp K$. Renormalize $[N\ N_\perp]$ by setting $[N\ N_\perp] := \mathrm{qf}\{N\}$. Go to Step 2.

Complexity: One iteration of Algorithm 15 requires $O(m^3)$ flops.

Remarks:

1) Algorithm 15 is equivalent to Algorithm 14 with $Q = I$ in that they both produce the same sequence of points $[N\ N_\perp]$.
2) In practice, $f(N + N_\perp K)$ in Step 5 of Algorithm 15 should be computed by first setting $N := N + N_\perp K$, renormalizing $[N\ N_\perp]$, and then computing $\mathrm{tr}\{N^TX^TXN\}$. Moreover, the value N^TX^TXN should be saved for subsequent use in Step 3.
3) Since it is faster to compute $\mathrm{qf}\{N_\perp\}$ rather than $\mathrm{qf}\{N\}$, a small computational saving will be made by maximizing $f(N_\perp)$ rather than minimizing $f(N)$; refer to Algorithm 12 to see how.
4) Dedicated algorithms for minimizing $f(N)$ in the unweighted case appear in [1], [5]. They have similar numerical behavior to Algorithm 15 but require fewer flops per iteration.

The rate of convergence of Algorithm 15 is cubic because, about the minimum of $f(N)$, the local cost function $g(K)$ defined in either (10) or (14) is symmetrical ($g(K) = g(-K)$ for all K) if $Q = I$. This means that the Taylor series expansion of $g(K)$ has no cubic term, and thus, the approximation (26) is correct up to degree three [8].

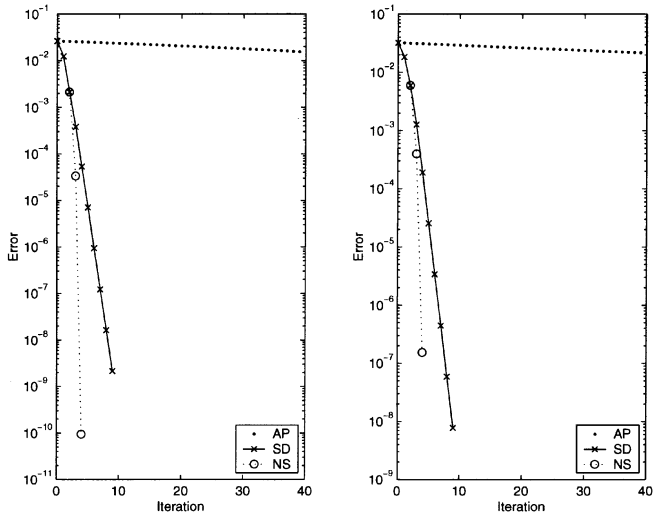


Fig. 1. Graphs illustrating the poor performance of the alternating projection method yet good performance of the steepest descent method when there is very little separation in the singular values of the data matrix.

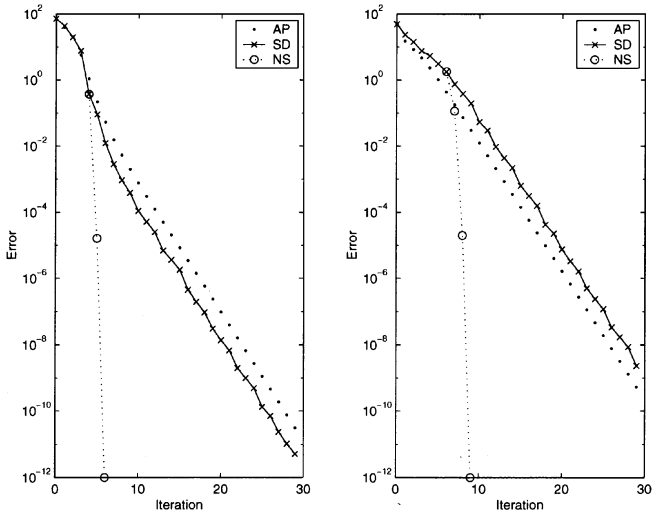


Fig. 2. Graphs illustrating comparable performance of the steepest descent method and alternating projection method when the singular values of the data matrix are well separated.

VI. Numerical Study

This section compares the performance of the following algorithms in a limited number of situations.

AP: The Alternating Projection algorithm described in Section IV-A.

SD: The Steepest Descent algorithm (Algorithm 11) and, in the unweighted case, its equivalent version (Algorithm 12).

SD (geod): The Steepest Descent along Geodesics algorithm (Algorithm 16) and, in the unweighted case, its equivalent version (Algorithm 17).

NS: The Newton Step algorithm (Algorithm 14) and, in the unweighted case, its equivalent version (Algorithm 15).

NS (geod): The Newton Step algorithm (either Algorithm 14 or Algorithm 15) appropriately modified to move along geodesics rather than straight lines.

Figs. 1–6 show the performance of the various algorithms in six different situations. Each figure contains two graphs, corresponding to initializing the algorithms at two different randomly chosen points. Within each graph, all algorithms were initialized identically. The error, which is defined as the current cost $f(N)$ (defined in (3)) minus the minimum cost, is graphed against the number of iterations taken by each algorithm. Only Fig. 4 used a weighting matrix; the other five figures studied the unweighted case [$Q = I$ in (1)].

In Fig. 1, the data matrix X was chosen to be $X = \text{diag}\{1, 1, 1, 0.99, 0.99, 0.99, 0.99\}$. Notice that the eigenvalues of $X^T X$ (equivalently, the singular values of X) are closely spaced. Each algorithm was required to find the best rank $r = 3$ approximation of X. As Fig. 1 shows, the AP algorithm performs extremely poorly. The SD method, however, exhibits rapid convergence. In fact, the SD method converges more quickly than it does in Fig. 2, showing that an ill-conditioned problem for the AP algorithm is a well-conditioned problem for the SD algorithm. Fig. 1 also shows that only two iterations of the NS algorithm (run after the second iteration of the SD algorithm) are required for convergence.

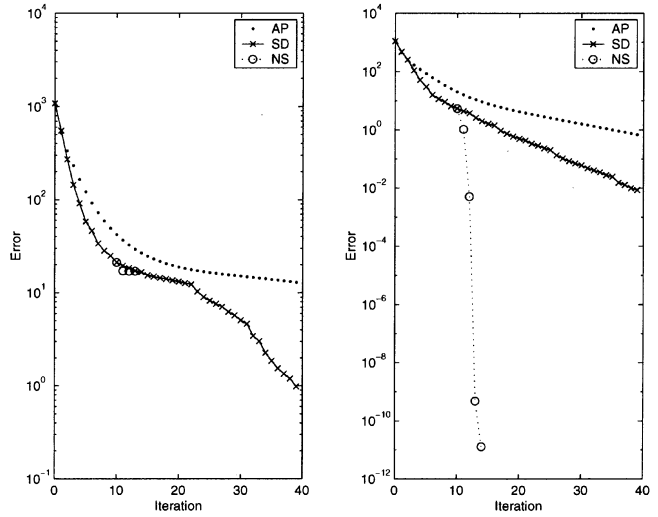


Fig. 3. Graphs illustrating better performance of the steepest descent method over the alternating projection method on a randomly chosen matrix. In addition, notice that the Newton method will converge to the closest critical point rather than continue downhill.

In Fig. 2, the data matrix $X = \text{diag}\{1, 2, 3, 4, 5, 6, 7\}$ was chosen to have well-separated singular values. Each algorithm sought to find the best rank $r = 3$ approximation. Fig. 2 shows that both AP and SD perform comparably in this situation. The NS algorithm exhibits cubic convergence. However, since the NS algorithm converges to the nearest critical point, it is just as likely to attempt to move uphill rather than downhill. (The test in Step 6 of Algorithm 14 will detect this, however.) It is thus necessary to start the NS algorithm after the fourth iteration of SD in the graph on the left of Fig. 2 and after the sixth iteration of SD in the graph on the right.

In Fig. 3, the data matrix X was a randomly chosen 120×100 matrix. The algorithms attempted to find the best rank $r = 4$ approximation. In both cases, the NS algorithm was run after ten iterations of the SD algorithm. In the first case, the NS converged to a local minimum, whereas in the second case, it converged to the global minimum. It is interesting to see how the

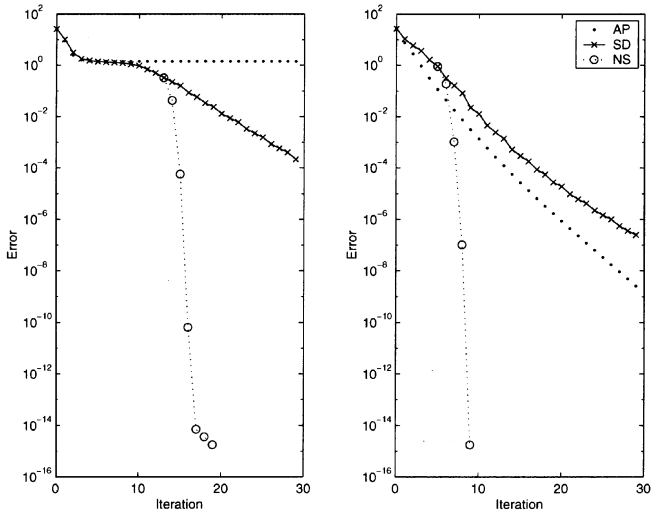


Fig. 4. Graphs illustrating more robust performance of the steepest descent method over the alternating projection method on a randomly chosen data and weighting matrix.

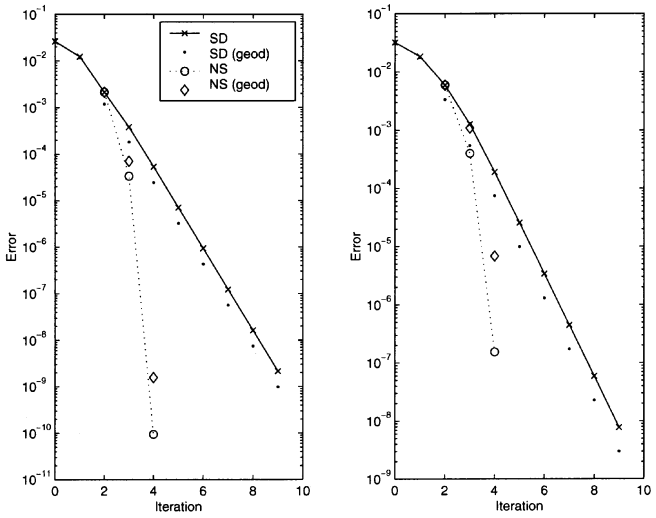


Fig. 5. Graphs illustrating better performance of straight line Newton method over geodesic Newton method and comparable performance of straight line steepest descent and geodesic steepest descent.

local minimum affects the performance of the AP and SD algorithms. The SD algorithm escapes from the local minimum on its 23rd iteration. The AP algorithm is still significantly hampered by the local minimum after 39 iterations.

In Fig. 4, the data matrix X was a randomly chosen 10×10 matrix. The weighting matrix Q was chosen at random with singular values between 0.2857 and 1. Each algorithm was required to find the best rank $r = 3$ approximation of X. For the NS algorithm to converge, it was necessary to run it after the 13th iteration of SD in the left-hand graph and after the fifth iteration of SD in the right-hand graph. The left-hand graph shows the AP algorithm converging to a local minimum, whereas the SD algorithm escapes the local minimum. The right-hand graph shows the AP algorithm performing slightly better than the SD algorithm.

The final two figures compare the straight line algorithms with the geodesic algorithms. Fig. 5 uses the same data as in Fig. 1, and Fig. 6 uses the same data as in Fig. 2. Figs. 5 and 6 show that the straight line and geodesic SD algorithms perform comparably, whereas the straight line NS algorithm is superior to the geodesic NS algorithm.

One important factor that the above results have neglected to show is the number of flops required per iteration. The AP algorithm generally requires the least number of flops per iteration. However, as Fig. 1 illustrates, the AP algorithm can suffer from exceptionally slow convergence. Moreover, the SD algorithm empirically appears to be more robust than the AP algorithm; Figs. 3 and 4 show the SD algorithm escaping from local minima. In certain circumstances, the quadratic (or, in the unweighted case, cubic) convergence of the NS algorithm more than compensates for its computational complexity.

A small number of simulations were done to compare the number of flops (as calculated by Matlab's flops command) required for the straight line and geodesic versions of SD and NS algorithms. It was found that SD and SD (geod) perform comparably; sometimes SD requires fewer flops per iteration, and other times, SD (geod) does. (The step selection rule in the SD algorithms makes the actual number of flops per iteration unpredictable in advance.) It was also found that the NS algorithm requires fewer flops per iteration than NS (geod) does.

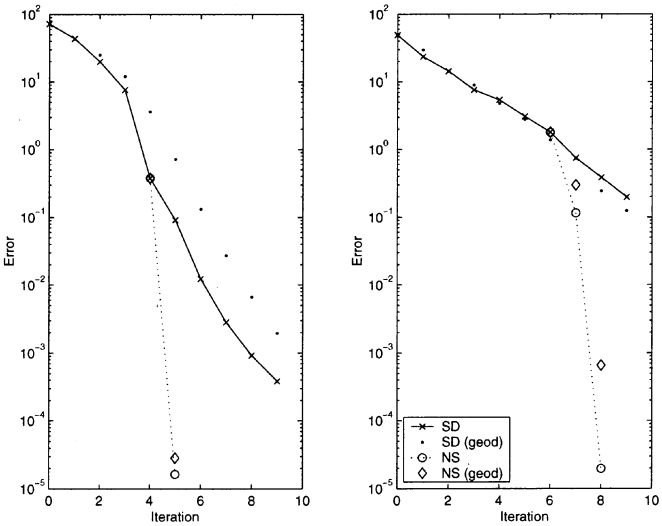


Fig. 6. Graphs illustrating better performance of straight line Newton method over geodesic Newton method and comparable performance of straight line steepest descent and geodesic steepest descent.

VII. Conclusion

This paper studied the weighted low-rank approximation problem (1). It generalized the EVD method for the unweighted case to the weighted case by showing that the best low-rank approximation can be found by first computing the minimizing subspace of a certain cost function (Theorem 1). This novel approach led to the derivation of closed-form solutions of (1) for certain weighting matrices (Theorem 3). A general framework for numerically finding the minimizing subspace of a cost function was given in Section III. The advantage of this framework is that it considerably reduces the dimension of the optimization problem. This framework was then applied

in Sections IV and V to derive modified steepest descent and Newton algorithms for the low-rank approximation problem. These algorithms are not standard optimization algorithms because the cost function changes at each iteration. The numerical study in Section VI demonstrated the advantages of these algorithms over the traditional alternating projection algorithm. For practical applications of weighted low-rank approximations, see [16] and [20].

APPENDIX A
PROOFS

Equalities (31)–(33), shown at the bottom of the page, are used in the following proofs.

A. Proof of Theorem 1

The method of Lagrange multipliers, as elucidated in [18], is applied to (3). Define

$$\begin{aligned}\phi(R) =& \text{vec}\{X-R\}^T Q \text{vec}\{X-R\} \in \mathbb{R}\\ G(R) =& RN \in \mathbb{R}^{n\times(m-r)}.\end{aligned}$$

Minimizing $\phi(R)$ subject to $G(R)=0$ results in the Lagrangian

$$\psi(R) = \phi(R) - \text{tr}\left\{L^T G(R)\right\}$$

where $L \in \mathbb{R}^{n\times(m-r)}$ is the Lagrange multiplier. Its differential is

$$\begin{aligned}d\psi =& \text{vec}\{-dR\}^T Q\ \text{vec}\{X-R\} + \text{vec}\{X-R\}^T Q\ \text{vec}\{-dR\}\\ &- \text{tr}\left\{L^T(dR)N\right\}\\ =& 2\ \text{vec}\{R-X\}^T Q\ \text{vec}\{dR\} - \text{tr}\left\{NL^T dR\right\}\\ =& 2\ \text{vec}\{R-X\}^T Q\ \text{vec}\{dR\} - \text{vec}\left\{LN^T\right\}^T \text{vec}\{dR\}\end{aligned}$$

where the last line is obtained by using (31). This shows that $d\psi = 0$ for all dR if and only if

$$2\ \text{vec}\{R-X\}^T Q = \text{vec}\left\{LN^T\right\}^T.$$

Writing both this condition and the condition $G(R)=0$ in matrix form yields

$$\begin{bmatrix} 2Q & -(N\otimes I_n)\\ (N^T\otimes I_n) & 0\end{bmatrix}\begin{bmatrix}\text{vec}\{R\}\\ \text{vec}\{L\}\end{bmatrix} = \begin{bmatrix}2Q\ \text{vec}\{X\}\\ 0\end{bmatrix} \quad (34)$$

where use has been made of (32). Using the fact that

$$\begin{bmatrix} A & -B\\ B^T & 0\end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} - A^{-1}B\left(B^TA^{-1}B\right)^{-1}B^TA^{-1} & *\\ -\left(B^TA^{-1}B\right)^{-1}B^TA^{-1} & *\end{bmatrix}$$

where $*$ denotes unimportant elements, (34) is readily solved for $\text{vec}\{R\}$, yielding (4). Substituting this solution into the cost function $\phi(R)$ immediately gives (5). Finally, the reason $f(N) = f(NS)$ for any invertible matrix S is that $RN=0$ if and only if $RNS=0$, that is, the constraint set $\{R : RN=0\}$ in (3) equals the constraint set $\{R : RNS=0\}$.

B. Proof of Corollary 2

If $Q=I$, then (5) becomes

$$\begin{aligned}f(N) =& \text{vec}\{X\}^T(N\otimes I_n)\left[(N\otimes I_n)^T(N\otimes I_n)\right]^{-1}\\ &\cdot (N\otimes I_n)^T\text{vec}\{X\}\\ =& \text{vec}\{X\}^T(N\otimes I_n)\left[\left(N^TN\right)^{-1}\otimes I_n\right]\\ &\cdot (N\otimes I_n)^T\text{vec}\{X\}\\ =& \text{vec}\{XN\}^T\text{vec}\left\{XN\left(N^TN\right)^{-1}\right\}\\ =& \text{tr}\left\{N^TX^TXN\left(N^TN\right)^{-1}\right\}\end{aligned}$$

where the second last line is obtained by using (32) and the last line by (31). Equation (7) is obtained similarly. □

C. Proof of Theorem 3

Substituting $Q = Q_1\otimes Q_2$ into (5) yields

$$\begin{aligned}f(N) =& \text{vec}\{X\}^T\left[N\left(N^TQ_1^{-1}N\right)^{-1}N^T\otimes Q_2\right]\text{vec}\{X\}\\ =& \text{tr}\left\{X^TQ_2XN\left(N^TQ_1^{-1}N\right)^{-1}N^T\right\}\\ =& \text{tr}\left\{\tilde{N}^T\tilde{X}^T\tilde{X}\tilde{N}\left(\tilde{N}^T\tilde{N}\right)^{-1}\right\}\end{aligned}$$

where $\tilde{X} = Q_2^{1/2}XQ_1^{1/2}$, and $\tilde{N} = Q_1^{-1/2}N$. Since $f(N)$ is a generalized Rayleigh quotient in $\tilde{N}$, cf., (6), its minimum occurs when the columns of $\tilde{N}$ span the same space as do the $m-r$ smallest eigenvectors of $\tilde{X}^T\tilde{X}$. The solution of (1) is found by

---

$$\text{vec}\{A\}^T\text{vec}\{B\} = \text{tr}\left\{A^TB\right\} \quad \text{provided } A \text{ and } B \text{ have the same dimensions.} \quad (31)$$

$$\text{vec}\{AB\} = \left(B^T\otimes I_n\right)\text{vec}\{A\} \quad \text{where } n \text{ is the number of rows of } A \quad (32)$$

$$= (I_m\otimes A)\,\text{vec}\{B\} \quad \text{where } m \text{ is the number of columns of } B.$$

$$\text{vec}\{ABC\} = \left(C^T\otimes A\right)\text{vec}\{B\}. \quad (33)$$

substituting this value of $\tilde{N}$ into (4). Substituting $Q = Q_1 \otimes Q_2$ into (4) yields

$$\begin{aligned} R =& X - XN\left(N^T Q_1^{-1} N\right)^{-1} N^T Q_1^{-1} \\ =& Q_2^{-1/2}\left[\tilde{X} - \tilde{X}\tilde{N}\left(\tilde{N}^T\tilde{N}\right)^{-1}\tilde{N}^T\right] Q_1^{-1/2}. \end{aligned}$$

If the columns of $\tilde{N}$ span the same space as the $m-r$ smallest eigenvectors of $\tilde{X}^T\tilde{X}$ and if $\tilde{X} = U\Sigma V^T$ is the SVD of $\tilde{X}$, then $\tilde{X} - \tilde{X}\tilde{N}(\tilde{N}^T\tilde{N})^{-1}\tilde{N}^T = U\Sigma_r V^T$. Thus, $R = Q_2^{-1/2} U \Sigma_r V^T Q_1^{-1/2}$. □

D. Proof of Proposition 5

Since $N + N_\perp K$ is affine in K, (11) clearly holds. Similarly, (12) follows from the chain rule for second differentials [18, Ch. 6, Th. 11]. □

E. Proof of Proposition 6

Under the Levi-Civita connection, the gradient and Hessian of a function on the Grassmann manifold are equivalent to the first and second derivatives of the function expressed in normal coordinates around the point at which the derivatives are taken. Since $NV\cos(\Sigma) + U\sin(\Sigma)$ is the exponential map (that is, it traces out geodesics) [8, Th. 2.3], $g(K)$ in (14) is precisely $f(N)$ expressed in normal coordinates. Thus, the first and second derivatives (and hence the differentials) of $g(K)$ are readily obtained from the formulae for the gradient and Hessian of a function on a Grassmann manifold, as given in [8, Sec. 2.5.3 and 2.5.4]. □

F. Proof of Theorem 8

Comparing (15) and (16) with (11) and (15) shows that they will be the same if $df(NdK^T dK) = 0$ for all dK. Under the hypothesis of the theorem, $f(N + NL) = f(N)$ for $L \in \mathbb{R}^{(m-r)\times(m-r)}$ sufficiently small because range$\{N + NL\}$ = range$\{N\}$ provided $(I+L)$ is invertible. Therefore, $df(NdL) = 0$ for all dL, implying $df(NdK^T dK) = 0$ for all dK as well. □

G. Proof of Proposition 9

If either A or B is fixed, the cost function $\|X - AB\|_Q^2$ is quadratic. By differentiating it and setting the result to zero, the proposition readily follows. □

H. Proof of Theorem 10

The gradient grad g of $g(K)$ is defined to be the unique matrix grad $g \in \mathbb{R}^{r\times(m-r)}$ for which $dg(dK) = \text{tr}\{(\text{grad } g)^T dK\}$. It is thus necessary to first compute $dg(dK)$. From Proposition 5, $dg(dK) = df(N_\perp dK)$. The differential df is now determined. For convenience, define the symmetric bilinear function

$$\begin{aligned} h(N_1, N_2) = \frac{1}{2}\Big[&(N_1 \otimes I_n)^T Q^{-1}(N_2 \otimes I_n) \\ &+ (N_2 \otimes I_n)^T Q^{-1}(N_1 \otimes I_n)\Big]. \end{aligned} \quad (35)$$

Define $h(N) = h(N, N)$ and $h^{-1}(N) = [h(N)]^{-1}$. Then, (5) becomes $f(N) = \text{vec}\{XN\}^T h^{-1}(N)\text{vec}\{XN\}$. Since $dh(dN) = 2h(N, dN)$

$$\begin{aligned} df(dN) =& 2\,\text{vec}\{XN\}^T h^{-1}(N)\text{vec}\{X\,dN\} \\ &- 2\,\text{vec}\{XN\}^T h^{-1}(N) h(N, dN) h^{-1}(N)\text{vec}\{XN\} \\ =& 2\,\text{vec}\{A\}^T \text{vec}\{X\,dN\} \\ &- 2\,\text{vec}\{A\}^T (N \otimes I_n)^T Q^{-1}(dN \otimes I_n)\,\text{vec}\{A\} \end{aligned}$$

where A is defined in (23). Defining B as in (23) shows that df can be compactly written as

$$\begin{aligned} df(dN) =& 2\,\text{tr}\left\{A^T X\,dN\right\} - 2\,\text{vec}\{B\}^T\,\text{vec}\left\{A\,dN^T\right\} \\ =& 2\,\text{tr}\left\{A^T(X - B)\,dN\right\}. \end{aligned} \quad (36)$$

Therefore, $dg(dK) = df(N_\perp dK) = 2\,\text{tr}\{A^T(X-B)N_\perp dK\}$, verifying (22). □

I. Proof of Theorem 13

Taylor's theorem implies that H is the unique symmetric matrix, which satisfies

$$d^2 g(dK, dK) = \text{vec}\{dK\}^T H \text{vec}\{dK\}. \quad (37)$$

In order to first calculate $d^2 f$, define h as in (35), and note that $dh(N, dN) = h(dN, dN)$. Differentiating (36) yields

$$\begin{aligned} d^2 f(dN, dN) = &\ 2\,\text{vec}\{(X - B)\,dN\}^T d\,\text{vec}\{A\} \\ &-2\,\text{vec}\left\{A\,dN^T\right\}^T d\,\text{vec}\{B\}. \end{aligned}$$

Differentiating (23) gives

$$\begin{aligned} d\,\text{vec}\{A\} =& h^{-1}(N)\Big[\text{vec}\{(X - B)\,dN\} \\ &- (N \otimes I_n)^T Q^{-1}\text{vec}\left\{A\,dN^T\right\}\Big] \\ d\,\text{vec}\{B\} =& Q^{-1}(N \otimes I_n)\,d\,\text{vec}\{A\} \\ &+ Q^{-1}\text{vec}\left\{A\,dN^T\right\}. \end{aligned}$$

Thus

$$\begin{aligned} d^2 f(dN, dN) =& -2\,\text{vec}\left\{A\,dN^T\right\}^T Q^{-1}\text{vec}\left\{A\,dN^T\right\} \\ &+ 2\Big[\text{vec}\{(X - B)\,dN\} \\ &\quad - (N \otimes I_n)^T Q^{-1}\text{vec}\left\{A\,dN^T\right\}\Big]^T \\ &\times h^{-1}(N)\Big[\text{vec}\{(X - B)\,dN\} \\ &\quad - (N \otimes I_n)^T Q^{-1}\text{vec}\left\{A\,dN^T\right\}\Big]. \end{aligned}$$

After replacing dN with $N_\perp dK$ (see Proposition 5), it follows that

$$\begin{aligned} H =& 2\Big\{(I_{m-r} \otimes (X - B)N_\perp)^T \\ &\cdot h^{-1}(N)(I_{m-r} \otimes (X - B)N_\perp) \\ &- (I_{m-r} \otimes (X - B)N_\perp)^T h^{-1}(N) \\ &\cdot (N \otimes I_n)^T Q^{-1}(N_\perp \otimes A)\,C \\ &- C^T (N_\perp \otimes A)^T Q^{-1}(N \otimes I_n) \\ &\cdot h^{-1}(N)(I_{m-r} \otimes (X - B)N_\perp) - C^T (N_\perp \otimes A)^T \\ &\cdot \Big(Q^{-1} - Q^{-1}(N \otimes I_n)\,h^{-1}(N) \\ &\quad \cdot (N \otimes I_n)^T Q^{-1}\Big)(N_\perp \otimes A)\,C\Big\}. \end{aligned}$$

This is equivalent to (27). □

APPENDIX B
GEODESIC-BASED ALGORITHMS

This section presents the geodesic based counterparts to the algorithms in Section IV-B.

Algorithm 16 (Steepest Descent Along Geodesics)
1) Choose $N \in \mathbb{R}^{m\times(m-r)}$ and $N_\perp \in \mathbb{R}^{m\times r}$ such that $[N\ N_\perp]^T[N\ N_\perp] = I$. Set Step size $\lambda := 1$.
2) Evaluate $f(N) = \mathrm{vec}\{XN\}^T[(N\otimes I_n)^T Q^{-1}(N\otimes I_n)]^{-1}\mathrm{vec}\{XN\}$.
3) Compute descent direction $K = -2N_\perp^T(X-B)^T A$, where A and B are defined in (23).
4 Determine the compact SVD of $N_\perp K$, that is, compute U, Σ, and V so that $N_\perp K = U\Sigma V^T$, Σ is square and diagonal, and $U^TU = V^TV = I$.
5 Evaluate $f(NV\cos(2\lambda\Sigma) + U\sin(2\lambda\Sigma))$. If $f(N) - f(NV\cos(\lambda\Sigma)+U\sin(\lambda\Sigma)) \geq \lambda\|K\|^2$ then set $\lambda := 2\lambda$, and repeat Step 5.
6) Evaluate $f(NV\cos(\lambda\Sigma) + U\sin(\lambda\Sigma))$. If $f(N) - f(NV\cos(\lambda\Sigma) + U\sin(\lambda\Sigma)) < 1/2\lambda\|K\|^2$, then set $\lambda := 1/2\lambda$, and repeat Step 6.
7) Set $N := NV\cos(\lambda\Sigma)+U\sin(\lambda\Sigma)$. Compute $N_\perp$ by setting $[N\ N_\perp] := \mathrm{qf}\{N\}$. Go to Step 2.

Complexity: Each iteration of Algorithm 16 requires $O(n^2m^2(m-r) + n^3(m-r)^3)$ flops in general and $O(nm(m-r)^2 + nmr)$ flops if Q is diagonal. If $r \ll m$, then these flop counts approach $O(n^3m^3)$ and $O(nm^3)$, respectively.

Algorithm 17 (Steepest Descent Along Geodesics, Unweighted Case)
1) Choose $N_\perp \in \mathbb{R}^{m\times r}$ such that $N_\perp^TN_\perp = I$. Set Step Size $\lambda := 1$. Precompute X^TX.
2) Evaluate $f(N_\perp) = \mathrm{tr}\{N_\perp^TX^TXN_\perp\}$.
3) Compute ascent direction $Z = 2(I - N_\perp N_\perp^T)X^TXN_\perp$.
4) Determine the compact SVD of Z, that is, compute U, Σ, and V so that $Z = U\Sigma V^T$, Σ is square and diagonal, and $U^TU = V^TV = I$.
5) Evaluate $f(N_\perp V\cos(2\lambda\Sigma) + U\sin(2\lambda\Sigma))$, where $f(Y) = \mathrm{tr}\{Y^TX^TXY\}$. If $f(N_\perp V\cos(2\lambda\Sigma) + U\sin(2\lambda\Sigma)) - f(N_\perp) \geq \lambda\|Z\|^2$, then set $\lambda := 2\lambda$, and repeat Step 5.
6) Evaluate $f(N_\perp V\cos(\lambda\Sigma) + U\sin(\lambda\Sigma))$, where $f(Y) = \mathrm{tr}\{Y^TX^TXY\}$. If $f(N_\perp V\cos(\lambda\Sigma) + U\sin(\lambda\Sigma)) - f(N_\perp) < 1/2\lambda\|Z\|^2$, then set $\lambda := (1/2)\lambda$, and repeat Step 6.
7) Set $N_\perp := N_\perp V\cos(\lambda\Sigma) + U\sin(\lambda\Sigma)$. Renormalize $N_\perp$. Go to Step 2.

Complexity: Each iteration of Algorithm 17 requires $O(m^2r)$ flops.

Remarks:
1) Algorithm 16 with $Q = I$ and Algorithm 17 are equivalent in that they both produce the same sequence of points $N_\perp$, although Algorithm 17 requires fewer flops per iteration.
2) If $N_\perp$ is not renormalized in the last step of Algorithm 17, round off error can cause the Armijo rule in Step 6 to repeat indefinitely.

ACKNOWLEDGMENT

The authors would like to thank P. Stoica for initial discussions on the weighted low-rank approximation problem and an anonymous reviewer for valuable comments.

REFERENCES

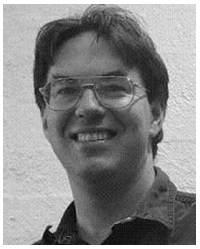
[1] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren, "A Grassmann-Rayleigh quotient iteration for computing invariant subspaces," in *Proc.Conf. Dec. Contr.*, Sydney, Australia, 2000, pp. 4241–4246.
[2] R. H. Bartels and G. W. Stewart, "Algorithm 432: Solution of the matrix equation $AX+XB = C$," *Commun. ACM*, vol. 15, pp. 820–826, 1972.
[3] D. R. Brillinger, *Time Series: Data Analysis and Theory*. New York: Holt, Rinehart, and Winston, 1975.
[4] B. De Moor *et al.*, "Convergence of an algorithm for the Riemannian SVD," in *Open Problems in Mathematical Systems and Control Theory*, V. D. Blondel *et al.*, Eds. New York: Springer, 1999, ch. 20, pp. 95–98.
[5] J. W. Demmel, "Three methods for refining estimates of invariant subspaces," *Comput.*, vol. 38, pp. 43–57, 1987.
[6] K. I. Diamantaras and S.-Y. Kung, "Multilayer neural networks for reduced-rank approximation," *IEEE Trans. Neural Networks*, vol. 5, pp. 684–697, Sept. 1994.
[7] G. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrica*, vol. 1, pp. 211–218, 1936.
[8] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Applicat.*, vol. 20, no. 2, pp. 303–353, 1998.
[9] J. S. Goldstein and I. S. Reed, "Reduced-rank adaptive filtering," *IEEE Trans. Signal Processing*, vol. 45, pp. 492–496, Feb. 1997.
[10] J. S. Goldstein, I. S. Reed, and L. L. Scharf, "A multistage representation of the Wiener filter based on orthogonal projections," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2943–2959, Dec. 1998.
[11] G. H. Golub, S. Nash, and C. F. Van Loan, "A Hessenberg-Schur method for the problem $AX + XB = C$," *IEEE Trans. Automat. Contr.*, vol. AC-24, pp. 909–913, 1979.
[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.
[13] U. Helmke and J. B. Moore, *Optimization and Dynamical Systems*. New York: Springer-Verlag, 1994.
[14] Y. Hua and M. Nikpour, "Computing the reduced-rank Wiener filter by IQMD," *IEEE Signal Processing Lett.*, vol. 6, pp. 240–242, Sept. 1999.
[15] Y. Hua, M. Nikpour, and P. Stoica, "Optimal reduced-rank estimation and filtering," *IEEE Trans. Signal Processing*, vol. 49, pp. 457–469, Mar. 2001.
[16] W.-S. Lu, S.-C. Pei, and P.-H. Wang, "Weighted low-rank approximation of general complex matrices and its application in the design of 2-D digital filters," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 650–655, July 1997.
[17] W.-S. Lu, H.-P. Wang, and A. Antoniou, "Design of 2-D digital filters using the singular value decomposition and balanced approximation," *IEEE Trans. Signal Processing*, vol. 39, pp. 2253–2262, Sept. 1991.
[18] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus With Applications in Statistics and Econometrics*. New York: Wiley, 1994.
[19] J. H. Manton, "A new algorithm for computing the extreme eigenvectors of a complex Hermitian matrix," in *Proc. Eleventh IEEE Workshop Statist. Signal Process.*, Singapore, Aug. 2001.
[20] J. H. Manton and Y. Hua, "Convolutive reduced-rank Wiener filtering," in *Proc. IEEE Conf. Acoust., Speech, Signal Process.*, Salt Lake City, UT, May 2001.
[21] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *Quart. J. Math. Oxford*, vol. 11, pp. 50–59, 1960.
[22] E. Polak, *Optimization: Algorithms and Consistent Approximations*. New York: Springer-Verlag, 1997.

[23] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*. Reading, MA: Addison-Wesley, 1991.
[24] ——, "The SVD and reduced-rank signal processing," *Signal Process.*, vol. 25, pp. 113–133, 1991.
[25] L. L. Scharf and J. K. Thomas, "Wiener filters in canonical coordinates for transform coding, filtering and quantizing," *IEEE Trans. Signal Processing*, vol. 46, pp. 647–654, Mar. 1998.
[26] L. L. Scharf and D. W. Tufts, "Rank reduction for modeling stationary signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, pp. 350–355, Mar. 1987.
[27] D. Shpak, "A weighted-least-squares matrix decomposition method with application to the design of 2-D digital filters," in *Proc. IEEE Midwest Symp. Circuits Syst.*, Calgary, AB, Canada, Aug. 1990, pp. 1070–1073.
[28] P. Stoica and M. Viberg, "Maximum likelihood parameter and rank estimation in reduced-rank multivariate linear regressions," *IEEE Trans. Signal Processing*, vol. 44, pp. 3069–3078, Dec. 1996.
[29] A. J. Thorpe and L. L. Scharf, "Data adaptive rank-shaping methods for solving least squares problems," *IEEE Trans. Signal Processing*, vol. 43, pp. 1591–1601, July 1995.
[30] S. Treitel and J. L. Shanks, "The design of multistage separable planar filters," *IEEE Trans. Geosci. Electron.*, vol. GE-9, pp. 10–27, 1971.

Jonathan H. Manton (M'02) was born in April 1973. He received the B.Sci. degree in mathematics, the B.E.E. degree in 1995, and the Ph.D. degree in 1998, all from the University of Melbourne, Parkville, Victoria, Australia.

From 1998 to 1999, he was a Research Fellow with the Electrical Engineering Department, Melbourne University, Melbourne, Australia, working on a range of topics including the theory of polynomial equations (applying techniques from algebraic geometry and commutative algebra), rank reduction, channel identification, and source separation. In 2000, he became a holder of the prestigious Research Fellow Award by the Australian Research Council, which he chose to take up at the University of Melbourne. His current research interests include wireless communications, stochastic filtering theory, and optimization on Riemannian manifolds and Lie groups.

Dr. Manton has served as an associate editor on the conference editorial board of the IEEE Control and Systems Society. He is currently an associate editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.



Robert Mahony received the science degree, majoring in applied mathematics and geology, from the Australian National University (ANU), Canberra, in 1989. After working for a year as a geophysicist processing marine seismic data, he returned to ANU and received the Ph.D. degree in systems engineering in 1994.

Between 1994 and 1997, he worked as a Research Fellow with the Cooperative Research Centre for Robust and Adaptive Systems, Research School of Information Sciences and Engineering, ANU. From 1997 to 1999, he held a post as a post-doctoral fellow at the CNRS laboratory for Heuristics Diagnostics and complex systems (Heudiasyc), Compiegne University of Technology, Compiegne, France. Between 1999 and 2001, he held a Logan Fellowship at the Department of Engineering and Computer Science, Monash University, Melbourne, Australia. Since July 2001, he has held the post of senior lecturer in mechatronics at the Department of Engineering, ANU. His research interests are in nonlinear control theory with applications in mechanical systems and motion systems, mathematical systems theory, and geometric optimization techniques with applications in linear algebra and digital signal processing.

Yingbo Hua (F'02) is a Professor of electrical engineering at the University of California, Riverside. He is an author/coauthor of 80 journal articles, five book chapters, and 130 conference papers in the fundamental areas of estimation, detection, system identification, and fast algorithms, with applications in communications, remote sensing, and medical data analysis. He is a coeditor of *Signal Processing Advances in Wireless and Mobile Communications* (Englewood Cliffs, NJ: Prentice-Hall, 2001).

Dr. Hua served as Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 1994 to 1997 and from 2001 to 2002 and as Associate Editor of the IEEE SIGNAL PROCESSING LETTERS from 1998 to 2002. He is an invited reviewer for over 16 international journals and an invited speaker, session chair, and organizer of many international conferences. He was an elected member of the IEEE Signal Processing Society's Technical Committees for Underwater Acoustic Signal Processing from 1997 to 1998, for Sensor Array and Multichannel Signal Processing both currently and from 1998 to 2001, and currently for Signal Processing for Communications.