EXPLOITING EFFICIENT REPRESENTATIONS IN LARGE-SCALE TENSOR DECOMPOSITIONS[*]

NICO VERVLIET[†], OTTO DEBALS[†], AND LIEVEN DE LATHAUWER[†]

Abstract. Decomposing tensors into simple terms is often an essential step to discover and understand underlying processes or to compress data. However, storing the tensor and computing its decomposition is challenging in a large-scale setting. Though, in many cases a tensor is structured, i.e., it can be represented using few parameters: a sparse tensor is determined by the positions and values of its nonzeros, a polyadic decomposition by its factor matrices, a Tensor Train by its core tensors, a Hankel tensor by its generating vector, etc. The complexity of tensor decomposition algorithms can be reduced significantly in terms of time and memory if these efficient representations are exploited directly. Only few core operations such as norms and inner products need to be specialized to achieve this, thereby avoiding the explicit construction of multiway arrays. To improve the interpretability of tensor models, constraints are often imposed or multiple datasets are fused through joint factorizations. While imposing these constraints prohibits the use of traditional compression techniques, our framework allows constraints and compression, as well as other efficient representations, to be handled trivially as the underlying optimization variables do not change. To illustrate this, large-scale nonnegative tensor factorization is performed using MLSVD and Tensor Train compression. We also show how vector and matrix data can be analyzed using tensorization while keeping a vector or matrix complexity through the concept of implicit tensorization, as illustrated for Hankelization and Löwnerization. The concepts and numerical properties are extensively investigated with experiments.

Key words. tensor decompositions, canonical polyadic decomposition, PARAFAC, Tucker, block term decomposition, Hankel, Löwner, large-scale, nonnegative factorization, tensorization

AMS subject classifications. 15A69

1. Introduction.

In signal processing and data analysis, tensors are often given as multiway arrays of numerical values. Tensor decompositions are then used to discover patterns, separate signals, model behavior, cluster similar phenomena, detect anomalies, predict missing data, etc. More concrete examples include intrusion detection in computer networks [67], analysis of food samples [6], crop classification using hyperspectral imaging [95], direction of arrival estimation using a grid of antennas [74] and modeling melting temperatures of alloys [93]. In these contexts, the canonical polyadic decomposition (CPD) and the block term decomposition (BTD) are common. The components, e.g., the rank-1 terms in a CPD, recovered by these decompositions can often be interpreted directly thanks to relatively mild uniqueness results compared to matrices. If prior knowledge is available, constraints such as nonnegativity, orthogonality, sparsity, smoothness, Vandermonde structure, (block) Toeplitz structure and so on can be imposed on the components to further improve their in-

[*]Submitted to the editors on October 16, 2017.

Funding: Nico Vervliet is supported by an Aspirant Grant from the Research Foundation — Flanders (FWO). Otto Debals is supported by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT). Research furthermore supported by: (1) Flemish Government: FWO: projects: G.0830.14N, G.0881.14N; (2) EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Advanced Grant: BIOTENSORS (n° 339804). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information; (3) KU Leuven Internal Funds C16/15/059.

[†]KU Leuven, Dept. of Electrical Engineering ESAT/STADIUS, Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven, Belgium; and Group Science, Engineering and Technology, KU Leuven - Kulak, E. Sabbelaan 53, 8500 Kortrijk, Belgium (Nico.Vervliet@kuleuven.be, Otto.Debals@gmail.com, Lieven.DeLathauwer@kuleuven.be).

terpretation in specific applications. When multiple datasets describing (partially) the same phenomenon are available, coupling or data fusion allows more meaningful patterns to be extracted. More examples can be found in several overview papers; see, e.g., [14, 52, 75]. While most publications tackle a single type of structure or constraint, we discuss a general framework in this paper.

Given as dense arrays of numerical values, the cost of storing and processing tensors tends to increase quickly, especially for high orders: for an Nth-order tensor of size $I \times I \times \cdots \times I$ the number of entries is I^N. To deal with these large-scale tensors, various techniques have emerged to handle their decompositions, e.g., by deliberately sampling entries [92, 93], through randomization [89], through compression and/or using parallelism, e.g., [13, 43, 55, 67, 76, 79], or by cross approximation [9, 56, 63, 66]. Such techniques rely on the fact that tensors are structured and can be represented efficiently using fewer parameters than the number of entries in the dense array. This is also our basic assumption in this paper. Moreover, we assume that structure is exploitable to reduce the computational and memory complexity from $\mathcal{O}$ (tensor entries) to $\mathcal{O}$ (parameters). Let us consider three important types of efficient representation that allow such exploitation: compact, exact representation, compression-based representation and (implicit) tensorization.

For the first type, the tensor has an *exact and known structure* that can be represented compactly. A sparse tensor, for instance, is represented compactly by the indices and values of the nonzeros, or more efficiently using tensor extensions of compressed sparse column or row formats [4, 77]. For sparse tensors, the efficient or compact representation has already been exploited extensively; see, e.g., [4, 13, 41, 42, 43, 44, 45, 53, 67, 77, 78, 79]. In some applications the tensor is given as a polyadic decomposition; the factor matrices then form the compact representation [57, 70].

Second, in the case of *compression-based representations*, the tensor is first approximated using a decomposition that is easier or cheaper to compute and this approximation is then used to speed up subsequent computations. For example, in the experiment in subsection 4.2 a rank-10 CPD is computed directly from a $1024 \times 1024 \times 1024$ tensor which takes 39 s, while the computation time for the decomposition is reduced to 0.6 s after a MLSVD compression step which takes 8 s. Common examples of this type are a truncated MLSVD [22], a Hierarchical Tucker approximation [31, 35] or a Tensor Train (TT) approximation [65]. MLSVD compression is used extensively as a preprocessing step to speed up the computation of an unconstrained CPD, which follows from the CANDELINC model [12, 50]: thanks to the orthogonal compression the rank-1 structure is preserved and the CPD can be computed from the compressed core tensor. To design effective updating algorithms, a previously computed compact representation of the old data combined with a newly arrived slice is used in [85]. In scientific computing, recompression or rounding are often used to lower the multilinear rank of a PD or a Tucker decomposition [48, 49, 64, 72, 73], or to lower the TT rank of a tensor given as a PD [65] or a TT approximation [37, 60]. Other examples include the approximation of a matrix or tensor by a greedy orthogonal rank-1 decomposition [51], a hierarchical Kronecker tensor product decomposition [34] or a two-level rank-$(r_1, \ldots, r_d)$ decomposition [47].

The third type is *implicit tensorization.* Tensorization involves mapping vectors or matrices to higher-order tensors [14, 25]. (Note that we use a much broader definition of tensorization than, e.g., the quantization concept in scientific computing; see, e.g., [32] and references therein.) By decomposing the obtained tensors, mild uniqueness conditions of tensor decompositions can be leveraged, which are key in a variety of applications such as blind source separation (BSS). However, for some types of tensorization, the number of data points increases drastically. For example, third-order Hankelization of a vector of length M results in a tensor with $\mathcal{O}\left(M^3\right)$ entries [74], while Löwnerization [27] of K observations of mixed rational functions, sampled in M points, may result in a tensor of dimensions $M/2 \times M/2 \times K$. In both examples, the signal length M that can be handled is limited by the respective cubic and quadratic dependence of the number of entries on M. To avoid this data explosion, we propose for this type of problems the concept of implicit tensorization: by operating on the underlying data directly, no explicit construction of the tensor is required. This way, implicit tensorization alleviates computation and memory cost if the data underlying the tensor has few parameters, and the structure of this representation can be exploited easily, e.g., through fast Fourier transforms (FFT); see section 3. Examples are Hankelization, Löwnerization, Toeplitzization, segmentation with overlap and outer product structures [5, 25]. Via implicit tensorization, deterministic BSS techniques become a viable alternative to classical methods such as ICA, even for datasets of realistic sizes. This is illustrated in subsection 4.3.

Optimization-based techniques are often used in, e.g., signal processing, data analysis and machine learning, to compute a low rank or a low multilinear rank approximation $\mathcal{M}$ of a (structured) tensor $\mathcal{T}$. For example, for a CPD, $\mathcal{M} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ and the following minimization problem is solved for the factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} \left|\left|\mathcal{M} - \mathcal{T}\right|\right|_{\mathrm{F}}^2 .$$

Often additional constraints are imposed to facilitate the interpretation of the resulting factor matrices, or multiple datasets are analyzed jointly. Many of these constraints can be implemented via projection, active set methods, penalties or using parametric constraints; see [90] for an overview. As these constraints are usually imposed on factor matrices, classical complexity reduction techniques such as MLSVD compression [7] can often not be used as the constraint is not preserved. For example, if $\mathbf{A} \in \mathbb{R}^{I \times R}$ is constrained to be nonnegative and $\mathbf{U}$ is an orthogonal basis for $\mathbf{A}$, then $\mathbf{A} = \mathbf{U}\hat{\mathbf{A}} \geq \mathbf{0}$ does not imply that $\hat{\mathbf{A}} \geq 0$. Hence, when using MLSVD compression, one cannot decompose the core tensor and impose nonnegativity. As the original factors remain the optimization variables in our approach, constrained decompositions can be computed using a compressed or structured tensor. An illustration in the case of compression is given in Figure 1.1.

1.1. Contributions. We explain how efficient representations originating from each of the three types of efficient representations (compact, exact representation, compression-based representation and (implicit) tensorization) can be exploited in optimization-based algorithms for the computation of a CPD, a decomposition into multilinear rank-$(L_r, L_r, 1)$ terms (LL1), an LMLRA or a general BTD. To be able to exploit this efficient representation, the computation of the residual is avoided by expanding the least squares loss function into inner products and norms. For example, for the CPD we have

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} \left|\left|\mathcal{M} - \mathcal{T}\right|\right|_{\mathrm{F}}^2 = \min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} \left|\left|\mathcal{M}\right|\right|^2 - \langle \mathcal{M}, \mathcal{T} \rangle + \frac{1}{2} \left|\left|\mathcal{T}\right|\right|^2 . \tag{1.1}$$

Each of these norms and the inner product, as well as the structured matrix-matrix products required for the gradient can be implemented such that the structure underlying the efficient representation is exploited. A number of papers have already proposed similar ideas for the computation of a CPD or LMLRA in the specific cases

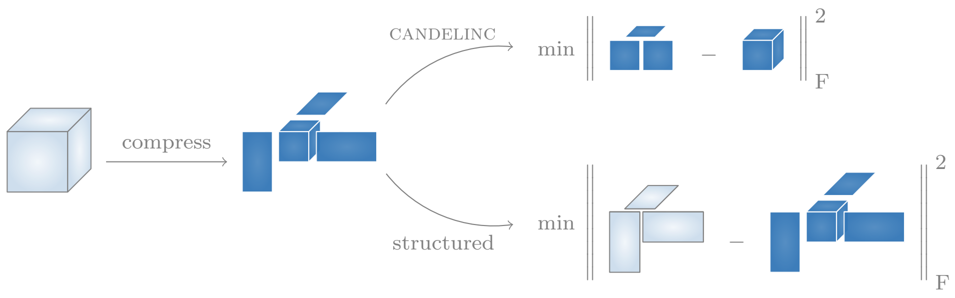

FIGURE 1.1. *In contrast to the classical CANDELINC model, the optimization variables remain the same in our framework when using compression. Instead of doing computations on a small core tensor, which is possible because the orthogonal transformation preserves low rank structure, we compute the decomposition using the compact Tucker model, factor matrices included. This requires that all operations are formulated in terms of the model and are implemented efficiently. (Figure adapted from [90].)*

of sparse tensors [4, 13, 41, 42, 43, 44, 45, 53, 67, 77, 78, 79], or polyadic and Tucker formats [96, 97, 98]. We illustrate that these ideas can be extended to more types of structured tensors, including tensors resulting from tensorization techniques. Moreover, we show that, in contrast to previous results, both first and second-order optimization algorithms can be used and that constraints and joint factorizations are handled trivially, which we illustrate in the experiments by using Gauss–Newton type methods in combination with active set and parametric constraints. We also pay attention to often neglected numerical accuracy issues. Finally, we show that our approach enables enormous speedups for large-scale problems even when constraints are imposed or when multiple datasets are jointly factorized, in contrast to the traditional compression approach using the CANDELINC model [7] which cannot be used in these cases, as the constraints are not preserved by orthogonal compression. We demonstrate this experimentally for large-scale nonnegative tensor factorization using MLSVD and TT compression and for implicit Hankelization of signals with up to 500 000 samples. Matlab implementations for the framework are available in Tensorlab [94].

1.2. Outline. After discussing the notation in the remainder of this section, section 2 explains how efficient representations of structured tensors can be exploited by rewriting the Frobenius norm objective function and gradient for the CPD, LL1, LMLRA and BTD. We give an overview of implementations of the four core operations — norm, inner product, matricized tensor times Khatri–Rao product and matricized tensor times Kronecker product — in section 3 for structured tensors given in the polyadic, Tucker or TT format and derive implementations for Hankel and Löwner tensors. Finally, numerical experiments are given in section 4. We discuss the influence of the condition number and illustrate the performance for compression-based nonnegative CPD and blind separation of exponential polynomials through implicit Hankelization. In a last experiment, we show how ECG signals can be analyzed efficiently using Löwnerization.

1.3. Notation. Scalars, vectors, matrices and tensors are denoted by lower case, e.g., a, bold lower case, e.g., $\mathbf{a}$, bold upper case, e.g., $\mathbf{A}$ and calligraphic, e.g., $\mathcal{T}$, letters, respectively. $\mathbb{K}$ denotes either $\mathbb{R}$ or $\mathbb{C}$. Sets are indexed by superscripts within parentheses, e.g., $\mathbf{A}^{(n)}$, $n = 1, \ldots, N$. A mode-n vector is the generalization of a column (mode-1) and row (mode-2) vector and is defined by fixing all but the nth index of an Nth-order tensor. The mode-n unfolding of a tensor $\mathcal{T}$ is denoted by $\mathbf{T}_{(n)}$ and has the mode-n vectors as its columns. The mode-(m,n) unfolding is defined analogously and is denoted by $\mathbf{T}_{(m,n)}$. The vectorization operator $\operatorname{vec}(\mathcal{T})$ stacks all mode-1 vectors into a column vector.

A number of products are needed. The mode-n tensor-matrix product is denoted by $\mathcal{T} \cdot_n \mathbf{A}$. The Kronecker and Hadamard, or element-wise, product are denoted by $\otimes$ and $*$, respectively. The column-wise (row-wise) Khatri–Rao products between two matrices, denoted by $\odot$ and $\odot^{\mathrm{T}}$, respectively, are defined as the column-wise (row-wise) Kronecker product. As we assume that the first index runs faster than the second and so on, we define the following shorthand notations to simplify expressions:

$$\bigotimes_{n} \equiv \bigotimes_{n=N}^{1} \qquad\qquad \bigotimes_{k\neq n} \equiv \bigotimes_{\substack{k=N\\k\neq n}}^{1},$$

where N is the order of the tensor and indices are traversed in reverse order. Similar definitions are used for $\odot$, $\odot^{\mathrm{T}}$ and $*$. To reduce the number of parentheses, we assume the matrix product takes precedence over $\otimes$, $\odot$, $\odot^{\mathrm{T}}$ and $*$, e.g., $\mathbf{AB} \odot \mathbf{CD} \equiv (\mathbf{AB}) \odot (\mathbf{CD})$. The complex conjugate, transpose and conjugated transpose are denoted by $\bar{\cdot}$, $\cdot^{\mathrm{T}}$ and $\cdot^{\mathrm{H}}$, respectively. The column-wise concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$ is written as $\mathbf{x} = [\mathbf{a}; \mathbf{b}]$ and is a shorthand for $\mathbf{x} = \begin{bmatrix}\mathbf{a}^{\mathrm{T}} & \mathbf{b}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$. The inner product between $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\langle \mathcal{A}, \mathcal{B} \rangle = \operatorname{vec}(\mathcal{B})^{\mathrm{H}} \operatorname{vec}(\mathcal{A})$. The Frobenius norm is denoted by $||\cdot||$. $\operatorname{Re}(\cdot)$ returns the real part of a complex number. $\mathbf{I}_I$ is the $I \times I$ identity matrix and $\mathbf{1}_I$ is a length I column vector with ones. Finally, $\mathbf{V}^f$ is the row-wise 'flipped' version of $\mathbf{V} \in \mathbb{K}^{I\times J}$, i.e., $\mathbf{V}^f(i,:) = \mathbf{V}(I-i+1,:)$ for $i = 1,\ldots,I$.

2. Exploiting efficient representations.

Many tensor decompositions used in signal processing and data analysis can be computed by minimizing the least squares (LS) error between the approximation and the given tensor $\mathcal{T}$

$$\min_{\mathbf{z}} f(\mathbf{z}) \quad \text{with} \quad f(\mathbf{z}) = \frac{1}{2} ||\mathcal{M}(\mathbf{z}) - \mathcal{T}||^2, \tag{2.1}$$

in which $\mathcal{M}(\mathbf{z})$ is a tensor decomposition with variables $\mathbf{z}$. Various algorithms have been proposed to solve (2.1), including alternating least squares (ALS) [8,11,23,36,54], nonlinear conjugate gradient (NCG) [1], quasi-Newton (qN) [82] and Gauss–Newton (GN) [82,83,84] algorithms. We first review the notation for common tensor decompositions in subsection 2.1. In subsection 2.2 we give an overview of LS optimization theory for algorithms using gradients (first-order information) and Hessian approximations (second-order information) and show that many common approximations to the Hessian do not depend on the data $\mathcal{T}$. Hence, second-order algorithms can be used without additional changes in implementation compared to first-order algorithms. Finally, the core operations required to reduce the computational complexity are introduced in subsection 2.3 by rewriting $f(\mathbf{z})$ similar to (1.1).

2.1. Overview of tensor decompositions.

The techniques derived in this paper focus on four main decompositions, which can be grouped into two families from an optimization point-of-view: CPD-based and BTD-based algorithms. Note that we use different families compared to the usual division between decompositions computed using numerical linear algebra techniques such as singular value decompositions (SVD), e.g., MLSVD, TT and hierarchical Tucker, and decompositions usually computed through optimization (CPD, LL1, BTD). Here, we focus mainly on notation; for pointers to uniqueness results and applications, we refer to the overview papers [14, 32, 33, 52, 75].

CPD-based algorithms. The (canonical) polyadic decomposition (CPD) writes an Nth-order tensor as a (minimal) sum of R rank-1 terms, each of which is an outer product, denoted by $\otimes$, of N nonzero factor vectors $\mathbf{a}_r^{(n)}$:

$$\mathcal{M}_{\text{CPD}}(\mathbf{z}) = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \otimes \cdots \otimes \mathbf{a}_r^{(N)} := \left[\!\left[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)} \right]\!\right].$$

Each factor matrix $\mathbf{A}^{(n)}$ contains $\mathbf{a}_r^{(n)}$, $r = 1, \ldots, R$, as its columns and the variables $\mathbf{z}$ are the vectorized factor matrices $\mathbf{A}^{(n)}$, i.e., $\mathbf{z} = \left[\text{vec}\left(\mathbf{A}^{(1)}\right); \ldots; \text{vec}\left(\mathbf{A}^{(N)}\right)\right]$. (The underlying variables will always be denoted by $\mathbf{z}$.) Depending on the field, the CPD is also known as PARAFAC, CANDECOMP or tensor/separation rank decomposition.

The decomposition into multilinear rank-$(L_r, L_r, 1)$ terms (LL1) is accommodated within this family as it is often computed as a constrained CPD [8, 23, 82]:

$$\mathcal{M}_{\text{LL1}}(\mathbf{z}) = \sum_{r=1}^{R} (\mathbf{A}_r \mathbf{B}_r^{\mathrm{T}}) \otimes \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{CP}]\!],$$

in which $\mathbf{A}_r \in \mathbb{K}^{I_1 \times L_r}$, $\mathbf{B}_r \in \mathbb{K}^{I_2 \times L_r}$ for $r = 1, \ldots, R$, $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \ldots & \mathbf{A}_R \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \ldots & \mathbf{B}_R \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} \mathbf{c}_1, \ldots, \mathbf{c}_R \end{bmatrix}$ and $\mathbf{P}$ is a binary matrix replicating column $\mathbf{c}_r$ L_r times, i.e., $\mathbf{P} = \text{blockdiag}(\mathbf{1}_{L_1}^{\mathrm{T}}, \ldots, \mathbf{1}_{L_R}^{\mathrm{T}})$. Similar to the CPD, $\mathbf{z} = \left[\text{vec}\left(\mathbf{A}\right); \text{vec}\left(\mathbf{B}\right); \text{vec}\left(\mathbf{C}\right)\right]$. The LL1 decomposition is similar to PARALIND [8]. CONFAC generalizes this decomposition by allowing an arbitrary full row rank matrix $\mathbf{P}$ in every mode [17]. In [82], the decomposition into (rank-$L_r \otimes$ rank-1) terms is discussed as one generalization of $\mathcal{M}_{\text{LL1}}$ to higher-order tensors.

BTD-based algorithms. The low-multilinear rank approximation (LMLRA) or Tucker decomposition writes a tensor as a multilinear transformation of a core tensor $\mathcal{S} \in \mathbb{K}^{J_1 \times \cdots \times J_N}$ by the factor matrices $\mathbf{U}^{(n)} \in \mathbb{K}^{I_n \times J_n}$, $n = 1, \ldots, N$:

$$\mathcal{M}_{\text{LMLRA}}(\mathbf{z}) = \mathcal{S} \cdot_1 \mathbf{U}^{(1)} \cdots \cdot_N \mathbf{U}^{(N)}.$$

An LMLRA can be computed using SVDs of tensors unfoldings, i.e., as a (truncated) multilinear singular value decomposition (MLSVD) [21, 88], by higher-order orthogonal iteration [22] or through optimization [39, 40, 82, 83].

The more general block term decomposition writes a tensor as a sum of R multilinear rank-$(J_{r1}, J_{r2}, \ldots, J_{rN})$ terms [18]:

$$\mathcal{M}_{\text{BTD}}(\mathbf{z}) = \sum_{r=1}^{R} \mathcal{S}^{(r)} \cdot_1 \mathbf{U}^{(r,1)} \cdots \cdot_N \mathbf{U}^{(r,N)}.$$

The difference between $\mathcal{M}_{\text{LMLRA}}$ and $\mathcal{M}_{\text{BTD}}$ is the additional summation over $r = 1, \ldots, R$. For simplicity of notation, we only use $\mathcal{M}_{\text{LMLRA}}$ in the derivations in this paper.

2.2. Optimization for least squares problems. In gradient-based optimization, an initial guess for the variables $\mathbf{z}_0$ is iteratively refined as

$$\mathbf{z}_k = \mathbf{z}_{k-1} + \alpha_k \mathbf{p}_k,$$

with the step direction $\mathbf{p}_k$ found by solving

$$\mathbf{H}_k \mathbf{p}_k = -\bar{\mathbf{g}}_k, \tag{2.2}$$

in which $\mathbf{g}_k$ and $\mathbf{H}_k$ are the gradient and the Hessian (or an approximation) of $f(\mathbf{z})$ w.r.t. $\mathbf{z}$ in (2.1), respectively, assuming $\mathcal{M}(\mathbf{z})$ is continuous in $\mathbf{z}$ and independent of $\bar{\mathbf{z}}$ [81]. The step size α_k ensures sufficient decrease in the objective function value, e.g., through line search. Alternatively, $\mathbf{z}$ can be updated using trust-region approaches or using plane search [58, 80, 82]. The linear system (2.2) can be solved using direct (pseudo)inversion or using (preconditioned) conjugate gradients [58].

For the LS objective function (2.1), various Hessian approximations are used depending on the algorithm, e.g.,

- $\mathbf{H}_k = \mathbf{I}$ for gradient descent,
- $\mathbf{H}_k = \mathbf{H}_{k-1} + \mathbf{U}_{k-1} + \mathbf{V}_{k-1}$ with $\mathbf{U}_{k-1}$ and $\mathbf{V}_{k-1}$ symmetric rank-1 matrices constructed using previous gradients and updates of $\mathbf{z}$ for BFGS,
- $\mathbf{H}_k = \mathbf{J}_k^{\mathrm{H}} \mathbf{J}_k$ with $\mathbf{J}_k = \frac{\partial}{\partial \mathbf{z}} \mathrm{vec}\left(\mathcal{M}(\mathbf{z}) - \mathcal{T}\right)\big|_{\mathbf{z}=\mathbf{z}_k} = \frac{\partial}{\partial \mathbf{z}} \mathrm{vec}\left(\mathcal{M}(\mathbf{z})\right)\big|_{\mathbf{z}=\mathbf{z}_k}$ for Gauss–Newton,
- $\mathbf{H}_k = \mathbf{J}_k^{\mathrm{H}} \mathbf{J}_k + \lambda_k \mathbf{I}$ for Levenberg–Marquardt.

None of these approximations of the Hessian depend on the tensor $\mathcal{T}$. Therefore, only the objective function evaluation and the gradient computation require adaptation to exploit structure in $\mathcal{T}$. Note that this also holds for ALS algorithms.

2.3. Exploiting efficient representations. Due to the explicit construction of the residual tensor $\mathcal{F} = \mathcal{M}(\mathbf{z}) - \mathcal{T}$ in (2.1) and the computation of the gradient, the per-iteration complexity is governed by the number of entries in the tensor, i.e., $\mathcal{O}\left(\prod_n I_n\right)$. To reduce the complexity, we assume $\mathcal{T}$ is a function of a limited number of parameters. The parameters can, for instance, be the set of positions and values of nonzero entries in a sparse tensor, the factor matrices of a CPD or the underlying signal in a Hankelized tensor. To exploit the structure in $\mathcal{T}$ and to avoid the creation of the residual tensor, we rewrite the objective function and gradients, similar to (1.1), which reveals the core operations. More concretely, we have

$$f(\mathbf{z}) = \frac{1}{2}\left|\left|\mathcal{M}(\mathbf{z})\right|\right|^2 - \langle \mathcal{M}(\mathbf{z}), \mathcal{T}\rangle + \frac{1}{2}\left|\left|\mathcal{T}\right|\right|^2, \tag{2.3}$$

which requires the efficient computation of the norms of the model and the tensor, and the inner product between the model and the tensor. Similarly, the gradient which is given by $\bar{\mathbf{g}} = \mathbf{J}^{\mathrm{H}} \mathrm{vec}\left(\mathcal{F}\right)$ with $\mathbf{J} = \frac{\partial \mathrm{vec}(\mathcal{F})}{\partial \mathbf{z}}$ the Jacobian matrix, is rewritten by grouping the model and data dependent terms as

$$\begin{aligned} \bar{\mathbf{g}} &= \underbrace{\mathbf{J}^{\mathrm{H}} \mathrm{vec}\left(\mathcal{M}(\mathbf{z})\right)} - \underbrace{\mathbf{J}^{\mathrm{H}} \mathrm{vec}\left(\mathcal{T}\right)} \\ &= \qquad \bar{\mathbf{g}}_{\mathcal{M}} \qquad - \qquad \bar{\mathbf{g}}_{\mathcal{T}}. \end{aligned} \tag{2.4}$$

(In the derivations, we work with $\mathbf{g}$ rather than $\bar{\mathbf{g}}$.) The core operations are the MTKRPROD for the CPD-based algorithms and the MTKRONPROD for the BTD-based algorithms as shown below. Extensions to constrained and coupled constraints conclude this section.

CPD-based algorithms. Let us partition $\mathbf{g}$ according to the variables $\mathbf{z}$, i.e., in the case $\mathcal{M}(\mathbf{z})$ is a CPD, we have

$$\mathbf{g} = \left[\mathrm{vec}\left(\mathbf{G}^{(1)}\right); \ldots; \mathrm{vec}\left(\mathbf{G}^{(N)}\right)\right],$$

with

$$\operatorname{vec}\left(\mathbf{G}^{(n)}\right) = \frac{\partial f}{\partial \operatorname{vec}\left(\mathbf{A}^{(n)}\right)} \qquad\qquad n = 1, \ldots, N.$$

By grouping the model and data dependent terms as in (2.4), the gradients become

$$\begin{aligned} \mathbf{G}_{\mathcal{M}}^{(n)} &= \bar{\mathbf{A}}^{(n)} \left(\mathop{\ast}_{k\neq n} \mathbf{A}^{(k)^{\mathrm{H}}} \mathbf{A}^{(k)} \right), \\ \mathbf{G}_{\mathcal{T}}^{(n)} &= \bar{\mathbf{T}}_{(n)} \left(\mathop{\odot}_{k\neq n} \mathbf{A}^{(k)} \right). \end{aligned} \tag{2.5}$$

It is clear that $\mathbf{G}_{\mathcal{M}}^{(n)}$ can be computed efficiently after forming the inner products $\mathbf{A}^{(k)^{\mathrm{H}}}\mathbf{A}^{(k)}$, $k = 1, \ldots, N$. We therefore focus on the operation in (2.5) which involves the common matricized tensor times Khatri–Rao product (MTKRPROD):

$$\bar{\mathbf{T}}_{(n)} \left(\mathbf{A}^{(N)} \odot \mathbf{A}^{(N-1)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)} \right). \tag{MTKRPROD}$$

In the case $\mathcal{M}(\mathbf{z})$ is an LL1 decomposition, $\mathbf{A}^{(3)} = \mathbf{CP}$ is used to form the inner products and to compute the MTKRPROD. From the chain rule follows that for $n = 3$, $\mathbf{G}^{(n)}$, which is now the derivative w.r.t. $\mathbf{C}$, i.e., $\mathbf{G}^{(3)} = \frac{\partial f}{\partial \mathbf{C}}$, is given by

$$\begin{aligned} \mathbf{G}_{\mathcal{M}}^{(3)} &= \bar{\mathbf{A}}^{(3)} \left(\mathop{\ast}_{k\neq 3} \mathbf{A}^{(k)^{\mathrm{H}}} \mathbf{A}^{(k)} \right) \mathbf{P}^{\mathrm{T}}, \\ \mathbf{G}_{\mathcal{T}}^{(3)} &= \bar{\mathbf{T}}_{(3)} \left(\mathop{\odot}_{k\neq 3} \mathbf{A}^{(k)} \right) \mathbf{P}^{\mathrm{T}}. \end{aligned}$$

BTD-based algorithms. Similarly, in the case $\mathcal{M}(\mathbf{z})$ is an LMLRA, the gradient is partitioned as

$$\mathbf{g} = \left[\operatorname{vec}\left(\mathcal{G}^{(0)}\right); \operatorname{vec}\left(\mathbf{G}^{(1)}\right); \ldots; \operatorname{vec}\left(\mathbf{G}^{(N)}\right)\right],$$

with

$$\begin{aligned} \operatorname{vec}\left(\mathcal{G}^{(0)}\right) &= \frac{\partial f}{\partial \operatorname{vec}\left(\mathcal{S}\right)} \\ \operatorname{vec}\left(\mathbf{G}^{(n)}\right) &= \frac{\partial f}{\partial \operatorname{vec}\left(\mathbf{U}^{(n)}\right)}, \qquad\qquad n = 1, \ldots, N. \end{aligned}$$

More specifically, the gradient expressions for the core tensors are given by

$$\begin{aligned} \mathcal{G}_{\mathcal{M}}^{(0)} &= \bar{\mathcal{S}} \cdot_1 \left(\mathbf{U}^{(1)^{\mathrm{T}}} \bar{\mathbf{U}}^{(1)} \right) \cdots \cdot_N \left(\mathbf{U}^{(N)^{\mathrm{T}}} \bar{\mathbf{U}}^{(N)} \right), \\ \mathcal{G}_{\mathcal{T}}^{(0)} &= \bar{\mathcal{T}} \cdot_1 \mathbf{U}^{(1)^{\mathrm{T}}} \cdots \cdot_N \mathbf{U}^{(N)^{\mathrm{T}}}, \end{aligned} \tag{2.6}$$

and the expressions for the factors matrices by

$$\begin{aligned} \mathbf{G}_{\mathcal{M}}^{(n)} &= \bar{\mathbf{U}}^{(n)} \bar{\mathbf{S}}_{(n)} \left(\mathop{\otimes}_{k\neq n} \mathbf{U}^{(k)^{\mathrm{H}}} \mathbf{U}^{(k)} \right) \mathbf{S}_{(n)}^{\mathrm{T}}, \\ \mathbf{G}_{\mathcal{T}}^{(n)} &= \bar{\mathbf{T}}_{(n)} \left(\mathop{\otimes}_{k\neq n} \mathbf{U}^{(k)} \right) \mathbf{S}_{(n)}^{\mathrm{T}}, \end{aligned} \tag{2.7}$$

for $n = 1, \ldots, N$. In the case of more general BTD with R terms, an additional summation is introduced; see, e.g., [83]. As (2.6) can be written as $\mathrm{vec}\left(\mathcal{G}_{\mathcal{T}}^{(0)}\right)^{\mathrm{T}} = \mathrm{vec}\left(\mathcal{T}\right)^{\mathrm{H}}\left(\otimes_n \mathbf{U}^{(n)}\right)$, both (2.6) and (2.7) require a vectorized or matricized tensor times Kronecker product (MTKRONPROD):

$$\text{(MTKRONPROD)} \qquad \bar{\mathbf{T}}_{(n)}\left(\mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \mathbf{U}^{(n+1)} \otimes \mathbf{U}^{(n-1)} \otimes \cdots \otimes \mathbf{U}^{(1)}\right).$$

Note that

$$\mathrm{vec}\left(\mathcal{G}_{\mathcal{T}}^{(0)}\right)^{\mathrm{T}} = \mathrm{vec}\left(\mathbf{U}^{(1)^{\mathrm{T}}}\bar{\mathbf{T}}_{(1)}\left(\bigotimes_{k\neq 1}\mathbf{U}^{(k)}\right)\right),$$

hence the MTKRONPROD from (2.7) can be reused, although this may lead to a higher complexity if the construction of the $I \times R^{N-1}$ matrix $\left(\otimes_{k\neq n}\mathbf{U}^{(k)}\right)\mathbf{S}_{(n)}^{\mathrm{T}}$ can be avoided such as, e.g., for the polyadic format.

Core operations, constraints and coupling. In order to reduce the complexity from $\mathcal{O}$(tensor entries) to $\mathcal{O}$(parameters), structure exploiting implementations of the Frobenius norm and inner products with CPD and BTD models are required to evaluate the objective function (2.3) as well as implementations for the MTKRPROD and the MTKRONPROD for CPD and BTD gradients, respectively. Examples of such implementations are discussed in section 3. In contrast to approaches taken in, e.g., [7, 15], the original factor matrices and core tensors remain the optimization variables. Therefore, algorithms for coupling datasets, e.g., [2, 83], only require new implementations of the core operations. Similarly, parametric constraints such as nonnegativity, orthogonality or Vandermonde structure can be handled using the chain rule as in [83], as well as other types of constraints involving projecting onto a feasible set; see, e.g., [38, 46]. An overview of techniques for constrained optimization for tensor decompositions is given in [90]. Concrete examples are given in the experiments in section 4.

3. Operations on efficient representations.

In the previous section, the computation of the norm, inner product, MTKRPROD and MTKRONPROD have been identified as the four key operations to allow tensor decomposition algorithms to work on structured tensors. In this section, we show that it is relatively straightforward to exploit efficient representations and to avoid the construction of the full tensor. We show the complexity is indeed governed by the number of parameters in the efficient representation, rather than the number of entries in the tensor. Unless stated otherwise, only third-order tensors $\mathcal{T} \in \mathbb{K}^{I\times I\times I}$ are considered to simplify notations and complexity expressions. In this section, we first give an overview of results for the polyadic (subsection 3.1) and Tucker (subsection 3.2) formats as efficient representations of $\mathcal{T}$, since expressions involving one of these formats occur in all of the discussed decompositions. Then, efficient implementations for the TT format (subsection 3.3) are discussed and two implicit tensorizations — Hankelization (subsection 3.4) and Löwnerization (subsection 3.5) — are derived. These and other implementations are available in Tensorlab [94].

We derive expressions for the two computational families: the rank-R CPD $\mathcal{M}_{\mathrm{CPD}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ and the multilinear rank-(R, R, R) LMLRA $\mathcal{M}_{\mathrm{LMLRA}} = \mathcal{S} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V} \cdot_3 \mathbf{W}$. The extension to a BTD introduces additional summations and is omitted here; see [83]. For the MTKRPROD and MTKRONPROD operations, only the case $n = 1$ is shown, unless the expressions do not generalize trivially for $n = 2, \ldots, N$.

Table 3.1 gives a summary of the computational per-iteration complexity when computing a rank-R CPD and clearly illustrates independence on the total number of entries I^N.

TABLE 3.1

Computational per-iteration complexity when computing a rank-R CPD of an Nth-order $I \times \cdots \times I$ tensor given in its efficient representation. For Hankelization and Löwnerization, second-order tensorization of K signals is assumed, resulting in a tensor of size $I \times I \times K$. The number of samples is $M = 2I - 1$ for Hankel and $M = 2I$ for Löwner. The number of iterations depends on the optimization algorithm used: when using second-order information, e.g., in GN, one typically needs far fewer iterations than when using only first-order information or ALS.

Structure	Function	Complexity
Dense	Objective	$\mathcal{O}\left(RI^N\right)$
	Gradient	$\mathcal{O}\left(NRI^N\right)$
Polyadic	Parameters	NIF
	Objective	$\mathcal{O}\left(NIFR + NIR^2\right)$
	Gradient	$\mathcal{O}\left(NIFR + NIR^2\right)$
Tucker	Parameters	$\mathcal{O}\left(NIJ + J^N\right)$
	Objective	$\mathcal{O}\left(NIJR + J^N R + NIR^2\right)$
	Gradient	$\mathcal{O}\left(NIJR + J^N NR + NIR^2\right)$
TT	Parameters	$\mathcal{O}\left(NIr^2\right)$
	Objective	$\mathcal{O}\left(NIr^2 R + NIR^2\right)$
	Gradient	$\mathcal{O}\left(NIr^2 R + NIR^2\right)$
Hankel	Parameters	MK
	Objective	$\mathcal{O}\left(M(K + \log_2 M)R + (M+K)R^2\right)$
	Gradient	$\mathcal{O}\left(M(K + \log_2 M)R + (M+K)R^2\right)$
Löwner	Parameters	$M(K+1)$
	Objective	$\mathcal{O}\left(M(K + \log_2 M)R + (M+K)R^2\right)$
	Gradient	$\mathcal{O}\left(M(K + \log_2 M)R + (M+K)R^2\right)$

3.1. Polyadic format.

3.1. Polyadic format. In the polyadic format, a third-order tensor $\mathcal{T}$ is given by a set of factor matrices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ with F columns, i.e., $\mathcal{T} = [\![\mathbf{X}, \mathbf{Y}, \mathbf{Z}]\!]$. The number of variables is therefore $F \sum_n I_n$ instead of $\prod_n I_n$. Efficient implementations in the case $\mathcal{T}$ admits a PD, with F not necessary equal to R, have been presented in [4] (see Kruskal tensor) and are here extended to complex data:

$$\begin{aligned} ||\mathcal{T}||^2 &= \mathbf{1}^{\mathrm{T}} \left(\mathbf{X}^{\mathrm{H}}\mathbf{X} * \mathbf{Y}^{\mathrm{H}}\mathbf{Y} * \mathbf{Z}^{\mathrm{H}}\mathbf{Z}\right)\mathbf{1}, \\ \langle \mathcal{M}_{\mathrm{CPD}}, \mathcal{T}\rangle &= \mathbf{1}^{\mathrm{T}} \left(\mathbf{X}^{\mathrm{H}}\mathbf{A} * \mathbf{Y}^{\mathrm{H}}\mathbf{B} * \mathbf{Z}^{\mathrm{H}}\mathbf{C}\right)\mathbf{1}, \\ \bar{\mathbf{T}}_{(1)}(\mathbf{C} \odot \mathbf{B}) &= \bar{\mathbf{X}}\left(\mathbf{Y}^{\mathrm{H}}\mathbf{B} * \mathbf{Z}^{\mathrm{H}}\mathbf{C}\right). \end{aligned} \tag{3.1}$$

The complexity of these operations is governed by the construction of the inner products which require $\mathcal{O}\left(NIF^2\right)$ and $\mathcal{O}\left(NIFR\right)$ flop. Each MTKRPROD in (3.1) required for $\mathbf{G}_{\mathcal{T}}^{(n)}$ can then be computed using $\mathcal{O}\left(IFR\right)$ flop.

When computing an LMLRA, the following inner product is required:

$$\langle \mathcal{M}_{\mathrm{LMLRA}}, \mathcal{T}\rangle = \mathrm{trace}\left(\mathcal{S} \cdot_1 \mathbf{X}^{\mathrm{H}}\mathbf{U} \cdot_2 \mathbf{Y}^{\mathrm{H}}\mathbf{V} \cdot_3 \mathbf{Z}^{\mathrm{H}}\mathbf{W}\right), \tag{3.2}$$

as well as the MTKRONPROD operation:

$$\bar{\mathbf{T}}_{(1)}(\mathbf{W}\otimes\mathbf{V}) = \bar{\mathbf{X}}\left(\mathbf{W}^{\mathrm{H}}\mathbf{Z}\odot\mathbf{V}^{\mathrm{H}}\mathbf{Y}\right)^{\mathrm{H}}, \tag{3.3}$$

$$\operatorname{vec}\left(\mathcal{T}\right)^{\mathrm{H}}\left(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U}\right) = \operatorname{vec}\left([\![\mathbf{U}^{\mathrm{H}}\mathbf{X},\mathbf{V}^{\mathrm{H}}\mathbf{Y},\mathbf{W}^{\mathrm{H}}\mathbf{Z}]\!]\right)^{\mathrm{H}}. \tag{3.4}$$

The computation of the trace in (3.2) requires only the diagonal entries of the LMLRA and takes $\mathcal{O}\left(NIFR + FR^N\right)$ flop, including the construction of the inner products. To compute the actual gradient $\mathbf{G}^{(1)}_{\mathcal{T}}$, the result of (3.3) is multiplied by $\mathbf{S}^{\mathrm{T}}_{(1)}$, requiring the product $\left(\mathbf{W}^{\mathrm{H}}\mathbf{Z}\odot\mathbf{V}^{\mathrm{H}}\mathbf{Y}\right)^{\mathrm{H}}\mathbf{S}^{\mathrm{T}}_{(1)}$ which is a transposed MTKRPROD involving usually small matrices. (In many practical applications the rank or multilinear rank is small compared to the tensor dimensions.) This product can be computed efficiently using, e.g., [69, 87] and requires $\mathcal{O}\left(NFR^N + NIFR\right)$ flop. The complexity of (3.4) is the same.

3.2. Tucker format. In the Tucker format a third-order tensor $\mathcal{T} = \mathcal{Q}\cdot_1\mathbf{X}\cdot_2\mathbf{Y}\cdot_3\mathbf{Z}$ is defined by a third-order core tensor $\mathcal{Q}\in\mathbb{K}^{J_1\times J_2\times J_3}$ and the factor matrices $\mathbf{X}\in\mathbb{K}^{I\times J_1}$, $\mathbf{Y}\in\mathbb{K}^{I\times J_2}$, $\mathbf{Z}\in\mathbb{K}^{I\times J_3}$. The Tucker format often follows from the computation of an LMLRA or an MLSVD, e.g., as the result of a compression step [7]. For the complexity analysis, we assume $J_n = J$ for $n = 1,\ldots,N$. We assume that the factor matrices have orthonormal columns, which is always possible through normalization.

The norm of $\mathcal{T}$ is computed in $\mathcal{O}\left(J^N\right)$ operations as

$$||\mathcal{T}||^2 = ||\mathcal{Q}||^2.$$

The inner product with $\mathcal{M}_{\mathrm{CPD}}$ and MTKRPROD is similar to (3.2) and (3.3):

$$\begin{aligned}\langle\mathcal{M}_{\mathrm{CPD}},\mathcal{T}\rangle &= \operatorname{trace}\left(\bar{\mathcal{Q}}\cdot_1\mathbf{A}^{\mathrm{T}}\bar{\mathbf{X}}\cdot_2\mathbf{B}^{\mathrm{T}}\bar{\mathbf{Y}}\cdot_3\mathbf{C}^{\mathrm{T}}\bar{\mathbf{Z}}\right)\\ \bar{\mathbf{T}}_{(1)}(\mathbf{C}\odot\mathbf{B}) &= \bar{\mathbf{X}}\bar{\mathbf{Q}}_{(1)}\left(\mathbf{Z}^{\mathrm{H}}\mathbf{C}\odot\mathbf{Y}^{\mathrm{H}}\mathbf{B}\right)\end{aligned}$$

and both require $\mathcal{O}\left(NIJR + J^NR\right)$ flop.

To compute an LMLRA, the following expressions for the inner product and the MTKRONPROD in (2.7) can be used

$$\begin{aligned}\langle\mathcal{M}_{\mathrm{LMLRA}},\mathcal{T}\rangle &= \langle\mathcal{S},\mathcal{Q}\cdot_1\mathbf{U}^{\mathrm{H}}\mathbf{X}\cdot_2\mathbf{V}^{\mathrm{H}}\mathbf{Y}\cdot_3\mathbf{W}^{\mathrm{H}}\mathbf{Z}\rangle,\\ \bar{\mathbf{T}}_{(1)}(\mathbf{W}\otimes\mathbf{V}) &= \bar{\mathbf{X}}\bar{\mathbf{Q}}_{(1)}\left(\mathbf{Z}^{\mathrm{H}}\mathbf{W}\otimes\mathbf{Y}^{\mathrm{H}}\mathbf{V}\right).\end{aligned}$$

Similarly, the MTKRONPROD in (2.6) is computed as

$$\operatorname{vec}(\mathcal{T})^{\mathrm{H}}(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U}) = \operatorname{vec}\left(\mathcal{Q}\cdot_1\mathbf{U}^{\mathrm{H}}\mathbf{X}\cdot_2\mathbf{V}^{\mathrm{H}}\mathbf{Y}\cdot_3\mathbf{W}^{\mathrm{H}}\mathbf{Z}\right)^{\mathrm{H}}.$$

All computations involving the LMLRA require $\mathcal{O}\left(NIJR + J^NR + JR^N\right)$ flop.

3.3. Tensor Train format. An Nth-order tensor $\mathcal{T}\in\mathbb{K}^{I_1\times\cdots\times I_N}$ can be represented by N core matrices or tensors $\mathcal{Q}^{(n)}\in\mathbb{K}^{r_{n-1}\times I_n\times r_n}$ serially linked by $N-1$ indices such that

$$\begin{aligned}t_{i_1,\ldots,i_N} &= \sum_{s_1=1}^{r_1}\cdots\sum_{s_{N-1}=1}^{r_{N-1}} q^{(1)}_{1,i_1,s_1}q^{(2)}_{s_1,i_2,s_2}\cdots q^{(N)}_{s_{N-1},i_N,1}, \quad \text{or}\\ \mathcal{T} &= \operatorname{tt}\left(\mathcal{G}^{(1)},\ldots,\mathcal{G}^{(N)}\right).\end{aligned}$$

The parameters r_n, $n = 0, \ldots, N$, are the compression ranks with $r_0 = r_N = 1$. This formulation is known as matrix product states [59] in computational chemistry and as Tensor Trains (TT) [60] in scientific computing and can be computed using SVDs, cross approximation or optimization [60, 61, 62, 66]. Assuming all dimensions and compression ranks are equal, i.e., $r_n = r$ for $n = 1, \ldots, N-1$, the total number of parameters is $\mathcal{O}\left(NIr^2\right)$.

The squared norm of $\mathcal{T}$ is given by $||\mathcal{T}||^2 = |\mathbf{F}^{(N)}|$, in which the scalar $\mathbf{F}^{(N)}$ is constructed using the recursive formula

$$\begin{aligned}\mathbf{F}^{(0)} &= 1,\\ \mathbf{F}^{(n)} &= \left(\mathbf{Q}^{(n)} \cdot_1 \mathbf{F}^{(n-1)}\right)^{\mathrm{H}}_{(1,2)} \mathbf{Q}^{(n)}_{(1,2)} \qquad n = 1, \ldots, N.\end{aligned}$$

This way $||\mathcal{T}||^2$ can be computed in $\mathcal{O}\left(NIr^3\right)$ flop. Note that this recursive formula is essentially identical to the formula derived in [66].

To compute inner products with $\mathcal{M}_{\mathrm{CPD}}$ and $\mathcal{M}_{\mathrm{LMLRA}}$, we first introduce the auxiliary core tensors $\hat{\mathcal{Q}}^{(n)}$. Let $\mathbf{A}^{(1)} = \mathbf{A}$, $\mathbf{A}^{(2)} = \mathbf{B}$ and $\mathbf{A}^{(3)} = \mathbf{C}$ for $\mathcal{M}_{\mathrm{CPD}}$, and $\mathbf{A}^{(1)} = \mathbf{U}$, $\mathbf{A}^{(2)} = \mathbf{V}$ and $\mathbf{A}^{(3)} = \mathbf{W}$ for $\mathcal{M}_{\mathrm{LMLRA}}$, then $\hat{\mathcal{Q}}^{(n)}$ is computed using $\mathcal{O}\left(Ir^2R\right)$ flop as

$$\hat{\mathcal{Q}}^{(n)} = \bar{\mathcal{Q}}^{(n)} \cdot_2 \mathbf{A}^{(n)^{\mathrm{T}}}.$$

The inner products are then computed as

$$\langle \mathcal{M}_{\mathrm{CPD}}, \mathcal{T} \rangle = \operatorname{trace}\left(\operatorname{tt}(\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})\right), \tag{3.5}$$

$$\langle \mathcal{M}_{\mathrm{LMLRA}}, \mathcal{T} \rangle = \left\langle \operatorname{tt}(\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)}), \bar{\mathcal{S}} \right\rangle. \tag{3.6}$$

As only the diagonal entries of $\operatorname{tt}(\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})$ are required to compute the trace in (3.5), the construction of the $R \times \cdots \times R$ tensor can be avoided. The construction of the diagonal entries in (3.5) and the inner product in (3.6) require $\mathcal{O}\left(NRr^2\right)$ and $\mathcal{O}\left(R^N r^2\right)$ flop, respectively.

To compute an MTKRPROD, the auxiliary cores $\hat{\mathcal{Q}}^{(n)}$, $n = 1, \ldots, N$ are used again. Define $\mathcal{P}^{(1)} = \operatorname{tt}(\bar{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})$ and $\mathbf{p}_r^{(1)}$ as the mode-1 vectors $\mathcal{P}^{(1)}(:, r, r)$, $r = 1, \ldots, R$, then

$$\bar{\mathbf{T}}_{(1)}(\mathbf{C} \odot \mathbf{B}) = \begin{bmatrix}\mathbf{p}_1^{(1)} & \mathbf{p}_2^{(1)} & \cdots & \mathbf{p}_R^{(1)}\end{bmatrix}.$$

The complexity of computing one MTKRPROD is therefore $\mathcal{O}\left(Ir^2R\right)$, assuming auxiliary cores have been constructed.

Similarly, the MTKRONPROD can be formed efficiently as

$$\bar{\mathbf{T}}_{(1)}(\mathbf{W} \otimes \mathbf{V}) = \left(\operatorname{tt}(\bar{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})\right)_{(1)}.$$

To compute the gradient terms $\mathbf{G}_{\mathcal{T}}^{(n)}$ in (2.7) the unfolding $\left(\operatorname{tt}(\bar{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})\right)_{(1)}$ can be constructed and multiplied with $\mathbf{S}_{(1)}^{\mathrm{T}}$, which requires $\mathcal{O}\left(IR^N + IR^{N-1}r^2\right)$ flop. However, by exploiting the TT structure, the product can be computed directly in $\mathcal{O}\left(rR^N + r^2R^{N-1} + Ir^2R\right)$ flop. The precise implementation is out of the scope of this paper. Finally, to compute $\mathcal{G}_{\mathcal{T}}^{(0)}$ in $\mathcal{O}\left(r^2R^N\right)$ flop, we can use

$$\operatorname{vec}(\mathcal{T})^{\mathrm{H}}(\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U}) = \operatorname{vec}\left(\operatorname{tt}(\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{Q}}^{(2)}, \hat{\mathcal{Q}}^{(3)})\right)^{\mathrm{T}}.$$

3.4. Implicit Hankelization. Hankelization of a (possibly complex) signal vector $\mathbf{m} \in \mathbb{K}^M$ of length $M = \sum_{n=1}^{N} I_n - N + 1$ yields an $I_1 \times I_2$ matrix $\mathbf{T}$ or $I_1 \times \cdots \times I_N$ tensor $\mathcal{T}$ with constant anti-diagonals ($t_{i_1 i_2} = m_{i_1+i_2-1}$) or constant anti-diagonal hyperplanes in the tensor ($t_{i_1 \ldots i_N} = m_{i_1+\cdots+i_N-N+1}$) for $N = 2$ and $N > 2$, respectively; see [25,68] for a signal processing background. In blind source separation (BSS), a set of K vectors $\mathbf{m}_k \in \mathbb{K}^M$ is often given [16,25]. Each vector $\mathbf{m}_k$ can be Hankelized separately and the resulting matrices or tensors can be concatenated along the $(N+1)$th mode. To simplify the expressions[1], we only consider second-order Hankelization and assume $I_1 = I_2 = I$. Let us stack the vectors $\mathbf{m}_k$ as the columns of $\mathbf{M}$. Each entry of the $I \times I \times K$ tensor $\mathcal{T}$ is given by $t_{ijk} = m_{i+j-1,k}$. As multiplying a Hankel matrix by a vector corresponds to a convolution, fast Fourier transforms (FFT) can be used to speed up computations [3,28]. Let $\mathfrak{F}$ denote the M-point FFT and $\mathfrak{F}^{-1}$ the inverse M-point FFT. Only the last I rows of the inverse M-point FFT are retained when the operator $\underline{\mathfrak{F}}^{-1}$ is used, i.e., $\underline{\mathfrak{F}}^{-1}\mathbf{x} = \begin{bmatrix}\mathbf{0}_{I\times(M-I)} & \mathbf{I}_I\end{bmatrix}(\mathfrak{F}^{-1}\mathbf{x})$ for $\mathbf{x} \in \mathbb{C}^M$.

The squared norm can be computed in $\mathcal{O}(MK)$ flop as

$$||\mathcal{T}||^2 = \mathbf{w}^{\mathrm{T}}(\mathbf{M} * \overline{\mathbf{M}})\mathbf{1}_K,$$

in which $\mathbf{w}$ is a vector that contains the number of occurrences of each entry of $\mathbf{M}$ in $\mathcal{T}$ and can be computed as $\mathbf{w} = \text{convolution}(\mathbf{1}_I, \mathbf{1}_I)$. The inner products can be computed as

$$\begin{aligned}\langle \mathcal{M}_{\text{CPD}}, \mathcal{T}\rangle &= \left\langle \mathfrak{F}^{-1}(\mathfrak{F}\mathbf{A} * \mathfrak{F}\mathbf{B})\mathbf{C}^{\mathrm{T}}, \mathbf{M}\right\rangle, \\ \langle \mathcal{M}_{\text{LMLRA}}, \mathcal{T}\rangle &= \left\langle \mathfrak{F}^{-1}\left((\mathfrak{F}\mathbf{V} \odot^{\mathrm{T}} \mathfrak{F}\mathbf{U})\mathbf{S}_{(1,2)}\mathbf{W}^{\mathrm{T}}\right), \mathbf{M}\right\rangle.\end{aligned}$$

The computational cost is $\mathcal{O}(RM(K + \log_2 M))$ and $\mathcal{O}\left((R+K)M\log_2 M + MR^3\right)$ flop for the inner product with $\mathcal{M}_{\text{CPD}}$ and $\mathcal{M}_{\text{LMLRA}}$, respectively.

To compute the MTKRPROD, the Hankel structure can be exploited as

$$\begin{aligned}\bar{\mathbf{T}}_{(1)}(\mathbf{C} \odot \mathbf{B}) &= \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\bar{\mathbf{M}}\mathbf{C} * \mathfrak{F}\mathbf{B}^f\right), \\ \bar{\mathbf{T}}_{(2)}(\mathbf{C} \odot \mathbf{A}) &= \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\bar{\mathbf{M}}\mathbf{C} * \mathfrak{F}\mathbf{A}^f\right), \\ \bar{\mathbf{T}}_{(3)}(\mathbf{B} \odot \mathbf{A}) &= \mathbf{M}^{\mathrm{H}}\mathfrak{F}^{-1}\left(\mathfrak{F}\mathbf{A} * \mathfrak{F}\mathbf{B}\right),\end{aligned}$$

requiring $\mathcal{O}(RM(K + \log_2 M))$ flop each. ($\mathbf{A}^f$ and $\mathbf{B}^f$ are the flipped versions of $\mathbf{A}$ and $\mathbf{B}$; see subsection 1.3.)

The MTKRONPROD in $\mathbf{G}_{\mathcal{T}}^{(n)}$ in (2.7) can be computed in $\mathcal{O}\left(R^2 M\log_2 M + RMK\right)$ flop as

$$\begin{aligned}\bar{\mathbf{T}}_{(1)}(\mathbf{W} \otimes \mathbf{V}) &= \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\bar{\mathbf{M}}\mathbf{W} \odot^{\mathrm{T}} \mathfrak{F}\mathbf{V}^f\right), \\ \bar{\mathbf{T}}_{(2)}(\mathbf{W} \otimes \mathbf{U}) &= \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\bar{\mathbf{M}}\mathbf{W} \odot^{\mathrm{T}} \mathfrak{F}\mathbf{U}^f\right), \\ \bar{\mathbf{T}}_{(3)}(\mathbf{V} \otimes \mathbf{U}) &= \mathbf{M}^{\mathrm{H}}\mathfrak{F}^{-1}\left(\mathfrak{F}\mathbf{V} \odot^{\mathrm{T}} \mathfrak{F}\mathbf{U}\right).\end{aligned}$$

The multiplication with $\mathbf{S}_{(n)}^{\mathrm{T}}$ in (2.7) additionally takes $\mathcal{O}\left(IR^3\right)$ flop for $n = 1, 2$ and $\mathcal{O}\left(KR^3\right)$ flop for $n = 3$. However, by first computing the multiplication of the row-wise Khatri–Rao product with $\mathbf{S}_{(n)}^{\mathrm{T}}$, i.e., before performing the inverse FFT, the total cost can be reduced to $\mathcal{O}\left(RM(K + \log_2 M) + MR^3\right)$. Finally, the MTKRONPROD in all modes can be computed in $\mathcal{O}\left(R^2 M\log_2 M + R^3 M + RMK\right)$ flop using

$$\text{vec}(\mathcal{T})^{\mathrm{H}}(\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U}) = \text{vec}\left((\mathfrak{F}^{-1}(\mathfrak{F}\mathbf{V} \odot^{\mathrm{T}} \mathfrak{F}\mathbf{U}))^{\mathrm{T}}\bar{\mathbf{M}}\mathbf{W}\right)^{\mathrm{T}}.$$

[1]More general implementations are available in Tensorlab 3.0 [94].

3.5. Implicit Löwnerization. Löwner matrices and tensors have attractive properties for applications involving rational functions [25, 27]. Given a function $h : \mathbb{K} \to \mathbb{K}$ evaluated at M points $t_i \in T = \{t_1, \ldots, t_M\}$, we partition T in two disjoint point sets $X = \{x_1, \ldots, x_I\}$ and $Y = \{y_1, \ldots, y_J\}$ such that $T = X \cup Y$ and $M = I + J$. The entries in a Löwner matrix $\mathbf{L} \in \mathbb{K}^{I \times J}$ are then given by

$$l_{ij} = \frac{h(x_i) - h(y_j)}{x_i - y_j}, \quad i = 1, \ldots, I, \text{ and } j = 1, \ldots, J.$$

While higher-order generalizations of the Löwner transformation exist [25, 26], we focus on third-order tensors in which each kth frontal slice is a Löwner matrix constructed from a function h_k evaluated in the points in X and Y. Let $\mathbf{P} \in \mathbb{K}^{I \times K}$ and $\mathbf{Q} \in \mathbb{K}^{J \times K}$ contain sampled function values at the points in X and Y, respectively, i.e.,

$$\begin{aligned} p_{ik} &= h_k(x_i), \quad i = 1, \ldots, I, \text{ and } k = 1, \ldots, K, \\ q_{jk} &= h_k(y_j), \quad j = 1, \ldots, J, \text{ and } k = 1, \ldots, K. \end{aligned}$$

The tensor $\mathcal{T} \in \mathbb{K}^{I \times J \times K}$ is then fully determined by $\mathbf{P}$, $\mathbf{Q}$, X and Y. To simplify complexity expressions, we take $I = J$. Each frontal slice $\mathbf{T}_k$, $k = 1, \ldots, K$ can be written as

$$\begin{aligned} \mathbf{T}_k &= \mathrm{Diag}(\mathbf{p}_k)\mathbf{M} - \mathbf{M}\mathrm{Diag}(\mathbf{q}_k) \\ &= \mathbf{p}_k \odot^{\mathrm{T}} \mathbf{M} - \mathbf{M}(\mathbf{q}_k \odot^{\mathrm{T}} \mathbf{I}_J), \end{aligned}$$

in which $\mathbf{M}$ is a Cauchy matrix with

$$m_{ij} = \frac{1}{x_i - y_j}, \quad 1 \leq i, j \leq I.$$

By assuming that all points in X and Y are equidistant, $\mathbf{M}$ is also a Toeplitz matrix. As multiplying a Toeplitz matrix with a vector $\mathbf{x}$ can be seen as a convolution of the generating vector $\mathbf{v} = \left[m_{1,I}; m_{1,I-1}; \ldots; m_{1,1}; m_{2,1}; \ldots; m_{I,1}\right] \in \mathbb{K}^{2I-1}$ with $\mathbf{x}$, it is not necessary to construct $\mathbf{M}$ and the multiplication can be performed using FFTs. Let $\mathfrak{F}$ be the $(2I-1)$-point FFT with zero padding for shorter vectors, $\mathfrak{F}^{-1}$ the inverse $(2I-1)$-point FFT and define $\underline{\mathfrak{F}}^{-1}$ to be the last I rows of the result of the inverse FFT. For a matrix $\mathbf{X} \in \mathbb{K}^{I \times K}$, $\mathbf{MX}$ and $\mathbf{M}^{\mathrm{T}}\mathbf{X}$ can then be computed as

$$\mathbf{MX} = \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\mathbf{v}\mathbf{1}_K^{\mathrm{T}} * \mathfrak{F}\mathbf{X}\right), \tag{3.7}$$

$$\mathbf{M}^{\mathrm{T}}\mathbf{X} = \underline{\mathfrak{F}}^{-1}\left(\mathfrak{F}\mathbf{v}^{f}\mathbf{1}_K^{\mathrm{T}} * \mathfrak{F}\mathbf{X}\right). \tag{3.8}$$

If $\mathfrak{F}\mathbf{v}$ is precomputed, the cost of this multiplication is $\mathcal{O}\left(2KI + 4KI\log_2 2I\right)$ flop.

We now define the different operations for Löwner tensors. As the unfoldings $\mathbf{T}_{(n)}$, $n = 1, 2, 3$, are given by

$$\begin{aligned} \mathbf{T}_{(1)} &= \mathbf{P} \odot^{\mathrm{T}} \mathbf{M} - \mathbf{M}\left(\mathbf{Q} \odot^{\mathrm{T}} \mathbf{I}_J\right), \\ \mathbf{T}_{(2)} &= \mathbf{M}^{\mathrm{T}}\left(\mathbf{P} \odot^{\mathrm{T}} \mathbf{I}_I\right) - \left(\mathbf{Q} \odot^{\mathrm{T}} \mathbf{M}^{\mathrm{T}}\right), \\ \mathbf{T}_{(3)} &= \mathbf{P}^{\mathrm{T}}\left(\mathbf{M} \odot^{\mathrm{T}} \mathbf{I}_I\right) - \mathbf{Q}^{\mathrm{T}}\left(\mathbf{I}_J \odot^{\mathrm{T}} \mathbf{M}^{\mathrm{T}}\right), \end{aligned}$$

the expressions for the MTKRPROD and MTKRONPROD can be derived easily using multilinear algebra identities, e.g., $(\mathbf{A} \odot^{\mathrm{T}} \mathbf{B})(\mathbf{C} \odot \mathbf{D}) = \mathbf{AC} * \mathbf{BD}$. The MTKRPROD can be computed efficiently as

$$\begin{aligned}\bar{\mathbf{T}}_{(1)}\left(\mathbf{C}\odot\mathbf{B}\right)&=\bar{\mathbf{P}}\mathbf{C}*\bar{\mathbf{M}}\mathbf{B}-\bar{\mathbf{M}}\left(\bar{\mathbf{Q}}\mathbf{C}*\mathbf{B}\right),\\ \bar{\mathbf{T}}_{(2)}\left(\mathbf{C}\odot\mathbf{A}\right)&=\mathbf{M}^{\mathrm{H}}\left(\bar{\mathbf{P}}\mathbf{C}*\mathbf{A}\right)-\bar{\mathbf{Q}}\mathbf{C}*\mathbf{M}^{\mathrm{H}}\mathbf{A},\\ \bar{\mathbf{T}}_{(3)}\left(\mathbf{B}\odot\mathbf{A}\right)&=\mathbf{P}^{\mathrm{H}}\left(\bar{\mathbf{M}}\mathbf{B}*\mathbf{A}\right)-\mathbf{Q}^{\mathrm{H}}\left(\mathbf{B}*\mathbf{M}^{\mathrm{H}}\mathbf{A}\right).\end{aligned}$$

As each MTKRPROD requires two FFT-based multiplications with $\mathbf{M}$ and two regular matrix-matrix multiplications with $\mathbf{P}$ and $\mathbf{Q}$, the complexity is $\mathcal{O}\left(RI(K+\log_2 I)\right)$. Similarly, the MTKRONPROD in (2.7) can be computed as

$$\begin{aligned}\bar{\mathbf{T}}_{(1)}\left(\mathbf{W}\otimes\mathbf{V}\right)&=\bar{\mathbf{P}}\mathbf{W}\odot^{\mathrm{T}}\bar{\mathbf{M}}\mathbf{V}-\bar{\mathbf{M}}\left(\bar{\mathbf{Q}}\mathbf{W}\odot^{\mathrm{T}}\mathbf{V}\right),\\ \bar{\mathbf{T}}_{(2)}\left(\mathbf{W}\otimes\mathbf{U}\right)&=\mathbf{M}^{\mathrm{H}}\left(\bar{\mathbf{P}}\mathbf{W}\odot^{\mathrm{T}}\mathbf{U}\right)-\bar{\mathbf{Q}}\mathbf{W}\odot^{\mathrm{T}}\mathbf{M}^{\mathrm{H}}\mathbf{U},\\ \bar{\mathbf{T}}_{(3)}\left(\mathbf{V}\otimes\mathbf{U}\right)&=\mathbf{P}^{\mathrm{H}}\left(\bar{\mathbf{M}}\mathbf{V}\odot^{\mathrm{T}}\mathbf{U}\right)-\mathbf{Q}^{\mathrm{H}}\left(\mathbf{V}\odot^{\mathrm{T}}\mathbf{M}^{\mathrm{H}}\mathbf{U}\right),\end{aligned}$$

in $\mathcal{O}\left(R^2I+R^2I\log_2 I+RKI\right)$ flop for $n=1,2$ and $\mathcal{O}\left(RI\log_2 I+R^2IK\right)$ flop for $n=3$. This can be reduced further to $\mathcal{O}\left(RI\log_2 I+RKI\right)$ flop for $n=1,2$, by exploiting the row-wise Khatri–Rao products when computing the subsequent multiplication with $\mathbf{S}_{(n)}^{\mathrm{T}}$, which costs $\mathcal{O}\left(IR^3\right)$. A similar improvement can be made for $n=3$. The MTKRONPROD in (2.6) is computed as

$$\operatorname{vec}\left(\mathcal{T}\right)^{\mathrm{H}}\left(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U}\right)=\operatorname{vec}\left(\mathbf{U}^{\mathrm{T}}\left(\bar{\mathbf{P}}\mathbf{W}\odot^{\mathrm{T}}\bar{\mathbf{M}}\mathbf{V}\right)-\left(\mathbf{M}^{\mathrm{H}}\mathbf{U}\right)^{\mathrm{T}}\left(\bar{\mathbf{Q}}\mathbf{W}\odot^{\mathrm{T}}\mathbf{V}\right)\right)^{\mathrm{T}}$$

and requires $\mathcal{O}\left(RI\log_2 I+R^3I\right)$ flop.

The squared Frobenius norm is computed as

$$\begin{aligned}||\mathcal{T}||^2=&\ \mathbf{1}_I^{\mathrm{T}}(\mathbf{M}*\overline{\mathbf{M}})^{\mathrm{T}}\left((\mathbf{P}*\overline{\mathbf{P}})\mathbf{1}_K\right),\\ &+\mathbf{1}_I^{\mathrm{T}}(\mathbf{M}*\overline{\mathbf{M}})\left((\mathbf{Q}*\overline{\mathbf{Q}})\mathbf{1}_K\right),\\ &-2\mathrm{Re}\left(\mathbf{1}_K^{\mathrm{T}}((\mathbf{M}*\overline{\mathbf{M}})^{\mathrm{T}}\mathbf{P}*\overline{\mathbf{Q}})\mathbf{1}_K\right).\end{aligned}$$

As only $K+2$ FFTs are required, the total computational cost is $\mathcal{O}\left(4IK\log_2 2I\right)$ flop. To compute the inner products with $\mathcal{M}_{\mathrm{CPD}}$ and $\mathcal{M}_{\mathrm{LMLRA}}$, the results from MTKRPROD and MTKRONPROD are reused:

$$\begin{aligned}\left\langle\mathcal{M}_{\mathrm{CPD}},\mathcal{T}\right\rangle&=\left\langle\bar{\mathbf{T}}_{(3)}\left(\mathbf{B}\odot\mathbf{A}\right),\bar{\mathbf{C}}\right\rangle,\\ \left\langle\mathcal{M}_{\mathrm{LMLRA}},\mathcal{T}\right\rangle&=\left\langle\mathbf{X}^{\mathrm{T}}\bar{\mathbf{T}}_{(1)}\left(\mathbf{W}\otimes\mathbf{V}\right),\bar{\mathbf{S}}_{(1)}\right\rangle.\end{aligned}$$

Apart from the cost of the MTKRPROD and the MTKRONPROD, an additional cost of $\mathcal{O}\left(KR\right)$ and $\mathcal{O}\left(R^N\right)$ flop is incurred for $\langle\mathcal{M}_{\mathrm{CPD}},\mathcal{T}\rangle$ and $\langle\mathcal{M}_{\mathrm{LMLRA}},\mathcal{T}\rangle$, respectively.

In the case the points sets X or Y do not contain equidistant points, $\mathbf{M}$ no longer admits a Toeplitz structure. By exploiting the low displacement rank of Cauchy matrices, it is again not necessary to explicitly construct the tensor. The computational complexity of the multiplications with $\mathbf{M}$ in (3.7) and (3.8) is slightly increased to $\mathcal{O}\left(4KI\log_2^2 2I\right)$ flop [29,30].

4. Experiments.

The following experiments illustrate the scalability of constrained CPD, LL1 and BTD algorithms that exploit efficient representations of tensors and the effect on the accuracy of the results. The latter is discussed in subsection 4.1 for ill-conditioned problems and in subsection 4.3 for Hankelized signals. The former is illustrated by computing the nonnegative CPD of a compressed tensor in subsection 4.2 and for the unconstrained LL1 decomposition and the constrained BTD in subsection 4.3. The CPD error E_{CPD} is defined as

$$E_{\text{CPD}} = \max_n \frac{\left|\left|\mathbf{A}^{(n)} - \hat{\mathbf{A}}^{(n)}\right|\right|}{\left|\left|\mathbf{A}^{(n)}\right|\right|},$$

in which $\mathbf{A}^{(n)}$ ($\hat{\mathbf{A}}^{(n)}$) are the exact (estimated) factor matrices, $n = 1, \ldots, N$, and scaling and permutation indeterminacies are assumed to be resolved using the congruence maximization heuristic; see `cpderr` [94]. All timing results are total computation times for a complete run of the optimization algorithm, in contrast to the derived per-iteration complexities in section 3. Tensorlab 3.0 [94] is used for all experiments in combination with Matlab 2016b running on a dual socket 20 core Intel Xeon E5-2660 v2 machine with 128 GiB of RAM running CentOS 7.

4.1. Accuracy and conditioning. In this first experiment, we study the accuracy of recovered factor matrices for the full tensor and its efficient representation. More specifically, the CPD of a tensor with an exact rank-R structure is computed while varying the relative condition number κ, which is defined as in [86]. Concretely, a rank-5 tensor of size $25 \times 25 \times 25$ is constructed using random factor matrices $\mathbf{A}^{(n)}$, $n = 1, 2, 3$. Each random factor vector $\mathbf{a}_r^{(n)}$ has norm one and a fixed angle α w.r.t. the other factor vectors in the same factor matrix, i.e., for $n = 1, 2, 3$, $\left|\left|\mathbf{a}_r^{(n)}\right|\right| = 1$ and $\cos\alpha = {\mathbf{a}_r^{(n)}}^{\mathrm{T}} \mathbf{a}_s^{(n)}$, for $r, s = 1, \ldots, R$, $r \neq s$. As α decreases, the rank-1 terms become more collinear and the condition number κ increases. A rank-5 CPD is computed from the full tensor and from the structured tensor in the polyadic format using inexact GN, implemented in `cpd_nls`, starting from a perturbed exact solution, i.e., the initial guesses for factor matrices $\mathbf{U}^{(n)}$ are constructed as $\mathbf{U}^{(n)} = \mathbf{A}^{(n)} + \sigma^{(n)}\mathbf{N}^{(n)}$ with $\mathbf{N}^{(n)}$ a random matrix with entries drawn from the normal distribution and $\sigma^{(n)} = 0.1 \cdot \left|\left|\mathbf{A}^{(n)}\right|\right| / \left|\left|\mathbf{N}^{(n)}\right|\right|$. Figure 4.1 shows the error E_{CPD} on the recovered factor matrices for $\alpha = \pi/2, \ldots, \pi/180$ (using a logarithmic scale). As expected, the error increases when the condition number κ worsens. For well-conditioned problems, E_{CPD} is in the order of the machine precision $\epsilon \approx 10^{-16}$ if the full tensor is used, while E_{CPD} is higher for the structured tensor. This can be explained by the fact that changes smaller than $\sqrt{\epsilon} \approx 10^{-8}$ cannot be distinguished when using the structured tensor, as the computation of the objective function (2.3) requires the difference of squared, almost equal numbers. For very ill-conditioned problems, E_{CPD} may be undesirably high when using the structured tensor. If the tensor admits an exact decomposition *and* if this exact decomposition is required up to machine precision, using the full tensor may be necessary. However, the structured tensor can still be used as an initialization to reduce the overall computational cost. Note that for very large-scale tensors, the explicit construction of the tensor may not be feasible anyway, and that in many applications, the condition number is limited as indicated by the reference points in Figure 4.1.

4.2. Compression for nonnegative CPD. The following experiments illustrate how compression can be used to reduce the complexity of constrained tensor decompositions. Here, we focus on the widely used nonnegative CPD, which is sped up by first computing a truncated MLSVD or a TT approximation. Two approaches to enforce the constraints are used: a projected Gauss–Newton approach generating updates such that the variables are always positive using active sets [46], and using parameter-based constraints, i.e., each entry is the square of an underlying

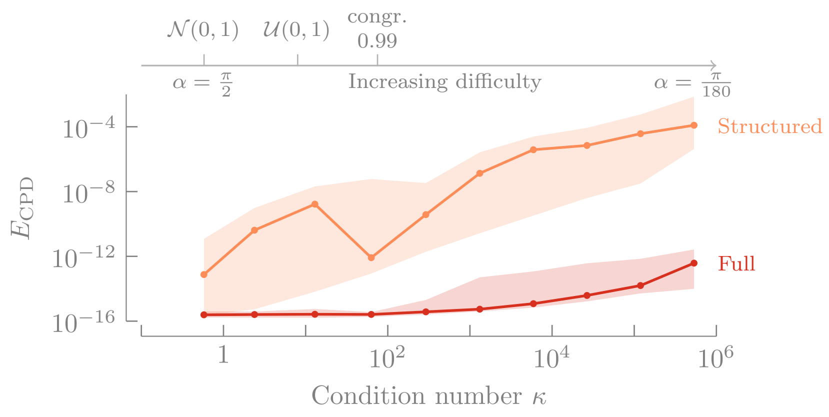

FIGURE 4.1. *By reducing the angle α between the factor vectors, the condition of the CPD worsens, resulting in a higher error E_{CPD}. For very ill-conditioned problems, using the structured tensor approach may result in an undesirably large error and a few additional iterations using the full tensor may be required to improve the accuracy. Above the graph, three typical test setups are indicated as references: factor entries drawn from a normal distribution $\mathcal{N}(0,1)$, from a uniform distribution $\mathcal{U}(0,1)$ or a triple bottleneck setup such that the congruence between all factor vectors is 0.99 in each mode. The latter is typically considered very difficult. Results are medians over 100 experiments. The shaded areas give the minimum and maximum errors over all experiments.*

variable [71]. The former approach is implemented using a boundary constrained Gauss–Newton algorithm `nlsb_gndl`; the latter is implemented using the structured data fusion framework [83] (`sdf_nls` and `struct_nonneg`).

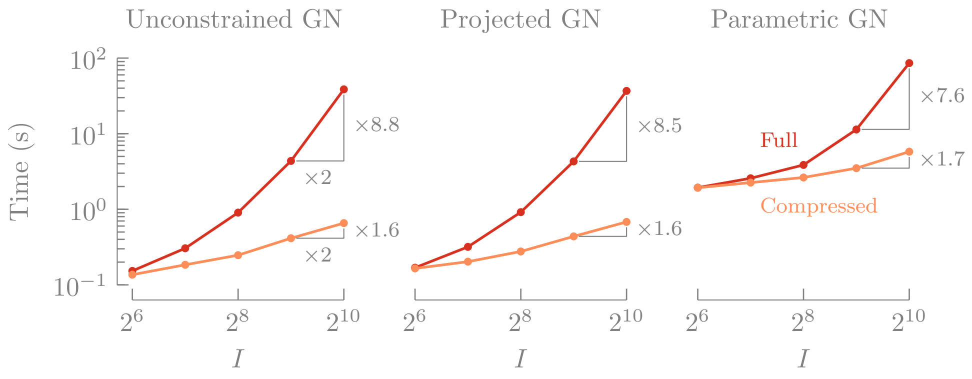

FIGURE 4.2. *When using a compressed tensor instead of the full tensor, the computational cost scales linearly in the tensor dimensions instead of cubically for a rank-10 nonnegative tensor of size $I \times I \times I$. MLSVD compression with a core size of $10 \times 10 \times 10$ is used. The increase for the structured tensor is actually less than linear ($\times 1.6 - \times 1.7$ instead of $\times 2$), which is caused by an improved multicore usage. The time required to compute the MLSVD increases cubically from* $20\,\text{ms}$ *for $I = 2^6$ to* $8\,\text{s}$ *for $I = 2^{10}$ and is not included. Medians over 100 experiments for each parameter are reported. All algorithms stop if the relative change in function value is smaller than* 10^{-7}. *The parametric GN approach has a larger overhead and is not preconditioned, explaining the higher computation time.*

For the first experiment, let $\mathcal{T}$ be a nonnegative rank-10 tensor of size $I \times I \times I$ constructed using random factor matrices $\mathbf{A}^{(n)}$ with entries drawn from a uniform distribution $\mathcal{U}(0,1)$. Gaussian i.i.d. noise is added such that the signal-to-noise ratio

(SNR) is 20 dB. Using a randomized MLSVD (`mlsvd_rsi` [91]) the tensor is compressed such that the core $\mathcal{G}$ has size $10 \times 10 \times 10$, i.e., $\mathcal{T} \approx [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}]\!]$. The core tensor $\mathcal{G}$ and the factors $\mathbf{U}^{(n)}$ are then used as structured format to compute a rank-10 CPD. Random initial factor matrices are drawn from the same distribution as $\mathbf{A}^{(n)}$. Figure 4.2 shows the time required to compute an unconstrained CPD (hence relying on CPD uniqueness), a constrained CPD using projected GN (`cpd_nls` with `nlsb_gndl` solver) and a constrained CPD using parametrization (`sdf_nls`). In the three cases, the computation time for the original tensor $\mathcal{T}$ rises cubically in the dimension I, for $I = 2^6, 2^7, \ldots, 2^{10}$, while the time rises linearly using the structured approach. Hence, existing CPD algorithms can be scaled to handle large problems, provided that the MLSVD approximation can be computed and the rank is modest. Note that the compression time, which also rises cubically in I from 20 ms for $I = 2^6$ to 8 s for $I = 2^{10}$, is not included. The total time for compression and decomposition is still (far) less than the time required for the decomposition of the full tensor, though. Moreover, in many signal processing and data analysis applications, the compression cost can be amortized by reusing the representation for multiple initializations, ranks or experiments. The compression cost can be reduced further by resorting to parallelism and/or cross approximation techniques; see, e.g., [9, 56, 63].

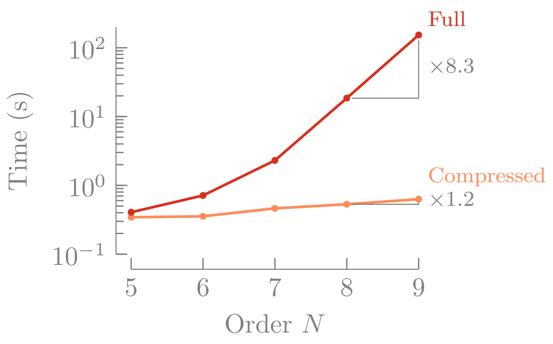

FIGURE 4.3. *Using a TT approximation instead of the full* $10 \times 10 \times \cdots \times 10$ *tensor removes the exponential dependence on the order* N *when computing a rank-5 nonnegative CPD: from* $N = 8$ *to* $N = 9$ *the time increases with a factor* $1.2 \approx 9/8$ *using the TT approximation, while the time increase is* $8.3 \approx 10$ *using the full tensor. The median TT compresssion time is not included and increases exponentially from* 20 ms *for* $N = 5$ *to* 114 s *for* $N = 9$. *The results are medians over 100 experiments.*

In a second experiment, the complexity in function of the order N of the tensor is investigated. We compare the required time to compute a nonnegative CPD of an Nth order rank-5 tensor with dimensions $10 \times \cdots \times 10$ using the full tensor and its TT approximation. The data is generated by constructing N random factor matrices with entries drawn from the uniform distribution $\mathcal{U}(0, 1)$. Random Gaussian i.i.d. noise is added to the generated tensor such that the SNR is 20 dB. The core tensors $\mathcal{G}^{(n)}$ corresponding to the TT approximation are computed using `tt_tensor` from the TT Toolbox [62], which we slightly adapted such that all compression ranks $r_n = 5$, $n = 1, \ldots, N - 1$. Random factor matrices using the same distribution are used as initialization. Nonnegativity is enforced using the projected GN approach (`cpd_nls` with `nlsb_gndl` solver). The timing results in Figure 4.3 clearly show that using the TT approximation avoids the curse of dimensionality as the time increases linearly in N in contrast to the time required for the decomposition of the full tensor. Note that we expect the time to increase with a factor 10 if the order increases by one, but the actual time increase is lower as seen in Figure 4.3. This is due to the fact that the decomposition problem becomes easier as relatively more data is available per variable. This is also reflected in the accuracy: for $N = 5$ the median accuracy $E_{\text{CPD}} = 4.7 \cdot 10^{-3}$, while $E_{\text{CPD}} = 3.8 \cdot 10^{-5}$ for $N = 9$.

4.3. Signal separation through Hankelization. We use the blind source separation (BSS) setup

$$\mathbf{X} = \mathbf{S}\mathbf{M}^{\mathrm{T}} + \mathbf{N} \tag{4.1}$$

to recover the unknown sources $\mathbf{S}$ and unknown mixing matrix $\mathbf{M}$ from the given signals $\mathbf{X}$. The additive Gaussian i.i.d. noise $\mathbf{N} \in \mathbb{R}^{M \times K}$ in the experiments is scaled such that a given SNR is attained. In general, $\mathbf{M}$ and $\mathbf{S}$ cannot be recovered uniquely. To achieve uniqueness, the classical independent component analysis assumes statistically independent sources, while deterministic BSS techniques assume that the sources are, e.g., exponential polynomials or rational functions. In the case when source signals are sums of exponential polynomials, these signals can be recovered using Hankelization [19]. Hankelization along the columns of $\mathbf{X}$ leads to a third-order tensor $\mathcal{T}$ of size $\left\lfloor \frac{M+1}{2} \right\rfloor \times \left\lceil \frac{M+1}{2} \right\rceil \times K$ in which $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor and ceil operators, respectively. Hence, if the number of samples M doubles, the number of entries in $\mathcal{T}$ quadruples. Given this fast increase, we show that exploiting the structure is crucial to apply deterministic BSS methods to problems of realistic sizes. If each of the R source signals is a sum of Q_r exponential polynomials of degree d_{qr}, $q = 1, \ldots, Q_r$, $\mathcal{T}$ can be decomposed into a sum of R multilinear rank-$(L_r, L_r, 1)$ terms with $L_r = \sum_{q=1}^{Q_r} (d_{qr} + 1)$ [19], i.e.,

$$\mathcal{T} = \sum_{r=1}^{R} (\mathbf{A}_r \mathbf{B}_r^{\mathrm{T}}) \otimes \mathbf{c}_r$$

in which $\mathbf{c}_r$ estimates the rth mixing vector $\mathbf{M}(:, r)$. The source signals can be estimated up to scaling and permutation by dehankelizing $\mathbf{A}_r \mathbf{B}_r^{\mathrm{T}}$, e.g., by averaging over the anti-diagonals. The example from [19] is slightly adapted in this experiment: two sources are mixed using

$$\mathbf{M} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix},$$

i.e., $K = R = 2$. The sources are given by

$$\begin{aligned} s_1(t) &= 0.5 \sin(6\pi t) \\ s_2(t) &= (4t^2 - 2.8t) \exp(-t), \end{aligned}$$

hence $L_1 = 2$ and $L_2 = 3$. Estimating R and L_r is out of the scope of this paper; see, e.g., [19]. The true R and L_r are therefore used. M equidistant samples are taken between 0 s and 1 s.

In the first experiment, the SNR is varied for a fixed number of $M = 501$ samples and the relative error E on the mixing matrix is compared when using the full Hankelized tensor or the implicitly Hankelized tensor, i.e., in the structured format. Hankelization is performed using `hankelize`, which returns both the explicit and implicit tensorization. The resulting tensors of size $251 \times 251 \times 2$ are decomposed using `ll1_nls` starting from a random initialization. From the factor matrices, the signals are recovered using `dehankelize`, without constructing the full tensor. As shown in Figure 4.4, when increasing the SNR from 0 dB to 300 dB, the error decreases at the same rate for the explicit and the implicit tensorization until the SNR is 180 dB: while the error continues to decrease for the explicit tensorization, the error stagnates at $E \approx 10^{-10}$ for the implicit tensorization. This loss of accuracy is explained in subsection 4.1 and only occurs for a very high SNR and in the case the sources are exact sums of exponential polynomials. As illustrated in the following experiments, a trade-off between computational cost and accuracy can be made. In the case of a high SNR, the solution using the implicit tensorization can be used to initialize the algorithm with the full tensor.

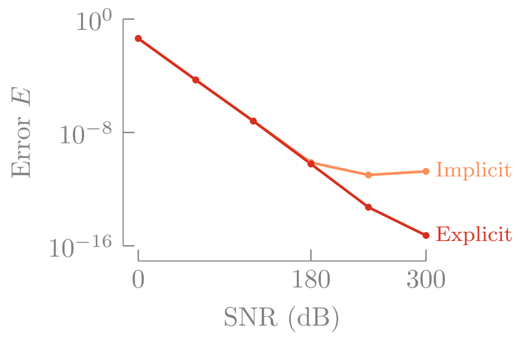

FIGURE 4.4. *For low and medium SNR, the errors on the mixing matrix for the explicit and implicit Hankelization approaches are equal. While the error for the implicit tensorization stagnates for SNR larger than* 180 dB*, the error continues to decrease when using the explicit tensorization. The errors are medians over 100 experiments, each using a best-out-of-five initializations strategy.*

In the second experiment, the scaling behavior in function of the number of samples is illustrated for $M = 501, 5\,001, 50\,001$ and $500\,001$ points. The resulting Hankelized tensors require 953 KiB, 95.3 MiB, 9.53 GiB and 953 GiB of memory, respectively, if formed explicitly. The SNR is fixed at 20 dB. We compute an unconstrained LL1 decomposition (`ll1_nls`) and a constrained BTD, which models the tensor as

$$\mathcal{T} = \sum_{r=1}^{R} \left(\mathbf{V}^{(r)} \mathbf{G}^{(r)} \mathbf{V}^{(r)^{\mathrm{T}}} \right) \otimes \mathbf{c}_r,$$

in which $\mathbf{V}^{(r)}$ are confluent Vandermonde matrices and $\mathbf{G}^{(r)}$ are upper anti-triangular matrices; see [19] for details. By imposing the structure in $\mathbf{V}^{(r)}$, the recovered signals are guaranteed to be exponential polynomials, and the underlying poles can be recovered easily. These parametric constraints are modeled in the SDF framework [83] using `struct_confvander` and `sdf_nls`. Figure 4.5 and Figure 4.6 show the median time and error on the mixing matrix over 50 noise realizations. Each algorithm is initialized three and six times using random variables for the LL1 and BTD model, respectively. The time and error for the best initialization, i.e., the one resulting in the lowest error on the mixing matrix, is retained. The timing results in Figure 4.5 clearly show that implicit Hankelization drastically reduces the computation time, allowing longer signals to be analyzed. The computation of the constrained BTD is sensitive to the initialization and is difficult from an optimization point-of-view, as the factors are ill-conditioned due to the generalized Vandermonde structure. This results in higher computation times and a lower accuracy for random initializations. Being able to analyze longer signal allows one to improve the accuracy for a fixed SNR, as is clear from Figure 4.6: by taking 100 times as many samples, E decreases by a factor 10.

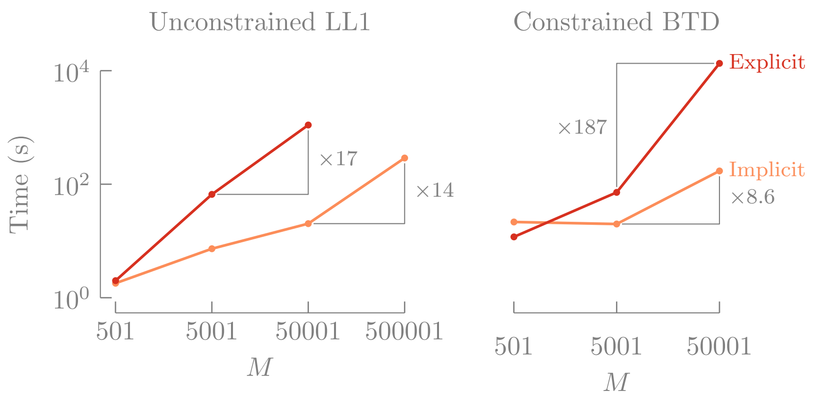

FIGURE 4.5. *For both the unconstrained LL1 decomposition and the constrained BTD, the computation time using implicit tensorization increases more slowly, enabling large-scale applications. For the LL1 decomposition, the time increases with a factor* $14 \approx 13$ $(\mathcal{O}(M \log_2 M))$ *using the efficient representation and with a factor* 17 *when the full tensor is used. The latter factor is better than the expected factor* 100 *as fewer iterations are required. Computing the constrained BTD is more difficult due to ill-conditioned factors and the variation in time is large causing larger deviatons from the expected scaling factors. The results are medians over 50 experiments with multiple initializations.*

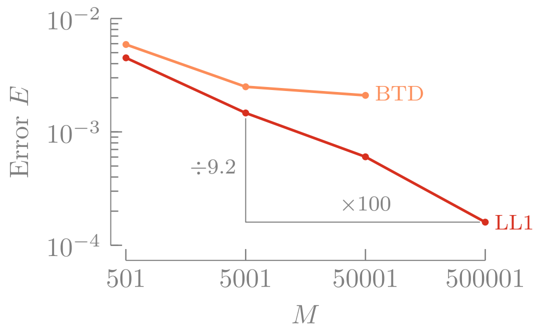

FIGURE 4.6. *The relative error* E *of the estimated mixing matrix decreases when the number of samples* M *increases. In the case of the unconstrained LL1 decomposition,* E *decreases with a factor* $9.2 \approx 10$ *when 100 times as many samples are used. The constrained BTD is more difficult from an optimization point-of-view and involves ill-conditioned factor matrices, resulting in a modest improvement in terms of error. The errors are computed using implicit tensorization and are medians over 50 noise realizations with multiple initializations. The results for full tensors are similar.*

4.4. Fetal ECG extraction. If the underlying sources can be modeled by rational functions, BSS techniques based on Löwnerization can be used to recover the mixing matrix $\mathbf{M}$ and the sources $\mathbf{S}$ in (4.1). In this last experiment, the goal is to separate the heart beats from the mother and the fetus, given five measured signals from five abdominal electrodes; see [20]. (The data is publicly available at [24].)

As illustrated in [27], each signal can be modeled by a rational function and the LL1 decomposition of the Löwnerized data can be used to recover the mixing matrix. Concretely, the signal matrix $\mathbf{X} \in \mathbb{R}^{2500\times5}$ is explicitly and implicitly mapped to a tensor $\mathcal{T} \in \mathbb{R}^{1250\times1250\times5}$ and an LL1 decomposition with $L_1 = L_2 = 56$, and $L_3 = 45$ is used to recover the 5×3 mixing matrix $\mathbf{M}$ using the Gauss–Newton implementation in `ll1_nls`. (The values for L_r have been determined experimentally.) Note that we need more than one source to model the mother's heartbeat which is a three dimensional signal [10]. The sources are recovered as $\mathbf{S} = \mathbf{X}\mathbf{M}^{\mathsf{T}^\dagger}$ and are shown in Figure 4.7. The difference in computation time is relatively limited (81 s and 51 s for the explicit and implicit tensorization, respectively), as the multilinear ranks determined by L_r, $r = 1, 2, 3$, are quite high in this application. However, to store $\mathcal{T}$ explicitly, 60 MB is required, while only 160 kB is needed to store the signal matrix and evaluation points in the efficient representation, allowing the signals to be processed on the limited resources of medical hardware.

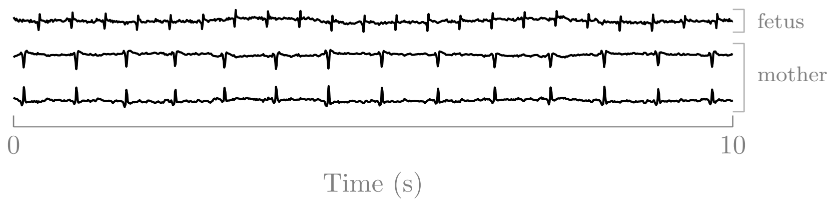

Figure 4.7. *The heart beats of the fetus and the mother can be separated using Löwnerization. An LL1 decomposition with $L_1 = L_2 = 56$, and $L_3 = 45$ is computed from the implicitly Löwnerized measured signals, after which the signals are reconstructed (shown) by pseudoinversion of mixing matrix.*

5. Conclusion.

By rewriting the objective function and gradient expressions commonly used to compute tensor decompositions, four core operations are exposed: squared norm, inner product, matricized tensor times Khatri-Rao product and matricized tensor times Kronecker product. By specializing these operations for efficient representations of tensors, both the computational and memory complexity are reduced from $\mathcal{O}$ (tensor entries) to $\mathcal{O}$ (parameters) as illustrated for the polyadic, Tucker and Tensor Train format, as well as for Hankelization and Löwnerization. The operations can be used in many decomposition algorithms and frameworks, including ALS, second-order algorithms and algorithms for constrained and/or coupled decompositions. The numerical consequences of exploiting efficient representations are studied in the case a highly accurate solution is required. Finally, important concepts such as Tucker or TT compression for constrained decompositions and implicit tensorization, allow large-scale datasets to be handled easily, as illustrated for nonnegative CPD and the Hankelization of mixtures of exponentials.

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, https://doi.org/10.1002/cem.1335.

[2] E. Acar, T. G. Kolda, and D. M. Dunlavy, *All-at-once optimization for coupled matrix and tensor factorizations*, May 2011.

[3] R. Badeau and R. Boyer, *Fast multilinear singular value decomposition for structured tensors*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1008–1021, https://doi.org/10.1137/060655936.

[4] B. Bader and T. G. Kolda, *Efficient matlab computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30 (2007), pp. 205–231, https://doi.org/10.1137/060676489.

[5] H. N. Bharath, D. Sima, N. Sauwen, U. Himmelreich, L. De Lathauwer, and S. Van Huffel, *Nonnegative canonical polyadic decomposition for tissue-type differentiation in gliomas*, IEEE J. Biomed. Health Inform., 21 (2017), pp. 1124–1132, https://doi.org/10.1109/JBHI.2016.2583539.

[6] R. Bro, *Multi-way analysis in the food industry: models, algorithms, and applications*, PhD thesis, University of Amsterdam, 1998.

[7] R. Bro and C. A. Andersson, *Improving the speed of multiway algorithms: Part II: Compression*, Chemometr. Intell. Lab., 42 (1998), pp. 105–113, https://doi.org/10.1016/S0169-7439(98)00011-2.

[8] R. Bro, R. A. Harshman, N. D. Sidiropoulos, and M. E. Lundy, *Modeling multi-way data with linearly dependent loadings*, J. Chemometrics, 23 (2009), pp. 324–340, https://doi.org/10.1002/cem.1206.

[9] C. Caiafa and A. Cichocki, *Generalizing the column-row matrix decomposition to multi-way arrays*, Linear Algebra and its Applications, 433 (2010), pp. 557 – 573, https://doi.org/10.1016/j.laa.2010.03.020.

[10] D. Callaerts, B. De Moor, J. Vandewalle, W. Sansen, G. Vantrappen, and J. Janssens, *Comparison of SVD methods to extract the foetal electrocardiogram from cutaneous electrode signals*, Med. Biol. Eng. Comput., 28 (1990), pp. 217–224, https://doi.org/10.1007/BF02442670.

[11] 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, https://doi.org/10.1007/bf02310791.

[12] J. D. Carroll, S. Pruzansky, and J. B. Kruskal, *CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters*, Psychometrika, 45 (1980), pp. 3–24, https://doi.org/10.1007/bf02293596.

[13] J. H. Choi and S. V. N. Vishwanathan, *DFacTo: Distributed factorization of tensors*, in Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14, Cambridge, MA, USA, 2014, MIT Press, pp. 1296–1304.

[14] A. Cichocki, D. Mandic, A.-H. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer, *Tensor decompositions for signal processing applications: From two-way to multiway component analysis*, IEEE Signal Process. Mag., 32 (2015), pp. 145–163, https://doi.org/10.1109/msp.2013.2297439.

[15] J. E. Cohen, R. Cabral Farias, and P. Comon, *Fast decomposition of large nonnegative tensors*, IEEE Signal Process. Lett., 22 (2015), pp. 862–866, https://doi.org/10.1109/lsp.2014.2374838.

[16] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*, Academic press, 2010.

[17] A. de Almeida, G. Favier, and J. C. M. Mota, *A constrained factor decomposition with application to MIMO antenna systems*, IEEE Trans. Signal Process., 56 (2008), pp. 2429–2442, https://doi.org/10.1109/TSP.2008.917026.

[18] L. De Lathauwer, *Decompositions of a higher-order tensor in block terms — Part II: Definitions and uniqueness*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1033–1066, https://doi.org/10.1137/070690729.

[19] L. De Lathauwer, *Blind separation of exponential polynomials and the decomposition of a tensor in rank-$(L_r, L_r, 1)$ terms*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1451–1474, https://doi.org/10.1137/100805510.

[20] L. De Lathauwer, B. de Moor, and J. Vandewalle, *Fetal electrocardiogram extraction by blind source subspace separation*, IEEE Trans. Biomed. Eng., 47 (2000), pp. 567–572, https://doi.org/10.1109/10.841326.

[21] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278, https://doi.org/10.1137/S0895479896305696.

[22] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342, https://doi.org/10.1137/S0895479898346995.

[23] L. De Lathauwer and D. Nion, *Decompositions of a higher-order tensor in block terms — Part III: Alternating least squares algorithms*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1067–1083, https://doi.org/10.1137/070690730.

[24] B. De Moor, *DaISy: Database for the identification of systems.* Department of Electrical Engineering, ESAT/SISTA, KU Leuven, Belgium, URL: http://www.esat.kuleuven.ac.be/

sista/daisy/, Jul. 1. 2018. [Used dataset: cutaneous potential recordings of a pregnant woman, biomedial systems, 96-012.].

[25] O. Debals and L. De Lathauwer, *The concept of tensorization*. Technical Report 17–99, ESAT-STADIUS, KU Leuven, Belgium, 2017.

[26] O. Debals, L. De Lathauwer, and M. Van Barel, *About higher-order Löwner tensors*. Technical Report 17–98, ESAT-STADIUS, KU Leuven, Belgium, 2017.

[27] O. Debals, M. Van Barel, and L. De Lathauwer, *Löwner-based blind signal separation of rational functions with applications*, IEEE Trans. Signal Process., 64 (2016), pp. 1909–1918, https://doi.org/10.1109/tsp.2015.2500179.

[28] W. Ding, L. Qi, and Y. Wei, *Fast Hankel tensor-vector product and its application to exponential data fitting*, Numer. Linear Algebra Appl., 22 (2015), pp. 814–832, https://doi.org/10.1002/nla.1970.

[29] I. Gohberg and V. Olshevsky, *Complexity of multiplication with vectors for structured matrices*, Linear Algebra Appl., 202 (1994), pp. 163–192, https://doi.org/10.1016/0024-3795(94)90189-9.

[30] I. Gohberg and V. Olshevsky, *Fast algorithms with preprocessing for matrix-vector multiplication problems*, Journal of Complexity, 10 (1994), pp. 411–427, https://doi.org/10.1006/jcom.1994.1021.

[31] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054, https://doi.org/10.1137/090764189.

[32] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78, https://doi.org/10.1002/gamm.201310004.

[33] W. Hackbusch, *Tensor spaces and numerical tensor calculus*, Springer Series in Computational Mathematics, (2012), https://doi.org/10.1007/978-3-642-28027-6.

[34] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Hierarchical Kronecker tensor-product approximations*, J. Numer. Math., 13 (2005), https://doi.org/10.1515/1569395054012767.

[35] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722, https://doi.org/10.1007/s00041-009-9094-9.

[36] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, Journal of Mathematics and Physics, 6 (1927), pp. 164–189, https://doi.org/10.1002/sapm192761164.

[37] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the Tensor Train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713, https://doi.org/10.1137/100818893.

[38] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, *A flexible and efficient algorithmic framework for constrained matrix and tensor factorization*, IEEE Trans. Signal Process., 64 (2016), pp. 5052–5065, https://doi.org/10.1109/TSP.2016.2576427.

[39] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, *Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 115–135, https://doi.org/10.1137/090764827.

[40] M. Ishteva, L. De Lathauwer, P.-A. Absil, and S. Van Huffel, *The best rank-(R_1, R_2, R_3) approximation of tensors by means of a geometric Newton method*, AIP Conference Proceedings, 1048 (2008), pp. 274–277, https://doi.org/10.1063/1.2990911.

[41] B. Jeon, I. Jeon, L. Sael, and U. Kang, *SCouT: Scalable coupled matrix-tensor factorization — algorithm and discoveries*, in 2016 IEEE 32nd International Conference on Data Engineering (ICDE), IEEE, May 2016, pp. 811–822, https://doi.org/10.1109/icde.2016.7498292.

[42] I. Jeon, E. E. Papalexakis, U. Kang, and C. Faloutsos, *HaTen2: Billion-scale tensor decompositions*, in 2015 IEEE 31st International Conference on Data Engineering, IEEE, Apr. 2015, pp. 1047–1058, https://doi.org/10.1109/icde.2015.7113355.

[43] U. Kang, E. E. Papalexakis, A. Harpale, and C. Faloutsos, *GigaTensor: Scaling tensor analysis up by 100 times - algorithms and discoveries*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12, (2012), https://doi.org/10.1145/2339530.2339583.

[44] O. Kaya and B. Uçar, *Scalable sparse tensor decompositions in distributed memory systems*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15, ACM, 2015, pp. 77:1–77:11, https://doi.org/10.1145/2807591.2807624.

[45] O. Kaya and B. Uçar, *High performance parallel algorithms for the Tucker decomposition of sparse tensors*, in 2016 45th International Conference on Parallel Processing (ICPP), IEEE, Aug. 2016, pp. 103–112, https://doi.org/10.1109/icpp.2016.19.

[46] C. Kelley, *Iterative Methods for Optimization*, SIAM, 1999.

[47] B. N. Khoromskij, *Structured rank-$(r_1,\dots,r_D)$ decomposition of function-related tensors in $\mathbb{R}^D$*, Comput. Methods Appl. Math., 6 (2006), https://doi.org/10.2478/cmam-2006-0010.

[48] B. N. Khoromskij and V. Khoromskaia, *Low rank Tucker-type tensor approximation to classical potentials*, Central European Journal of Mathematics, 5 (2007), pp. 523–550, https://doi.org/10.2478/s11533-007-0018-0.

[49] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comput., 31 (2009), pp. 3002–3026, https://doi.org/10.1137/080730408.

[50] H. A. L. Kiers and R. A. Harshman, *Relating two proposed methods for speedup of algorithms for fitting two- and three-way principal component and related multilinear models*, Chemometr. Intell. Lab., 36 (1997), pp. 31–40, https://doi.org/10.1016/s0169-7439(96)00074-3.

[51] T. G. Kolda, *Orthogonal tensor decompositions*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 243–255, https://doi.org/10.1137/S0895479800368354.

[52] T. G. Kolda and B. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111X.

[53] T. G. Kolda and J. Sun, *Scalable tensor decompositions for multi-aspect data mining*, 2008 Eighth IEEE International Conference on Data Mining, (2008), https://doi.org/10.1109/icdm.2008.89.

[54] P. M. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97, https://doi.org/10.1007/bf02293599.

[55] A. P. Liavas and N. D. Sidiropoulos, *Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers*, IEEE Trans. Signal Process., 63 (2015), pp. 5450–5463, https://doi.org/10.1109/tsp.2015.2454476.

[56] M. W. Mahoney, M. Maggioni, and P. Drineas, *Tensor-CUR decompositions for tensor-based data*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 957–987, https://doi.org/10.1137/060665336.

[57] T. Müller, K. Kruppa, G. Lichtenberg, and N. Réhault, *Fault detection with qualitative models reduced by tensor decomposition methods*, IFAC-PapersOnLine, 48 (2015), pp. 416–421, https://doi.org/10.1016/j.ifacol.2015.09.562.

[58] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, second edition ed., 2006.

[59] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Ann. Physics, 349 (2014), pp. 117 – 158, https://doi.org/10.1016/j.aop.2014.06.013.

[60] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[61] I. V. Oseledets and S. Dolgov, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739, https://doi.org/10.1137/110833142.

[62] I. V. Oseledets, S. Dolgov, V. Kazeev, D. V. Savostyanov, O. Lebedeva, P. Zhlobich, T. Mach, and L. Song, *TT-Toolbox*. Available online at https://github.com/oseledets/TT-Toolbox.

[63] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 939–956, https://doi.org/10.1137/060655894.

[64] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Linear algebra for tensor problems*, Computing, 85 (2009), pp. 169–188, https://doi.org/10.1007/s00607-009-0047-6.

[65] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759, https://doi.org/10.1137/090748330.

[66] I. V. Oseledets and E. E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88, https://doi.org/10.1016/j.laa.2009.07.024.

[67] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, *ParCube: Sparse parallelizable CANDECOMP-PARAFAC tensor decomposition*, ACM Trans. Knowl. Discov. Data, 10 (2015), pp. 1–25, https://doi.org/10.1145/2729980.

[68] J. M. Papy, L. De Lathauwer, and S. Van Huffel, *Exponential data fitting using multilinear algebra: the single-channel and multi-channel case*, Numer. Linear Algebra Appl., 12 (2005), pp. 809–826, https://doi.org/10.1002/nla.453.

[69] 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, https://doi.org/10.1109/tsp.2013.2269903.

[70] M. J. Reynolds, G. Beylkin, and A. Doostan, *Optimization via separated representations and the canonical tensor decomposition*, J. Comput. Phys., 348 (2017), pp. 220–230, https://doi.org/10.1016/j.jcp.2017.07.012.

[71] J.-P. Royer, N. Thirion-Moreau, and P. Comon, *Computing the polyadic decomposition of nonnegative third order tensors*, Signal Processing, 91 (2011), pp. 2159–2171, https://doi.org/10.1016/j.sigpro.2011.03.006.

[72] D. V. Savostyanov, *Fast revealing of mode ranks of tensor in canonical form*, Numer. Math. Theor. Meth. Appl, 2 (2009), pp. 439–444, https://doi.org/10.4208/nmtma.2009.m9006s.

[73] D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *Fast truncation of mode ranks for bilinear tensor operations*, Numer. Linear Algebra Appl., 19 (2011), pp. 103–111, https://doi.org/10.1002/nla.765.

[74] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, *Parallel factor analysis in sensor array processing*, IEEE Trans. Signal Process., 48 (2000), pp. 2377–2388, https://doi.org/10.1109/78.852018.

[75] 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 (2017), pp. 3551–3582, https://doi.org/10.1109/TSP.2017.2690524.

[76] N. D. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos, *Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition*, IEEE Signal Process. Mag., 31 (2014), pp. 57–70, https://doi.org/10.1109/MSP.2014.2329196.

[77] S. Smith and G. Karypis, *Tensor-matrix products with a compressed sparse tensor*, in Proceedings of the 5th Workshop on Irregular Applications Architectures and Algorithms - IA3 '15, ACM, 2015, pp. 5:1–5:7, https://doi.org/10.1145/2833179.2833183.

[78] S. Smith and G. Karypis, *A medium-grained algorithm for sparse tensor factorization*, in 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, May 2016, pp. 902–911, https://doi.org/10.1109/ipdps.2016.113.

[79] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis, *SPLATT: Efficient and parallel sparse tensor-matrix multiplication*, in 2015 IEEE International Parallel and Distributed Processing Symposium, IEEE, May 2015, pp. 61–70, https://doi.org/10.1109/ipdps.2015.27.

[80] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer, *Exact line and plane search for tensor optimization*, Comput. Optim. Appl., 63 (2015), pp. 121–142, https://doi.org/10.1007/s10589-015-9761-5.

[81] L. Sorber, M. Van Barel, and L. De Lathauwer, *Unconstrained optimization of real functions in complex variables*, SIAM J. Optim., 22 (2012), pp. 879–898, https://doi.org/10.1137/110832124.

[82] 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, https://doi.org/10.1137/120868323.

[83] L. Sorber, M. Van Barel, and L. De Lathauwer, *Structured data fusion*, IEEE J. Sel. Topics Signal Process., 9 (2015), pp. 586–600, https://doi.org/10.1109/JSTSP.2015.2400415.

[84] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Stat. Data Anal., 50 (2006), pp. 1700 – 1734, https://doi.org/10.1016/j.csda.2004.11.013.

[85] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, *Nonlinear least squares updating of the canonical polyadic decomposition*, in 2017 25th European Signal Processing Conference (EUSIPCO17), Aug. 2017, pp. 693–697, https://doi.org/10.23919/EUSIPCO.2017.8081290.

[86] N. Vannieuwenhoven, *Condition numbers for the tensor rank decomposition*, Linear Algebra Appl., 535 (2017), pp. 35–86, https://doi.org/10.1016/j.laa.2017.08.014.

[87] 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, https://doi.org/10.1137/14097968x.

[88] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A new truncation strategy for the higher-order singular value decomposition*, SIAM J. Sci. Comput., 34 (2012), pp. A1027–A1052, https://doi.org/10.1137/110836067.

[89] N. Vervliet and L. De Lathauwer, *A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors*, IEEE J. Sel. Topics Signal Process., 10 (2016), pp. 284–295, https://doi.org/10.1109/JSTSP.2015.2503260.

[90] N. Vervliet and L. De Lathauwer, *Numerical optimization based algorithms for data fusion*, in Data Fusion Methodology and Applications, M. Cocchi, ed., Elsevier, 2018. Accepted for publication.

[91] N. Vervliet, O. Debals, and L. De Lathauwer, *Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization*, in 2016 50th Asilomar Conference on Signals, Systems and Computers, Nov. 2016, pp. 1733–1738, https://doi.org/10.1109/ACSSC.2016.7869679.

[92] N. Vervliet, O. Debals, and L. De Lathauwer, *Canonical polyadic decomposition of incomplete tensors with linearly constrained factors*. Technical Report 16–172, ESAT-STADIUS, KU Leuven, Belgium, Apr. 2017.

[93] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, *Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis*, IEEE Signal Process. Mag., 31 (2014), pp. 71–79, https://doi.org/10.1109/MSP.2014.2329429.

[94] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, *Tensorlab 3.0*, Mar. 2016. Available online at https://www.tensorlab.net.

[95] Z. Yawen, D. Guangjun, and X. Zhixiang, *Hyperspectral image tensor feature extraction based on fusion of multiple spectral-spatial features*, in Proceedings of the 2016 International Conference on Intelligent Information Processing - ICIIP '16, ACM, 2016, pp. 43:1–43:8, https://doi.org/10.1145/3028842.3028885.

[96] G. Zhou, A. Cichocki, and S. Xie, *Decomposition of big tensors with low multilinear rank*, 2014.

[97] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, *Nonnegative matrix and tensor factorizations: An algorithmic perspective*, IEEE Signal Process. Mag., 31 (2014), pp. 54–65, https://doi.org/10.1109/msp.2014.2298891.

[98] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, *Efficient nonnegative Tucker decompositions: Algorithms and uniqueness*, IEEE Trans. Image Process., 24 (2015), pp. 4990–5003, https://doi.org/10.1109/tip.2015.2478396.