SIAM J. SCI. COMPUT.
Vol. 45, No. 6, pp. A2781–A2812

ALTERNATING MAHALANOBIS DISTANCE MINIMIZATION FOR ACCURATE AND WELL-CONDITIONED CP DECOMPOSITION*

NAVJOT SINGH† AND EDGAR SOLOMONIK†

Abstract. Canonical polyadic decomposition (CPD) is prevalent in chemometrics, signal processing, data mining, and many more fields. While many algorithms have been proposed to compute the CPD, alternating least squares (ALS) remains one of the most widely used algorithms for computing the decomposition. Recent works have introduced the notion of eigenvalues and singular values of a tensor and explored applications of eigenvectors and singular vectors in signal processing, data analytics, and various other fields. We introduce a new formulation for deriving singular values and vectors of a tensor by considering the critical points of a function differently from previous works. Computing these critical points in an alternating manner motivates an alternating optimization algorithm which corresponds to the ALS algorithm in the matrix case. However, for tensors with order greater than or equal to 3, it minimizes an objective function which is different from the commonly used least squares loss. Alternating optimization of this new objective leads to simple updates to the factor matrices with the same asymptotic computational cost as ALS. We show that a subsweep of this algorithm can achieve a superlinear convergence rate for exact CPD when the known rank is not larger than the mode lengths of the input tensor. We verify our theoretical arguments experimentally. We then view the algorithm as optimizing a Mahalanobis distance with respect to each factor with the ground metric dependent on the other factors. This perspective allows us to generalize our approach to interpolate between updates corresponding to the ALS and the new algorithm to manage the tradeoff between stability and fitness of the decomposition. Our experimental results show that for approximating synthetic and real-world tensors, this algorithm and its variants converge to a better conditioned decomposition with comparable and sometimes better fitness as compared to the ALS algorithm.

Key words. tensor decomposition, CP decomposition, alternating least squares, eigenvalues, singular values, Mahalanobis distance, condition number

MSC codes. 15A69, 15A72, 65K10, 65Y20, 65Y04, 68W25

DOI. 10.1137/22M1490739

1. Introduction.

The canonical polyadic or CANDECOMP/PARAFAC (CP) tensor decomposition [21, 24] is used for analysis and compression of multiparameter datasets and is prevalent in tensor methods for scientific simulation [15, 40, 44, 54]. For an order 3 tensor $\boldsymbol{\mathcal{T}}$, indexed as t_{ijk}, a rank-R CP decomposition with factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ is defined as

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!], \quad t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr},$$

where $\boldsymbol{A}$ is indexed as a_{ir}, and similarly for $\boldsymbol{B}$ and $\boldsymbol{C}$. Determining the CP rank and finding an approximate CP decomposition of a tensor, so as to minimize

$$f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \frac{1}{2} \left\| \boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] \right\|_F^2, \tag{1.1}$$

are NP-hard problems [23]. The CP decomposition of a tensor can be computed via various optimization algorithms, such as alternating least squares (ALS) [6, 22, 26, 53]

*Submitted to the journal's Methods and Algorithms for Scientific Computing section April 15, 2022; accepted for publication (in revised form) June 22, 2023; published electronically November 3, 2023.

https://doi.org/10.1137/22M1490739

Funding: This work was supported by the US NSF OAC SSI program grant 1931258.

†Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (navjot2@illinois.edu, solomon2@illinois.edu).

which aims to minimize the objective (1.1) in an alternating manner by considering all except one factor matrix fixed. There have been several attempts to improve the performance of the ALS algorithm [38, 42, 45, 50, 56]. Several methods which aim to minimize (1.1) with respect to all the factor matrices use gradient-based information [1, 46, 48, 57, 60, 63] to update all the factors simultaneously. Another set of methods optimize for all the factors simultaneously by formulating (1.1) as a nonlinear least squares problem by considering the entries of all the factors as variables. In addition to gradient information, these iterative methods use second order information to compute the next step which requires a system solve [46, 62] and can be achieved by an implicit conjugate gradient algorithm [55, 57].

Tensor eigenvalue problems are relevant in the context of solving multilinear systems, simulating quantum systems, exponential data fitting, and many other application areas [49]. However, the study of tensor eigenvalues and tensor singular values is at a relatively nascent stage. [34, 37] provide a definition and introduction to eigenvalues and singular values of a tensor. Computing eigenvalues of a tensor is a hard problem and can be solved via iterative methods for special cases such as computing a subset of eigenvalues of a tensor [32] or computing the real eigenvalues pairs of a real symmetric tensor [16]. The tensor eigenvalue problem is motivated by applications like blind source separation [8] and independent component analysis [25], which also motivate the closely related problem of diagonalizing a tensor. The concept of tensor diagonalization was introduced in [14], where approximate diagonalization of the tensor is considered by minimizing the sum of squares of off-diagonal entries or maximizing the sum of squares of diagonal entries of the tensor. There have been many follow-up works [35, 36, 61, 65] which consider approximate diagonalization of the tensor by invertible and orthogonal transformations.

In this work, we introduce a formulation for computing the singular values and vectors of a tensor by considering a logarithmic penalty function instead of Lagrangian variables [37] and computing the critical points of this function. This formulation, when generalized to computing invariant subspaces of a matrix, leads to diagonalization of the matrix and can be linked to the singular values and vectors of the matrix. When extended to tensors with order greater than or equal to 3, this formulation leads to another notion of diagonalization of the tensor which is different from the one introduced in prior work. The critical points of this new function spectrally diagonalize the tensor; i.e., the transformed equidimensional tensor of mode length R has R elementary eigenvectors with unit eigenvalues. Computing these stationary points alternatively motivates an alternating optimization algorithm for computing the CP decomposition of a tensor.

1.1. Motivation: Eigenvectors via Lagrangian optimization. In the case of low-rank matrix approximation, the Eckart–Young–Mirsky theorem shows that the best low-rank approximation may be obtained from the singular value decomposition. This connection relates low-rank factors to critical points of the bilinear form $f_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{y}$ with $\|\boldsymbol{x}\| \neq 0$, $\|\boldsymbol{y}\| \neq 0$. For tensors of order 3 and higher, tensor singular values have been similarly derived from critical points of multilinear forms. In particular, Lim [37] derives singular vectors and singular values by imposing constraints $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$ and considering the critical points of the Lagrangian function. The same results may be obtained by instead considering a logarithmic interior point barrier function for the constraints, $\|\boldsymbol{x}\| \neq 0$, $\|\boldsymbol{y}\| \neq 0$, so

$$\begin{aligned} f_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{y} - \log(\|\boldsymbol{x}\|\|\boldsymbol{y}\|), \\ \nabla f_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{0} \;&\Rightarrow\; \boldsymbol{M}\boldsymbol{y} = \boldsymbol{x}/\|\boldsymbol{x}\|^2,\; \boldsymbol{M}^T\boldsymbol{x} = \boldsymbol{y}/\|\boldsymbol{y}\|^2. \end{aligned}$$

Consequently, with $\sigma = 1/(\|\boldsymbol{x}\|\|\boldsymbol{y}\|)$ and $\boldsymbol{u} = \boldsymbol{x}/\|\boldsymbol{x}\|$, $\boldsymbol{v} = \boldsymbol{y}/\|\boldsymbol{y}\|$, we have $\boldsymbol{M}\boldsymbol{v} = \sigma\boldsymbol{u}$ and $\boldsymbol{M}^T\boldsymbol{u} = \sigma\boldsymbol{v}$. The use of a coefficient for the barrier function only affects the scaling of any critical point vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$. For tensors of order 3 and higher, tensor singular values can be similarly derived from critical points of

$$f_{\boldsymbol{\mathcal{T}}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}) = \sum_{i_1 \ldots i_N} t_{i_1 \ldots i_N} x_{i_1}^{(1)} \cdots x_{i_N}^{(N)} - \left(\sum_{i=1}^{N} \log(\|\boldsymbol{x}^{(i)}\|) \right),$$

$$\nabla f_{\boldsymbol{\mathcal{T}}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}) = \boldsymbol{0} \;\Rightarrow\; \frac{x_{i_j}^{(j)}}{\|\boldsymbol{x}^{(j)}\|^2} = \sum_{i_1 \ldots \hat{i}_j \ldots i_N} t_{i_1 \ldots i_N} x_{i_1}^{(1)} \cdots \hat{x}_{i_j}^{(j)} \cdots x_{i_N}^{(N)},$$

where $i \ldots \hat{j} \ldots k$ implies j is omitted from the sequence. The use of the 2-norm in the above definitions leads to l^2 singular vectors [37], and with a symmetric tensor and each $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(j)}$ it yields Z-eigenvectors [34]. Choosing another vector norm in $\{3, \ldots, N\}$ leads to other notions of singular vectors and eigenvectors [34].

In the previous literature, there have been several works that show correspondence between CP decomposition and eigenvectors or singular vectors of a tensor. For latent variable models, under certain assumptions, an orthogonal CP decomposition is also an eigendecomposition of a symmetric tensor [3]. It has been shown that orthogonally decomposable symmetric tensors have orthogonal eigenvectors and that the CP decomposition of these tensors is given by orthogonal eigenvectors [51]. An extension of this result to nonsymmetric tensors with orthogonal singular vectors defining the CP decomposition has also been explored [52].

In this work, motivated by an efficient iterative scheme, we consider an extension of the variational notion of one singular vector tuple to many.

1.2. Convergence results. The new alternating update scheme is highly effective at finding an exact CP decomposition, if one exists. In particular, we show that the method achieves a superlinear rate of local convergence to exact CP decompositions if the CP rank is not greater than any mode length of the input tensor. In section 4, we prove that the convergence order is α per subproblem or α^N per sweep of alternating updates, where α is the unique real root of the polynomial $x^{N-1} - \sum_{i=0}^{N-2} x^i$. For $N = 3$, $\alpha = (1 + \sqrt{5})/2$, while for higher N, α increases. A superlinear convergence rate for CP decomposition is also achievable via general optimization algorithms such as Gauss–Newton [55, 57] which does not involve any restrictions on the CP rank. However, the alternating optimization scheme we propose has a much lower per-iteration cost (about the same as ALS, which achieves only linear convergence).

For a given tensor and any choice of rank, the critical points are generally not unique (as in the case of matrices). Theoretical characterization of the conditions under which critical points of (3.3) exist in this scenario remains an open problem, as it requires proving existence of roots of a system of nonlinear equations that have the same number of variables and equations. Note that the problem of proving if the best CP rank approximation exists also requires existence of a solution of system of nonlinear equations. The best CP rank approximation may not exist, which has led to the notion of border rank [31]. Consequently, establishing existence of critical points for our scenario is likely also nontrivial. Therefore, with the assumption that a critical point exists, we show in section 4.1 that the proposed iterative scheme achieves local convergence to it (in this case, at a linear rate).

We have performed numerical experiments to confirm the rate of convergence of the algorithm for computing CP decomposition of different tensors with known rank. The observed rate of convergence values agrees with the theoretical rate of convergence with an error of about 0.2% for order 3 and an error of about 1% for order 4 tensors. The numerical experiments in section 6 also confirm the convergence analysis of the algorithm to stationary points for approximate decomposition of tensors as described in Lemma 4.4.

1.3. Generalizations and experimental evaluation. The proposed algorithm may also be used for approximate CP decomposition but does not minimize the Frobenius norm of the residual directly. Instead, when optimizing for $\boldsymbol{A}$, the algorithm minimizes

$$\left\|(\boldsymbol{I} \otimes \boldsymbol{B}^{\dagger} \otimes \boldsymbol{C}^{\dagger}) \operatorname{vec}(\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!])\right\|_F,$$

where $\operatorname{vec}(\boldsymbol{\mathcal{T}})$ is vectorization of the tensor $\boldsymbol{\mathcal{T}}$ and $\otimes$ is the Kronecker product operation on two matrices. For a fixed residual error in the decomposition, the magnitude of this error metric will generally depend on the conditioning of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. Hence, this alternating minimization procedure tends to converge to well-conditioned factors (and well-conditioned CP decompositions [10, 66]).

We generalize this method by considering a Mahalanobis distance between the input and reconstructed tensors. The original motivation for Mahalanobis distance minimization of tensors came from the work on minimization of Wasserstein distance between tensors for nonnegative CP decomposition [2]. This generalization allows us to reformulate each update to a factor of the above-introduced algorithm as a minimizer of a Mahalanobis distance [19] with the ground metric dependent on the other remaining factors. This reformulation helps extend the introduced algorithm to any CP rank by using the same ground metric. Moreover, we are able to interpolate between the introduced algorithm and ALS by interpolating the ground metric. Our experiments in section 6 suggest that the interpolated updates help manage the tradeoff between fitness and conditioning of the decomposition. We measure conditioning of the decomposition by computing the normalized CP decomposition condition number [10]. The condition number can be computed in an efficient manner for decompositions with small CP rank by reducing the size of the matrix for which the smallest singular value needs to be computed. The reduction in size is performed by removing some components of the matrix which contribute to larger singular value components and henceforth reducing the computational cost significantly. We present a proof of this reduction in Appendix A which is specific to CP decomposition and is different from the proof presented for structured block term decompositions [20]. By using this efficient approach, we are able to track the condition number of the decomposition at each iteration of the algorithms. For synthetic as well as real-world tensors, we observe that by utilizing hybrid updates of the introduced algorithm, we can find decompositions with a condition number lower by a factor as large as 10^4 with a change in relative residual of only about 0.01 when compared to ALS.

2. Background. We introduce the notation and definitions used in the subsequent sections here along with a brief introduction to the ALS algorithm for computing CP decomposition [12, 21].

2.1. Notation and definitions. We use tensor algebra notation in both elementwise form and specialized form for tensor operations [31]. For vectors, bold

lowercase letters are used, e.g., $\boldsymbol{x}$. For matrices, bold uppercase letters are used, e.g., $\boldsymbol{X}$. For tensors, bold calligraphic fonts are used, e.g., $\boldsymbol{\mathcal{X}}$. An order N tensor corresponds to an N-dimensional array with dimensions $s_1 \times \cdots \times s_N$. Elements of vectors, matrices, and tensors are denoted in subscript, e.g., x_i for a vector $\boldsymbol{x}$, x_{ij} for a matrix $\boldsymbol{X}$ and x_{ijkl} for an order 4 tensor $\boldsymbol{\mathcal{X}}$. The ith column of a matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{x}_i$. The mode-n matrix product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{s_1 \times \cdots \times s_N}$ with a matrix $\boldsymbol{A} \in \mathbb{R}^{J \times s_n}$ is denoted by $\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A}$, with the result having dimensions $s_1 \times \cdots \times s_{n-1} \times J \times s_{n+1} \times \cdots \times s_N$. Matricization is the process of reshaping a tensor into a matrix. Given a tensor $\boldsymbol{\mathcal{X}}$, the mode-n matricized version is denoted by $\boldsymbol{X}_{(n)} \in \mathbb{R}^{s_n \times K}$, where $K = \prod_{m=1, m \neq n}^{N} s_m$. We use parenthesized superscripts as labels for different tensors and matrices; e.g., $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$ are different matrices.

The Hadamard product of two matrices $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{I \times J}$ resulting in matrix $\boldsymbol{W} \in \mathbb{R}^{I \times J}$ is denoted by $\boldsymbol{W} = \boldsymbol{U} * \boldsymbol{V}$, where $w_{ij} = u_{ij} v_{ij}$. The inner product of matrices $\boldsymbol{U}, \boldsymbol{V}$ is denoted by $\langle \boldsymbol{U}, \boldsymbol{V} \rangle = \sum_{i,j} u_{ij} v_{ij}$, and similarly for tensors. The outer product of K vectors $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(K)}$ of corresponding sizes $s_1, \ldots, s_K$ is denoted by $\boldsymbol{\mathcal{X}} = \boldsymbol{u}^{(1)} \circ \cdots \circ \boldsymbol{u}^{(K)}$, where $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{s_1 \times \cdots \times s_K}$ is an order K tensor.

For matrices $\boldsymbol{A} \in \mathbb{R}^{I \times K} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_K]$ and $\boldsymbol{B} \in \mathbb{R}^{J \times K} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$, their Khatri–Rao product results in a matrix of size $(IJ) \times K$ defined by $\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1, \ldots, \boldsymbol{a}_K \otimes \boldsymbol{b}_K]$, where $\boldsymbol{a} \otimes \boldsymbol{b}$ denotes the Kronecker product of the two vectors. We define the Mahalanobis norm for a matrix $\boldsymbol{A}$ with ground metric $\boldsymbol{M}$ as $\|\boldsymbol{A}\|_{\boldsymbol{M}}^2 = \operatorname{vec}(\boldsymbol{A})^T \boldsymbol{M} \operatorname{vec}(\boldsymbol{A})$, and similarly for a tensor $\boldsymbol{\mathcal{T}}, \|\boldsymbol{\mathcal{T}}\|_{\boldsymbol{M}}^2 = \operatorname{vec}(\boldsymbol{\mathcal{T}})^T \boldsymbol{M} \operatorname{vec}(\boldsymbol{\mathcal{T}})$. To ease the notation for N Khatri–Rao products, we use $\bigodot_{n=1}^{N} = \boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(1)}$, and similarly for Kronecker products of N matrices, $\bigotimes_{n=1}^{N} \boldsymbol{A}^{(n)} = \boldsymbol{A}^{(N)} \otimes \ldots \otimes \boldsymbol{A}^{(1)}$,$\circledast_{n=1}^{N} \boldsymbol{A}^{(n)} = \boldsymbol{A}^{(N)} * \ldots * \boldsymbol{A}^{(1)}$. We use $\sigma_{\min}(\boldsymbol{P})$ to denote the minimum singular value of the matrix $\boldsymbol{P}$.

2.2. ALS for CP decomposition. The CP tensor decomposition [21, 24] for an input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is denoted by

$$\boldsymbol{\mathcal{X}} \approx \left[\!\!\left[\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(N)} \right]\!\!\right], \quad \text{where} \quad \boldsymbol{A}^{(i)} = \left[\boldsymbol{a}_1^{(i)}, \cdots, \boldsymbol{a}_r^{(i)} \right],$$

and serves to approximate a tensor by a sum of R tensor products of vectors,

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}.$$

It is sometimes useful to normalize the factor matrices so that each column of the factors has a unit 2-norm and the weights are absorbed into a vector $\boldsymbol{d} \in \mathbb{R}^R$, given as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} = \sum_{r=1}^{R} d_r \bar{\boldsymbol{a}}_r^{(1)} \circ \cdots \circ \bar{\boldsymbol{a}}_r^{(N)},$$

where $\bar{\boldsymbol{A}}^{(n)}$ are column normalized for all n and denoted by

$$\boldsymbol{\mathcal{X}} \approx \left[\!\!\left[\boldsymbol{D}; \bar{\boldsymbol{A}}^{(1)}, \ldots, \bar{\boldsymbol{A}}^{(N)} \right]\!\!\right],$$

where $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{d}$ on the diagonal. The CP-ALS method aims to minimize the nonlinear least squares problem

$$f(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}) := \frac{1}{2} \left|\!\left| \boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!] \right|\!\right|_F^2 \tag{2.1}$$

by alternatively minimizing a sequence of least squares problems for each of the factor matrices $\boldsymbol{A}^{(n)}$. This results in linear least squares problems for each row,

$$\boldsymbol{A}^{(n)}_{\text{new}}\boldsymbol{P}^{(n)T} \cong \boldsymbol{X}_{(n)},$$

where the matrix $\boldsymbol{P}^{(n)} \in \mathbb{R}^{D_n \times R}$, where $D_n = \prod_{i=1, i\neq n}^{N} I_i$, is formed by Khatri–Rao products of the other factor matrices,

$$\boldsymbol{P}^{(n)} = \bigodot_{m=1, m\neq n}^{N} \boldsymbol{A}^{(m)}. \tag{2.2}$$

These linear least squares problems are often solved via the normal equations [31],

$$\boldsymbol{A}^{(n)}_{\text{new}}\boldsymbol{\Gamma}^{(n)} \leftarrow \boldsymbol{X}_{(n)}\boldsymbol{P}^{(n)},$$

where $\boldsymbol{\Gamma} \in \mathbb{R}^{R\times R}$ can be computed via

$$\boldsymbol{\Gamma}^{(n)} = \circledast_{i=1, i\neq n}^{N} \boldsymbol{S}^{(i)} \tag{2.3}$$

with each $\boldsymbol{S}^{(i)} = \boldsymbol{A}^{(i)T}\boldsymbol{A}^{(i)}$. The *matricized tensor times Khatri–Rao product* or MTTKRP computation $\boldsymbol{M}^{(n)} = \boldsymbol{X}_{(n)}\boldsymbol{P}^{(n)}$ is the main computational bottleneck of CP-ALS [5]. For a rank-R CP decomposition, this computation has the cost of $O(I^N R)$ if $I_n = I$ for all $n \in \{1, \ldots, N\}$. There have been various developments to optimize computation of MTTKRP, like the dimension tree algorithm [4, 27, 28, 29, 47, 67] for dense tensors and sparse MTTKRP [13] for sparse tensors.

3. Exact CP decomposition. In this section, we provide a motivation for the derivation of the new alternating optimization scheme to compute exact CP decomposition when the rank is known and is not greater than any of the mode lengths of the input tensor. The method is derived by extending the notion of computing singular vectors via Lagrangian optimization by considering more than one vector at a time by computing the fixed points of a more general function. These fixed points lead to a new alternating update scheme which solves a different least squares problem as compared to ALS. This new scheme has a leading order cost which is the same as ALS, depends on the MTTKRP operation, and therefore is efficient in computation.

3.1. Tensor spectral diagonalization via Lagrangian optimization. An equidimensional tensor of order N with mode length s is said to be spectrally diagonalized if it has s unit eigenvalues with s corresponding eigenvectors that are columns of identity. We may obtain a spectral diagonalization of the matrix $\boldsymbol{A}$ by considering the critical points of a generalization of $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{y} = \langle \boldsymbol{A}, \boldsymbol{x}\boldsymbol{y}^T\rangle$ to the matrix case,

$$f(\boldsymbol{X}, \boldsymbol{Y}) = \langle \boldsymbol{A}, \boldsymbol{X}\boldsymbol{Y}^T\rangle \text{ s.t. } \det(\boldsymbol{X}^T\boldsymbol{X}) \neq 0, \det(\boldsymbol{Y}^T\boldsymbol{Y}) \neq 0.$$

Transforming the inequality constraint into a logarithmic barrier function, we obtain

$$\mathcal{L}_f(\boldsymbol{X}, \boldsymbol{Y}) = \langle \boldsymbol{A}, \boldsymbol{X}\boldsymbol{Y}^T\rangle - \frac{1}{2}\big(\log(\det(\boldsymbol{X}^T\boldsymbol{X})) + \log(\det(\boldsymbol{Y}^T\boldsymbol{Y}))\big) \tag{3.1}$$

$$= \operatorname{tr}(\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{Y}) - \frac{1}{2}\operatorname{tr}\big(\log(\boldsymbol{X}^T\boldsymbol{X}) + \log(\boldsymbol{Y}^T\boldsymbol{Y})\big). \tag{3.2}$$

The above equivalence is due to Jacobi's formula, as $\det(\boldsymbol{X}^T\boldsymbol{X}) \neq 0, \det(\boldsymbol{Y}^T\boldsymbol{Y}) \neq 0$. The critical points of $\mathcal{L}_f$ satisfy

$$\boldsymbol{A}\boldsymbol{Y}\boldsymbol{X}^T \cong \boldsymbol{I} \text{ and } \boldsymbol{A}^T\boldsymbol{X}\boldsymbol{Y}^T \cong \boldsymbol{I}.$$

The above least squares equations arise from the fact that derivative of $\frac{1}{2}\operatorname{tr}\log(\boldsymbol{X}^T\boldsymbol{X})$ with respect to $\boldsymbol{X}$ is $\boldsymbol{X}^{\dagger T}$. These critical points diagonalize $\boldsymbol{A}$ in the sense that $\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{Y} = \boldsymbol{I}$. In the case of a tensor $\boldsymbol{\mathcal{T}}$, a critical point $(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)})$ of

(3.3)

$$f(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)}) = \langle \boldsymbol{\mathcal{T}}, [\![\boldsymbol{X}^{(1)}, \ldots \boldsymbol{X}^{(N)}]\!] \rangle \text{ s.t. } \det(\boldsymbol{X}^{(n)T}\boldsymbol{X}^{(n)}) \neq 0 \, \forall n \in \{1, \ldots, N\},$$

$$\mathcal{L}_f(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)}) = \sum_{r=1}^{R} \sum_{i_1 \ldots i_N} t_{i_1 \ldots i_N} x^{(1)}_{i_1 r} \cdots x^{(N)}_{i_N r} - \frac{1}{2} \operatorname{tr}\left(\sum_{i=1}^{N} \log({\boldsymbol{X}^{(i)}}^T \boldsymbol{X}^{(i)}) \right)$$

gives invariant subspaces of $\boldsymbol{\mathcal{T}}$ in the sense that, for all $i \in \{1, \ldots, N\}$,

$$\forall \boldsymbol{v} \in \operatorname{span}\left\{ \bigotimes_{j \neq i} \boldsymbol{x}^{(j)}_1, \ldots, \bigotimes_{j \neq i} \boldsymbol{x}^{(j)}_R \right\}, \quad \boldsymbol{T}_{(i)}\boldsymbol{v} \in \operatorname{span}\left\{ \boldsymbol{x}^{(i)}_1, \ldots, \boldsymbol{x}^{(i)}_R \right\},$$

where $\boldsymbol{T}_{(i)}$ is the mode-i matricization (unfolding) of the tensor $\boldsymbol{\mathcal{T}}$. Note that the objective function in (3.3) is neither convex nor concave with respect to any of the factors $\boldsymbol{X}, \boldsymbol{Y}$, unlike the least squares loss function minimized in ALS which is convex with respect to the factor matrices.

The reconstructed tensor $\tilde{\boldsymbol{\mathcal{T}}}$, where $\tilde{\boldsymbol{\mathcal{T}}} = [\![\boldsymbol{Y}^{(1)}, \ldots \boldsymbol{Y}^{(N)}]\!]$, $\boldsymbol{Y}^{(n)} = {\boldsymbol{X}^{(n)\dagger}}^T$ for all $n \in \{1, \ldots, N\}$,captures the action of $\boldsymbol{T}_{(i)}$ on an invariant subspace as described above; the application of each matricization may be performed with bounded backward error. In section 5.1.1, we show that the backward error is bounded by $\|\boldsymbol{T}_{(i)}\boldsymbol{z}^{\perp}\|$, where $\boldsymbol{z}^{\perp}$ is the projection of $\boldsymbol{z}$ onto the orthogonal complement of the column span of $\bigodot_{j=1, j\neq n}^{N} \boldsymbol{X}^{(j)}$. We show that this bound also holds for ALS, and in general for a family of algorithms based on alternating minimization of Mahalanobis distance [19] between the input and reconstructed tensors.

Another observation regarding the critical points of (3.3) is that each matricization of the tensor reconstructed from a critical point, $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{X}^{(1)}, \ldots \boldsymbol{X}^{(N)}]\!]$, is a right inverse of the corresponding matricization of the input tensor $\boldsymbol{\mathcal{T}}$ when the CP rank is equal to the mode length, and more generally,

$$\boldsymbol{T}_{(i)}\boldsymbol{X}^T_{(i)} = \boldsymbol{\Pi}^{(i)} \quad \forall i \in \{1, \ldots, N\},$$

where each $\boldsymbol{\Pi}^{(i)}$ is a projector onto the column space of $\boldsymbol{X}^{(i)}$. This $\boldsymbol{\mathcal{X}}$ satisfies some but not all of the properties of previously proposed generalizations of the Moore–Penrose inverse to tensors [36, 58].

Further, the critical point gives a transformation that spectrally diagonalizes $\boldsymbol{\mathcal{T}}$,

$$z_{j_1 \ldots j_N} = \sum_{i_1 \ldots i_N} t_{i_1 \ldots i_N} x^{(1)}_{i_1 j_1} \cdots x^{(N)}_{i_N j_N},$$

so that $\boldsymbol{\mathcal{Z}}$ has R eigenvectors that are elementary vectors with unit eigenvalues (for any tensor eigenvector/eigenvalue definition, i.e., l^p eigenvector for any choice of p [34]), since

$$z_{j_k j_k \ldots j_p j_k \ldots j_k} = \delta_{j_k j_p} \quad \forall p \neq k \in \{1, \ldots, N\}.$$

Beyond these properties, we show that $\boldsymbol{X}^{(1)\dagger^T}, \ldots, \boldsymbol{X}^{(N)\dagger^T}$ may be used to obtain an exact CP decomposition or an effective low-rank approximate CP decomposition.

3.2. Alternating optimization method. The critical points defined above may be computed efficiently by a method similar to ALS. For a CP decomposition $[\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$ of tensor $\boldsymbol{\mathcal{T}}$, ALS solves a set of overdetermined linear equations at each step to minimize the Frobenius norm error relative to one factor; e.g., it solves for $\boldsymbol{A}$ in

$$(\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{A}^T \cong \boldsymbol{T}_{(1)}^T.$$

The update rule for each subproblem may be written as a product of the pseudoinverse of the Khatri–Rao product of two factors and an unfolding of the tensor, i.e.,

$$\boldsymbol{A} = \boldsymbol{T}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})^{\dagger T}.$$

Some of the major advantages of the ALS algorithm are its guaranteed monotonic decrease in residual, low per-iteration computational cost and its amenability to parallelization. It has been shown that ALS achieves linear local convergence to minima of the CP residual norm under certain conditions [64].

To obtain a critical point in the high order tensor function (3.3), we propose a different alternating update scheme, which finds the solution $\boldsymbol{U}$ to the linear least squares problem,

$$(\boldsymbol{T}_{(1)}(\boldsymbol{W} \odot \boldsymbol{V}))\boldsymbol{U}^T \cong \boldsymbol{I}.$$

With $\boldsymbol{A}^T = \boldsymbol{U}^\dagger$, $\boldsymbol{B}^T = \boldsymbol{V}^\dagger$, and $\boldsymbol{C}^T = \boldsymbol{W}^\dagger$, we observe that the update is similar to that of ALS,

$$\boldsymbol{A} = \boldsymbol{T}_{(1)}(\boldsymbol{C}^{\dagger T} \odot \boldsymbol{B}^{\dagger T}).$$

A stationary point of this alternating update scheme provides a critical point of (3.3).

We first provide a complete description of the alternating update scheme proposed above. To compute the decomposition of a tensor $\boldsymbol{\mathcal{X}}$, the algorithm maintains a CP decomposition given by

$$[\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]$$

and updates each $\boldsymbol{A}^{(n)}$ in an alternating manner,

$$\boldsymbol{A}^{(n)} = \boldsymbol{X}_{(n)} \left(\bigodot_{m=1, m\neq n}^{N} \boldsymbol{A}^{(m)\dagger T} \right). \tag{3.4}$$

This update may be written in elementwise form in terms of the pseudoinverses $\boldsymbol{U}^{(m)} = \boldsymbol{A}^{(m)\dagger}$ as

$$a_{i_n r}^{(n)} = \sum_{i_1 \ldots \hat{i}_n \ldots i_N} x_{i_1 \ldots i_N} \prod_{m=1, m\neq n}^{N} u_{r i_m}^{(m)}.$$

Algorithm 3.1 details each sweep of such updates. As with the ALS, it is advisable to recalibrate the norms of the columns of each factor before the subsweep corresponding to the nth factor so that $\|\boldsymbol{a}_r^{(k)}\| = 1$ for all $k \neq n$ and r. With this recalibration, a valid convergence criterion is to check whether the magnitude of change

Algorithm 3.1 Basic description of the new alternating update scheme.

Input: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, rank $R \leq I_n \forall n \in \{1, \ldots, N\}$
Initialize $\{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}\}$ so each $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ is random
while not converged **do**
  for $n \in \{1, \ldots, N\}$ **do**
    $\boldsymbol{A}^{(n)} = \boldsymbol{X}_{(n)}^{(N)} \left(\bigodot_{m=1, m \neq n}^{N} \boldsymbol{A}^{(m)\dagger T} \right)$
  end for
end while
return factor matrices $\{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}\}$

in the factors exceeds a predefined threshold at each sweep. The method is invariant to the rescaling in exact arithmetic, but this calibration helps reduce the effects of round-off error as in [64]. This calibration is also cost efficient, since the pseudoinverse of only one matrix changes per subsweep. This makes the algorithm accessible to all the optimizations involved in computing the MTTKRP in each subsweep of ALS such as the dimension tree algorithm [4, 27, 28, 29, 47, 67].

3.3. Cost analysis. The cost of each sweep of Algorithm 3.1 corresponds to the cost of computing the pseudoinverse of each factor, as well as a set of N MTTKRP operations computed across different modes. For computing a set of MTTKRP operations for a sweep of the algorithm, a dimension tree [4, 28, 47] or multisweep dimension tree [39] may be used as in the ALS algorithm. Dimension trees provide an efficient way to compute the set of MTTKRP in one sweep by storing and reusing partial computations performed in computing MTTKRP in each subsweep of the sweep. A multisweep dimension tree improves the computation by reusing intermediates across sweeps, reducing the cost per sweep even further. The overall per-sweep cost with a multisweep dimension tree [39] is then given by

$$\frac{2N}{N-1}\left(\prod_{n=1}^{N} I_n\right)R + O\left(\left(\sum_{n=1}^{N} I_n\right)R^2\right).$$

The cost of an ALS sweep with a multisweep dimension tree [39] is

$$\frac{2N}{N-1}\left(\prod_{n=1}^{N} I_n\right)R + O(NR^3),$$

which is less expensive, as solving an overdetermined system via normal equations is cheaper than computing the pseudoinverse of a matrix. If $\boldsymbol{\mathcal{X}}$ is sparse, the method can benefit from existing work on efficiently performing MTTKRP with a sparse tensor [13] and therefore has the same leading order cost per sweep as that of ALS for sparse tensors as well.

4. Convergence rate for exact decomposition. In this section, we theoretically analyze the asymptotic rate of local convergence of Algorithm 3.1 for when an exact CP decomposition of rank R less than equal to the length of all modes of the tensor exists. To derive the rate of convergence for an exact decomposition, we relate the distance between the computed factor in each subproblem and the true factor, to the error in the other factor matrices. The following lemma states the error in

computing $\boldsymbol{A}^{(N)}$ but can be trivially extended to any $\boldsymbol{A}^{(n)}$ for $n \in \{1,\dots,n\}$. We consider the error in the normalized factor and the error in the magnitude of CP decomposition components modulo column scaling.

LEMMA 4.1. *Suppose* $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{D}; \boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!]$, *where each* $\boldsymbol{A}^{(i)} \in \mathbb{R}^{s_i \times R}$ *with* $s_i \geq R$ *is full rank and has normalized columns, i.e.,* $\|\boldsymbol{a}_j^{(i)}\|_2 = 1$ *for all* i,j *and* $\bar{\boldsymbol{A}}^{(n)} = \boldsymbol{A}^{(n)} + \boldsymbol{\Delta}^{(n)}$ *also has normalized columns and satisfies* $\|\boldsymbol{\Delta}^{(n)}\|_F = \epsilon_n$ *for* $n = 1,\dots,N-1$. *Then there exists* $\epsilon > 0$ *such that if* $\epsilon_n < \epsilon$ *for* $n = 1,\dots,N-1$, *then*

$$\tilde{\boldsymbol{A}}^{(N)} = \boldsymbol{X}_{(N)} \left(\bar{\boldsymbol{A}}^{(1)\dagger T} \odot \cdots \odot \bar{\boldsymbol{A}}^{(N-1)\dagger T} \right),$$

where $\tilde{\boldsymbol{A}}^{(N)} = \bar{\boldsymbol{A}}^{(N)} \bar{\boldsymbol{D}}$ *ensures that* $\bar{\boldsymbol{A}}^{(N)}$ *are normalized and satisfies*

$$\|\bar{\boldsymbol{A}}^{(N)} - \boldsymbol{A}^{(N)}\|_F = O\left(\prod_{n=1}^{N-1} \epsilon_n \right)$$
$$\text{and} \quad \|\bar{\boldsymbol{D}} - \boldsymbol{D}\|_F = O(\epsilon).$$

Proof. Let $\epsilon < 1$ and $\epsilon < \min_n(\sigma_{\min}(\boldsymbol{A}^{(n)}))$ for each n; therefore $\bar{\boldsymbol{A}}^{(n)}$ is full rank for each $n = 1,\dots,N-1$. Substituting the decomposition of $\boldsymbol{\mathcal{X}}$ into the computed solution, we obtain

$$\begin{aligned}
\tilde{\boldsymbol{A}}^{(N)} &= \boldsymbol{A}^{(N)} \boldsymbol{D} \Big((\bar{\boldsymbol{A}}^{(1)\dagger}(\bar{\boldsymbol{A}}^{(1)} - \boldsymbol{\Delta}^{(1)})) * \cdots * (\bar{\boldsymbol{A}}^{(N-1)\dagger}(\bar{\boldsymbol{A}}^{(N-1)} - \boldsymbol{\Delta}^{(N-1)})) \Big)^T \\
&= \boldsymbol{A}^{(N)} \boldsymbol{D} \Big((\boldsymbol{I} - \bar{\boldsymbol{A}}^{(1)\dagger} \boldsymbol{\Delta}^{(1)}) * \cdots * (\boldsymbol{I} - \bar{\boldsymbol{A}}^{(N-1)\dagger} \boldsymbol{\Delta}^{(N-1)}) \Big)^T \\
&= \boldsymbol{A}^{(N)} \boldsymbol{D} \Big(\boldsymbol{S} + (-1)^{N-1} \bar{\boldsymbol{A}}^{(1)\dagger} \boldsymbol{\Delta}^{(1)} * \cdots * \bar{\boldsymbol{A}}^{(N-1)\dagger} \boldsymbol{\Delta}^{(N-1)} \Big)^T,
\end{aligned}$$

where $\boldsymbol{S}$ includes all cross terms of the Hadamard products, which must be diagonal since any such term includes a Hadamard product with an identity matrix. Since

$$\|\boldsymbol{I} - \boldsymbol{S}\|_F = O(\max_n(\epsilon_n)) = O(\epsilon),$$

$\boldsymbol{S}$ is full rank for sufficiently small ϵ. Let

$$\boldsymbol{\Delta} = \Big((-1)^{N-1} \bar{\boldsymbol{A}}^{(1)\dagger} \boldsymbol{\Delta}^{(1)} * \cdots * \bar{\boldsymbol{A}}^{(N-1)\dagger} \boldsymbol{\Delta}^{(N-1)} \Big)^T.$$

Now, the norm calibration diagonal matrix is defined so that $\bar{d}_{ii} = \|\tilde{\boldsymbol{a}}_i^{(N)}\|_2$. Since

$$\tilde{\boldsymbol{A}}^{(N)} = \boldsymbol{A}^{(N)} \boldsymbol{D}(\boldsymbol{S} + \boldsymbol{\Delta}) = \boldsymbol{A}^{(n)} \boldsymbol{D} + \boldsymbol{A}^{(n)} \boldsymbol{D}(\boldsymbol{S} + \boldsymbol{\Delta} - \boldsymbol{I})$$

and $\|\boldsymbol{S} + \boldsymbol{\Delta} - \boldsymbol{I}\|_2 = O(\epsilon)$, we have

$$\begin{aligned}
\bar{d}_{ii} &= \|\tilde{\boldsymbol{a}}_i^{(N)}\|_2 \leq \|\boldsymbol{a}_i^{(N)}\|_2 d_{ii} + \|\boldsymbol{A}^{(N)} \boldsymbol{D}\|_F \|\boldsymbol{S} + \boldsymbol{\Delta} - \boldsymbol{I}\|_2 \\
&= \|\boldsymbol{a}_i^{(N)}\|_2 d_{ii} + O(\epsilon) = d_{ii} + O(\epsilon).
\end{aligned}$$

Consequently, $\|\bar{\boldsymbol{D}} - \boldsymbol{D}\|_2 = O(\epsilon)$. Further, we can obtain a tighter bound (in terms of $O(\|\boldsymbol{\Delta}\|_F)$ instead of $O(\epsilon)$) by considering $\tilde{\boldsymbol{A}}^{(N)} = \boldsymbol{A}^{(N)} \boldsymbol{D} \boldsymbol{S}(\boldsymbol{I} + \boldsymbol{S}^{-1} \boldsymbol{\Delta})$, so

$$\bar{d}_{ii} = \|\tilde{\boldsymbol{a}}_i^{(N)}\|_2 \leq \|\boldsymbol{a}_i^{(N)}\|_2 d_{ii} s_{ii} + \|\boldsymbol{A}^{(N)} \boldsymbol{D} \boldsymbol{\Delta}\|_F = d_{ii} s_{ii} + \|\boldsymbol{D}\|_2 \|\boldsymbol{\Delta}\|_F.$$

This bound allows us to get the desired result for the error in the factor matrices,

$$\begin{aligned}\left\|\boldsymbol{A}^{(N)}-\bar{\boldsymbol{A}}^{(N)}\right\|_F &= \left\|\boldsymbol{A}^{(N)}-\tilde{\boldsymbol{A}}^{(N)}\bar{\boldsymbol{D}}^{-1}\right\|_F = \left\|\boldsymbol{A}^{(N)}-\boldsymbol{A}^{(N)}\boldsymbol{D}\boldsymbol{S}(\boldsymbol{I}+\boldsymbol{S}^{-1}\boldsymbol{\Delta})\bar{\boldsymbol{D}}^{-1}\right\|_F \\ &= O\left(\|\boldsymbol{I}-\boldsymbol{D}\boldsymbol{S}(\boldsymbol{I}+\boldsymbol{S}^{-1}\boldsymbol{\Delta})\bar{\boldsymbol{D}}^{-1}\|_F\right). \end{aligned} \tag{4.1}$$

Since we have that $\|\bar{\boldsymbol{D}}-\boldsymbol{D}\boldsymbol{S}\|_F = \|\boldsymbol{D}\|_2\|\boldsymbol{\Delta}\|_F$,

$$\begin{aligned}\left\|\boldsymbol{I}-\boldsymbol{D}\boldsymbol{S}(\boldsymbol{I}+\boldsymbol{S}^{-1}\boldsymbol{\Delta})\bar{\boldsymbol{D}}^{-1}\right\|_F &= \left\|\boldsymbol{I}-\boldsymbol{D}\boldsymbol{S}\bar{\boldsymbol{D}}^{-1}+\boldsymbol{D}\boldsymbol{\Delta}\bar{\boldsymbol{D}}^{-1}\right\|_F \\ &= \left\|\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}^{-1}-\boldsymbol{D}\boldsymbol{S}\bar{\boldsymbol{D}}^{-1}+\boldsymbol{D}\boldsymbol{\Delta}\bar{\boldsymbol{D}}^{-1}\right\|_F \\ &= O(\|\boldsymbol{\Delta}\|_F)+\left\|\boldsymbol{D}\boldsymbol{\Delta}\bar{\boldsymbol{D}}^{-1}\right\|_F = O(\|\boldsymbol{\Delta}\|_F).\end{aligned}$$

Since $\|\boldsymbol{\Delta}\|_F = O(\prod_{n=1}^{N-1}\epsilon_n)$, this completes the proof. ∎

The above lemma states that in Algorithm 3.1, the error in the updated CP decomposition factor relative to the true CP decomposition factor is bounded by the product of errors in the previous $N-1$ factors. Using this error bound, we derive the convergence rate for Algorithm 3.1.

LEMMA 4.2. *For any algorithm where the error in the update is of the order of the product of errors in the previous k updates, the rate of convergence is equal to the positive root of the polynomial $\alpha^k - \sum_{i=0}^{k-1}\alpha^i$.*

Proof. Let the error at the nth iteration be given as e_n. The worst case error at the nth iteration then satisfies the following due to Lemma 4.1:

$$e_n = L\prod_{i=1}^{k} e_{n-i}, \quad \text{where } L \text{ is a constant.} \tag{4.2}$$

The above recurrence can be solved by assuming that the error satisfies the following asymptotic relation:

$$e_n = Ce_{n-1}^{\alpha}, \tag{4.3}$$

where C is some constant and α is the rate of convergence. From (4.2) and (4.3),

$$\begin{aligned} Ce_{n-1}^{\alpha} &= L\prod_{i=1}^{k} e_{n-i}, \\ \text{by assumption in (4.3), } C^{\frac{1}{\alpha^p}}e_{n-1-p} &= e_{n-1}^{\frac{1}{\alpha^p}} \quad \forall p\in\{1,\dots k\}, \\ \frac{C^{\sum_{i=0}^{k-1}\frac{1}{\alpha^i}}}{L} &= e_{n-1}^{(-\alpha+\sum_{i=0}^{k-1}\frac{1}{\alpha^i})}. \end{aligned}$$

Since the left-hand side is constant for $n\to\infty$ due to the assumed convergence rate,

$$\alpha^k - \sum_{i=0}^{k-1}\alpha^i = 0.$$ ∎

Now, with all the pieces together we can show that for exact CP decomposition, Algorithm 3.1 locally converges at a rate which is given in the following theorem.

THEOREM 4.3. *Suppose $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{D};\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!]$, where each $\boldsymbol{A}^{(i)}\in\mathbb{R}^{s_i\times R}$ with $s_i \geq R$ is full rank and has normalized columns. Algorithm* 3.1 *for computing the*

exact CP decomposition of $\boldsymbol{\mathcal{X}}$ converges locally with a rate of convergence equal to α^N, where α is the unique real root of the polynomial $x^{N-1} - \sum_{i=0}^{N-2} x^i$.

Proof. Consider the computation of the CP decomposition of the tensor $\boldsymbol{\mathcal{X}}$ with exact rank R and initial guess $\bar{\boldsymbol{A}}^{(i)}$ with normalized columns such that $\bar{\boldsymbol{A}}^{(i)} = \boldsymbol{A}^{(i)} + \boldsymbol{\Delta}^{(i)}$ with $\|\boldsymbol{\Delta}^{(i)}\|_2 < \epsilon$ sufficiently small for $i = 1, \dots, N-1$ (as described in Lemma 4.1). Let the error in CP decomposition at the nth iteration be given as

$$E_n = \max\left\{\|\bar{\boldsymbol{D}} - \boldsymbol{D}\|_F, \|\bar{\boldsymbol{A}}^{(1)} - \boldsymbol{A}^{(1)}\|_F, \dots, \|\bar{\boldsymbol{A}}^{(N)} - \boldsymbol{A}^{(N)}\|_F\right\}.$$

Also, let the error in the kth subiteration of the nth iteration be given as $\epsilon_n^{(k)}$. From Lemma 4.1, we know that the error in CP decomposition is bounded by the maximum error in factor matrices. Since the error decreases at each subiteration, there exists n such that $E_n = O(\epsilon_n^{(1)})$.

From Lemma 4.1, we know that the error in a subsweep of the algorithm modulo the column scaling is of the order of the product of errors in previous $N-1$ subsweeps; therefore the error at the $(n+1)$th iteration is bounded by the error in the first factor matrix, given by

$$E_{n+1} = O\left(\prod_{k=1}^{N-1} \epsilon_n^{(k)}\right).$$

By using Lemma 4.2, we know that the error in a subiteration is given by the following recurrence, where α is the positive root of $x^{N-1} - \sum_{i=0}^{N-2} x^i$:

$$\epsilon_n^{(k+1)} = O\big((\epsilon_n^{(k)})^{\alpha}\big) \quad \forall\, k = 1, \dots, N-1.$$

Therefore, the error at the $(n+1)$th iteration of Algorithm 3.1 can be expressed as

$$E_{n+1} = O\left(\prod_{i=1}^{N-1} \epsilon_1^{\alpha^i}\right) = O\left(E_n^{\sum_{i=1}^{N-1} \alpha^i}\right).$$

The fact that α is a root of the polynomial $x^{N-1} - \sum_{i=0}^{N-2} x^i$ implies that $\alpha^{N-1} - \sum_{i=0}^{N-2} \alpha^i = 0$; that is, $\sum_{i=1}^{N-1} \alpha^i = \alpha^N$. Therefore,

$$E_{n+1} = O\Big(E_n^{\alpha^N}\Big). \qquad \square$$

This completes the proof to show that Algorithm 3.1 locally converges superlinearly for exact CP rank cases. We verify our theoretical results in the section 6.

4.1. Convergence to other stationary points. The result in Theorem 4.3 can be generalized to the case where a tensor $\boldsymbol{\mathcal{X}}$ can be represented as the sum of two tensors, $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{E}}$, where $\boldsymbol{\mathcal{T}}$ has an underlying CP decomposition structure of rank R and $\boldsymbol{\mathcal{E}}$ has a CP decomposition that is mostly orthogonal to the decomposition of $\boldsymbol{\mathcal{T}}$. For such an input tensor $\boldsymbol{\mathcal{X}}$, we show that Algorithm 3.1 with CP rank R locally converges to the underlying CP factors, provided that a stationary point associated with $\boldsymbol{\mathcal{T}}$ exists. We analyze the convergence rate of the algorithm and show that this is a generalization of the previous result, since we converge to a subset of the CP factors with same convergence rate as in Theorem 4.3, if the factors of $\boldsymbol{\mathcal{T}}$ are in the orthogonal complement of the column space of corresponding CP factors of $\boldsymbol{\mathcal{E}}$. We show that the error in the normalized factor matrix is a summation of two terms which have separate upper bounds. We keep these separate since the upper bounds are in a different parameter space for each term.

LEMMA 4.4. *For a given tensor $\boldsymbol{\mathcal{X}}$, assume there exists a stationary point of Algorithm* 3.1 *yielding a positive diagonal matrix and factors $(\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)})$, where each $\boldsymbol{A}^{(i)} \in \mathbb{R}^{s_i \times R}$ with $s_i \geq R$ is full rank and has normalized columns; i.e., $\|\boldsymbol{a}_j^{(i)}\|_2 = 1$ for all i,j, and their pseudoinverse transposes $(\boldsymbol{U}^{(1)},\dots,\boldsymbol{U}^{(N)})$, so $\boldsymbol{A}^{(n)\dagger} = \boldsymbol{U}^{(n)T}$. The stationary point conditions imply that for all $n \in \{1,\dots,N\}$, we have*

$$\boldsymbol{A}^{(n)}\boldsymbol{D} = \boldsymbol{X}_{(n)}\left(\bigodot_{m=1,m\neq n}^{N} \boldsymbol{U}^{(m)}\right).$$

Further, assume that $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{D};\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}]\!] + [\![\hat{\boldsymbol{D}};\hat{\boldsymbol{A}}^{(1)},\dots,\hat{\boldsymbol{A}}^{(N)}]\!]$ with $\|\hat{\boldsymbol{A}}^{(n)T}\boldsymbol{U}^{(n)}\|_F \leq \epsilon_\perp$, and define $k = \|\hat{\boldsymbol{D}}\|_2\|\boldsymbol{D}^{-1}\|_2$. Given approximations $\tilde{\boldsymbol{A}}^{(n)} = \boldsymbol{A}^{(n)} + \boldsymbol{\Delta}_A^{(n)}$ and $\tilde{\boldsymbol{U}}^{(n)} = \tilde{\boldsymbol{A}}^{(n)\dagger T} = \boldsymbol{U}^{(n)} + \boldsymbol{\Delta}_U^{(n)}$ with normalized columns, there exists $\epsilon > 0$ such that if $\|\boldsymbol{\Delta}_A^{(n)}\|_F, \|\boldsymbol{\Delta}_U^{(n)}\|_F \leq \epsilon_n \leq \epsilon$ for all $n \in \{1,\dots,N-1\}$, then

$$\tilde{\boldsymbol{A}}^{(N)} = \boldsymbol{X}_{(N)}\left(\tilde{\boldsymbol{U}}^{(1)} \odot \cdots \odot \tilde{\boldsymbol{U}}^{(N-1)}\right)$$

satisfies $\|\tilde{\boldsymbol{A}}^{(N)}\bar{\boldsymbol{D}}^{-1} - \boldsymbol{A}^{(N)}\|_F = P + Q$, where $P = O(\prod_{i=1}^{N-1}\epsilon_i)$, $Q = O(k\epsilon\epsilon_\perp)$, and $\bar{\boldsymbol{D}}$ normalizes $\tilde{\boldsymbol{A}}^{(N)}$, i.e., $\bar{d}_{ii} = \|\tilde{\boldsymbol{a}}_i^{(N)}\|_2$. Further, $\|\bar{\boldsymbol{D}} - \boldsymbol{D}\|_F = O(\epsilon)$.

Proof. We expand the update as follows:

$$\begin{aligned}
\tilde{\boldsymbol{A}}^{(N)} &= \left([\![\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}\boldsymbol{D}]\!] + [\![\hat{\boldsymbol{A}}^{(1)},\dots,\hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}]\!]\right)_{(N)}\left(\tilde{\boldsymbol{U}}^{(1)} \odot \cdots \odot \tilde{\boldsymbol{U}}^{(N-1)}\right)\\
&= \boldsymbol{A}^{(N)}\boldsymbol{D}\left(\boldsymbol{A}^{(1)T}\tilde{\boldsymbol{U}}^{(1)} * \cdots * \boldsymbol{A}^{(N-1)T}\tilde{\boldsymbol{U}}^{(N-1)}\right)\\
&\quad + \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left(\hat{\boldsymbol{A}}^{(1)T}\tilde{\boldsymbol{U}}^{(1)} * \cdots * \hat{\boldsymbol{A}}^{(N-1)T}\tilde{\boldsymbol{U}}^{(N-1)}\right)\\
&= \boldsymbol{A}^{(N)}\boldsymbol{D}\left(\left(\boldsymbol{I} + \boldsymbol{A}^{(1)T}\boldsymbol{\Delta}_U^{(1)}\right) * \cdots * \left(\boldsymbol{I} + \boldsymbol{A}^{(N-1)T}\boldsymbol{\Delta}_U^{(N-1)}\right)\right)\\
&\quad + \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\Big(\left(\hat{\boldsymbol{A}}^{(1)T}\boldsymbol{U}^{(1)} + \hat{\boldsymbol{A}}^{(1)T}\boldsymbol{\Delta}_U^{(1)}\right) * \cdots * \Big(\hat{\boldsymbol{A}}^{(N-1)T}\boldsymbol{U}^{(N-1)}\\
&\quad + \hat{\boldsymbol{A}}^{(N-1)T}\boldsymbol{\Delta}_U^{(N-1)}\Big)\Big).
\end{aligned}$$

By the stationary point condition, we have that

$$\hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left(\hat{\boldsymbol{A}}^{(1)T}\boldsymbol{U}^{(1)} * \cdots * \hat{\boldsymbol{A}}^{(N-1)T}\boldsymbol{U}^{(N-1)}\right) = \boldsymbol{0}.$$

Consequently, the error reduces to the cross terms of the summations, i.e.,[1]

$$\begin{aligned}
&\tilde{\boldsymbol{A}}^{(N)} - \boldsymbol{A}^{(N)}\boldsymbol{D}\\
&\quad = \boldsymbol{A}^{(N)}\boldsymbol{D}\left[\mathop{\circledast}_{n=1}^{N}\boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)} + \boldsymbol{I} * \sum_{k=1}^{N-1}\left(\sum_{\{m_1,\dots,m_k\}\subset\{1,\dots,N\}}\mathop{\circledast}_{l=1}^{k}\boldsymbol{A}^{(m_l)T}\boldsymbol{\Delta}_U^{(m_l)}\right)\right]\\
&\quad + \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left[\sum_{k=1}^{N-1}\left(\sum_{\substack{\{m_1,\dots,m_k\}\subset\{1,\dots,N\},\\ \{w_1,\dots,w_{N-k}\}=\{1,\dots,N\}\setminus\{m_1,\dots,m_k\}}}\mathop{\circledast}_{l=1}^{k}\hat{\boldsymbol{A}}^{(m_l)T}\boldsymbol{U}^{(m_l)}\mathop{\circledast}_{l=1}^{N-k}\hat{\boldsymbol{A}}^{(w_l)T}\boldsymbol{\Delta}_U^{(w_l)}\right)\right].
\end{aligned}$$

[1]For $N = 3$, the right-hand side of this formula is

$$\begin{aligned}
&\boldsymbol{A}^{(3)}\boldsymbol{D}\left[\boldsymbol{A}^{(1)T}\boldsymbol{\Delta}_U^{(1)} * \boldsymbol{A}^{(2)T}\boldsymbol{\Delta}_U^{(2)} + \boldsymbol{I} * (\boldsymbol{A}^{(1)T}\boldsymbol{\Delta}_U^{(1)} + \boldsymbol{A}^{(2)T}\boldsymbol{\Delta}_U^{(2)})\right]\\
&\quad + \hat{\boldsymbol{A}}^{(3)}\hat{\boldsymbol{D}}\left[\hat{\boldsymbol{A}}^{(1)T}\boldsymbol{U}^{(1)} * \hat{\boldsymbol{A}}^{(2)T}\boldsymbol{\Delta}_U^{(2)} + \hat{\boldsymbol{A}}^{(2)T}\boldsymbol{U}^{(2)} * \hat{\boldsymbol{A}}^{(1)T}\boldsymbol{\Delta}_U^{(1)}\right].
\end{aligned}$$

First, since each error term has Frobenius norm $O(\|\boldsymbol{\Delta}^{(n)}\|) = O(\epsilon)$, the column norms of $\tilde{\boldsymbol{A}}^{(N)}$ will yield $\bar{\boldsymbol{D}}$ with $\|\bar{\boldsymbol{D}} - \boldsymbol{D}\|_F = O(\epsilon)$. Further, in order to get the error bound on $\tilde{\boldsymbol{A}}^{(N)}$, we consider the diagonal matrix,

$$\boldsymbol{S} = \boldsymbol{I} + \boldsymbol{I} * \sum_{k=1}^{N-1} \left(\sum_{\{m_1,\ldots,m_k\} \subset \{1,\ldots,N\}} \overset{k}{\underset{l=1}{\circledast}} \boldsymbol{A}^{(m_l)T} \boldsymbol{\Delta}_U^{(m_l)} \right).$$

We define $\boldsymbol{S}$ as above in order to show that the column norms of of $\tilde{\boldsymbol{A}}^{(N)}$, i.e., $\bar{\boldsymbol{D}}$, are close to $\boldsymbol{DS}$. Using the above definition of $\boldsymbol{S}$, we have

$$\begin{aligned}
&\boldsymbol{A}^{(N)}\boldsymbol{D} + \boldsymbol{A}^{(N)}\boldsymbol{D}\left[\overset{N}{\underset{n=1}{\circledast}} \boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)} + \boldsymbol{I} * \sum_{k=1}^{N-1} \left(\sum_{\{m_1,\ldots,m_k\} \subset \{1,\ldots,N\}} \overset{k}{\underset{l=1}{\circledast}} \boldsymbol{A}^{(m_l)T}\boldsymbol{\Delta}_U^{(m_l)} \right) \right] \\
&\quad = \boldsymbol{A}^{(N)}\boldsymbol{DS} + \boldsymbol{A}^{(N)}\boldsymbol{DS}\left[\boldsymbol{S}^{-1} \overset{N}{\underset{n=1}{\circledast}} \boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)} \right].
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\tilde{\boldsymbol{A}}^{(N)} - \boldsymbol{A}^{(N)}\boldsymbol{DS} &= \boldsymbol{A}^{(N)}\boldsymbol{DS}\left[\boldsymbol{S}^{-1} \overset{N}{\underset{n=1}{\circledast}} \boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)} \right] \\
&+ \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left[\sum_{k=1}^{N-1} \left(\sum_{\substack{\{m_1,\ldots,m_k\} \subset \{1,\ldots,N\}, \\ \{w_1,\ldots,w_{N-k}\} = \{1,\ldots,N\} \setminus \{m_1,\ldots,m_k\}}} \overset{k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(m_l)T}\boldsymbol{U}^{(m_l)} \overset{N-k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(w_l)T}\boldsymbol{\Delta}_U^{(w_l)} \right) \right],
\end{aligned}$$

$$\|\tilde{\boldsymbol{A}}^{(N)} - \boldsymbol{A}^{(N)}\boldsymbol{DS}\|_F = \|\hat{\boldsymbol{P}}\|_F + \|\hat{\boldsymbol{Q}}\|_F,$$

where $\|\hat{\boldsymbol{P}}\|_F = O\big(\prod_{i=1}^{N-1} \epsilon_i\big)$ and $\|\hat{\boldsymbol{Q}}\|_F = O(\epsilon\epsilon_\perp)$. The first term in the above equations is a Hadamard product of terms $\boldsymbol{A}^{(i)T}\boldsymbol{\Delta}_U^{(i)}$ and therefore leads to an upper bound of $O\big(\prod_{i=1}^{N-1} \epsilon_i\big)$. The second term in the above equations is a summation of terms with products $\big(\hat{\boldsymbol{A}}^{(i)T}\boldsymbol{U}^{(i)}\big)\hat{\boldsymbol{A}}^{(j)T}\boldsymbol{\Delta}_U^{(j)}$ which results in an upper bound of $O(\epsilon\epsilon_\perp)$.

The same bound follows for each column, so the column norms of $\tilde{\boldsymbol{A}}^{(N)}$, $\bar{\boldsymbol{D}}$ satisfy

$$\|\bar{\boldsymbol{D}} - \boldsymbol{DS}\|_F = \|\bar{\boldsymbol{P}}\|_F + \|\bar{\boldsymbol{Q}}\|_F,$$

where $\|\bar{\boldsymbol{P}}\|_F = O\big(\prod_{i=1}^{N-1} \epsilon_i\big)$ and $\|\bar{\boldsymbol{Q}}\|_F = O(\epsilon\epsilon_\perp)$.

Now that we have shown that $\tilde{\boldsymbol{A}}^{(N)}$ is close to $\boldsymbol{A}^{(N)}\boldsymbol{DS}$ and that its column norms ($\bar{\boldsymbol{D}}$) are close to $\boldsymbol{DS}$, we obtain the final error bound after normalization,

$$\begin{aligned}
\tilde{\boldsymbol{A}}^{(N)}\bar{\boldsymbol{D}}^{-1} - \boldsymbol{A}^{(N)}\boldsymbol{DS}\bar{\boldsymbol{D}}^{-1} &= \boldsymbol{A}^{(N)}\boldsymbol{D}\left[\overset{N}{\underset{n=1}{\circledast}} \boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)} \right]\bar{\boldsymbol{D}}^{-1} \\
&+ \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left[\sum_{k=1}^{N-1} \left(\sum_{\substack{\{m_1,\ldots,m_k\} \subset \{1,\ldots,N\}, \\ \{w_1,\ldots,w_{N-k}\} = \{1,\ldots,N\} \setminus \{m_1,\ldots,m_k\}}} \overset{k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(m_l)T}\boldsymbol{U}^{(m_l)} \overset{N-k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(w_l)T}\boldsymbol{\Delta}_U^{(w_l)} \right) \right]\bar{\boldsymbol{D}}^{-1}.
\end{aligned}$$

Since $\boldsymbol{DS}\bar{\boldsymbol{D}}^{-1} = (\bar{\boldsymbol{D}} - \bar{\boldsymbol{P}} - \bar{\boldsymbol{Q}})\bar{\boldsymbol{D}}^{-1} = \boldsymbol{I} - (\bar{\boldsymbol{P}} + \bar{\boldsymbol{Q}})\bar{\boldsymbol{D}}^{-1}$,

$$\boldsymbol{A}^{(N)}\boldsymbol{DS}\bar{\boldsymbol{D}}^{-1} = \boldsymbol{A}^{(N)} - \boldsymbol{A}^{(N)}\big(\bar{\boldsymbol{P}} + \bar{\boldsymbol{Q}}\big)\bar{\boldsymbol{D}}^{-1}.$$

Therefore,

$$\begin{aligned}
&\|\tilde{\boldsymbol{A}}^{(N)}\bar{\boldsymbol{D}}^{-1} - \boldsymbol{A}^{(N)}\|_F \\
&\quad = \left\| \boldsymbol{A}^{(N)}\boldsymbol{D}\left[\overset{N}{\underset{n=1}{\circledast}} \boldsymbol{A}^{(n)T}\boldsymbol{\Delta}_U^{(n)}\right]\bar{\boldsymbol{D}}^{-1}\right\|_F \\
&\quad + \left\| \hat{\boldsymbol{A}}^{(N)}\hat{\boldsymbol{D}}\left[\sum_{k=1}^{N-1}\left(\sum_{\substack{\{m_1,\ldots,m_k\}\subset\{1,\ldots,N\},\\ \{w_1,\ldots,w_{N-k}\}=\{1,\ldots,N\}\setminus\\ \{m_1,\ldots,m_k\}}} \overset{k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(m_l)T}\boldsymbol{U}^{(m_l)} \overset{N-k}{\underset{l=1}{\circledast}} \hat{\boldsymbol{A}}^{(w_l)T}\boldsymbol{\Delta}_U^{(w_l)}\right)\right]\bar{\boldsymbol{D}}^{-1}\right\|_F \\
&\quad + \|\boldsymbol{A}^{(N)}\Big(\bar{\boldsymbol{P}}+\bar{\boldsymbol{Q}}\Big)\bar{\boldsymbol{D}}^{-1}\|_F \\
&\quad = P + Q,
\end{aligned}$$

where $P = O(\prod_{i=1}^{N-1}\epsilon_i)$ and $Q = O(k\epsilon\epsilon_\perp)$. ∎

This bound shows that Algorithm 3.1 achieves local convergence to fixed points for which $\epsilon_\perp = O(1/k)$. In contrast to the exact case, our analysis suggests only a linear convergence rate for approximation with the algorithm.

5. Approximate CP decomposition. In the above sections, we have provided a motivation for Algorithm 3.1 to compute a CP decomposition of rank R with R being less than or equal to the smallest mode length of the input tensor. We have shown in Theorem 4.3 that this algorithm exhibits a superlinear local convergence rate for exact CP decomposition problems and achieves a desirable approximation for special input tensors as described in Lemma 4.4. We now focus on the case of finding a good CP approximation for an arbitrary input tensor. We show that Algorithm 3.1 can be viewed as performing coupled minimization of the residual error of the decomposition in terms of a Mahalanobis distance metric [19]. Note that this perspective on the algorithm is different from the one introduced in section 3.1; however, it allows us to formulate an alternating minimization algorithm which generalizes Algorithm 3.1 to any CP rank and to interpolate between the updates of ALS and Algorithm 3.1.

5.1. Mahalanobis distance minimization. Each update of Algorithm 3.1 may be viewed as minimizing a residual error with rescaled components. For an order 3 tensor $\boldsymbol{\mathcal{X}}$ in updating the first factor, it minimizes

$$\|(\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}]\!])_{(1)}(\boldsymbol{C}^{\dagger T}\otimes\boldsymbol{B}^{\dagger T})\|_F^2. \tag{5.1}$$

We can recast the above expression in a way such that the objective function is transformed into a non-Euclidean distance metric with respect to the input tensor. We can rewrite the above expression as

$$\mathrm{vec}\big(\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}]\!]\big)^T\boldsymbol{M}\mathrm{vec}\big(\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}]\!]\big),$$

where $\boldsymbol{M} = (\boldsymbol{C}^{\dagger T}\boldsymbol{C}^\dagger \otimes \boldsymbol{B}^{\dagger T}\boldsymbol{B}^\dagger \otimes \boldsymbol{I})$. This may be viewed as minimizing the Mahalanobis distance metric relative to each factor while keeping the distance metric associated with that factor independent (and then updating it thereafter). This interpretation then enables us to extend Algorithm 3.1 for any CP rank and introduce methods that are a hybrid of Algorithm 3.1 and standard ALS.

5.1.1. Alternating Mahalanobis distance minimization. We consider a variant of Mahalanobis distance [18], which computes the distance between vectors

$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ as $(d(\boldsymbol{x}, \boldsymbol{y}))^2 = (\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{y})$ for a given symmetric positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$. The matrix $\boldsymbol{M}$ is called the ground metric matrix. The ground metric generalizes the Euclidean distance to Mahalanobis distance by rotation and scaling of the axes along which the distance is computed. While the underlying ground metric may already be known, it may also be learned via various metric learning techniques [7, 33]. In optimal transport applications and other applications which require computation of distance between probability distributions, a Wasserstein distance is considered instead. Wasserstein distance between tensors with a given ground metric has been considered for nonnegative CP decomposition [2]. The ground metric in Wasserstein distance may be learned via similar metric learning techniques as in Mahalanobis distance [17]. Simultaneous optimization for a ground metric and Wasserstein distance between matrices has been used for nonnegative matrix factorization [68].

We consider minimization of the Mahalanobis distance between tensors with a fixed ground metric which may be updated later. In particular, the objective function minimized for an input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, factors $\boldsymbol{A}^{(i)} \in \mathbb{R}^{I_i \times R}$, and ground metric $\mathbf{M} \in \mathbb{R}^{I_1 \ldots I_N \times I_1 \ldots I_N}$ is

$$f(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}) = \frac{1}{2} \operatorname{vec}\big(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\big)^T \boldsymbol{M} \operatorname{vec}\big(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\big), \quad \text{where } \boldsymbol{\mathcal{Y}} = [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]. \tag{5.2}$$

We restrict the ground metric matrix $\boldsymbol{M}$ to be Kronecker structured defined as

$$\boldsymbol{M} = \bigotimes_{k=1}^{N} \big(\boldsymbol{M}^{(k)}\big)^{-1},$$

where each $\big(\boldsymbol{M}^{(k)}\big)^{-1} \in \mathbb{R}^{I_k \times I_k}$ may be viewed as a ground metric for each mode of the tensor. This restriction allows us to exploit the computational benefits of the structure and enables us to formulate an efficient alternating minimization algorithm. We consider the objective in (5.2) for a general (fixed) ground metric for alternating optimization, which also allows us to formulate different algorithms for CP decomposition by changing the ground metric. We derive an update for the alternating minimization with respect to the nth factor matrix, given by

$$\boldsymbol{A}^{(n)} = \min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{X}_{(n)} - \boldsymbol{A}^{(n)} \boldsymbol{P}^{(n)T}\|_{\boldsymbol{M}}^2, \quad \text{where } \boldsymbol{P}^{(n)} = \bigodot_{m=1, m \neq n}^{N} \boldsymbol{A}^{(m)}. \tag{5.3}$$

For succinct writing, let $\boldsymbol{M}_{(n)} = \bigotimes_{k=1, k \neq n}^{N} (\boldsymbol{M}^{(k)})^{-1}$. Since the objective function is quadratic in $\boldsymbol{A}^{(n)}$, a minimizer of (5.3) can be found by obtaining obtaining a gradient $\boldsymbol{G}^{(n)}$ with respect to the nth factor matrix and setting it to $\mathbf{0}$. The gradient is

$$\boldsymbol{G}^{(n)} = \big(\boldsymbol{M}^{(n)}\big)^{-1} \boldsymbol{A}^{(n)} \boldsymbol{P}^{(n)T} \boldsymbol{M}_{(n)} \boldsymbol{P}^{(n)} - \big(\boldsymbol{M}^{(n)}\big)^{-1} \boldsymbol{X}_{(n)} \boldsymbol{M}_{(n)} \boldsymbol{P}^{(n)}.$$

Setting the gradient above to be $\mathbf{0}$ and equating $\boldsymbol{M}^{(n)} \big(\boldsymbol{M}^{(n)}\big)^{-1} = \boldsymbol{I}$, we get an update for the nth factor given as the solution of the following system:

$$\boldsymbol{A}^{(n)} \Big(\boldsymbol{P}^{(n)T} \boldsymbol{M}_{(n)} \boldsymbol{P}^{(n)} \Big) = \boldsymbol{X}_{(n)} \boldsymbol{M}_{(n)} \boldsymbol{P}^{(n)}. \tag{5.4}$$

Using the properties of Khatri–Rao products and Kronecker products, the update for the nth factor matrix reduces to the system of equations

$$\boldsymbol{A}^{(n)}\boldsymbol{Z}^{(n)} = \boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)}, \tag{5.5}$$

$$\text{where } \boldsymbol{L}^{(n)} = \bigodot_{k=1,k\neq n}^{N} \left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)}$$

$$\text{and } \boldsymbol{Z}^{(n)} = \circledast_{k=1,k\neq n}^{N} \boldsymbol{A}^{(k)T}\left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)}.$$

The above update leads to the ALS algorithm if $\boldsymbol{M}^{(k)} = \boldsymbol{I}$ for all k. We can retrieve Algorithm 3.1 by defining

$$\boldsymbol{M}^{(k)} = \boldsymbol{A}^{(k)}\boldsymbol{A}^{(k)T} + (\boldsymbol{I} - \boldsymbol{A}^{(k)}\boldsymbol{A}^{(k)\dagger}) \quad \forall k \in \{1,\dots,N\}. \tag{5.6}$$

The matrix $\boldsymbol{I} - \boldsymbol{A}^{(k)}\boldsymbol{A}^{(k)\dagger}$ is inconsequential when applied to the factor matrices as it is an orthogonal projector that projects onto the null space of the kth factor matrix. It is included to ensure that each $\boldsymbol{M}^{(k)}$ is symmetric positive definite. Since Algorithm 3.1 can be retrieved from the above update, we refer to Algorithm 3.1 as alternating Mahalanobis distance minimization (AMDM).

Let us assume that the iteration involving the above-derived alternating updates to each factor as in (5.4) converges to a critical point. We can then bound the backward error in application of each matricization of the reconstructed tensor $\boldsymbol{\mathcal{Y}} = [\![\boldsymbol{A}^{(1)},\cdots,\boldsymbol{A}^{(N)}]\!]$, since from (5.4), for each n, we have

$$\boldsymbol{A}^{(n)}\Big(\boldsymbol{P}^{(n)T}\boldsymbol{M}_{(n)}\boldsymbol{P}^{(n)}\Big) - \boldsymbol{X}_{(n)}\boldsymbol{M}_{(n)}\boldsymbol{P}^{(n)} = \boldsymbol{0},$$

$$\Big(\boldsymbol{Y}_{(n)} - \boldsymbol{X}_{(n)}\Big)\boldsymbol{M}_{(n)}\boldsymbol{P}^{(n)} = \boldsymbol{0}.$$

Therefore, we have that

$$\|\boldsymbol{Y}_{(n)}\boldsymbol{z} - \boldsymbol{X}_{(n)}\boldsymbol{z}\| = \|\boldsymbol{X}_{(n)}\boldsymbol{z}^{\perp}\|,$$

where $\boldsymbol{z}^{\perp}$ is the projection of $\boldsymbol{z} \in \mathbb{R}^{\prod_{j=1 j\neq n} I_j}$ onto the orthogonal complement of the column span of $\boldsymbol{M}_{(n)}\boldsymbol{P}^{(n)}$ or $\bigodot_{j=1,j\neq n}^{N}\boldsymbol{M}^{(j)}\boldsymbol{A}^{(j)}$. For ALS, AMDM, and the hybrid methods (introduced in section 5.3) that interpolate between them, it is sufficent to consider the projection onto the orthogonal complement of the column span of $\bigodot_{j=1,j\neq n}^{N}\boldsymbol{A}^{(j)}$. This is because for each j, the ground metric matrices are chosen such that the column span of $\boldsymbol{A}^{(j)}$ is an invariant subspace of $\boldsymbol{M}^{(j)}$.

5.1.2. Comparison of AMDM and ALS for approximate rank-2 CP decomposition. We use the formulation introduced in the previous subsection to generalize AMDM to the case when the CP rank R is greater than the mode lengths and to derive hybrid methods that interpolate between AMDM and ALS. The residual transformation tends to equalize the weight of contribution to the objective function attributed to components of the error associated with different rank-1 parts of the CP decomposition, $[\![\boldsymbol{a}_i,\boldsymbol{b}_i,\boldsymbol{c}_i]\!]$, without increasing the collinearity of the columns of the factors. We provide an example as an intuition for this assertion.

Consider a tensor $\boldsymbol{\mathcal{X}} = \lambda_1\boldsymbol{\mathcal{X}}_1 + \lambda_2\boldsymbol{\mathcal{X}}_2 + \boldsymbol{\mathcal{N}}$, where $\boldsymbol{\mathcal{X}}_1$ and $\boldsymbol{\mathcal{X}}_2$ are normalized rank-1 tensors and $\boldsymbol{\mathcal{N}}$ is noise of small magnitude. Assume that $\lambda_1 \gg \lambda_2$ and the rank-1 tensors are highly correlated, i.e., the factors have collinear columns. Let the

current CP decomposition approximation be $\boldsymbol{\mathcal{Y}} = [\![\bar{\boldsymbol{\lambda}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \bar{\lambda}_1 \boldsymbol{\mathcal{Y}}_1 + \bar{\lambda}_2 \boldsymbol{\mathcal{Y}}_2$. The least squares objective minimizes

$$\begin{aligned}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\|_F^2 &= \mathrm{vec}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}})^T \mathrm{vec}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}) \\ &= \mathrm{vec}(\boldsymbol{\mathcal{E}}_1 + \boldsymbol{\mathcal{E}}_2 + \boldsymbol{\mathcal{N}})^T \mathrm{vec}(\boldsymbol{\mathcal{E}}_1 + \boldsymbol{\mathcal{E}}_2 + \boldsymbol{\mathcal{N}}) \\ &= \|\boldsymbol{\mathcal{E}}_1\|_F^2 + \|\boldsymbol{\mathcal{E}}_2\|_F^2 + 2\mathrm{vec}(\boldsymbol{\mathcal{E}}_1)^T \mathrm{vec}(\boldsymbol{\mathcal{E}}_2) + 2\mathrm{vec}(\boldsymbol{\mathcal{N}})^T (\boldsymbol{\mathcal{E}}_1 + \boldsymbol{\mathcal{E}}_2 + \boldsymbol{\mathcal{N}}),\end{aligned}$$

where $\boldsymbol{\mathcal{E}}_1 = \lambda_1 \bar{\boldsymbol{\mathcal{X}}}_1 - \bar{\lambda}_1 \boldsymbol{\mathcal{Y}}_1$ and $\boldsymbol{\mathcal{E}}_2 = \lambda_2 \boldsymbol{\mathcal{X}}_2 - \bar{\lambda}_2 \boldsymbol{\mathcal{Y}}_2$. The ALS algorithm may reduce $\|\boldsymbol{\mathcal{E}}_1\|_F$ and the component of $\boldsymbol{\mathcal{E}}_2$ in the direction of $\boldsymbol{\mathcal{E}}_1$, since it leads to reduction of the terms with larger contribution in the error. This causes an increase in collinearity of the approximated factors and a more ill-conditioned decomposition. On the other hand, the objective in (5.1) can be expressed as

$$\begin{aligned}\|(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}) \times_1 \boldsymbol{I} \times_2 \boldsymbol{B}^{\dagger T} \times_3 \boldsymbol{C}^{\dagger T}\|_F^2 &= \mathrm{vec}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}})^T \boldsymbol{M} \mathrm{vec}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}) \\ &= \|\boldsymbol{\mathcal{E}}_1\|_{\boldsymbol{M}}^2 + \|\boldsymbol{\mathcal{E}}_2\|_{\boldsymbol{M}}^2 + 2\mathrm{vec}(\boldsymbol{\mathcal{E}}_1)^T \boldsymbol{M} \mathrm{vec}(\boldsymbol{\mathcal{E}}_2) \\ &\quad + 2\mathrm{vec}(\boldsymbol{\mathcal{N}})^T \boldsymbol{M} (\boldsymbol{\mathcal{E}}_1 + \boldsymbol{\mathcal{E}}_2 + \boldsymbol{\mathcal{N}}),\end{aligned}$$

where $\boldsymbol{M} = \boldsymbol{C}^{\dagger T} \boldsymbol{C}^{\dagger} \otimes \boldsymbol{B}^{\dagger T} \boldsymbol{B}^{\dagger} \otimes \boldsymbol{I}$. The matrix $\boldsymbol{M}$ rescales the components of the error according to the inverse of square of singular values of the factors, since

$$\begin{aligned}\|\boldsymbol{\mathcal{E}}_1\|_{\boldsymbol{M}}^2 &= \mathrm{vec}(\boldsymbol{\mathcal{E}}_1)^T \boldsymbol{M} \mathrm{vec}(\boldsymbol{\mathcal{E}}_1) \\ &= \mathrm{vec}(\bar{\boldsymbol{\mathcal{E}}}_1)^T (\boldsymbol{\Sigma}_{\boldsymbol{C}}^{-2} \otimes \boldsymbol{\Sigma}_{\boldsymbol{B}}^{-2} \otimes \boldsymbol{I}) \mathrm{vec}(\bar{\boldsymbol{\mathcal{E}}}_1),\end{aligned}$$

where $\bar{\boldsymbol{\mathcal{E}}}_1 = \boldsymbol{\mathcal{E}}_1 \times_1 \boldsymbol{I} \times_2 \boldsymbol{U}_{\boldsymbol{B}} \times_3 \boldsymbol{U}_{\boldsymbol{C}}$ with $\boldsymbol{U}_{\boldsymbol{B}}$ and $\boldsymbol{U}_{\boldsymbol{C}}$ being the left singular vectors of $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively. Thus, the error is rotated by the left singular vectors of $\boldsymbol{B}$ and $\boldsymbol{C}$ and then rescaled by the square of the inverse of the singular values of $\boldsymbol{B}$ and $\boldsymbol{C}$, i.e., $\Sigma_{\boldsymbol{B}}^{-2}$, $\Sigma_{\boldsymbol{C}}^{-2}$. Therefore, if the approximated factors are collinear, the contribution in the direction of the singular vectors of $\boldsymbol{B}$ and $\boldsymbol{C}$ with largest singular value is weighed proportionally less, and similarly the contribution of the error in the direction of those with smaller singular value is weighed more. This reduces the imbalance in error and leads to a better conditioned decomposition.

5.2. Generalizing AMDM to any CP rank. The AMDM algorithm as described in Algorithm 3.1 imposes a constraint that CP rank should be less than or equal to the smallest mode length of the tensor. As mentioned above, the update in (5.5) with the ground metric as defined in (5.6) leads to the AMDM algorithm with the rank constraint. We now describe how this definition of ground metric leads to an AMDM update without any restrictions imposed on the rank. Note that for each $\boldsymbol{M}^{(k)}$ as defined in (5.6),

$$\left(\boldsymbol{M}^{(k)}\right)^{-1} = \boldsymbol{A}^{(k)\dagger T} \boldsymbol{A}^{(k)\dagger} + (\boldsymbol{I} - \boldsymbol{A}^{(k)} \boldsymbol{A}^{(k)\dagger}).$$

The above equivalence can be verified simply by looking at the singular value decomposition of $\boldsymbol{M}^{(k)}$ and $\boldsymbol{A}^{(k)}$. The first part simply inverts the nonunit singular values of $\boldsymbol{M}^{(k)}$, and the second part is a projector and orthogonal to the first part and is therefore kept as is. The linear system for updating the nth factor matrix as in (5.5) can then be simplified to get

$$\boldsymbol{A}^{(n)}\boldsymbol{Z}^{(n)} = \boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)},$$

$$\text{where } \boldsymbol{L}^{(n)} = \bigodot_{k=1,k\neq n}^{N} \left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)} = \bigodot_{k=1,k\neq n}^{N} \boldsymbol{A}^{(k)\dagger T}$$

$$\text{and } \boldsymbol{Z}^{(n)} = \circledast_{k=1,k\neq n}^{N} \boldsymbol{A}^{(k)T}\left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)} = \circledast_{k=1,k\neq n}^{N} \boldsymbol{A}^{(k)\dagger}\boldsymbol{A}^{(k)}.$$

Note that $(\boldsymbol{M}^{(k)})^{-1}$ is not computed explicitly. The above update is equivalent to Algorithm 3.1 when the CP rank $R \leq I_m$ for all $m \in \{1,\ldots,N\}$, since in that case $\boldsymbol{A}^{(k)\dagger}\boldsymbol{A}^{(k)} = \boldsymbol{I}$ for all k, and, consequently, $\boldsymbol{Z}^{(n)} = \boldsymbol{I}$ for all n. For the case when the CP rank R is larger than the mode lengths, $\boldsymbol{A}^{(k)\dagger}\boldsymbol{A}^{(k)}$ is a projector onto the row space of $\boldsymbol{A}^{(k)}$, and therefore $\boldsymbol{Z}^{(n)}$ is symmetric positive semidefinite for each n. Since the system is semidefinite, a pivoted Cholesky decomposition followed by a triangular solve must be used for the solution to exist. Alternatively, as in ALS, a regularization term may be introduced to make the system positive definite. The cost of forming this system, $\boldsymbol{Z}^{(n)}$, is of the same leading order as ALS, i.e., $O(IR^2)$ per subsweep. It requires obtaining a pseudoinverse of the factor in the $(n-1)$th iteration, matrix multiplication, and Hadamard products of the factors and their previously obtained pseudoinverses. The system solve amounts to a computational cost of $O(R^3)$.

5.3. Interpolating between AMDM and ALS. Performing AMDM with an identity ground metric is equivalent to performing ALS. To explore methods that interpolate between AMDM and ALS, we can interpolate the ground metric between the identity matrix and the one associated with AMDM given in (5.6). We can find such ground metrics by decomposing each factor matrix into two low rank matrices such that $\boldsymbol{A}^{(k)} = \boldsymbol{A}_1^{(k)} + \boldsymbol{A}_2^{(k)}$, where $\boldsymbol{A}_1^{(k)}$ is the best rank-t approximation of $\boldsymbol{A}^{(k)}$. In other words, $\boldsymbol{A}_1^{(k)}$ contains the largest t singular values and the corresponding singular vectors of $\boldsymbol{A}^{(k)}$, and $\boldsymbol{A}^{(2)}$ contains the rest. Consequently, $\boldsymbol{A}_1^{(k)}$ and $\boldsymbol{A}_2^{(k)}$ are orthogonal to each other. The ground metric for each mode can then be defined using only the first part as

$$\boldsymbol{M}^{(k)} = \boldsymbol{A}_1^{(k)}\boldsymbol{A}_1^{(k)T} + \left(\boldsymbol{I} - \boldsymbol{A}_1^{(k)}\boldsymbol{A}_1^{(k)\dagger}\right) \quad \forall k \in \{1,\ldots,N\}. \tag{5.7}$$

Defining a ground metric based on only the first part of the singular value decomposition of the factors leads to hybrid methods, since if the first part is all of the singular value decomposition, then we get back the ground metric in AMDM, and if it is none of the same, then we get back the identity matrix by convention as the orthogonal complement of none is everything.

Note that for each k,

$$\left(\boldsymbol{M}^{(k)}\right)^{-1} = \boldsymbol{A}_1^{(k)\dagger T}\boldsymbol{A}_1^{(k)\dagger} + \left(\boldsymbol{I} - \boldsymbol{A}_1^{(k)}\boldsymbol{A}_1^{(k)\dagger}\right).$$

The update for the nth factor matrix also becomes a combination of AMDM and ALS where the first part of the singular value decomposition of factors is treated as in AMDM and the second one as in ALS. More precisely, the system of equations solved for updating the nth factor matrix is

$$\boldsymbol{A}^{(n)}\boldsymbol{Z}^{(n)} = \boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)},$$

$$\text{where } \boldsymbol{L}^{(n)} = \bigodot_{k=1,k\neq n}^{N} \left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)} = \bigodot_{k=1,k\neq n}^{N} \left(\boldsymbol{A}_1^{(k)\dagger T} + \boldsymbol{A}_2^{(k)}\right)$$

$$\text{and } \boldsymbol{Z}^{(n)} = \circledast_{k=1,k\neq n}^{N} \boldsymbol{A}^{(k)T}\left(\boldsymbol{M}^{(k)}\right)^{-1}\boldsymbol{A}^{(k)} = \circledast_{k=1,k\neq n}^{N} \left(\boldsymbol{A}_1^{(k)\dagger}\boldsymbol{A}_1^{(k)} + \boldsymbol{A}_2^{(k)T}\boldsymbol{A}_2^{(k)}\right). \tag{5.8}$$

As mentioned above, in the above update, $\boldsymbol{L}^{(n)}$ and $\boldsymbol{Z}^{(n)}$ can be split into two orthogonal parts as shown. The first part which involves $\boldsymbol{A}_1^{(k)}$ can be characterized as the AMDM part, and the one which involves $\boldsymbol{A}_2^{(k)}$ can be characterized as the ALS part. Note that computing these interpolating updates does not involve extra computational cost asymptotically when compared to ALS. Using these interpolating updates, we describe a hybrid algorithm in the subsection below. This algorithm is shown to perform the best in terms of residual and conditioning in section 6.

5.4. Hybrid algorithm. In this section, we leverage the interpolating updates to describe the hybrid algorithm which interpolates between AMDM and ALS. This algorithm starts by computing AMDM updates and transitions into ALS by implicitly reducing components used in the ground metric or reducing the size of $\boldsymbol{A}_1^{(k)}$ in (5.7) until the ground metric is $\boldsymbol{I}$. The transition in the hybrid algorithm is chosen to be from AMDM to ALS and not the other way around as ALS updates perturb the condition number less and make more progress when the updates are well-conditioned, and AMDM keeps the condition number bounded but may compromise on the Frobenius norm of the residual. The number of components of $\boldsymbol{A}^{(k)}$ in $\boldsymbol{A}_1^{(k)}$ is controlled by a hyperparameter, t. Note that t controls if the updates are more likely to behave as ALS or AMDM; for example, if $\boldsymbol{A}_1^{(k)}$ has a larger column space, then the updates are more likely to behave like AMDM. This hyperparameter may depend on the input tensor and conditioning of the decomposition and therefore needs to be tuned according to the problem. The details of the algorithm are described below and summarized in Algorithm 5.1.

The hybrid algorithm starts by normalizing the columns of all the factors and absorbing norms in the first factor as described in section 3 and then computing a reduced singular value decomposition of all the factors which costs $O(\sum_{n=1}^{N} I_n R \min(I_n, R))$. At the nth subiteration of the algorithm, the right- and left-hand sides of the system, $\boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)}$ and $\boldsymbol{Z}^{(n)}$, are computed using (5.8). Let $\boldsymbol{U}_1^{(m)}, \boldsymbol{V}_1^{(m)}$ be the left and right singular vectors of $\boldsymbol{A}_1^{(m)}$, respectively, $\boldsymbol{U}_2^{(m)}, \boldsymbol{V}_2^{(m)}$ be the left and right singular vectors of $\boldsymbol{A}_2^{(m)}$, respectively, and $\boldsymbol{S}_1^{(m)}, \boldsymbol{S}_2^{(m)}$ be the singular values of $\boldsymbol{A}_1^{(m)}$ and $\boldsymbol{A}_2^{(m)}$, respectively. The computation of $\boldsymbol{Z}^{(n)}$ requires $O(R^2 \min(I_n, R))$ operations. This computation can be made efficient by first storing the singular value decomposition of each of the factors and then computing $\boldsymbol{Z}_m = \boldsymbol{V}_1^{(m)}\boldsymbol{V}_1^{(m)T} + \boldsymbol{V}_2^{(m)}(\boldsymbol{S}_2^{(m)})^2\boldsymbol{V}_2^{(m)T}$ for each m. Then, $\boldsymbol{Z}^{(n)}$ can be computed by taking Hadamard products of $\boldsymbol{Z}_m$ as shown in Algorithm 5.1. A further reduction in cost can be obtained by updating $\boldsymbol{Z}_m$ at each subiteration by only computing the singular value decomposition of the previous factor. This amortization is similar to the one used in ALS while computing the left-hand sides, leading to a reduction in cost by a factor of N. The computation of $\boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)}$ can be similarly performed by using $\boldsymbol{L}_m = \boldsymbol{U}_1^{(m)}(\boldsymbol{S}_1^{(m)})^{-1}\boldsymbol{V}_1^{(m)T} + \boldsymbol{U}_2^{(m)}\boldsymbol{S}_2^{(m)}\boldsymbol{V}_2^{(m)T}$. Then $\boldsymbol{X}_{(n)}\boldsymbol{L}^{(n)} = \boldsymbol{X}_{(n)} \bigodot_{m=1,m\neq n}^{N} \boldsymbol{L}_m$ which is an MTTKRP computation and can be performed efficiently and amortized as in ALS. The symmetric semidefinite system solve requires $O(R^3)$. Computing the right-hand side, i.e., performing MTTKRP, is the most computationally expensive operation with a cost of $O(\prod_{n=1}^{N} I_n R)$ for each subsweep. In addition to this, the factor matrix obtained after the solve is normalized, and a reduced singular value decomposition is obtained to update the singular value decomposition which costs $O(I_n R \min(I_n, R))$. Therefore, the asymptotic computational cost of the algorithm is the same as ALS, being $O(\prod_{n=1}^{N} I_n R)$.

We summarize the above-described hybrid algorithm in Algorithm 5.1.

Algorithm 5.1 General AMDM: AMDM with singular value thresholding.

Input: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, threshold t, rank R
Initialize $\{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}\}$ so each $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ is random
for $n \in \{2, \ldots, N\}$ **do**
  $\boldsymbol{A}^{(n)} = \text{normalize}(\boldsymbol{A}^{(n)})$
  $\boldsymbol{U}^{(n)} = \min(I_n, R)$ left singular vectors of $\boldsymbol{A}^{(n)}$
  $\boldsymbol{V}^{(n)} = \min(I_n, R)$ right singular vectors of $\boldsymbol{A}^{(n)}$
  $\boldsymbol{s}^{(n)} = \min(I_n, R)$ singular values of $\boldsymbol{A}^{(n)}$
end for
while `Convergence` **do**
  for $n \in \{1, \ldots, N\}$ **do**
    for $m \in \{1, \ldots, N\}, m \neq n$ **do**
      $\boldsymbol{s}_{\text{ps}}^{(m)} =$ first t values of $\boldsymbol{s}^{(m)}$ inverted and others as it is
      $\boldsymbol{L}_m = \boldsymbol{U}^{(m)} \mathbf{diag}(\boldsymbol{s}_{\text{ps}}^{(m)}) \boldsymbol{V}^{(m)T}$
      $\boldsymbol{Z}_m = \boldsymbol{V}^{(m)} \mathbf{diag}(\boldsymbol{s}_{\text{ps}}^{(m)} * \boldsymbol{s}^{(m)}) \boldsymbol{V}^{(m)T}$
    end for
    Compute $\boldsymbol{Z}^{(n)} = \circledast_{m=1, m \neq n}^{N} \boldsymbol{Z}_m$
    Compute $\boldsymbol{X}_{(n)} \boldsymbol{L}^{(n)}$ where $\boldsymbol{L}^{(n)} = \bigodot_{m=1, m \neq n}^{N} \boldsymbol{L}_m$
    Solve for $\boldsymbol{A}^{(n)}$ in $\boldsymbol{A}^{(n)} \boldsymbol{Z}^{(n)} = \boldsymbol{X}_{(n)} \boldsymbol{L}^{(n)}$ as in (5.8)
    if n is N: `Check Convergence, if converged: Break`
    $\boldsymbol{A}^{(n)} = \text{normalize}(\boldsymbol{A}^{(n)})$
    Update $\boldsymbol{U}^{(n)} = \min(I_n, R)$ left singular vectors of $\boldsymbol{A}^{(n)}$
    Update $\boldsymbol{V}^{(n)} = \min(I_n, R)$ right singular vectors of $\boldsymbol{A}^{(n)}$
    Update $\boldsymbol{s}^{(n)} = \min(I_n, R)$ singular values of $\boldsymbol{A}^{(n)}$
  end for
end while
return factor matrices $\{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}\}$

6. Numerical experiments. We perform numerical experiments to demonstrate the convergence behavior of the AMDM algorithm compared to ALS for various tensors which include synthetic examples and tensors arising in different applications. Both algorithms are implemented in Python and are publicly available.[2] The ALS implementation used in this work has been used in several previous works [38, 39, 55]. We use absolute residual and fitness of the decomposition in the Frobenius norm as metrics to measure the closeness of the decomposition to the input tensor. For an input tensor $\boldsymbol{\mathcal{X}}$, these are given as

$$r = \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\|_F \text{ and } f = 1 - \frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\|_F}{\|\boldsymbol{\mathcal{X}}\|_F},$$

respectively, where $\boldsymbol{\mathcal{Y}} = [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]$ is the approximated tensor. For measuring the stability of the decomposition or the degree of overlap of CP components, we use the normalized CP decomposition condition number [10] to measure the sensitivity or degree of overlap of rank-1 components of the decomposition.

[2] https://github.com/cyclops-community/tensor_decomposition.

The normalized CP condition number is given by the reciprocal of the smallest singular value of Terracini's matrix associated with the CP decomposition. For an equidimensional tensor of order N with mode length s and CP rank R, the size of Terracini's matrix is $s^N \times (N(s-1)+1)R$. For CP rank lower than mode lengths of the tensor, this matrix can be compressed to $R^N \times (N(R-1)+1)R$, and the CP decomposition condition number can be efficiently computed with a cost of $O(R^{N+4})$. The details of computation of the condition number are in Appendix A. A more general result regarding computing the condition number of joint decompositions efficiently is proved in [20]. Our experiments consider two types of synthetic tensors.

Tensor made by random matrices (*random* tensor). We create these tensors based on known uniformly distributed randomly generated factor matrices $\boldsymbol{A}^{(n)} \in (0,1)^{s\times R}$, $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]$.

Tensor made by collinear random matrices (*collinearity* tensor). We use a similar approach as used in [1] to generate factor matrices with a fixed value of collinearity, say C. That means that these tensors are created with randomly generated factors $\mathbf{A}^{(n)} \in \mathbb{R}^{s\times R}$ with the following property:

$$\begin{aligned} &\mathbf{a}_r^{(n)T}\mathbf{a}_z^{(n)} = C \\ &\text{s.t. } \|\mathbf{a}_r^{(n)}\| = 1 \; \forall r \neq z \in \{1, \ldots, R\}. \end{aligned}$$

We then set $d_{ii} = i$ for all $i \in \{1, \ldots, R\}$ to create a tensor, $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{D}; \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}]\!]$.

We consider four tensors from various real-world applications.

Sleep-EDF tensor: This dataset has been used to identify sleeping patterns. It comprises electroencephalogram (EEG) and electromyography (EMG) data in the nonrapid eye movement stage of sleep [30].

MGH tensor: This dataset consists of data from Massachusetts General Hospital. It includes combinations of EEG data, respiratory signals, and EMG signals. This dataset was used to analyze sleep using deep neural networks [9].

SCF tensor. We consider the density fitting tensor (Cholesky factor of the two-electron integral tensor) arising in quantum chemistry. This tensor has been used previously in [55] to compare the efficacy of the Gauss–Newton and ALS algorithms. We leverage the PySCF library [59] to generate the three-dimensional compressed density fitting tensor, representing the compressed restricted Hartree–Fock wave function of water molecule chain systems with a STO-3G basis set. The number of molecules in the system is set to three for this experiment.

Amino acid tensor. This dataset consists of five simple laboratory-made samples. Each sample contains different amounts of tyrosine, tryptophan, and phenylalanine dissolved in phosphate-buffered water. The samples were measured by fluorescence [11].

The experiments are divided broadly into two categories:

Exact decomposition. We create synthetic tensors with known CP rank R and compare the convergence behavior of the AMDM algorithm with the ALS algorithm for exact CP decomposition.

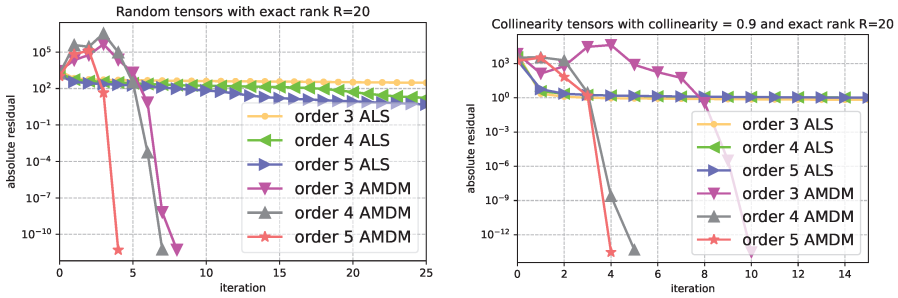
Approximate decomposition. We create synthetic tensors with known CP rank and special structure such as with added noise or as described in Lemma 4.4. We then approximate these tensors with CP rank R which is lower than the underlying decomposition rank. We also consider real-world tensors from different applications with unknown CP rank. For synthetic tensors, we create 10 different tensors and run each with a random initialization for ALS and different variants of AMDM and plot the mean of fitness and condition number at each iteration with confidence intervals

as the minimum and maximum values. The random initialization are factor matrices with entries sampled from a uniform distribution of real numbers between 0 and 1. For real-world tensors, we plot the mean of 5 initializations for larger tensors, Sleep-EDF, MGH, and SCF tensors and the mean of 10 iterations for amino acid tensors with confidence intervals as minimum and maximum values.

6.1. Exact CP decomposition. We compare ALS and Algorithm 3.1 for computing exact CP decomposition of synthetic tensors in Figures 1 and 2 and verify our theoretical results. We create *collinearity* and *random* tensors of specified CP rank to analyze the convergence of these algorithms. We claim that the algorithm has converged for a CP decomposition with exact rank if the absolute residual is below 10^{-9}.

In Figure 1, we create equidimensional synthetic tensors of order N with each mode length s to follow $s = 100$ for order 3, $s = 53$ for order 4, and $s = 24$ for order 5 with fixed CP rank $R = 20$. We initialize the factors with uniformly distributed random matrices for both algorithms and plot the absolute residual for each iteration. For the *collinearity* tensors, collinearity value, i.e., C, is set to be 0.9. We can observe superlinear convergence of Algorithm 3.1 for both cases, while ALS appears to be slowed down by the "swamp" phenomenon for the *collinearity* tensors.

In Figure 2, we create a *random* equidimensional tensor with mode length $s = 100$ and CP rank $R = 200$. We can observe that Algorithm 5.1 converges to an approximate tensor which is $\epsilon = 10^{-10}$ close to the exact tensor while taking only a few iterations to do so. Since $R > s$, there is a possibility that the synthetic tensor might have a border rank lower than 200, and we may be finding the corresponding factor matrices.



(a) Random tensor residual (b) Collinearity tensor residual

FIG. 1. *Superlinear convergence of the AMDM algorithm for exact CP decomposition.*

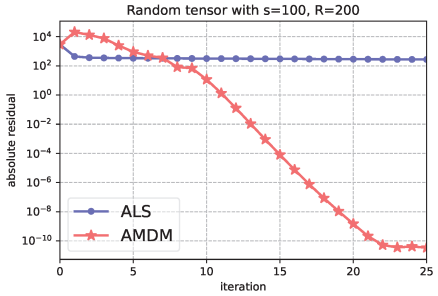


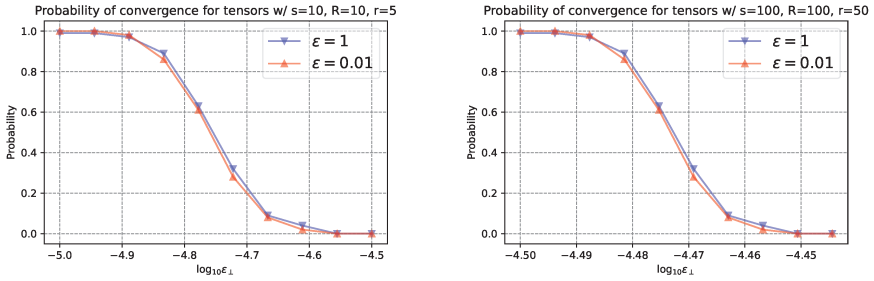
FIG. 2. *Linear rate of convergence of AMDM for large rank for exact CP decomposition.*

Therefore we cannot guarantee that we find the exact input tensor [43]. However, we do observe a linear convergence rate to obtain this approximation. ALS makes slow progress for this case. A reason for that might again be related to the collinearity of the factors, since when $R > s$, collinearity is high and ALS is more likely to experience the "swamp" phenomenon [41].

6.2. Approximate CP decomposition. We plot the probability of convergence to the desired decomposition for synthetic tensors as described in Lemma 4.4 with respect to the $\epsilon_\perp$ to verify our theoretical results in Figure 3. We construct these tensors by constructing the first $R/2$ columns of the factors with random matrices and then constructing the other half by projecting it onto the orthogonal complement of the column space of the first half and adding Gaussian noise of amplitude $\epsilon_\perp$ to the same. We construct 100 such tensors, and for each tensor we consider 5 initial guesses which are ϵ away from the desired decomposition as described in 4.4. We plot the probability of convergence over these 100 tensors by considering if at least 1 initial guess is within 10^{-9} of the desired factors. We observe that the probability of convergence to the desired decomposition is 1 when the $\epsilon_\perp$ is small, irrespective of the size and ϵ, thereby verifying that the first half of CP decomposition is a stationary point. The probability decreases as we increase $\epsilon_\perp$; i.e., the decomposition converges to different stationary points for these tensors.

We compare ALS and different variants of AMDM for computing approximate CP decomposition of synthetic tensors in Figure 4 and application tensors that admit approximations with a low CP rank in Figure 5, Figure 6, Figure 7, and Figure 8. We plot the fitness and the condition number of the CP decomposition to compare ALS and variants of AMDM. The integer associated with AMDM $t = \#$ corresponds to the number of singular values inverted for each factor or best rank-t approximation $\boldsymbol{A}_1$ in (5.7). The hybrid algorithm starts by using threshold $t = R$ in Algorithm 5.1; i.e., it starts with Algorithm 3.1 and gradually decreases the threshold to 0 to recover the ALS algorithm.

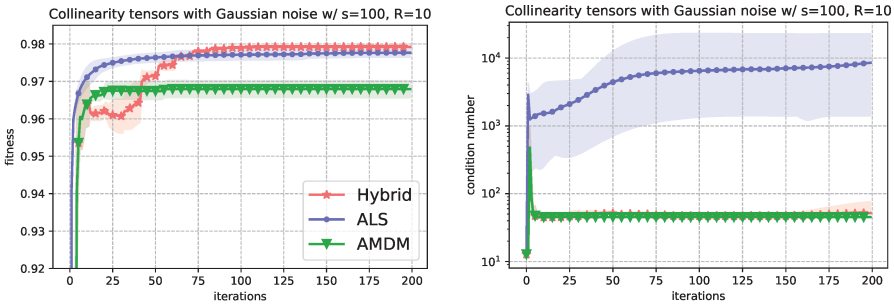
In Figure 4, we compute a rank-10 CP decomposition of the *collinearity* tensor with collinearity $C = 0.9$ and exact CP rank $R = 10$ with added Gaussian noise tensor. Each entry of the noise tensor is distributed normally with mean $\mu = 0$ and standard deviation $\sigma = 0.001$. We create 10 such tensors randomly and, for each tensor, initialize the algorithms with a random initial guess. We observe that



(a) Probability of convergence for equidimensional tensors with mode length= 10, exact CP rank= 10 and approximate rank= 5

(b) Probability of convergence for equidimensional tensors with mode length= 100, exact CP rank= 100 and approximate rank= 50

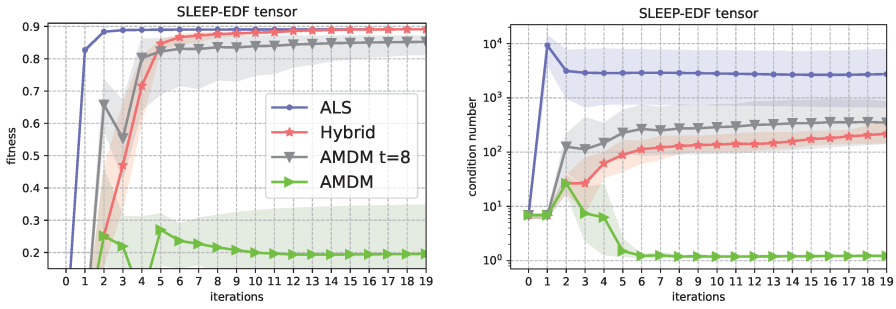
FIG. 3. *Probability of convergence for* 100 *tensors with* 5 *initial guesses in the setting as described in Lemma* 4.4.



(a) *Collinearity* tensor with Gaussian noise fitness

(b) *Collinearity* tensor with Gaussian noise condition number

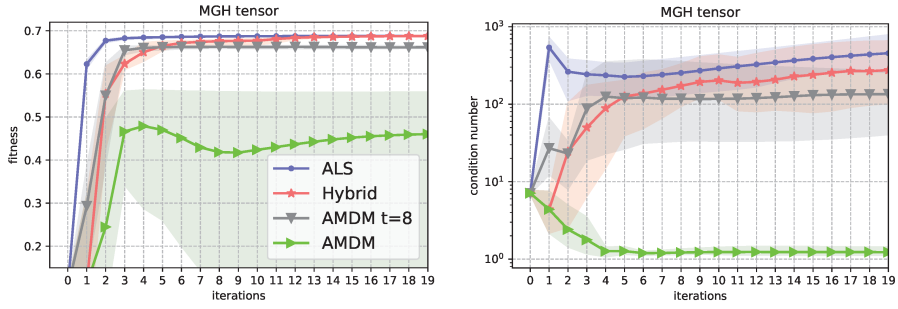
FIG. 4. *Collinearity tensor of size* $100 \times 100 \times 100$ *with exact CP rank* $R = 10$ *with added Gaussian noise with each entry distributed with mean* $\mu = 0$ *and standard deviation* $\sigma = 0.001$.



(a) SLEEP tensor fitness

(b) SLEEP tensor condition number

FIG. 5. *Sleep-EDF tensor of dimensions* $2048 \times 14 \times 129 \times 86$ *approximated with CP rank* $R = 10$, *where* t *is the singular value threshold in Algorithm* 5.1.

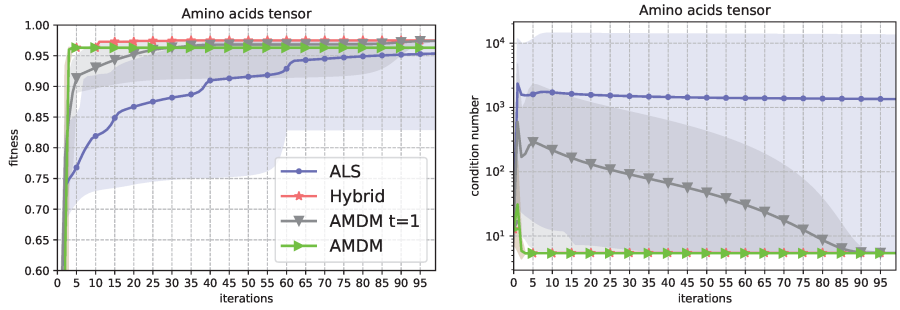


(a) MGH tensor fitness

(b) MGH tensor condition number

FIG. 6. *MGH tensor of dimensions* $2048 \times 12 \times 257 \times 43$ *approximated with CP rank* $R = 10$, *where* t *is the singular value threshold in Algorithm* 5.1.

the mean fitness of the AMDM algorithm is comparable to ALS with tight max-min confidence intervals and a low condition number for all the tensors. On the other hand, ALS maintains a higher fitness with $15 - 1500\times$ the mean condition number. Note



(a) Amino acids tensor fitness (b) Amino acids tensor condition number

FIG. 7. *Amino acid tensor of dimensions* $5 \times 61 \times 201$ *approximated with CP rank* $R = 3$, *where* t *is the singular value threshold in Algorithm* 5.1.

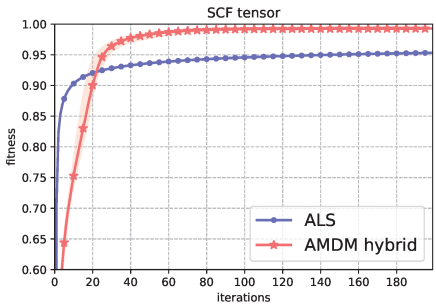


FIG. 8. *SCF tensor of size* $339 \times 21 \times 21$ *approximated with CP rank* $R = 200$, *where* t *is the* $\arg\min_i \sigma_{max}/\sigma_i < 100$ *threshold in Algorithm* 5.1.

that in this approximate CP decomposition setting, the convergence of the AMDM algorithm behaves similarly to ALS; i.e., the rate is linear, as shown in Lemma 4.4. We can clearly see the advantage of the hybrid algorithm where we start with AMDM and decrease the number of singular values inverted by 1 after every 10 iterations. The hybrid algorithm has a mean fitness that is higher than ALS while maintaining a condition number that is very close to AMDM. This suggests that for these types of tensors a more stable decomposition with desirable accuracy can be found via the hybrid algorithm.

In Figure 5 and Figure 6, we compute the CP decomposition of the Sleep-EDF tensor and MGH tensor with CP rank $R = 10$. We consider several variants of the hybrid Algorithm 5.1. For the hybrid algorithm, one less singular value is inverted after every iteration. We clearly see a pattern in both tensors that if lesser singular values are inverted, then the fitness is higher and the CP decomposition condition number is larger. We also see that the hybrid algorithm is able to achieve a fitness as high as ALS while maintaining a lower condition number. For the Sleep-EDF tensor we see this effect being more pronounced as we get a mean condition number that is around an order of magnitude smaller with small confidence intervals while the fitness is the same. For the MGH tensor, we have that the mean condition number of the hybrid algorithm is not very different from the condition number in ALS, and after 50 iterations this difference is also lesser.

In Figure 7, we compute the CP decomposition of the amino acid tensor with rank $R = 3$. We use 10 random initializations for this tensor because of its small size. This tensor is different from Sleep-EDF and MGH as we have the a higher mean fitness for AMDM variants as compared to ALS. The large confidence interval of ALS suggests that it is sensitive to initializations, whereas AMDM and the hybrid algorithm where we decrease the number of singular values inverted by 1 after every 10 iterations have a tighter max-min confidence interval suggesting that they are less susceptible to initialization and maintain a higher fitness with a several orders of magnitude lower condition number. Also, note that all the AMDM variants converge to a similar fitness (the confidence intervals also collapse) with a similar condition number which is close to unity for this case.

In Figure 8, we compute CP decomposition of the SCF tensor with rank $R = 200$ (exceeding 2 of the 3 tensor dimensions). We use a relative tolerance criterion for computing $t = \text{argmin}_i(\frac{\sigma_{\max}}{\sigma_i} < 100)$ in the hybrid algorithm; i.e., singular values σ_i are inverted only if $\frac{\sigma_{\max}}{\sigma_i} < 100$, where $\sigma_{\max}$ is the maximum singular value. The number of initialization used here is 5, and the hybrid algorithm outperforms ALS in terms of fitness, as the mean fitness reaches 0.992 fitness in 150 iterations, whereas ALS reaches 0.95 in 200 iterations; the max-min confidence bounds suggests that the hybrid algorithm is a clear winner here.

7. Conclusion. In this work, we have proposed an alternating optimization algorithm, AMDM, to compute a CP decomposition of the tensor. This algorithm achieves superlinear local convergence for exact CP rank problems when the CP rank is smaller than or equal to all the mode lengths of the tensor with the same asymptotic computational cost as that of ALS. For approximating a tensor via CP decomposition, we theoretically show that the algorithm locally converges to the stationary points of (3.3) for tensors with special CP structure. Although the existence of these stationary points for any tensor is an open problem, we empirically confirm that the AMDM algorithm converges to these stationary points for various tensors. Viewing the algorithm as minimizing a Mahalanobis distance helps in generalization of the method for CP rank larger than the mode lengths and interpolation between the AMDM and ALS algorithms. We also formulate an efficient way to compute the CP decomposition condition number to track the condition of the decomposition throughout the algorithm. Our numerical experiments confirm that interpolation of algorithms between AMDM and ALS leads to a better conditioned decomposition without significant difference in fitness as compared to ALS for synthetic tensors as well as most of the tested real-world tensors. We provide an intuitive reasoning of this phenomenon and leave the detailed analysis as a future direction of research.

Appendix A. Computing the condition number of a CP decomposition. It has been shown that the CP decomposition condition number is the reciprocal of the smallest singular value of a matrix called Terracini's matrix. This matrix consists of the orthogonal basis for the tangent space of each of the rank-1 components of the reconstructed tensor. We will refer to the normalized condition number as the condition number of CP decomposition, and we refer the reader to [10, 66] for details about how the notion of condition number of a CP decomposition is defined and derived. Consider an equidimensional order 3 real tensor $\boldsymbol{\mathcal{X}}$ with mode length s. Let the CP decomposition approximation of rank R be given by $[\![\boldsymbol{D}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$; then Terracini's matrix $\boldsymbol{V}$ is $\boldsymbol{V} = [\boldsymbol{V}_1 \dots \boldsymbol{V}_R]$, where for all $i \in \{1, \dots, R\}$,

$$\boldsymbol{V}_i = [\boldsymbol{a}_i \otimes \boldsymbol{b}_i \otimes \boldsymbol{c}_i \quad \boldsymbol{Q}_{\boldsymbol{a}_i}^{\perp} \otimes \boldsymbol{b}_i \otimes \boldsymbol{c}_i \quad \boldsymbol{a}_i \otimes \boldsymbol{Q}_{\boldsymbol{b}_i}^{\perp} \otimes \boldsymbol{c}_i \quad \boldsymbol{a}_i \otimes \boldsymbol{b}_i \otimes \boldsymbol{Q}_{\boldsymbol{c}_i}^{\perp}],$$

and $\boldsymbol{Q}_{\boldsymbol{a}_i}^{\perp} \in \mathbb{R}^{s\times(s-1)}$ is an orthogonal basis of the orthogonal complement of $\boldsymbol{a}_i$, and $\boldsymbol{Q}_{\boldsymbol{b}_i}^{\perp}$, and $\boldsymbol{Q}_{\boldsymbol{c}_i}^{\perp}$ are defined similarly. Consequently, Terracini's matrix is of size $s^3 \times R(3s-2)$, and the computational cost of computing the smallest singular value via a Krylov subspace method is $O(s^5R^2)$. For an order N tensor, this cost is $O(N^2s^{N+2}R^2)$ and therefore expensive to compute for decompositions with moderately large mode lengths.

The cost of computing the condition number can be decreased significantly for when the rank of the CP decomposition is less than all the mode lengths of the input tensor, i.e., if $R < s$. Assume that $R \le s$; then since the condition number is invariant to orthogonal transformations [10], for a CP decomposition of an order 3 tensor,

$$\kappa\big([\![\boldsymbol{D};\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}]\!]\big) = \kappa\big([\![\boldsymbol{D};\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{A},\boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{B},\boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{C}]\!]\big),$$

where $\boldsymbol{Q}_{\boldsymbol{A}} \in \mathbb{R}^{s\times s} = [\boldsymbol{Q}_{\boldsymbol{A}}^{(1)}\boldsymbol{Q}_{\boldsymbol{A}}^{(2)}]$ and the columns of $\boldsymbol{Q}_{\boldsymbol{A}}^{(1)} \in \mathbb{R}^{s\times R}$ are an orthogonal basis of the column space of $\boldsymbol{A}$, while the columns of $\boldsymbol{Q}_{\boldsymbol{A}}^{(2)} \in \mathbb{R}^{s\times(s-R)}$ are an orthogonal basis for the orthogonal complement of the column space of $\boldsymbol{A}$. We define $\boldsymbol{Q}_{\boldsymbol{B}} = [\boldsymbol{Q}_{\boldsymbol{B}}^{(1)}\boldsymbol{Q}_{\boldsymbol{B}}^{(2)}]$ and $\boldsymbol{Q}_{\boldsymbol{C}} = [\boldsymbol{Q}_{\boldsymbol{C}}^{(1)}\boldsymbol{Q}_{\boldsymbol{C}}^{(2)}]$ similarly. The transformed Terracini's matrix $\boldsymbol{U} = [\boldsymbol{U}_1 \dots \boldsymbol{U}_R]$, where for all $i \in \{1,\dots,R\}$,

$$\boldsymbol{U}_i = \Bigg[\underbrace{\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i \otimes \boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_i \otimes \boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_i}_{\boldsymbol{U}_i^{(1)}} \quad \underbrace{\bar{\boldsymbol{Q}}_{\boldsymbol{a}_i}^{\perp} \otimes \boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_i \otimes \boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_i}_{\boldsymbol{U}_i^{(2)}}$$
$$\underbrace{\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i \otimes \bar{\boldsymbol{Q}}_{\boldsymbol{b}_i}^{\perp} \otimes \boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_i}_{\boldsymbol{U}_i^{(3)}} \quad \underbrace{\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i \otimes \boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_i \otimes \bar{\boldsymbol{Q}}_{\boldsymbol{c}_i}^{\perp}}_{\boldsymbol{U}_i^{(4)}} \Bigg],$$

where $\bar{\boldsymbol{Q}}_{\boldsymbol{a}_i}^{\perp} = \boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{Q}_{\boldsymbol{a}_i}^{\perp} \in \mathbb{R}^{s\times(s-1)}$ is an orthogonal basis of the orthogonal complement of $\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i$ and $\bar{\boldsymbol{Q}}_{\boldsymbol{b}_i}^{\perp}$, and $\bar{\boldsymbol{Q}}_{\boldsymbol{c}_i}^{\perp}$ are defined similarly. Note that $\boldsymbol{U}_i^{(j)T}\boldsymbol{U}_i^{(k)} = \boldsymbol{0}$ for $j \ne k$, since $\bar{\boldsymbol{Q}}_{\boldsymbol{a}_i}^{\perp\,T}\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i = 0$. Consequently,

$$\sigma_{\min}(\boldsymbol{U}) = \min_{j\in\{2,3,4\}} \sigma_{\min}\Big(\begin{bmatrix} \boldsymbol{U}_1^{(1)} & \boldsymbol{U}_1^{(j)} & \dots & \boldsymbol{U}_R^{(1)} & \boldsymbol{U}_R^{(j)} \end{bmatrix} \Big).$$

We analyze $[\boldsymbol{U}_i^{(1)}\ \boldsymbol{U}_i^{(j)}]$ instead of $\boldsymbol{U}_i^{(j)}$ separately in order to have Kronecker products with orthogonal (square) matrices instead of nonsquare matrices with orthonormal columns, thereby simplifying analysis. After the above transformation, we can obtain a reduced form of smaller dimensions each of the four matrices to compute the condition number more efficiently. Note that $\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_i = [\begin{smallmatrix} \boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i \\ \boldsymbol{0} \end{smallmatrix}]$, and similarly for $\boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_i$ and $\boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_i$. Further, we can choose the columns $\boldsymbol{Q}_{\boldsymbol{a}_i}^{\perp}$ so that $\bar{\boldsymbol{Q}}_{\boldsymbol{a}_i}^{\perp} = \boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{Q}_{\boldsymbol{a}_i}^{\perp} = [\begin{smallmatrix} \boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i}^{\perp} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{smallmatrix}]$, where $\boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i}^{\perp} \in \mathbb{R}^{R\times(R-1)}$ is an orthogonal basis of the orthogonal complement of $\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i$, and similarly for $\boldsymbol{Q}_{\boldsymbol{b}_i}^{\perp}$ and $\boldsymbol{Q}_{\boldsymbol{c}_i}^{\perp}$. Consequently, for $j = 2$,

$$
\begin{aligned}
&\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{U}_1^{(1)} & \boldsymbol{U}_1^{(2)} & \dots & \boldsymbol{U}_R^{(1)} & \boldsymbol{U}_R^{(2)}\end{bmatrix}\right)\\
&\quad=\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{A}}^T\boldsymbol{a}_1\otimes\boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_1 & \bar{\boldsymbol{Q}}_{\boldsymbol{a}_1}^{\perp}\otimes\boldsymbol{Q}_{\boldsymbol{B}}^T\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^T\boldsymbol{c}_1\dots\end{bmatrix}\right)\\
&\quad=\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1 & \boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1}^{\perp} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\otimes\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\\ \boldsymbol{0}\end{bmatrix}\otimes\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1\\ \boldsymbol{0}\end{bmatrix}\quad\dots\right)\\
&\quad=\min\left\{\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1}\otimes\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1\dots\end{bmatrix}\right),\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1\dots\end{bmatrix}\right)\right\}\\
&\quad=\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1}\otimes\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1 & \dots & \boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_R}\otimes\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_R\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_R\end{bmatrix}\right)\\
&\quad=\sigma_{\min}\left(\begin{bmatrix}\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1\otimes\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1 & \boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_1}^{\perp}\otimes\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_1\otimes\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_1\dots\end{bmatrix}\right)\\
&\quad=\sigma_{\min}\left(\begin{bmatrix}\bar{\boldsymbol{U}}_1^{(1)} & \bar{\boldsymbol{U}}_1^{(2)} & \dots & \bar{\boldsymbol{U}}_R^{(1)} & \bar{\boldsymbol{U}}_R^{(2)}\end{bmatrix}\right),
\end{aligned}
$$

where $\boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i}=[\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i\,\boldsymbol{Q}_{\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i}^{\perp}]\in\mathbb{R}^{R\times R}$ is an orthogonal (square) matrix. A similar argument can also be shown for $j=3,4$ to show that

$$
\sigma_{\min}(\boldsymbol{U})=\min_{j\in\{2,3,4\}}\sigma_{\min}\left(\begin{bmatrix}\bar{\boldsymbol{U}}_1^{(1)} & \bar{\boldsymbol{U}}_1^{(j)} & \dots & \bar{\boldsymbol{U}}_R^{(1)} & \bar{\boldsymbol{U}}_R^{(j)}\end{bmatrix}\right)=\sigma_{\min}(\bar{\boldsymbol{U}}),
$$

where

$$
\bar{\boldsymbol{U}}_i=\begin{bmatrix}\underbrace{\bar{\boldsymbol{a}}_i\otimes\bar{\boldsymbol{b}}_i\otimes\bar{\boldsymbol{c}}_i}_{\bar{\boldsymbol{U}}_i^{(1)}} & \underbrace{\boldsymbol{Q}_{\bar{\boldsymbol{a}}_i}^{\perp}\otimes\bar{\boldsymbol{b}}_i\otimes\bar{\boldsymbol{c}}_i}_{\bar{\boldsymbol{U}}_i^{(2)}} & \underbrace{\bar{\boldsymbol{a}}_i\otimes\boldsymbol{Q}_{\bar{\boldsymbol{b}}_i}^{\perp}\otimes\bar{\boldsymbol{c}}_i}_{\bar{\boldsymbol{U}}_i^{(3)}} & \underbrace{\bar{\boldsymbol{a}}_i\otimes\bar{\boldsymbol{b}}_i\otimes\boldsymbol{Q}_{\bar{\boldsymbol{c}}_i}^{\perp}}_{\bar{\boldsymbol{U}}_i^{(4)}}\end{bmatrix},
$$

$\bar{\boldsymbol{a}}_i=\boldsymbol{Q}_{\boldsymbol{A}}^{(1)T}\boldsymbol{a}_i,\bar{\boldsymbol{b}}_i=\boldsymbol{Q}_{\boldsymbol{B}}^{(1)T}\boldsymbol{b}_i$, and $\bar{\boldsymbol{c}}_i=\boldsymbol{Q}_{\boldsymbol{C}}^{(1)T}\boldsymbol{c}_i$. The size of the above reduced Terracini's matrix is $R^3\times R(3R-2)$; hence a direct computation of the singular value decomposition can be used to compute the condition number with a cost of $O(R^7)$ for $R\le s$.

All the above arguments can be easily generalized to an order N nonequidimensional tensor. Therefore, we showed that the condition number of CP decomposition is invariant to the following transformation:

$$
\kappa\big(\llbracket\boldsymbol{D};\boldsymbol{A}_1\dots,\boldsymbol{A}_N\rrbracket\big)=\kappa\left(\llbracket\boldsymbol{D};\boldsymbol{Q}_{\boldsymbol{A}_1}^{(1)T}\boldsymbol{A}_1,\dots,\boldsymbol{Q}_{\boldsymbol{A}_N}^{(N)T}\boldsymbol{A}_N\rrbracket\right),
$$

where for all $i\in\{1,\dots,N\}$, the columns of $\boldsymbol{Q}_{\boldsymbol{A}_i}^{(1)}$ are an orthonormal basis of the column space of $\boldsymbol{A}_i$.

Acknowledgments. The authors would like to thank Ardavan (Ari) Afshar for detailed discussions about his work on minimizing Wasserstein distance between tensors, from which this work is derived. The authors would also like to thank Jimeng Sun, Cheng Qian, and Chaoqi Yang for fruitful discussions and for providing datasets which motivated this work. We would also like to thank Nick Dewaele and Nick Vannieuwenhoven for their useful comments and feedback on the manuscript. We would also like to thank the anonymous reviewers for their valuable feedback and suggestions which improved the manuscript by a substantial amount.

REFERENCES

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda, *A scalable optimization approach for fitting canonical tensor decompositions*, J. Chemometrics, 25 (2011), pp. 67–86.

[2] A. Afshar, K. Yin, S. Yan, C. Qian, J. C. Ho, H. Park, and J. Sun, *Swift: Scalable Wasserstein factorization for sparse nonnegative tensors*, in Proceedings of the AAAI Conference, 2021.

[3] A. Anandkumar, R. Ge, D. J. Hsu, S. M. Kakade, and M. Telgarsky, *Tensor decompositions for learning latent variable models*, J. Mach. Learn. Res., 15 (2014), pp. 2773–2832.

[4] G. Ballard, K. Hayashi, and R. Kannan, *Parallel Nonnegative CP Decomposition of Dense Tensors*, preprint, arXiv:1806.07985, 2018.

[5] G. Ballard, N. Knight, and K. Rouse, *Communication lower bounds for matricized tensor times Khatri-Rao product*, in Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2018, pp. 557–567.

[6] C. Battaglino, G. Ballard, and T. G. Kolda, *A practical randomized CP tensor decomposition*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 876–901.

[7] A. Bellet, A. Habrard, and M. Sebban, *A Survey on Metric Learning for Feature Vectors and Structured Data*, preprint, arXiv:1306.6709, 2013.

[8] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, *A blind source separation technique using second-order statistics*, IEEE Trans. Signal Process., 45 (1997), pp. 434–444.

[9] S. Biswal, H. Sun, B. Goparaju, M. B. Westover, J. Sun, and M. T. Bianchi, *Expert-level sleep scoring with deep neural networks*, J. Amer. Med. Inform. Assoc., 25 (2018), pp. 1643–1650.

[10] P. Breiding and N. Vannieuwenhoven, *The condition number of joint decompositions*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 287–309.

[11] R. Bro, *PARAFAC tutorial and applications*, Chemometrics Intell. Lab. Syst., 38 (1997), pp. 149–171.

[12] J. D. Carroll and J.-J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[13] J. Choi, X. Liu, S. Smith, and T. Simon, *Blocking optimization techniques for sparse tensor computation*, in Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2018, pp. 568–577.

[14] P. Comon, *Tensor diagonalization, a useful tool in signal processing*, IFAC Proc., 27 (1994), pp. 77–82.

[15] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, *Tensor decomposition of EEG signals: A brief review*, J. Neurosci. Methods, 248 (2015), pp. 59–69.

[16] C.-F. Cui, Y.-H. Dai, and J. Nie, *All real eigenvalues of symmetric tensors*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1582–1601.

[17] M. Cuturi and D. Avis, *Ground metric learning*, J. Mach. Learn. Res., 15 (2014), pp. 533–564.

[18] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, *The Mahalanobis distance*, Chemometrics Intell. Lab. Syst., 50 (2000), pp. 1–18.

[19] P. C. Mahalanobis, *On the generalised distance in statistics*, Proc. Natl. Instit. Sci. India, 2 (1936), pp. 49–55.

[20] N. Dewaele, P. Breiding, and N. Vannieuwenhoven, *The Condition Number of Many Tensor Decompositions Is Invariant Under Tucker Compression*, manuscript, 2021.

[21] R. A. Harshman, *Foundations of the PARAFAC Procedure: Models and Conditions for an Explanatory Multimodal Factor Analysis*, University of California at Los Angeles, Los Angeles, CA, 1970.

[22] K. Hayashi, G. Ballard, J. Jiang, and M. Tobia, *Shared Memory Parallelization of MTTKRP for Dense Tensors*, preprint, arXiv:1708.08976, 2017.

[23] C. J. Hillar and L.-H. Lim, *Most tensor problems are NP-hard*, J. ACM, 60 (2013), pp. 45:1–45:39.

[24] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, Stud. Appl. Math., 6 (1927), pp. 164–189.

[25] A. Hyvärinen, *Survey on Independent Component Analysis*, manuscript, 1999.

[26] L. Karlsson, D. Kressner, and A. Uschmajew, *Parallel algorithms for tensor completion in the CP format*, Parallel Comput., 57 (2016), pp. 222–234.

[27] O. Kaya, *High Performance Parallel Algorithms for Tensor Decompositions*, Ph.D. thesis, Université de Lyon, 2017.

[28] O. Kaya and Y. Robert, *Computing dense tensor decompositions with optimal dimension trees*, Algorithmica, 81 (2019), pp. 2092–2121.
[29] O. Kaya and B. Uçar, *Parallel CP Decomposition of Sparse Tensors Using Dimension Trees*, Ph.D. thesis, Inria-Research Centre Grenoble-Rhône-Alpes, 2016.
[30] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye, *Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG*, IEEE Trans. Biomed. Eng., 47 (2000), pp. 1185–1194.
[31] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[32] T. G. Kolda and J. R. Mayo, *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1095–1124.
[33] B. Kulis, *Metric learning: A survey*, Found. Trends Mach. Learn., 5 (2013), pp. 287–364.
[34] G. Li, L. Qi, and G. Yu, *The Z-eigenvalues of a symmetric tensor and its application to spectral hypergraph theory*, Numer. Linear Algebra Appl., 20 (2013), pp. 1001–1029.
[35] J. Li, K. Usevich, and P. Comon, *Globally convergent Jacobi-type algorithms for simultaneous orthogonal symmetric tensor diagonalization*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1–22.
[36] M. Liang and B. Zheng, *Further results on Moore-Penrose inverses of tensors with application to tensor nearness problems*, Comput. Math. Appl., 77 (2019), pp. 1282–1293.
[37] L.-H. Lim, *Singular values and eigenvalues of tensors: A variational approach*, in Proceedings of the 1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, IEEE, 2005, pp. 129–132.
[38] L. Ma and E. Solomonik, *Accelerating Alternating Least Squares for Tensor Decomposition by Pairwise Perturbation*, preprint, arXiv:1811.10573, 2018.
[39] L. Ma and E. Solomonik, *Efficient parallel CP decomposition with pairwise perturbation and multi-sweep dimension tree*, in Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2021, pp. 412–421.
[40] K. Maruhashi, F. Guo, and C. Faloutsos, *Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis*, in Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, IEEE, 2011, pp. 203–210.
[41] B. C. Mitchell and D. S. Burdick, *Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies*, J. Chemometrics, 8 (1994), pp. 155–168.
[42] D. Mitchell, N. Ye, and H. De Sterck, *Nesterov Acceleration of Alternating Least Squares for Canonical Tensor Decomposition*, preprint, arXiv:1810.05846, 2018.
[43] M. J. Mohlenkamp, *Musings on multilinear fitting*, Linear Algebra Appl., 438 (2013), pp. 834–852.
[44] K. R. Murphy, C. A. Stedmon, D. Graeber, and R. Bro, *Fluorescence spectroscopy and multi-way techniques. PARAFAC*, Anal. Methods, 5 (2013), pp. 6557–6566.
[45] D. Nion and L. De Lathauwer, *An enhanced line search scheme for complex-valued tensor decompositions. Application in DS-CDMA*, Signal Process., 88 (2008), pp. 749–755.
[46] P. Paatero, *A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis*, Chemometrics Intell. Lab. Syst., 38 (1997), pp. 223–242.
[47] A.-H. Phan, P. Tichavský, and A. Cichocki, *Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations*, IEEE Trans. Signal Process., 61 (2013), pp. 4834–4846.
[48] A.-H. Phan, P. Tichavský, and A. Cichocki, *Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 126–147.
[49] L. Qi, H. Chen, and Y. Chen, *Tensor Eigenvalues and their Applications*, Springer, Berlin, 2018.
[50] M. Rajih, P. Comon, and R. A. Harshman, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1128–1147.
[51] E. Robeva, *Orthogonal decomposition of symmetric tensors*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 86–102.
[52] E. Robeva and A. Seigal, *Singular vectors of orthogonally decomposable tensors*, Linear Multilinear Algebra, 65 (2017), pp. 2457–2471.
[53] M. D. Schatz, T. M. Low, R. A. van de Geijn, and T. G. Kolda, *Exploiting symmetry in tensors for high performance: Multiplication with symmetric tensors*, SIAM J. Sci. Comput., 36 (2014), pp. C453–C479.

[54] 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 Trans. Signal Process., 65, pp. 3551–3582.

[55] N. Singh, L. Ma, H. Yang, and E. Solomonik, *Comparison of accuracy and scalability of Gauss–Newton and alternating least squares for CANDECOMC/PARAFAC decomposition*, SIAM J. Sci. Comput., 43 (2021), pp. C290–C311.

[56] L. Sorber, I. Domanov, M. Van Barel, and L. Lathauwer, *Exact line and plane search for tensor optimization*, Comput. Optim. Appl., 63 (2016), pp. 121–142.

[57] L. Sorber, M. Van Barel, and L. De Lathauwer, *Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l_r,l_r,1) terms, and a new generalization*, SIAM J. Optim., 23 (2013), pp. 695–720.

[58] L. Sun, B. Zheng, C. Bu, and Y. Wei, *Moore-Penrose inverse of tensors via Einstein product*, Linear Multilinear Algebra, 64 (2016), pp. 686–698.

[59] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, et al., *PySCF: The Python-based simulations of chemistry framework*, Wiley Interdiscip. Rev. Comput. Mol. Sci., 8 (2018), e1340.

[60] P. Tichavský, A. H. Phan, and A. Cichocki, *A further improvement of a fast damped Gauss-Newton algorithm for CANDECOMP-PARAFAC tensor decomposition*, in Proceedings of the International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 5964–5968.

[61] P. Tichavský, A. H. Phan, and A. Cichocki, *Non-orthogonal tensor diagonalization*, Signal Process., 138 (2017), pp. 313–320.

[62] G. Tomasi and R. Bro, *PARAFAC and missing values*, Chemometrics Intell. Lab. Syst., 75 (2005), pp. 163–180.

[63] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Statist. Data Anal., 50 (2006), pp. 1700–1734.

[64] A. Uschmajew, *Local convergence of the alternating least squares algorithm for canonical tensor approximation*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 639–652.

[65] K. Usevich, J. Li, and P. Comon, *Approximate matrix and tensor diagonalization by unitary transformations: Convergence of Jacobi-type algorithms*, SIAM J. Optim., 30 (2020), pp. 2998–3028.

[66] N. Vannieuwenhoven, *Condition numbers for the tensor rank decomposition*, Linear Algebra Appl., 535 (2017), pp. 35–86.

[67] N. Vannieuwenhoven, K. Meerbergen, and R. Vandebril, *Computing the gradient in optimization algorithms for the CP decomposition in constant memory through tensor blocking*, SIAM J. Sci. Comput., 37 (2015), pp. C415–C438.

[68] G. Zen, E. Ricci, and N. Sebe, *Simultaneous ground metric learning and matrix factorization with earth mover's distance*, in Proceedings of the 22nd International Conference on Pattern Recognition, IEEE, 2014, pp. 3690–3695.