RIEMANNIAN PRECONDITIONED COORDINATE DESCENT FOR LOW MULTI-LINEAR RANK APPROXIMATION[*]

MOHAMMAD HAMED[‡†] AND RESHAD HOSSEINI [‡]

Abstract. This paper presents a memory efficient, first-order method for low multi-linear rank approximation of high-order, high-dimensional tensors. In our method, we exploit the second-order information of the cost function and the constraints to suggest a new Riemannian metric on the Grassmann manifold. We use a Riemmanian coordinate descent method for solving the problem, and also provide a global convergence analysis matching that of the coordinate descent method in the Euclidean setting. We also show that each step of our method with the unit step-size is actually a step of the orthogonal iteration algorithm. Experimental results show the computational advantage of our method for high-dimensional tensors.

Key words. Tucker decomposition, Riemannian optimization, Preconditioning, Coordinate descent, Riemannian metric

AMS subject classifications. 15A69, 49M37, 53A45, 65F08

1. Introduction.

Higher-order tensors are ubiquitous in factor analysis problems across multiple disciplines, including psychometrics, econometrics, biomedical signal processing, data mining, and social network analysis. In particular, tensor decomposition techniques with low-rank approximations offer several benefits, such as reducing the number of dimensions, removing noise, and uncovering latent variables. The utility of tensor decomposition has been demonstrated in a range of applications, from identifying underlying factors in psychometric studies to identifying sources of signals in biomedical research. To gain a comprehensive understanding of tensors and their decomposition methods, Kolda and Bader's review [20] provides a broad perspective, including mathematical foundations and algorithmic approaches, while Sidiropoulos et al.'s work [30] provides a more recent overview with a specific emphasis on signal and data analysis.

Tucker decomposition, first introduced by [33], is a mathematical technique that involves factorizing a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of multi-linear rank-$(r_1, ..., r_d)$ into a core tensor and d factor matrices. This approach extends principal component analysis from matrices to tensors, as noted in [37]. So, in Tucker decomposition, the factor matrices can be viewed as the principal components for each mode of the tensor. The factorization of the tensor $\mathcal{X}$ can be expressed as follows:

$$\mathcal{X} = \mathcal{C} \times_1 U_1 \times_2 \cdots \times_d U_d \ ,$$

where $\mathcal{X} \times_i U_i$ is the i-mode product (see Definition 2.4) between $\mathcal{X}$ and U_i, $\mathcal{C} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ and $U_i \in St(n_i, r_i)$ (see Definition 2.6) denote the core tensor and each of the orthonormal factor matrices, respectively. Typically, $r_i \ll n_i$, which enables $\mathcal{C}$ to be considered as a compressed or dimensionally reduced version of the original tensor $\mathcal{X}$.

[*]Submitted to the editors March 22, 2024.

Funding: The first author was partially supported by the Research Foundation Flanders (FWO) research project G081222N and UA BOF DocPRO4 projects with ID 46929 and 48996.

[†]Department of Mathematics, University of Antwerp, Antwerp, Belgium (mohammad.hamed@uantwerp.be)

[‡]School of ECE, College of Engineering, University of Tehran, Tehran, Iran (mohamad.hamed@ut.ac.ir, reshad.hosseini@ut.ac.ir).

The storage complexity of Tucker decomposition is proportional to $O(\prod_{i=1}^{d} r_i + \sum_{i=1}^{d} n_i r_i)$, as opposed to $O(\prod_{i=1}^{d} n_i)$ for the original tensor $\mathcal{X}$. Once the factor matrices are found, the core tensor can be computed as

$$\mathcal{C} = \mathcal{X} \times_1 U_1^T \times_2 \cdots \times_d U_d^T \ .$$

The manifold $St(n_i, r_i)$, which consists of all $n_i \times r_i$ matrices with orthonormal columns, does not guarantee the uniqueness of the decomposition due to the symmetry inherent in the problem. Any modified core tensor $\bar{\mathcal{C}} = \mathcal{C} \times_1 Q_1^T \times_2 \cdots \times_d Q_d^T$ where Q_is are orthogonal matrices, is another low-rank representation of the original tensor, i.e., $\mathcal{X} = \bar{\mathcal{C}} \times_1 U_1 Q_1 \times_2 \cdots \times_d U_d Q_d$. However, we can achieve uniqueness by constraining the matrices U_i to lie on the Grassmann manifold, as we will demonstrate later in this paper. Therefore, we propose a Riemannian coordinate descent algorithm that operates on the product space of Grassmann manifolds to solve this problem. In addition to the Grassmann manifold, other common constraints for the U_i matrices include statistical independence, sparsity, and non-negativity [9, 26]. These constraints provide prior information about the underlying factors and lead to more interpretable results.

In the manifold optimization literature [1, 5], it is a common practice to transform a constrained optimization problem in Euclidean space into an unconstrained problem on a manifold that represents the constraints. This approach offers several advantages over traditional constrained optimization methods. One significant advantage of Riemannian optimization methods is that they ensures exact satisfaction of constraints at every iteration, whereas classical constrained optimization methods only approximately satisfy constraints. Additionally, manifold optimization respects the geometry of the problem, meaning that the definition of the inner product can provide more meaningful Riemannian gradient directions.

The approach of coordinate descent methods [35], involves making partial updates to the decision variables. This method offers an advantage in terms of the ease of generating search directions and updating variables. This feature is particularly useful when working with large-scale problems. Additionally, coordinate descent methods tend to exhibit fast empirical convergence, particularly during the initial optimization steps. As such, they are a suitable option for approximations.

Gutman and Ho-Nguyen [16] proposed an extension of the coordinate descent method that operates in the manifold domain. Instead of minimizing over coordinates, the authors performed inexact minimization over subspaces of the tangent space at every point. They noted that the convergence rate in the case of product manifolds is comparable to that in the Euclidean setting. Drawing on this insight, we employed a coordinate descent approach to compute the factor matrices in Tucker decomposition. Specifically, we solved an optimization problem for each factor matrix using a reformulated cost function subject to the Grassmann manifold constraint.

Gradient-based algorithms are widely used in solving large-scale problems, but sometimes they encounter convergence issues. To achieve better convergence rates, it is beneficial to find a suitable metric. Although the construction of a Riemannian metric typically focuses on the geometry of the constraints, considering the role of the cost function can also be helpful when possible. This approach was introduced in [24], where the second-order information of the Lagrangian was encoded into the metric. We adopt this method to develop a new metric that demonstrates excellent performance. In addition, incorporating preconditioning into the coordinate descent algorithm in Euclidean space has also been explored in the literature (see [32] for an example).

By combining all these factors, we introduce a new method, termed Riemannian Preconditioned Coordinate Descent (RPCD). The contributions of our paper are as follows:

- RPCD is a first-order optimization-based algorithm which has advantages over SVD-based methods and second-order methods in large scale cases. It is also very efficient with respect to the memory complexity.
- We construct a Riemannian metric by using the second-order information of the cost function and constraint to solve the Tucker decomposition as a series of unconstrained problems on the Grassmann manifold.
- We provide a convergence analysis for the Riemannian coordinate descent algorithm in a relatively general setting. This is done by modification of the convergence analysis in [16] to the case of product manifolds when exponential map and parallel transport are replaced by retraction and vector transport. Our proposed RPCD algorithm for Tucker decomposition is a special case of Riemannian coordinate descent, and therefore the proofs hold for its global convergence.

The results of our experiments, conducted on both synthetic and real data, demonstrate the superior performance of the proposed algorithm.

1.1. Related work. There are two algorithmic approaches to solving the Tucker decomposition problem. The first approach is based on Singular Value Decomposition (SVD) and aims to extend truncated SVD from matrices to tensors. This approach originated with the development of Higher-Order SVD (HOSVD) [10]. The basic idea behind HOSVD is to identify a low-dimensional subspace within the column span of each unfolding of the tensor $\mathcal{X}$, denoted as $X_{(i)}$ for $i = 1, ..., d$. Although HOSVD provides a sub-optimal solution, it is often used as an initialization for other methods when the computational cost is reasonable.

The authors who presented HOSVD proposed a method called Higher Order Orthogonal Iteration (HOOI) [11]. HOOI seeks orthonormal basis for the dominant subspace of each $Y_{(i)}$, which is the matricization of the tensor $\mathcal{Y} = \mathcal{X} \times_{-i} \{U^T\}$ (see Definition 3.4). It is performed using a least-square approach while fixing other factor matrices. By finding a low dimension subspace of $Y_{(i)}$ instead of $X_{(i)}$, HOOI provides a better low multi-linear rank$-(r_1, ..., r_d)$ approximation of $\mathcal{X}$ compared to HOSVD. The Sequentially Truncated HOSVD (ST-HOSVD) is a variation of HOSVD that improves its efficiency. After finding each factor matrix, the tensor is projected using the obtained factor matrix, and the remaining operations are performed on the projected tensor. This method was introduced by Vannieuwenhoven et al. in 2012 [34]. The HOOI method required careful initialization for efficient convergence, while the ST-HOSVD method performs better by choosing a suitable mode sequence - the order in which modes are processed. Multi-linear Principal Component Analysis (MPCA) [23] is another method in this category that is similar to HOSVD but focuses on maximizing the variation in the projected tensor $\mathcal{C}$. In the literature, various versions of HOSVD have been discussed, including hierarchical [15], streaming [31], parallel [2], randomized [8], and scalable [28]. Recently, a fast and memory-efficient method called D-Tucker was introduced in [18].

The second approach to solving the Tucker decomposition problem involves using common second-order optimization algorithms. Eldén and Savas [12], Savas and Lim [29], and Ishteva et al. [17] have reformulated the original problem and apply Newton, quasi-Newton, and trust region methods, respectively, on the product of Grassmann manifolds. Although, utilizing the second-order information leads to algorithms with faster local convergence, these methods suffer from high computational complexity.

Tensor completion is a distinct problem from tensor decomposition, but noteworthy works include those of Kressner et al. [22] and Kasai et al. [19]. These studies are notable for using a first-order Riemannian method on a variant of tensor completion that employs Tucker decomposition. In [22], the Riemannian conjugate gradient method is applied to the manifold of tensors with fixed low multi-linear rank to solve the tensor completion problem. In [19], the authors address the same problem by applying the same method as in [22] but on a product of Grassmann manifolds. The difference between our method and the latter is in the cost function and the optimization approach.

The structure of this paper is as follows: Section 2 presents some preliminary and background information. Section 3 discusses the problem description and reformulation, metric construction, and the proposed algorithms. In Section 4, the convergence analysis of the Riemannian coordinate descent algorithm is presented. Sections 5 and 6 contain the experimental results and conclusion, respectively.

2. Preliminaries and background.

In this paper, calligraphic letters are used for representing tensors $(\mathcal{A}, \mathcal{B}, ...)$ and capital letters for representing matrices $(A, B, ...)$. In the following subsections, we provide some definitions and after that some background on the Riemannian preconditioning.

2.1. Definitions.

In this subsection, we give some definitions.

Definition 2.1 (Tensor). *A tensor is a d-mode multi-dimensional array $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ with n_i as the dimension of the ith mode. Each element in a tensor is denoted by $\mathcal{X}(k_1, ..., k_d)$, for $k_i \in [n_i] = \{1, ..., n_i\}$. Scalars, vectors and matrices are 0-, 1- and 2-mode tensors, respectively.*

Definition 2.2 (Matricization (unfolding)). *The matricization along the ith mode, denoted by $X_{(i)} \in \mathbb{R}^{n_i\times\prod_{j\neq i} n_j}$, is constructed by putting tensor fibers of the ith mode alongside each other. Tensor mode-i fibers are determined by fixing indices in all modes except the ith mode, i.e. $\mathcal{X}(k_1, ..., k_{i-1}, :, k_{i+1}, ..., k_d)$.*

Definition 2.3 (Multi-linear rank). *A tensor is called a rank-$(r_1, ..., r_d)$ tensor, if we have $rank(X_{(i)}) = r_i$, for $i = 1, ..., d$, which indicates the dimension of the vector space spanned by mode-i fibers. It is a generalization of the matrix rank.*

Definition 2.4 (i-mode product). *For tensor $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ and matrix $A \in \mathbb{R}^{m\times n_i}$, the i-mode product $\mathcal{X} \times_i A \in \mathbb{R}^{n_1\times\cdots\times n_{i-1}\times m\times n_{i+1}\cdots\times n_d}$ can be computed by the following formula:*

$$(\mathcal{X} \times_i A)(k_1, ..., k_{i-1}, l, k_{i+1}, ..., k_d) = \sum_{k_i=1}^{n_k} \mathcal{X}(k_1, ..., k_i, ..., k_d)A(l, k_i) \ .$$

Because of the relation $(\mathcal{X} \times_i A)_{(i)} = AX_{(i)}$, i-mode product can be thought of a transformation from a n_i-dimensional space to a m-dimensional space.

Definition 2.5 (Tensor norm). *The norm of a tensor $\mathcal{X}$ is given by*

$$\|\mathcal{X}\|_F = \|X_{(i)}\|_F = \|vec(\mathcal{X})\|,$$

where F is the Frobenious norm and $vec(.)$ is the vectorization operator.

DEFINITION 2.6 (Stiefel manifold $St(n,r)$). *The set of all orthonormal r_i frames in $\mathbb{R}^{n_i}$ is called the Stiefel manifold:*

$$St(n,r) = \{X \in \mathbb{R}^{n \times r} : X^T X = I_r\}.$$

In this manifold, *tangent vectors* at a point X can be written as $\xi_X = X\Omega + X^{\perp}B$, where $\Omega \in Skew(r) = \{A \in \mathbb{R}^{r\times r} : A^T = -A\}$ and $X^{\perp}$ completes the orthonormal basis formed by X, so $X^T X^{\perp} = 0$. If vectors in the normal space are identified by $\nu_X = XA$, we can specify A by implying the orthogonality between tangent vectors and normal vectors.

$$\xi_X \perp \nu_X \ : \ \langle \xi_X, \nu_X \rangle = \langle X\Omega + X^{\perp}B, XA \rangle = 0 \quad \Longrightarrow \quad A \in Sym(r),$$

where $Sym(r)$ is the set of all $r \times r$ symmetric matrices.

The *projection* of an arbitrary vector $Z \in \mathbb{R}^{n\times r}$ onto the tangent space is given by $Proj_X Z = Z - XA$, wherein A is chosen such that the projection complies to the tangent vectors constraint, i.e. $\xi^T X + X^T \xi = 0$:

$$(Z - XA)^T X + X^T (Z - XA) = 0 \qquad \Longrightarrow \qquad A = sym(X^T Z),$$

where $sym(\cdot)$ returns the symmetirc part of the input matrix.

In a Stiefel manifold, like any embedded submanifold, the Riemannian gradient ∇f can be obtained by projecting the Euclidean gradient G onto the tangent space of the current point.

$$\nabla f(X) = Proj_X G(X) = G(X) - X sym(X^T G(X)).$$

A *retraction* $\mathcal{R}_x : T_x\mathcal{M} \to \mathcal{M}$ at the point x on the manifold $\mathcal{M}$ is a way of moving along a direction in the tangent space $T_x\mathcal{M}$ while staying on the manifold. More on that can be found in [5, chapter 3.6]. For the Stiefel manifold we use QR-decomposition for retraction, that is $\mathcal{R}_X(\xi_X) = qr(X + \xi_X)$, where qr is the orthonormal part in the QR-decomposition.

DEFINITION 2.7 (Grassmann manifold $Gr(n,r)$). *We define two matrices X and Y to be equal under* equivalence relation $\sim$ *over $St(n,r)$, if their column spaces span the same subspace. We can define one of these matrices as a transformed version of the other, i.e., $X = YQ$, for some $Q \in O(r)$, where $O(r)$ is the set of all $r \times r$ orthogonal matrices.*

We identify elements in the Grassmann manifold with this equivalence class, that is:

$$[X] = \{Y \in St(n,r) : X \sim Y\} = \{XQ : Q \in O(r)\}.$$

The Grassmann manifold $Gr(n,r)$ is a quotient manifold, $St(n,r)/O(r) = \{[X] : X \in St(n,p)\}$, which represents the set of all linear r-dimensional subspaces in a n-dimensional vector space.

Consider a quotient manifold that is embedded in a total space $\mathcal{M}$ given by the set of equivalence relation $\sim$. A Riemannian metric $\langle \cdot, \cdot \rangle_x$ at $x \in \mathcal{M}$ in the total space can induce a Riemannian metric $\langle \cdot, \cdot \rangle_{[x]}$ on the quotient manifold $\mathcal{M}/\sim$

$$\langle \xi_{[x]}, \eta_{[x]} \rangle_{[x]} = \langle \xi_x, \eta_x \rangle_x,$$

where vectors ξ_x and η_x belong to $\mathcal{H}_x$, the horizontal space of $T_x\mathcal{M}$. This subspace provides a valid matrix representation of the abstract tangent space $T_{[x]}\mathcal{M}/\sim$ as ξ_x and η_x are unique representations of the abstract tangent vectors $\xi_{[x]}$ and $\eta_{[x]}$, respectively. The horizontal space is the orthogonal complement to the vertical space in this Riemannian metric. The vertical space is defined as $\mathcal{V}_x = \ker D\pi(x)$, where $\pi : x \mapsto \pi(x) = [x]$ is the natural projection that links the total space to its quotient.

If the cost function in the total space does not change in the directions of vectors in the vertical space, then the Riemannian gradient in the quotient manifold is given by,

$$\nabla_{[x]} f = \nabla_x f.$$

A *retraction* operator $\mathcal{R}_x : \mathcal{H}_x \to \mathcal{M}$ can be given by,

$$\mathcal{R}_{[x]}(\xi_{[x]}) = [\mathcal{R}_x(\xi_x)],$$

where $\mathcal{R}_x(.)$ is a retraction in the total manifold. For further information on the aforementioned concepts in Riemannian manifold optimization, refer to [1, 5].

2.2. Riemannian preconditioning. Mishra and Sepulchre [24] brought attention to relation between sequential quadratic programming which embeds constraints into the Lagrangian and the Riemannian Newton method which encodes constraints into the search space. Then, they exploited this relation and introduced a way of building Riemannian metrics. Here we bring the gist of their work.

Consider the optimization problem,

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{s.t.} \quad & h(x) = 0, \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^p, n \geq p$ are smooth functions. Sequential quadratic programming deals with the the unconstrained Lagrangian which is defined as

$$\mathcal{L}(x, \lambda) = f(x) - \langle \lambda, h(x) \rangle,$$

in which $\lambda \in \mathbb{R}^p$ represents the Lagrange multiplier. From the optimality condition we know that at a local minimum x, Lagrange multiplier λ_x, can be obtained by

$$h_x(x)\lambda_x = f_x(x),$$

where f_x is the first-order derivative of the cost function $f(x)$ and $h_x(x) \in \mathbb{R}^{n\times p}$ is the Jacobian of the constraints $h(x)$. Applying the pseudo inverse of $h_x(x)$ to $f_x(x)$, we arrive at

$$\lambda_x = (h_x(x)^T h_x(x))^{-1} h_x(x)^T f_x(x),$$

where $h_x(x)$ is assumed to have a full column rank. Therefore the set $\mathcal{M} := \{x \in \mathbb{R}^n \mid h(x) = 0\}$ has an embedded differentiable submanifold structure, and we can recast the equality constrained problem (2.1) as an unconstrained optimization problem on a nonlinear search space. The authors stated that in the neighborhood of a local minimum, the second-order derivative of the Lagrangian in the total space efficiently gives us the second-order information of the problem. The theorem below is brought for more clarification.

Theorem 2.8 (Theorem 3.1 in [24]). *Consider an equivalence relation $\sim$ in $\mathcal{M}$. Assume that both $\mathcal{M}$ and $\mathcal{M}/\sim$ have the structure of a Riemannian manifold and a function $f : \mathcal{M} \to \mathbb{R}$ is a smooth function with isolated minima on the quotient manifold. Assume also that $\mathcal{M}$ has the structure of an embedded submanifold in $\mathbb{R}^n$. If $x^* \in \mathcal{M}$ is a local minimum of f on $\mathcal{M}$, then the following hold:*

- $\langle \eta_{x^*}, D^2\mathcal{L}(x^*, \lambda_{x^*})[\eta_{x^*}]\rangle = 0, \quad \forall \eta_{x^*} \in \mathcal{V}_{x^*}$,
- *the quantity $\langle \xi_{x^*}, D^2\mathcal{L}(x^*, \lambda_{x^*})[\xi_{x^*}]\rangle$ captures all second-order information of the cost function f on $\mathcal{M}/\sim$ for all $\xi_{x^*} \in \mathcal{H}_{x^*}$,*

where $\mathcal{V}_{x^*}$ is the vertical space, and $\mathcal{H}_{x^*}$ is the horizontal space (that subspace of $T_{x^*}\mathcal{M}$ which is orthogonal to the vertical space) and $D^2\mathcal{L}(x^*, \lambda_{x^*})[\xi_{x^*}]$ is the second-order derivative of $\mathcal{L}(x, \lambda_x)$ with respect to x at $x^* \in \mathcal{M}$ applied in the direction of $\xi_{x^*} \in \mathcal{H}_{x^*}$ and keeping λ_{x^*} fixed to its least-squares estimate.

The proper search direction in sequential quadratic programming is computed by solving the following optimization problem in the neighborhood of a minimum:

$$\arg\min_{\xi_x \in \mathcal{H}_x} f(x) - \langle f_x(x), \xi_x\rangle + \frac{1}{2}\langle \xi_x, D^2\mathcal{L}(x, \lambda_x)[\xi_x]\rangle.$$

If $\langle \xi_x, D^2\mathcal{L}(x_k, \lambda_x)[\xi_x]\rangle$ is strictly positive for all tangent vectors ξ_x in the horizontal space $\mathcal{H}_x$ at the point x, then this optimization problem has a unique solution. After updating the variables by moving along the obtained direction, to maintain strict feasibility, it needs a projection onto the constraint, thus they name this method *feasibly projected sequential quadratic programming*.

Now that we know that the Lagrangian captures second-order information of the problem, the authors in [24] introduced a family of regularized metrics that incorporate the second information by using the second-order derivative of the Lagrangian,

$$\langle \xi_x, \eta_x\rangle_x = \omega_1\langle \xi_x, D^2 f(x)[\eta_x]\rangle + \omega_2\langle \xi_x, D^2 c(x, \lambda_x)[\eta_x]\rangle,$$

in which $c(x, \lambda_x) = -\langle \lambda_x, h(x)\rangle$ and $\omega_1 \in [0,1], \omega_2 \in [0,1]$. The first and second terms of this regulated metric correspond to the cost function and the constraint, respectively. In addition to invariance, the metric needs to be positive definite, so:

$$\begin{aligned} &\text{if} \quad D^2 f \succ 0 \quad \text{then} \quad \omega_1 = 1, \quad \omega_2 = \omega \in [0,1),\\ &\text{if} \quad D^2 f \prec 0 \quad \text{then} \quad \omega_2 = 1, \quad \omega_1 = \omega \in [0,1), \end{aligned}$$

where ω can also be updated in each iteration by a rule like $\omega^k = 1 - 2^{1-k}$. We refer the reader to the original paper for further details [24, Section 3.3]. Mishra and Kasai in [19] exploited the idea of Riemannian preconditioning for tensor completion.

3. Problem statement.

In the Tucker decomposition, we want to decompose a dense d-mode tensor $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ into a core tensor $\mathcal{C} \in \mathbb{R}^{r_1\times\cdots\times r_d}$ and d orthonormal factor matrices $U_i \in St(n_i, r_i)$. The Domain of the objective function is the following product manifold,

$$(\mathcal{C}, U_1, ..., U_d) \in \mathcal{M} := \mathbb{R}^{r_1\times\cdots\times r_d} \times St(n_1, r_1) \times \cdots \times St(n_d, r_d).$$

We solve the following optimization problem:

$$\min_{(\mathcal{C}, U_1, \ldots, U_d) \in \mathcal{M}} \quad \|\mathcal{X} - \mathcal{C} \times_1 U_1 \times_2 \cdots \times_d U_d\|_F^2, \tag{3.1}$$

where $\|.\|_F$ is the Frobenius norm. The objective function has a symmetry for the manifold of orthogonal matrices $O(r_i)$, i.e.,

$$f(\mathcal{C}, U_1, ..., U_d) = f(\mathcal{C} \times_1 O_1^T \times_2 \cdots \times_d O_d^T, U_1 O_1, \cdots, U_d O_d).$$

So, this problem is actually an optimization problem on the product of Grassmann manifolds.

We know from [11] that

$$\|\mathcal{X} - \mathcal{C} \times \{U\}\|_F^2 = \|\mathcal{X}\|_F^2 - \|\mathcal{C}\|_F^2, \tag{3.2}$$

where $\mathcal{C} \times \{U\} = \mathcal{C} \times_1 U_1 \times_2 \cdots \times_d U_d$. This means that the minimization of the *reconstruction error* is equivalent to the maximization of *the energy of the projected tensor*. So, for solving the problem (3.1) in a coordinate descent fashion we can recast it as a series of subproblems involving the following maximization problem for the *ith* factor matrix:

$$\max_{[U_i] \in Gr(n_i, r_i)} \frac{1}{2} \|U_i^T Y_{(i)}\|_F^2. \tag{3.3}$$

In this problem $Y_{(i)}$ is the matricization of the tensor $\mathcal{Y}_i$ along the ith mode. The tensor $\mathcal{Y}_i \in \mathbb{R}^{r_1 \times \cdots \times r_{i-1} \times n_i \times r_{i+1} \times \cdots \times r_d}$ is produced by projecting the tensor $\mathcal{X}$ to a lower dimensional subspace by the help of the assumed fixed factor matrices while mode i is excluded, i.e.,

$$\mathcal{Y}_i = \mathcal{X} \times_{-i} \{U^T\} = \mathcal{X} \times_1 U_1^T \times_2 \cdots \times_{i-1} U_{i-1}^T \times_{i+1} U_{i+1}^T \times_{i+2} \cdots \times_d U_d^T. \tag{3.4}$$

If we proceed to decompose a dense tensor $\mathcal{X}$ with a low multi-linear rank using the formulation given in (3.3) on the product of Grassmannian manifolds equipped with the Euclidean metric

$$\langle \xi_{U_i}, \eta_{U_i} \rangle_{U_i} = \mathrm{trace}(\xi_{U_i}^T \eta_{U_i}),$$

we observe that coordinate descent would have poor convergence results. The relative error plots for 10 random samples of $\mathcal{X} \in \mathbb{R}^{100 \times 100 \times 100}$ with multi-linear rank-$(5, 5, 5)$ can be seen in Figure 1.

To give a remedy for the slow convergence using the Euclidean metric, in the next subsection we apply Riemannian preconditioning to construct a new Riemannian metric which we will see in the experiments that it results in an excellent experimental performance.

3.1. Riemannian Preconditioned Coordinate Descent.

In this section, we want to utilize the idea of Riemannian preconditioning in solving the problem (3.3). For the problem

$$\max_{[U_i] \in Gr(n_i, r_i)} \frac{1}{2} \|U_i^T Y_{(i)}\|_F^2,$$

where $Y_{(i)} \in \mathbb{R}^{n_i \times \prod_{j=1, j \neq i}^d r_j}$, the Lagrangian is defined as

$$\mathcal{L}(U_i, \lambda) = -\frac{1}{2} Trace(Y_{(i)}^T U_i U_i^T Y_{(i)}) + \frac{1}{2} \langle \lambda, U_i^T U_i - I \rangle,$$

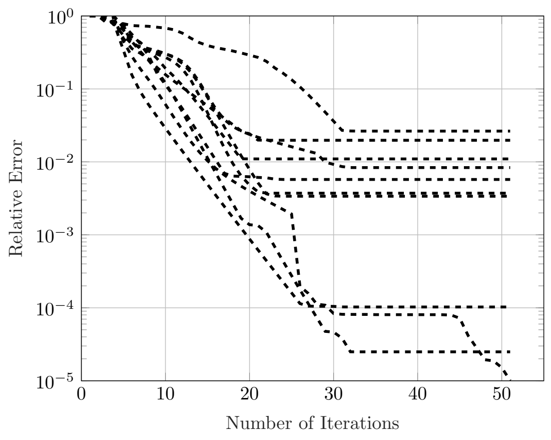


Fig. 1. *Convergence of Riemannian coordinate descent with the Euclidean metric for decomposing a random tensor having a low multi-linear rank. The best attainable relative error is zero, and it is clear that the coordinate descent method has convergence problems in the Euclidean space.*

in which $\lambda \in \mathbb{R}^{r\times r}$ is a symmetric matrix representing the Lagrange multiplier. The gradient of the Lagrangian w.r.t. U_i is

$$\mathcal{L}_{U_i}(U_i, \lambda) = -Y_{(i)}Y_{(i)}^T U_i + U_i\lambda,$$

which due to the optimality condition must be equal to zero at a local minimum. We estimate λ in a least square sense as follows:

$$\lambda_{U_i} = U_i^T Y_{(i)} Y_{(i)}^T U_i.$$

We assumed that λ_{U_i} would be invertible. It is because we know for a fact that $r_i \ll n_i$ and as a result $r_i < \prod_{j\neq i} r_j$, so it is a reasonable guess that λ_{U_i} would be invertible [7]. On top of that, in all of the experiments we have conducted, we never had any issue with the invertiblity of λ_{U_i}.

The second-order derivative of the Lagrangian along ξ_{U_i} while keeping λ_{U_i} fixed (see Theorem 2.8), is

$$D^2\mathcal{L}(U_i, \lambda_{U_i})[\xi_{U_i}] = -Y_{(i)}Y_{(i)}^T\xi_{U_i} + \xi_{U_i}\lambda_{U_i}.$$

As we explained in Section 2.2, the second-order derivative of the Lagrangian captures all the second-order information of the cost function near the minimum. So, to incorporate this information we introduce the Riemannian metric,

$$\langle \eta_{U_i}, \xi_{U_i}\rangle_{U_i} = -\omega_1\langle \eta_{U_i}, Y_{(i)}Y_{(i)}^T\xi_{U_i}\rangle + \omega_2\langle \eta_{U_i}, \xi_{U_i}\lambda_{U_i}\rangle,$$

where $\eta_{U_i}, \xi_{U_i} \in \mathbb{R}^{n_i\times r_i}$ are tangent vectors in $T_{U_i}\mathcal{M}$. If we could have set both ω_1 and ω_2 equal to one, then the Riemannian gradient using this metric would have been the Euclidean Newton direction. But the constants $\omega_1 \in [0, 1]$ and $\omega_2 \in [0, 1]$ should be chosen in a way to make sure that the proposed metric stays positive definite for all points on the manifold. Since, the matrix $-Y_{(i)}Y_{(i)}^T$ is negative semi-definite, we choose $\omega_1 = 0$ and $\omega_2 = 1$, which results in

$$\langle \eta_{U_i}, \xi_{U_i}\rangle_{U_i} = \langle \eta_{U_i}, \xi_{U_i}\lambda_{U_i}\rangle. \tag{3.5}$$

Variables in the search space are invariant under the symmetry transformation, therefore the proposed metric must be invariant under the associated symmetries, i.e.

$$U_i \mapsto U_i Q \quad \text{and} \quad \lambda_{U_i} \mapsto Q^T \lambda_{U_i} Q, \quad Q \in O(r).$$

It can be verified that this property holds for the Riemannian metric (3.5):

$$\langle \eta_{U_i} Q, \xi_{U_i} Q \rangle_{U_i} = \langle \eta_{U_i} Q, \xi_{U_i} Q Q^T \lambda_{U_i} Q \rangle = \langle \eta_{U_i}, \xi_{U_i} \lambda_{U_i} \rangle = \langle \eta_{U_i}, \xi_{U_i} \rangle_{U_i}.$$

In embedded submanifolds, the Riemannian gradient is obtained by orthogonally projecting the Euclidean gradient onto the tangent space. Tangent vectors at the point U_i in a Stiefel manifold can be represented by $\xi = U_i \Omega + U_i^{\perp} B \in T_{U_i}\mathcal{M}$, where $\Omega \in Skew(r)$. Writing normal vectors as $\nu = U_i A + U_i^{\perp} C \in N_{U_i}\mathcal{M}$, we have

$$\begin{aligned} 0 = \langle \xi, \nu \rangle_{U_i} &= \left\langle U_i \Omega + U_i^{\perp} B, U_i A + U_i^{\perp} C \right\rangle_{U_i} \\ &= \left\langle U_i \Omega + U_i^{\perp} B, U_i A \lambda_{U_i} + U_i^{\perp} C \lambda_{U_i} \right\rangle \\ &= \langle \Omega, A \lambda_{U_i} \rangle + \langle B, C \lambda_{U_i} \rangle. \end{aligned}$$

Considering λ_{U_i} to be invertible, it results in $C = 0$ and

$$A = S \lambda_{U_i}^{-1}, \quad S \in Sym(r).$$

Therefore, by putting the normal vectors at U_i as $\nu = U_i S \lambda_{U_i}^{-1}$, the projection of a given matrix G onto the tangent space of the Stiefel manifold is given by,

$$\mathrm{Proj}_{U_i} G = \ G - U_i S \lambda_{U_i}^{-1},$$

which must comply to the tangent vector constraint on the manifold

$$U_i^T (\mathrm{Proj}_{U_i} G) + (\mathrm{Proj}_{U_i} G)^T U_i = 0.$$

Consequently, we have the following Sylvester equation for the symmetric matrix S:

$$\lambda_{U_i} S + S \lambda_{U_i} \ = \ \lambda_{U_i} (U_i^T G + G^T U_i) \lambda_{U_i}.$$

By Riemannian submersion theory [1, section 3.6.2] , we know that this projection belongs to the horizontal space. Thus, there is no need for further projection onto the horizontal space. If we define G as the Euclidean gradient in the total space, we can simply compute the Riemannian gradient by

$$\nabla f_{[U_i]} = G + U_i.$$

Although we used second-order information to form a new preconditioned metric, when we apply the Riemannian coordinate descent method to the Tucker decomposition problem, it leads to a simple first-order algorithm. The preconditioned metric gives us the second-order insight into the problem but the method itself is a first-order method.

With the help of the metric, we introduce the proposed RPCD method in Algorithm 3.1. RPCD is memory efficient, which is desirable because we wanted to reduce the storage complexity of the original tensor $\mathcal{X}$ in the first place. To be specific, assume $n_i = n$ and $r_i = r$, then tensor $\mathcal{Y}_i$ has nr^{d-1} elements and the Euclidean and the Riemannian gradient both have nr elements. Empirically, we observed that the best choice for the step size is $\alpha = 1$. In this case, one step of the inner loop in RPCD is equivalent to one step of the classic orthogonal iteration method (also called subspace iteration) [14, Section 8.2.4] for finding invariant subspaces. In other words, we have shown that the orthogonal iteration method can be seen as a preconditioned Riemannian gradient descent algorithm. The unit step size is also compatible with the fact that the obtained direction is an approximation of the Newton direction.

Algorithm 3.1 RPCD/RPCD+

```
Input: Dense tensor $\mathcal{X}$, set of orthonormal factor matrices $\{U\}$ randomly initialized
  on the Stiefel manifold, retraction operator $\mathcal{R}$, step size $\alpha$, Plus_flag for selecting
  between RPCD and RPCD+, tolerance error of the outer loop $\epsilon$, tolerance error
  of the inner loop specific for RPCD+ $\epsilon'$, maximum number of iterations in the
  outer loop Maxiter and maximum number of iterations in the inner loop specific
  for RPCD+ Maxiter_inner.
  for $k = 1 :$ Maxiter do
    for $i = 1 : d$ do
      $\mathcal{Y}_i \leftarrow \mathcal{X} \times_{-i} \{U^T\}$
      $U_i, C_i \leftarrow$ UPDATE$(\mathcal{R}_{U_i}, Y_{(i)}, U_i)$
      if Plus_flag = true then
        $\bar{E}_0 \leftarrow \sqrt{\|\mathcal{X}\|_F^2 - \|C_i\|_F^2}/\|\mathcal{X}\|_F$
        for $k' = 1 :$ Maxiter_inner do
          $U_i, C_i \leftarrow$ UPDATE$(\mathcal{R}_{U_i}, Y_{(i)}, U_i)$
          $\bar{E}_{k'} \leftarrow \sqrt{\|\mathcal{X}\|_F^2 - \|C_i\|_F^2}/\|\mathcal{X}\|_F$
          if $\bar{E}_{k'} - \bar{E}_{k'-1} \leq \epsilon'$ then
            break
    $E_k \leftarrow \sqrt{\|\mathcal{X}\|_F^2 - \|U_d^T Y_{(d)}\|_F^2}/\|\mathcal{X}\|_F$
    if $k \geq 2$ then
      if $E_k - E_{k-1} \leq \epsilon$ then
        break
  function UPDATE$(\mathcal{R}_{U_i}, Y_{(i)}, U_i)$
    $C_i \leftarrow U_i^T Y_{(i)}$
    $G \leftarrow -Y_{(i)} C_i^T$
    $\nabla f \leftarrow G + U_i$
    $U_i \leftarrow \mathcal{R}_{U_i}(U_i - \alpha \nabla f)$
    return $U_i, C_i$
Output: Set of factor matrices $\{U\}$
```

In Algorithm 3.1 for the retraction, we use the *QR-decomposition* implemented by the *Householder* algorithm in Matlab which has computational complexity $\mathcal{O}(nr^2)$ for $n \times r$ matrices. For the stopping criterion, we use relative error delta which is the amount of difference in the relative error in two consecutive iterations, i.e., $|\mathrm{E}_k - \mathrm{E}_{k-1}| < \epsilon$,

$$\mathrm{E} = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F} = \frac{\sqrt{\|\mathcal{X}\|_F^2 - \|U_i^T Y_{(i)}\|_F^2}}{\|\mathcal{X}\|_F} \tag{3.6}$$

where second equality comes from equation (3.2) and $\|\mathcal{C}\|_F = \|U_i^T Y_{(i)}\|_F$. Since computing the tensor $\mathcal{Y}_i$ is a lot more expensive than the rest of the inner loop computations, $\mathcal{O}(n^d r^{d-1})$, so it would be a good idea to do multiple updates in every inner loop. In RPCD+, when `plus_flag = true`, we repeat the updating process as long as the change in the relative error would be less than a certain threshold ϵ', which can be much smaller than the stopping criterion threshold ϵ.

In the next section, we provide a convergence analysis for the proposed method as an extension of the coordinate descent method to the Riemannian domain in a special case that the search space is a product manifold.

4. Convergence analysis.

The RPCD method can be thought of as an extension of Tangent Subspace Descent (TSD) [16]. TSD is a recent generalization of the coordinate descent method to the manifold domain. To provide a convergence analysis for RPCD, we generalize the convergence analysis of [16] to the case of product manifolds where the exponential map and parallel transport are substituted by retraction and vector transport, respectively. Convergence analysis of the TSD method is a generalization of the Euclidean block coordinate descent method described in [3]. The TSD method with retraction and vector transport is outlined in Algorithm 4.1. The projections in TSD are updated in each iteration of the inner loop with the help of the vector transport operator.

Algorithm 4.1 TSD with retraction and vector transport

```
  Given $\mathcal{R}_x(\xi)$ as a retraction from a point $x$ in the direction of $\xi$ and $\mathcal{T}_x^y$ as a vector
  transport operator from a point $x$ to a point $y$, see Definition 4.1.
Input: Initial point $x^0 \in \mathcal{M}$, and $\tilde{P}^0 = \{P_i^{x^0}\}_{i=1}^m$ are orthogonal projections onto $m$
  orthogonal subspaces of the tangent space at $x^0$
  for $t = 1, 2, \dots$ do
    Set $y^0 := x^{t-1}$, $\tilde{P}^{y^0} := \tilde{P}^{t-1}$
    for $k = 1, \dots, m$ do
      $\alpha_k = \frac{1}{L_k}$          ▷ $L_k$ is the Lipschitz constant for each block of variables determined by Lemma 4.8
      Update $y^k = \mathcal{R}_{y^{k-1}}(-\alpha_k P_k^{y^{k-1}} \nabla f(y^{k-1}))$
      Update $P_i^{y^k} = \mathcal{T}_{y^{k-1}}^{y^k} P_i^{y^{k-1}} \mathcal{T}_{y^k}^{y^{k-1}}$ for $i = 1, \dots, m$.
    Update $x^t := y^m$, $\tilde{P}^t := \tilde{P}^{y^m}$
Output: Sequence $\{x^t\} \subset \mathcal{M}$
```

Before, we start to study the convergence analysis, it would be helpful to quickly review some definitions:

Definition 4.1 (Vector transport [1, Definition 8.1.1]). *A vector transport on a manifold $\mathcal{M}$ is a smooth mapping*

$$\mathcal{T}_{y^{k-1}}^{y^k} : T_{y^{k-1}}\mathcal{M} \mapsto T_{y^k}\mathcal{M},$$

associated with a retraction $y^k = \mathcal{R}_{y^{k-1}}(\eta)$.

Here we assume that our vector transport is an isometry. See [5, Section 10.5] for other properties.

Definition 4.2 (Radially Lipschitz continuously differentiable function). *We say that the pull-back function $f \circ \mathcal{R}$ is radially Lipschitz continuously differentiable for all $x \in \mathcal{M}$ if there exist a positive constant L_{RL} such that for all x and all $\xi \in T_x\mathcal{M}$ the following holds for $r > 0$,*

$$\left| \frac{d}{dt}(f \circ \mathcal{R})(t\xi)|_{t=r} - \frac{d}{dt}(f \circ \mathcal{R})(t\xi)|_{t=0} \right| \le r L_{RL} \|\xi\|$$

where $\frac{d}{dt}$ is the first-order derivative of a single-variable function.

Definition 4.3 (Operator S^k). *It is given as,*

$$S^0 = id_{T_{y^0}\mathcal{M}} \quad , \quad S^k = \mathcal{T}_{y^1}^{y^0} \cdots \mathcal{T}_{y^k}^{y^{k-1}} = S^{k-1}\mathcal{T}_{y^k}^{y^{k-1}} \;\; ; \; 1 \le k \le l,$$

where id is the identity operator. With this operator, we can write the update rule for the projection matrices as $P_i^{y^k} = (S^k)^{-1}P_i^{y^0}S^k$.

Definition 4.4 (retraction-convex). *Function $f : \mathcal{M} \to \mathbb{R}$ is retraction-convex w.r.t $\mathcal{R}$ for all $\eta \in T_x\mathcal{M}$, $\|\eta\|_x = 1$, if the pull-back function $f \circ \mathcal{R}_x$ is convex in its domain.*

Proposition 4.5 (First-order characteristic of a retraction-convex function). *If $f : \mathcal{M} \to \mathbb{R}$ is retraction-convex w.r.t a retraction $\mathcal{R}_x : T_x\mathcal{M} \to \mathcal{M}$, then we know by definition that the pull-back function is convex. For any direction $\eta \in T_x\mathcal{M}$, by the first-order characteristic of the convex function $f(\mathcal{R}_x(r\eta)) : \mathbb{R} \to \mathcal{M}$, we have*

$$f(\mathcal{R}_x(r\eta)) \ge f(\mathcal{R}_x(s\eta)) + (r-s)\frac{d}{dt}(f \circ \mathcal{R}_x)(t)|_{t=s},$$

where $r, s \in \mathbb{R}$. The second term can be interpreted as

$$\frac{d}{dt}(f \circ \mathcal{R}_x)(t)|_{t=s} = Df(\mathcal{R}_x(s\eta))[\boldsymbol{J}\mathcal{R}_x(s\eta)] = \langle \nabla f(\mathcal{R}_x(s\eta)), \boldsymbol{J}\mathcal{R}_x(s\eta)\rangle_{\mathcal{R}_x(s\eta)},$$

where $\boldsymbol{J}\mathcal{R}$ is the Jacobian of the retraction operator and $Df(\cdot)[\cdot]$ is the directional derivative of function f in a specified direction. Thus, for $r = 1$ and $s = 0$,

$$f(\mathcal{R}_x(\eta)) \ge f(x) + \langle \nabla f(x), \eta\rangle_x.$$

Proposition 4.6 (Restricted Lipschitz-type gradient for the pullback function). *We know by [6, Lemma 2.7] that if $\mathcal{M}$ is a compact submanifold of Euclidean space and if f has Lipschitz continuous gradients, then*

$$f(\mathcal{R}_x(\eta)) \le f(x) + \langle \nabla f(x), \eta\rangle_x + \frac{L_g}{2}\|\eta\|_x^2, \qquad \forall \eta \in T_x\mathcal{M},$$

for some $L_g > 0$.

First, we study the first-order optimality condition in the following proposition.

Proposition 4.7 (Optimality condition). *Assume f is a retraction-convex function and there is a retraction curve between any two points on the Riemannian manifold $\mathcal{M}$, then*

$$\nabla f(x^*) = 0 \quad \Leftrightarrow \quad x^* \textit{ is a minimizer.}$$

Proof. Considering any differentiable curve $\mathcal{R}_{x^*}(t\eta)$, which starts at a local optimum point x^*, the pull-back function $f(\mathcal{R}_{x^*}(t\eta))$ has a minimum at $t = 0$ because $\mathcal{R}_{x^*}(t\eta)\big|_{t=0} = x^*$. We know that $(f \circ \mathcal{R})'(0) = \langle \nabla f(x^*), \eta\rangle_{x^*}$, so for this to be zero for all $\eta \in T_{x^*}\mathcal{M}$, we must have $\nabla f(x^*) = 0$.

From the first-order characteristic of the retraction-convex function f we have,

$$f(\mathcal{R}_{x^*}(\eta)) \ge f(x^*) + \langle \nabla f(x^*), \eta\rangle_{x^*}, \qquad \forall \eta \in \mathcal{R}_{x^*}^{-1}(x),$$

and if $\nabla f(x^*) = 0$ then $f(x) \ge f(x^*)$, hence the point x^* is a global minimum point.□

With the following Lip-Block lemma and the descent direction advocated by Algorithm 4.1, we can prove the Sufficient Decrease lemma.

LEMMA 4.8 (Lip-Block). *If f has the restricted Lipschitz-type gradient, then for any $i, k \in \{1, ..., m\}$ and all $\nu \in Im(P_i^{k-1}) \subset T_{y^{k-1}}\mathcal{M}$, where $Im(\cdot)$ is the subspace that a projection matrix spans, there exist constants $0 < L_1, ..., L_m < \infty$ such that*

$$f(\mathcal{R}_{y^{k-1}}(\nu)) \leq f(y^{k-1}) + \langle \nabla f(y^{k-1}), \nu \rangle_{y^{k-1}} + \frac{L_i}{2} \|\nu\|^2_{y^{k-1}}. \tag{4.1}$$

Proof. By the fact that $\nu \in T_{y^{k-1}}\mathcal{M}$, it can be seen easily that (4.1) is the block version of the restricted Lipschitz-type gradient for the pullback function. ∎

LEMMA 4.9 (Sufficient decrease). *Assume f has the restricted Lipschitz-type gradient, and furthermore $f \circ \mathcal{R}$ is a radially Lipschitz continuous differentiable function. Using the projected gradient onto the kth subspace in each inner loop iteration of Algorithm 4.1, i.e., $\nu = -\frac{1}{L_k} P_k^{y^{k-1}} \nabla f(y^{k-1})$, where $P_k^{y^{k-1}}$ is the orthogonal projection to kth subspace of $T_{y^{k-1}}\mathcal{M}$, then we have*

$$f(y^0) - f(y^m) \geq \sum_{k=1}^{m} \frac{1}{2L_k} \|P_k^{y^{k-1}} \nabla f(y^{k-1})\|^2_{y^{k-1}} . \tag{4.2}$$

The following inequality also holds

$$\|P_i^{y^0} \nabla f(y^0) - P_i^{y^0} S^{i-1} \nabla f(y^{i-1})\|^2_{y^0} \leq C \sum_{k=1}^{i-1} \|P_k^{y^k-1} \nabla f(y^{k-1})\|^2_{y^{k-1}} , \tag{4.3}$$

for $C = (m-1)L_{RL}^2/L_{min}^2$, where $L_{\min} = \min\{L_1, ..., L_m\}$ and L_{RL} is the radially Lipschitz constant. For more details on the operator S see Definition 4.3. Furthermore, there is a lower bound on the cost function decrease at each iteration of the outer loop in Algorithm 4.1:

$$f(y^0) - f(y^m) \geq \frac{1}{4L_{max}(1 + Cm)} \|\nabla f(y^0)\|^2_{y^0}, \tag{4.4}$$

where $L_{\max} = \max\{L_1, ..., L_m\}$.

Proof. The inequality (4.3) was proved in [16, Lemma 4.3] with a slight difference in notation and using the exponential map and parallel transport instead of a retraction and a corresponding vector transport. Because of radially Lipschitz continuous differentiability of $f \circ \mathcal{R}$, we have constant L_{RL} instead of L_f. The inequalities (4.2) and (4.4) were proven in [16, Lemma 4.1] with the mentioned changes. ∎

In the following theorem, we give a convergence rate for the global convergence, then we prove a tighter global rate of convergence for retraction-convex functions.

THEOREM 4.10 (Global convergence). *Assume f has restricted Lipschitz-type gradient and is lower bounded. Then for the sequence generated by Algorithm 4.1, we have $\|\nabla f(x^t)\|^2_{x^t} \to 0$, and we have the following as the rate of convergence:*

$$\min_{i=\{1,...,t\}} \|\nabla f(x^{i-1})\|_{x^{i-1}} \leq \sqrt{\frac{1}{t}\Big(f(x^0) - f(x^t)\Big) 4L_{max}(1 + Cm)} . \tag{4.5}$$

Proof. From (4.4), we have

$$f(x^0) - f(x^t) \geq \frac{1}{4L_{max}(1+Cm)} \sum_{i=1}^{t} \|\nabla f(x^{i-1})\|^2_{x^{i-1}} , \tag{4.6}$$

So, from the fact that f is lower bounded, we can easily conclude that

$$t \to \infty \quad : \quad \|\nabla f(x^t)\|^2_{x^t} \to 0.$$

The inequality (4.5) can be derived effortlessly from (4.6). □

THEOREM 4.11 (Tighter global convergence). *Let $f : \mathcal{M} \to \mathbb{R}$ be a retraction-convex function and the Sufficient Decrease lemma holds for the sequence $\{x^t\} \subset \mathcal{M}$. If we denote the sufficient decrease constant $1/K$, i.e.*

$$\frac{L^2_{\min}}{4L_{\max}\left(L^2_{\min} + L^2_{RL} m(m-1)\right)} = \frac{1}{K},$$

then for $t > 1$ and $\eta_t = \mathcal{R}^{-1}_{x^t}(x^*)$

$$f(x^{t+1}) - f^* \leq \frac{K\|\eta_t\|^2_{x^t}(f(x^1) - f^*)}{K\|\eta_t\|^2_{x^t} + t(f(x^1) - f^*)} \tag{4.7}$$

Proof. From the retraction-convexity of function f we have

$$0 \leq f(x^t) - f(x^*) \leq -\langle \nabla f(x^t), \eta_t \rangle_{x^t} \leq \|\nabla f(x^t)\|_{x^t} \|\eta_t\|_{x^t}.$$

After combining that with the result of Sufficient Decrease lemma 4.9, we will have

$$f(x^t) - f(x^{t+1}) \geq \frac{1}{K\|\eta_t\|^2_{x^t}} \left[f(x^t) - f(x^*)\right]^2.$$

We know that for every real-valued decreasing sequence A_t if $A_t - A_{t+1} \geq \alpha A_t^2$ for some α, then $A_{t+1} \leq \frac{A_1}{1+A_1\alpha t}$. Using this on the above inequality, we reach the convergence bound (4.7). □

Transitivity for vector transports, i.e. $\mathcal{T}^y_x \mathcal{T}^z_y = \mathcal{T}^z_x$, does not hold for Riemannian manifolds in general. But due to the fact that each point and each tangent vector in a product manifold is represented by Cartesian products, we can obtain the constant C in a simpler way than what has come in the proof of Lemma 4.9. For product manifold in the Tucker decomposition problem (3.1), each orthogonal projection of the gradient is simply the gradient of the cost function w.r.t the variables of one of the manifolds in the product manifold, and therefore the gradient projection belongs to the tangent space of that manifold. For a tangent vector satisfying $\xi_{y^0} = \mathcal{R}^{-1}_{y^0}(y^{i-1})$ which is the case for product manifolds, we have

$$\begin{aligned}
\|\nabla f(y^0) - S^{i-1}\nabla f(y^{i-1})\|^2_{y^0} &\leq L^2_{RL}\|\xi_{y^0}\|^2_{y^0} \\
&\leq L^2_{RL} \sum_{j=1}^{i-1} \left\| -\frac{1}{L_j} P_j^{y^{j-1}} \nabla f(y^{j-1}) \right\|^2_{y^{i-1}} \\
&\leq \frac{L^2_{RL}}{L^2_{min}} \sum_{j=1}^{i-1} \left\| P_j^{y^{j-1}} \nabla f(y^{j-1}) \right\|^2_{y^{i-1}},
\end{aligned}$$

where in the second inequality we exploited triangular inequality on $T_{y_0}\mathcal{M}$ and the fact that vector transport is isometric. So, the factor $m-1$ is removed from the rates of convergence in Theorem 4.10 and Theorem 4.11, thus they match the rates of convergences of the coordinate descent method in the Euclidean setting [3].

The Tucker decomposition problem (3.1) is not retraction-convex, so we can not use the result of Theorem 4.11 for it. But by Proposition 4.6 and the fact that the objective function is lower bounded, we reach the following corollary from Theorem 4.10.

COROLLARY 4.12. *The RPDC algorithm given in Algorithm 3.1 with $m = d$ has the same rate of global convergence as given in Theorem 4.10, wherein $C = L_{RL}^2/L_{min}^2$.*

It is worth noting that the proof of convergence for the HOOI method which solves the same objective function was investigated in [36], but it did not provide a convergence rate.

5. Experimental Results.

In this section, we evaluate the performance of our proposed methods on *synthetic* and *real* data. We implemented the proposed methods in Matlab[1] R2023a and also used available Matlab implementations of other methods for comparison. The experiments were performed on a laptop computer with an Intel Core-i7 12700H CPU and 32 GB of memory. We used Matlab's default setting regarding the use of CPU cores usage. The stepsize for RPCD and RPCD+ is set to one. For the tables, ϵ is put to 0.001 and for the figures, it is set to 10^{-5}. For the RPCD+ algorithm, we choose $\epsilon' = \epsilon/10$.

To establish a fair and objective evaluation of our approach relative to others, we employed a uniform stopping criterion across all algorithms, based on a measure of relative error delta. This metric quantifies the degree to which an algorithm can reduce relative error, ensuring that any observed differences in performance are not due to variations in stopping criteria.

The reported time for each method is the actual time that the method spends on the computations which leads to the update of the parameters, and the time for calculating the relative error or other computations are not taken into account. For the RPCD+ algorithm, we also take into the count the time needed to evaluate the relative error in the inner loop. For HOOI and ST-HOSVD implementations, we use `tucker_als` and `hosvd` from the Tensor Toolbox [21][2], respectively.

5.1. Synthetic Data.

In this part, we give the results for two cases of Tucker decomposition on dense random tensors. In both cases, the elements of random matrices or tensors are drawn from a normal distribution with zero mean and unit variance. In the first case, we generate each rank-(r_1, r_2, r_3) tensor $\mathcal{A}_1$ from the i-mode production of a random core tensor in $\mathbb{R}^{r_1 \times r_2 \times r_3}$ and 3 orthonormal matrices constructed by the *QR-decomposition* of random n_i by r_i matrices. In the second case, which has more resemblance with real data with intrinsic low-rank representations, we construct each tensor $\mathcal{A}_2$ by adding noise to a low-rank tensor,

$$\mathcal{A}_2 = \mathcal{L}/\|\mathcal{L}\|_F + 0.1 \cdot \mathcal{N}/\|\mathcal{N}\|_F,$$

where $\mathcal{L}$ is a low-rank tensor generated similar to $\mathcal{A}_1$ and $\mathcal{N}$ is a tensor with random elements.

In both set of experiments, we set $r_1 = r_2 = r_3 = 5$. Because of the memory limitation, except in the last experiment we increase the dimension of just the first

[1]Implementation of RPCD and RPCD+ can be found via https://github.com/utvisionlab/rpcd
[2]The latest release can be found here https://gitlab.com/tensors/tensor_toolbox/-/releases

Table 1
Execution time comparison in seconds for the RPCD+, HOOI and ST-HOSVD methods in the low-rank ($\mathcal{A}_1$) and low-rank with noise ($\mathcal{A}_2$) settings.

n	$\mathcal{A}_1$				$\mathcal{A}_2$			
	RPCD+	HOOI	ST-HOSVD (eigs)	ST-HOSVD (svds)	RPCD+	HOOI	ST-HOSVD (eigs)	ST-HOSVD (svds)
[100,100,100]	0.04	0.1	**0.02**	0.06	0.05	0.1	**0.02**	0.07
[1000,100,100]	0.08	0.14	**0.05**	0.18	0.08	0.14	**0.05**	0.39
[10,000,100,100]	**0.38**	1.71	0.84	1.52	**0.4**	1.72	0.86	4.22
[20,000,100,100]	**0.72**	5.52	2.70	3.04	**0.74**	5.64	2.83	9.19
[40,000,100,100]	**1.33**	20.17	9.54	6.09	**1.45**	19.75	9.72	16.90
[1000,1000,1000]	**4.52**	4.56	7.69	10.72	**5.02**	5.27	9.34	92.3

mode of the input dense tensor to have the performance comparison in higher dimensions. In the last experiment, we uniformly increase all dimensions of the input tensor to demonstrate the effect of tensor volume on different methods. Each experiment is repeated 5 times and the reported time is the average value. The results can be seen in Table 1.

As it can be seen in Table 1, increasing the dimensionality leads to the emergence of a performance gap between the RPCD+ and HOOI algorithms. This is because in RPCD+, the subproblem (3.3) is solved *inexactly* by the *QR-decomposition*; but in HOOI, it is solved almost *exactly* by finding the *eigenvectors* of the large matrix $Y_{(i)}Y_{(i)}^T$. In `tucker_als`, it is done by the Matlab function `eigs` that uses the *Lanczos* algorithm with the number of computations (the number of additions and multiplications) approximately equal to $T(4n_i^2 r_i + 8r_i^3)$, where T is the number of iterations in the loop of the *Lanczos* algorithm. Without loss of the generality, we analyze the computational complexity for the last block $i = d$. The number of computations of $\mathcal{Y}_d$ is equal to $2\sum_{k=1}^{d-1}\prod_{l=1}^{k} r_l \prod_{m=k}^{d} n_m$. Therefore in total, the number of computations of HOOI is approximately equal to $2\sum_{k=1}^{d-1}\prod_{l=1}^{k} r_l \prod_{m=k}^{d} n_m + n_d^2\prod_{k=1}^{d-1} r_k + T(4n_d^2 r + 8r_d^3)$ which is higher than RPCD/RPCD+ with the number of computations approximately equal to $2\sum_{k=1}^{d-1}\prod_{l=1}^{k} r_l \prod_{m=k}^{d} n_m + T'(4n_d\prod_{k=1}^{d} r_k + 2r_d^2 n_d - 2r_d^3/3)$, where T' is the number of iterations in the inner loop of RPCD+. RPCD+ solves the the subproblem (3.3) inexactly, therefore T' is significantly smaller than T.

Both the RPCD+ and HOOI methods reach desirable relative error, zero in the first case and 10% for the second case, in the same number of iterations. But as the cost of each iteration is less for the RPCD+ method, we observe a reduced computational time in total.

As the Tensor Toolbox implementation of the ST-HOSVD method is inefficient due to the using the `eig` function, we modified the `hosvd` function to utilize the more efficient `eigs` and `svds` functions. As can be seen from Table 1, the ST-HOSVD method performs exceptionally well in lower dimensions, but as the dimension increases, although it outperforms HOOI, it still lags behind RPCD+.

We also investigated the case of overestimating the multi-linear rank for the synthetic data. We observed that in such cases, unlike the HOOI method which gets very slow, the proposed method does not need much additional time to compute the factor matrices. For example, in decomposing a noisy tensor $\mathcal{X} \in \mathbb{R}^{10000\times100\times100}$ and multi-linear rank$-(5,5,5)$ with over estimated lower multi-linear rank$-(7,7,7)$, the required time for RPCD+ and HOOI methods went from .39 and 1.71 seconds to .41 and 3.16 seconds, respectively.

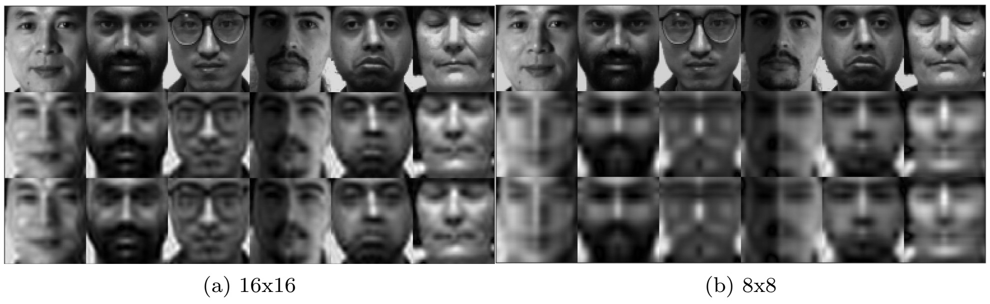


Fig. 2. *Compression of Yale face database (1th row) with HOOI (2th row) and RPCD+ (3th row)*

5.2. Real Data. In the first experiment of this subsection, we compare the RPCD+ and HOOI methods for compressing the images in *Yale face database*[3][4]. This dataset contains 165 grayscale images of 15 individuals. There are 11 images per subject in different facial expressions or configuration, thus we have a dense tensor $\mathcal{X} \in \mathbb{R}^{64\times64\times11\times15}$. For two levels of compression, we decompose $\mathcal{X}$ to three tucker tensor with multi-linear rank $(16, 16, 11, 15)$ and $(8, 8, 11, 15)$, respectively. The results are shown in Figure 2.

The first row in each figure contains the original images, the second and third rows contain the results of the compression using the HOOI and RPCD+ methods, respectively. The attained relative error for both algorithms are the same but RPCD+ is faster (0.09 vs 0.15 seconds) for the case of 16×16. The difference in speed becomes larger (0.06 vs 0.12 seconds), when we want to compress the data more, that is the case of 8×8.

In another comparison for the real data, we compare RPCD, RPCD+ and HOOI with a newly introduced SVD-based method called D-Tucker[4] [18]. D-Tucker compresses the original tensor by performing randomized SVD on slices of the re-ordered tensor and then computes the orthogonal factor matrices and the core tensor using SVD. The paper [18] reported that this method works well when the dimensions of a tensor is high in two modes, and the rest of the modes are low dimensional. That is, for $\mathcal{X}_{re} \in \mathbb{R}^{I_1\times I_2\times K_3\times\cdots\times K_d}$ where we have $I_1 \geq I_2 \gg K_3 \geq \cdots \geq K_d$, the algorithm needs to compute $L = K_3 \times \cdots \times K_d$ randomized SVDs.

The results from the real data are presented in Table 2 and Figure 3. As shown in Table 2, three different target ranks for each dataset are used. The first set of target ranks was chosen based on the D-Tucker paper [18]. For the second set, we multiplied the first set of target ranks by the factor of five. For the third target ranks we aim for compression, around 10% relative error, raising the ranks of the first set uniformly to achieve the desired quality. The first set of target ranks was used for the plots of Figure 3. We initialized the factor matrices using the standard practice of setting the main diagonal to one and the rest to zero, as commonly done in iterative eigen-solvers. The reason for the discrepancy of timing results between the figure and table is the smaller stopping criterion chosen to produce the plots. For Figure 3, we allowed the algorithms to iterate more to clearly see the convergence behavior of different methods. The vertical axis in Figure 3 is representing the difference between

[3]A 64x64 version can be found here http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

[4]We use the Matlab implementation they provided in https://datalab.snu.ac.kr/dtucker/

TABLE 2
Execution time in seconds and relative error for the D-Tucker, RPCD, RPCD+ and HOOI methods on the real datasets.

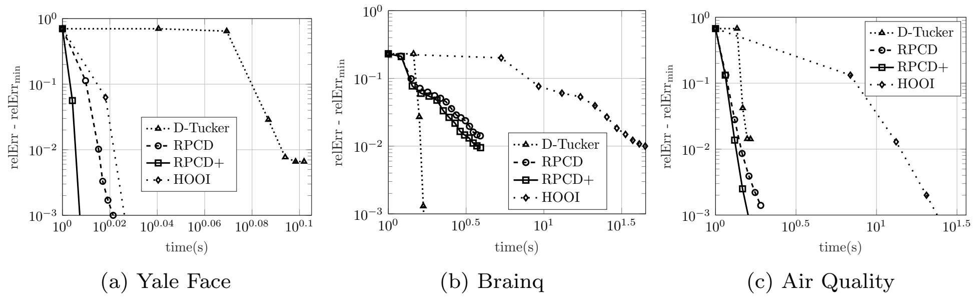
Dataset	Yale [4]		Brainq [25]		Air Quality[5]		HSI [13]		Coil-100 [27]	
Dimension	[64 64 11 15]		[360 21764 9]		[30648 376 6]		[1021 1340 33 8]		[128 128 72 100]	
Target Rank	[5 5 5 5]		[10 10 5]		[10 10 5]		[10 10 10 5]		[5 5 5 5]	
	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)
D-Tucker	0.11	30.46	**0.8**	**77.38**	**0.58**	33.08	**3.13**	45.17	5.84	36.64
RPCD	0.04	30.02	2.88	78.35	0.87	32.87	6.33	43.69	1.24	36.42
RPCD+	**0.02**	29.93	2.81	77.87	0.84	32.74	4.20	43.48	**0.72**	**36.35**
HOOI	**0.02**	**29.92**	41.09	77.92	33.82	**32.72**	4.31	**43.42**	0.74	**36.35**
Target Rank	[25 25 5 5]		[50 50 5]		[50 50 5]		[50 50 10 5]		[25 25 25 25]	
	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)
D-Tucker	–	–	3.93	**60.78**	–	–	12.18	32.60	–	–
RPCD	0.06	26.84	**2.75**	67.86	**1.51**	24.44	9.28	32.18	2.77	24.37
RPCD+	**0.04**	26.56	4.53	65.18	**1.52**	24.39	**6.57**	32.10	1.73	24.30
HOOI	0.05	**26.54**	96.51	65.00	84.50	**24.32**	**6.57**	**32.06**	**1.65**	**24.24**
Target Rank	[17 17 11 15]		[360 2700 9]		[300 300 6]		[220 220 33 8]		[70 70 70 70]	
	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)	time(s)	E(%)
D-Tucker	–	–	–	–	–	–	–	–	–	–
RPCD	0.08	10.42	**65.75**	10.30	**8.05**	10.89	27.38	10.52	14.58	10.15
RPCD+	0.06	10.37	101.56	10.16	14.34	10.84	22.87	10.47	10.76	10.02
HOOI	**0.05**	**10.34**	1243.86	**10.12**	395.23	**10.83**	**17.96**	**10.44**	**4.58**	**9.93**

relative error of each algorithm and the relative error of the most accurate algorithm (a.k.a E_{min}). The plots are cut at 10^{-3} as smaller differences in relative error is minuscule and we can ignore them.

As shown in Table 2, the RPCD+ method consistently achieves a better final relative error than RPCD, thanks to its precision update process. Additionally, in some cases, RPCD+ is faster due to its smaller number of required iterations to converge. We observe that in the Air Quality and HSI datasets, D-Tucker is computationally advantageous, but as shown in Figure 3, this advantage is due to early stopping, resulting in poorer precision. In contrast, RPCD+ and HOOI perform well in terms of speed and precision in the Yale and Coil-100 datasets, which have large L values and the provided code for D-Tucker returns `‘‘rank deficient warning’’`. For the third set of target ranks, the implementation of D-Tucker did not work. For Brainq and Air Quality datasets, we can see the advantage of RPCD as the cost of updating each factor matrix becomes higher for large target ranks. In such cases, it is better to alternate between factor matrices instead of finding a better update for each of them.

Both RPCD+ and HOOI offer the best low multi-linear rank approximation, but HOOI is considerably slower in high-dimensional cases. It is worth noting that the slight difference in the relative Error between RPCD+ and HOOI reported in Table 2 which is a bit in favor of HOOI is the equal stopping criterion used for the methods. Each step of HOOI decreased the error more than one step of RPCD+ as it can be seen in Figure 3. Therefore, it is natural to choose a lower stopping criterion for RPCD+ but we used equal stopping criterion to avoid the concern of its arbitrary selection. An important observation from these experiments is that RPCD+ performs well in lower dimensions and offers superior performance in high-dimensional cases, as seen in the results from the synthetic data. Therefore, RPCD+ is a reliable general method for multi-linear data analysis.

For all datasets except Brainq, we observe almost identical convergence behavior when we start at different starting points. The effect of different initialization on the



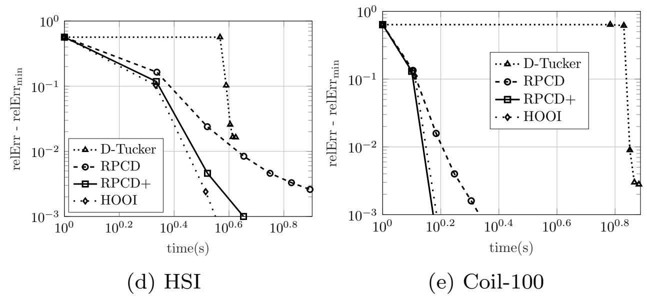


FIG. 3. *Convergence behavior of different methods for the real datasets. Y-axis is the difference between the relative error at each iteration and the best achieved relative error.*

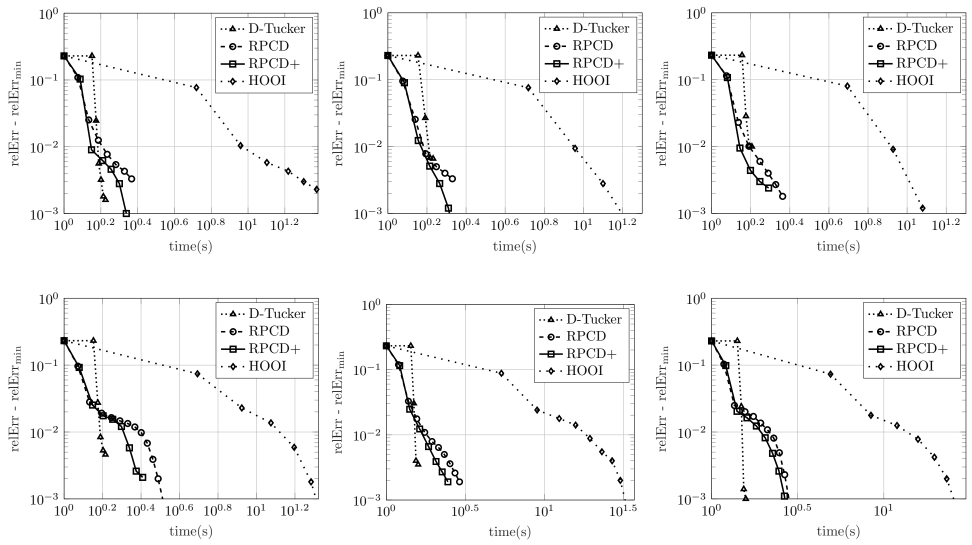


FIG. 4. *Convergence behavior for decomposing the Brainq dataset using different random initializations.*

performance of different methods for Brainq can be seen in Figure 4.

6. Conclusion. In this paper, we introduced RPCD and its improved version RPCD+, first-order method solving the Tucker decomposition problem for high-order, -dimensional dense tensors with the Riemannian coordinate descent method. For these methods, we constructed a Riemannian metric by incorporating the second order information of the reformulated cost function and the constraint. We proved a convergence rate for general tangent subspace descent on Riemannian manifolds, which for the special case of product manifolds like the Tucker decomposition matches the rate in the Euclidean setting. Experimental results showed that RPCD+ as a general method has competitive performance among competing methods for high-order, high-dimensional tensors.

For a future work, it would be interesting to examine the RPCD method in solving tensor completion problems. Another interesting line of work would be to incorporate latent tensors between the original tensor $\mathcal{X}$ and the projected tensor $\mathcal{Y}$ for further reducing computation costs.

Acknowledgments. The authors would like to express their gratitude for the exceptional careful comments of the anonymous reviewers in all rounds of the review process that helped improving the quality of the paper.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, United States, 2009.

[2] W. Austin, G. Ballard, and T. G. Kolda, *Parallel tensor compression for large-scale scientific data*, in IEEE International parallel and distributed processing symposium (IPDPS), 2016, pp. 912–922.

[3] A. Beck and L. Tetruashvili, *On the convergence of block coordinate descent type methods*, SIAM Journal on Optimization, 23 (2013), pp. 2037–2060.

[4] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, *Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997), pp. 711–720.

[5] N. Boumal, *An introduction to optimization on smooth manifolds*, Cambridge University Press, Cambridge, United Kingdom, 2023.

[6] N. Boumal, P.-A. Absil, and C. Cartis, *Global rates of convergence for nonconvex optimization on manifolds*, IMA Journal of Numerical Analysis, 39 (2019), pp. 1–33.

[7] E. Carlini and J. Kleppe, *Ranks derived from multilinear maps*, Journal of Pure and Applied Algebra, 215 (2011), pp. 1999–2004.

[8] M. Che and Y. Wei, *Randomized algorithms for the approximations of Tucker and the tensor train decompositions*, Advances in Computational Mathematics, 45 (2019), pp. 395–428.

[9] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, West Sussex, United Kingdom, 2009.

[10] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278.

[11] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-$(r_1, r_2, ..., r_n)$ approximation of higher-order tensors*, SIAM journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342.

[12] 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 Journal on Matrix Analysis and applications, 31 (2009), pp. 248–271.

[13] D. H. Foster, K. Amano, S. M. Nascimento, and M. J. Foster, *Frequency of metamerism in natural scenes*, Journal of the Optical Society of America A, 23 (2006), pp. 2359–2372.

[14] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, United States, 1996.

[15] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2029–2054.

[16] D. H. Gutman and N. Ho-Nguyen, *Coordinate descent without coordinates: Tangent subspace descent on Riemannian manifolds*, Mathematics of Operations Research, (2022).

[17] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, *Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 115–135.

[18] J.-G. Jang and U. Kang, *D-Tucker: Fast and memory-efficient tucker decomposition for dense tensors*, in IEEE International Conference on Data Engineering (ICDE), 2020, pp. 1850–1853.

[19] H. Kasai and B. Mishra, *Low-rank tensor completion: a Riemannian manifold preconditioning approach*, in International Conference on Machine Learning (ICML), 2016, pp. 1012–1021.

[20] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[21] T. G. Kolda and B. W. Bader, *Tensor toolbox for MATLAB, version 3.2.1*, 2021, https://www.tensortoolbox.org.

[22] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT Numerical Mathematics, 54 (2014), pp. 447–468.

[23] H. Lu, K. Plataniotis, and A. Venetsanopoulos, *MPCA: Multilinear principal component analysis of tensor objects*, IEEE Transactions on Neural Networks, 19 (2008), pp. 18–39.

[24] B. Mishra and R. Sepulchre, *Riemannian preconditioning*, SIAM Journal on Optimization, 26 (2016), pp. 635–660.

[25] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just, *Predicting human brain activity associated with the meanings of nouns*, Science, 320 (2008), pp. 1191–1195.

[26] M. Mørup, L. K. Hansen, and S. M. Arnfred, *Algorithms for sparse nonnegative Tucker decompositions*, Neural Computation, 20 (2008), pp. 2112–2131.

[27] S. A. Nene, S. K. Nayar, and H. Murase, *Columbia object image library (COIL-100)*, Tech. Report CUCS-006-96, Department of Computer Science, Columbia University, 1996.

[28] J. Oh, K. Shin, E. E. Papalexakis, C. Faloutsos, and H. Yu, *S-hot: Scalable high-order Tucker decomposition*, in ACM International Conference on Web Search and Data Mining (WSDM), 2017, pp. 761–770.

[29] B. Savas and L.-H. Lim, *Quasi-Newton methods on Grassmannians and multilinear approximations of tensors*, SIAM Journal on Scientific Computing, 32 (2010), pp. 3352–3393.

[30] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Transactions on Signal Processing, 65 (2017), pp. 3551–3582.

[31] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, *Low-rank Tucker approximation of a tensor from streaming data*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1123–1150.

[32] R. Tappenden, P. Richtárik, and J. Gondzio, *Inexact coordinate descent: complexity and preconditioning*, Journal of Optimization Theory and Applications, 170 (2016), pp. 144–176.

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

[34] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A new truncation strategy for the higher-order singular value decomposition*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1027–A1052.

[35] S. J. Wright, *Coordinate descent algorithms*, Mathematical Programming, 151 (2015), pp. 3–34.

[36] Y. Xu, *On the convergence of higher-order orthogonal iteration*, Linear and Multilinear Algebra, 66 (2018), pp. 2247–2265.

[37] A. Zare, A. Ozdemir, M. A. Iwen, and S. Aviyente, *Extension of PCA to higher order data structures: An introduction to tensors, tensor decompositions, and tensor PCA*, Proceedings of the IEEE, 106 (2018), pp. 1341–1358.