NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2017; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Linear Systems with a Canonical Polyadic Decomposition Constrained Solution: Algorithms and Applications

M. Boussé[1]*, N. Vervliet[1], I. Domanov[2], O. Debals[1,2], L. De Lathauwer[1,2]

[1]*Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.* [2]*Group Science, Engineering and Technology, KU Leuven Kulak, E. Sabbelaan 53, 8500 Kortrijk, Belgium.*

SUMMARY

Real-life data often exhibit some structure and/or sparsity, allowing one to use parsimonious models for compact representation and approximation. When considering matrix and tensor data, low-rank models such as the (multilinear) singular value decomposition (SVD), canonical polyadic decomposition (CPD), tensor train (TT), and Hierachical Tucker (HT) model are very common. The solution of (large-scale) linear systems is often structured in a similar way, allowing one to use compact matrix and tensor models as well. In this paper we focus on linear systems with a CPD constrained solution (LS-CPD). Our main contribution is the development of optimization-based and algebraic methods to solve LS-CPDs. Furthermore, we propose a condition that guarantees generic uniqueness of the obtained solution. We also show that LS-CPDs provide a broad framework for the analysis of multilinear systems of equations. The latter are a higher-order generalization of linear systems, similar to tensor decompositions being a generalization of matrix decompositions. The wide applicability of LS-CPDs in domains such as classification, multilinear algebra, and signal processing is illustrated. Copyright © 2017 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: linear systems; multilinear systems; higher-order tensor; tensor decompositions; multilinear algebra;

1. INTRODUCTION

Real-life data can often be modeled using compact representations because of some intrinsic structure and/or sparsity [1]. Well-known representations are low-rank matrix and tensor models such as nonnegative matrix factorization (NMF), the (multilinear) singular value decomposition (SVD), canonical polyadic decomposition (CPD), tensor train (TT), and hierarchical Tucker (HT) models [2, 3, 4, 5, 6, 7, 8]. Examples of data that can be represented or well approximated by such models are exponential polynomials, rational functions (and in a broader sense smooth signals), as well as periodic functions [9, 10, 11, 12]. When dealing with vector/matrix data, one often reshapes the data into higher-order tensors which are then modeled using low-rank approximations, enabling efficient processing in the compressed format. This strategy has been used in tensor-based scientific computing and signal processing to handle various large-scale problems [9, 10, 11, 13].

Similarly, the solution of a (large-scale) linear system can often be expressed by a low-rank tensor. Such problems are well-known in tensor-based scientific computing, see [11]. They arise, e.g., after discretizing high-dimensional partial differential equations. The low-rank model ensures efficient computations and a compact representation of the solution. In such large-scale problems, one often

*Correspondence to: Martijn Boussé, Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. Email: martijn.bousse@kuleuven.be.

assumes that the coefficient matrix and/or right-hand side have some additional structure or can also be expressed using a tensor model. Several methods have been developed for linear systems with a Kronecker structured coefficient matrix and a CPD structured solution such as the projection method [14], alternating least squares (ALS) [15], and a gradient method [16]. TTs or HT models are also often used because they combine large compression rates and good numerical properties [7, 8].

In this paper, we present a new framework for linear systems of equations with a CPD constrained solution, abbreviated as LS-CPD. In other words, we want to solve linear systems of the form:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with} \quad \mathbf{x} = \operatorname{vec}\left(\text{CPD}\right),$$

in which $\operatorname{vec}(\cdot)$ is a vectorization. A simple second-order rank-1 example is $\mathbf{x} = \operatorname{vec}(\mathbf{u} \otimes \mathbf{v})$ with $\otimes$ the outer product. In particular we develop algebraic as well as optimization-based algorithms that properly address the CPD structure. A naive method to solve LS-CPDs could be to first solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ without structure and subsequently decompose a tensorized version $\operatorname{ten}(\mathbf{x})$ of the obtained solution. This approach works well if the linear system is overdetermined, but, in contrast to our algebraic and optimization-based methods, fails in the underdetermined case. The proposed optimization-based method computes a solution of the LS-CPD problem by minimizing a least-squares objective function. We have derived expressions for the gradient, Jacobian, and approximation of the Hessian which are the ingredients for standard quasi-Newton (QN) and nonlinear least squares (NLS) techniques. We use the complex optimization framework in Tensorlab, a toolbox for tensor computations in MATLAB, as numerical optimization solver [17, 18, 19, 20]. The optimization-based methods allow us to work much more efficiently and avoid error accumulation in contrast to the naive or algebraic methods. The latter two methods can be used to obtain a good initialization for the optimization-based methods when considering perturbed LS-CPD problems. Our framework can be extended to other tensor decompositions such as the block term decomposition (BTD), multilinear singular value decomposition (MLSVD), low-multilinear rank approximation (LMLRA), TT or HT models [5, 6, 11, 21].

Furthermore, LS-CPDs can be interpreted as multilinear systems of equations which are a generalization of linear systems of equations. The latter can be expressed by a matrix-vector product between the coefficient matrix and the solution vector, e.g., $\mathbf{A}\mathbf{x} = \mathbf{b}$, or, equivalently, $\mathbf{A} \cdot_2 \mathbf{x}^{\mathrm{T}}$, using the mode-$n$ product [3]. The generalization to a multilinear system is then straightforward because it can be expressed by tensor-vector products between the coefficient tensor and multiple solution vectors: $\mathcal{A} \cdot_2 \mathbf{x}^{\mathrm{T}} \cdot_3 \mathbf{y}^{\mathrm{T}} = \mathbf{b}$. This is very similar to tensor decompositions which are higher-order generalizations of matrix decompositions [3, 4, 5]. However, in contrast to tensor decompositions, the domain of multilinear systems is relatively unexplored. To the best of the authors' knowledge, only a few cases have been studied in a disparate manner such as the fully symmetric rank-1 tensor case with a particular coefficient structure [22], sets of bilinear equations [23, 24, 25], and a particular type of multilinear systems that can be solved via so-called tensor inversion [26]. LS-CPDs provide a general framework to solve multilinear systems, see Figure 1.

The CPD structure in LS-CPDs strongly reduces the number of parameters needed to represent the solution. For example, a cubic third-order tensor of size $I \times I \times I$ contains I^3 entries but its CPD needs only $\mathcal{O}(3RI)$ parameters with R the number of terms in the decomposition. The possibly very compact representation of the solution enables one to solve the LS-CPD problem for the underdetermined case in a compressed-sensing (CS) style [1, 27]. A similar idea has been studied for the low-rank matrix case [28]. In contrast to well-known CS reconstruction conditions, we derive a uniqueness condition for LS-CPDs that holds with probability one. In particular, we derive a generic uniqueness condition for the solution $\mathbf{x}$ of the LS-CPD problems given a coefficient matrix $\mathbf{A}$ of which the entries are drawn from absolutely continuous probability density functions.

LS-CPDs appear in a wide range of applications, see, e.g., [29, 30, 31], but the CPD structure is often not recognized or not fully exploited. In this paper, the applicability of LS-CPDs is illustrated in three different domains: classification, multilinear algebra, and signal processing. In the first case, we show that tensor-based classification can be formulated as the computation of a LS-CPD. Although we illustrate the technique with face recognition [32], one can consider other classification tasks such as irregular heartbeat classification and various computer vision

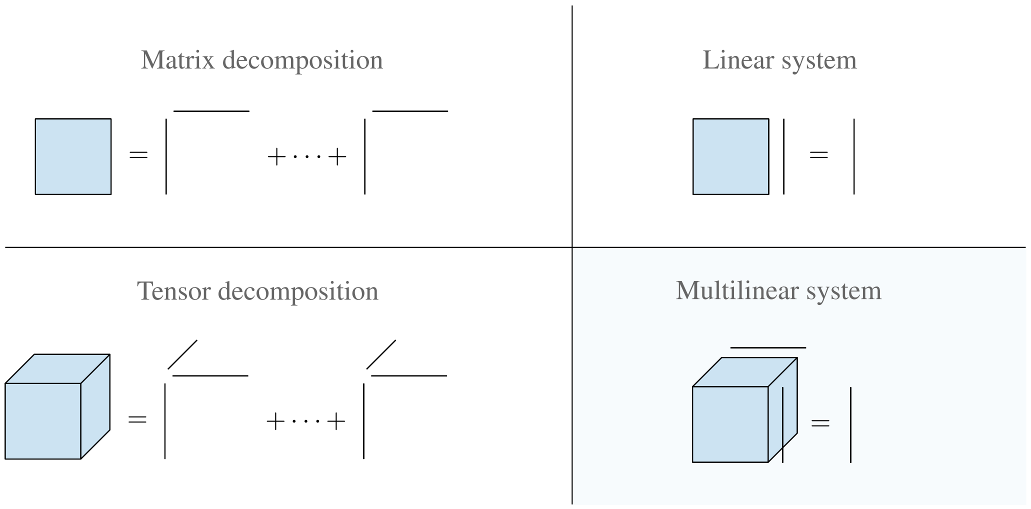

Figure 1. Tensor decompositions are a higher-order generalization of matrix decompositions and are well-known tools in many applications within various domains. Although multilinear systems are a generalization of linear systems in a similar way, this domain is relatively unexplored. LS-CPDs can be interpreted as multilinear systems of equations, providing a broad framework for the analysis of these types of problems.

problems [29, 33, 34]. Next, the construction of real-valued tensor that has particular multilinear singular values is formulated as a LS-CPD. By properly exploiting the symmetry in the resulting problem, our method is faster than literature methods. We conclude with the blind deconvolution of constant modulus such as 4-QAM or BPSK signals using LS-CPDs.

In the remainder of this introduction we give an overview of the notation, basic definitions, and multilinear algebra prerequisites. In Section 2 we define LS-CPDs and briefly discuss structure and generic uniqueness. Next, we develop an algebraic algorithm and an optimization-based algorithm to compute LS-CPDs in Section 3. Numerical experiments and applications are presented in Sections 4 and 5, respectively. We conclude the paper and discuss possible future work in Section 6.

1.1. Notation and definitions

A tensor is a higher-order generalization of a vector (first-order) and a matrix (second-order). We denote tensors by calligraphic letters, e.g., $\mathcal{A}$. Vectors and matrices are denoted by bold lower and bold uppercase letters, respectively, e.g., $\mathbf{a}$ and $\mathbf{A}$. A mode-n vector of a tensor $\mathcal{A} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ (with $\mathbb{K}$ meaning $\mathbb{R}$ or $\mathbb{C}$) is defined by fixing every index except the nth, e.g., $\mathbf{a}_{i_1 \ldots i_{n-1} : i_{n+1} \ldots i_N}$, and is a natural extension of a row or a column of a matrix. The mode-n unfolding of $\mathcal{A}$ is the matrix $\mathbf{A}_{(n)}$ with the mode-n vectors as its columns (following the ordering convention in [3]). An Mth-order slice of $\mathcal{A}$ is obtained by fixing all but M indices. The vectorization of $\mathcal{A}$, denoted as $\text{vec}(\mathcal{A})$, maps each element $a_{i_1 i_2 \ldots i_N}$ onto $\text{vec}(\mathcal{A})_j$ with $j = 1 + \sum_{k=1}^{N} (i_k - 1) J_k$ and $J_k = \prod_{m=1}^{k-1} I_m$ (with $\prod_{m}^{k-1}(\cdot) = 1$ if $m > k-1$). The $\text{unvec}(\cdot)$ operation is defined as the inverse of $\text{vec}(\cdot)$.

The nth element in a sequence is indicated by a superscript between parentheses, e.g., $\{\mathbf{A}^{(n)}\}_{n=1}^{N}$. The complex conjugate, transpose, conjugated transpose, inverse, and pseudoinverse are denoted as $\bar{\cdot}$, $\cdot^{\mathrm{T}}$, $\cdot^{\mathrm{H}}$, $\cdot^{-1}$ and $\cdot^{\dagger}$, respectively. A vector of length K with all entries equal to one is denoted as $\mathbf{1}_K$. The identity matrix of size $K \times K$ is denoted as $\mathbf{I}_K$. The binomial coefficient is denoted by $C_n^k = \frac{n!}{(n-k)!k!}$. $\mathbf{A} = \text{diag}(\mathbf{a})$ is a diagonal matrix with the elements of $\mathbf{a}$ on the main diagonal.

The outer and Kronecker product are denoted by $\otimes$ and $\otimes$, respectively, and are related through a vectorization: $\text{vec}(\mathbf{a} \otimes \mathbf{b}) = \mathbf{b} \otimes \mathbf{a}$. The mode-$n$ product of a tensor $\mathcal{A} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{B} \in \mathbb{K}^{J_n \times I_n}$, denoted by $\mathcal{A} \cdot_n \mathbf{B} \in \mathbb{K}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$, is defined element-wise as $(\mathcal{A} \cdot_n \mathbf{B})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 i_2 \ldots i_N} b_{j_n i_n}$. Hence, each mode-$n$ vector of the tensor $\mathcal{A}$ is multiplied with the matrix $\mathbf{B}$, i.e., $(\mathcal{A} \cdot_n \mathbf{B})_{(n)} = \mathbf{B}\mathbf{A}_{(n)}$. The inner product of two tensors $\mathcal{A}, \mathcal{B} \in$

$\mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\langle \mathcal{A}, \mathcal{B} \rangle$ and defined as $\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} a_{i_1 i_2 \ldots i_N} \bar{b}_{i_1 i_2 \ldots i_N}$. The Khatri–Rao and Hadamard product are denoted by $\odot$ and $*$, respectively.

An Nth-order tensor has rank one if it can be written as the outer product of N nonzero vectors. The rank of a tensor is defined as the minimal number of rank-1 terms that generate the tensor as their sum. The mode-n rank of a tensor is defined as the rank of the mode-n unfolding. The multilinear rank of an Nth-order tensor is equal to the tuple of mode-n ranks.

1.2. *Multilinear algebraic prerequisites*

The CPD is a powerful model for various applications within signal processing, biomedical sciences, computer vision, data mining and machine learning [3, 4, 5].

Definition 1. A *polyadic decomposition* (PD) writes an Nth-order tensor $\mathcal{T} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ as a sum of R rank-1 terms:

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \otimes \mathbf{u}_r^{(2)} \otimes \cdots \otimes \mathbf{u}_r^{(N)} = \left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right],$$

in which the columns of the factor matrices $\mathbf{U}^{(n)} \in \mathbb{K}^{I_n \times R}$ are equal to the factor vectors $\mathbf{u}_r^{(n)}$ for $1 \leq r \leq R$. The PD is called *canonical* (CPD) if R is equal to the rank of $\mathcal{T}$, i.e., R is minimal.

The decomposition is *essentially unique* if it is unique up to trivial permutation of the rank-1 terms and scaling and counterscaling of the factors in the same rank-1 term. In the matrix case ($N = 2$) the CPD is not unique without additional assumptions for $R > 1$. Uniqueness is typically expected under rather mild conditions when $N > 2$, see, e.g., [35, 36, 37, 38, 39] and references therein.

The multilinear singular value decomposition (MLSVD) of a higher-order tensor is a multilinear generalization of the singular value decomposition (SVD) of a matrix [4, 5, 6].

Definition 2. A *multilinear singular value decomposition* (MLSVD) writes a tensor $\mathcal{T} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ as the product

$$\mathcal{T} = \mathcal{S} \cdot_1 \mathbf{U}^{(1)} \cdot_2 \mathbf{U}^{(2)} \cdots \cdot_N \mathbf{U}^{(N)} = \left[\!\left[\mathcal{S}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right]. \quad (1)$$

The factor matrices $\mathbf{U}^{(n)} \in \mathbb{K}^{I_n \times I_n}$, for $1 \leq n \leq N$, are unitary matrices and the core tensor $\mathcal{S} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ is ordered and all-orthogonal [6].

The (truncated) MLSVD is a powerful tool in various applications such as compression, dimensionality reduction, and face recognition [3, 29, 40]. The decomposition is related to the low-multilinear rank approximation (LMLRA) and the Tucker decomposition (TD), see [6, 41] and references therein. The mode-n unfolding of (1) is given by:

$$\mathbf{T}_{(n)} = \mathbf{U}^{(n)} \mathbf{S}_{(n)} \left(\mathbf{U}^{(N)} \otimes \cdots \otimes \mathbf{U}^{(n+1)} \otimes \mathbf{U}^{(n-1)} \otimes \cdots \otimes \mathbf{U}^{(1)} \right)^{\mathrm{T}}.$$

2. LINEAR SYSTEMS WITH A CPD CONSTRAINED SOLUTION

First, we define linear systems with a CPD constrained solution in Subsection 2.1. Next, we discuss structure of the coefficient matrix and generic uniqueness in Subsection 2.2 and 2.3, respectively.

2.1. *Definition*

In this paper, linear systems of equations of which the solution can be represented by a tensor decomposition are considered. We limit ourselves to linear systems with a CPD structured solution, abbreviated as LS-CPD, but one can also use other decompositions such as the MLSVD, TT or HT [6, 7, 11]. Concretely, consider a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with coefficient matrix $\mathbf{A} \in \mathbb{K}^{M \times K}$, solution vector $\mathbf{x} \in \mathbb{K}^{K}$, and right-hand side $\mathbf{b} \in \mathbb{K}^{M}$. As such, we define LS-CPD as

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with} \quad \mathbf{x} = \operatorname{vec}\left(\left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right] \right) \quad (2)$$

with $\mathbf{U}^{(n)} \in \mathbb{K}^{I_n \times R}$, for $1 \leq n \leq N$, and $K = \prod_{n=1}^{N} I_n$. Equation (2) can be interpreted as a decomposition of a tensor $\mathcal{X} = \text{unvec}(\mathbf{x})$ that is only *implicitly* known via the solution $\mathbf{x}$ of a linear system. Rather than K variables, the CPD structure allows the vector $\mathbf{x}$ of length K to be represented by only $R(I' - N + 1)$ variables with $I' = \sum_{n=1}^{N} I_n$. This allows one to solve the structured linear system in (2) in the underdetermined case ($M < K$), enabling a compressed-sensing-style approach [1, 27].

We show that LS-CPDs are multilinear systems of equations. Let $\mathcal{A}$ be a tensor of order $N+1$ and size $M \times I_1 \times I_2 \times \cdots \times I_N$ such that its mode-1 unfolding $\mathbf{A}_{(1)}$ equals the coefficient matrix $\mathbf{A}$, i.e., we have $\mathbf{A}_{(1)} = \mathbf{A}$. We can then rewrite (2) as a set of inner products:

$$\left\langle \mathcal{A}_m, \left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right] \right\rangle = b_m, \text{ for } 1 \leq m \leq M, \tag{3}$$

in which $\mathcal{A}_m$ is the Nth-order "horizontal slice" of $\mathcal{A}$, i.e., $\mathcal{A}_m = \mathcal{A}(m, :, :, \ldots, :)$. If $N = R = 1$, we obtain a linear system of equations and (3) reduces to:

$$\langle \mathbf{a}_m, \mathbf{x} \rangle = \mathbf{b}_m, \text{ for } 1 \leq m \leq M,$$

with $\mathbf{a}_m$ the mth row of $\mathbf{A}$. Clearly, (3) is a set of M multilinear equations. For example, consider the following simple LS-CPD with $N = 2$ and $R = 1$:

$$\mathbf{A}\text{vec}(\mathbf{u} \otimes \mathbf{v}) = \mathbf{b}, \quad \text{or equivalently,} \quad \mathbf{A}(\mathbf{v} \otimes \mathbf{u}) = \mathbf{b} \tag{4}$$

with $\mathbf{A} \in \mathbb{K}^{M \times K}$, $\mathbf{u} \in \mathbb{K}^{I}$, and $\mathbf{v} \in \mathbb{K}^{J}$ such that $K = IJ$. Equation (4) is clearly a compact form of the following set of multilinear equations (with $I = J = 2$ and $M = 4$):

$$\begin{cases} a_{111}v_1u_1 + a_{121}v_1u_2 + a_{112}v_2u_1 + a_{122}v_2u_2 = b_1, \\ a_{211}v_1u_1 + a_{221}v_1u_2 + a_{212}v_2u_1 + a_{222}v_2u_2 = b_2, \\ a_{311}v_1u_1 + a_{321}v_1u_2 + a_{312}v_2u_1 + a_{322}v_2u_2 = b_3, \\ a_{411}v_1u_1 + a_{421}v_1u_2 + a_{412}v_2u_1 + a_{422}v_2u_2 = b_4. \end{cases}$$

2.2. *LS-CPD as CPD by exploiting structure of* A

For particular types of structure on the coefficient matrix $\mathbf{A}$ in (2), the LS-CPD problem can be reformulated as a (constrained) tensor decomposition. Two examples are investigated here. First, if the coefficient matrix in (2) is a diagonal matrix $\mathbf{D} = \text{diag}(\mathbf{d})$, the LS-CPD model reduces to a *weighted* CPD of a tensor $\mathcal{B} = \text{unvec}(\mathbf{b})$ [42, 43], i.e., we have:

$$\mathcal{B} = \mathcal{D} * \left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right]$$

with $\mathcal{D}$ a tensor defined such that $\mathcal{D} = \text{unvec}(\mathbf{d})$. This model can also be used to handle missing entries by setting the corresponding weights to zero [41, 44]. It is clear that a LS-CPD reduces to a CPD if $\mathbf{D}$ is the identity matrix.

Next, we consider a coefficient matrix $\mathbf{A} \in \mathbb{K}^{M \times K}$ that has a Kronecker product structure: $\mathbf{A} = \mathbf{A}^{(N)} \otimes \mathbf{A}^{(N-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}$ with $\mathbf{A}^{(n)} \in \mathbb{K}^{J_n \times I_n}$ such that $M = \prod_{n=1}^{N} J_n$ and $K = \prod_{n=1}^{N} I_n$. Note that $\text{vec}(\left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\right]) = \left(\mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{U}^{(1)} \right) \mathbf{1}_R$. One can then show that (2) can be written as:

$$\begin{aligned} \left(\mathbf{A}^{(N)} \otimes \mathbf{A}^{(N-1)} \otimes \cdots \otimes \mathbf{A}^{(1)} \right) \left(\mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{U}^{(1)} \right) \mathbf{1}_R &= \mathbf{b}, \\ \left(\mathbf{A}^{(N)}\mathbf{U}^{(N)} \odot \mathbf{A}^{(N-1)}\mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{A}^{(1)}\mathbf{U}^{(1)} \right) \mathbf{1}_R &= \mathbf{b}, \\ \text{vec}\left(\left[\!\left[\mathbf{A}^{(1)}\mathbf{U}^{(1)}, \mathbf{A}^{(2)}\mathbf{U}^{(2)}, \ldots, \mathbf{A}^{(N)}\mathbf{U}^{(N)} \right]\!\right] \right) &= \mathbf{b}, \end{aligned}$$

which is equivalent to:

$$\left[\!\left[\mathbf{A}^{(1)}\mathbf{U}^{(1)}, \mathbf{A}^{(2)}\mathbf{U}^{(2)}, \ldots, \mathbf{A}^{(N)}\mathbf{U}^{(N)} \right]\!\right] = \mathcal{B}. \tag{5}$$

Expression (5) is a CPD with linear constraints on the factor matrices and is also known as the CANDELINC model [3, 45]; note that compatibility of the dimensions of $\mathbf{U}^{(n)}$ and $\mathbf{A}^{(n)}$ is essential to reformulate the LS-CPD as (5). Expression (5) can be computed using projection or by using a specific algorithm if the tensor $\mathcal{B}$ has missing entries [46].

2.3. *Generic uniqueness*

We show that generic uniqueness is possible when more equations than variables plus one are available. More specifically, we present a bound on M guaranteeing uniqueness of $\mathbf{x}$ in (2) for a generic $M \times K$ coefficient matrix $\mathbf{A}$. Generic uniqueness means that we have uniqueness with probability one when the entries of $\mathbf{A}$ are drawn from absolutely continuous probability density functions. We refer the reader to [35, 36, 37, 38, 39] and references therein regarding (generic) uniqueness conditions for the factor matrices in the CPD of $\mathcal{X}$. Our main result states that in order to have a generically unique solution, we need at least as many equations as free variables (compensated for scaling).

Lemma 1. Let $\mathbf{A}$ be a generic $M \times K$ matrix with $K = I_1 \cdots I_N$. Define $\mathbf{b} = \mathbf{A}\text{vec}(\mathcal{X}_0)$ with $\mathcal{X}_0$ a $I_1 \times \cdots \times I_N$ tensor with rank less than or equal to R. In that case, the solution vector $\mathbf{x}$ in (2) is unique if $M \geq (I_1 + \cdots + I_N - N + 1)R + 1$.

Proof
Consider an irreducible algebraic variety $V \in \mathbb{K}^K$ of dimension d_V. It is known that a generic plane of dimension less than or equal to $K - d_V - 1$ does not intersect with V [47]. It is clear that a generic plane of dimension $K - M$ can be interpreted as the null space of a generic $M \times K$ matrix. Hence, if $\mathbf{A}$ is a generic $M \times K$ matrix, $\mathbf{v}_0 \in V$ and $\mathbf{b} := \mathbf{A}\mathbf{v}_0$, then the problem

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \text{with} \quad \mathbf{x} \in V \tag{6}$$

has a unique solution whenever $K - M \leq K - d_V - 1$ or $M \geq d_V + 1$. We interpret (2) as (6) in which V is the Zariskii closure of the set of $I_1 \times \cdots \times I_N$ tensors whose rank does not exceed R. Since a generic tensor in V can be parameterized with at most $(I_1 + \cdots + I_N - N + 1)R$ parameters, it follows that $d_V \geq (I_1 + \cdots + I_N - N + 1)R$. Hence, a solution vector $\mathbf{x}$ in (2) is unique if $M \geq d_V + 1 \geq (I_1 + \cdots + I_N - N + 1)R + 1$. □

3. ALGORITHMS

First, we derive an algebraic method to solve a LS-CPD with $R = 1$ in Subsection 3.1. Next, we develop an optimization-based algorithm for general LS-CPDs in Subsection 3.2.

3.1. *Algebraic computation*

We present an algebraic method to solve (8) in Subsections 3.1.1 and 3.1.2. The derivation is closely related to [48]. Importantly, all steps can be performed by means of conventional linear algebra. The overall algebraic procedure is summarized in Algorithm 1. This method finds the exact solution in the case of exact problems but can also be used to obtain a good initialization for optimization-based methods in the case of perturbed problems.

It is well-known that a tensor $\mathcal{X}$ of order N has rank one if and only if all its matrix unfoldings have rank one, i.e., we have:

$$\text{rank}\left(\mathbf{X}_{(n)}\right) = R = 1, \quad \text{for } 1 \leq n \leq N. \tag{7}$$

In this particular case a solution $\mathbf{x}$ to (2) is also a solution of

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with} \quad \mathbf{x} = \text{vec}\left(\mathcal{X}\right), \text{ where } \mathcal{X} \text{ satisfies (7)} \tag{8}$$

and a solution to (8) is also a solution to (2). The case $R > 1$ relates to linear systems with a MLSVD constrained solution, see [49]. We can compute a solution of (2) *algebraically* in two steps as follows. First, we use (8) to recover $\mathcal{X}$. Next, we compute the (exact) rank-1 CPD of $\mathcal{X}$.

3.1.1. Trivial case. Assume that the solution of the unstructured linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is unique, i.e., the null space of the extended matrix $\begin{bmatrix}\mathbf{A} \ \mathbf{b}\end{bmatrix}$ is one-dimensional. In that case, we can compute the solution to (8) by ignoring the multilinear structure (7), i.e., we solve for $\mathbf{x}$ and subsequently compute a CPD of $\mathcal{X} = \text{unvec}(\mathbf{x})$. Clearly, the tensor $\mathcal{X}$ is unique if $\mathbf{b} \neq \mathbf{0}$ or is unique up to a scaling factor if $\mathbf{b} = \mathbf{0}$. This approach is the naive method that we have mentioned in Section 1.

3.1.2. Reduction of the general case to the trivial case. We explain how to find a solution of (8) when the dimension of the null space of $\begin{bmatrix}\mathbf{A} \ \mathbf{b}\end{bmatrix}$ is larger than one, e.g., when $\mathbf{A}$ is a fat matrix or rank-deficient. We limit ourselves to the case where $\mathbf{b} \neq \mathbf{0}$, which implies that the dimension of the null space of $\mathbf{A}$ is at least one. It can be shown that the case where $\mathbf{b} = \mathbf{0}$ follows in a similar way.

Let $\mathbf{f}^{(0)}$ be a particular solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ and let the vectors $\mathbf{f}^{(l)}$, for $1 \leq l \leq L$, form a basis of the L-dimensional null space of $\mathbf{A}$. Consider the tensorized versions of $\mathbf{f}^{(l)}$ denoted by $\mathcal{F}^{(l)} \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$, for $0 \leq l \leq L$. In order to solve (8), we have to find values c_l, for $1 \leq l \leq L$, such that $\mathcal{X} = \mathcal{F}^{(0)} + c_1\mathcal{F}^{(1)} + \cdots + c_L\mathcal{F}^{(L)}$ satisfies (7), i.e., we have that:

$$\text{rank}\left(\mathbf{X}_{(n)}\right) = \text{rank}\left(\mathbf{F}_{(n)}^{(0)} + c_1\mathbf{F}_{(n)}^{(1)} + \cdots + c_L\mathbf{F}_{(n)}^{(L)}\right) = 1, \quad 1 \leq n \leq N. \tag{9}$$

We can reformulate (9) as the following LS-CPD problem:

$$\tilde{\mathbf{A}}(\mathbf{c} \otimes \mathbf{c}) = \mathbf{0} \quad \text{with} \quad \mathbf{c} = [1 \; c_1 \; \cdots \; c_L]^T \tag{10}$$

with, as explained below, $\tilde{\mathbf{A}}$ constructed from the tensors $\mathcal{F}^{(l)}$ such that each row of $\tilde{\mathbf{A}}$ is a vectorized $L+1 \times L+1$ symmetric matrix. We make the assumption that the intersection of the null space of $\tilde{\mathbf{A}}$ with the subspace of vectorized symmetric matrices is one-dimensional. In practice this assumption is satisfied when the difference between the number of rows and columns of $\tilde{\mathbf{A}}$ is sufficiently large. In that case, the solution $\mathbf{c} \otimes \mathbf{c}$ is unique and can be computed as explained in Subsection 3.1.1 from which $\mathbf{c}$ can be easily recovered.

We explain the construction of $\tilde{\mathbf{A}}$ in more detail. First, partition $\tilde{\mathbf{A}}$ as follows:

$$\tilde{\mathbf{A}} = \begin{bmatrix}\tilde{\mathbf{A}}^{(1)^{\mathrm{T}}} & \tilde{\mathbf{A}}^{(2)^{\mathrm{T}}} & \cdots & \tilde{\mathbf{A}}^{(N)^{\mathrm{T}}}\end{bmatrix}^{\mathrm{T}}, \tag{11}$$

where the matrices $\tilde{\mathbf{A}}^{(n)}$ correspond to the constraints in (9). Consider the following definition.

Definition 3. The second compound matrix $\mathcal{C}_2(\mathbf{F})$ of an $I \times J$ matrix $\mathbf{F}$, with $2 \leq \min(I, J)$, is a $C_I^2 \times C_j^2$ matrix containing all 2×2 minors of $\mathbf{F}$ ordered lexicographically [50].

It is well-known that the following algebraic identity holds for any 2×2 matrices $\mathbf{F}^{(0)}, \ldots, \mathbf{F}^{(L)}$ and values $c_0, \ldots, c_L$:

$$\begin{aligned}&\det(c_0\mathbf{F}^{(0)} + c_1\mathbf{F}^{(1)} + \cdots + c_L\mathbf{F}^{(L)}) = \\ &\frac{1}{2}\sum_{j_1,j_2=1}^{L+1} c_{j_1}c_{j_2}\left[\det(\mathbf{F}^{(j_1)} + \mathbf{F}^{(j_2)}) - \det(\mathbf{F}^{(j_1)}) - \det(\mathbf{F}^{(j_2)})\right].\end{aligned} \tag{12}$$

By applying (12) to each 2×2 submatrix of $c_0\mathbf{F}_{(n)}^{(0)} + c_1\mathbf{F}_{(n)}^{(1)} + \cdots + c_L\mathbf{F}_{(n)}^{(L)}$, we obtain:

$$\begin{aligned}&\mathcal{C}_2\left(c_0\mathbf{F}_{(n)}^{(0)} + c_1\mathbf{F}_{(n)}^{(1)} + \cdots + c_L\mathbf{F}_{(n)}^{(L)}\right) = \\ &\frac{1}{2}\sum_{j_1,j_2=1}^{L+1} c_{j_1}c_{j_2}\left[\mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_1)} + \mathbf{F}_{(n)}^{(j_2)}\right) - \mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_1)}\right) - \mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_2)}\right)\right].\end{aligned} \tag{13}$$

Condition (9) states that all 2×2 minors of the matrix $\mathbf{X}_{(n)} = \mathbf{F}_{(n)}^{(0)} + c_1\mathbf{F}_{(n)}^{(1)} + \cdots + c_L\mathbf{F}_{(n)}^{(L)}$ are zero, or, in other words, we have that $\mathcal{C}_2\left(\mathbf{X}_{(n)}\right) = \mathbf{0}$. Hence, according to (13), we have:

$$\sum_{j_1,\ldots,j_{R+1}=1}^{L+1} c_{j_1}c_{j_2}\left[\mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_1)} + \mathbf{F}_{(n)}^{(j_2)}\right) - \mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_1)}\right) - \mathcal{C}_2\left(\mathbf{F}_{(n)}^{(j_2)}\right)\right], \quad \text{with} \quad c_0 = 1,$$

which is equivalent to

$$\tilde{\mathbf{A}}^{(n)}(\mathbf{c}\otimes\mathbf{c}) = \mathbf{0}, \quad \text{with} \quad \mathbf{c} = [1\ c_1\ \dots\ c_L]^T,\ 1 \leq n \leq N,$$

in which $\tilde{\mathbf{A}}^{(n)}$ has size $C^2_{I_n} C^2_{KI_n^{-1}} \times (L+1)^2$ and is defined column-wise as follows:

$$\tilde{\mathbf{a}}^{(n)}_{j_2+(L+1)(j_1-1)} = \operatorname{vec}\left(\mathcal{C}_2\left(\mathbf{F}^{(j_1)}_{(n)} + \mathbf{F}^{(j_2)}_{(n)}\right) - \mathcal{C}_2\left(\mathbf{F}^{(j_1)}_{(n)}\right) - \mathcal{C}_2\left(\mathbf{F}^{(j_2)}_{(n)}\right)\right). \tag{14}$$

In Algorithm 1, the number of rows of $\tilde{\mathbf{A}}$ should be at least the dimension of the subspace of the symmetric $L+1 \times L+1$ matrices minus one. Hence, a *necessary* condition for the algebraic computation is that $\sum_{n=1}^{N} C^2_{I_n} C^2_{KI_n^{-1}} \geq (L+1)(L+2)/2 - 1 \geq (K-M+1)(K-M+2) - 1$. The computational complexity of Algorithm 1 is dominated by the construction of $\tilde{\mathbf{A}}$.

Algorithm 1: Algebraic algorithm to solve $\mathbf{Ax} = \mathbf{b}$ in which $\mathbf{x}$ has a rank-1 CPD structure

```
  Input: A and non-zero b
  Output: $\{\mathbf{u}^{(n)}\}_{n=1}^N$
1 Find $\mathbf{f}^{(0)} \in \mathbb{K}^K$ such that $\mathbf{Af}^{(0)} = \mathbf{b}$;
2 Find $\mathbf{f}^{(l)} \in \mathbb{K}^K$, for $1 \leq l \leq L$, that form a basis for $\operatorname{null}(\mathbf{A}) \in \mathbb{K}^{K\times L}$;
3 Reshape $\mathbf{f}^{(l)}$ into $I_1 \times I_2 \times \cdots \times I_N$ tensors $\mathcal{F}^{(l)}$, for $0 \leq l \leq L$;
4 Construct $\tilde{\mathbf{A}}^{(1)}, \dots, \tilde{\mathbf{A}}^{(N)}$ as in (14) and construct $\tilde{\mathbf{A}}$ as in (11);
5 Find a non-zero solution of $\tilde{\mathbf{A}}\tilde{\mathbf{c}} = 0$ (if $\operatorname{null}(\tilde{\mathbf{A}})$ is one-dimensional);
6 Find the vector $\mathbf{c} = [1\ c_1\ \dots\ c_L]^T$ such that $\mathbf{c}\otimes\mathbf{c}$ is proportional to $\tilde{\mathbf{c}}$;
7 Construct $\mathcal{X} = \mathcal{F}^{(0)} + c_1\mathcal{F}^{(1)} + \cdots + c_L\mathcal{F}^{(L)}$;
8 Compute the rank-1 CPD of $\mathcal{X} = [\![\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(N)}]\!]$;
```

3.2. *Optimization-based methods*

In this subsection, we solve the LS-CPD problem in (2) via a least-squares approach, leading to the following optimization problem:

$$\min_{\mathbf{z}} f = \frac{1}{2}\,||\mathbf{r}(\mathbf{z})||^2_{\text{F}} \tag{15}$$

in which the residual $\mathbf{r}(\mathbf{z}) \in \mathbb{K}^M$ is defined as

$$\mathbf{r}(\mathbf{z}) = \mathbf{A}\operatorname{vec}\left([\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots, \mathbf{U}^{(N)}]\!]\right) - \mathbf{b}$$

where we have concatenated the optimization variables in a vector $\mathbf{z} \in \mathbb{K}^{RI'}$ with $I' = \sum_{n=1}^{N} I_n$ as follows: $\mathbf{z} = \left[\operatorname{vec}\left(\mathbf{U}^{(1)}\right);\ \operatorname{vec}\left(\mathbf{U}^{(2)}\right);\ \cdots;\ \operatorname{vec}\left(\mathbf{U}^{(N)}\right)\right]$. To solve the NLS problem (15), we use the Gauss–Newton (GN) method which is a particular nonlinear least squares (NLS) algorithm [51]. The latter requires expressions for the objective function, gradient, Gramian, and Gramian-vector product. Although we focus on the GN method, the expressions can be used to implement other NLS algorithms as well as quasi-Newton (qN) algorithms. In order to implement the methods we use the complex optimization framework from [17, 19] which provides implementations for qN and NLS algorithms as well as line search, plane search, and trust region methods.

The GN method solves (15) by linearizing the residual vector $\mathbf{r}(\mathbf{z})$ and solving a least-squares problem in each iteration k:

$$\min_{\mathbf{p}_k} \frac{1}{2}\,||\mathbf{r}(\mathbf{z}_k) + \mathbf{J}_k\mathbf{p}_k||^2_{\text{F}} \quad \text{s.t.} \quad ||\mathbf{p}_k|| \leq \Delta_k$$

with step $\mathbf{p}_k = \mathbf{z}_{k+1} - \mathbf{z}_k$ and trust-region radius Δ_k [51]. The Jacobian $\mathbf{J} = \partial\mathbf{r}(\mathbf{z})/\partial\mathbf{z} \in \mathbb{K}^{M\times RI'}$ is evaluated at $\mathbf{z}_k$. The exact solution to the linearized problem is given by the normal equations:

$$\mathbf{J}_k^{\text{H}}\mathbf{J}_k\mathbf{p}_k = -\mathbf{J}_k^{\text{H}}\mathbf{r}(\mathbf{z}_k)$$

$$\mathbf{H}_k \mathbf{p}_k = -\overline{\mathbf{g}}_k. \tag{16}$$

In the NLS formulation $\mathbf{H} \in \mathbb{K}^{RI' \times RI'}$ is the Gramian of the Jacobian which is an approximation to the Hessian of f [51]. The conjugated gradient $\overline{\mathbf{g}} \in \mathbb{K}^{RI'}$ is defined as $\overline{\mathbf{g}} = (\partial f / \partial \mathbf{z})^{\mathrm{H}}$. The normal equations are solved inexactly using several preconditioned conjugated gradient (CG) iterations to reduce the computational complexity. After solving (16), the variables can be updated as $\mathbf{z}_{k+1} = \mathbf{z}_k + \mathbf{p}_k$. While a dogleg trust-region method is used here, other updating methods such as line and plane search can be used as well, see [51] for details. In the remainder of this subsection we derive the required expressions for the GN method summarized in Algorithm 2.

Algorithm 2: LS-CPD using Gauss–Newton with dogleg trust region

```
  Input: $\mathbf{A}$, $\mathbf{b}$, and $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$
  Output: $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$
1 while not converged do
2     Compute gradient $\mathbf{g}$ using (17).
3     Use PCG to solve $\mathbf{Hp} = -\overline{\mathbf{g}}$ for $\mathbf{p}$ using Gramian-vector products as in (20) using a
        (block)-Jacobi preconditioner, see Section 3.2.3.
4     Update $\mathbf{U}^{(n)}$, for $1 \leq n \leq N$, using dogleg trust region from $\mathbf{p}$, $\mathbf{g}$, and function
        evaluation (15).
5 end
```

3.2.1. Objective function We evaluate the objective function f by taking the sum of squared entries of the residual $\mathbf{r}(\mathbf{z})$. The latter can be computed by using contractions as follows:

$$\mathbf{r}(\mathbf{z}) = \sum_{r=1}^{R} \mathcal{A} \cdot_2 \mathbf{u}_r^{(1)^{\mathrm{T}}} \cdot_3 \mathbf{u}_r^{(2)^{\mathrm{T}}} \cdots \cdot_{N+1} \mathbf{u}_r^{(N)^{\mathrm{T}}} - \mathbf{b}.$$

3.2.2. Gradient We partition the gradient as $\mathbf{g} = \left[\mathbf{g}^{(1,1)};\ \mathbf{g}^{(1,2)};\ \ldots;\ \mathbf{g}^{(1,R)};\ \mathbf{g}^{(2,1)};\ \ldots;\ \mathbf{g}^{(N,R)}\right]$ in which the subgradients $\mathbf{g}^{(n,r)} \in \mathbb{K}^{I_n}$ are defined as

$$\mathbf{g}^{(n,r)} = \left(\mathbf{J}^{(n,r)}\right)^{\mathrm{T}} \overline{\mathbf{r}(\mathbf{z})} \tag{17}$$

in which $\mathbf{J}^{(n,r)}$ is defined as

$$\mathbf{J}^{(n,r)} = \frac{\partial \mathbf{r}(\mathbf{z})}{\mathbf{u}_r^{(n)}} = \left(\mathcal{A} \cdot_2 \mathbf{u}_r^{(1)^{\mathrm{T}}} \cdots \cdot_n \mathbf{u}_r^{(n-1)^{\mathrm{T}}} \cdot_{n+2} \mathbf{u}_r^{(n+1)^{\mathrm{T}}} \cdots \cdot_{N+1} \mathbf{u}_r^{(N)^{\mathrm{T}}}\right)_{(1)}. \tag{18}$$

Expression (18) equals the (n, r)th sub-Jacobian, using a similar partitioning for $\mathbf{J}$. The sub-Jacobians require a contraction in all but the first and nth mode and are precomputed.

3.2.3. Gramian of the Jacobian We partition the Gramian $\mathbf{H}$ into a grid of $NR \times NR$ blocks $\mathbf{H}^{(n,r,m,l)}$ with $1 \leq n, m \leq N$ and $1 \leq r, l \leq R$. Each block $\mathbf{H}^{(n,r,m,l)}$ is defined by:

$$\mathbf{H}^{(n,m,r,l)} = \left(\mathbf{J}^{(n,r)}\right)^{\mathrm{H}} \mathbf{J}^{(n,r)}, \tag{19}$$

using the sub-Jacobians in (18). Expression (19) approximates the second-order derivative of f with respect to the variables $\mathbf{u}_r^{(n)}$ and $\mathbf{u}_l^{(m)}$.

As preconditioned CG (PCG) is used, only matrix vector-products are needed. The full Gramian is never constructed because one can exploit the block structure to compute fast matrix-vector products. Hence, in each iteration we compute Gramian-vector products of the form $\mathbf{z} = \mathbf{J}^{\mathrm{H}}\mathbf{J}\mathbf{y}$ as

Table I. The per-iteration computational complexity of the NLS algorithm for LS-CPD is dominated by the computation of the Jacobian. The algorithm uses a trust region approach to determine the update, which requires $\mathrm{it}_{\mathrm{TR}}$ additional evaluations of the objective function.

	Calls per iteration	Complexity
Objective function	1 + $\mathrm{it}_{\mathrm{TR}}$	$\mathcal{O}(MRI^N)$
Jacobian	1	$\mathcal{O}(MRNI^N)$
Gradient	1	$\mathcal{O}(MRNI)$
Gramian	1	$\mathcal{O}(MRNI^2)$
Gramian-vector	$\mathrm{it}_{\mathrm{CG}}$	$\mathcal{O}(MRNI)$

follows:

$$\mathbf{z}^{(n,r)} = \left(\mathbf{J}^{(n,r)}\right)^{\mathrm{H}} \left(\sum_{n=1}^{N}\sum_{r=1}^{R} \mathbf{J}^{(n,r)}\mathbf{y}^{(n,r)}\right), \text{ for } 1 \leq n \leq N, \text{ and } 1 \leq r \leq R, \tag{20}$$

in which we partitioned $\mathbf{z}$ and $\mathbf{y}$ in a similar way as before.

In this paper, we use either a block-Jacobi or Jacobi preconditioner to improve convergence or reduce the number of CG iterations. In the former case, we compute the $(I_n \times I_n)$ Gramians $\mathbf{H}^{(n,n,r,r)}$, for $1 \leq n \leq N$ and $1 \leq r \leq R$, *and* their inverses in each iteration. Combining both operations leads to a per-iteration computational complexity of $\mathcal{O}(MI_n^2 + I_n^3)$ which is relatively expensive, especially for large problems. One can instead use a Jacobi preconditioner which uses a diagonal approximation of the Gramian and, consequently, an inexpensive computation of the inverse. The diagonal elements are computed as the inner product $\mathbf{J}_{i_n}^{(n,r)^{\mathrm{H}}}\mathbf{J}_{i_n}^{(n,r)}$, for $1 \leq i_n \leq I_n$, leading to an overall computational complexity of $\mathcal{O}(MI_n + I_n)$ which is relatively inexpensive. We compare the effectiveness of the Jacobi and block-Jacobi preconditioner in Section 4.

3.2.4. Complexity We report the per-iteration complexity of the NLS algorithm for LS-CPD in Table I. For simplicity, we assume that $I_1 = I_2 = \cdots = I_N = I$ in (2). Clearly the computational complexity is dominated by the computation of the sub-Jacobians in (18). The computational complexity can be reduced by computing the contractions in (18) as efficiently as possible. Note that the evaluation of the objective function is a factor N less expensive.

3.2.5. Efficient contractions The per-iteration complexity of the NLS algorithm is relatively high, however, only a few iterations are often necessary in order to obtain convergence. One can reduce the overall computation time of the algorithm by reducing the computational cost per iteration or the number of iterations. We have shown that the computation of the Jacobians is relatively expensive, see Table I. Computing the sub-Jacobians requires the sequential computation of $N-1$ contractions of the form $\mathcal{A} \cdot_n \mathbf{x}^{(n)^{\mathrm{T}}} \in \mathbb{K}^{I_1\times\cdots\times I_{n-1}\times I_{n+1}\times\cdots\times I_N}$ which are defined as

$$(\mathcal{A} \cdot_n \mathbf{x}^{(n)^{\mathrm{T}}})_{i_1\ldots i_{n-1}i_{n+1}\ldots i_N} = \sum_{i_n=1}^{I_n} a_{i_1\ldots i_N} x_{i_n}^{(n)} \tag{21}$$

with $\mathcal{A} \in \mathbb{K}^{I_1\times\cdots\times I_N}$ and a vector $\mathbf{x}^{(n)} \in \mathbb{K}^{I_n}$. Clearly, it is important to perform the contractions as efficiently as possible to reduce the per-iteration complexity of the algorithm. Note that the computation of the contractions can be done in a memory-efficient way by computing the contractions sequentially via the matrix unfoldings and permuting the first mode of $\mathcal{A}$ to the middle. This approach guarantees that $\mathcal{A}$ is permuted in memory at most once.

One way to compute contractions efficiently is by exploiting all possible structure of the coefficient tensor $\mathcal{A}$ in (21). For example, if $\mathbf{A}$ is the identity matrix, the LS-CPD problem reduces to a CPD. In that case, the Gramians and their inverses can be computed efficiently by storing the Gramians of the factor matrices, see [18]. If $\mathbf{A}$ has a Kronecker product structure, the LS-CPD

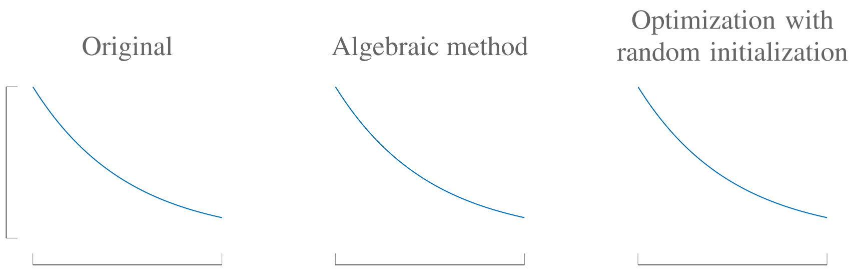

Figure 2. Our algebraic method and optimization-based method (with random initialization) can perfectly reconstruct an exponential solution vector in the noiseless case.

problem reduces to a CANDELINC model, as shown in Subsection 2.2, which can be computed efficiently in both the dense and sparse case, see [3, 46].

Let us illustrate how we can compute efficient contractions in the case of a sparse $\mathcal{A}$. Assume we have a vector $\mathbf{a} \in \mathbb{K}^M$ containing the M non-zero values of $\mathcal{A}$ and corresponding index sets $S_m = \{i_1^{(m)}, i_2^{(m)}, \ldots, i_N^{(m)}\}$, for $1 \leq m \leq M$. We can then compute (21) efficiently as $\mathbf{w} = \mathbf{v} * \mathbf{u}^{(n)}_{S'_n}$ with $S'_n = \{i_n^{(1)}, i_n^{(2)}, \ldots, i_n^{(M)}\}$, for $1 \leq n \leq N$. As such, we obtain a new index-value pair with $\mathbf{w} \in \mathbb{K}^M$ and $R_m = S_m \backslash i_n^{(m)}$ for $1 \leq m \leq M$.

4. NUMERICAL EXPERIMENTS

First, two proof-of-concept experiments are conducted to illustrate the algebraic and optimization-based methods in Subsection 4.1. Next, we compare accuracy and time complexity of the naive, algebraic, and NLS method in Subsection 4.2. We also compare algebraic and random initialization methods for the NLS algorithm in Subsection 4.3. In Subsection 4.4, we compare the Jacobi and block-Jacobi preconditioner for the NLS algorithm. All computations are done with Tensorlab [20]. We define the relative error $\epsilon_{\mathbf{x}}$ as the relative difference in Frobenius norm $\|\mathbf{x} - \hat{\mathbf{x}}\|_{\mathrm{F}} / \|\mathbf{x}\|_{\mathrm{F}}$ with $\hat{\mathbf{x}}$ an estimate for $\mathbf{x}$. We use factor matrices in which the elements are drawn from the standard normal distribution, unless stated otherwise, to generate tensors. In that case the factor matrices are well-conditioned because the expected angle between the factor vectors is 90° for large matrices. The coefficient matrices $\mathbf{A}$ are constructed in a similar way, unless stated otherwise. We use i.i.d. Gaussian noise to perturb the entries of a tensor. The noise is scaled to obtain a given signal-to-noise ratio (SNR) (with the signal equal to the noiseless tensor). If we consider a perturbed LS-CPD problem, we perturb the right-hand side in (2), unless stated otherwise.

4.1. Proof-of-concept

We give two simple proof-of-concept experiments, illustrating our algorithms for linear systems with a solution that can represented or well approximated by a low-rank model. First, consider a LS-CPD with a solution $\mathbf{x} \in \mathbb{K}^K$ that is constrained to be an exponential, i.e., $x_k = e^{-2k}$ evaluated in K equidistant samples in $[0, 1]$. It is known that sums of exponentials can be exactly represented by low-rank tensors [9, 10, 52]. In this case, the corresponding tensor $\mathcal{X} = \text{unvec}(\mathbf{x})$ has rank one ($R = 1$) [52]. We choose $N = 3$, $I_1 = I_2 = I_3 = I = 4$, $K = I^3 = 64$, and $M = 34$. We compute a solution using the algebraic method and the NLS algorithm with random initialization. Perfect reconstruction of the exponential is obtained with both methods as shown in Figure 2.

In the previous experiment the solution vector could be exactly represented by a low-rank tensor. Many signals, such as as Gaussians, rational functions, and periodic signals, can also be well approximated by a low-rank model [9, 10]. In this experiment, we consider a LS-CPD of which

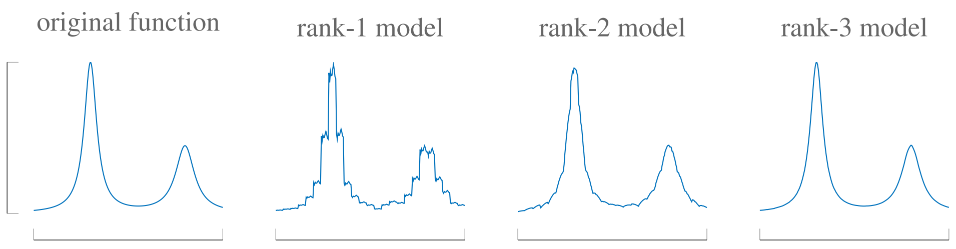

Figure 3. Our optimization-based method, initialized with the naive approach, can reconstruct the rational solution vector in the noiseless case. Increasing the rank of the CPD model improves the accuracy of the solution. For example, the rank-3 model is almost indistinguishable from the original function.

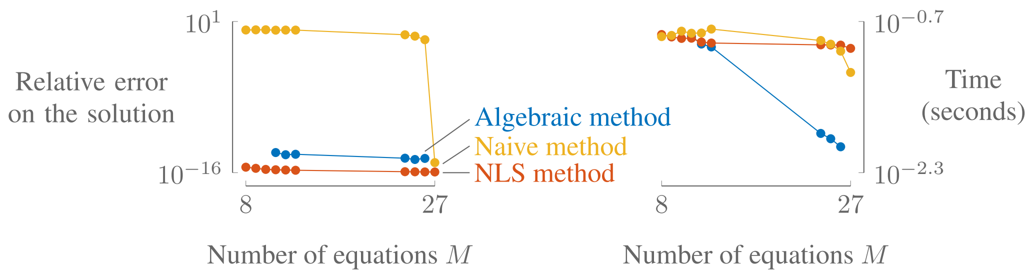

Figure 4. The naive method fails for an underdetermined LS-CPD while the NLS and algebraic method both perform well. The computational complexity of the algebraic method is much higher than the other two methods, especially for the highly underdetermined case (i.e., M close to the number of free variables).

the solution vector is a rational function:

$$x_k = \frac{1}{(k-0.3)^2 + 0.04^2} + \frac{1}{(k-0.8)^2 + 0.06^2},$$

evaluated at K equidistant samples in $[0, 1]$. We take $N = 2$, $I_1 = 10$, $I_2 = 25$, $K = I_1 I_2 = 250$, and $M = 350$. The NLS algorithm with random initialization is used for $R = \{1, 2, 3\}$ to compute a solution. In Figure 3, one can see that the accuracy of the approximation increases when using higher rank values.

4.2. *Comparison of methods*

We compare the algebraic method in Algorithm 1, the NLS method in Algorithm 2, and the naive method, i.e., the trivial case of the algebraic method, see Section 3.1. Remember that the latter can be computed by first solving the unstructured system and afterwards fitting the CPD structure on the obtained solution. Consider a LS-CPD with $N = 3$, $I_1 = I_2 = I_3 = I = 3$, $R = 1$, $K = I^3 = 27$. We choose $M^{(\min)} \leq M \leq K$ with $M^{(\min)} = 8$ which equals the minimal value of M to obtain a (generically) unique solution according to Lemma 1. We report the median relative error on the solution $\epsilon_{\mathbf{x}}$ and the time across 100 experiments in Figure 4. The naive method fails when $M < K$ because we solve an underdetermined linear system, resulting in a non-unique solution due to the non-emptiness of the null space of $\mathbf{A}$. The algebraic method works well, but fails if $M \leq 10$ because then the dimension of the null space of $\tilde{\mathbf{A}}$ in (10) is larger than one, see Subsection 3.1. For $M = K$, the algebraic method coincides with the naive method. The NLS method performs well for all M using five random initializations. Note that NLS typically needs many random initializations when M is close to $M^{(\min)}$. The accuracy is slightly higher than the algebraic method. The computational cost of the algebraic methods increases when M decreases because $\tilde{\mathbf{A}}$ in (10) depends quadratically on L which is the dimension of the null space $\mathbf{A}$.

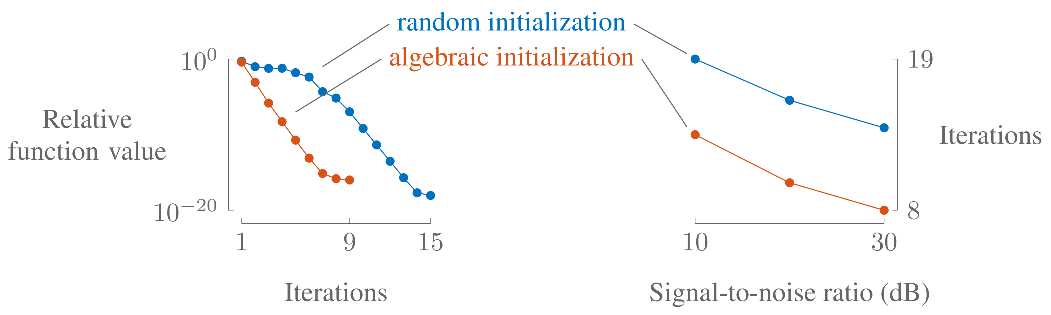

Figure 5. By initializing the NLS algorithm with the algebraic solution instead of using a random initialization, fewer iterations are needed to achieve convergence.

4.3. *Initialization methods*

The algebraic method in Algorithm 1 finds the exact solution in the case of exact problems. In the case of perturbed problems, however, the solution can be used to obtain a good initialization for optimization-based methods such as the NLS algorithm from Subsection 3.2. Often the algebraic solution provides a better starting value for optimization-based algorithms than a random initialization. We illustrate this for an underdetermined LS-CPD of the form (2) with $N = 3$, $R = 2$, $I_1 = I_2 = I_3 = I = 4$, $K = I^3 = 64$, and $M = 60$. We compute a solution using the NLS algorithm with random and algebraic initialization. In Figure 5, we report the median number of iterations across 20 experiments for several values of the SNR; we also show the convergence plot for 20 dB SNR on the left. By starting the NLS algorithm from the algebraic solution instead of using a random initialization, we need fewer iterations to achieve convergence. Importantly, the algebraic method can still find a solution in the noisy case but the accuracy is typically low. Optimization-based methods such as the NLS algorithm can use this solution as an initialization and improve the accuracy.

4.4. *Preconditioner*

The overall computation time of the NLS algorithm can be reduced by reducing the computational cost per iteration or the number of iterations. Remember that we solve the normal equations in the NLS algorithm inexactly via a number of PCG iterations. Good preconditioning is essential to lower the number of CG iterations and, consequently, reduce the per-iteration complexity of the NLS algorithm. Here, we compare the Jacobi and block-Jacobi preconditioner (PC), see Section 3.

Consider a LS-CPD problem with $N = 2$, $R = 3$, $I_1 = 250$, $I_2 = 10$, $K = I_1 I_2 = 2500$. We consider three different scenarios: the highly underdetermined case ($M = R(I_1 + I_2) + 5 = 785$), the underdetermined case ($M = 1.5R(I_1 + I_2) = 1170$), and the square case $M = K = 2500$. We simulate a typical iteration of the NLS algorithm as follows. We compute the Gramian $\mathbf{H}$ and the gradient g for random factor matrices $\mathbf{U}^{(n)}$ and solve the normal equations in (16) using PCG until convergence (i.e., up to a tolerance of 10^{-6}). In Table II we report the average number of CG iterations across 50 experiments when using no PC, the Jacobi PC, and the block-Jacobi PC for the three different scenarios. In this experiment, the block-Jacobi preconditioner reduces the number of CG iterations more than the Jacobi preconditioner, especially for the highly underdetermined case. In the square case, both PCs have only slightly different performance, but the Jacobi PC is preferred because of its lower computational complexity.

5. APPLICATIONS

LS-CPDs provide a generic framework that can be used in a wide range of applications. In this paper, we illustrate with three applications in classification, multilinear algebra, and signal processing, respectively. First, LS-CPDs are used for tensor-based face recognition in Subsection 5.1. The

Table II. Both PCs reduce the number of CG iterations in the underdetermined and square case. In the strongly underdetermined case only the block-Jacobi PC can reduce the number of CG iterations. We reported the average (and standard deviation of the) number of CG iterations across 50 experiments

Scenario	No PC	Jacobi PC	block-Jacobi PC
highly underdetermined	780 (0)	743 (56)	599 (97)
underdetermined	141 (24)	65 (10)	53 (9)
square	66 (14)	30 (6)	27 (6)

technique is very generic and can be used for other classification tasks as well. For example, a similar method was used in [34] for irregular heartbeat classification in the analysis of electrocardiogram data. Next, it is shown in Subsection 5.2 that tensors that have particular multilinear singular values can be constructed using LS-CPDs. Finally, in Subsection 5.3, LS-CPDs are used for the blind deconvolution of constant modulus signals.

5.1. Tensor-based face recognition using LS-CPDs

LS-CPDs can be used for generic classification tasks which is illustrated here using tensor-based face recognition [29, 32]. Consider a set of matrices of size $M_x \times M_y$ representing the facial images of J persons, taken under I different illumination conditions. All *vectorized* images of length $M = M_x M_y$ are stacked in a third-order tensor $\mathcal{D} \in \mathbb{K}^{M\times I\times J}$ with modes pixels (px) $\times$ illumination (i) $\times$ persons (p). Next, we perform a multilinear analysis by computing a (truncated) MLSVD of the tensor $\mathcal{D}$, i.e., we have:

$$\mathcal{D} \approx \mathcal{S} \cdot_1 \mathbf{U}_{\mathrm{px}} \cdot_2 \mathbf{U}_{\mathrm{i}} \cdot_3 \mathbf{U}_{\mathrm{p}}.$$

with $\mathbf{U}_{\mathrm{px}}$, $\mathbf{U}_{\mathrm{i}}$, and $\mathbf{U}_{\mathrm{p}}$ forming an orthonormal basis for the pixel, illumination, and person mode, respectively. The core tensor $\mathcal{S}$ explains the interaction between the different modes. The vectorized image $\mathbf{d} \in \mathbb{K}^M$ for a *particular* illumination i and person p satisfies:

$$\mathbf{d} = (\mathcal{S} \cdot_1 \mathbf{U}_{\mathrm{px}}) \cdot_2 \mathbf{c}_{\mathrm{i}}^{\mathrm{T}} \cdot_3 \mathbf{c}_{\mathrm{p}}^{\mathrm{T}} \tag{22}$$

with $\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}$ and $\mathbf{c}_{\mathrm{p}}^{\mathrm{T}}$ rows of $\mathbf{U}_{\mathrm{i}}$ and $\mathbf{U}_{\mathrm{p}}$ and $\mathbf{U}_{\mathrm{p}}$ acting as a database. The mode-1 unfolding of (22) is a LS-CPD of the form (4):

$$\mathbf{d} = \mathbf{U}_{\mathrm{px}}\mathbf{S}_{(1)}(\mathbf{c}_{\mathrm{p}} \otimes \mathbf{c}_{\mathrm{i}}).$$

Consider a previously unknown image $\mathbf{d}^{(\mathrm{new})}$ of a person that is included in the database. Classification or recognition of this person corresponds to finding the coefficient vector $\mathbf{c}_{\mathrm{p}}$, i.e., we solve a LS-CPD of the form:

$$\mathbf{d}^{(\mathrm{new})} = \mathbf{U}_{\mathrm{px}}\mathbf{S}_{(1)}\left(\mathbf{c}_{\mathrm{p}}^{(\mathrm{new})} \otimes \mathbf{c}_{\mathrm{i}}^{(\mathrm{new})}\right),$$

resulting into estimates $\tilde{\mathbf{c}}_{\mathrm{p}}^{(\mathrm{new})}$ and $\tilde{\mathbf{c}}_{\mathrm{i}}^{(\mathrm{new})}$. The coefficient vector for the person dimension $\tilde{\mathbf{c}}_{\mathrm{p}}^{(\mathrm{new})}$ is compared with the rows of $\mathbf{U}_{\mathrm{p}}$ using the Frobenius norm of the difference (after fixing scaling and sign invariance). We then classify the person in the image according to the label of the closest match.

Let us illustrate the above strategy for the extended YaleB dataset[†]. This real-life dataset consists of cropped facial images of 39 persons in 64 illumination conditions. We remove illumination conditions for which some of the images are missing and retain one of the conditions as test data, resulting into $I = 56$ conditions. We vectorize each image of 51×58 pixels into a vector of length $M = 2958$ for $J = 37$ persons. The resulting data tensor $\mathcal{D}$ has size $2958 \times 56 \times 37$. We compute the MLSVD of $\mathcal{D}$ using a randomized algorithm called `mlsvd_rsi`, which is faster than non-randomized algorithms but achieves similar accuracy [53]. We compress the pixel mode to reduce

[†]Available from `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`.

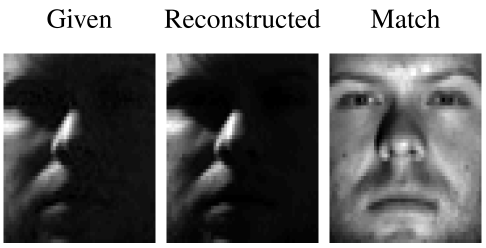

Figure 6. Correct classification of an image of a person under a *new* illumination condition. We can identify the person even though the picture is mostly dark.

noise influences. As such, we obtain a core tensor $\mathcal{S} \in \mathbb{K}^{500\times 56\times 37}$ and matrices $\mathbf{U}_{\text{px}} \in \mathbb{K}^{2958\times 500}$, $\mathbf{U}_{\text{i}} \in \mathbb{K}^{56\times 56}$, and $\mathbf{U}_{\text{p}} \in \mathbb{K}^{37\times 37}$. We use the NLS algorithm to compute a solution, starting from a random initialization. We project the new image $\mathbf{d}^{(\text{new})}$ onto the column space of the pixel matrix $\mathbf{U}_{\text{px}}$ in order to decrease computation time, i.e., $\mathbf{b} = {\mathbf{U}_{\text{px}}}^{\text{T}}\mathbf{d}$. We compare the estimated coefficient vector with $\mathbf{U} = \mathbf{U}_{\text{p}}$. To accommodate for scaling and sign invariance, we normalize the rows of $\mathbf{U}$ and $\tilde{\mathbf{c}}_{\text{p}}^{(\text{new})}$ as follows: a vector $\mathbf{c}$ is normalized as $\text{sign}(c_1)\frac{\mathbf{c}}{||\mathbf{c}||}$. On the left in Figure 6, we see the facial image of a person that is known to our model but for a new illumination condition. In the middle one can see the reconstruction of the image using the estimated coefficient vectors. Moreover, we correctly classified the person as the person on the right in Figure 6.

5.2. *Constructing a tensor that has particular multilinear singular values*

Constructing a matrix with particular singular values is trivial. One can simply use the SVD: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\text{T}}$ in which $\mathbf{\Sigma}$ is a diagonal matrix containing the given singular values and $\mathbf{U}$ and $\mathbf{V}$ are random orthogonal matrices. For tensors, this is not straightforward. It is of fundamental importance to understand the behavior of multilinear singular values [54, 55, 56]. In this section, we show how one can construct an all-orthogonal tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ with particular multilinear singular values using a LS-CPD. Consider the following expressions:

$$\begin{cases} \mathbf{T}_{(1)}\mathbf{T}_{(1)}^{\text{T}} = \mathbf{\Sigma}^{(1)}, \\ \mathbf{T}_{(2)}\mathbf{T}_{(2)}^{\text{T}} = \mathbf{\Sigma}^{(2)}, \\ \mathbf{T}_{(3)}\mathbf{T}_{(3)}^{\text{T}} = \mathbf{\Sigma}^{(3)}, \end{cases} \tag{23}$$

in which $\mathbf{\Sigma}^{(n)} = \text{diag}(\boldsymbol{\sigma}^{(n)^2})$ is a diagonal matrix containing the squared multilinear singular values $\boldsymbol{\sigma}^{(n)}$, $n = 1, 2, 3$. Expression (23) states that $\mathcal{T}$ is all-orthogonal and has multilinear singular values $\boldsymbol{\sigma}^{(n)}$. In order to reformulate (23) as a LS-CPD, we only take the upper triangular parts into account because of symmetry in the left- and right-hand side in (23), leading to the following equations for the first expression in (23):

$$\sum_{j,k} t_{ijk}t_{ijk} = \left(\sigma_i^{(1)}\right)^2, \text{ for } 1 \leq i \leq I,$$

$$\sum_{j,k} t_{i_1jk}t_{i_2jk} = 0, \text{ for } 1 \leq i_1 < i_2 \leq I,$$

and similarly for the second and third expression. We can write this more compactly as a LS-CPD:

$$\mathbf{A}(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b} \quad \text{with} \quad \mathbf{u} = \text{vec}(\mathcal{T}).$$

$\mathbf{A}$ is a binary and sparse matrix of size $I \times J^2$ with $I = \sum_{n=1}^{N} \frac{I_n(I_n+1)}{2}$ and $J = \prod_{n=1}^{N} I_n$. The right-hand side $\mathbf{b} \in \mathbb{K}^I$ is defined as $\mathbf{b} = \left[\text{triu}\left(\mathbf{\Sigma}^{(1)}; \mathbf{\Sigma}^{(2)}; \cdots; \mathbf{\Sigma}^{(N)}\right)\right]$ in which each entry is either zero or a squared multilinear singular value. One can show that the Jacobian for this particular

Table III. The LS-CPD method for constructing a tensor with particular multilinear singular values is faster than APM. This is illustrated by comparing the median computation time (in seconds) across 20 experiments for an $I_1 \times I_2 \times I_3$ tensor with $I_1 = I_2 = 10\alpha$ and $I_3 = 5\alpha$ in which $\alpha = \{1, 5, 10\}$.

	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
Alternating projection method (APM) [55]	0.100	34.4	1747
Construction of $\mathbf{A}$	0.004	1.4	23
Initialization (i.e., one iteration of APM)	0.002	0.4	12
LS-CPD	0.023	15.4	444
Total computation time of LS-CPD	0.029	17.2	479

problem is also a sparse matrix of size $I \times J$ with $\sum_{n=1}^{N} I_n$ non-zeros in each column. More specifically, the Jacobian has the form: $\mathbf{J} = \mathbf{J}^{(1)} + \mathbf{J}^{(2)}$ with $\mathbf{J}^{(1)}$ and $\mathbf{J}^{(2)}$ the derivative to the first and second $\mathbf{u}$, respectively. Computing the sub-Jacobians $\mathbf{J}^{(n)}$ is reduced to filling in elements of $\mathcal{T}$ at the correct position in $\mathbf{J}^{(n)}$ for the orthogonality constraints. For the multilinear singular value constraints, one has to multiply by two. By exploiting the structure, no additional operations are required. We implemented this using a C/mex function that replaces entries to avoid the overhead of constructing sparse matrices in MATLAB. The Gramian of the Jacobian is computed using sparse matrix-vector products.

We compare the optimized NLS algorithm with the alternating projection method (APM) [55] in terms of computation time needed to construct an $I_1 \times I_2 \times I_3$ tensor with given multilinear singular values. We take $I_1 = I_2 = 10\alpha$ and $I_3 = 5\alpha$ in which $\alpha = \{1, 5, 10\}$. The multilinear singular values are chosen by constructing a tensor that can be written as a sum of a multilinear rank-$(L_1, 1, L_1)$ term and a multilinear rank-$(1, L_2, L_2)$ term with $L_1 = I_3 - 1$ and $L_2 = I_3 + 1$. The elements of the factor matrices are drawn from the standard normal distribution. We normalize the multilinear singular values such that the tensor has unit Frobenius norm. We initialize the NLS algorithm with the solution obtained after one iteration of APM. In Table III, we report the median computation time across 20 experiments. The time to construct $\mathbf{A}$ is reported separately because it depends only on the size of the tensor and its construction has to be performed only once. Clearly, the computation time of LS-CPD is much lower than APM, even if we include the time needed to construct $\mathbf{A}$.

5.3. *Blind deconvolution of constant modulus signals*

LS-CPDs can also be used in signal processing applications. We illustrate this by reformulating the blind deconvolution of a constant modulus (CM) signal [57] as the computation of a LS-CPD. In this paper, we investigate the single-input-single-output (SISO) case using an autoregressive (AR) model [58], i.e., we have:

$$\sum_{l=0}^{L} w_l \cdot y[k-l] = s[k] + n[k], \text{ for } 1 \leq k \leq K, \tag{24}$$

with $y[k]$, $s[k]$, and $n[k]$ the measured output, the input, and the additive noise at the kth time instance, respectively. The lth filter coefficient is denoted as w_l. Assume we have $K + L - 1$ samples $y[-L+1], \ldots, y[K]$ and let $\mathbf{Y} \in \mathbb{K}^{L \times K}$ be a Toeplitz matrix defined as $y_{lk} = y[k-l]$. Also, the filter coefficients are collected in $\mathbf{w} \in \mathbb{K}^{L}$ and the source vector $\mathbf{s} \in \mathbb{K}^{K}$ is defined as $s_k = s[k]$. We ignore the noise in the derivation of our method for simplicity. Equation (24) can then be expressed in matrix form as:

$$\mathbf{Y}^{\mathrm{T}}\mathbf{w} = \mathbf{s}. \tag{25}$$

The goal of blind deconvolution is to find the vector $\mathbf{w}$ using only the measured output values [59]. In order to make this problem identifiable, additional prior knowledge has to be exploited. Here, we assume that the input signal has constant modulus (CM), i.e., each sample s_k satisfies the following property [60]:

$$|s_k|^2 = s_k \cdot \overline{s}_k = c, \text{ for } 1 \leq k \leq K \tag{26}$$

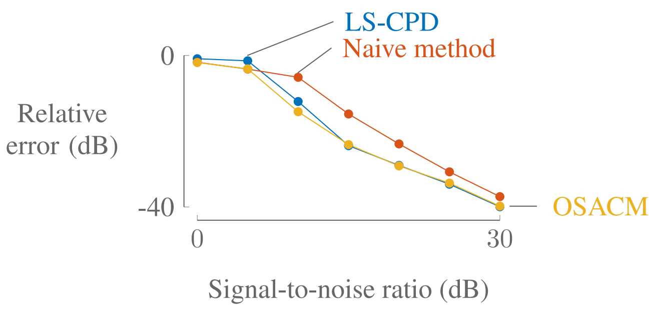

Figure 7. The LS-CPD approach obtains more accurate results than the relaxation-based method and achieves similar accuracy as the dedicated OSACM method.

with c the squared constant modulus which is known *a priori*. By using (25) in (26), we obtain:

$$(\mathbf{y}_k^{\mathrm{T}}\mathbf{w})\left(\overline{\mathbf{y}_k^{\mathrm{T}}\mathbf{w}}\right) = c, \quad \text{or equivalently,} \quad (\mathbf{y}_k \otimes \overline{\mathbf{y}}_k)^{\mathrm{T}} (\mathbf{w} \otimes \overline{\mathbf{w}}) = c, \tag{27}$$

in which $\mathbf{y}_k$ is the kth row of $\mathbf{Y}$, for $1 \leq k \leq K$. Taking into account all equations, (27) reduces to the following LS-CPD:

$$\left(\mathbf{Y} \odot \overline{\mathbf{Y}}\right)^{\mathrm{T}} (\mathbf{w} \otimes \overline{\mathbf{w}}) = c \cdot \mathbf{1}_K. \tag{28}$$

We illustrate the approach by means of the following straightforward example. Consider an AR model of degree $L = 5$ with uniformly distributed coefficients between zero and one, sample length $K = 100$, and $c = 1$. We perturb the measurements with additive Gaussian noise which is scaled to obtain a particular signal-to-noise ratio (SNR). We solve (28) by relaxing $\overline{\mathbf{w}}$ to $\mathbf{v}$:

$$\left(\mathbf{Y} \odot \overline{\mathbf{Y}}\right)^{\mathrm{T}} (\mathbf{w} \otimes \mathbf{v}) = c \cdot \mathbf{1}_K \tag{29}$$

using the NLS algorithm starting from the algebraic solution. In Figure 7, we report the median relative error on $\mathbf{w}$ across 50 experiments for several values of the signal-to-noise ratio (SNR). We compare our approach to the naive method, i.e., the method that relaxes the Kronecker structure in (29), solves the system, and subsequently fits the Kronecker structure to the least-squares solution. These are the core elements of the well-known analytical constant modulus algorithm (ACMA) [57, 61]. We also compare with a state-of-the-art SISO CM algorithm (CMA) called optimal step-size CMA (OSCMA) [62, 63]. It is clear that our method obtains more accurate results than the relaxation-based technique and achieves similar accuracy as the dedicated OSCMA method.

6. CONCLUSION AND FUTURE RESEARCH

In this paper we presented a new framework for linear systems with a CPD constrained solution (LS-CPD). We defined the LS-CPD problem, discussed links between particular types of structured coefficient matrices and the CPD problem, and derived a condition guaranteeing generic uniqueness of the solution. In contrast to the naive method, the proposed algebraic and optimization-based methods allow one to solve the LS-CPD problem in the underdetermined case. Although we focused on the Gauss–Newton (GN) method, the derivations of the expressions for the objective function, gradient, Gramian, and Gramian-vector product can also be used to implement various nonlinear least squares and quasi-Newton algorithms. Numerical experiments show that the algebraic method is a good starting point for optimization-based methods. We also compared the effectiveness of two preconditioners for the GN method. The wide applicability of LS-CPDs is illustrated with three applications from classification, multilinear algebra, and signal processing. Importantly, we have explained that many classification tasks can be formulated as the computation of a LS-CPD. In order to reduce the per-iteration computational complexity of the NLS algorithm, application-dependent structure can be exploited, as we have shown with the construction of tensors that have particular singular values. The focus of this paper was on CPD constrained solutions, however, our framework can be extended to other tensor decompositions such as the MLSVD, TT or HT models.

ACKNOWLEDGEMENTS

Nico Vervliet is supported by an Aspirant Grant from the Research Foundation — Flanders (FWO). Otto Debals is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT). Research furthermore supported by: (1) Research Council KU Leuven: C1 project C16/15/059-nD, CoE PFV/10/002 (OPTEC). (2) FWO projects: G.0830.14N, G.0881.14N, (3) 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 (no. 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.

REFERENCES

1. Candès EJ, Wakin MB. An introduction to compressive sampling. *IEEE Signal Processing Magazine*. Mar 2008; **25**(2):21–30.
2. Gillis N. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, Suykens JAK, Signoretto M, Argyriou A (eds.). chap. 12, Machine Learning and Pattern Recognition, Chapman & Hall / CRC, 2014; 257–291.
3. Kolda TG, Bader BW. Tensor decompositions and applications. *SIAM Review*. Aug 2009; **51**(3):455–500.
4. Cichocki A, Mandic DP, De Lathauwer L, Zhou G, Zhao Q, Caiafa CF, Phan AH. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*. Mar 2015; **32**(2):145–163.
5. Sidiropoulos N, De Lathauwer L, Fu X, Huang K, Papalexakis E, Faloutsos C. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*. 2017; (to appear).
6. De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*. Apr 2000; **21**(4):1253–1278.
7. Oseledets I. Tensor-train decomposition. *SIAM Journal of Scientific Computing*. Sept 2011; **33**(5):2295–2317.
8. Grasedyck L. Polynomial approximation in hierarchical Tucker format by vector tensorization. 2010. Preprint 43, DFG/SPP1324, RWTH Aachen.
9. Boussé M, Debals O, De Lathauwer L. A tensor-based method for large-scale blind source separation using segmentation. *IEEE Transactions on Signal Processing*. Jan 2017; **65**(2):346–358, doi:10.1109/TSP.2016.2617858.
10. Boussé M, Debals O, De Lathauwer L. Tensor-based large-scale blind system identification using segmentation. *Technical Report 16-109, ESAT-STADIUS, KU Leuven, Leuven, Belgium*. 2016; (submitted to IEEE Transactions on Signal Processing).
11. Grasedyck L, Kressner D, Tobler C. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*. Feb 2013; **36**(1):53–78.
12. Khoromskij BN. $\mathcal{O}(d\log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling. *Constructive Approximation*. Oct 2011; **34**(2):257–280, doi:10.1007/s00365-011-9131-1.
13. Hackbusch W. *Tensor spaces and numerical tensor calculus*, vol. 42. Springer, 2012.
14. Ballani J, Grasedyck L. A projection method to solve linear systems in tensor format. *Numerical Linear Algebra with Applications*. Jan 2013; **20**:27–43.
15. Beylkin G, Mohlenkamp MJ. Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*. 2005; **26**(6):2133–2159.
16. Espig M, Hackbusch W, Rohwedder T, Schneider R. Variational calculus with sums of elementary tensors of fixed rank. *Numerische Mathematik*. Nov 2012; **122**(3):469–488.
17. Sorber L, Van Barel M, De Lathauwer L. Unconstrained optimization of real functions in complex variables. *SIAM Journal on Optimization*. July 2012; **22**(3):879–898.
18. Sorber L, Van Barel M, De Lathauwer L. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization. *SIAM Journal on Optimization*. Apr 2013; **23**(2):695–720.
19. Sorber L, Van Barel M, De Lathauwer L. Complex optimization toolbox v1. Feb 2013. URL `http://esat.kuleuven.be/stadius/cot/`.
20. Vervliet N, Debals O, Sorber L, Van Barel M, De Lathauwer L. Tensorlab 3.0 Mar 2016. URL `http://www.tensorlab.net/`.
21. De Lathauwer L. Decompositions of a higher-order tensor in block terms — Part II: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*. Sept 2008; **30**(3):1033–1066.
22. Ding W, Wei Y. Solving multilinear systems with $\mathcal{M}$-tensors. *Journal of Scientific Computing*. Jan 2016; **68**(2).
23. Bai EW, Liu Y. On the least squares solutions of a system of bilinear equations. *Proceedings of the 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference (CDC-ECC 2005, Seville, Spain)*, 2005; 1197–1202.
24. Johnson CR, Šmigoc H, Yang D. Solution theory for systems of bilinear equations. *Linear and Multilinear Algebra*. 2014; **62**(12):1553–1566.
25. Cohen S, Tomasi C. Systems of bilinear equations. *Technical Report*, Stanford, CA, USA 1997.
26. Brazell M, Li N, Navasca C, Tamon C. Solving multilinear systems via tensor inversion. *SIAM Journal on Matrix Analysis and Applications*. May 2013; **34**(2):542–570, doi:10.1137/100804577.

27. Donoho DL. Compressed sensing. *IEEE Transactions on Information Theory*. Apr 2006; **52**(4):1289–1306.
28. Tu S, Boczar R, Simchowitz M, Soltanolkotabi M, Recht B. Low-rank solutions of linear matrix equations via Procrustes flow. *Proceedings of the International Conference on Machine Learning (ICML 2016, New York, USA)*, 2016.
29. Vasilescu MAO, Terzopoulos D. Multilinear image analysis for facial recognition. *Object recognition supported by user interaction for service robots*, vol. 2, 2002; 511–514.
30. Tiels K, Schoukens M, Schoukens J. Generation of initial estimates for Wiener-Hammerstein models via basis function expansions. *IFAC Proceedings Volumes*. 2014; **47**(3):481–486.
31. Nion D, Sidiropoulos ND. Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. *IEEE Transactions on Signal Processing*. June 2009; **57**(6):2299–2310.
32. Boussé M, Vervliet N, Debals O, De Lathauwer L. Tensor-based face recognition using linear systems with a canonical polyadic structure on the solution. *Technical Report 17-41*, ESAT-STADIUS, KU Leuven, Leuven, Belgium 2017.
33. Vasilescu MAO, Terzopoulos D. Multilinear analysis of image ensembles: Tensorfaces. *Proceedings of the European Conference on Computer Vision (ECCV '02, Copenhagen, Denmark)*, 2002; 447–460.
34. Boussé M, Goovaerts G, Vervliet N, Debals O, Van Huffel S, De Lathauwer L. Irregular heartbeat classification using kronecker product equations. *Technical Report 17-20*, ESAT-STADIUS, KU Leuven, Leuven, Belgium 2017.
35. Kruskal JB. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*. Dec 1977; **18**(2):95–138.
36. Domanov I, De Lathauwer L. On the uniqueness of the canonical polyadic decomposition of third-order tensors — Part I: Basic results and uniqueness of one factor matrix. *SIAM Journal on Matrix Analysis and Applications*. July-Sept 2013; **34**(3):855–875.
37. Domanov I, De Lathauwer L. On the uniqueness of the canonical polyadic decomposition of third-order tensors — Part II: Uniqueness of the overal decomposition. *SIAM Journal on Matrix Analysis and Applications*. July-Sept 2013; **34**(3):876–903.
38. Domanov I, De Lathauwer L. Generic uniqueness conditions for the canonical polyadic decomposition and INDSCAL. *SIAM Journal on Matrix Analysis and Applications*. Nov 2015; **36**(4):1567–1589.
39. Domanov I, De Lathauwer L. Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm. *Linear Algebra and its Applications*. Jan 2017; **513**:342–375.
40. De Lathauwer L, Vandewalle J. Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_N)$ reduction in multilinear algebra. *Linear Algebra and its Applications*. Nov 2004; **391**:31–55.
41. Vervliet N, Debals O, Sorber L, De Lathauwer L. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Processing Magazine*. Sept 2014; **31**(5):71–79.
42. Tomasi G, Bro R. PARAFAC and missing values. *Chemometrics and Intelligent Laboratory Systems*. Feb 2005; **75**(2):163–180.
43. Paatero P. A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis. *Chemometrics and Intelligent Laboratory Systems*. Oct 1997; **38**:223–242.
44. Acar E, Dunlavy DM, Kolda TG, Mørup M. Scalable tensor factorizations with missing data. *SIAM International Conference on Data Mining*, 2010; 701–712.
45. Carrol JD, Pruzansky S, Kruskal JB. CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*. Mar 1980; **45**(1):3–24.
46. Vervliet N, Debals O, De Lathauwer L. Canonical polyadic decomposition of incomplete tensors with linearly constrained factors. *Technical Report 16-172, ESAT-STADIUS, KU Leuven, Leuven, Belgium*. 2017; .
47. Sommese AJ, Wampler II CW. *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific, Hackensack, NJ, 2005.
48. De Lathauwer L. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM Journal on Matrix Analysis and Applications*. Sept 2006; **28**(3):642–666.
49. Boussé M, Domanov I, De Lathauwer L. Linear systems with a multilinear singular value decomposition constrained solution. *Technical Report 17-56*, ESAT-STADIUS, KU Leuven, Leuven, Belgium 2017.
50. Horn RA, Johnson CR. *Matrix analysis*. Cambridge University Press, Cambridge, 2013.
51. Nocedal J, Wright S. *Numerical optimization*. Springer New York, 2006.
52. De Lathauwer L. Blind separation of exponential polynomials and the decomposition of a tensor in rank-$(L_r, L_r, 1)$ terms. *SIAM Journal on Matrix Analysis and Applications*. Dec 2011; **32**(4):1451–1474.
53. Vervliet N, Debals O, De Lathauwer L. Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization. *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers (Pacific Grove, CA)*, 2016; 1733–1738.
54. Domanov I, Stegeman A, De Lathauwer L. On the largest multilinear singular values of higher-order tensors. *arXiv:1612.03751*. 2016; (submitted to SIAM Journal on Matrix Analysis and Applications (SIMAX)).
55. Hackbusch W, Kressner D, Uschmajew A. Perturbation of higher-order singular values. 2016. INS Preprint No. 1616.
56. Hackbusch W, Uschmajew A. On the interconnection between the higher-order singular values. *Numerische Mathematik*. 2016; Published online.
57. van der Veen AJ. Algebraic methods for deterministic blind beamforming. *Proceedings of the IEEE*. Oct 1998; **86**(10):1987–2008.
58. Ljung L. *System identification: Theory for the user*. second edn., Prentice hall, 1999.
59. Abed-Meraim K, Qiu W, Hua Y. Blind system identification. *Proceedings of the IEEE*. Aug 1997; **85**(8):1310–1322.
60. Debals O, Sohail M, De Lathauwer L. Analytical multi-modulus algorithms based on coupled canonical polyadic decompositions. *Technical Report 16-150*, ESAT-STADIUS, KU Leuven, Leuven, Belgium 2016.
61. van der Veen AJ, Paulraj A. An analytical constant modulus algorithm. *IEEE Transactions on Signal Processing*. May 1996; **44**(5):1136–115.

62. Zarzoso V, Comon P. Optimal step-size constant modulus algorithm. *IEEE Transactions on communications*. 2008; **56**(1).
63. Treichler J, Agee B. A new approach to multipath correction of constant modulus signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*. 1983; **31**(2):459–472.