SIAM J. MATRIX ANAL. APPL.
Vol. 32, No. 1, pp. 115–135

BEST LOW MULTILINEAR RANK APPROXIMATION OF HIGHER-ORDER TENSORS, BASED ON THE RIEMANNIAN TRUST-REGION SCHEME*

MARIYA ISHTEVA†, P.-A. ABSIL‡, SABINE VAN HUFFEL§, AND LIEVEN DE LATHAUWER§¶

Abstract. Higher-order tensors are used in many application fields, such as statistics, signal processing, and scientific computing. Efficient and reliable algorithms for manipulating these multiway arrays are thus required. In this paper, we focus on the best rank-(R_1, R_2, R_3) approximation of third-order tensors. We propose a new iterative algorithm based on the trust-region scheme. The tensor approximation problem is expressed as a minimization of a cost function on a product of three Grassmann manifolds. We apply the Riemannian trust-region scheme, using the truncated conjugate-gradient method for solving the trust-region subproblem. Making use of second order information of the cost function, superlinear convergence is achieved. If the stopping criterion of the subproblem is chosen adequately, the local convergence rate is quadratic. We compare this new method with the well-known higher-order orthogonal iteration method and discuss the advantages over Newton-type methods.

Key words. multilinear algebra, higher-order tensor, rank reduction, singular value decomposition, trust-region scheme, Riemannian manifold, Grassmann manifold

AMS subject classifications. 15A69, 65F99, 90C48, 49M37, 53B20

DOI. 10.1137/090764827

1. Introduction.

Higher-order tensors are multiway arrays of numbers, i.e., higher-order generalizations of vectors (first order) and matrices (second order). They are used as a tool in higher-order statistics [33, 29, 44, 32], independent component analysis [10, 11, 14, 8], and harmonic analysis [36, 37]. Fields of application of higher-order tensors also include chemometrics [42], biomedical signal processing [31, 15, 16, 4], telecommunications [40, 41, 10], scientific computing [7, 6, 20, 35], and image processing [48]. The main applications of the tensor approximation discussed in this paper are the dimensionality reduction of tensors with high dimensions [14, 4, 15, 16, 27, 42, 26] and signal subspace estimation [36, 37, 27, 42, 26, 19].

*Received by the editors July 13, 2009; accepted for publication (in revised form) December 1, 2010; published electronically February 8, 2011. This research was supported by (1) The Belgian Federal Science Policy Office, Interuniversity Attraction Poles P6/04, Dynamical Systems, Control, and Optimization (DYSCO), 2007–2011; (2) Communauté française de Belgique - Actions de Recherche Concertées; (3) Research Council, K.U. Leuven under GOA-AMBioRICS, GOA-MaNet, and CoE EF/05/006 Optimization in Engineering (OPTEC); (4) Fonds Wetenschappelijk Onderzoek–Vlaanderen project G.0427.10N "Integrated EEG-fMRI"; (5) Impulsfinanciering Campus Kortrijk (2007–2012)(CIF1) and STRT1/08/023. Part of this research was carried out while the first author was with K.U. Leuven, supported by OE/06/25, OE/07/17, OE/08/007, and OE/09/004. The scientific responsibility rests with the authors.

http://www.siam.org/journals/simax/49-1/76482.html

†Department of Mathematical Engineering, Université catholique de Louvain, Bâtiment Euler, Av. Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium. Current address: College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332 (mariya.ishteva@cc.gatech.edu).

‡Department of Mathematical Engineering, Université catholique de Louvain, Bâtiment Euler, Av. Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium (absil@inma.ac.be).

§Department of Electrical Engineering-ESAT/SCD, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10/2446, B-3001 Leuven, Belgium (sabine.vanhuffel@esat.kuleuven.be, lieven.delathauwer@esat.kuleuven.be).

¶Group Science, Engineering and Technology, Katholieke Universiteit Leuven Campus Kortrijk, E. Sabbelaan 53, B-8500 Kortrijk, Belgium (lieven.delathauwer@kuleuven-kortrijk.be).

Even some basic matrix concepts cannot be generalized to higher-order tensors in a unique manner. For example, generalization of different matrix-rank properties leads to different tensor-rank definitions. In this paper, we consider a definition of tensor-rank introduced in [21, 22] as a direct generalization of row and column rank of a matrix. The so-called mode-n vectors ($n = 1, 2, \ldots, N$) of an Nth-order tensor are vectors obtained by varying the nth index, while keeping the other indices fixed. The dimension of the vector space spanned by the mode-n vectors is called "mode-n rank." The multilinear rank of a tensor is then the n-tuple of the mode-n ranks. Contrary to the matrix case, different mode-n ranks are not necessarily equal to each other. In this paper, we look for the best low rank-$(R_1, R_2, \ldots, R_N)$ approximation of an Nth-order tensor. This is a tensor with bounded mode-n ranks as close as possible to the original tensor. In the sense of multilinear rank, a generalization of the singular value decomposition (SVD), giving the best low rank approximation of a matrix is the higher-order singular value decomposition (HOSVD) [12], also known as the Tucker decomposition [46, 47]. Its truncation results in a suboptimal solution of the best low rank-$(R_1, R_2, \ldots, R_N)$ approximation problem, which can serve as a good starting point for iterative algorithms.

The most widely used iterative algorithm is the higher-order orthogonal iteration (HOOI) [13, 27, 28]. Recently, Newton-type algorithms for the best low multilinear rank approximation of tensors have been developed in [25, 17, 39], the first one being based on [2]. In this paper, we develop an algorithm based on the trust-region scheme on Riemannian manifolds; we refer to [1] for the theory of the Riemannian trust-region method and to the generic Riemannian trust-region (GenRTR)[1] package for a generic MATLAB implementation. A summary of the first version of our algorithm has been proposed in [24]. The main advantage over HOOI of our new algorithm is its superlinear convergence speed. Moreover, for a well-chosen stopping criterion, the algorithm has quadratic convergence. The convergence speed of HOOI is only linear. On the other hand, the HOOI iterations are cheaper in general. However, if the order of the tensor is high or if the imposed multilinear rank is small compared to the tensor dimensions, which is often the case in practice, the cost per iteration of both algorithms is comparable. A detailed comparison of these two algorithms is given in section 5. Compared to the pure Newton-type algorithms, the advantage of the proposed algorithm is its more stable behavior. For the trust-region algorithm, convergence to stationary points is guaranteed for all initial points. This is not the case for the pure Newton-type methods. Moreover, in practice, the trust-region algorithm converges to a local minimum, whereas pure Newton-type methods may converge to any type of stationary point (minimum, maximum, or saddle point). A strength of the trust-region over the quasi-Newton algorithms is that the trust-region algorithms have rich convergence analysis [1]. On the other hand, less is known about quasi-Newton algorithms [39]. Finally, we mention other related methods that have appeared in the literature. A Krylov method has been proposed in [38] for large sparse tensors. Fast HOSVD algorithms for symmetric, Toeplitz, and Hankel tensors have been proposed in [5]. For tensors with large dimensions, a Tucker-type decomposition is developed in [35]. In the latter paper, it is required that a good Tucker approximation exists with low multilinear rank. The computation then involves only few entries of the given tensor and takes only linear time.

[1] http://www.math.fsu.edu/∼cbaker/GenRTR/.

This paper is organized as follows. In section 2, some basic definitions are given, and the problem of approximating a tensor by a low multilinear rank tensor is formulated in mathematical terms. Both HOSVD and HOOI are briefly presented. We consider third-order tensors for simplicity. In section 3, the basics of the trust-region method in $\mathbb{R}^n$, in a Euclidean space, and on a Riemannian manifold are given. Our trust-region–based algorithm for the best rank-(R_1, R_2, R_3) approximation of third-order tensors is described in section 4. Section 5 makes a comparison between the trust-region algorithm and HOOI. A comparison with Newton-type methods is made in section 6. We draw our conclusions in section 7.

Throughout this paper, we use the following notation. Tensors are denoted by calligraphic letters $(\mathcal{A}, \mathcal{B}, \ldots)$, boldface capitals $(\mathbf{A}, \mathbf{B}, \ldots)$ correspond to matrices, and scalars are written as lowercase letters $(a, b, \ldots)$. This means that the elements of a third-order tensor $\mathcal{A}$ are $a_{ijk} = (\mathcal{A})_{ijk}$. Some special scalars and upper bounds are denoted by capital letters $(I, I_1, I_2, I_3, N, R, \ldots)$. We use "$\times$" for the Cartesian product of sets and "$\otimes$" for the Kronecker product. The identity matrix is $\mathbf{I}$, $O_R = \{\mathbf{X} \in \mathbb{R}^{R\times R} : \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = \mathbf{I}\}$ denotes the orthogonal group (the set of all orthogonal $R \times R$ matrices), and $St(R, I) = \{\mathbf{X} \in \mathbb{R}^{I\times R} : \mathbf{X}^T\mathbf{X} = \mathbf{I}\}$ denotes the Stiefel manifold (the set of all columnwise orthonormal $I \times R$ matrices).

2. Problem formulation. In this section, we start with some basic definitions and notations. Further, we formulate the main problem and then we briefly summarize HOSVD and HOOI.

2.1. Basic definitions. We denote the *mode-n products* of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times I_3}$ with matrices $\mathbf{M}^{(n)} \in \mathbb{R}^{J_n\times I_n}$, $n = 1, 2, 3$, by $\mathcal{A} \bullet_n \mathbf{M}^{(n)}$, where

$$(\mathcal{A} \bullet_1 \mathbf{M}^{(1)})_{j_1 i_2 i_3} = \sum_{i_1} a_{i_1 i_2 i_3} m^{(1)}_{j_1 i_1},$$

$$(\mathcal{A} \bullet_2 \mathbf{M}^{(2)})_{i_1 j_2 i_3} = \sum_{i_2} a_{i_1 i_2 i_3} m^{(2)}_{j_2 i_2},$$

$$(\mathcal{A} \bullet_3 \mathbf{M}^{(3)})_{i_1 i_2 j_3} = \sum_{i_3} a_{i_1 i_2 i_3} m^{(3)}_{j_3 i_3}$$

and $1 \le i_n \le I_n$, $1 \le j_n \le J_n$. To make computations easier, it is convenient to define *matrix representations* $\mathbf{A}_{(n)}$, $n = 1, 2, 3$, of a tensor $\mathcal{A}$ in the following way:

$$(\mathbf{A}_{(1)})_{i_1,(i_2-1)I_3+i_3} = (\mathbf{A}_{(2)})_{i_2,(i_3-1)I_1+i_1} = (\mathbf{A}_{(3)})_{i_3,(i_1-1)I_2+i_2} = a_{i_1 i_2 i_3},$$

with $1 \le i_1 \le I_1$, $1 \le i_2 \le I_2$, $1 \le i_3 \le I_3$. The *scalar product* of two tensors $\mathcal{A}$ and $\mathcal{B}$ is defined as $\langle \mathcal{A}, \mathcal{B}\rangle = \sum_{i_1}\sum_{i_2}\sum_{i_3} a_{i_1 i_2 i_3}\, b_{i_1 i_2 i_3}$. Finally, the *Frobenius norm* of a tensor $\mathcal{A}$ is $\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A}\rangle}$.

2.2. The problem. In this paper, we look for the best rank-(R_1, R_2, R_3) approximation $\hat{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$ of an (unstructured) third-order tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times I_3}$. $\hat{\mathcal{A}}$ has to minimize the least-squares cost function $F : \mathbb{R}^{I_1\times I_2\times I_3} \to \mathbb{R}$,

$$F : \hat{\mathcal{A}} \mapsto \|\mathcal{A} - \hat{\mathcal{A}}\|^2, \tag{2.1}$$

under the constraints $\mathrm{rank}_1(\hat{\mathcal{A}}) \le R_1$, $\mathrm{rank}_2(\hat{\mathcal{A}}) \le R_2$, $\mathrm{rank}_3(\hat{\mathcal{A}}) \le R_3$, where $\mathrm{rank}_n(.)$ stands for the mode-n rank.

Instead of minimizing the cost function (2.1) we will solve the equivalent problem (see [13]) of maximizing the function $\overline{g} : St(R_1, I_1) \times St(R_2, I_2) \times St(R_3, I_3) \to \mathbb{R}$,

$$\overline{g} : (\mathbf{U}, \mathbf{V}, \mathbf{W}) \mapsto \|\mathcal{A} \bullet_1 \mathbf{U}^T \bullet_2 \mathbf{V}^T \bullet_3 \mathbf{W}^T\|^2 \tag{2.2}$$

over the columnwise orthonormal matrices $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$. It is sufficient to determine $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ in order to optimize (2.1). Knowing these matrices, the optimal tensor $\hat{\mathcal{A}}$ is computed as

$$\hat{\mathcal{A}} = \mathcal{B} \bullet_1 \mathbf{U} \bullet_2 \mathbf{V} \bullet_3 \mathbf{W}, \tag{2.3}$$

where $\mathcal{B} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is given by

$$\mathcal{B} = \mathcal{A} \bullet_1 \mathbf{U}^T \bullet_2 \mathbf{V}^T \bullet_3 \mathbf{W}^T .$$

2.3. HOSVD. For matrices, the solution of the best low rank approximation problem is given by the truncated SVD. The HOSVD [12, 46, 47] is a generalization of SVD to higher-order tensors. It decomposes a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ as a product of a so-called core tensor $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and three orthogonal matrices $\mathbf{U}^{(i)} \in \mathbb{R}^{I_i \times I_i}$, $i = 1, 2, 3$, i.e.,

$$\mathcal{A} = \mathcal{S} \bullet_1 \mathbf{U}^{(1)} \bullet_2 \mathbf{U}^{(2)} \bullet_3 \mathbf{U}^{(3)} .$$

In general, it is impossible to obtain a diagonal core tensor. However, the core tensor has certain properties implying a diagonal matrix in the second order case.

Computing HOSVD is straightforward. The singular matrices $\mathbf{U}^{(i)}, i = 1, 2, 3$, are obtained as the matrices of left singular vectors of the matrix representations $\mathbf{A}_{(i)}, i = 1, 2, 3$. The core tensor is computed as

$$\mathcal{S} = \mathcal{A} \bullet_1 \mathbf{U}^{(1)^T} \bullet_2 \mathbf{U}^{(2)^T} \bullet_3 \mathbf{U}^{(3)^T} .$$

Unfortunately, in general, truncated HOSVD gives a suboptimal solution of (2.1). However, it is often a good starting value for iterative algorithms.

2.4. HOOI. The HOOI [13, 27, 28] is an alternating least-squares algorithm for finding the best rank-(R_1, R_2, R_3) approximation of a tensor. The initial matrices $\mathbf{U}_0$, $\mathbf{V}_0$, $\mathbf{W}_0$ are taken from the truncated HOSVD. At each step, one of the matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ in (2.2) is updated, while the other two are kept constant. One cycles through updates of $\mathbf{U}, \mathbf{V}, \mathbf{W}$ in an alternating manner. In order to optimize $\mathbf{U}$, the left R_1-dimensional dominant subspace of $\mathbf{A}_{(1)}(\mathbf{V} \otimes \mathbf{W})$ has to be computed. Updates of $\mathbf{V}$ and $\mathbf{W}$ are obtained in a similar way. The convergence rate of HOOI is at most linear.

3. Riemannian trust-region scheme. In this section, we recall the idea behind trust-region methods. We consider cost functions in $\mathbb{R}^n$, then move to a general Euclidean space, and finally to the more general case of a Riemannian manifold.

The trust-region method (see [9, 34]) is an iterative method for minimizing a cost function. At each iteration step a quadratic model of the cost function is obtained. This model is assumed to be adequate in a region (the trust-region) around the current iterate. Then an update is computed as the minimizer of the model in the trust region. The quality of the updated iterate is evaluated; it is consequently accepted or rejected, and the trust-region radius is adjusted.

3.1. Basic trust-region method. In $\mathbb{R}^n$, given $f : \mathbb{R}^n \to \mathbb{R}$, the trust-region subproblem for finding the update $\eta \in \mathbb{R}^n$ for the current iterate $x_k \in \mathbb{R}^n$ is given by

$$\min_{\eta \in \mathbb{R}^n} m(\eta), \quad m(\eta) = f(x_k) + \partial f(x_k)\eta + \frac{1}{2}\eta^T \partial^2 f(x_k)\eta, \quad \|\eta\| \leq \Delta_k,$$

where $\partial f(x_k)$ is the gradient row vector of the first partial derivatives of $f(x_k)$, $\partial^2 f(x_k)$ is the Hessian matrix of the second partial derivatives of $f(x_k)$, and Δ_k is the trust-region radius. The quality of the model m is evaluated by means of the quotient

$$\rho = \frac{f(x_k) - f(x_k + \eta)}{m(0) - m(\eta)}. \tag{3.1}$$

If ρ is close to 0 or negative, then the model is very inaccurate; i.e., the step must be rejected, and the trust-region radius must be reduced. If ρ is larger but still small, the step is accepted, and the trust-region radius is reduced. Finally, if ρ is close to 1, then there is a good correspondence between the model and the cost function; the step is accepted, and the trust-region radius can be increased.

In a Euclidean space E, if $f : E \to \mathbb{R}$ and $\langle \cdot, \cdot \rangle$ is the inner product in E, then the trust-region subproblem can be formulated as follows without resorting to a basis of E :

(3.2)
$$\begin{aligned} \min_{\eta \in E} m(\eta), \quad m(\eta) &= f(x_k) + \mathrm{D}f(x_k)[\eta] + \tfrac{1}{2}\mathrm{D}^2 f(x_k)[\eta, \eta] \\ &= f(x_k) + \langle \mathrm{grad} f(x_k), \eta \rangle + \tfrac{1}{2}\langle \mathrm{Hess} f(x_k)[\eta], \eta \rangle, \quad \langle \eta, \eta \rangle \leq \Delta_k^2, \end{aligned}$$

where $\mathrm{D}F(x_k)[z]$ denotes the directional derivative of the function F at x_k in the direction of z and $\mathrm{grad} f(x_k)$ and $\mathrm{Hess} f(x_k)$ stand for the gradient and the Hessian of f at x_k, respectively.

3.2. Trust-region method on a Riemannian manifold. Now let the function f be defined on a Riemannian manifold M. In this case, the trust-region subproblem at a point $x_k \in M$ takes place on the tangent space $T_{x_k}M$, which is a Euclidean space. The update vector $\eta \in T_{x_k}M$ is then a tangent vector. Its direction corresponds to the direction in which the next iterate is to be found, and its length indicates how far in this direction to go. However, the new iterate should be on the manifold and not on the tangent space. The actual correspondence between η and the new point on the manifold is given through a smooth mapping, called a retraction. The retraction at point x_k is denoted by R_{x_k}, and its application on a vector $\xi \in T_{x_k}M$ is illustrated in Figure 3.1. The retraction has to satisfy certain conditions as described in [3, 1], but these conditions do not determine a unique function. The so-called exponential map is always an option. There are, however, often computationally better choices depending on the particular problem.

Given a cost function $f : M \to \mathbb{R}$ and a current iterate $x_k \in M$, we look for $x^* \in M$ such that $f(x^*)$ is minimal (see [1]). Using R_{x_k}, the minimization problem for f on M is locally mapped onto a minimization problem on $T_{x_k}M$ for the cost function $\hat{f}_{x_k} : T_{x_k}M \to \mathbb{R}$:

$$\hat{f}_{x_k} : \xi \mapsto f(R_{x_k}(\xi)).$$

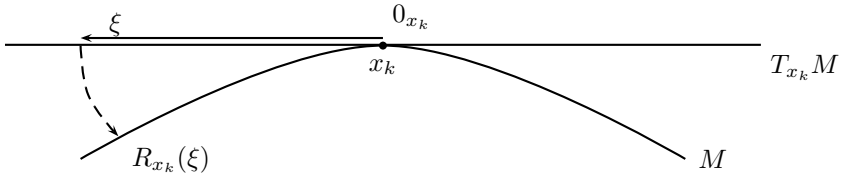


FIG. 3.1. *Retraction* R_{x_k}.

Following (3.2), the trust-region subproblem for $T_{x_k}M$ becomes

$$\min_{\eta\in T_{x_k}M} m_{x_k}(\eta),\ \ m_{x_k}(\eta) = \hat{f}_{x_k}(0_{x_k}) + \mathrm{D}\hat{f}_{x_k}(0_{x_k})[\eta] + \tfrac{1}{2}\mathrm{D}^2\hat{f}_{x_k}(0_{x_k})[\eta,\eta]$$
$$= \hat{f}_{x_k}(0_{x_k}) + \langle \mathrm{grad}\hat{f}_{x_k}(0_{x_k}), \eta\rangle + \tfrac{1}{2}\langle \mathrm{Hess}\hat{f}_{x_k}(0_{x_k})[\eta], \eta\rangle.$$

Alternatively [3, section 7.2.2], we could also solve

$$\min_{\eta\in T_{x_k}M} m_{x_k}(\eta),\ \ m_{x_k}(\eta) = f(x_k) + \langle \mathrm{grad}f(x_k), \eta\rangle + \frac{1}{2}\langle \mathrm{Hess}f(x_k)[\eta], \eta\rangle\,, \tag{3.3}$$

where $\langle \eta, \eta\rangle \leq \Delta_k^2$ and $\mathrm{Hess}f(x_k)$ is the Riemannian Hessian [3].

An approximate but sufficiently accurate solution to the trust-region subproblem is given by the truncated conjugate gradient algorithm (tCG) as suggested in [43, 45]. An advantage of this algorithm is that the Hessian matrix is not required explicitly, but only its application to a tangent vector has to be computed. This is the approach that we will consider. We present the details of tCG in the appendix. Other possible methods for (approximately) solving the trust-region subproblem (3.3) can be found in [30, 9]. The candidate for the new iterate is given by

$$x_+ = R_{x_k}(\eta_k),$$

where η_k is the solution of (3.3). The quotient

$$\rho_k = \frac{f(x_k) - f(R_{x_k}(\eta_k))}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)} = \frac{\hat{f}_{x_k}(0_{x_k}) - \hat{f}_{x_k}(\eta_k)}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)} \tag{3.4}$$

is evaluated in the same way as (3.1), the new iterate is accepted or rejected, and the trust-region radius is updated.

Under mild conditions, the trust-region algorithm converges globally (i.e., for all initial points) to stationary points [1]. Furthermore, since the cost function is monotonically decreasing, it is expected that convergence to a local minimum is achieved in practice. Convergence to a saddle point or a local maximum is only observed in specially designed numerical examples.

4. Riemannian trust-region–based rank-(R_1, R_2, R_3) approximation of a tensor. In this section, we apply the Riemannian trust-region scheme described in section 3 to the problem (2.2). For this purpose, we need to go through the "checklist" in [1, section 5.1] and give closed-form expressions for all the necessary components. Due to an invariance property of $\overline{g}$, we consider a Riemannian quotient manifold on which the computations take place. This will first be explained. Then we present the closed-form expressions for the inner product on the manifold, the retraction function, and the gradient and Hessian of the cost function, as required by [1, section 5.1]. Note that in order to maximize $\overline{g}$ using the trust-region scheme, we, in fact, minimize the function $-\overline{g}$.

4.1. The quotient manifold. For third-order tensors, the cost function $\overline{g}$ from (2.2) can be computed as

$$\begin{aligned}\overline{g}(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \|\mathcal{A} \bullet_1 \mathbf{U}^T \bullet_2 \mathbf{V}^T \bullet_3 \mathbf{W}^T\|^2 &= \|\mathbf{U}^T \mathbf{A}_{(1)}(\mathbf{V} \otimes \mathbf{W})\|^2 \\ &= \|\mathbf{V}^T \mathbf{A}_{(2)}(\mathbf{W} \otimes \mathbf{U})\|^2 \\ &= \|\mathbf{W}^T \mathbf{A}_{(3)}(\mathbf{U} \otimes \mathbf{V})\|^2.\end{aligned} \tag{4.1}$$

It is easily seen that $\overline{g}$ has the following invariance property:

$$\overline{g}(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \overline{g}(\mathbf{U}\mathbf{Q}^{(1)}, \mathbf{V}\mathbf{Q}^{(2)}, \mathbf{W}\mathbf{Q}^{(3)}),$$

where $\mathbf{Q}^{(i)} \in O_{R_i}, i = 1, 2, 3$, are orthogonal matrices. It is thus advisable to work on the manifold $M = St(R_1, I_1)/O_{R_1} \times St(R_2, I_2)/O_{R_2} \times St(R_3, I_3)/O_{R_3}$, which can be thought of as a product of three Grassmann manifolds. The function $\overline{g}$ has one and only one projection $g : M \to \mathbb{R}$:

$$g : (\mathbf{U}O_{R_1}, \mathbf{V}O_{R_2}, \mathbf{W}O_{R_3}) \mapsto \overline{g}(\mathbf{U}, \mathbf{V}, \mathbf{W}). \tag{4.2}$$

Although the elements of M are more difficult to comprehend because they are equivalence classes of matrices, simplifications in the formulas and better convergence results follow, which justify (4.2). It will also be useful to define an extension of $\overline{g}$, $\tilde{g} : \mathbb{R}^{I_1 \times R_1} \times \mathbb{R}^{I_2 \times R_2} \times \mathbb{R}^{I_3 \times R_3} \to \mathbb{R}$,

$$\tilde{g} : (\mathbf{U}, \mathbf{V}, \mathbf{W}) \mapsto \|\mathcal{A} \bullet_1 \mathbf{U}^T \bullet_2 \mathbf{V}^T \bullet_3 \mathbf{W}^T\|^2,$$

the domain of which is a vector space, so that all classical techniques of real analysis are applicable. Another advantage of the latter function is that it is suitable for computer computations since matrices are readily implemented as two-dimensional arrays.

Even though we can work with $\tilde{g}(\mathbf{U}, \mathbf{V}, \mathbf{W})$ as a representation of $g(\mathbf{U}, \mathbf{V}, \mathbf{W})$, we still have to keep in mind the structure of M. For this purpose, some additional notions have to be taken into account (see [3] for details). For simplicity, we first consider a manifold $M_1 = St(p, n)/O_p$. The vertical space at $\mathbf{X} \in St(p, n)$ is defined to be the tangent space at $\mathbf{X}$ to the equivalence class $\mathbf{X}O_p$, i.e.,

$$\mathcal{V}_{\mathbf{X}} = \{\mathbf{X}\mathbf{M} : \mathbf{M} = -\mathbf{M}^T \in \mathbb{R}^{p \times p}\}.$$

The tangent space to $St(p, n)$ at $\mathbf{X}$ is

$$\begin{aligned}T_{\mathbf{X}} St(p, n) &= \{\dot{\mathbf{X}} : \mathbf{X}^T\dot{\mathbf{X}} + \dot{\mathbf{X}}^T\mathbf{X} = 0\} \\ &= \{\mathbf{X}\mathbf{M} + \mathbf{X}_\perp\mathbf{K} : \mathbf{M} = -\mathbf{M}^T \in \mathbb{R}^{p \times p}, \mathbf{K} \in \mathbb{R}^{(n-p) \times p}\},\end{aligned}$$

where $\mathbf{X}_\perp$ is any matrix such that $\begin{bmatrix}\mathbf{X} & \mathbf{X}_\perp\end{bmatrix} \in O_n$. Let $\dot{\mathbf{X}}_1, \dot{\mathbf{X}}_2 \in T_{\mathbf{X}} St(p, n)$. We define an inner product on $T_{\mathbf{X}} St(p, n)$ in the following way:

$$\langle \dot{\mathbf{X}}_1, \dot{\mathbf{X}}_2 \rangle_{\mathbf{X}} = \operatorname{trace}(\dot{\mathbf{X}}_1^T \dot{\mathbf{X}}_2).$$

We choose the horizontal space at $\mathbf{X}$ to be the orthogonal complement of the vertical space in $T_{\mathbf{X}} St(p, n)$:

$$\mathcal{H}_{\mathbf{X}} = \{\mathbf{X}_\perp \mathbf{K} : \mathbf{K} \in \mathbb{R}^{(n-p)\times p}\}.$$

Given $\mathbf{Z}_{\mathbf{X}O_p} \in T_{\mathbf{X}O_p} M_1$, there exists a unique $\overline{\mathbf{Z}}_{\mathbf{X}} \in \mathcal{H}_{\mathbf{X}}$, such that $\mathrm{D}\pi(\mathbf{X})[\overline{\mathbf{Z}}_{\mathbf{X}}] = \mathbf{Z}_{\mathbf{X}O_p}$, where

$$\begin{aligned} \pi : St(p,n) &\to M_1 \\ \mathbf{X} &\mapsto \mathbf{X}O_p \end{aligned}$$

is the quotient map. $\overline{\mathbf{Z}}_{\mathbf{X}}$ is termed the horizontal lift of $\mathbf{Z}_{\mathbf{X}O_p}$ at $\mathbf{X}$. It can be shown that $\overline{\mathbf{Z}}_{\mathbf{XQ}} = \overline{\mathbf{Z}}_{\mathbf{X}}\mathbf{Q}$ for all $\mathbf{Q} \in O_p$. Observe that $\langle \overline{\mathbf{Z}}^{(1)}_{\mathbf{XQ}}, \overline{\mathbf{Z}}^{(2)}_{\mathbf{XQ}} \rangle = \langle \overline{\mathbf{Z}}^{(1)}_{\mathbf{X}}, \overline{\mathbf{Z}}^{(2)}_{\mathbf{X}} \rangle$; hence $\langle\langle \mathbf{Z}^{(1)}_{\mathbf{X}O_p}, \mathbf{Z}^{(2)}_{\mathbf{X}O_p} \rangle\rangle = \langle \overline{\mathbf{Z}}^{(1)}_{\mathbf{X}}, \overline{\mathbf{Z}}^{(2)}_{\mathbf{X}} \rangle$ is a well-defined inner product on $T_{\mathbf{X}O_p} M_1$. Finally, the orthogonal projection onto the horizontal space at $\mathbf{X}$ is given by

$$P_{\mathbf{X}}(\dot{\mathbf{X}}) = (\mathbf{I} - \mathbf{X}\mathbf{X}^T)\dot{\mathbf{X}}. \tag{4.3}$$

The elements of the vertical space can be thought of as variations that do not modify the column space of $\mathbf{X}$, whereas the elements of the horizontal space, applied to $\mathbf{X}$, modify the span of $\mathbf{X}$. The projection (4.3) reflects the fact that only variations of $\mathbf{X}$ that are orthogonal to the column space of $\mathbf{X}$ are important. Any other variations are irrelevant, since they are variations in the same equivalence class, so they do not affect points in $St(p,n)/O_p$.

For the more general manifold M, we apply the following orthogonal projection:

$$P_{(\mathbf{U},\mathbf{V},\mathbf{W})}(\dot{\mathbf{U}}, \dot{\mathbf{V}}, \dot{\mathbf{W}}) = ((\mathbf{I} - \mathbf{U}\mathbf{U}^T)\dot{\mathbf{U}}, (\mathbf{I} - \mathbf{V}\mathbf{V}^T)\dot{\mathbf{V}}, (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\dot{\mathbf{W}}). \tag{4.4}$$

The inner product on the horizontal space $\mathcal{H}_{(\mathbf{U},\mathbf{V},\mathbf{W})}$ is then defined as the sum of the inner products of its components. A *retraction* R defines a unique point on the manifold corresponding to a given tangent vector $(\mathbf{Z}_{\mathbf{U}O_{R_1}}, \mathbf{Z}_{\mathbf{V}O_{R_2}}, \mathbf{Z}_{\mathbf{W}O_{R_3}})$ at a point $(\mathbf{U}O_{R_1}, \mathbf{V}O_{R_2}, \mathbf{W}O_{R_3})$. We use the following definition [3, section 4.1.2]:

$$\begin{aligned} &R_{(\mathbf{U}O_{R_1},\mathbf{V}O_{R_2},\mathbf{W}O_{R_3})}(\mathbf{Z}_{\mathbf{U}O_{R_1}}, \mathbf{Z}_{\mathbf{V}O_{R_2}}, \mathbf{Z}_{\mathbf{W}O_{R_3}}) \\ &= (\mathrm{qf}(\mathbf{U} + \overline{\mathbf{Z}}_{\mathbf{U}})O_{R_1}, \mathrm{qf}(\mathbf{V} + \overline{\mathbf{Z}}_{\mathbf{V}})O_{R_2}, \mathrm{qf}(\mathbf{W} + \overline{\mathbf{Z}}_{\mathbf{W}})O_{R_3}), \end{aligned} \tag{4.5}$$

where qf denotes the Q factor of the thin QR decomposition [18]. As mentioned in the previous section, other possibilities exist for the retraction function. However, our choice is computationally efficient and motivated by the fact that we are only interested in column spaces of the matrices $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$ and not in their actual values.

Finally, closed-form expressions for the gradient $\mathrm{grad}\, g$ and the Hessian $\mathrm{Hess}\, g$ are obtained in the next two subsections.

From now on, we abuse notation and use g for $\overline{g}$, $\mathbf{X}$ for $\mathbf{X}O_p$, $\mathbf{Z}_{\mathbf{X}}$ for $\overline{\mathbf{Z}}_{\mathbf{X}}$, and $\langle \cdot, \cdot \rangle$ for $\langle\langle \cdot, \cdot \rangle\rangle$. This is admissible since $g(\mathbf{X}O_p) = \overline{g}(\mathbf{X})$ and $\langle\langle \mathbf{Z}^{(1)}_{\mathbf{X}O_p}, \mathbf{Z}^{(2)}_{\mathbf{X}O_p} \rangle\rangle = \langle \overline{\mathbf{Z}}^{(1)}_{\mathbf{X}}, \overline{\mathbf{Z}}^{(2)}_{\mathbf{X}} \rangle$.

4.2. Computation of the gradient. We have the following link between the *gradient* of g and the gradient of $\tilde{g}$ [3, equation (3.39)]:

$$\mathrm{grad} g(\mathbf{U}, \mathbf{V}, \mathbf{W}) = P_{(\mathbf{U},\mathbf{V},\mathbf{W})} \mathrm{grad} \tilde{g}(\mathbf{U}, \mathbf{V}, \mathbf{W}), \tag{4.6}$$

where $P_{(\mathbf{U},\mathbf{V},\mathbf{W})}(\mathbf{Z_U}, \mathbf{Z_V}, \mathbf{Z_W})$ projects the columns of $\mathbf{Z_U}, \mathbf{Z_V}$, and $\mathbf{Z_W}$ onto the orthogonal complements of the column spaces of $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$, respectively. Next, based on (4.1), we derive the differential of $\tilde{g}$ as

$$\begin{aligned} \mathrm{D}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] &= 2\mathrm{trace}(\mathbf{Z_U}^T \mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{U}) \\ &\quad + 2\mathrm{trace}(\mathbf{Z_V}^T \mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{U})(\mathbf{W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{V}) \\ &\quad + 2\mathrm{trace}(\mathbf{Z_W}^T \mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{V})(\mathbf{U}\otimes\mathbf{V})^T(\mathbf{A}_{(3)})^T\mathbf{W}). \end{aligned}$$

The formula for the gradient of $\tilde{g}$ follows directly from the latter expression, namely,

$$\begin{aligned} \mathrm{grad}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W}) &= (\mathrm{grad}_{\mathbf{U}}\tilde{g}, \mathrm{grad}_{\mathbf{V}}\tilde{g}, \mathrm{grad}_{\mathbf{W}}\tilde{g}) \\ &= (2\mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{U}, \\ &\quad 2\mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{U})(\mathbf{W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{V}, \\ &\quad 2\mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{V})(\mathbf{U}\otimes\mathbf{V})^T\ (\mathbf{A}_{(3)})^T\mathbf{W}). \end{aligned} \tag{4.7}$$

Combining (4.6), (4.4), and (4.7), a closed-form expression for the gradient of g is obtained.

4.3. Computation of the Hessian. A starting point for computing the *Hessian* of g is the following formula [3, equation (7.2) and Proposition 5.3.4]:

$$\mathrm{Hess}\ g(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] = P_{(\mathbf{U},\mathbf{V},\mathbf{W})}\mathrm{D}(\mathrm{grad}g)(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})]. \tag{4.8}$$

Taking into account (4.6) and (4.4), we can transform the first component of $\mathrm{D}(\mathrm{grad}g)(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})]$ to

$$\begin{aligned} &\mathrm{D}((\mathbf{I}-\mathbf{U}\mathbf{U}^T)\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] \\ &= \mathrm{D}(\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] - \mathrm{D}(\mathbf{U}\mathbf{U}^T\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] \\ &= \mathrm{D}(\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] - \mathbf{Z_U}\mathbf{U}^T\mathrm{grad}_{\mathbf{U}}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W}) \\ &\quad -\mathbf{U}\mathrm{D}\{\mathbf{U}^T\mathrm{grad}_{\mathbf{U}}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W})\}[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})]\,. \end{aligned}$$

Similar expressions are obtained for the other two components. Substituting these expressions in (4.8), using (4.4) and the equality $(\mathbf{I}-\mathbf{U}\mathbf{U}^T)\mathbf{U} = 0$, (4.8) is simplified to

$$\begin{aligned} &\mathrm{Hess}\ g(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] = \\ &\Big((\mathbf{I}-\mathbf{U}\mathbf{U}^T)\Big(\mathrm{D}(\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] - \mathbf{Z_U}\mathbf{U}^T\mathrm{grad}_{\mathbf{U}}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W})\Big), \\ &\ (\mathbf{I}-\mathbf{V}\mathbf{V}^T)\Big(\mathrm{D}(\mathrm{grad}_{\mathbf{V}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] - \mathbf{Z_V}\mathbf{V}^T\mathrm{grad}_{\mathbf{V}}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W})\Big), \\ &(\mathbf{I}-\mathbf{W}\mathbf{W}^T)\Big(\mathrm{D}(\mathrm{grad}_{\mathbf{W}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] - \mathbf{Z_W}\mathbf{W}^T\mathrm{grad}_{\mathbf{W}}\tilde{g}(\mathbf{U},\mathbf{V},\mathbf{W})\Big)\Big). \end{aligned} \tag{4.9}$$

The only quantities in (4.9) that still lack a closed-form expression are of the form $\mathrm{D}(\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})]$. From (4.7) we have

$$\begin{aligned} \mathrm{D}(\mathrm{grad}_{\mathbf{U}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] &= 2\mathbf{A}_{(1)}(\mathbf{Z_V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{U} \\ &\quad + 2\mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{Z_W})(\mathbf{V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{U} \\ &\quad + 2\mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{W})(\mathbf{Z_V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{U} \\ &\quad + 2\mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{Z_W})^T(\mathbf{A}_{(1)})^T\mathbf{U} \\ &\quad + 2\mathbf{A}_{(1)}(\mathbf{V}\otimes\mathbf{W})(\mathbf{V}\otimes\mathbf{W})^T(\mathbf{A}_{(1)})^T\mathbf{Z_U}\,. \end{aligned}$$

By analogy,

$$\begin{aligned}\mathrm{D}(\mathrm{grad}_{\mathbf{V}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] &= 2\mathbf{A}_{(2)}(\mathbf{Z_W}\otimes\mathbf{U})(\mathbf{W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{V}\\ &\quad+2\mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{Z_U})(\mathbf{W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{V}\\ &\quad+2\mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{U})(\mathbf{Z_W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{V}\\ &\quad+2\mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{U})(\mathbf{W}\otimes\mathbf{Z_U})^T(\mathbf{A}_{(2)})^T\mathbf{V}\\ &\quad+2\mathbf{A}_{(2)}(\mathbf{W}\otimes\mathbf{U})(\mathbf{W}\otimes\mathbf{U})^T(\mathbf{A}_{(2)})^T\mathbf{Z_V}\end{aligned}$$

and

$$\begin{aligned}\mathrm{D}(\mathrm{grad}_{\mathbf{W}}\tilde{g})(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})] &= 2\mathbf{A}_{(3)}(\mathbf{Z_U}\otimes\mathbf{V})(\mathbf{U}\otimes\mathbf{V})^T(\mathbf{A}_{(3)})^T\mathbf{W}\\ &\quad+2\mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{Z_V})(\mathbf{U}\otimes\mathbf{V})^T(\mathbf{A}_{(3)})^T\mathbf{W}\\ &\quad+2\mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{V})(\mathbf{Z_U}\otimes\mathbf{V})^T(\mathbf{A}_{(3)})^T\mathbf{W}\\ &\quad+2\mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{V})(\mathbf{U}\otimes\mathbf{Z_V})^T(\mathbf{A}_{(3)})^T\mathbf{W}\\ &\quad+2\mathbf{A}_{(3)}(\mathbf{U}\otimes\mathbf{V})(\mathbf{U}\otimes\mathbf{V})^T(\mathbf{A}_{(3)})^T\mathbf{Z_W}\ .\end{aligned}$$

A closed-form expression for $\mathrm{Hess}\, g(\mathbf{U},\mathbf{V},\mathbf{W})[(\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})]$ is derived by substituting the last three expressions and (4.7) in (4.9).

4.4. Summary of the algorithm. In this subsection, we present a summary of the whole algorithm and two theorems concerning its global and local convergence properties.

In order to simplify the presentation, let $\mathbf{X} = (\mathbf{U},\mathbf{V},\mathbf{W})$ and $\mathbf{Z} = (\mathbf{Z_U},\mathbf{Z_V},\mathbf{Z_W})$. Note that in practice, instead of minimizing (2.1) we maximize (2.2), or even more precisely, we minimize the function $-g$. The computational steps are presented in Algorithm 1.

Based on [1, Theorems 4.4 and 4.13], we can state the following global and local convergence theorems.

THEOREM 4.1 (global convergence). *Let $\{\mathbf{X}_k\}$ be a sequence of iterates $\mathbf{X}_k = (\mathbf{U}_k,\mathbf{V}_k,\mathbf{W}_k)$ generated by Algorithm 1, and let an iterate be accepted if $\rho_k > \rho'$ with $\rho' \in (0,\frac{1}{4})$. Then*

$$\lim_{k\to\infty} \mathrm{grad}\, g(\mathbf{X}_k) = 0.$$

In other words, the algorithm converges to stationary points. Moreover, since g is monotonically increasing ($-g$ is monotonically decreasing), the cost function (2.1) is monotonically decreasing, and thus convergence to local minima of (2.1) is expected. Convergence to saddle points or local maxima could be observed only in specially constructed examples.

THEOREM 4.2 (local convergence speed). *Consider Algorithms 1–2 with retraction R as in (4.5) and stopping criterion in Algorithm 2 (see Appendix A for Algorithm 2) as in (5.1). Let $\mathbf{X}^* = (\mathbf{U}^*,\mathbf{V}^*,\mathbf{W}^*) \in M$ be a (nondegenerate) local maximum of g (local minimum of $-g$). Then there exists $c > 0$ such that for all sequences $\{\mathbf{X}_k\}$ generated by the algorithm converging to $\mathbf{X}^*$ there exists $K > 0$ such that for all $k > K$,*

$$\mathrm{dist}(\mathbf{X}_{k+1},\mathbf{X}^*) \leq c\,(\mathrm{dist}(\mathbf{X}_k,\mathbf{X}^*))^{\min\{\theta+1,2\}},$$

with $\theta > 0$ as in (5.1).

Note that for our numerical experiments in sections 5 and 6 we use $\theta = 1$, which yields quadratic convergence.

ALGORITHM 1 TRUST-REGION ALGORITHM FOR MINIMIZING (2.1).

Input: Higher-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and R_1, R_2, R_3.

Output: Projection matrices $\mathbf{U} \in St(R_1, I_1), \mathbf{V} \in St(R_2, I_2), \mathbf{W} \in St(R_3, I_3)$ such that the rank-(R_1, R_2, R_3) approximation $\hat{\mathcal{A}} = \mathcal{A} \bullet_1 (\mathbf{U}\mathbf{U}^T) \bullet_2 (\mathbf{V}\mathbf{V}^T) \bullet_3 (\mathbf{W}\mathbf{W}^T)$ corresponds to a local minimum of (2.1).

1: Obtain initial iterate $\mathbf{X}_0 \in M$, e.g., using the truncated HOSVD from section 2.3. Set initial value for Δ_0.

2: **for** $k = 0, 1, 2, \ldots$ **do**

3: $m_{\mathbf{X}_k}(\mathbf{Z}) = -g(\mathbf{X}_k) + \langle -\operatorname{grad} g(\mathbf{X}_k), \mathbf{Z}\rangle + \frac{1}{2}\langle -\operatorname{Hess} g(\mathbf{X}_k)[\mathbf{Z}], \mathbf{Z}\rangle$.

4: Solve for $\mathbf{Z}_k$ the trust-region subproblem

$$\min_{\mathbf{Z} \in T_{\mathbf{X}_k} M} m_{\mathbf{X}_k}(\mathbf{Z}) \quad \text{subject to } \langle \mathbf{Z}, \mathbf{Z}\rangle \leq \Delta_k^2.$$

An approximate solution is given by the tCG algorithm (Algorithm 2).

5: Form the quotient

$$\rho_k = \frac{-g(\mathbf{X}_k) + g(R_{\mathbf{X}_k}(\mathbf{Z}_k))}{m_{\mathbf{X}_k}(0_{\mathbf{X}_k}) - m_{\mathbf{X}_k}(\mathbf{Z}_k)}.$$

6: **if** ρ_k is large enough, $\mathbf{X}_{k+1} = R_{\mathbf{X}_k}(\mathbf{Z}_k)$; **else** $\mathbf{X}_{k+1} = \mathbf{X}_k$.

7: **if** ρ_k is too small, $\Delta_{k+1} = \frac{1}{4}\Delta_k$; **else if** ρ_k is large, $\Delta_{k+1} = 2\Delta_k$; **else** $\Delta_{k+1} = \Delta_k$.

8: **end for**

9: Set $\mathbf{U} = \mathbf{U}_{k+1}$, $\mathbf{V} = \mathbf{V}_{k+1}$, $\mathbf{W} = \mathbf{W}_{k+1}$.

10: The low multilinear rank approximation $\hat{\mathcal{A}}$ is given by

$$\hat{\mathcal{A}} = \mathcal{A} \bullet_1 (\mathbf{U}\mathbf{U}^T) \bullet_2 (\mathbf{V}\mathbf{V}^T) \bullet_3 (\mathbf{W}\mathbf{W}^T).$$

5. Comparison with HOOI. In this section, we perform numerical experiments illustrating the properties of the proposed algorithm. All experiments are carried out in parallel for the trust-region and HOOI algorithms. The simulations were run using MATLAB. The machine precision is approximately $2 * 10^{-16}$.

5.1. Convergence speed. We start our comparison with the convergence speed of both algorithms. HOOI has only linear convergence, whereas the local rate of convergence of the trust-region–based algorithm is superlinear [1]. This improvement in the trust-region–based algorithm is due to using second order information of the cost function. For illustration, we considered tensors $\mathcal{A} \in \mathbb{R}^{20 \times 20 \times 20}$ with elements uniformly distributed in the interval $[0, 1]$. We set $R_1 = R_2 = R_3 = 5$. Initial matrices were taken from the truncated HOSVD. Then both algorithms were run until convergence ($\|\operatorname{grad} g(\mathbf{X})\|/g(\mathbf{X}) < 10^{-9}$) or until a maximum of 100 iterations was reached. In Figure 5.1, the results of this experiment are shown for one representative example. Three stages of the trust-region algorithm evolution can be distinguished. Due to a large initial trust-region radius, a poor correspondence between the cost function and the quadratic model was obtained for the first two updates, and as a result they were rejected. In the plots this corresponds to constant values of the cost function and of the relative norm of the gradient. After reducing the trust-region radius, the values of the quotient (3.4) considerably improved, so the majority of the following steps were successful. As a consequence, during this second phase, the value of the cost

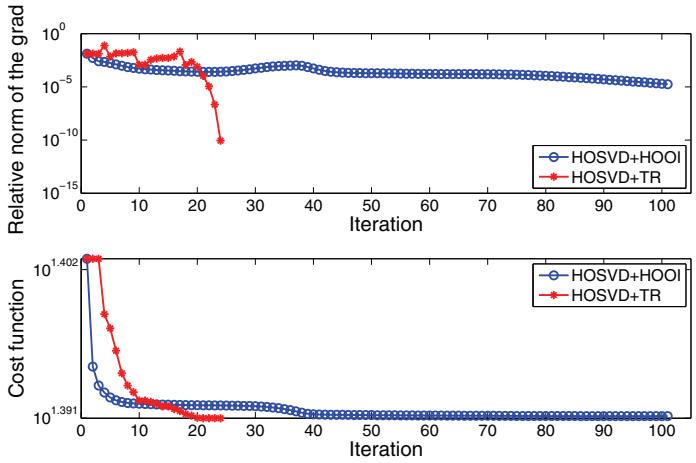
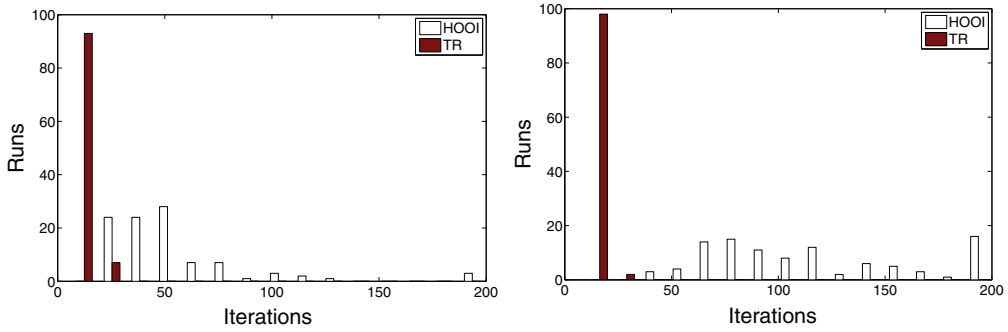


FIG. 5.1. *Evolution of the relative norm of the gradient* $\|grad\ g(\mathbf{X})\|/g(\mathbf{X})$ *and the cost function* $\|\mathcal{A} - \hat{\mathcal{A}}\|$. *The tensor* $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ *has elements uniformly distributed in the interval* $[0, 1]$*;* $R_1 = R_2 = R_3 = 5$. *The starting values are taken from the truncated HOSVD.*



(a) $(R_1, R_2, R_3) = (7, 8, 9)$ (b) $(R_1, R_2, R_3) = (2, 2, 2)$

FIG. 5.2. *Number of iterations for HOOI and trust-region algorithms for tensors* $\mathcal{A} \in \mathbb{R}^{10\times 10\times 10}$ *with elements taken from a normal distribution with zero mean and unit standard deviation. Performed are* 100 *runs with initial values* $\mathbf{U}_0, \mathbf{V}_0$, *and* $\mathbf{W}_0$ *taken from the truncated HOSVD. A run was stopped if the corresponding algorithm did not converge in* 200 *iterations* ($\|\text{grad}\ g(\mathbf{X})\|/g(\mathbf{X}) < 10^{-9}$).

function decreased monotonically, as can be seen from the bottom plot of Figure 5.1. The relative norm of the gradient did not decrease monotonically because of the complex nature of the cost function. However, its decrease is not required in order to improve the cost function value. Finally, in the last steps, the iterates were close enough to the solution for superlinear convergence, as can be observed in the gradient plot.

The number of iterations necessary for the algorithms to converge is strongly related to the convergence speed. In Figure 5.2, we compare the two algorithms for tensors $\mathcal{A} \in \mathbb{R}^{10\times 10\times 10}$ with elements taken from a normal distribution with zero mean and unit standard deviation and for two multilinear ranks $(7, 8, 9)$ and $(2, 2, 2)$, respectively. The results shown are for 100 runs, and initial values were taken from the truncated HOSVD. Both algorithms were run until convergence ($\|\text{grad}\ g(\mathbf{X})\|/g(\mathbf{X}) <$

10^{-9}) or until a maximum of 200 iterations was reached. The trust-region algorithm did not reach the maximum number of iterations in either test. HOOI had three runs in (a) and 13 runs in (b) with 200 iterations. In order to converge, the trust-region–based algorithm tends to require less iterations than HOOI. The difference seems to be stronger for smaller multilinear rank.

5.2. Computational cost. While the trust-region method converges faster than HOOI, the cost for one iteration is higher than the cost for one HOOI iteration, which is $O(I^3R + IR^4 + R^6)$ as shown in [17, 25]. For simplicity, we assume that $R_1 = R_2 = R_3 = R$ and $I_1 = I_2 = I_3 = I$.

In order to compute the cost for one trust-region iteration, we need to estimate the cost of one inner tCG iteration and multiply it by the number of tCG iterations within one trust-region step. The cost for the operations outside the inner tCG loop is smaller and can be neglected. As we have already mentioned, using a simple retraction as in (4.5) is in general faster than using the exponential map. However, the cost of computing any of them is low in comparison with the computation of the Hessian, so it can be neglected. The computationally heaviest operation in the trust-region algorithm is the computation of the Hessian or, more precisely, the application of the Hessian on a tangent vector. This is dominated by expressions such as $\mathbf{A}_{(1)}(\mathbf{Z_V} \otimes \mathbf{W})$. Another representation of these expressions is $(\mathcal{A} \bullet_2 \mathbf{Z}_\mathbf{V}^T) \bullet_3 \mathbf{W}^T = (\mathcal{A} \bullet_3 \mathbf{W}^T) \bullet_2 \mathbf{Z}_\mathbf{V}^T$, which requires $O(I^3R)$ flops. Such expressions are computed in each inner iteration, i.e., in each step of tCG. However, the products $\mathcal{A}\bullet_1 \mathbf{U}^T$, $\mathcal{A}\bullet_2 \mathbf{V}^T$, $\mathcal{A}\bullet_3 \mathbf{W}^T$, $\mathcal{A}\bullet_2 \mathbf{V}^T \bullet_3 \mathbf{W}^T$, etc., which involve only the tensor $\mathcal{A}$ and the matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ and which do not involve the components $\mathbf{Z_U}, \mathbf{Z_V}, \mathbf{Z_W}$ of tangent vectors, can be computed at the beginning of each trust-region step. They require $O(I^3R)$ flops. Thus, within the inner iterations only expressions of the form $(\mathcal{A}\bullet_3 \mathbf{W}^T)\bullet_2 \mathbf{Z}_\mathbf{V}^T$, given $\mathcal{A}\bullet_3 \mathbf{W}^T$, are computed. The cost of such an operation is only $O(I^2R^2)$. Finally, the maximum number of inner iterations per trust-region iteration is $3(I - R)R$, which equals the dimension of the manifold. This leads us to a total cost of $O(I^3R)+3(I-R)R*O(I^2R^2) = O(I^3R^3)$ for one trust-region step. It should be taken into account that in applications, the values of R_n are often much smaller than the corresponding values of I_n. Interestingly, the case of Nth-order tensors with $N > 3$ is more favorable than one would think. The corresponding computational cost is $O(I^NR) + N(I - R)R * O(I^2R^{N-1}) = O(I^NR + NI^3R^N)$. This is approximately of the same order as the cost of one HOOI iteration $O(I^NR + NIR^{2(N-1)} + NR^{3(N-1)})$.

Note, however, that one can reduce the computational cost of the trust-region algorithm without losing its fast local convergence rate. This can be done by choosing a stopping criterion for the inner iteration (Algorithm 2) based on the gradient of the cost function as in [1]. In this case, few inner tCG steps are taken when the outer iterate is far away from the solution, i.e., when the gradient is large, and more inner tCG steps are taken close to the solution. This behavior is illustrated in Figure 5.3. The tCG inner iteration is stopped according to [1, section 3]

$$\|r_j\| \leq \|r_0\| \min(\|r_0\|^\theta, \kappa), \tag{5.1}$$

where r_j is the residual at the jth step of tCG. We chose $\theta = 1$, which yields quadratic convergence of the trust-region algorithm. We used this inner stopping criterion for all our numerical experiments involving the trust-region method. For Figure 5.3, the entries of $\mathcal{A} \in \mathbb{R}^{20\times20\times20}$ are taken from a normal distribution with zero mean and unit standard deviation. We set $R_1 = R_2 = R_3 = 2$, and $\mathbf{U}_0, \mathbf{V}_0$, and $\mathbf{W}_0$ are obtained from the truncated HOSVD. The algorithm was stopped when $\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X}) < 10^{-9}$.

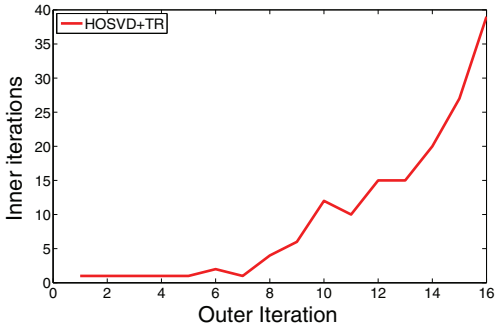


FIG. 5.3. *Number of inner tCG steps per trust-region step. The tensor* $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ *has elements taken from a normal distribution with zero mean and unit standard deviation;* $R_1 = R_2 = R_3 = 2$ *and* $\mathbf{U}_0, \mathbf{V}_0$, *and* $\mathbf{W}_0$ *are taken from the truncated HOSVD. Few tCG iterations are performed in the first trust-region steps and more in the neighborhood of the solution.*

Large scale problems present new types of difficulties. Since the Hessian has to be computed (to be applied to vectors) at each step, the computational cost of the trust-region iterates might become excessive in comparison with the lighter-weight (but usually slower-converging) HOOI. Nontrivial modifications are necessary. An option might be to consider a type of hierarchical Tucker format. As an initial step, the HOSVD of the original tensor can be truncated so that the smallest mode-n singular values be discarded. In this way, the dimensions of the original tensor are reduced without losing much precision. As a second step, an essential low multilinear rank approximation on the already smaller scale can be performed.

6. Comparison with Newton-type methods. Besides HOOI, Newton-type methods for the best rank-(R_1, R_2, R_3) approximation have appeared in the literature [25, 17, 39]. Pure Newton methods have a local quadratic convergence rate, and their computational cost per iteration is of the same order as the one of the trust-region method. However, they are not globally convergent and depend strongly on the initial values of the matrices $\mathbf{U}, \mathbf{V}$, and $\mathbf{W}$. In practice, these methods might diverge. The truncated HOSVD often gives good initial values, but sometimes these values are not good enough. On the other hand, the trust-region method converges globally to local optima (Theorem 4.1) except for very special examples that are artificially constructed.

Another advantage of the trust-region method is that the cost function is monotonically decreasing. This explains why convergence to saddle points or local maxima is not observed in practice. Pure Newton methods do not distinguish between minima, maxima, and saddle points. This means that if the stationary points are close to each other, even if a relatively good starting point is chosen, these algorithms might converge to a maximum or to a saddle point instead of to a minimum.

Quasi-Newton methods have cheaper iterations but need considerably more iterations in order to converge. When enhanced with an adequate globalization strategy (e.g., a line-search procedure and a safeguard that ensures that the approximate Hessian stays "safely" positive-definite), quasi-Newton methods converge globally to stationary points of the cost function. Otherwise, they can diverge. A strength of the trust-region over the quasi-Newton algorithms [39] is the more detailed convergence analysis from [1].

To illustrate the above, we considered tensors $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ with elements uniformly distributed in the interval $[0, 1]$ and estimated their best rank-$(5, 5, 5)$ approximation. We started from values taken from the truncated HOSVD and performed 20 iterations of HOOI. Then we ran HOOI, the geometric Newton [25], and the trust-region algorithms until convergence, with a maximum of 500 iterations. In Figure 6.1, one example is shown where HOOI and the trust-region algorithm both converge to a local minimum of (2.1), but the geometric Newton algorithm converges to a saddle point. For clarity, the figure presents only the iterations in the interval $[20, 100]$. The type of the end points was evaluated using the Hessian of the cost function. Cases like this seem rather rare. Often, the Newton-type and the trust-region algorithms perform in a similar way in the same setting, i.e., converge to a local minimum with a similar number of iterations. In our simulations, it also happened that both types of algorithms converged to a local minimum but that the Newton-type algorithm needed considerably more iterations. This occurred especially when the starting point of the Newton algorithm had an indefinite Hessian.

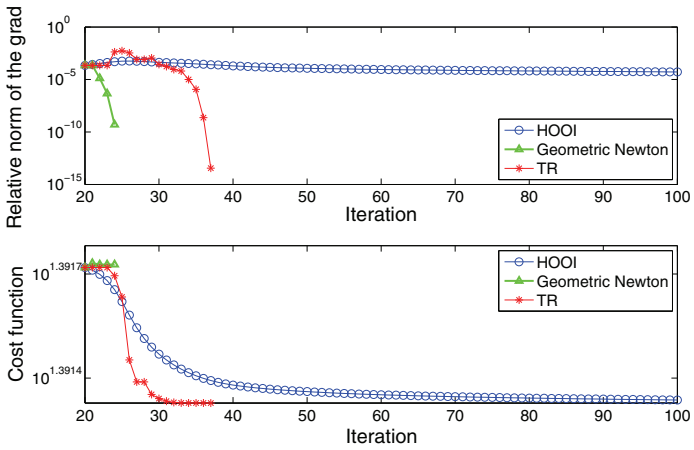


FIG. 6.1. *Evolution of the relative norm of the gradient* $\|grad\ g(\mathbf{X})\|/g(\mathbf{X})$ *and the cost function* $\|\mathcal{A} - \hat{\mathcal{A}}\|$. *The tensor* $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ *has elements uniformly distributed in the interval* $[0, 1]$*;* $R_1 = R_2 = R_3 = 5$. *The starting point of the plots is obtained by initializing with matrices taken from the truncated HOSVD and performing* 20 *additional HOOI iterations.*

In tests with low multilinear rank tensors affected by additive noise with a modest signal-to-noise ratio, the performance of the Newton-type and trust-region algorithms was similar. We considered rank-$(5, 5, 5)$ tensors $\mathcal{T} \in \mathbb{R}^{20\times 20\times 20}$, affected by additive noise in the following way:

$$\mathcal{A} = \mathcal{T}/\|\mathcal{T}\| + 0.1\ \mathcal{E}/\|\mathcal{E}\|, \tag{6.1}$$

in which $\mathcal{E} \in \mathbb{R}^{20\times 20\times 20}$ represents noise with elements uniformly distributed in the interval $[0, 1]$. The tensor $\mathcal{T}$ was constructed as the product of a tensor $\mathcal{C} \in \mathbb{R}^{5\times 5\times 5}$ and three matrices $\mathbf{M}^{(i)} \in \mathbb{R}^{20\times 5}$, $i = 1, 2, 3$, all with elements uniformly distributed in the interval $[0, 1]$, i.e.,

$$\mathcal{T} = \mathcal{C} \bullet_1 \mathbf{M}^{(1)} \bullet_2 \mathbf{M}^{(2)} \bullet_3 \mathbf{M}^{(3)}.$$

In Figure 6.2, one representative example is given. The initial matrices $\mathbf{U}_0$, $\mathbf{V}_0$, and $\mathbf{W}_0$ were taken from the truncated HOSVD and 5 additional HOOI iterations were performed. As can be seen from the Figure 6.2, the differential-geometric Newton method and the trust-region algorithm have similar convergence behavior, whereas HOOI needs more but computationally less expensive iterations.

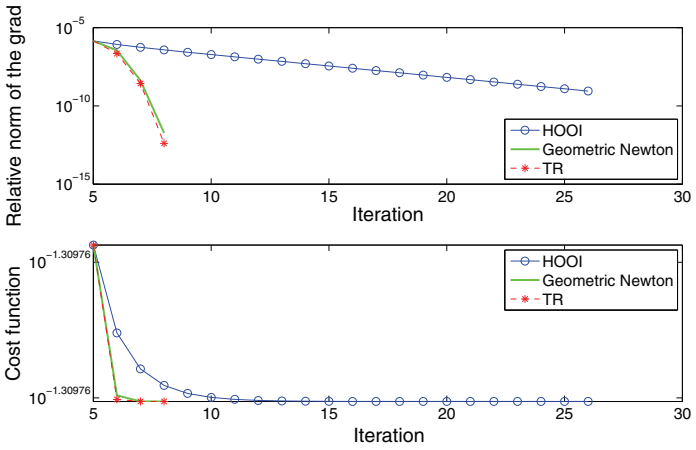


FIG. 6.2. *Evolution of the relative norm of the gradient* $\|grad\ g(\mathbf{X})\|/g(\mathbf{X})$ *and the cost function* $\|\mathcal{A} - \hat{\mathcal{A}}\|$. *The tensor* $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ *is as in* (6.1), *with* $R_1 = R_2 = R_3 = 5$. *The starting point of the plots is obtained by initializing with matrices taken from the truncated HOSVD and performing* 5 *additional HOOI iterations.*

We conclude this part by reporting that in our simulations the Newton methods in [25, 17] generally behaved in the same way, although for a particular dataset the behavior was sometimes different.

Concerning the computational time necessary for the algorithms to converge, it is difficult to make fair comparisons. This is because the algorithms depend on various adjustable parameters, have different stopping criteria, and may have suboptimal implementations. The size of the problem, the required precision, and the type of tensor to be approximated are also important factors affecting the comparisons. It is of course possible to unify the stopping criteria and use, for example, the relative norm of the gradient for all algorithms. However, for HOOI this would not be an optimal strategy, since HOOI does not need the computation of the gradient in order to converge. On the other hand, the computation of HOOI is based on SVDs, and there is a MATLAB built-in function for this purpose. This gives HOOI an advantage over the other methods that do not use such efficient functions. We stress the fact that our time comparisons should be interpreted as indications only. For the Newton-type algorithms we used the code accompanying [39, 17],[2,3] which also includes an HOOI implementation. The implementation of the trust-region algorithm is based on the GenRTR package mentioned in section 1.

We performed two types of experiments. First, we considered tensors $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ with elements taken from a normal distribution with zero mean and unit

[2]Available at http://www.mai.liu.se/∼besav/soft.html and based on the tensor toolbox http://csmr.ca.sandia.gov/∼tgkolda/TensorToolbox/.

[3]We have, however, removed the printing on the screen at each iteration since this can affect considerably the results.

standard deviation. The imposed multilinear rank was $(5, 5, 5)$. The initial values $\mathbf{U}_0, \mathbf{V}_0$, and $\mathbf{W}_0$ were obtained by first computing the truncated HOSVD and then running 20 HOOI iterations. This is done in order to obtain a good starting point, which is necessary for the pure-Newton algorithms. The relative norm of the gradient ($\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X})$) at point $(\mathbf{U}_0, \mathbf{V}_0, \mathbf{W}_0)$ was then around 0.01. A run was stopped if the corresponding algorithm converged ($\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X}) < 5 * 10^{-13}$) or reached the maximum number of iterations, which was 50 for the pure-Newton and 300 for all other algorithms. In this experiment, the pure-Newton algorithm often reached the maximum number of iterations. The fastest was usually the trust-region algorithm. The quasi-Newton algorithm in local coordinates was either the second fastest or reached the maximum number of iterations. In the latter case, the second fastest was HOOI, although it also often reached the maximal number of iterations. The exact timings for two representative runs are given in Table 6.1. The simulations were run on a 4-CPU PC (Intel Xeon X5355 2.66 GHz) with 12GB RAM.

TABLE 6.1

Time comparisons (in seconds) for tensors $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ with elements taken from a normal distribution with zero mean and unit standard deviation for all algorithms considered in this paper. The imposed multilinear rank was $(5, 5, 5)$. The initial values $\mathbf{U}_0, \mathbf{V}_0$, and $\mathbf{W}_0$ were obtained by first computing the truncated HOSVD and then running 20 HOOI iterations. A run was stopped if the corresponding algorithm converged ($\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X}) < 5 * 10^{-13}$) or reached the maximum number of iterations, which was 50 for the pure-Newton and 300 for all other algorithms.

Algorithm	Case 1	Case 2
Quasi-Newton (global coordinates)	3.8173	5.5414
Quasi-Newton (local coordinates)	1.2673	9.9610
Pure Newton (global coordinates)	9.5830	10.8236
Pure Newton (local coordinates)	6.9660	6.9954
Quasi-Newton (LBFGS)	3.1175	3.8311
HOOI	1.7641	1.8470
Trust-region	0.4100	0.4800

In the second experiment we considered tensors $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ with dominant rank-$(5, 5, 5)$ part. The imposed multilinear rank of the approximation was also $(5, 5, 5)$. The test tensors were constructed as follows:

$$\mathcal{A} = \mathcal{T}/\|\mathcal{T}\| + 0.1\, \mathcal{E}/\|\mathcal{E}\|, \tag{6.2}$$

where $\mathcal{T} \in \mathbb{R}^{20\times 20\times 20}$ represents data and $\mathcal{E} \in \mathbb{R}^{20\times 20\times 20}$ represents noise. The tensor $\mathcal{T}$ was built as the product of a random $(5\times 5\times 5)$-tensor and three random columnwise orthonormal matrices with matching dimensions, i.e., taken equal to the $\mathbf{Q}$ factors of the thin QR factorization of matrices with elements drawn from a normal distribution with zero mean and unit standard deviation. The elements of the $(5\times 5\times 5)$-tensor and of $\mathcal{E}$ were also drawn from a normal distribution with zero mean and unit standard deviation. The initial matrices $\mathbf{U}_0$, $\mathbf{V}_0$, and $\mathbf{W}_0$ were obtained by truncating HOSVD and performing an additional HOOI iteration. This resulted in a point at which the relative norm of the gradient ($\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X})$) was around 10^{-5}. The stopping criterion was the same as in the previous experiment. In this case, the fastest algorithm was HOOI, followed by the trust-region algorithm. The third algorithm was the quasi-Newton algorithm in local coordinates. The timings for one representative run are given in Table 6.2. Finally, we performed the same experiment but with tensors

TABLE 6.2

Time comparisons (in seconds) for tensors $\mathcal{A} \in \mathbb{R}^{20\times 20\times 20}$ and $\mathcal{A} \in \mathbb{R}^{100\times 100\times 100}$ as in (6.2) for all algorithms considered in this paper. The imposed multilinear rank was $(5, 5, 5)$. The initial values $\mathbf{U}_0, \mathbf{V}_0$, and $\mathbf{W}_0$ were obtained by first computing the truncated HOSVD and then computing one additional HOOI iteration. A run was stopped if the corresponding algorithm converged ($\|\text{grad } g(\mathbf{X})\|/g(\mathbf{X}) < 5 * 10^{-13}$) or reached the maximum number of iterations, which was 50 for the pure-Newton and 300 for all other algorithms.

Algorithm	Case $\mathbb{R}^{20\times 20\times 20}$	Case $\mathbb{R}^{100\times 100\times 100}$
Quasi-Newton (global coordinates)	1.0309	76.7010
Quasi-Newton (local coordinates)	0.2853	34.8386
Pure Newton (global coordinates)	0.7813	82.2929
Pure Newton (local coordinates)	0.4328	74.4794
Quasi-Newton (LBFGS)	0.8605	6.7545
HOOI	0.0287	0.1985
Trust-region	0.1900	0.9800

$\mathcal{A} \in \mathbb{R}^{100\times 100\times 100}$. The results are given again in Table 6.2. The difference with the previous case is that the third place here was for the limited memory quasi-Newton algorithm. In the simulations corresponding to Table 6.2, the maximal number of iterations was not reached by any of the algorithms.

The type of examples with random tensors can be considered as difficult, since the structure of the original tensor does not correspond to the structure of the low multilinear rank approximation that we are looking for. The examples with dominant low multilinear rank where we compute approximations with the same multilinear rank structure seem easier. A discussion on this issue can be found in [23].

7. Conclusions. We have developed a new algorithm for computing the best rank-(R_1, R_2, R_3) approximation of higher-order tensors. It is based on the Riemannian trust-region scheme on quotient manifolds. In order to solve the trust-region subproblems, the tCG method is applied. The algorithm converges globally to local minima of the cost function (2.1). Its convergence rate is superlinear and even quadratic for a suitable stopping criterion.

In comparison with the higher-order orthogonal iteration method [13], the advantage of the new algorithm is its faster local convergence rate. The maximal computational cost per iteration of these two algorithms is comparable if the order of the tensor is high or if the imposed multilinear rank is low. This is the case in many applications. Moreover, the trust-region–based algorithm reaches its maximum number of inner iterations and hence its maximal computational cost per iteration only in iterations close to the solution. Thus, the overall performance of the proposed method is to be preferred in many cases.

In the literature also Newton-type methods have been presented [25, 17, 39]. In rather difficult cases, pure Newton algorithms have problems distinguishing between minima, maxima, and saddle points and might even diverge. The trust-region algorithm overcomes these difficulties. The computational cost per iteration of the pure Newton algorithms is of the same order as the one of the trust-region method. The quasi-Newton algorithms have lower cost per iteration but require more iterations in order to converge. A strength of the trust-region over the quasi-Newton algorithms is that the trust-region algorithms have deeper convergence analysis [1].

Appendix A. Here we summarize the idea behind the tCG algorithm on tangent spaces [1] (a generalization of the Steihaug–Toint algorithm [43, 45]) for approximately solving the trust-region subproblem (3.3).

ALGORITHM 2 TRUNCATED CG.

Input: Function f, iterate x_k, trust-region radius Δ_k, inner product $\langle \cdot, \cdot \rangle$, grad$f(x_k)$, and a routine for computing Hess$f(x_k)[\cdot]$.

Output: Approximate solution η of the trust-region subproblem (3.3).

Set $\eta^0 = 0$, $r_0 = \text{grad}f(x_k)$, $\delta_0 = -r_0$.
for $j = 0, 1, 2, \ldots$ **do**
  if a stopping criterion is satisfied **then**
    Return η.
  else
    if $\langle \delta_j, \text{Hess}f(x_k)[\delta_j] \rangle \leq 0$ (negative curvature) **then**
      Compute τ such that[4]
      $\eta = \eta^j + \tau\delta_j$ minimizes $m_{x_k}(\eta)$ in (3.3) and satisfies $\langle \eta, \eta \rangle = \Delta_k^2$.
      Return η.
    else
      Set $\alpha_j = \langle r_j, r_j \rangle / \langle \delta_j, \text{Hess}f(x_k)[\delta_j] \rangle$.
      Set $\eta^{j+1} = \eta^j + \alpha_j \delta_j$.
      if $\langle \eta^{j+1}, \eta^{j+1} \rangle \geq \Delta_k^2$ (exceeded trust region) **then**
        Compute $\tau \geq 0$ such that $\eta = \eta^j + \tau\delta_j$ satisfies $\langle \eta, \eta \rangle = \Delta_k^2$.
        Return η.
      else
        Set $r_{j+1} = r_j + \alpha_j \text{Hess}f(x_k)[\delta_j]$.
        Set $\beta_{j+1} = \langle r_{j+1}, r_{j+1} \rangle / \langle r_j, r_j \rangle$.
        Set $\delta_{j+1} = -r_{j+1} + \beta_{j+1}\delta_j$.
      end if
    end if
  end if
end for

REFERENCES

[1] P.-A. Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds*, Found. Comput. Math., 7 (2007), pp. 303–330.

[2] P.-A. Absil, M. Ishteva, L. De Lathauwer, and S. Van Huffel, *A geometric Newton method for Oja's vector field*, Neural Comput., 21 (2009), pp. 1415–1433.

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

[4] E. Acar, C. A. Bingol, H. Bingol, R. Bro, and B. Yener, *Multiway analysis of epilepsy tensors*, Bioinformatics, 23 (2007), pp. i10–i18.

[5] R. Badeau and R. Boyer, *Fast multilinear singular value decomposition for structured tensors*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1008–1021.

[6] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 10246–10251.

[7] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[8] P. Comon, *Independent component analysis, a new concept?*, Signal Process., 36 (1994), pp. 287–314.

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, PA, 2000.

[10] L. De Lathauwer and J. Castaing, *Tensor-based techniques for the blind separation of DS-CDMA signals*, Signal Process., 87 (2007), pp. 322–336.

[4] It can be shown [43] that τ has to be equal to the positive root of $\|\eta^j + \tau\delta_j\| = \Delta_k$, i.e., $\tau = \left(-\langle \eta^j, \delta_j \rangle + \sqrt{\langle \eta^j, \delta_j \rangle^2 + (\Delta_k^2 - \langle \eta^j, \eta^j \rangle)\langle \delta_j, \delta_j \rangle} \right) / \langle \delta_j, \delta_j \rangle$.

[11] L. De Lathauwer, B. De Moor, and J. Vandewalle, *An introduction to independent component analysis*, J. Chemometrics, 14 (2000), pp. 123–149.
[12] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[13] L. De Lathauwer, B. De Moor, and J. Vandewalle, *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), pp. 1324–1342.
[14] 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.
[15] M. De Vos, L. De Lathauwer, B. Vanrumste, S. Van Huffel, and W. Van Paesschen, *Canonical decomposition of ictal scalp EEG and accurate source localization: Principles and simulation study*, J. Comput. Intell. Neurosci., 2007 (2007), pp. 1–10.
[16] M. De Vos, A. Vergult, L. De Lathauwer, W. De Clercq, S. Van Huffel, P. Dupont, A. Palmini, and W. Van Paesschen, *Canonical decomposition of ictal scalp EEG reliably detects the seizure onset zone*, NeuroImage, 37 (2007), pp. 844–854.
[17] 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), pp. 248–271.
[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[19] M. Haardt, F. Roemer, and G. Del Galdo, *Higher-order SVD-based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems*, IEEE Trans. Signal Process., 56 (2008), pp. 3198–3213.
[20] W. Hackbusch and B. Khoromskij, *Tensor-product approximation to multidimensional integral operators and Green's functions*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1233–1253.
[21] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys., 6 (1927), pp. 164–189.
[22] F. L. Hitchcock, *Multiple invariants and generalized rank of a p-way matrix or tensor*, J. Math. Phys., 7 (1927), pp. 39–79.
[23] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, *Tucker compression and local optima*, Chemometrics Intell. Lab. Syst., (2010), DOI 10.1016/j.chemolab.2010.06.006.
[24] 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), pp. 337–343.
[25] M. Ishteva, L. De Lathauwer, P.-A. Absil, and S. Van Huffel, *Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors*, Numer. Algorithms, 51 (2009), pp. 179–194.
[26] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[27] P. M. Kroonenberg, *Applied Multiway Data Analysis*, Wiley, New York, 2008.
[28] P. M. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97.
[29] P. McCullagh, *Tensor Methods in Statistics*, Chapman and Hall, London, 1987.
[30] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 553–572.
[31] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred, *Parallel factor analysis as an exploratory tool for wavelet transformed event-related EEG*, NeuroImage, 29 (2006), pp. 938–947.
[32] C. L. Nikias and J. M. Mendel, *Signal processing with higher-order spectra*, IEEE Signal Process. Mag., 10 (1993), pp. 10–37.
[33] C. L. Nikias and A.P. Petropulu, *Higher-Order Spectra Analysis. A Nonlinear Signal Processing Framework*, Prentice Hall, Englewood Cliffs, NJ, 1993.
[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer Ser. Oper. Res., Springer, New York, 2006.
[35] 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), pp. 939–956.
[36] J.-M. Papy, L. De Lathauwer, and S. Van Huffel, *Exponential data fitting using multilinear algebra: The single-channel and the multichannel case*, Numer. Linear Algebra Appl., 12 (2005), pp. 809–826.
[37] J.-M. Papy, L. De Lathauwer, and S. Van Huffel, *Exponential data fitting using multilinear algebra: The decimative case*, J. Chemometrics, 23 (2009), pp. 341–351.

[38] B. Savas and L. Eldén, *Krylov Subspace Methods for Tensor Computations*, Technical report LITH-MAT-R-2009-02-SE, Department of Mathematics, Linköping University, Linköping, Sweden, 2009.

[39] B. Savas and L.-H. Lim, *Quasi-Newton methods on Grassmannians and multilinear approximations of tensors*, SIAM J. Sci. Comput., 32 (2010), pp. 3352–3393.

[40] N. Sidiropoulos, R. Bro, and G. Giannakis, *Parallel factor analysis in sensor array processing*, IEEE Trans. Signal Process., 48 (2000), pp. 2377–2388.

[41] N. Sidiropoulos, G. Giannakis, and R. Bro, *Blind PARAFAC receivers for DS-CDMA systems*, IEEE Trans. Signal Process., 48 (2000), pp. 810–823.

[42] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis. Applications in the Chemical Sciences*, Wiley, Chichester, U.K., 2004.

[43] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[44] A. Swami and G. Giannakis, *Bibliography on HOS*, Signal Process., 60 (1997), pp. 65–126.

[45] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, London, 1981, pp. 57–88.

[46] L. R. Tucker, *The extension of factor analysis to three-dimensional matrices*, in Contributions to Mathematical Psychology, H. Gulliksen and N. Frederiksen, eds., Holt, Rinehart and Winston, New York, 1964, pp. 109–127.

[47] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[48] M. A. O. Vasilescu and D. Terzopoulos, *Multilinear subspace analysis for image ensembles*, in Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, WI, 2003, pp. 93–99.