QUASI-NEWTON METHODS ON GRASSMANNIANS AND MULTILINEAR APPROXIMATIONS OF TENSORS

BERKANT SAVAS* AND LEK-HENG LIM†

Abstract. In this paper we proposed quasi-Newton and limited memory quasi-Newton methods for objective functions defined on Grassmannians or a product of Grassmannians. Specifically we defined BFGS and L-BFGS updates in local and global coordinates on Grassmannians or a product of these. We proved that, when local coordinates are used, our BFGS updates on Grassmannians share the same optimality property as the usual BFGS updates on Euclidean spaces. When applied to the best multilinear rank approximation problem for general and symmetric tensors, our approach yields fast, robust, and accurate algorithms that exploit the special Grassmannian structure of the respective problems, and which work on tensors of large dimensions and arbitrarily high order. Extensive numerical experiments are included to substantiate our claims.

Key words. Grassmann manifold, Grassmannian, product of Grassmannians, Grassmann quasi-Newton, Grassmann BFGS, Grassmann L-BFGS, multilinear rank, symmetric multilinear rank, tensor, symmetric tensor, approximations

AMS subject classifications. 65F99, 65K10, 15A69, 14M15, 90C53, 90C30, 53A45

1. Introduction.

1.1. Quasi-Newton and limited memory quasi-Newton algorithms on Grassmannians.

We develop quasi-Newton and limited memory quasi-Newton algorithms for functions defined on a Grassmannian $\mathrm{Gr}(n,r)$ as well as a product of Grassmannians $\mathrm{Gr}(n_1,r_1)\times\cdots\times\mathrm{Gr}(n_k,r_k)$, with BFGS and L-BFGS updates. These are algorithms along the lines of the class of algorithms studied by Edelman, Arias, and Smith in [24] and more recently, the monograph of Absil, Mahony, and Sepulchre in [2]. They are algorithms that respect the Riemannian metric structure of the manifolds under consideration, and not mere applications of the usual BFGS and L-BFGS algorithms for functions on Euclidean space. The actual computations of our BFGS and L-BFGS algorithms on Grassmannians, like the algorithms in [24], require nothing more than standard numerical linear algebra routines and can therefore take advantage of the many high quality softwares developed for matrix computations [3, 40]. In other words, manifold operations such as movement along geodesics and parallel transport of tangent vectors and linear operators do not require actual numerical solutions of the differential equations defining these operations; instead they are characterized as matrix operations on local and global coordinate representations (as matrices) of points on the Grassmannians or points on an appropriate vector bundle.

A departure and improvement from existing algorithms for manifold optimization [1, 2, 24, 27] is that we undertake a *local coordinates* approach. This allows our computational costs to be reduced to the order of the *intrinsic dimension* of the manifold as opposed to the dimension of ambient Euclidean space. For a Grassmannian embedded in the Euclidean space of $n\times r$ matrices, i.e. $\mathrm{Gr}(n,r)\subseteq\mathbb{R}^{n\times r}$, computations in local coordinates have $r(n-r)$ unit cost whereas computations in global coordinates, like the ones in [24], have nr unit cost. This difference becomes more pronounced when we deal

*Department of Mathematics, Linköping University, SE-581 83 Linköping. *Current address*: Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin TX 78712 (`berkant@cs.utexas.edu`)

†Department of Mathematics, University of California, Berkeley, CA 94720
(`lekheng@math.berkeley.edu`)

with products of Grassmannians $\operatorname{Gr}(n_1, r_1)\times\cdots\times\operatorname{Gr}(n_k, r_k) \subseteq \mathbb{R}^{n_1\times r_1}\times\cdots\times\mathbb{R}^{n_k\times r_k}$— for $n_i = O(n)$ and $r_i = O(r)$, we have a computational unit cost of $O(kr(n-r))$ and $O(krn)$ flops between the local and global coordinates versions the BFGS algorithms. More importantly, we will show that our BFGS update in local coordinates on a product of Grassmannians (and Grassmannian in particular) shares the same well-known optimality property of its Euclidean counterpart, namely, it is the best possible update of the current Hessian approximation that satisfies the secant equations and preserves symmetric positive definiteness (cf. Theorem 6.6). For completeness and as an alternative, we also provide the global coordinate version of our BFGS and L-BFGS algorithms analogous to the algorithms described in [24]. However the aforementioned optimality is not possible for BFGS in global coordinates.

While we have limited our discussions to BFGS and L-BFGS updates, it is straightforward to substitute these updates with other quasi-Newton updates (e.g. DFP or more general Broyden class updates) by applying the same principles in this paper.

1.2. Multilinear approximations of tensors and symmetric tensors. In part to illustrate the efficiency of these algorithms, this paper also addresses the following two related problems about the multilinear approximations of tensors and symmetric tensors, which are also important problems in their own right with various applications in analytical chemistry [53], bioinformatics [47], computer vision [55], machine learning [44, 41], neuroscience [45], quantum chemistry [35], signal processing [10, 13, 18], etc. See also the very comprehensive bibliography of the recent survey [38]. In data analytic applications, the multilinear approximation of general tensors is the basis behind the *Tucker model* [54] while the multilinear approximation of symmetric tensors is used in *independent components analysis* [14, 18] and *principal cumulant components analysis* [44, 41]. The algorithms above provide a natural method to solve these problems that exploits their unique structures.

The first problem is that of finding a best multilinear rank-(p, q, r) approximation to a tensor, i.e. approximating a given tensor $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$ by another tensor $\mathcal{B} \in \mathbb{R}^{l\times m\times n}$ of lower multilinear rank [31, 20],

$$\min_{\operatorname{rank}(\mathcal{B})\leq(p,q,r)} \|\mathcal{A} - \mathcal{B}\|.$$

For concreteness we will assume that the norm in question is the Frobenius or Hilbert-Schmidt norm $\|\cdot\|_F$. In notations that we will soon define, we seek matrices X, Y, Z with orthonormal columns and a tensor $\mathcal{C} \in \mathbb{R}^{p\times q\times r}$ such that

$$\operatorname*{argmin}_{X\in\mathrm{O}(l,p),Y\in\mathrm{O}(m,q),Z\in\mathrm{O}(n,r),\mathcal{C}\in\mathbb{R}^{l\times m\times n}} \|\mathcal{A} - (X, Y, Z)\cdot\mathcal{C}\|_F. \tag{1.1}$$

The second problem is that of finding a best multilinear rank-r approximation to a symmetric tensor $\mathcal{S} \in \mathsf{S}^3(\mathbb{R}^n)$. In other words, we seek a matrix Q whose columns are mutually orthonormal, and a symmetric tensor $\mathcal{C} \in \mathsf{S}^3(\mathbb{R}^r)$ such that a multilinear transformation of $\mathcal{C}$ by Q approximates $\mathcal{S}$ in the sense of minimizing a sum-of-squares loss. Using the same notation as in (1.1), the problem is

$$\operatorname*{argmin}_{Q\in\mathrm{O}(n,r),\mathcal{C}\in\mathsf{S}^3(\mathbb{R}^r)} \|\mathcal{S} - (Q, Q, Q)\cdot\mathcal{C}\|_F. \tag{1.2}$$

This problem is significant because many important tensors that arise in applications are symmetric tensors.

We will often refer to the first problem as the *general case* and the second problem as the *symmetric case*. Most discussions are presented for the case of 3-tensors for notational simplicity but key expressions are given for tensors of arbitrary order to facilitate structural analysis of the problem and algorithmic implementation. The MATLAB codes of all algorithms in this paper are available for download at [50, 51]. All of our implementations will handle 3-tensors and in addition, our implementation of the BFGS with scaled identity as initial Hessian approximation will handle tensors of arbitrary order. In fact the reader will find an example of a tensor of order-10 in Section 11, included to show that our algorithms indeed work on high-order tensors.

Our approach is summarized as follows. Observe that due to the unitary invariance of the sum-of-squares norm $\|\cdot\|_F$, the orthonormal matrices U, V, W in (1.1) and the orthonormal matrix Q in (1.2) are only determined up to an action of $\mathrm{O}(p) \times \mathrm{O}(q) \times \mathrm{O}(r)$ and an action of $\mathrm{O}(r)$, respectively. We exploit this to our advantage by reducing the problems to an optimization problem on a product of Grassmannians and a Grassmannian, respectively. Specifically, we reduce (1.1), a minimization problem over a product of three Stiefel manifolds and a Euclidean space $\mathrm{O}(l,p) \times \mathrm{O}(m,q) \times \mathrm{O}(n,r) \times \mathbb{R}^{p\times q\times r}$, to a maximization problem over a product of three Grassmannians $\mathrm{Gr}(l,p) \times \mathrm{Gr}(m,q) \times \mathrm{Gr}(n,r)$; and likewise we reduce (1.2) from $\mathrm{O}(n,r) \times \mathsf{S}^3(\mathbb{R}^r)$ to a maximization problem over $\mathrm{Gr}(n,r)$. This reduction of (1.1) to product of Grassmannians has been exploited in [25, 34, 33]. The algorithms in [25, 34, 33] involve the Hessian, either explicitly or implicitly via its approximation on a tangent. Whichever the case, the reliance on Hessian in these methods results in them quickly becoming infeasible as the size of the problem increases. With this in mind, we consider the quasi-Newton and limited memory quasi-Newton approaches described in the first paragraph of this section.

An important case not addressed in [25, 34, 33] is the multilinear approximation of symmetric tensors (1.2). Note that the general (1.1) and symmetric (1.2) cases are related but different, not unlike the way the singular value problem differs from the symmetric eigenvalue problem for matrices. The problem (1.1) for general tensors is linear in the entries of U, V, W (quadratic upon taking norm-squared) whereas the problem (1.2) for symmetric tensors is cubic in the entries of Q (sextic upon taking norm-squared). To the best of our knowledge, all existing solvers for (1.2) are unsatisfactory because they rely on algorithms for (1.1). A typical heuristic is as follows: find *three* orthonormal matrices Q_1, Q_2, Q_3 and a nonsymmetric $\mathcal{C}' \in \mathbb{R}^{r\times r\times r}$ that approximates $\mathcal{S}$,

$$\mathcal{S} \approx (Q_1, Q_2, Q_3) \cdot \mathcal{C}',$$

then artificially set $Q_1 = Q_2 = Q_3 = Q$ by either averaging or choosing the last iterate and then symmetrize $\mathcal{C}'$. This of course is not ideal. Furthermore, using the framework developed for the general tensor approximation to solve the symmetric tensor approximation problem will be computationally much more expensive. In particular, to optimize $\mathcal{S} \approx (Q_1, Q_2, Q_3) \cdot \mathcal{C}'$ without taking the symmetry into account incurs a k-fold increase in computational cost relative to $\mathcal{S} \approx (Q, \dots, Q) \cdot \mathcal{C}$. The algorithm proposed in this paper solves (1.2) directly. It finds a *single* $Q \in \mathrm{O}(n,r)$ and a *symmetric* $\mathcal{C} \in \mathsf{S}^3(\mathbb{R}^r)$ with

$$\mathcal{S} \approx (Q, Q, Q) \cdot \mathcal{C}.$$

The symmetric case can often be more important than the general case, not surprising since symmetric tensors are common in practice, arising as higher order derivatives of smooth multivariate real-valued functions, higher order moments and cumulants of a vector-valued random variable, etc.

Like the Grassmann-Newton algorithms in [25, 34] and the trust-region approach in [33, 32], the quasi-Newton algorithms proposed in this article have guaranteed convergence to stationary points and represent an improvement over Gauss-Seidel type coordinate-cycling heuristics like alternating least squares (ALS), higher-order orthogonal iteration (HOOI), or higher-order singular value decomposition (HOSVD) [14]. And as far as accuracy is concerned, our algorithms perform as well as the algorithms in [25, 32, 34] and outperform Gauss-Seidel type strategies in many cases. As far as robustness and speed are concerned, our algorithms work on much larger problems and perform vastly faster than Grassmann-Newton algorithm. Asymptotically the memory storage requirements of our algorithms are of the same order-of-magnitude as Gauss-Seidel type strategies. For large problems, our Grassmann L-BFGS algorithm outperforms even Gauss-Seidel strategies (in this case HOOI), which is not unexpected since it has the advantage of requiring only a small number of prior iterates.

We will give the reader a rough idea of the performance of our algorithms. Using MATLAB on a laptop computer, we attempted to find a solution to an accuracy within machine precision, i.e. $\approx 10^{-13}$. For general 3-tensors of size $200 \times 200 \times 200$, our Grassmann L-BFGS algorithm took less than 13 minutes while for general 4-tensors of size $50 \times 50 \times 50 \times 50$, it took about 7 minutes. For symmetric 3-tensors of size $200 \times 200 \times 200$, our Grassmann BFGS algorithm took about 5 minutes while for symmetric 4-tensors of size $50 \times 50 \times 50 \times 50$, it took less than 2 minutes. In all cases, we seek a rank-$(5, 5, 5)$ or rank-$(5, 5, 5, 5)$ approximation. For a general order-10 tensor of dimensions $5 \times \cdots \times 5$, a rank-$(2, \dots, 2)$ approximation took about 15 minutes to reach the same accuracy as above. More extensive numerical experiments are reported in Section 11. The reader is welcomed to try our algorithms, which have been made publicly available at [50, 51].

1.3. Outline. The structure of the article is as follows. In Sections 2 and 3, we present a more careful discussion of tensors, symmetric tensors, multilinear rank, and their corresponding multilinear approximation problems. In Section 4 we will discuss how quasi-Newton methods in Euclidean space may be extended to Riemannian manifolds and, in particular, Grassmannians. Section 5 contains a discussion on geodesic curves and transport of vectors on Grassmannians. In Section 6 we present the modifications on quasi-Newton methods with BFGS updates in order for them to be well-defined on Grassmannians. Also, the reader will find proof of the optimality properties of BFGS updates on products of Grassmannians. Section 7 gives the corresponding modifications for limited memory BFGS updates. Section 8 states the corresponding expressions for the tensor approximation problem, which are defined on a product of Grassmannians. The symmetric case is detailed in section 9. Section 10 contains a few examples with numerical calculations illustrating the presented concepts. The implementation and the experimental results are found in Section 11. Related work and the conclusions are discussed in Section 12 and 13 respectively.

1.4. Notations. Tensors will be denoted by calligraphic letters, e.g. $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$. Matrices will be denoted in upper case letters, e.g. X, Y, Z. We will also use upper case letters X, X_k to denote iterates or elements of a Grassmannian since we represent them as (equivalence classes of) matrices with orthonormal columns. Vectors and iterates in vector form are denoted with lower case letters, e.g. x, y, x_k, y_k, where the subscript is the iteration index. To denote scalars we use lower case Greek letters, e.g. α, β, and t, t_k.

We will use the usual symbol $\otimes$ to denote the *outer product of tensors* and a large boldfaced version $\bigotimes$ to denote the *Kronecker product of operators.* For example, if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ are matrices, then $A \otimes B$ will be a 4-tensor in $\mathbb{R}^{m\times n\times p\times q}$ whereas $A \bigotimes B$ will be a matrix in $\mathbb{R}^{mp\times nq}$. In the former case, we regard A and B as 2-tensors while in the latter case, we regard them as matrix representations of linear operators. The contracted products used in this paper are defined in Appendix A.

For $r \le n$, we will let $\mathrm{O}(n, r) = \{X \in \mathbb{R}^{n\times r} \mid X^{\mathsf{T}}X = I\}$ denote the *Stiefel manifold* of $n \times r$ matrices with orthonormal columns. The special case $r = n$, i.e. the *orthogonal group*, will be denoted $\mathrm{O}(n)$. For $r \le n$, $\mathrm{O}(r)$ acts on $\mathrm{O}(n, r)$ via right multiplication. The set of orbit classes $\mathrm{O}(n, r)/\,\mathrm{O}(r)$ is a manifold called the *Grassmann manifold* or *Grassmannian* (we adopt the latter name throughout this article) and will be denoted $\mathrm{Gr}(n, r)$.

In this paper, we will only cover a minimal number of notions and notations required to describe our algorithm. Further mathematical details concerning tensors, tensor ranks, tensor approximations, as well as the counterpart for symmetric tensors may be found in [8, 20, 25]. Specifically we will use the notational and analytical framework for tensor manipulations introduced in [25, Section 2] and assume that these concepts are familiar to the reader.

2. General and symmetric tensors.

Let $V_1, \dots, V_k$ be real vector spaces of dimensions $n_1, \dots, n_k$ respectively and let $\mathbf{A}$ be an element of the tensor product $V_1 \otimes \ldots \otimes V_k$, i.e. $\mathbf{A}$ is a *tensor* of order k [30, 39, 56]. Up to a choice of bases on $V_1, \dots, V_k$, one may represent a tensor $\mathbf{A}$ as a k-dimensional *hypermatrix* $\mathcal{A} = [a_{i_1\cdots i_k}] \in \mathbb{R}^{n_1\times\cdots\times n_k}$. Similarly, let V be a real vector space of dimension n and $\mathbf{S} \in \mathsf{S}^k(V)$ be a *symmetric tensor* of order k [30, 39, 56]. Up to a choice of basis on V, $\mathbf{S}$ may be represented as a k-dimensional hypermatrix $\mathcal{S} = [s_{i_1\cdots i_k}] \in \mathbb{R}^{n\times\cdots\times n}$ whose entries are invariant under any permutation of indices, i.e.

$$s_{i_{\sigma(1)}\cdots i_{\sigma(k)}} = s_{i_1\cdots i_k} \qquad \text{for every } \sigma \in \mathfrak{S}_k \tag{2.1}$$

We will write $\mathsf{S}^k(\mathbb{R}^n)$ for the subspace of $\mathbb{R}^{n\times\cdots\times n}$ satisfying (2.1). Henceforth, we will assume that there are some predetermined bases and will not distinguish between a tensor $\mathbf{A} \in V_1 \otimes \ldots \otimes V_k$ and its hypermatrix representation $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_k}$ and likewise for a symmetric tensor $\mathbf{S} \in \mathsf{S}^k(V)$ and its hypermatrix representation $\mathcal{S} \in \mathsf{S}^k(\mathbb{R}^n)$. Furthermore we will sometimes present our discussions for the case $k = 3$ for notational simplicity. We often call an order-k tensor simply as a k-tensor and an order-k symmetric tensor as a symmetric k-tensor.

As we have mentioned in Section 1, symmetric tensors are common in applications, largely because of the two examples below. The use of higher-order statistics in signal processing and neuroscience, most notably the technique of independent component analysis, symmetric tensors often play a central role. The reader is referred to [8] for further discussion of symmetric tensors.

Example 2.1. *Let $m \in \{1, 2, \dots, \infty\}$ and $\Omega \subseteq \mathbb{R}^n$ be an open subset. If $f \in C^m(\Omega)$, then for $k = 1, \dots, m$, the kth derivative of f at $\mathbf{a} \in \Omega$ is a symmetric tensor of order k,*

$$D^k f(\mathbf{a}) = \left[\frac{\partial^k f}{\partial x_1^{i_1} \ldots \partial x_n^{i_n}}(\mathbf{a})\right]_{i_1+\cdots+i_n=k} \in \mathsf{S}^k(\mathbb{R}^n).$$

For $k = 1, 2$, the vector $D^1 f(\mathbf{a})$ and the matrix $D^2 f(\mathbf{a})$ are the gradient and Hessian of f at $\mathbf{a}$, respectively.

Example 2.2. *Let $X_1, \dots, X_n$ be random variables with respect to the same probability distribution μ. The moments and cumulants of the random vector $\mathbf{X} = (X_1, \dots, X_n)$ are symmetric tensors of order k defined by*

$$m_k(\mathbf{X}) = \big[E(x_{i_1} x_{i_2} \cdots x_{i_k})\big]_{i_1,\dots,i_k=1}^{n}$$
$$= \left[\int \cdots \int x_{i_1} x_{i_2} \cdots x_{i_k} \; d\mu(x_{i_1}) \cdots d\mu(x_{i_k})\right]_{i_1,\dots,i_k=1}^{n}$$

and

$$\kappa_k(\mathbf{X}) = \left[\sum\nolimits_{\substack{A_1 \sqcup \cdots \sqcup A_p \\ = \{i_1,\dots,i_k\}}} (-1)^{p-1}(p-1)! E(\textstyle\prod_{i \in A_1} x_i) \cdots E(\prod_{i \in A_p} x_i)\right]_{i_1,\dots,i_k=1}^{n}$$

respectively. The sum above is taken over all possible partitions $\{i_1, \dots, i_k\} = A_1 \sqcup \cdots \sqcup A_p$. It is not hard to show that both $m_k(\mathbf{X})$ and $\kappa_k(\mathbf{X}) \in \mathsf{S}^k(\mathbb{R}^n)$. For $n = 1$, the quantities $\kappa_k(\mathbf{X})$ for $k = 1, 2, 3, 4$ have well-known names: they are the expectation, variance, skewness, and kurtosis of the random variable X, respectively.

3. Multilinear transformation and multilinear rank. Matrices can act on other matrices through two independent multiplication operations: left-multiplication and right-multiplication. If $C \in \mathbb{R}^{p \times q}$ and $X \in \mathbb{R}^{m \times p}$, $Y \in \mathbb{R}^{n \times q}$, then the matrix C may be transformed into the matrix $A \in \mathbb{R}^{m \times n}$, by

$$A = XCY^{\mathsf{T}}, \qquad a_{ij} = \sum_{\alpha=1}^{p} \sum_{\beta=1}^{q} x_{i\alpha} y_{j\beta} c_{\alpha\beta}\,. \tag{3.1}$$

Matrices act on order-3 tensors via *three* different multiplication operations. As in the matrix case, these can be combined into a single formula. If $\mathcal{C} \in \mathbb{R}^{p \times q \times r}$ and $X \in \mathbb{R}^{l \times p}$, $Y \in \mathbb{R}^{m \times q}$, $Z \in \mathbb{R}^{n \times r}$, then the 3-tensor $\mathcal{C}$ may be transformed into the 3-tensor $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ via

$$a_{ijk} = \sum_{\alpha=1}^{p} \sum_{\beta=1}^{q} \sum_{\gamma=1}^{r} x_{i\alpha} y_{j\beta} z_{k\gamma} c_{\alpha\beta\gamma}\,. \tag{3.2}$$

We call this operation the *trilinear multiplication* of $\mathcal{A}$ by matrices X, Y and Z, which we write succinctly as

$$\mathcal{A} = (X, Y, Z) \cdot \mathcal{C}. \tag{3.3}$$

This is nothing more than the trilinear equivalent of (3.1), which in this notation has the form

$$A = XCY^{\mathsf{T}} = (X, Y) \cdot C.$$

Informally, (3.2) amounts to multiplying the 3-tensor $\mathcal{A}$ on its three 'sides' or modes by the matrices X, Y, and Z respectively.

An alternative but equivalent way of writing (3.2) is as follows. Define the *outer product* of vectors $x \in \mathbb{R}^l$, $y \in \mathbb{R}^m$, $z \in \mathbb{R}^n$ by

$$x \otimes y \otimes z = [x_i y_j z_k] \in \mathbb{R}^{l \times m \times n},$$

and call a tensor of the form $x \otimes y \otimes z$ a *decomposable tensor* or, if non-zero, a *rank*-1 *tensor.* One may also view (3.2) as a *trilinear combination* (as opposed to a linear combination) of the decomposable tensors given by

$$\mathcal{A} = \sum_{\alpha=1}^{p} \sum_{\beta=1}^{q} \sum_{\gamma=1}^{r} c_{\alpha\beta\gamma} x_\alpha \otimes y_\beta \otimes z_\gamma. \tag{3.4}$$

The vectors $x_1, \dots, x_p \in \mathbb{R}^l$, $y_1, \dots, y_q \in \mathbb{R}^m$, $z_1, \dots, z_r \in \mathbb{R}^n$ are, of course, the column vectors of the respective matrices X, Y, Z above. In this article, we find it more natural to present our algorithms in the form (3.3) and therefore we refrain from using (3.4).

More abstractly, given linear transformations of real vector spaces $T_u : U \to U'$, $T_v : V \to V'$, $T_w : W \to W'$, the functoriality of tensor product [30, 39] (denoted by the usual notation $\otimes$ below) implies that one has an induced linear transformation between the tensor product of the respective vector spaces

$$T_u \otimes T_v \otimes T_w : U \otimes V \otimes W \to U' \otimes V' \otimes W'.$$

The trilinear matrix multiplication above is a coordinatized version of this abstract transformation. In the special case where $U = U'$, $V = V'$, $W = W'$ and T_u, T_v, T_w are (invertible) change-of-basis transformations, the operation in (3.2) describes the manner a (contravariant) 3-tensor transforms under change-of-coordinates of U, V and W.

Associated with the trilinear matrix multiplication (3.3) is the following notion of tensor rank that generalizes the row-rank and column-rank of a matrix. Let $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$. For fixed values of $j \in \{1, \dots, m\}$ and $k \in \{1, \dots, n\}$, consider the column vector, written in a MATLAB-like notation, $\mathcal{A}(:, j, k) \in \mathbb{R}^l$. Likewise we may consider $\mathcal{A}(i, :, k) \in \mathbb{R}^m$ for fixed values of i, k, and $\mathcal{A}(i, j, :) \in \mathbb{R}^n$ for fixed values of i, j. Define

$$\begin{aligned}
r_1(\mathcal{A}) &:= \dim(\operatorname{span}\{\mathcal{A}(:, j, k) \mid 1 \le j \le m, 1 \le k \le n\}),\\
r_2(\mathcal{A}) &:= \dim(\operatorname{span}\{\mathcal{A}(i, :, k) \mid 1 \le i \le l, 1 \le k \le n\}),\\
r_3(\mathcal{A}) &:= \dim(\operatorname{span}\{\mathcal{A}(i, j, :) \mid 1 \le i \le l, 1 \le j \le m\}).
\end{aligned}$$

Note that $\mathbb{R}^{l \times m \times n}$ may also be viewed as $\mathbb{R}^{l \times mn}$. Then $r_1(\mathcal{A})$ is simply the rank of $\mathcal{A}$ regarded as an $l \times mn$ matrix, see the discussion on tensor matricization in [25] with similar interpretations for $r_2(\mathcal{A})$ and $r_3(\mathcal{A})$. The *multilinear rank* of $\mathcal{A}$ is the 3-tuple $(r_1(\mathcal{A}), r_2(\mathcal{A}), r_3(\mathcal{A}))$ and we write

$$\operatorname{rank}(\mathcal{A}) = (r_1(\mathcal{A}), r_2(\mathcal{A}), r_3(\mathcal{A})).$$

We need to 'store' all three numbers as $r_1(\mathcal{A}) \neq r_2(\mathcal{A}) \neq r_3(\mathcal{A})$ in general—a clear departure from the case of matrices where row-rank and column-rank are always equal. Note that a rank-1 tensor must necessarily have multilinear rank $(1, 1, 1)$, i.e. $\operatorname{rank}(x \otimes y \otimes z) = (1, 1, 1)$ if x, y, z are non-zero vectors.

For symmetric tensors, one would be interested in transformation that preserves the symmetry. For a symmetric matrix $C \in \mathsf{S}^2(\mathbb{R}^n)$, this would be

$$S = XCX^{\mathsf{T}}, \qquad s_{ij} = \sum_{\alpha=1}^{r} \sum_{\beta=1}^{r} x_{i\alpha} x_{j\beta} c_{\alpha\beta}\,. \tag{3.5}$$

Matrices act on symmetric order-3 tensors $\mathcal{S} \in \mathsf{S}^3(\mathbb{R}^n)$ via the symmetric version of (3.7)

$$s_{ijk} = \sum_{\alpha=1}^{r} \sum_{\beta=1}^{r} \sum_{\gamma=1}^{r} x_{i\alpha} x_{j\beta} x_{k\gamma} c_{\alpha\beta\gamma} \,. \tag{3.6}$$

We call this operation the *symmetric trilinear multiplication* of $\mathcal{S}$ by matrix X, which in the notation above, is written as

$$\mathcal{S} = (X, X, X) \cdot \mathcal{C}.$$

This is nothing more than the cubic equivalent of (3.5), which in this notation becomes

$$S = XCX^{\mathsf{T}} = (X, X) \cdot C.$$

Informally, (3.6) amounts to multiplying the 3-tensor $\mathcal{A}$ on its three 'sides' or modes by the same matrix X. In the multilinear combination form, this is

$$\mathcal{S} = \sum_{\alpha=1}^{r} \sum_{\beta=1}^{r} \sum_{\gamma=1}^{r} c_{\alpha\beta\gamma} x_\alpha \otimes x_\beta \otimes x_\gamma$$

where the vectors $x_1, \ldots, x_r \in \mathbb{R}^n$ are the column vectors of the matrix X above.

More abstractly, given a linear transformation of real vector spaces $T : V \to V'$, the functoriality of symmetric tensor product [30, 39] implies that one has an induced linear transformation between the tensor product of the respective vector spaces

$$\mathsf{S}^k(T) : \mathsf{S}^k(V) \to \mathsf{S}^k(V').$$

The symmetric trilinear matrix multiplication above is a coordinatized version of this abstract transformation. In the special case where $V = V'$ and T is an (invertible) change-of-basis transformation, the operation in (3.6) describes the manner a symmetric 3-tensor transforms under change-of-coordinates of V.

For a symmetric tensor $\mathcal{S} \in \mathsf{S}^3(\mathbb{R}^n)$, we must have

$$r_1(\mathcal{S}) = r_2(\mathcal{S}) = r_3(\mathcal{S})$$

by its symmetry. See Lemma 9.1 for a short proof. The *symmetric multilinear rank* of $\mathcal{S}$ is the common value, denoted $r_{\mathsf{S}}(\mathcal{S})$. When referring to a symmetric tensor in this article, rank would always mean symmetric multilinear rank; e.g. a rank-s symmetric tensor $\mathcal{S}$ would be one with $r_{\mathsf{S}}(\mathcal{S}) = s$.

We note that there is a different notion of tensor rank and symmetric tensor rank, defined as the number of terms in a minimal decomposition of a tensor (resp. symmetric tensor) into rank-1 tensors (resp. rank-1 symmetric tensors). Associated with this notion of rank are low-rank approximation problems for tensors and symmetric tensors analogous to the ones discussed in this paper. Unfortunately, these are ill-posed problems that may not even have a solution [8, 20]. As such, we will not discuss this other notion of tensor rank. It is implicitly assumed that whenever we discuss tensor rank or symmetric tensor rank, it is with the multilinear rank or symmetric multilinear rank defined above in mind.

3.1. Multilinear approximation as maximization over Grassmannians. Let $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$ be a given third order tensor and consider the problem

$$\min\{\|\mathcal{A}-\mathcal{B}\|_F \mid \operatorname{rank}(\mathcal{B}) \leq (p,q,r)\}.$$

Under this rank constraint, we can write $\mathcal{B}$ in factorized form

$$\mathcal{B} = (X,Y,Z)\cdot\mathcal{C}, \quad b_{ijk} = \sum_{\alpha=1}^{p}\sum_{\beta=1}^{q}\sum_{\gamma=1}^{r} x_{i\alpha}y_{j\beta}z_{k\gamma}c_{\alpha\beta\gamma}, \tag{3.7}$$

where $\mathcal{C} \in \mathbb{R}^{p\times q\times r}$ and $X \in \mathbb{R}^{l\times p}$, $Y \in \mathbb{R}^{m\times q}$, $Z \in \mathbb{R}^{n\times r}$, are full rank matrices. In other words, one would like to solve the following *best multilinear rank approximation problem*,

$$\min_{X,Y,Z,\mathcal{C}} \|\mathcal{A}-(X,Y,Z)\cdot\mathcal{C}\|_F. \tag{3.8}$$

This is the optimization problem underlying the *Tucker model* [54] that originated in psychometrics but has become increasingly popular in other areas of data analysis. In fact, there is no loss of generality if we assume $X^{\mathsf{T}}X = I$, $Y^{\mathsf{T}}Y = I$ and $Z^{\mathsf{T}}Z = I$. This is verified as follows, for any full column-rank matrices $\widetilde{X}$, $\widetilde{Y}$ and $\widetilde{Z}$ we can compute their QR-factorizations

$$\widetilde{X} = XR_X, \quad \widetilde{Y} = YR_Y, \quad \widetilde{Z} = ZR_Z$$

and multiply the right triangular matrices into the core tensor, i.e.

$$\|\mathcal{A}-(\widetilde{X},\widetilde{Y},\widetilde{Z})\cdot\widetilde{\mathcal{C}}\|_F = \|\mathcal{A}-(X,Y,Z)\cdot\mathcal{C}\|_F \quad \text{where} \quad \mathcal{C} = (R_X,R_Y,R_Z)\cdot\widetilde{\mathcal{C}}.$$

With the orthonormal constraints on X ,Y and Z the tensor approximation problem can be viewed as an optimization problem on a product of Stiefel manifolds. Using the identity

$$(U^{\mathsf{T}},V^{\mathsf{T}},W^{\mathsf{T}})\cdot\mathcal{A} \equiv \mathcal{A}\cdot(U,V,W),$$

we can rewrite the tensor approximation problem as a maximization problem with the objective function

$$\Phi(X,Y,Z) = \frac{1}{2}\,\|\mathcal{A}\cdot(X,Y,Z)\|_F^2 \quad \text{s.t.} \quad X^{\mathsf{T}}X = I, \quad Y^{\mathsf{T}}Y = I, \quad Z^{\mathsf{T}}Z = I, \tag{3.9}$$

in which the small core tensor $\mathcal{C}$ is no longer present. See references [16, 25] for the elimination of $\mathcal{C}$. The objective function $\Phi(X,Y,Z)$ is invariant under orthogonal transformation of the variable matrices X,Y,Z from the right. Specifically, for any orthogonal matrices $Q_1 \in \mathrm{O}(p)$, $Q_2 \in \mathrm{O}(q)$ and $Q_3 \in \mathrm{O}(r)$ it holds that $\Phi(X,Y,Z) = \Phi(XQ_1,YQ_2,ZQ_3)$. This homogeneity property implies that $\Phi(X,Y,Z)$ is in fact defined on a product of three Grassmannians $\mathrm{Gr}(l,p)\times\mathrm{Gr}(m,q)\times\mathrm{Gr}(n,r)$.

Let $\mathcal{S} \in \mathsf{S}^3(\mathbb{R}^n)$ be a given symmetric 3-tensor and consider the problem

$$\min\{\|\mathcal{S}-\mathcal{T}\|_F \mid \mathcal{T} \in \mathsf{S}^3(\mathbb{R}^n),\ r_{\mathsf{S}}(\mathcal{S}) \leq r\}.$$

Under this rank constraint, we can write $\mathcal{T}$ in factorized form

$$\mathcal{T} = (X,X,X)\cdot\mathcal{C}, \quad t_{ijk} = \sum_{\alpha=1}^{r}\sum_{\beta=1}^{r}\sum_{\gamma=1}^{r} x_{i\alpha}x_{j\beta}x_{k\gamma}c_{\alpha\beta\gamma},$$

where $\mathcal{C} \in \mathsf{S}^3(\mathbb{R}^r)$ and $X \in \mathbb{R}^{n\times r}$ is a full-rank matrix. In other words, one would like to solve the following *best symmetric multilinear rank approximation problem*,

$$\min\{\|\mathcal{A} - (X, X, X) \cdot \mathcal{C}\|_F \mid X \in \mathrm{O}(n, r),\ \mathcal{C} \in \mathsf{S}^3(\mathbb{R}^r)\}.$$

As with the general case, there is no loss of generality if we assume $X^{\mathsf{T}}X = I$. With the orthonormal constraints on X, the tensor approximation problem can be viewed as an optimization problem on a single Stiefel manifold (as opposed to a product of Stiefel manifolds in (3.9)). Using the identity

$$(V^{\mathsf{T}}, V^{\mathsf{T}}, V^{\mathsf{T}}) \cdot \mathcal{S} \equiv \mathcal{S} \cdot (V, V, V),$$

we may again rewrite the tensor approximation problem as a maximization problem with the objective function

$$\Phi(X) = \frac{1}{2}\,\|\mathcal{S} \cdot (X, X, X)\|_F^2 \quad \text{s.t.} \quad X^{\mathsf{T}}X = I,$$

in which the core-tensor $\mathcal{C}$ is no longer present. As with the general case, the objective function $\Phi(X)$ also has an invariance property, namely $\Phi(X) = \Phi(XQ)$ for any orthogonal $Q \in \mathrm{O}(r)$. As before, this homogeneity property implies that $\Phi(X)$ is well-defined on a single Grassmannian $\mathrm{Gr}(n, r)$.

These multilinear approximation problems may be viewed as 'dimension reduction' or 'rank reduction' for tensors and symmetric tensors respectively. In general, a matrix requires $O(n^2)$ storage and an order-k tensor requires $O(n^k)$ storage. While it is sometimes important to perform dimension reduction to a matrix, a dimension reduction is almost always necessary if one wants to work effectively with a tensor of higher order. A dimension reduction of a matrix $A \in \mathbb{R}^{n\times n}$ of the form $A \approx UCV^{\mathsf{T}}$, where $U, V \in \mathrm{O}(n, r)$ and diagonal $C \in \mathbb{R}^{r\times r}$ reduces dimension from $O(n^2)$ to $O(nr + r)$. A dimension reduction of a tensor $\mathcal{A} \in \mathbb{R}^{n\times\cdots\times n}$ of the form $\mathcal{A} \approx (Q_1, \dots, Q_k) \cdot \mathcal{C}$ where $Q_1, \dots, Q_k \in \mathrm{O}(n, r)$ and $\mathcal{C} \in \mathbb{R}^{r\times\cdots\times r}$, reduces dimension from $O(n^k)$ to $O(knr + r^k)$. If r is significantly smaller than n, e.g. $r = O(n^{1/k})$, then a dimension reduction in the higher order case could reduce problem size by orders of magnitude.

4. Optimization in Euclidean space and on Riemannian manifolds.

In this section we discuss the necessary modifications for generalizing an optimization algorithm from Euclidean space to Riemannian manifolds. Specifically we consider the quasi-Newton methods with BFGS and *limited memory* BFGS (L-BFGS) updates. First we state the expressions in Euclidean space and then we point out what needs to be modified. The convergence properties of quasi-Newton methods defined on manifolds were established by Gabay [27]. Numerical treatment of algorithms on the Grassmannian are given in [24, 1, 52, 42, 32]. A recent book on optimization on manifolds is [2]. In this and the next three sections, i.e. Sections 4 through 7, we will discuss our algorithms in the context of minimization problems, as is conventional. It will of course be trivial to modify them for maximization problem. Indeed the tensor approximation problems discussed in Sections 8 through 11 will all be solved as maximization problems on Grassmannians or product of Grassmannians.

4.1. BFGS updates in Euclidean Space.

Assume that we want to minimize a nonlinear real valued function $f(x)$ where $x \in \mathbb{R}^n$. As is well-known, in quasi-Newton methods, one solves

$$H_k p_k = -g_k, \tag{4.1}$$

to obtain the direction of descent p_k from the current iterate x_k and the gradient $g_k = \nabla f(x_k)$ at x_k. Unlike Newton method, which uses the exact Hessian for H_k, in (4.1) H_k is only an approximation of the Hessian at x_k. After computing the (search) direction p_k one obtains the next iterate as $x_{k+1} = x_k + t_k p_k$ in which the step length t_k is usually given by a line search method satisfying the Wolfe or the Goldstein conditions [46]. Instead of recomputing the Hessian at each new iterate x_{k+1}, it is updated from the previous approximation. The BFGS update has the following form,

$$H_{k+1} = H_k - \frac{H_k s_k s_k^{\mathsf{T}} H_k}{s_k^{\mathsf{T}} H_k s_k} + \frac{y_k y_k^{\mathsf{T}}}{y_k^{\mathsf{T}} s_k}, \tag{4.2}$$

where

$$s_k = x_{k+1} - x_k = t_k p_k, \tag{4.3}$$

$$y_k = g_{k+1} - g_k. \tag{4.4}$$

Quasi-Newton methods with BFGS updates are considered to be the most computationally efficient algorithms for minimization of general nonlinear functions. This efficiency is obtained by computing a new Hessian approximation as a rank-2 modification of the previous Hessian. The convergence of quasi-Newton methods is superlinear in a vicinity of a local minimum. In most cases the quadratic convergence of the Newton method is outperformed by quasi-Newton methods since each iteration is computationally much cheaper than a Newton iteration. A thorough study of quasi-Newton methods may be found in [46, 23]. The reader is reminded that quasi-Newton methods do not necessarily converge to local minima but only to stationary points. Nevertheless, using the Hessians that we derived in Sections 8.1 and 9.3 (for the general and symmetric cases respectively), the nature of these stationary points can often be determined.

4.2. Quasi-Newton methods on a Riemannian manifold. We will give a very brief overview of Riemannian geometry tailored specifically to our needs in this paper. First we sketch the modifications that are needed in order for an optimization algorithm to be well-defined when the objective function is defined on a manifold. For details and proof, the reader should refer to standard literature on differential and Riemannian geometry [11, 5, 2]. Informally a manifold, denoted with M, is an object locally homeomorphic to $\mathbb{R}^n$. We will regard M as a submanifold of some high-dimensional ambient Euclidean space $\mathbb{R}^N$. Our objective function $f : M \to \mathbb{R}$ will be assumed to have (at least) continuous second order partial derivatives. We will write $f(x)$, $g(x)$, and $H(x)$ for the value of the function, the gradient, and the Hessian at $x \in M$. These will be reviewed in greater detail in the following.

Equations (4.1)–(4.4) are the basis of any algorithmic implementation involving BFGS or L-BFGS updates. The key operations are (1) computation of the gradient, (2) computation of the Hessian or its approximation, (3) subtraction of iterates, e.g. to get s_k or x_{k+1} and (4) subtraction of gradients. Each of these points needs to be modified in order for these operations to be well-defined on manifolds.

Computation of the gradient. The gradient $g(x) = \operatorname{grad} f(x) = \nabla f(x)$ at $x \in M$ of a real-valued function defined on a manifold, $f : M \to \mathbb{R}$, $M \ni x \mapsto f(x) \in \mathbb{R}$, is a vector in the tangent space $\mathbf{T}_x = \mathbf{T}_x(M)$ of the manifold at the given point x. We write $g \in \mathbf{T}_x$.

To facilitate computations, we will often embed our manifold M in some ambient Euclidean space $\mathbb{R}^N$ and in turn endow M with a system of global coordinates $(x_1, \dots, x_N)$ where N is usually larger than $d := \dim(M)$, the intrinsic dimension of M. The function f then inherits an expression in terms of $x_1, \dots, x_N$, say, $f(x) = \tilde{f}(x_1, \dots, x_N)$. We would like to caution our readers that computing ∇f on M is not a matter of simply taking partial derivatives of $\tilde{f}$ with respect to $x_1, \dots, x_N$. An easy way to observe this is that ∇f is a d-tuple when expressed in local coordinates whereas $\left(\partial \tilde{f}/\partial x_1, \dots, \partial \tilde{f}/\partial x_N\right)$ is an N-tuple.

Computation of the Hessian or its approximation. The Hessian $H(x) = \operatorname{Hess} f(x) = \nabla^2 f(x)$ at $x \in M$ of a function $f : M \to \mathbb{R}$ is a linear transformation of the tangent space $\mathbf{T}_x$ to itself, i.e.

$$H(x) : \mathbf{T}_x \to \mathbf{T}_x. \tag{4.5}$$

As in the case of gradient, when f is expressed in terms of global coordinates, differentiating the expression twice will in general not give the correct Hessian.

Updating the current iterate. Given an iterate $x_k \in M$, a step length t_k and a search direction $p_k \in \mathbf{T}_{x_k}$ the update $x_{k+1} = x_k + t_k p_k$ will in general not be a point on the manifold. The corresponding operation on a manifold is to move along the geodesic curve of the manifold given by the direction p_k. Geodesics on manifolds correspond to straight lines in Euclidean spaces. The operation $s_k = x_{k+1} - x_k$ is undefined in general when the points on the right hand side belong to a general manifold.

Updating vectors and operators. The quantities s_k and y_k are in fact tangent vectors and the Hessians (or Hessian approximations) are linear operators all defined at a specific point of the manifold. A given Hessian $H(x)$ is only defined at a point $x \in M$ and correspondingly only acts on vectors in the tangent space $\mathbf{T}_x$. In the right hand side of the BFGS update (4.2) all quantities need to be defined at the same point x_k in order to have well-defined operations between the terms. In addition the resulting sum will define an operator at a new point x_{k+1}. The notion of parallel transporting vectors along geodesic curves resolves all of these issues. The operations to the Hessian are similar and involves parallel transport operation back and forth between two different points.

5. Grassmann geodesics and parallel transport of vectors.

In this paper, the Riemannian manifold M of most interest to us is $\operatorname{Gr}(n, r)$, the Grassmannian or Grassmannian of r-planes in $\mathbb{R}^n$. Our discussion will proceed with $M = \operatorname{Gr}(n, r)$.

5.1. Algorithmic considerations.

Representing points on Grassmannians as (equivalence classes of) matrices allows us to take advantage of matrix arithmetic and matrix algorithms, as well as the readily available libraries of highly optimized and robust matrix computational softwares developed over the last five decades [3, 40]. A major observation of [24] is the realization that common differential geometric operations on points of Grassmann and Stiefel manifolds can all be represented in terms of matrix operations. For our purposes, the two most important operations are (1) the determination of a geodesic at a point along a tangent vector and (2) the parallel transport of a tangent vector along a geodesic. On a product of Grassmannians, these operations may likewise be represented in terms of matrix operations [25]. We will give explicit expressions for geodesic curves on Grassmannians and two different ways of parallel transporting tangent vectors.

5.2. Grassmannians in terms of matrices.

First we will review some preliminary materials from [24]. A point $\langle X \rangle$ on the Grassmannian $\operatorname{Gr}(n, r)$ is an equivalence class of orthonormal matrices whose columns form an orthonormal basis for an

r-dimensional subspace of $\mathbb{R}^n$. Explicitly, we write

$$\langle X \rangle = \{XQ \in \mathrm{O}(n,r) \mid Q \in \mathrm{O}(r)\},$$

where $X \in \mathrm{O}(n,r)$, i.e. X is an $n \times r$ matrix and $X^{\mathsf{T}} X = I_r$. The set $\mathrm{O}(n,r) = \{X \in \mathbb{R}^{n \times r} \mid X^{\mathsf{T}} X = I_r\}$ is also a manifold, often called the Stiefel manifold. When $n = r$, $\mathrm{O}(r,r) = \mathrm{O}(r)$ is the orthogonal group. It is easy to see that the dimensions of these manifolds are

$$\dim(\mathrm{O}(r)) = \frac{1}{2} r(r-1), \quad \dim(\mathrm{O}(n,r)) = nr - \frac{1}{2} r(r+1), \quad \dim(\mathrm{Gr}(n,r)) = r(n-r).$$

In order to use standard linear algebra in our computations, we will not be able to work with a whole equivalence class of matrices. So by a point on a Grassmannian, we will always mean some $X \in \langle X \rangle$ that represents the equivalence class. The functions f that we optimize in this paper will take orthonormal matrices in $\mathrm{O}(n,r)$ as arguments but will always be well-defined on Grassmannians: given two representatives $X_1, X_2 \in \langle X \rangle$, we will have $f(X_1) = f(X_2)$, i.e.

$$f(XQ) = f(X) \quad \text{for every } Q \in \mathrm{O}(r).$$

Abusing notations slightly, we will sometimes write $f : \mathrm{Gr}(n,r) \to \mathbb{R}$.

The tangent space $\mathbf{T}_X$, where $X \in \mathrm{Gr}(n,r)$, is an affine vector space with elements in $\mathbb{R}^{n \times r}$. It can be shown that any element $\Delta \in \mathbf{T}_X$ satisfies

$$X^{\mathsf{T}} \Delta = 0.$$

The projection on the tangent space is

$$\Pi_X = I - XX^{\mathsf{T}} = X_\perp X_\perp^{\mathsf{T}},$$

where $X_\perp$ is an orthogonal complement of X, i.e. the square matrix $[X\ X_\perp]$ is an $n \times n$ orthogonal matrix. Since by definition $X^{\mathsf{T}} X_\perp = 0$, any tangent vector can also be written as $\Delta = X_\perp D$ where D is an $(n-r) \times r$ matrix. This shows that the columns of $X_\perp$ may be interpreted as a basis for $\mathbf{T}_X$. We say that Δ is a *global coordinate representation* and D is a *local coordinate representation* of the same tangent. Note that the number of degrees of freedom in D equals the dimension of the tangent space $\mathbf{T}_X$, which is $r(n-r)$. It follows that for a given tangent in global coordinates Δ, its local coordinate representation is given by $D = X_\perp^{\mathsf{T}} \Delta$. Observe that to a given local representation D of a tangent there is an associated basis matrix $X_\perp$. Tangent vectors are also embedded in $\mathbb{R}^{nr}$ since in global coordinates they are given by $n \times r$ matrices. We will define algorithms using both global coordinates as well as intrinsic local coordinates. When using global coordinates, the Grassmannian $\mathrm{Gr}(n,r)$ is (isometrically) embedded in the Euclidean space $\mathbb{R}^{n \times r}$ and a product of Grassmannians in a corresponding product of Euclidean spaces. The use of Plücker coordinates to represent points on Grassmannian is not useful for our purpose.

5.3. Geodesics. Let $X \in \mathrm{Gr}(n,r)$ and Δ be a tangent vector at X, i.e. $\Delta \in \mathbf{T}_X$. The geodesic path from X in the direction Δ is given by

$$X(t) = [XV \quad U] \begin{bmatrix} \cos \Sigma t \\ \sin \Sigma t \end{bmatrix} V^{\mathsf{T}}, \tag{5.1}$$

where $\Delta = U\Sigma V^{\mathsf{T}}$ is the thin SVD and we identify $X(0) \equiv X$. Observe that omitting the last V in (5.1) will give the same path on the manifold but with a different[1] representation. This information is useful because some algorithms require a consistency in the matrix representations along a path but other algorithms do not. For example, in a Newton-Grassmann algorithm we may omit the second V [25] but in quasi-Newton-Grassmann algorithms V is necessary.

5.4. Parallel transport in global and local coordinates. Let X be a point on a Grassmannian and consider the geodesic given by the tangent vector $\Delta \in \mathbf{T}_X$. The matrix expression for the parallel transport of an arbitrary tangent vector $\Delta_2 \in \mathbf{T}_X$ is given by

$$\mathbf{T}_{X(t)} \ni \Delta_2(t) = \left([XV \quad U] \begin{bmatrix} -\sin \Sigma t \\ \cos \Sigma t \end{bmatrix} U^{\mathsf{T}} + (I - UU^{\mathsf{T}}) \right) \Delta_2 \equiv T_{X,\Delta}(t)\Delta_2, \tag{5.2}$$

where $\Delta = U\Sigma V^T$ is the thin SVD and we define $T_{X,\Delta}(t)$ to be the *parallel transport matrix*[2] from the point X in the direction Δ. If $\Delta_2 = \Delta$ expression (5.2) can be simplified.

Let $X \in \mathrm{Gr}(n,r)$, $\Delta \in \mathbf{T}_X$, and $X_\perp$ be an orthogonal complement of X so that $[X \; X_\perp]$ is orthogonal. Recall that we may write $\Delta = X_\perp D$, where we view $X_\perp$ as a basis for $\mathbf{T}_X$ and D as a local coordinate representation of the tangent vector Δ. Assuming that $X(t)$ is the geodesic curve given in (5.1), the parallel transport of the corresponding basis $X_\perp(t)$ for $\mathbf{T}_{X(t)}$ is given by

$$X_\perp(t) = T_{X,\Delta}(t) X_\perp, \tag{5.3}$$

where $T_{X,\Delta}(t)$ is the transport matrix defined in (5.2). It is straightforward to show that the matrix $[X(t) \; X_\perp(t)]$ is orthogonal for all t, i.e.

$$X_\perp^{\mathsf{T}}(t) X_\perp(t) = I_{n-r} \quad \text{and} \quad X_\perp^{\mathsf{T}}(t) X(t) = 0 \quad \text{for every } t.$$

Using (5.3) we can write the parallel transport of a tangent vector Δ_2 as

$$\Delta_2(t) = T(t)\Delta_2 = T(t) X_\perp D_2 = X_\perp(t) D_2. \tag{5.4}$$

Equation (5.4) shows that the local coordinate representation of the tangent vector is constant at all points of the geodesic path $X(t)$ when the basis for $\mathbf{T}_{X(t)}$ is given by $X_\perp(t)$. The global coordinate representation, on the other hand, varies with t. This is an important observation since explicit parallel transport of tangents and Hessians (cf. Section 6.5) can be avoided if the algorithm is implemented using local coordinates. The computational complexity for these two operations[3] are $O(n^2 r)$ and $O(n^3 r^2)$ respectively. The cost saved in avoiding parallel transports of tangents and Hessians is paid instead in the parallel transport of the basis $X_\perp$. This matrix is computed in the first iteration at a cost of at most $O(n^3)$ operations, and in each of the consecutive iterations, it is parallel transported at a cost of $O(n^2(n-r))$ operations. There are also differences in memory requirements: In global coordinates tangents are stored as $n \times r$ matrices and Hessians as $nr \times nr$ matrices, whereas in local coordinates tangents and Hessians are stored as $(n-r) \times r$ and $(n-r)r \times (n-r)r$ matrices

[1] A given matrix representation of a point on a Grassmannian can be postmultiplied by any orthogonal matrix, giving a new representation of the same point.

[2] We will often omit subscripts X and Δ and just write $T(t)$ when there is no risk for confusion.

[3] Here we assume that the parallel transport operator has been computed and stored.

respectively. Local coordinate implementation also requires the additional storage of $X_\perp$ as an $n \times (n-r)$ matrix. In most cases, the local coordinate implementation provides greater computational and memory savings, as we observed in our numerical experiments.

By introducing the thin SVD of $D = \bar{U}\bar{\Sigma}\bar{V}^\mathsf{T}$, we can also write (5.3) as

$$X_\perp(t) = T(t)X_\perp = \begin{bmatrix} X\bar{V} & X_\perp\bar{U} \end{bmatrix} \begin{bmatrix} -\sin\bar{\Sigma}t \\ \cos\bar{\Sigma}t \end{bmatrix} \bar{U}^\mathsf{T} + X_\perp \left(I - \bar{U}\bar{U}^\mathsf{T}\right).$$

This follows from the identities

$$\bar{U} = X_\perp^\mathsf{T} U, \quad \bar{\Sigma} = \Sigma, \quad \bar{V} = V,$$

which are obtained from $\Delta = U\Sigma V^\mathsf{T} = X_\perp D$. Using this, we will derive a general property of inner products for our later use.

THEOREM 5.1. *Let $X \in \operatorname{Gr}(n,r)$ and $\Delta, \Delta_1, \Delta_2 \in \mathbf{T}_X$. Define the transport matrix in the direction Δ*

$$T_{X,\Delta}(t) = T(t) = \begin{bmatrix} XV & U \end{bmatrix} \begin{bmatrix} -\sin\Sigma t \\ \cos\Sigma t \end{bmatrix} U^\mathsf{T} + (I - UU^\mathsf{T}),$$

where $\Delta = U\Sigma V^\mathsf{T}$ is the thin SVD. *Then*

$$\langle \Delta_1, \Delta_2 \rangle = \langle \Delta_1(t), \Delta_2(t) \rangle \qquad \textit{for every } t,$$

where $\Delta_1(t) = T_{X,\Delta}(t)\Delta_1$ and $\Delta_2(t) = T_{X,\Delta}(t)\Delta_2$ are parallel transported tangents.

Proof. The proof is a direct consequence of the Levi-Civita connection used in the definition of the parallel transport of tangents. Or we can use the canonical inner product on the Grassmannian $\langle \Delta_1, \Delta_2 \rangle = \operatorname{tr}(\Delta_1^\mathsf{T}\Delta_2)$. Then, inserting the parallel transported tangents we obtain

$$\begin{aligned} \langle \Delta_1(t), \Delta_2(t) \rangle &= \operatorname{tr}\big(\Delta_1(t)^\mathsf{T}\Delta_2(t)\big) = \operatorname{tr}\big(\Delta_1^\mathsf{T} T(t)^\mathsf{T} T(t)\Delta_2\big) \\ &= \operatorname{tr}\left(\Delta_1^\mathsf{T}\left(I - U\sin(\Sigma t)V^\mathsf{T}X^\mathsf{T} - XV\sin(\Sigma t)U^\mathsf{T}\right)\Delta_2\right) \\ &= \operatorname{tr}\left(\Delta_1^\mathsf{T}\Delta_2\right) - \operatorname{tr}\left(\Delta_1^\mathsf{T}U\sin(\Sigma t)V^\mathsf{T}X^\mathsf{T}\Delta_2\right) - \operatorname{tr}\left(\Delta_1^\mathsf{T}XV\sin(\Sigma t)U^\mathsf{T}\Delta_2\right). \end{aligned}$$

The proof is concluded by observing that the second and third terms after the last equality are zero because $X^\mathsf{T}\Delta_2 = 0$ and $\Delta_1^\mathsf{T}X = 0$. □

Remark. We would like to point out that it is the tangents that are parallel transported. In global coordinates tangents are represented by $n \times r$ matrices and their parallel transport is given by (5.2). On the other hand, in local coordinates, tangents are represented by $(n-r) \times r$ matrices and this representation does not change when the basis for the tangent space is parallel transported according to (5.3). In other words, in local coordinates, parallel transported tangents are represented by the *same matrix* at every point along a geodesic. This is to be contrasted with the global coordinate representation of points on the manifold $\operatorname{Gr}(n,r)$, which are $n \times r$ matrices that differ from point to point on a geodesic.

6. Quasi-Newton methods with BFGS updates on a Grassmannian.

In this section we will present the necessary modifications in order for BFGS updates to be well-defined on a Grassmannian. We will write $f(X)$ instead of $f(x)$ since the argument to the function is a point on a Grassmannian and represented by a matrix $X = [x_{ij}]_{i,j=1}^{n,r} \in \mathbb{R}^{n\times r}$. Similarly the quantities s_k and y_k from equations (4.3) and (4.4) will be written as matrices S_k and Y_k, respectively.

6.1. Computations in global coordinates. We describe here the expressions of various quantities required for defining BFGS updates in global coordinates. The corresponding expressions in local coordinates are in the next section.

Gradient. The Grassmann gradient of the objective function $f(X)$ is given by

$$\nabla f(X) = \Pi_X \frac{\partial f}{\partial X}, \quad \frac{\partial f}{\partial X} := \left[\frac{\partial f}{\partial x_{ij}}\right]_{i,j=1}^{n,r}, \tag{6.1}$$

where $\Pi_X = I - XX^\mathsf{T}$ is the projection on the tangent space $\mathbf{T}_X$.

Computing S_k. We will now modify the operations in equation (4.3), i.e.

$$s_k = x_{k+1} - x_k = t_k p_k,$$

so that it is valid on a Grassmannian. Let X_{k+1} be given by $X_{k+1} = X_k(t_k)$ where the geodesic path originating from X_k is defined by the tangent (or search direction) $\Delta \in \mathbf{T}_{X_k}$. The step size is given by t_k. We will later assume that $S_k \in \mathbf{T}_{X_{k+1}}$ and with the tangent $\Delta \in \mathbf{T}_{X_k}$, corresponding to p_k, we conclude that

$$S_k = t_k \Delta(t_k) = t_k T(t_k)\Delta, \tag{6.2}$$

where $T(t_k)$ is the transport matrix defined in (5.2).

Computing Y_k. Similarly, we will translate

$$y_k = g_{k+1} - g_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

from equation (4.4). Computing the Grassmann gradient at X_{k+1} we get $\nabla f(X_{k+1}) \in \mathbf{T}_{X_{k+1}}$. Parallel transporting $\nabla f(X_k) \in \mathbf{T}_{X_k}$ along the direction Δ and subtracting the two gradients as in equation (4.4) we get

$$\mathbf{T}_{X_{k+1}} \ni Y_k = \nabla f(X_{k+1}) - T(t_k)\nabla f(X_k), \tag{6.3}$$

where we again use the transport matrix (5.2). Recall that Y_k corresponds to y_k.

The expressions for ∇f, S_k and Y_k are given in matrix form, i.e. they have the same dimensions as the variable matrix X. It is straightforward to obtain the corresponding vectorized expressions. For example, with $\partial f/\partial X \in \mathbb{R}^{n\times r}$, the vector form of the Grassmann gradient is given by

$$\operatorname{vec}(\nabla f) = (I_r \otimes \Pi_X) \operatorname{vec}\left(\frac{\partial f}{\partial X}\right) \in \mathbb{R}^{nr}.$$

where $\operatorname{vec}(\cdot)$ is the ordinary column-wise vectorization of a matrix. For simplicity we switch to this presentation when working with the Hessian.

Updating the Hessian (approximation). Identify the tangents (matrices) $\Delta \in \mathbb{R}^{n\times r}$ with vectors in $\mathbb{R}^{nr}$ and assume that the Grassmann Hessian

$$\mathbb{R}^{nr\times nr} \ni H_k = H(X_k) : \mathbf{T}_{X_k} \to \mathbf{T}_{X_k}$$

at the iterate X_k is given. Then

$$\bar{H}_k = (I_r \otimes T(t_k)) H_k (I_r \otimes \widetilde{T}(t_k)) : \mathbf{T}_{X_{k+1}} \to \mathbf{T}_{X_{k+1}}, \tag{6.4}$$

is the transported Hessian defined at iterate X_{k+1}. As previously $T(t_k)$ is the transport matrix from X_k to X_{k+1} given in (5.2) and $\widetilde{T}(t_k)$ is the transport matrix from

X_{k+1} to X_k along the same geodesic path. Informally we can describe the operations in (6.4) as follows. Tangent vectors from $\mathbf{T}_{X_{k+1}}$ are transported with $\widetilde{T}(t_k)$ to $\mathbf{T}_{X_k}$ on which H_k is defined. The Hessian H_k transforms the transported vectors on $\mathbf{T}_{X_k}$ and the result is then forwarded with $T(t_k)$ to $\mathbf{T}_{X_{k+1}}$.

Since all vectors and matrices are now defined at X_{k+1}, the BFGS update is computed using equation (4.2) in which we replace H_k with $(I_r \otimes T(t_k))H_k(I_r \otimes \widetilde{T}(t_k))$ and use $s_k = \mathrm{vec}(S_k)$ and $y_k = \mathrm{vec}(Y_k)$ from equations (6.2) and (6.3) respectively.

6.2. Computations in local coordinates. Using local coordinates we obtain several simplifications. First given the current iterate X we need the orthogonal complement $X_\perp$. When it is obvious we will omit the iteration subscript k.

Grassmann gradient. In local coordinates the Grassmann gradient is given by

$$\nabla \widehat{f} = X_\perp^\mathsf{T} \nabla f = X_\perp^\mathsf{T} \Pi_X \frac{\partial f}{\partial X} = X_\perp^\mathsf{T} \frac{\partial f}{\partial X}, \tag{6.5}$$

where we have used the global coordinate representation for the Grassmann gradient (6.1). We denote quantities in local coordinates with a hat to distinguish them from those in global coordinates.

Parallel transporting the basis $X_\perp$. It is necessary to parallel transport the basis matrix $X_\perp$ from the current iterate X_k to the next iterate X_{k+1}. Only in this basis will the local coordinates of parallel transported tangents be constant. The parallel transport of the basis matrix is given by equation (5.3).

Computing $\widehat{S}_k$ and $\widehat{Y}_k$. According to the discussion in Section 5.4, in the transported tangent basis $X_\perp(t)$, the local coordinate representation of any tangent is constant. Specifically this is true for $\widehat{S}_k$ and $\widehat{Y}_k$. The two quantities are obtained with the same expressions as in the Euclidean space.

Updating the Hessian (approximation). Since explicit parallel transport is not required in local coordinates, the Hessian remains constant as well. The local coordinate representations for H_k in the basis $X_\perp(t)$ for points on the geodesic path $X(t)$ are the same. This statement is proven in Theorem 6.4.

The effect of using local coordinates on the Grassmannian is only in the geodesic transport of the current point X_k and its orthogonal complement $X_{k\perp}$. The transported orthogonal complement $X_{k\perp}(t_k)$ is used to compute the Grassmann gradient $\nabla\widehat{f}(X_{k+1})$ in local coordinates at the new iterate $X_{k+1} = X_k(t_k)$. Assuming tangents are in local coordinates at X_k in the basis $X_{k\perp}$ and tangents at X_{k+1} are given in the basis $X_{k\perp}(t_k)$, the BFGS update is given by (4.2), i.e. exactly the same update as in the Euclidean space. This is a major advantage compared with the global coordinate update of H_k. In global coordinates H_k is multiplied by matrices from the left and from the right (6.4). This is relatively expensive since the BFGS update itself is just a rank-2 update, see equation (4.2).

6.3. BFGS update in tensor form. It is not difficult to see that if the gradient $\nabla f(X)$ is written as an $n \times r$ matrix, then the second derivative will take the form of a 4-tensor $\mathcal{H}_k \in \mathbb{R}^{n\times r\times n\times r}$. The BFGS update (4.2) can be written in a different form using the tensor structure of the Hessian. The action of this operator will map matrices to matrices. Assuming $s_k = \mathrm{vec}(S_k)$ and H_k is a matricized form of $\mathcal{H}_k$, the matrix-vector contraction $H_k s_k$ can be written as $\langle \mathcal{H}_k, S_k\rangle_{1,2}$. Obviously the result of the first operation is a vector whereas the result of the second operation is a matrix, and of course $H_k s_k = \mathrm{vec}\left(\langle \mathcal{H}_k, S_k\rangle_{1,2}\right)$.

Furthermore keeping the tangents, e.g. S_k or Y_k, in matrix form, the parallel transport of the Hessian in equation (6.4) can be written as a multilinear product between $\mathcal{H}_k \in \mathbb{R}^{n\times r\times n\times r}$ and the two transport matrices $T(t_k)$ and $\widetilde{T}(t_k)$, both in $\mathbb{R}^{n\times n}$, along the first and third modes,

$$\mathbb{R}^{n\times r\times n\times r} \ni \bar{\mathcal{H}} = \mathcal{H}_k \cdot (T(t_k), I, \widetilde{T}(t_k), I).$$

Finally, noting that the outer product between vectors corresponds to tensor products between matrices the BFGS update becomes[4]

$$\mathcal{H}_{k+1} = \bar{\mathcal{H}}_k + \frac{\langle \bar{\mathcal{H}}_k, S_k\rangle_{1,2} \otimes \langle \bar{\mathcal{H}}_k, S_k\rangle_{1,2}}{\langle\langle \bar{\mathcal{H}}_k, S_k\rangle_{1,2}, S_k\rangle} + \frac{Y_k \otimes Y_k}{\langle S_k, Y_k\rangle}, \tag{6.6}$$

where the matrices $S_k, Y_k \in \mathbf{T}_{X_{k+1}}$ are given by (6.2) and (6.3) respectively.

In local coordinates the update is even simpler since we do not have to parallel transport the Hessian operator,

$$\widehat{\mathcal{H}}_{k+1} = \widehat{\mathcal{H}}_k + \frac{\langle \widehat{\mathcal{H}}_k, \widehat{S}_k\rangle_{1,2} \otimes \langle \widehat{\mathcal{H}}_k, \widehat{S}_k\rangle_{1,2}}{\langle\langle \widehat{\mathcal{H}}_k, \widehat{S}_k\rangle_{1,2}, \widehat{S}_k\rangle} + \frac{\widehat{Y}_k \otimes \widehat{Y}_k}{\langle \widehat{S}_k, \widehat{Y}_k\rangle}, \tag{6.7}$$

where $\widehat{\mathcal{H}}_k \in \mathbb{R}^{(n-r)\times r\times (n-r)\times r}$ and $\widehat{S}_k, \widehat{Y}_k \in \mathbb{R}^{(n-r)\times r}$. The *hat* indicates that the corresponding variables are in local coordinates.

6.4. BFGS update on a product of Grassmannians. Assume now that the objective function f is defined on a product of three[5] Grassmannians, i.e.

$$f : \mathrm{Gr}(l,p) \times \mathrm{Gr}(m,q) \times \mathrm{Gr}(n,r) \to \mathbb{R},$$

and is twice continuously differentiable. We write $f(X,Y,Z)$ where $X \in \mathrm{Gr}(l,p)$, $Y \in \mathrm{Gr}(m,q)$ and $Z \in \mathrm{Gr}(n,r)$. The Hessian of the objective function will have a 'block tensor' structure but the blocks will not have conforming dimensions. The action of a (approximate) Hessian operator on tangents $\Delta_X \in \mathbf{T}_X$, $\Delta_Y \in \mathbf{T}_Y$ and $\Delta_Z \in \mathbf{T}_Z$ may be written symbolically as

$$\begin{aligned}
&\begin{bmatrix} \mathcal{H}_{XX} & \mathcal{H}_{XY} & \mathcal{H}_{XZ} \\ \mathcal{H}_{YX} & \mathcal{H}_{YY} & \mathcal{H}_{YZ} \\ \mathcal{H}_{ZX} & \mathcal{H}_{ZY} & \mathcal{H}_{ZZ} \end{bmatrix} \begin{bmatrix} \Delta_X \\ \Delta_Y \\ \Delta_Z \end{bmatrix} \qquad (6.8)\\
&= \begin{bmatrix} \langle \mathcal{H}_{XX}, \Delta_X\rangle_{3,4;1,2} + \langle \mathcal{H}_{XY}, \Delta_Y\rangle_{3,4;1,2} + \langle \mathcal{H}_{XZ}, \Delta_Z\rangle_{3,4;1,2} \\ \langle \mathcal{H}_{YX}, \Delta_X\rangle_{3,4;1,2} + \langle \mathcal{H}_{YY}, \Delta_Y\rangle_{3,4;1,2} + \langle \mathcal{H}_{YZ}, \Delta_Z\rangle_{3,4;1,2} \\ \langle \mathcal{H}_{ZX}, \Delta_X\rangle_{3,4;1,2} + \langle \mathcal{H}_{ZY}, \Delta_Y\rangle_{3,4;1,2} + \langle \mathcal{H}_{ZZ}, \Delta_Z\rangle_{3,4;1,2} \end{bmatrix} \\
&= \begin{bmatrix} \langle \mathcal{H}_{XX}, \Delta_X\rangle_{1,2} + \langle \mathcal{H}_{YX}, \Delta_Y\rangle_{1,2} + \langle \mathcal{H}_{ZX}, \Delta_Z\rangle_{1,2} \\ \langle \mathcal{H}_{XY}, \Delta_X\rangle_{1,2} + \langle \mathcal{H}_{YY}, \Delta_Y\rangle_{1,2} + \langle \mathcal{H}_{ZY}, \Delta_Z\rangle_{1,2} \\ \langle \mathcal{H}_{XZ}, \Delta_X\rangle_{1,2} + \langle \mathcal{H}_{YZ}, \Delta_Y\rangle_{1,2} + \langle \mathcal{H}_{ZZ}, \Delta_Z\rangle_{1,2} \end{bmatrix}.
\end{aligned}$$

The blocks of the Hessian are 4-tensors and elements of the tangent spaces are matrices. The result of the operation[3] is a triplet where each element is in the corresponding tangent space. For example $\mathcal{H}_{XX}$ is an $l \times p \times l \times p$ tensor which acts on the tangent matrix Δ_X of size $l \times p$ with the result $\langle \mathcal{H}_{XX}, \Delta_X\rangle_{1,2} \in \mathbf{T}_X$. Off diagonal example

[4] The contractions denoted by $\langle\cdot,\cdot\rangle_*$ are defined in Appendix A.

[5] We assume $k = 3$ for notational simplicity; generalization of these discussions to arbitrary k is straightforward.

may look as follows, $\mathcal{H}_{YZ}$ is an $m \times q \times n \times r$ tensor which acts on the tangent matrix Δ_Z of size $n \times r$ with the result $\langle \mathcal{H}_{YZ}, \Delta_Z \rangle_{3,4;1,2} = \langle \mathcal{H}_{ZY}, \Delta_Z \rangle_{1,2} \in \mathbf{T}_Y$. The equality in the last step follows from the fact that the $n \times r \times m \times q$ tensor $\mathcal{H}_{ZY}$ is a permutation of the $m \times q \times n \times r$ tensor $\mathcal{H}_{YZ}$. This is expected since for twice continuously differentiable functions $f_{xy} = f_{yx}$. But in our case they have different 'shapes'. The three tangent spaces $\mathbf{T}_X$, $\mathbf{T}_Y$ and $\mathbf{T}_Z$ are interconnected through the Hessian of $f(X, Y, Z)$ in the sense that every block in (6.8) is a linear operator mapping matrices from one tangent space to another tangent space. For example $\mathcal{H}_{YX} : \mathbf{T}_X \to \mathbf{T}_Y$ and $\mathcal{H}_{ZX} : \mathbf{T}_X \to \mathbf{T}_Z$.

The corresponding BFGS in the product manifold case has basically the same form as equations (6.6) and (6.7) where the action of the Hessian on S_k, which will be a triplet with an element on each tangent space, is replaced with formulas as in (6.8). Also the tensor/outer product needs to be modified in the obvious way, i.e. if $\Delta = (\Delta_X, \Delta_Y, \Delta_Z)$ and $\Gamma = (\Gamma_X, \Gamma_Y, \Gamma_Z)$ then we let

$$\begin{aligned} \Delta \hat{\otimes} \Gamma &= (\Delta_X, \Delta_Y, \Delta_Z) \hat{\otimes} (\Gamma_X, \Gamma_Y, \Gamma_Z) \\ &:= \begin{bmatrix} \Delta_X \otimes \Gamma_X & \Delta_X \otimes \Gamma_Y & \Delta_X \otimes \Gamma_Z \\ \Delta_Y \otimes \Gamma_X & \Delta_Y \otimes \Gamma_Y & \Delta_Y \otimes \Gamma_Z \\ \Delta_Z \otimes \Gamma_X & \Delta_Z \otimes \Gamma_Y & \Delta_Z \otimes \Gamma_Z \end{bmatrix}, \end{aligned} \tag{6.9}$$

where the results are conveniently stored in a 'block matrix' whose blocks are tensors of different dimensions (possibly nonconforming).

6.5. Optimality of BFGS on Grassmannians. The BFGS update in quasi-Newton methods is optimal because it is the solution to

$$\min_{H \in \mathbb{R}^{n \times n}} \|H - H_k\|_F \qquad \text{subject to} \qquad H = H^{\mathsf{T}}, \qquad H s_k = y_k,$$

where s_k and y_k are given by (4.3) and (4.4) respectively [46]. For the Euclidean case it is immaterial whether H is considered as an abstract operator or explicitly represented as a matrix. The final conclusion with respect to optimality is the same—it amounts to a rank-2 change of H_k. The situation is different when considering the corresponding optimality problem on Grassmannians. In particular, a given Hessian (or approximate Hessian) matrix H_k considered in a global coordinate representation and defined at $X_k \in \mathrm{Gr}(n, r)$ has the following form when parallel transported along a geodesic,

$$\bar{H}_k = (I_r \otimes T(t_k)) H_k (I_r \otimes \widetilde{T}(t_k)).$$

This is the same expression as equation (6.4). While the Hessian operator should not change by a parallel transport to a new point on the manifold, its representation evidently changes. This has important numerical and computational ramifications. In fact, the global coordinate representation of the Hessian at the previous point is usually very different from the global coordinate representation of the transported Hessian at the current point.

Assume now the Hessian matrix (or its approximation) is given in local coordinates $\widehat{H}_k$ at $X_k \in \mathrm{Gr}(n, r)$ and let $X_{k\perp}$ be the associated basis matrix for the tangent space. Representation of the parallel transported Hessian will not change if the associated basis matrix $X_{k\perp}(t)$ is transported according to (5.3). The updated Hessian at the current point is a rank-2 modification of the Hessian from the previous point given by the BFGS update. The optimality of BFGS update on Euclidean spaces is with respect to a change in successive Hessian matrices; we will prove that in the correct tangent space basis and in local coordinates, the BFGS update is also optimal on Grassmannians.

We now give a self contained proof for this statement. First we will state the optimality results for the Euclidean case. The proofs of Theorem 6.1, Lemma 6.2, and Theorem 6.3 are based on [21, 22]. We will then use these to deduce the corresponding optimality result on a product of Grassmannians in Theorem 6.6.

THEOREM 6.1. *Let $B \in \mathbb{R}^{n\times n}$, $y \in \mathbb{R}^n$, $0 \neq s \in \mathbb{R}^n$. The solution to*

$$\min\{\|A - B\|_F \mid A \in \mathbb{R}^{n\times n},\ As = y\}$$

is given by

$$\bar{B} = B + \frac{(y - Bs)s^{\mathsf{T}}}{s^{\mathsf{T}}s}.$$

Proof. Note that while the set $Q(y, s) := \{A \in \mathbb{R}^{n\times n} \mid As = y\}$ is non-compact (closed but unbounded), for a fixed B, the function $f : Q(y, s) \to \mathbb{R}$, $f(A) = \|A-B\|_F$ is coercive and therefore a minimizer $A_* \in Q(y, s)$ is attained. This demonstrates existence. The minimizer is also unique since $Q(y, s)$ is convex while f is strictly convex. We claim that $A_* = \bar{B}$: Observe that $\bar{B}s = y$ and so $\bar{B} \in Q(y, s)$; for any $A \in Q(y, s)$,

$$\|\bar{B} - B\|_F = \left\|\frac{ys^{\mathsf{T}}}{s^{\mathsf{T}}s} - \frac{Bss^{\mathsf{T}}}{s^{\mathsf{T}}s}\right\|_F = \left\|(A - B)\frac{ss^{\mathsf{T}}}{s^{\mathsf{T}}s}\right\|_F \leq \|A - B\|_F \left\|\frac{ss^{\mathsf{T}}}{s^{\mathsf{T}}s}\right\|_F = \|A - B\|_F.$$

□

LEMMA 6.2. *Let $y \in \mathbb{R}^n$, $0 \neq s \in \mathbb{R}^n$. Then the set $Q(y, s) = \{A \in \mathbb{R}^{n\times n} \mid As = y\}$ contains a symmetric positive definite matrix iff $y = Lv$ and $v = L^{\mathsf{T}}s$ for some $0 \neq v \in \mathbb{R}^n$ and $L \in \mathrm{GL}(n)$.*

Proof. If such v and L exist, then $y = Lv = LL^{\mathsf{T}}s$ and so LL^{T} is a symmetric positive definite matrix in $Q(y, s)$. On the other hand, if $A \in Q(y, s)$ is symmetric positive definite, its Cholesky factorization $A = LL^{\mathsf{T}}$ yields an $L \in \mathrm{GL}(n)$. If we let $v = L^{\mathsf{T}}s$, then $Lv = As = y$, as required. □

THEOREM 6.3. *Let $y \in \mathbb{R}^n$, $0 \neq s \in \mathbb{R}^n$. Let $L \in \mathrm{GL}(n)$ and $H = LL^{\mathsf{T}}$. There is a symmetric positive definite matrix $H_+ \in Q(y, s)$ iff $y^{\mathsf{T}}s > 0$. In this case, the* BFGS *update $H_+ = L_+L_+^{\mathsf{T}}$ is one where*

$$L_+ = L + \frac{(y - \alpha Hs)(L^{\mathsf{T}}s)^{\mathsf{T}}}{\alpha s^{\mathsf{T}}Hs} \qquad \text{with} \qquad \alpha = \pm\sqrt{\frac{y^{\mathsf{T}}s}{s^{\mathsf{T}}Hs}}. \tag{6.10}$$

Proof. In order for the update (6.10) to exist it is necessary that there exists $0 \neq v \in \mathbb{R}^n$ and $L_+ \in \mathrm{GL}(n)$ such that $y = L_+v$ and $v = L_+^{\mathsf{T}}s$. Hence

$$0 < v^{\mathsf{T}}v = (L_+^{\mathsf{T}}s)^{\mathsf{T}}(L_+^{-1}y) = s^{\mathsf{T}}y$$

as required.

If v is known, then the nearest matrix to L that takes v to y would be the update given in Theorem 6.1, i.e.

$$L_+ = L + \frac{(y - Lv)v^{\mathsf{T}}}{v^{\mathsf{T}}v}.$$

Hence we need to find the vector v. By Lemma 6.2,

$$v = L_+^\mathsf{T} s = L^\mathsf{T} s + \frac{y^\mathsf{T} s - v^\mathsf{T} L^\mathsf{T} s}{v^\mathsf{T} v} v \tag{6.11}$$

and so

$$v = \alpha L^\mathsf{T} s \tag{6.12}$$

for some $\alpha \in \mathbb{R}$. Now it remains to find the scalar α. Plugging (6.12) into (6.11) and using $H = LL^\mathsf{T}$, we get

$$\alpha = 1 + \frac{y^\mathsf{T} s - \alpha s^\mathsf{T} H s}{\alpha^2 s^\mathsf{T} H s} \cdot \alpha \quad \Rightarrow \quad \alpha^2 = \frac{y^\mathsf{T} s}{s^\mathsf{T} H s}.$$

If $y^\mathsf{T} s > 0$, this defines an update in $Q(y, s)$ that is symmetric positive definite. It is straightforward to verify that $H_+ = L_+ L_+^\mathsf{T}$ yields the BFGS update

$$H_+ = H - \frac{H s s^\mathsf{T} H}{s^\mathsf{T} H s} + \frac{y y^\mathsf{T}}{y^\mathsf{T} s}.$$

□

Theorem 6.4. *Let $X \in \mathrm{Gr}(n, r)$ and $X_\perp$ be the orthogonal complement to X, i.e. $[X\, X_\perp]$ is orthogonal. Let $\Delta \in \mathbf{T}_X$ and $X_\Delta(t)$ be a geodesic with $T_{X,\Delta}(t)$ the corresponding transport matrix, defined according to equations (5.1) and (5.2). Identify $\mathbf{T}_X$ with $\mathbb{R}^{(n-r)r}$ and consider a linear operator in local coordinates $\hat{A} : \mathbb{R}^{(n-r)r} \to \mathbb{R}^{(n-r)r}$. Consider the corresponding linear operator in global coordinates $A : \mathbb{R}^{nr} \to \mathbb{R}^{nr}$, in which tangents in $\mathbf{T}_X$ are embedded. The relation between the two operators is given by*

$$A = (I \otimes X_\perp) \hat{A} (I \otimes X_\perp^\mathsf{T}), \tag{6.13}$$

$$\hat{A} = (I \otimes X_\perp^\mathsf{T}) A (I \otimes X_\perp). \tag{6.14}$$

Furthermore, the parallel transported operator $\hat{A}$ has the same representation for all t along the geodesic $X(t)$, i.e. $\hat{A}(t) \equiv \hat{A}$.

Proof. Let $d_1 \in \mathbb{R}^{(n-r)r}$ be a tangent vector with corresponding global coordinate matrix representation $\Delta_1 = X_\perp D_1 \in \mathbf{T}_X$. Obviously $d_1 = \mathrm{vec}(D_1)$. We may write $\mathrm{vec}(\Delta_1) = (I \otimes X_\perp) d_1$. Set $d_2 = \hat{A} d_1$ and it follows that $\mathrm{vec}(\Delta_2) = (I \otimes X_\perp) d_2$. The corresponding operation in global coordinates are

$$\begin{aligned} \mathrm{vec}(\Delta_2) = A\, \mathrm{vec}(\Delta_1) \quad &\Leftrightarrow \quad (I \otimes X_\perp) d_2 = A (I \otimes X_\perp) d_1 \\ &\Leftrightarrow \quad d_2 = (I \otimes X_\perp^\mathsf{T}) A (I \otimes X_\perp) d_1 \end{aligned}$$

and it follows that $\hat{A} = (I \otimes X_\perp^\mathsf{T}) A (I \otimes X_\perp)$, which proves (6.14).

For any tangent $\Delta_* \in \mathbf{T}_X$ it holds that $\Delta_* = \Pi_X \Delta_* = X_\perp X_\perp^\mathsf{T} \Delta_*$, where Π_X is a projection onto $\mathbf{T}_X$, and consequently $\mathrm{vec}(\Delta_*) = (I \otimes X_\perp X_\perp^\mathsf{T})\, \mathrm{vec}(\Delta_*)$. Thus the operations in global coordinates also satisfy

$$\begin{aligned} \mathrm{vec}(\Delta_2) &= (I \otimes X_\perp X_\perp^\mathsf{T}) A (I \otimes X_\perp X_\perp^\mathsf{T})\, \mathrm{vec}(\Delta_1) \\ &= (I \otimes X_\perp) \hat{A} (I \otimes X_\perp^\mathsf{T})\, \mathrm{vec}(\Delta_1) \\ &\equiv A\, \mathrm{vec}(\Delta_1). \end{aligned}$$

This proves equation (6.13).

For the third part we have $A : \mathbf{T}_X \to \mathbf{T}_X$ and $A(t) : \mathbf{T}_{X(t)} \to \mathbf{T}_{X(t)}$ with $A(0) \equiv A$. We want to prove that $\hat{A}(t) \equiv \hat{A}$ for all t. The operator $A(t)$ is defined in the following sense: a tangent $\Delta_1(t) \in \mathbf{T}_{X(t)}$ is parallel transported with $\widetilde{T}_{X(t),-\Delta(t)}(t)$ to $X(0)$ along $X(t)$, the operator transformations is performed in $\mathbf{T}_X$, thus $\Delta_2 = A(\Delta_1(0)) \in \mathbf{T}_X$ and the result is forwarded to $\mathbf{T}_{X(t)}$, i.e. $\Delta_2(t) = T_{X,\Delta}(t)\Delta_2$. The parallel transported operator in global coordinates takes the form

$$A(t) = (I \otimes T_{X,\Delta}(t))A(I \otimes \widetilde{T}_{X(t),-\Delta(t)}(t)). \tag{6.15}$$

Then, in the basis $X_\perp(t)$, the local coordinate representation of the operator is

$$\hat{A}(t) = (I \otimes X_\perp^\top(t))A(t)(I \otimes X_\perp(t)). \tag{6.16}$$

Substituting (6.15) into (6.16), we obtain

$$\hat{A}(t) = (I \otimes X_\perp^\top(t)T_{X,\Delta}(t)X_\perp)\hat{A}(I \otimes X_\perp^\top \widetilde{T}_{X(t),-\Delta(t)}(t)X_\perp(t)).$$

Recall that $T_{X,\Delta}(t)X_\perp = X_\perp(t)$ and thus $X_\perp^\top(t)T_{X,\Delta}(t)X_\perp = I$. Similarly one can show that $X_\perp^\top \widetilde{T}_{X(t),-\Delta(t)}(t)X_\perp(t) = I$ and we get $\hat{A}(t) = \hat{A}$ for all t. □

A different proof of essentially the same statement may be found in [52].

LEMMA 6.5. *Let $X_i \in \mathrm{Gr}(n_i, r_i)$, $i = 1, \dots, k$, with corresponding tangent spaces $\mathbf{T}_{X_i}$. Let Y_i be given such that $[X_i\ Y_i]$ is orthogonal. On each Grassmannian $\mathrm{Gr}(n_i, r_i)$, let $X_i(t)$ be a geodesic and $Y_i(t)$ be its orthogonal complement corresponding to the tangent $\Delta_i \in \mathbf{T}_{X_i}$. Then a local coordinate representation of the linear operator*

$$\hat{A} : \mathbf{T}_{X_1} \times \cdots \times \mathbf{T}_{X_k} \to \mathbf{T}_{X_1} \times \cdots \times \mathbf{T}_{X_k}$$

is independent of t when parallel transported along the geodesics $X_i(t)$ and in the tangent basis $Y_i(t)$.

Proof. First we observe that the operator $\hat{A}$ must necessarily have the structure

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \cdots & \hat{A}_{1k} \\ \vdots & \ddots & \vdots \\ \hat{A}_{k1} & \cdots & \hat{A}_{kk} \end{bmatrix}$$

where each $\hat{A}_{ij}$, $1 \le i, j \le k$ is such that $\hat{A}_{ij} : \mathbf{T}_{X_j} \to \mathbf{T}_{X_i}$. Now, applying a similar procedure as in Theorem 6.4 on each block $\hat{A}_{ij}$ proves that local coordinate representation of $\hat{A}_{ij}$ and thus $\hat{A}$ is independent of t along the geodesics $X_i(t)$ in the tangent space basis $Y_i(t)$. □

Now we will give an explicit expression for the general BFGS update in tensor form and in local coordinates. We omit the hat and iteration index below for clarity. For a function defined on a product of k Grassmannians $f : \mathrm{Gr}(n_1, r_1) \times \cdots \times \mathrm{Gr}(n_k, r_k) \to \mathbb{R}$, we write $f(X_1, \dots, X_k)$ and $S = (S_1, \dots, S_k)$, $Y = (Y_1, \dots, Y_k)$ where $X_i \in \mathrm{Gr}(n_i, r_i)$ and $S_i, Y_i \in \mathbf{T}_{X_i}$ for $i = 1, \dots, k$. The Hessian or its approximation has the symbolic form

$$\mathcal{H} = \begin{bmatrix} \mathcal{H}_{11} & \cdots & \mathcal{H}_{1k} \\ \vdots & \ddots & \vdots \\ \mathcal{H}_{k1} & \cdots & \mathcal{H}_{kk} \end{bmatrix}$$

where each block is a 4-tensor. The BFGS update takes the form,

$$\mathcal{H}_+ = \mathcal{H} + \frac{\langle \mathcal{H}, S\rangle_{1,2} \mathbin{\hat{\otimes}} \langle \mathcal{H}, S\rangle_{1,2}}{\langle \langle \mathcal{H}, S\rangle_{1,2}, S\rangle} + \frac{Y \mathbin{\hat{\otimes}} Y}{\langle S, Y\rangle}, \tag{6.17}$$

where the $\langle \mathcal{H}, S\rangle_{1,2}$ is given by a formula similar to (6.8) with the result being a k-tuple, the tensor product between k-tuples of tangents is an obvious generalization of (6.9), and of course $\langle S, Y\rangle = \sum_{i=1}^{d} \langle S_i, Y_i\rangle$.

Finally, we have all the ingredients required to prove the optimality of the BFGS update on a product of Grassmannians.

THEOREM 6.6 (Optimality of BFGS update on product of Grassmannians). *Consider a function $f(X_1, \dots, X_k)$ in the variables $X_i \in \operatorname{Gr}(n_i, r_i)$, $i = 1, \dots, k$, that we want to minimize. Let $X_i(t)$ be geodesic defined by $\Delta_i \in \mathbf{T}_{X_i}$ with the corresponding tangent space basis matrices $Y_i(t)$. In these basis for the tangent spaces, the BFGS updates in (6.17) on the product Grassmannians have the same optimality properties as a function with variables in a Euclidean space, i.e. it is the least change update of the current Hessian approximation that satisfies the secant equations.*

Proof. First we observe that the Grassmann Hessian of $f(X_1, \dots, X_k)$ (or its approximation) is a linear operator

$$H_f : \mathbf{T}_{X_1} \times \cdots \times \mathbf{T}_{X_k} \to \mathbf{T}_{X_1} \times \cdots \times \mathbf{T}_{X_k}$$

and according to Lemma 6.5 its local coordinate representation is constant along the geodesics $X_i(t)$. Given this, the BFGS optimality result on product Grassmannians is a consequence from Theorem 6.3—the optimality of BFGS in Euclidean space. □

Remark. An important difference on (product) Grassmannians is that we need to keep track of the basis for the tangent spaces—$Y_i(t)$ from equation (5.3). Only then will the local coordinate representation of an operator be independent of t when transported along geodesics.

Note that Theorem 6.6 is a coordinate dependent result. If we regard Hessians as abstract operators, there will no longer be any difference between the global and the local scenario. But the corresponding optimality as the least amount of change in successive Hessians cannot be obtained in global coordinate representation and is thus not true if the Hessians are regarded as abstract operators.

6.6. Other alternatives. Movement along geodesics and parallel transport of tangents are the most straightforward and natural generalizations to the key operations from Euclidean spaces to manifolds. There are also methods for dealing with the manifold structure in optimization algorithms based on different principles. For example, instead of moving along geodesics from one point to another on the manifold one could use the notion of *retractions*, which is a smooth mapping from the tangent bundle of the manifold onto the manifold. Another example is the notion of *vector transport* that generalizes the parallel translation/transport of tangents used in this paper. All these notions are defined and described in [2]. It is not clear how the use of the more general vector transport would effect the convergence properties of the resulting BFGS methods.

7. Limited memory BFGS. We give a brief summary of the limited memory quasi-Newton method with L-BFGS updates on Euclidean spaces [6] that we need later for our Grassmann variant. See also the discussion in [52, Chapter 7]. In Euclidean space the BFGS update can be represented in the following compact form

$$H_k = H_0 + \begin{bmatrix} S_k & H_0 Y_k \end{bmatrix} \begin{bmatrix} R_k^{-\mathsf{T}}(D_k + Y_k^{\mathsf{T}} H_0 Y_k) R_k^{-1} & -R_k^{-\mathsf{T}} \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^{\mathsf{T}} \\ Y_k^{\mathsf{T}} H_0 \end{bmatrix}, \tag{7.1}$$

where $S_k = [s_0, \ldots, s_{k-1}]$, $Y_k = [y_0, \ldots, y_{k-1}]$, $D_k = \operatorname{diag}\left[s_0^{\mathsf{T}} y_0, \ldots, s_{k-1}^{\mathsf{T}} y_{k-1}\right]$ and

$$R_k = \begin{bmatrix} s_0^{\mathsf{T}} y_0 & s_0^{\mathsf{T}} y_1 & \cdots & s_0^{\mathsf{T}} y_{k-1} \\ 0 & s_1^{\mathsf{T}} y_1 & \cdots & s_1^{\mathsf{T}} y_{k-1} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & s_{k-1}^{\mathsf{T}} y_{k-1} \end{bmatrix},$$

are obtained using equations (4.3) and (4.4). Observe that in this section S_k and Y_k are not the same as in (6.2) and (6.3) respectively. The limited memory version of the algorithm is obtained when replacing the initial Hessian H_0 by a sparse matrix, usually this is a suitably scaled identity matrix $\gamma_k I$, and only keep the m most resent s_j and y_j in the update (7.1). Since $m \ll n$ the amount of storage and computations in each iteration is only a small fraction compared to the regular BFGS. According to [46] satisfactory results are often achieved with $5 \le m \le 20$, even for large problems. Our experiments confirm this heuristic. Thus for the limited memory BFGS we have

$$H_k = \gamma_k I + \begin{bmatrix} S_k & \gamma_k Y_k \end{bmatrix} \begin{bmatrix} R_k^{-\mathsf{T}}(D_k + \gamma_k Y_k^{\mathsf{T}} Y_k) R_k^{-1} & -R_k^{-\mathsf{T}} \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^{\mathsf{T}} \\ \gamma_k Y_k^{\mathsf{T}} \end{bmatrix}, \tag{7.2}$$

where now

$$S_k = [s_{k-m}, \ldots, s_{k-1}], \quad Y_k = [y_{k-m}, \ldots, y_{k-1}], \quad D_k = \operatorname{diag}\left[s_{k-m}^{\mathsf{T}} y_{k-m}, \ldots, s_{k-1}^{\mathsf{T}} y_{k-1}\right]$$

and

$$R_k = \begin{bmatrix} s_{k-m}^{\mathsf{T}} y_{k-m} & s_{k-m}^{\mathsf{T}} y_{k-m+1} & \cdots & s_{k-m}^{\mathsf{T}} y_{k-1} \\ 0 & s_{k-m+1}^{\mathsf{T}} y_{k-m+1} & \cdots & s_{k-m+1}^{\mathsf{T}} y_{k-1} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & s_{k-1}^{\mathsf{T}} y_{k-1} \end{bmatrix}.$$

7.1. Limited memory BFGS on Grassmannians. Analyzing the L-BFGS update above with the intent of modifying it to be applicable on Grassmannians, we observe the following:

1. The columns in the matrices S_k and Y_k represent tangents, and as such, they are defined on a specific point of the manifold. In each iteration we need to parallel transport these vectors to the next tangent space. Assuming s_k and y_k are vectorized forms of (6.2) and (6.3) the transport amounts to computing $\bar{S}_k = (I_r \otimes T(t_k))S_k$ and $\bar{Y}_k = (I_r \otimes T(t_k))Y_k$ where $T(t_k)$ is the Grassmann transport matrix.
2. The matrices R_k and D_k contain inner products between tangents. Fortunately, the inner products are invariant with respect to parallel transporting. Given vectors $u, v \in \mathbf{T}_{X_k}$ and a transport matrix T from $\mathbf{T}_{X_k}$ to $\mathbf{T}_{X_{k+1}}$, i.e. $Tu, Tv \in \mathbf{T}_{X_{k+1}}$, we have that $\langle Tu, Tv \rangle = \langle u, v \rangle$. This is a direct result from Theorem 5.1, showing that there is no need for modifying R_k or D_k. Because of this property one may wonder whether the transport matrix T is orthogonal, but this is not the case, $T^{\mathsf{T}} T \neq I$.
3. Recalling the relation from equation (6.13) between local and global coordinate representation of an operator, we conclude that the global representation is necessarily a singular matrix, simply because the local coordinate representation of the operator is a smaller matrix. The same is true for the Hessian

> using global coordinates. But by construction, the L-BFGS update H_k in (7.2) is positive definite and thus nonsingular. This causes no problem since $\mathbf{T}_{X_k}$ is an invariant subspace of H_k, i.e. if $v \in \mathbf{T}_{X_k}$ then $H_k v \in \mathbf{T}_{X_k}$, see Lemma 7.1. Similarly for the solution of the (quasi-)Newton equations (4.1) since $y_k \in \mathbf{T}_{X_k}$ and $H_k : \mathbf{T}_{X_k} \to \mathbf{T}_{X_k}$, then obviously $p_k \in \mathbf{T}_{X_k}$. This is valid for H_k from both (7.1) and (7.2).

LEMMA 7.1. *The tangent space $\mathbf{T}_{X_k}$ is an invariant subspace of the operator obtained by the* L-BFGS *update.*

Proof. This is straightforward. Simply observe that for a vector $v_k \in \mathbf{T}_{X_k}$ we have that $H_k v_k$ is a linear combination of vectors, and all of them belong to $\mathbf{T}_{X_k}$. □

L-BFGS algorithms are intended for large scale problems where the storage of the full Hessian may not be possible. With this in mind we realize that the computation and storage of the orthogonal complement $X_\perp$, which is used in local coordinate implementations, may not be practical. For large and sparse problems it is more economical to do the parallel transports explicitly than to update a basis for the tangent space. The computational time is reasonable since only $2(m-1)$ vectors are parallel transported each step and m is usually very small compared to the dimensions of the Hessian.

8. Quasi-Newton methods for the best multilinear rank approximation of a tensor.

In this section we apply the algorithms developed in the last three sections to the tensor approximation problem described earlier. Recall from Section 3.1 that the best multilinear rank-(p,q,r) approximation of a general tensor is equivalent to the maximization of

$$\Phi(X,Y,Z) = \frac{1}{2}\|\mathcal{A}\cdot(X,Y,Z)\|_F^2 \quad \text{s.t.} \quad X^\mathsf{T}X = I, \quad Y^\mathsf{T}Y = I, \quad Z^\mathsf{T}Z = I,$$

where $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$ and $X \in \mathbb{R}^{l\times p}$, $Y \in \mathbb{R}^{m\times q}$, $Z \in \mathbb{R}^{n\times r}$. Recall also that X, Y, Z may be regarded as elements of $\mathrm{Gr}(l,p)$, $\mathrm{Gr}(m,q)$, and $\mathrm{Gr}(n,r)$ respectively and Φ may be regarded as a function defined on a product of the three Grassmannians. The Grassmann gradient of Φ will consist of three parts. Setting $\mathcal{F} = \mathcal{A}\cdot(X,Y,Z)$, one can show that in global coordinates the gradient is the triplet $\nabla\Phi = (\Pi_X\Phi_x, \Pi_Y\Phi_y, \Pi_Z\Phi_z)$, where

$$\Pi_X\Phi_x = \langle\mathcal{A}\cdot(\Pi_X,Y,Z),\mathcal{F}\rangle_{-1} \quad \in \mathbb{R}^{l\times p}, \qquad \Pi_X = I - XX^\mathsf{T}, \tag{8.1}$$

$$\Pi_Y\Phi_y = \langle\mathcal{A}\cdot(X,\Pi_Y,Z),\mathcal{F}\rangle_{-2} \quad \in \mathbb{R}^{m\times q}, \qquad \Pi_Y = I - YY^\mathsf{T}, \tag{8.2}$$

$$\Pi_Z\Phi_z = \langle\mathcal{A}\cdot(X,Y,\Pi_Z),\mathcal{F}\rangle_{-3} \quad \in \mathbb{R}^{n\times r}, \qquad \Pi_Z = I - ZZ^\mathsf{T}, \tag{8.3}$$

and $\Phi_x = \partial\Phi/\partial X$, $\Phi_y = \partial\Phi/\partial Y$ and $\Phi_z = \partial\Phi/\partial Z$, see equation (6.1). For derivation of these formulas[3] see [25].

To obtain the corresponding expressions in local coordinates we observe that a projection matrix can also be written as $\Pi_X = X_\perp X_\perp^\mathsf{T}$. Then for tangent vectors $\Delta_x \in \mathbf{T}_X$, we have

$$\Delta_x = \Pi_X\Delta_x = X_\perp X_\perp^\mathsf{T}\Delta_x \equiv X_\perp D_x,$$

which gives the local coordinates of Δ_x as $X_\perp^\mathsf{T}\Delta_x = D_x$. The practical implication of these manipulations is that in local coordinates we simply replace the projection matrices Π_X, Π_Y, Π_Z with $X_\perp^\mathsf{T}, Y_\perp^\mathsf{T}, Z_\perp^\mathsf{T}$. We get $\nabla\widehat{\Phi} = (X_\perp^\mathsf{T}\Phi_x, Y_\perp^\mathsf{T}\Phi_y, Z_\perp^\mathsf{T}\Phi_z)$, where

$$X_\perp^\mathsf{T}\Phi_x = \langle \mathcal{A} \cdot (X_\perp, Y, Z), \mathcal{F}\rangle_{-1} \quad \in \mathbb{R}^{(l-p)\times p}, \tag{8.4}$$

$$Y_\perp^\mathsf{T}\Phi_y = \langle \mathcal{A} \cdot (X, Y_\perp, Z), \mathcal{F}\rangle_{-2} \quad \in \mathbb{R}^{(m-q)\times q}, \tag{8.5}$$

$$Z_\perp^\mathsf{T}\Phi_z = \langle \mathcal{A} \cdot (X, Y, Z_\perp), \mathcal{F}\rangle_{-3} \quad \in \mathbb{R}^{(n-r)\times r}. \tag{8.6}$$

Note that the expressions of the gradient in global and local coordinates are different. In order to distinguish between them we put a hat on the gradient, i.e. $\nabla\widehat{\Phi}$, when it is expressed in local coordinates.

8.1. General expression for Grassmann gradients and Hessians. In the general case we will have an order-k tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_k}$ and the objective function takes the form

$$\Phi(X_1, \ldots, X_k) = \frac{1}{2}\|\mathcal{A} \cdot (X_1, \ldots, X_k)\|_F^2.$$

The low rank approximation problem becomes

$$\max \Phi(X_1, \ldots, X_k) \quad \text{s.t.} \quad X_i^\mathsf{T} X_i = I, \quad i = 1, \ldots, k.$$

The same procedure used to derive the gradients for the order-3 case can be used for the general case. The results are obvious modifications of what we have for 3-tensors. First we introduce matrices $X_{i\perp}$, $i = 1, \ldots, k$, such that each $[X_i \; X_{i\perp}]$ forms an orthogonal matrix and we define the tensors

$$\begin{aligned}
\mathcal{F} &= \mathcal{A} \cdot (X_1, X_2, X_3, \ldots, X_k),\\
\mathcal{B}_1 &= \mathcal{A} \cdot (X_{1\perp}, X_2, X_3, \ldots, X_k),\\
\mathcal{B}_2 &= \mathcal{A} \cdot (X_1, X_{2\perp}, X_3, \ldots, X_k),\\
&\vdots\\
\mathcal{B}_{k-1} &= \mathcal{A} \cdot (X_1, \ldots, X_{k-2}, X_{k-1,\perp}, X_k),\\
\mathcal{B}_k &= \mathcal{A} \cdot (X_1, \ldots, X_{k-2}, X_{k-1}, X_{k\perp}).
\end{aligned} \tag{8.7}$$

The Grassmann gradient of the objective function in local coordinates is given by the k-tuple

$$\nabla\widehat{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_k), \qquad \Phi_i = \langle \mathcal{B}_i, \mathcal{F}\rangle_{-i}, \quad i = 1, 2, \ldots, k.$$

Each Φ_i is an $(n_i - r_i) \times r_i$ matrix representing a tangent in $\mathbf{T}_{X_i}$. To obtain the corresponding global coordinate representation, simply replace each $X_{i\perp}$ with the projection $\Pi_{X_i} = I - X_i X_i^\mathsf{T}$.

We will also give the expression of the Hessian since we may wish to initialize our approximate Hessian with the exact Hessian. Furthermore, in our numerical experiments in Section 11, the expression for the Hessian will be useful for checking whether our algorithms have indeed arrived at a local maximum. In order to express the Hessian, we will need to introduce the additional variables

$$\begin{matrix}
\mathcal{C}_{12} & & & \\
\mathcal{C}_{13} & \mathcal{C}_{23} & & \\
\vdots & \vdots & \ddots & \\
\mathcal{C}_{1,k} & \mathcal{C}_{2,k} & \cdots & \mathcal{C}_{k-1,k}
\end{matrix} \tag{8.8}$$

where each term is a multilinear tensor-matrix product involving the tensor $\mathcal{A}$ and a subset of the matrices in $\{X_1, \dots, X_k, X_{1\perp}, \dots, X_{k\perp}\}$. The subscripts i and j in $\mathcal{C}_{ij}$ indicate that $X_{i\perp}$ and $X_{j\perp}$ are multiplied in the ith and jth mode of $\mathcal{A}$, respectively. All other modes are multiplied with the corresponding X_d, $d \neq i$ and $d \neq j$. For example we have

$$\begin{aligned}
\mathcal{C}_{12} &= \mathcal{A} \cdot (X_{1\perp}, X_{2\perp}, X_3, \dots, X_k),\\
\mathcal{C}_{24} &= \mathcal{A} \cdot (X_1, X_{2\perp}, X_3, X_{4\perp}, X_5, \dots, X_k),\\
\mathcal{C}_{k-1,k} &= \mathcal{A} \cdot (X_1, \dots, X_{k-2}, X_{k-1,\perp}, X_{k\perp}).
\end{aligned}$$

Together with $\mathcal{B}_1, \dots, \mathcal{B}_k$, introduced earlier, one can express the complete Grassmann Hessian of the objective function $\Phi(X_1, \dots, X_k)$. The derivation of the Hessian is somewhat tricky. The interested reader should refer to [25] for details. In this paper we only state the final result in a form that can be directly implemented in a solver.

The diagonal blocks of the Hessian are Sylvester operators and have the form

$$\mathcal{H}_{ii}(D_i) = \langle \mathcal{B}_i, \mathcal{B}_i \rangle_{-i} D_i - D_i \langle \mathcal{F}, \mathcal{F} \rangle_{-i}, \quad i = 1, 2, \dots, k.$$

The off-diagonal block operators are

$$\begin{aligned}
\mathcal{H}_{12}(D_2) &= \langle \langle \mathcal{C}_{12}, \mathcal{F} \rangle_{-(1,2)}, D_2 \rangle_{2,4;1,2} + \langle \langle \mathcal{B}_1, \mathcal{B}_2 \rangle_{-(1,2)}, D_2 \rangle_{4,2;1,2},\\
&\vdots\\
\mathcal{H}_{ij}(D_j) &= \langle \langle \mathcal{C}_{ij}, \mathcal{F} \rangle_{-(i,j)}, D_j \rangle_{2,4;1,2} + \langle \langle \mathcal{B}_i, \mathcal{B}_j \rangle_{-(i,j)}, D_j \rangle_{4,2;1,2},
\end{aligned}$$

where $i \neq j$, $i < j$ and $i, j = 1, 2, \dots, k$. See Appendix A for definition of the contracted products $\langle \cdot, \cdot \rangle_{-(i,j)}$.

9. Best multilinear rank approximation of a symmetric tensor. Recall from Section 2 that an order-k tensor $\mathcal{S} \in \mathbb{R}^{n \times \cdots \times n}$ is called *symmetric* if

$$s_{i_{\sigma(1)} \cdots i_{\sigma(k)}} = s_{i_1 \cdots i_k}, \qquad i_1, \dots, i_k \in \{1, \dots, n\},$$

where $\sigma \in \mathfrak{S}_k$, the set of all permutations with k integers. For example, a third order cubical tensor $\mathcal{S} \in \mathbb{R}^{n \times n \times n}$ is symmetric iff

$$s_{ijk} = s_{ikj} = s_{jik} = s_{jki} = s_{kij} = s_{kji}$$

for all $i, j, k \in \{1, \dots, n\}$. The definition given above is equivalent to the usual definition given in, say [30]; see [8] for a proof of this simple equivalence. Recall also that the set of all order-k dimension-n symmetric tensors is denoted $\mathsf{S}^k(\mathbb{R}^n)$. This is a subspace of $\mathbb{R}^{n \times \cdots \times n}$ and

$$\dim \mathsf{S}^k(\mathbb{R}^n) = \binom{n+k-1}{k}.$$

LEMMA 9.1. *If $\mathcal{S} \in \mathsf{S}^k(\mathbb{C}^n)$ and $\operatorname{rank}(\mathcal{S}) = (r_1, \dots, r_k)$, then*

$$r_1 = \cdots = r_k.$$

In other words, the multilinear rank of a symmetric tensor is always of the form $(r, \dots, r)$ for some r. We will write $r_{\mathsf{S}}(\mathcal{S})$ for this common value. Furthermore, we have a multilinear decomposition of the following form

$$\mathcal{S} = (X, X, \dots, X) \cdot \mathcal{C}, \tag{9.1}$$

where $\mathcal{C} \in \mathsf{S}^k(\mathbb{R}^r)$ *and* $X \in \mathrm{O}(n,r)$.

Proof. The ranks r_i being equal follows from observing that the matricizations $S^{(1)}, \dots, S^{(k)}$ of $\mathcal{S}$ are, due to symmetry, all equal. The factorization (9.1) is a consequence of the higher order singular value decomposition (HOSVD) [15]. □

In application where noise is an inevitable factor, we would like to study instead the approximation problem

$$\mathcal{S} \approx (X, X, \dots, X) \cdot \mathcal{C},$$

instead of the exact decomposition in (9.1). More precisely, we want to solve

$$\min\{\|\mathcal{S} - \mathcal{T}\|_F \mid \mathcal{T} \in \mathsf{S}^k(\mathbb{R}^n),\ r_{\mathsf{S}}(\mathcal{T}) \leq r\}. \tag{9.2}$$

Similar analysis as in the general case shows that the minimization problem (9.2) can be reformulated as a maximization of $\|\mathcal{S} \cdot (X, \dots, X)\|_F$, with the constraint $X^{\mathsf{T}}X = I$. The objective function becomes $\Phi(X) = \frac{1}{2}\langle \mathcal{F}, \mathcal{F}\rangle$ where now $\mathcal{F} = \mathcal{S} \cdot (X, \dots, X)$. Observe that the symmetric tensor approximation problem is defined on one Grassmannian only, regardless of the order of the tensor. These problems require much less storage and computations compared to a general problem of the same dimensions. Applications involving symmetric tensors are found in signal processing, independent component analysis, and the analysis of multivariate cumulants in statistics [10, 8, 37, 14, 18, 19, 9, 44, 41]. We refer interested readers to [8] for discussion of a different notion of rank for symmetric tensors.

9.1. The symmetric Grassmann gradient. The same procedure for deriving the gradient for the general case can be used to obtain the gradient for the symmetric case. In particular it involves the very same terms as the nonsymmetric gradient with obvious modifications. It is straightforward to show that, due to symmetry of $\mathcal{S}$,

$$\langle \mathcal{S} \cdot (\Pi_X, X, X), \mathcal{F}\rangle_{-1} = \langle \mathcal{S} \cdot (X, \Pi_X, X), \mathcal{F}\rangle_{-2} = \langle \mathcal{S} \cdot (X, X, \Pi_X), \mathcal{F}\rangle_{-3}.$$

We will use the first expression without loss of generality. In which case, the Grassmann gradient in global coordinates becomes

$$\nabla\Phi = \Pi_X \Phi_x = 3\langle \mathcal{S} \cdot (\Pi_X, X, X), \mathcal{F}\rangle_{-1}, \tag{9.3}$$

where $\Pi_X = I - XX^{\mathsf{T}}$; and in local coordinate it is

$$\nabla\widehat{\Phi} = X_\perp \Phi_x = 3\langle \mathcal{S} \cdot (X_\perp, X, X), \mathcal{F}\rangle_{-1}, \tag{9.4}$$

where $X_\perp$ is the orthogonal complement of X. Compare these with equations (8.1)–(8.3) for the general case.

9.2. The symmetric Grassmann Hessian. As for the general case discussed in [25], we may identify the second order terms in the Taylor expansion of $\Phi(X_\Delta(t))$. There are 15 second order terms and all have the form

$$\langle \Delta, \mathcal{H}_*(\Delta)\rangle, \quad \Delta \in \mathbf{T}_X \text{ and } X \in \mathrm{Gr}(n,r),$$

for some linear operator $\mathcal{H}_*$. Two specific examples are

$$\begin{aligned}
\langle \Delta, \mathcal{H}_{11}(\Delta)\rangle &= \langle \Delta, \langle \mathcal{B}_1, \mathcal{B}_1\rangle_{-1}\Delta - \Delta\langle \mathcal{F}, \mathcal{F}\rangle_{-1}\rangle\,, \\
\langle \Delta, \mathcal{H}_{12}(\Delta)\rangle &= \big\langle \Delta, \langle\langle \mathcal{C}_{12}, \mathcal{F}\rangle_{-(1,2)}, \Delta\rangle_{2,4;1,2} + \langle\langle \mathcal{B}_1, \mathcal{B}_2\rangle_{-(1,2)}, \Delta\rangle_{4,2;1,2}\big\rangle\,,
\end{aligned}$$

where $\mathcal{B}_1 = \mathcal{S} \cdot (\Pi_X, X, X)$, $\mathcal{B}_2 = \mathcal{S} \cdot (X, \Pi_X, X)$ and $\mathcal{C}_{12} = \mathcal{S} \cdot (\Pi_X, \Pi_X, X)$. The subscripts 1 and 2 indicate that the projection matrix Π_X is multiplied with $\mathcal{S}$ in the first and second mode respectively. Not surprisingly, analysis of these terms reveals equality among the second order terms due to the symmetry of $\mathcal{S}$. Gathering like terms and summing up the expressions, we see that the Hessian is a sum of three different terms,

$$\begin{aligned}
\langle \Delta, \mathcal{H}_1(\Delta) \rangle =& \langle \Delta, 3\langle \mathcal{B}_1, \mathcal{B}_1 \rangle_{-1} \Delta - 3\Pi_X \Delta \langle \mathcal{F}, \mathcal{F} \rangle_{-1} \rangle, && (9.5)\\
\langle \Delta, \mathcal{H}_2(\Delta) \rangle =& \langle \Delta, 6\langle \langle \mathcal{C}_{12}, \mathcal{F} \rangle_{-(1,2)}, \Delta \rangle_{2,4;1,2} \rangle, && (9.6)\\
\langle \Delta, \mathcal{H}_3(\Delta) \rangle =& \langle \Delta, 6\langle \langle \mathcal{B}_1, \mathcal{B}_2 \rangle_{-(1,2)}, \Delta \rangle_{4,2;1,2} \rangle. && (9.7)
\end{aligned}$$

So the action of the Hessian on a tangent is simply

$$\mathcal{H}(\Delta) = \mathcal{H}_1(\Delta) + \mathcal{H}_2(\Delta) + \mathcal{H}_3(\Delta).$$

Observe that the second term in (9.5) arises from the fact that the objective function is defined on a Grassmannian, see [24] for details.

9.3. General expression for Grassmann gradients and Hessians for a symmetric tensor. With the analysis and expressions for symmetric 3-tensors at hand, generalization to symmetric k-tensors is straightforward. We will only state the final results and in local coordinates. Assume we have an order-k symmetric tensor $\mathcal{S} \in \mathsf{S}^k(\mathbb{R}^n)$. The corresponding symmetric low rank tensor approximation problem is written as

$$\max \Phi(X) = \max \frac{1}{2} \|\mathcal{S} \cdot (X, \ldots, X)\|_F^2 \quad \text{s.t.} \quad X \in \mathrm{Gr}(n, r).$$

Using the tensor products

$$\begin{aligned}
\mathcal{F} &= \mathcal{S} \cdot (X, X, \ldots, X) && X \text{ appears } k \text{ times},\\
\mathcal{B}_1 &= \mathcal{S} \cdot (X_\perp, X, \ldots, X) && X \text{ appears } k-1 \text{ times},
\end{aligned}$$

where $X_\perp$ is such that $[X \ X_\perp]$ forms an orthogonal matrix, the Grassmann gradient becomes

$$\nabla \Phi = k \langle \mathcal{B}_1, F \rangle_{-1}.$$

Observe that the symmetric case involves the very same tensor products $\mathcal{B}_i$ as in the general case (given in Section 8.1) but due to the symmetry of the problem all terms are equal.

We also introduce tensor-matrix multilinear products $\mathcal{C}_{ij}$ similar to those in equation (8.8). Two specific examples are

$$\begin{aligned}
\mathcal{C}_{12} &= \mathcal{S} \cdot (X_\perp, X_\perp, X, \ldots, X),\\
\mathcal{C}_{24} &= \mathcal{S} \cdot (X, X_\perp, X, X_\perp, X, \ldots, X).
\end{aligned}$$

In general $\mathcal{C}_{ij}$, where $i \neq j$, $i < j$ and $i, j = 1, \ldots, k$, is a multilinear product of two $X_\perp$'s that are multiplied in the ith and jth mode of $\mathcal{S}$. All other modes are multiplied with X.

The second order terms of the Taylor expansion of $\Phi(X)$ contain the following diagonal block operators

$$H_{ii}(D) = \langle \mathcal{B}_i, \mathcal{B}_i \rangle_{-i} D - \Delta \langle \mathcal{F}, \mathcal{F} \rangle_{-i}, \quad i = 1, 2, \ldots, k.$$

Again, due to symmetry all these are identical and summing them up we get

$$H_{\text{diag}}(D) = k\left(\langle \mathcal{B}_1, \mathcal{B}_1\rangle_{-1} D - D\langle \mathcal{F}, \mathcal{F}\rangle_{-1}\right).$$

The off-diagonal block operators have the form

$$\begin{aligned}
\mathcal{H}_{12}(D) &= \langle \langle \mathcal{C}_{12}, \mathcal{F}\rangle_{-(1,2)}, D\rangle_{2,4;1,2} + \langle \langle \mathcal{B}_1, \mathcal{B}_2\rangle_{-(1,2)}, D\rangle_{4,2;1,2},\\
&\ \,\vdots\\
\mathcal{H}_{ij}(D) &= \langle \langle \mathcal{C}_{ij}, \mathcal{F}\rangle_{-(i,j)}, D\rangle_{2,4;1,2} + \langle \langle \mathcal{B}_i, \mathcal{B}_j\rangle_{-(i,j)}, D\rangle_{4,2;1,2},
\end{aligned}$$

where $i \neq j$, $i < j$ and $i, j = 1, \dots, k$. Similarly, due to symmetry all of them are identical. We have

$$H_{\text{off-diag}}(D) = k(k-1)\left(\langle \langle \mathcal{C}_{12}, \mathcal{F}\rangle_{-(1,2)}, D\rangle_{2,4;1,2} + \langle \langle \mathcal{B}_1, \mathcal{B}_2\rangle_{-(1,2)}, D\rangle_{4,2;1,2}\right).$$

The complete Grassmann Hessian operator is simply

$$H = H_{\text{diag}} + H_{\text{off-diag}}.$$

9.4. Matricizing the Hessian operator. The second order terms are described using the canonical inner product on Grassmannians and contracted tensor products. Next we will derive the expression of the Hessian as a matrix acting on the vector $d = \text{vec}(\Delta)$.

The terms in (9.5) involve only matrix operations and vectorizing the second argument in the inner product yields

$$\begin{aligned}
\text{vec}\left((\mathcal{H}_1(\Delta)\right) &= \text{vec}(3\langle \mathcal{B}_1, \mathcal{B}_1\rangle_{-1}\Delta - 3\Pi_X \Delta \langle \mathcal{F}, \mathcal{F}\rangle_{-1})\\
&= 3\left(I \otimes \langle \mathcal{B}_1, \mathcal{B}_1\rangle_{-1} - \langle \mathcal{F}, \mathcal{F}\rangle_{-1} \otimes \Pi_X\right)\text{vec}(\Delta)\\
&\equiv H_1 d.
\end{aligned}$$

The vectorization of the terms from (9.6) and (9.7) involve the 4-tensors

$$\begin{aligned}
\mathcal{H}_2 &= \langle \mathcal{C}_{12}, \mathcal{F}\rangle_{-(1,2)} \in \mathbb{R}^{n\times n\times r\times r},\\
\mathcal{H}_3 &= \langle \mathcal{B}_1, \mathcal{B}_2\rangle_{-(1,2)} \in \mathbb{R}^{n\times r\times r\times n},
\end{aligned}$$

and is done using the tensor matricization described in [25]. We get

$$\text{vec}(\langle \mathcal{H}_2, \Delta\rangle_{2,4;1,2}) = H_2^{(3,1;4,2)}\,\text{vec}(\Delta) \equiv H_2 d, \tag{9.8}$$

$$\text{vec}(\langle \mathcal{H}_3, \Delta\rangle_{4,2;1,2}) = H_3^{(3,1;2,4)}\,\text{vec}(\Delta) \equiv H_3 d. \tag{9.9}$$

In $\mathcal{H}_2$ we map indices of the first and third mode to row-indices and indices of the second and fourth mode to column-indices obtaining the matrix $H_2^{(3,1;4,2)}$. In this way the contractions in the matrix-vector product coincide with the tensor-matrix contractions. Similarly for H_3. The matrix form of the Hessian becomes

$$H = H_1 + H_2 + H_3.$$

To obtain the Hessian in local coordinates we replace Π_X with $X_\perp$ in the computations of the factors involved and thereafter perform the same matricization procedure.

10. Examples. We will now give two small explicit examples to illustrate the computations involved in the algorithms for tensor approximation described before.

EXAMPLE 10.1. *In this example we will compute the gradient of the objective function, both in global and in local coordinates. Let the $3 \times 3 \times 3$ tensor $\mathcal{A}$ be given by*

$$\mathcal{A}(:,:,1) = \begin{bmatrix} 9 & -3 & 8 \\ 2 & 7 & 0 \\ 7 & 0 & -1 \end{bmatrix}, \mathcal{A}(:,:,2) = \begin{bmatrix} 2 & 7 & 0 \\ -7 & 5 & -3 \\ 0 & -3 & 1 \end{bmatrix}, \mathcal{A}(:,:,3) = \begin{bmatrix} 3 & 0 & -2 \\ 0 & 4 & -1 \\ 0 & -2 & 1 \end{bmatrix}.$$

Let the current point of the product manifold be given by (X, Y, Z) where

$$X = Y = Z = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \textit{and} \qquad \Pi_X = \Pi_Y = \Pi_Z = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

are the corresponding projection matrices onto the three tangent spaces. The expression for the Grassmann gradient at the current iterate is given by (8.1)–(8.3). *The intermediate quantities, cf. equation* (8.7), *needed in the calculations of the Grassmann gradient are*

$$\begin{aligned} \mathcal{F} &= \mathcal{A} \cdot (X, Y, Z) = 9, \\ \mathcal{B}_x &= \mathcal{A} \cdot (\Pi_X, Y, Z) = (0\ \ 2\ \ 7)^{\mathsf{T}}, \\ \mathcal{B}_y &= \mathcal{A} \cdot (X, \Pi_Y, Z) = (0\ \ -3\ \ 8)^{\mathsf{T}}, \\ \mathcal{B}_z &= \mathcal{A} \cdot (X, Y, \Pi_Z) = (0\ \ 2\ \ 3)^{\mathsf{T}}, \end{aligned} \tag{10.1}$$

and the Grassmann gradient in global coordinates is given by

$$\nabla\Phi = (\langle \mathcal{B}_x, \mathcal{F} \rangle_{-1}, \langle \mathcal{B}_y, \mathcal{F} \rangle_{-2}, \langle \mathcal{B}_z, \mathcal{F} \rangle_{-3}) = \left(\begin{bmatrix} 0 \\ 18 \\ 63 \end{bmatrix}, \begin{bmatrix} 0 \\ -27 \\ 72 \end{bmatrix}, \begin{bmatrix} 0 \\ 18 \\ 27 \end{bmatrix} \right).$$

To compute the Grassmann gradient in local coordinates we need a basis for the tangent spaces. For the current iterate we choose

$$X_\perp = Y_\perp = Z_\perp = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

as the corresponding basis matrices for the tangent spaces at X, Y and Z. Obviously $[X\ X_\perp]$, $[Y\ Y_\perp]$ and $[Z\ Z_\perp]$ are orthogonal and $X^{\mathsf{T}} X_\perp = Y^{\mathsf{T}} Y_\perp = Z^{\mathsf{T}} Z_\perp = 0$. Replacing the projection matrices Π_X, Π_Y and Π_Z by the orthogonal complements $X_\perp$, $Y_\perp$ and $Z_\perp$ in (10.1), *we obtain $\widehat{\mathcal{B}}_x, \widehat{\mathcal{B}}_y, \widehat{\mathcal{B}}_z$, and thus the local coordinate representation of the Grassmann gradient is given by*

$$\nabla\widehat{\Phi} = (\langle \widehat{\mathcal{B}}_x, \mathcal{F} \rangle_{-1}, \langle \widehat{\mathcal{B}}_y, \mathcal{F} \rangle_{-2}, \langle \widehat{\mathcal{B}}_z, \mathcal{F} \rangle_{-3}) = \left(\begin{bmatrix} 18 \\ 63 \end{bmatrix}, \begin{bmatrix} -27 \\ 72 \end{bmatrix}, \begin{bmatrix} 18 \\ 27 \end{bmatrix} \right).$$

Recall that we use a hat to distinguish local coordinate representation from global coordinate representation. The local coordinate representation is depending on the choice of basis matrices for the tangent spaces. A different choice of $X_\perp$, $Y_\perp$ and $Z_\perp$ would yield a different representation of $\nabla\widehat{\Phi}$.

Example 10.2. *Next we will illustrate the parallel transport of tangent vectors along geodesics on a product of Grassmannians. Let the tensor $\mathcal{A}$, the current iterate, and the corresponding gradient be the same as in the previous example. Introduce tangent vectors*

$$\Delta = (\Delta_x,\, \Delta_y,\, \Delta_z) = \left(\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right).$$

Clearly we have $X^{\mathsf{T}}\Delta_x = Y^{\mathsf{T}}\Delta_y = Z^{\mathsf{T}}\Delta_z = 0$. The tangent Δ will determine the geodesic path from the current point and in turn the transport of the Grassmann gradient (see Figure 10.1). We may also verify that $\nabla\Phi$ is indeed a tangent of the product Grassmannian at the current iterate.

The thin or compact SVD*s, written $\Delta_* = U_* \cdot \Sigma_* \cdot V_*^{\mathsf{T}}$, of the tangents are*

$$\Delta_x = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \cdot 1 \cdot 1, \quad \Delta_y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot 1 \cdot 1, \quad \Delta_z = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot 1 \cdot 1.$$

The transport matrix, cf. equation (5.2)*, in the direction Δ_x at X with a step size $t = \pi/4$ is given by*

$$T_{X,\Delta_x}(t)\big|_{t=\pi/4} = \begin{bmatrix} XV_x & U_x \end{bmatrix} \begin{bmatrix} -\sin\Sigma_x(\pi/4) \\ \cos\Sigma_x(\pi/4) \end{bmatrix} U_x^{\mathsf{T}} + (I - U_xU_x^{\mathsf{T}}) = \begin{bmatrix} 1 & 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Similarly, it is straightforward to calculate

$$T_{Y,\Delta_y}(t)\big|_{t=\pi/4} = \begin{bmatrix} 1 & 0 & -1/\sqrt{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1/\sqrt{2} \end{bmatrix} \quad \text{and} \quad T_{Z,\Delta_z}(t)\big|_{t=\pi/4} = \begin{bmatrix} 1 & -1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Parallel transporting one tangent we get $T_{X,\Delta_x}(\pi/4)\Delta_x = (-1/\sqrt{2},\ -1/\sqrt{2},\ 0)^{\mathsf{T}}$. For all tangents in Δ and $\nabla\Phi$ we get

$$\Delta(t)|_{t=\pi/4} = \left(\begin{bmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix}, \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}, \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix} \right),$$

$$\nabla\Phi(t)|_{t=\pi/4} = \left(\begin{bmatrix} 18/\sqrt{2} \\ 18/\sqrt{2} \\ 63 \end{bmatrix}, \begin{bmatrix} -72/\sqrt{2} \\ -27 \\ 72/\sqrt{2} \end{bmatrix}, \begin{bmatrix} -18/\sqrt{2} \\ 18/\sqrt{2} \\ 27 \end{bmatrix} \right).$$

The above are calculations in global coordinates. In local coordinates we parallel transport the basis matrices $X_\perp$, $Y_\perp$ and $Z_\perp$ so that the local coordinate representation of a tangent is the same as in the previous point. The computations are given by equation (5.3) *and in this example we get*

$$X_\perp(\pi/4) = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \\ 0 & 1 \end{bmatrix}, \ Y_\perp(\pi/4) = \begin{bmatrix} 0 & -1/\sqrt{2} \\ 1 & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}, \ Z_\perp(\pi/4) = \begin{bmatrix} -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \\ 0 & 1 \end{bmatrix},$$

i.e. the second and third columns of each transport matrix due to the specific choice of $X_\perp$, $Y_\perp$ and $Z_\perp$.

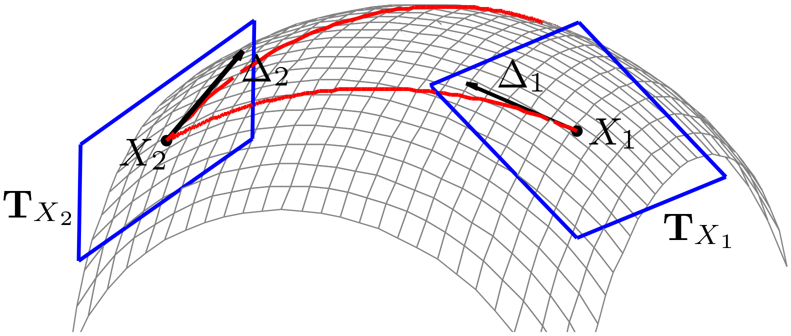

FIG. 10.1. *Pictorial depiction of the main algorithmic procedure for the Grassmannian* Gr(3, 1)*, which is simply the sphere* S^2.

Taking a step of size $t = \pi/4$ *from* X, Y *and* Z *along the specified geodesic we arrive at*

$$X(\pi/4) = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix}, \quad Y(\pi/4) = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}, \quad Z(\pi/4) = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}.$$

The value of the objective function at the starting point is $\Phi(X, Y, Z) = 40.5$ *and at the new point is* $\Phi\left(X(\pi/4), Y(\pi/4), X(\pi/4)\right) \approx 45.5625$*, an increment as expected.*

Figure 10.1 illustrates the procedures involved in the algorithms on the Grassmannian Gr(3, 1), which we may regard as the 2-sphere S^2 (unit sphere in $\mathbb{R}^3$). For the best rank-1 tensor approximation of a $3 \times 3 \times 3$ tensor, the optimization takes place on a product of three spheres $S^2 \times S^2 \times S^2$, one for each vector that needs to be determined. The procedure starts at a point X_1 and a direction of ascent[6], the tangent $\Delta_1 \in \mathbf{T}_{X_1}$, is obtained through some method. Next we perform a movement of the point X_1 along the geodesic defined by Δ_1. Geodesics on spheres are just great circles. At the new point $X_2 \in \text{Gr}(3, 1)$ we repeat the procedure, i.e. determine a new direction of ascent Δ_2 and take a geodesic step in this direction.

11. Numerical experiments and computational complexity.

All algorithms described here and the object oriented Grassmann classes required for them are available for download as two MATLAB packages [50] and [51]. We encourage our readers to try them out.

11.1. Initialization and stopping condition.

We will now test the actual performance of our algorithms with a few large numerical examples. All algorithms in a given test are started with the same initial points on a Grassmannian, represented as truncated singular matrices from the HOSVD and a number of additional *higher order orthogonal iterations*—HOOI iterations [15, 16], which are introduced to make the initial Hessian of Φ negative definite. The number of initial HOOI iterations ranges between 5 and 50 depending on the size of the problem. The BFGS algorithm is either started with (possibly a modification of) the exact Hessian or a scaled identity matrix according to [46, pp. 143]. The L-BFGS algorithm is always started with a scaled identity matrix but one can modify the number of columns m in the matrices representing the Hessian approximation, see equation (7.2). This number is between

[6] Recall that we are maximizing Φ, therefore 'ascent' as opposed to 'descent'.

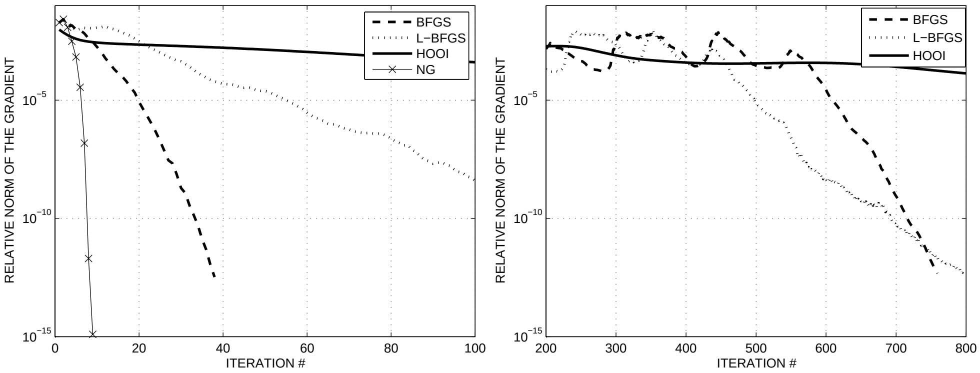

FIG. 11.1. *Left: A $20 \times 20 \times 20$ tensor is approximated by a rank-$(5,5,5)$ tensor.* BFGS *initiated with the exact Hessian; In* L-BFGS *$m = 5$. Right: A $100 \times 100 \times 100$ tensor is approximated by a rank-$(5,10,20)$ tensor. In this case the initial Hessian is a scaled identity and $m = 10$.*

5 and 30. Although we use the HOSVD to initialize our algorithms, any other reasonable initialization procedure would work as long as the initial Hessian approximate is negative definite. The quasi-Newton methods can be used as stand-alone algorithms for solving the tensor approximation problem as well as other problems defined on Grassmannians.

In the following figures, the y-axis measures the norm of the relative gradient, i.e. $\|\nabla\Phi(X)\|/\|\Phi(X)\|$, and the x-axis shows iterations. This ratio is also used as our stopping condition, which typically requires that $\|\nabla\Phi(X)\|/\|\Phi(X)\| \approx 10^{-13}$, the machine precision of our computer. At a true local maximizer the gradient of the objective function is zero and its Hessian is negative definite. In the various figures we present convergence results for four principally different algorithms. These are (1) quasi-Newton-Grassmann with BFGS, (2) quasi-Newton-Grassmann with L-BFGS, (3) Newton-Grassmann, denoted with NG and (4) HOOI which is an alternating least squares approach. In addition, the tags for BFGS methods may be accompanied by I or H indicating whether the initial Hessian was a scaled identity matrix or the exact Hessian, respectively.

11.2. Experimental results. We run all our numerical experiments in MATLAB on a MacBook with a 2.4-GHz Intel Core 2 Duo processor and 4 GB of physical memory.

Figure 11.1 shows convergence results for two tests with tensors generated with $N(0,1)$-distributed values. In the left plot a $20 \times 20 \times 20$ tensor is approximated with a rank-$(5,5,5)$ tensor. One can observe superlinear convergence in the BFGS method. The right plot shows convergence results of a $100 \times 100 \times 100$ tensor approximated with a rank-$(5,10,20)$ tensor. Both BFGS and L-BFGS methods exhibit rapid convergence in the vicinity of a stationary point.

Figure 11.2 (*left*) shows convergence for an even larger $200 \times 200 \times 200$ tensor approximated by a tensor of rank-$(10,10,10)$ using L-BFGS with $m = 20$. In the right plot we approximate a $50 \times 50 \times 50$ tensor by a rank-$(20,20,20)$ tensor where we vary over a range of values of m in the L-BFGS algorithm, namely, $m = 5, 10, 15, 20, 25, 30$. $m = 5$ gives (in general) slightly poorer performance, otherwise the different runs cannot be distinguished. In other words, our Grassmann L-BFGS algorithm can in practice work as well as our Grassmann BFGS algorithm, just as one would expect

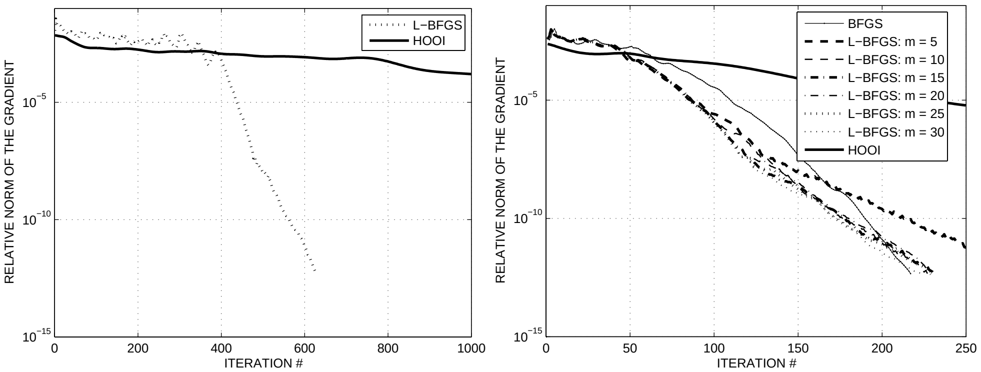

FIG. 11.2. *Left: Convergence plots of a* $200\times200\times200$ *tensor approximated by a rank-*$(10,10,10)$ *tensor. Right: Effect of varying* m *in* L-BFGS. *A* $50\times50\times50$ *tensor approximated by a rank-*$(20,20,20)$ *tensor with* $m=5,10,15,20,25,30$.

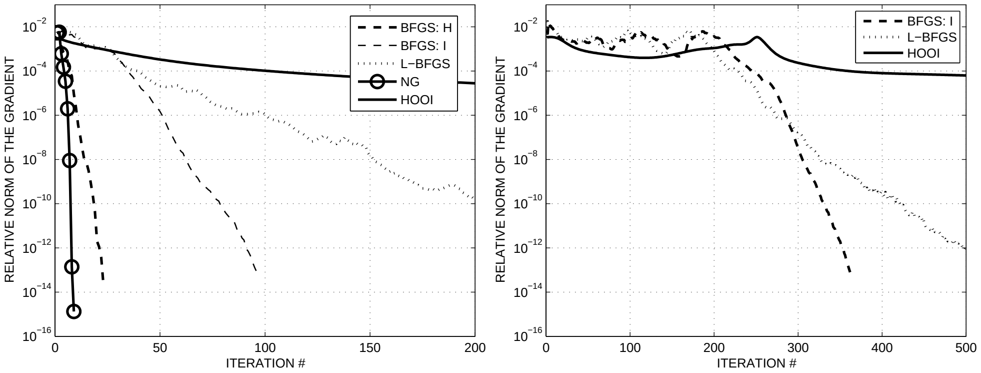

FIG. 11.3. *Left: A* $50\times50\times50$ *symmetric tensor is approximated by a rank-*5 *symmetric tensor;* $m=10$. *Right: Here we have a* $100\times100\times100$ *symmetric tensor approximated by a rank-*20 *symmetric tensor;* $m=10$.

(from the numerical experiments performed) in the Euclidean case.

Figure 11.3 shows convergence plots for two symmetric tensor approximation problems. In the left plot we approximate a symmetric $50\times50\times50$ tensor by a rank-5 symmetric tensor. We observe that BFGS initialized with the exact Hessian (BFGS:H tag) converges much more rapidly, almost as fast as the Newton-Grassmann method, than when initialized with a scaled identity matrix (BFGS:I tag). In the right plot we give convergence results for a $100\times100\times100$ symmetric tensor approximated by a rank-20 symmetric tensor. In both cases $m=10$.

In Figure 11.4 we show the performance of a local coordinate implementation of the BFGS algorithm on problems with 4-tensors. The first plot shows convergence results for a $50\times50\times50\times50$ tensor approximated by a rank-$(5,5,5,5)$ tensor. The second convergence plot is for a symmetric 4-tensor with the same dimensions approximated by a symmetric rank-5 tensor. Again the H and I tags indicate whether the exact Hessian or a scaled identity is used for initialization.

We end this section with two unusual examples to illustrate the extent of our algorithms' applicability: a high order tensor and an objective function that includes tensors of different orders. The left plot in Figure 11.5 is a high-order example: it

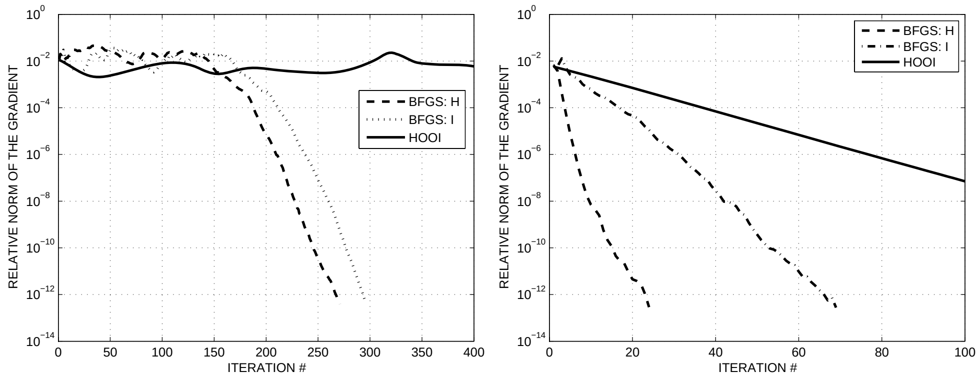

FIG. 11.4. *Left: A* $50 \times 50 \times 50 \times 50$ *tensor is approximated by a rank-*$(5, 5, 5, 5)$ *tensor. Right: Here we have a* $50 \times 50 \times 50 \times 50$ *symmetric tensor approximated by a rank-*5 *symmetric tensor.*

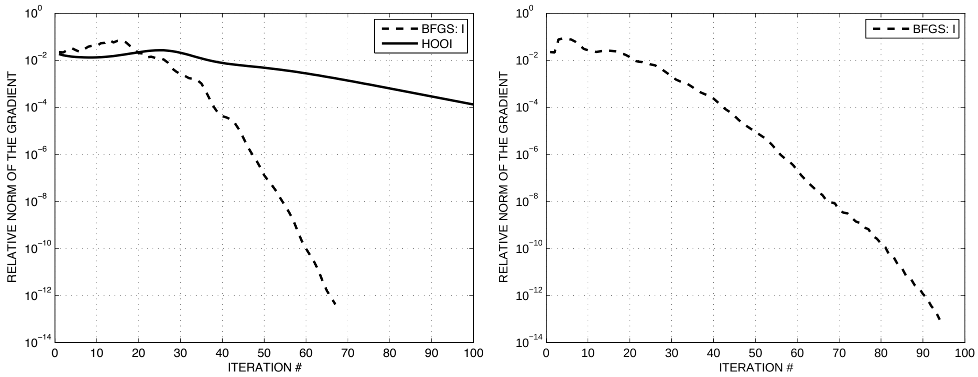

FIG. 11.5. *Left: A* $5 \times 5 \times \cdots \times 5$ *tensor of order-10 is approximated with a rank-*$(2, 2, \ldots, 2)$*. Right: A 'simultaneous' rank-*5 *approximation of a weighted sum of tensors of orders* 2*,* 3*, and* 4*.*

shows the convergence of BFGS verses HOOI when approximating an order-10 tensor with dimensions $5 \times 5 \times \cdots \times 5$ with a rank-$(2, 2, \ldots, 2)$ tensor. The right plot in Figure 11.5 has an unusual objective function that involves an order-2, an order-3, and an order-4 tensor,

$$\Phi(X) = \frac{1}{2!}\|S_2 \cdot (X, X)\|_F^2 + \frac{1}{3!}\|\mathcal{S}_3 \cdot (X, X, X)\|_F^2 + \frac{1}{4!}\|\mathcal{S}_4 \cdot (X, X, X, X)\|_F^2$$

where S_2 is a 30×30 symmetric matrix, $\mathcal{S}_3$ is a $30 \times 30 \times 30$ symmetric 3-tensor, and $\mathcal{S}_4$ is a $30 \times 30 \times 30 \times 30$ symmetric 4-tensor. Such objective functions have appeared in *independent component analysis with soft whitening* [18] and in *principal cumulants components analysis* [44, 41] where $S_2, \mathcal{S}_3, \mathcal{S}_4$ measure the multivariate variance, kurtosis, skewness respectively (cf. Example 2.2). In both examples we observe a fast rate of convergence at the vicinity of a local minimizer for the BFGS algorithm.

It is evident from the convergence plots here that BFGS and L-BFGS have faster rate of convergence compared with HOOI. The Newton-Grassmann algorithm takes few iterations but is computationally more expensive, specifically for larger problems. Our implementation of the different algorithms in MATLAB give shortest runtime for the BFGS and L-BFGS methods. The time for one iteration of BFGS, L-BFGS and HOOI is

Algorithm 1 Algorithmic framework for BFGS and L-BFGS on Grassmannians.

```
Given tensor $\mathcal{A}$ and starting points $(X_0, Y_0, Z_0) \in \mathrm{Gr}^3$ and an initial Hessian $H_0$
repeat
    1 Compute the Grassmann gradient.
    2 Parallel transport the Hessian approximation to the new point.
    3 Update the Hessian or its compact representation.
    4 Solve the quasi-Newton equations to obtain $\Delta = (\Delta_x, \Delta_y, \Delta_z)$.
    5 Move the points $(X_k, Y_k, Z_k)$ along the geodesic curve given by $\Delta$.
until $\|\nabla\widehat{\Phi}\|/\Phi <$ TOL
```

of the same magnitude for smaller problems. In larger problems, the L-BFGS performs much faster than all other methods.

Our algorithms use the basic arithmetic and data types in the TensorToolbox [4] for convenience. We use our own object-oriented routines for operations on Grassmannians and product of Grassmannians, e.g. geodesic movements and parallel transports [51]. We note that there are several different ways to implement BFGS updates [46]; for simplicity reasons, we have chosen to update the inverse of the Hessian approximation. A possibly better alternative will be to update the Cholesky factors of the approximate Hessians so that one may monitor the approximate Hessians for indefiniteness during the iterations [23, 28, 7].

11.3. Computational complexity, curse of dimensionality, and convergence. The Grassmann quasi-Newton methods presented in this report all fit within the procedural framework given in Algorithm 1.

General case. In analyzing computational complexity, we will assume for simplicity that $\mathcal{A}$ is a general $n \times n \times n$ 3-tensor being approximated with a rank-(r, r, r) 3-tensor. A problem of these dimensions will give rise to a $3nr \times 3nr$ Hessian matrix in global coordinates and a $3(n-r)r \times 3(n-r)r$ Hessian matrix in local coordinates. Table 11.1 gives approximately the amount of computations required in each step of Algorithm 1. Recall that in L-BFGS m is a small number, see Section 7. We have omitted terms of lower asymptotic complexity as well as the cost of point **5** since that is negligible compared with the costs of points **1**–**4**. For example, the geodesic movement of X_k requires the thin SVD $U_x \Sigma_x V_x^{\mathsf{T}} = \Delta_x \in \mathbb{R}^{n \times r}$ which takes $6nr^2 + 20r^3$ flops (floating point operations) [29]. On the other hand, given the step length t and U, Σ, V in (5.1), the actual computation of $X(t)$ amounts to only $4nr^2$ flops.

	BFGS-GC	BFGS-LC	L-BFGS
1	$6n^3r + 12n^2r^2$	$6n^3r + 6n^2r^2 + 6n(n-r)r^3$	$6n^3r + 12n^2r^2$
2	$18n^3r^2$	—	$12n^2rm$
3	$36n^2r^2$	$36(n-r)^2r^2$	—
4	$18n^2r^2$	$18(n-r)^2r^2$	$24nrm$

TABLE 11.1

Computational complexity of the BFGS-GC *(global coordinates),* BFGS-LC *(local coordinates) and* L-BFGS *algorithms. The numbers in the first column correspond to the steps in Algorithm 1.*

Symmetric case. The symmetric tensor approximation problem involves the determination of one $n \times r$ matrix, resulting in an $nr \times nr$ Hessian in global coordinates and an $(n-r)r \times (n-r)r$ Hessian in local coordinates. Therefore the complexity of the symmetric problem differs only by a constant factor from that of the general case.

Curse of dimensionality. The approximation problem will suffer from the *curse of dimensionality* when the order of a tensor increases. In general, an $n \times \cdots \times n$ order-k tensor requires the storage of n^k entries in memory. The additional memory requirement, mainly for storing the Hessian, is of order $O(n^2r^2k^2)$ for the BFGS methods and $O(2nrkm)$ for the L-BFGS method, respectively. In the current approach we assume that the tensor is explicitly given. Our proposed algorithms are applicable as long as the given tensor fits in memory. There have been various proposals to deal with the curse of dimensionality using tensors [35, 36, 48]. For cases where the tensor is represented in compact or functional forms our methods can take direct advantage of these simply by computing the necessary gradients (and Hessians) using the specific representations. In fact this was considered in [43] for symmetric tensor approximations.

Convergence. There is empirical evidence suggesting that ALS based algorithms have fast convergence rate for specific tensors. This was also pointed out in [12]. These are tensors that have inherently low multilinear rank and the approximating tensor has the correct low ranks, or tensors that have fast decay in its multilinear singular values [15], or a substantial gap in the multilinear singular values at the site of truncation, e.g. the source tensor is given by a low rank tensor with noise added. On the other hand not all tensors have gaps or fast decaying multilinear singular vales. This is specifically true for sparse tensors. It is still desirable to obtain low rank approximations for these “more difficult” tensors. And on these tensors ALS performs very poorly, but methods using first and second order derivatives of the objective function, including the methods presented in this paper perform good. Among the methods that are currently available, quasi-Newton methods presented in this paper have the best computational efficiency.

12. Related work.

There are several different approaches to solve the tensor approximation problem. In this section we will briefly describe them and point out the main differences with our work. The algorithms most closely related to the quasi-Newton methods are given in [25, 34, 33]. All three references address the best low rank tensor approximation based on the Grassmannian structure of the problem and use explicit computation of the Hessian. The obtained Newton equations are solved either fully [25, 34] or approximately [33]. In the latter case the authors used a truncated conjugate gradient approach to approximately solve the Newton equations. The iterates are updated using the more general notion of retractions instead of taking a step along the geodesic on the manifold. In addition a trust region scheme is incorporated making the procedure more stable with respect to occasional indefinite Hessians. The computation of the Hessian is a limiting factor in these algorithms. This is the case even when the Hessian is not formed explicitly but used implicitly via its action on a tangent. In our experiments, on moderate-sized problems, e.g. 3-tensors of dimensions around $20 \times 20 \times 20$, the BFGS methods noticeably outperformed Hessian-based methods; and for dimensions around $100 \times 100 \times 100$, we were unable to get any methods relying on Hessians to work despite our best efforts.

There is a different line of algorithms for related tensor approximation problems based on ALS and mutltigrid accelerated ALS [35, 36]. In our experience, the convergence of ALS-type methods depend on the decay of the multilinear singular values of the given tensor. The exact dependence is unclear but the relation seems to be that the faster the decay, the faster the convergence of ALS. In this regard the class of functions and operators considered in [35, 36] appears to possess these favorable properties.

Yet a third approach to obtain low multilinear rank tensor approximations are the cross methods in [49, 48, 26]. The novelty of such methods is that they discard some given information and retain only a fraction of the original tensor, and as such it is markedly different from our approach, which uses all given information to achieve maximal accuracy. In addition, there is an assumption on the tensor that there exist approximations within pre-specified bounds and of specific low ranks while we make no such assumptions.

13. Conclusion. In this paper we studied quasi-Newton algorithms adapted to optimization problems on Riemannian manifolds. More specifically, we proposed algorithms with BFGS and L-BFGS updates on a product of Grassmannians that (1) respect the Riemannian metric structure and (2) require only standard matrix operations in their implementations. Two different algorithmic implementations are presented: one based on local/intrinsic coordinates while the other one uses global/embedded coordinates. In particular, our use of local coordinates is a novelty not previously explored in other manifold optimization [1, 2, 24, 27]. We proved the optimality of our Grassmannian BFGS updates in local coordinates, showing that the well-known BFGS optimality [21, 22] extends to Grassmannian and products of Grassmannians.

We also applied these algorithms to the problem of determining a best multilinear rank approximation of a tensor and the analogous (but very different) problem for a symmetric tensor. While a Newton version of this was proposed in [25], here we make substantial improvements with respect to the Grassmann-Newton algorithm in terms of speed and robustness. Furthermore, we presented specialized algorithms that take into account the symmetry in the multilinear approximation of symmetric tensors and related problems. In addition to the numerical experiments in this paper, we have made our codes freely available for download [50, 51] so that the reader may verify the speed, accuracy, and robustness of these algorithms for himself.

Appendix A. Notation for tensor contractions.

In this section we define the contracted tensor product notation used throughout this paper. For given third order tensors $\mathcal{A}$ and $\mathcal{B}$ we define the following contracted products:

$$\mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_1, \qquad c_{ijkl} = \sum_{\lambda} a_{\lambda ij} b_{\lambda kl}. \tag{A.1}$$

When contracting several indices, with the corresponding indices of the two arguments being the same, we write

$$\mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_{1,2}, \qquad c_{ij} = \sum_{\lambda,\nu} a_{\lambda\nu i} b_{\lambda\nu j}. \tag{A.2}$$

The subscript '1' in $\langle \mathcal{A}, \mathcal{B} \rangle_1$ and subscripts '1,2' in $\langle \mathcal{A}, \mathcal{B} \rangle_{1,2}$ indicate that the contraction is over the first index and both the first and second indices respectively. If instead the contraction is to be performed on different indices, we write

$$\mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_{1;2}, \quad c_{ijkl} = \sum_{\lambda} a_{\lambda ij} b_{k\lambda l} \quad \text{or} \quad \mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_{1,3;2,1}, \quad c_{ij} = \sum_{\lambda,\nu} a_{\lambda i\nu} b_{\nu\lambda j}.$$

The subscripts indicating the indices to be contracted are separated by a semicolon. It is also convenient to introduce a notation when contraction is performed in all but one or a few indices. For example the products in (A.2) and (A.2) may also be written

$$\langle \mathcal{A}, \mathcal{B} \rangle_{1,2} = \langle \mathcal{A}, \mathcal{B} \rangle_{-3} \quad \text{or} \quad \langle \mathcal{A}, \mathcal{B} \rangle_{1} = \langle \mathcal{A}, \mathcal{B} \rangle_{-(2,3)}.$$

REFERENCES

[1] P.-A. Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds*, Found. Comput. Math., 7 (2007), no. 3, pp. 303–330.

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

[3] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen, *LAPACK Users' Guide*, 3rd Ed., SIAM, Philadelphia, PA, 1999.

[4] B. W. Bader and T. G. Kolda, *Algorithm 862: MATLAB tensor classes for fast algorithm prototyping*, ACM Trans. Math. Software, 32 (2006), no. 4, pp. 635–653.

[5] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, 2nd Ed., Academic Press, Orlando, FL, 1986.

[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited memory methods*, Math. Program., 63 (1994), no. 2, pp. 129–156.

[7] S. H. Cheng and N. J. Higham, *A modified cholesky algorithm based on a symmetric indefinite factorization*, SIAM J. Matrix Anal. Appl., 19 (1998), no. 4, pp. 1097–1110.

[8] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, *Symmetric tensors and symmetric tensor rank*, SIAM J. Matrix Anal. Appl., 39 (2008), no. 3, pp. 1254–1279.

[9] ———, *Genericity and rank deficiency of high order symmetric tensors*, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process. (ICASSP '06), 31 (2006), pp. 125–128.

[10] P. Comon and B. Mourrain, *Decomposition of quantics in sums of powers of linear forms*, Signal Process., 53 (1996), pp. 93–107.

[11] L. Conlon, *Differentiable Manifolds*, 2nd Ed., Birkhäuser, Boston, MA, 2001.

[12] L. De Lathauwer, *Tucker compression, parallel factor analysis and block term decompositions: New results*, European Meeting on Challenges in Modern Massive Data Sets (EMMDS '09), Copenhagen, Denmark, 2009.

[13] ———, *Signal Processing Based on Multilinear Algebra*, Ph.D. thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[14] L. De Lathauwer, B. De Moor, and J. Vandewalle, *An introduction to independent component analysis*, J. Chemometrics, 14 (2000), pp. 123–149.

[15] ———, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), no. 4, pp. 1253–1278.

[16] ———, *On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), no. 4, pp. 1324–1342.

[17] L. De Lathauwer, L. Hoegaerts, and J. Vandewalle, *A Grassmann-Rayleigh quotient iteration for dimensionality reduction in ICA*, Proc. Int. Conf. Independent Component Analysis and Blind Signal Separation (ICA '04), 5 (2004), pp. 335–342.

[18] L. De Lathauwer and J. Vandewalle, *Dimensionality reduction in higher-order signal processing and rank-$(r_1, r_2, \ldots, r_n)$ reduction in multilinear algebra*, Linear Algebra Appl., 391 (2004), pp. 31–55.

[19] ———, *Dimensionality reduction in ICA and rank-$(r_1, r_2, \ldots, r_n)$ reduction in multilinear algebra*, Proc. Int. Conf. Independent Component Analysis and Blind Signal Separation (ICA '04), 5 (2004), pp. 295–302.

[20] V. De Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), no. 3, pp. 1084–1127.

[21] J. E. Dennis and J. Moré, *Quasi-newton methods, motivation and theory*, SIAM Rev., 19 (1997), no. 1, pp. 46–89.

[22] J. E. Dennis and R. B. Schnabel, *A new derivation of symmetric positive definite secant updates*, Nonlinear Program., 4 (1981), pp. 167–199.

[23] ———, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, PA, 1996.

[24] A. Edelman, T. A. Arias, and S. T. Steven, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), no. 2, pp. 303–353.

[25] L. Eldén and B. Savas, *A Newton–Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor*, SIAM J. Matrix Anal. Appl., 31 (2009), no. 2, pp. 248–271.

[26] H.-J. Flad, B. N. Khoromskij, D. Savostianov, and E. Tyrtyshnikov, *Verification of the cross 3D algorithm on quantum chemistry data*, Russian J. Numer. Anal. Math. Modelling, 4 (2008), pp. 1–16.

[27] D. Gabay, *Minimizing a differentiable function over a differential manifold*, J. Optim. Theory Appl., 37 (1982), no. 2, pp. 177–219.

[28] D. Goldfarb, *Factorized variable metric methods for unconstrained optimization*, Math. Comp., 30 (1976), no. 136, pp. 796–811.

[29] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd Ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[30] W. Greub, *Multilinear Algebra*, 2nd Ed., Springer-Verlag, New York, NY, 1978.

[31] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys., 6 (1927), no. 1, pp. 164–189.

[32] M. Ishteva, L. De Lathauwer, P.-A. Absil, and S. Van Huffel, *Dimensionality reduction for higher-order tensors: Algorithms and applications*, Int. J. Pure Appl. Math., 42 (2008), no. 3, pp. 337–343.

[33] ———, *Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme*, Tech. Rep., 09-142, ESAT-SISTA, Katholieke Universiteit Leuven, Leuven, Belgium, 2009.

[34] ———, *Differential-geometric newton algorithm for the best rank-*(r_1, r_2, r_3) *approximation of tensors*, Numer. Algorithms, 51 (2009), no. 2, pp. 179–194.

[35] B. N. Khoromskij and V. Khoromskaia, *Low rank Tucker-type tensor approximation to classical potentials*, Cent. Eur. J. Math., 5 (2007), no. 3, pp. 523–550.

[36] ———, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comput., 31 (2009), no. 4, pp. 3002–3026.

[37] E. Kofidis and P. A. Regalia, *On the best rank-1 approximation of higher-order supersymmetric tensors*, SIAM J. Matrix Anal. Appl., 23 (2002), no. 3, pp. 863–884.

[38] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51(2009), no. 3, pp. 455–500.

[39] S. Lang, *Algebra*, Rev. 3rd Ed., Springer-Verlag, New York, NY, 2002.

[40] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide*, SIAM, Philadelphia, PA, 1998.

[41] L.-H. Lim and J. Morton, *Cumulant component analysis: a simultaneous generalization of PCA and ICA*, Computational Algebraic Statistics, Theories and Applications (CASTA '08), Kyoto University, Kyoto, Japan, December 10–11 2008.

[42] E. Lundström and L. Eldén, *Adaptive eigenvalue computations using Newton's method on the Grassmann manifold*, SIAM J. Matrix Anal. Appl., 23 (2002), no. 3, pp. 819–839.

[43] J. Morton, *Scalable implicit symmetric tensor approximation*, preprint, 2010.

[44] J. Morton and L.-H. Lim, *Principal cumulant components analysis*, preprint, 2009.

[45] M. Mørup, L. K. Hansen, S. M. Arnfred, *ERPWAVELAB: A toolbox for multi-channel analysis of time-frequency transformed event related potentials*, J. Neurosci. Methods, 161 (2007), no. 2, pp. 361–368.

[46] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd Ed., Springer, New York, NY, 2006.

[47] L. Omberg, G. H. Golub, and O. Alter, *A tensor higher-order singular value decomposition for integrative analysis of DNA microarray data from different studies*, Proc. Natl. Acad. Sci., 104 (2007), no. 47, pp. 18371–18376.

[48] I. V. Oseledets, *Compact matrix form of the d-dimensional tensor decomposition*, Tech. Rep., Institute of Numerical Mathematics, Russian Academy of Science, Moscow, Russia, 2009.

[49] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), no. 3, pp. 939–956.

[50] B. Savas, *Algorithm package manual: Best low rank tensor approximation*, Department of Mathematics, Linköping Univeristy, Linköping, Sweden, 2008. (`http://www.mai.liu.se/~besav/soft.html`)

[51] ———, *Toolbox for Grassmann manifold computations*, Department of Mathematics, Linköping Univeristy, Linköping, Sweden, 2008. (`http://www.mai.liu.se/~besav/soft.html`)

[52] L. Simonsson, *Subspace Computations via Matrix Decompositions and Geometric Optimization*, Linköping studies in science and technology, 1052, Linköping University, Linköping, Sweden, 2007.

[53] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*, John Wiley, West Sussex, England, 2004.

[54] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), no. 3, pp. 279–311.

[55] M. A. O. Vasilescu and D. Terzopoulos, *Multilinear subspace analysis of image ensembles*, Proc. Comput. Vis. Pattern Recognition Conf. (CVPR '03), 2 (2003), pp. 93–99.

[56] T. Yokonuma, *Tensor Spaces and Exterior Algebra*, AMS, Providence, RI, 1992.