NLS Algorithm for Kronecker-Structured Linear Systems with a CPD Constrained Solution

Martijn Bousse*, Nikos Sidiropoulos‡, Lieven De Lathauwer*†
*Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.
†Group Science, Engineering and Technology, KU Leuven Kulak, E. Sabbelaan 53, 8500 Kortrijk, Belgium.
‡Department of Electrical and Computer Engineering, University of Virginia, Thornton Hall 351 Mc-Cormick Road, Charlottesville, VA22904, USA. Email: {martijn.bousse,lieven.delathauwer}@kuleuven.be, nikos@virginia.edu

***Abstract*—In various applications within signal processing, system identification, pattern recognition, and scientific computing, the canonical polyadic decomposition (CPD) of a higher-order tensor is only known via general linear measurements. In this paper, we show that the computation of such a CPD can be reformulated as a sum of CPDs with linearly constrained factor matrices by assuming that the measurement matrix can be approximated by a sum of a (small) number of Kronecker products. By properly exploiting the hypothesized structure, we can derive an efficient non-linear least squares algorithm, allowing us to tackle large-scale problems.**

I. Introduction

Even though the decomposition of a tensor that is known *explicitly* is a prevalent problem in signal processing and machine learning [1], [2], we often want to compute a decomposition of a tensor that is only known via linear measurements [3]. Applications can be found in a wide range of domains such as signal processing [3], [4], system identification [5], [6], pattern recognition [3], [7]–[9], and scientific computing [10]–[13]. By limiting ourselves to a canonical polyadic decomposition (CPD) in this paper, we can formulate the problem as a linear system of equations with a CPD constrained solution (LS-CPD) [3], i.e., $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{x} = \text{vec}\,(\text{CPD})$. Or, equivalently, we want to compute a CPD of a tensor $\mathcal{X} = \text{unvec}\,(\mathbf{x})$ that is only defined *implicitly* via the solution of a linear system.

By fully exploiting all structure of the measurement matrix $\mathbf{A}$, the computational complexity of a dedicated algorithm can be significantly reduced, enabling efficient processing for large-scale problems [3]. For example, if $\mathbf{A}$ is equal to the identity (or a diagonal) matrix, the problem reduces to a (weighted) CPD of a known tensor, allowing efficient computations [14]–[16]. In the special case where $\mathbf{A}$ is sparse, we can also obtain efficient algorithms, see [3].

In this paper, we assume that $\mathbf{A}$ can be written as a sum of L Kronecker products. This strategy is employed to reduce the computational complexity of algorithms in various applications within signal processing [17], [18], system identification [5], [6], [19], [20], and tensor-based scientific computing [10]–[12], among others. Depending on the application, the products are considered to be given or they can be computed. As a matter of fact, *any* measurement matrix can be approximated by a sum of Kronecker products for sufficiently large L. For a given L, a least-squares approximation can be computed via a Kronecker product decomposition [21]–[23].

By explicitly leveraging the Kronecker structure of $\mathbf{A}$, the LS-CPD problem can be reformulated as a sum of L CPDs with linear constraints. In constrast to existing methods that employ projection [11], alternating least-squares [4], [12], or a gradient approach [13], we develop numerical optimization-based techniques such as quasi-Newton (qN) and nonlinear least-squares (NLS) with known convergence properties [24]. By carefully exploiting all available structure, our algorithm can tackle large-scale problems [25]. For $L = 1$, the problem can be related to CANDELINC [26], [27], which can be computed efficiently in the dense [26] and sparse [28] case.

In the remainder of this section, we discuss notations and basic definitions. In Section II, we reformulate the LS-CPD via Kronecker structure. By properly exploiting the structure, we obtain an efficient optimization-based algorithm in Section III. Numerical experiments are discussed in Section IV.

A. Notations and basic definitions

A tensor (denoted by $\mathcal{A}$) is a higher-order generalization of a vector and a matrix (denoted by $\mathbf{a}$ and $\mathbf{A}$, respectively). 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. The mode-n unfolding of $\mathcal{A}$ is the matrix $\mathbf{A}_{(n)}$ with the mode-n vectors as its columns (using the ordering convention in [27]). 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 $\text{vec}(\cdot)$. We denote the outer, Kronecker and Khatri–Rao product as $\otimes$, $\otimes$ and $\odot$, respectively. We say that a 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.

B. Canonical Polyadic Decomposition (CPD)

The CPD is an important tool for tensor analysis in signal processing, data mining and machine learning [1], [2], [27]. The decomposition is unique under rather mild conditions [29], [30], which is a powerful advantage over matrices [2].

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

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

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 said to be *canonical* (CPD) when R is equal to the rank of $\mathcal{A}$.
The CANDELINC model is a popular tool to incorporate prior knowledge in the CPD by means of linear constraints, allowing one to improve the accuracy and/or interpretability [26], [28]. One assumes that $\mathbf{U}^{(n)} = \mathbf{A}^{(n)}\mathbf{C}^{(n)}$ in which $\mathbf{A}^{(n)}$ is known and $\mathbf{C}^{(n)}$ is the unknown coefficient matrix.

C. Identities and derivatives

We use the following identities in this paper [31]:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}, \tag{1}$$

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \odot \mathbf{D}) = \mathbf{AC} \odot \mathbf{BD}, \tag{2}$$

$$(\mathbf{A} \odot \mathbf{B})^{\mathrm{T}} (\mathbf{C} \odot \mathbf{D}) = \mathbf{A}^{\mathrm{T}}\mathbf{C} * \mathbf{B}^{\mathrm{T}}\mathbf{D}, \tag{3}$$

$$\mathrm{vec}\,(\mathbf{ABC}) = (\mathbf{C}^{\mathrm{T}} \otimes \mathbf{A})\,\mathrm{vec}\,(\mathbf{B}), \tag{4}$$

$$\mathrm{vec}\,(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket_R) = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\,\mathbf{1}_R. \tag{5}$$

Given a matrix $\mathbf{P}^{(n)}$ that permutes the nth mode of a vectorized tensor to the first mode and $\mathbf{P}^{(n)^{\mathrm{T}}}\mathbf{P}^{(n)} = \mathbf{I}$, we have:

$$\mathbf{P}^{(n)}\mathrm{vec}\left(\left\llbracket \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right\rrbracket\right) = \mathrm{vec}\left(\left\llbracket \mathbf{U}^{(n)}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n-1)}, \mathbf{U}^{(n+1)}, \ldots, \mathbf{U}^{(N)} \right\rrbracket\right).$$

By defining $\mathbf{V}^{\{n\}} = \odot_{q=1, q\neq N-n+1}^{N} \mathbf{U}^{(N-q+1)}$, we obtain:

$$\mathbf{P}^{(n)^{\mathrm{T}}}\left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}\right)\mathrm{vec}\,(\mathbf{X}) = \mathrm{vec}\left(\left\llbracket \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n-1)}, \mathbf{X}, \mathbf{U}^{(n+1)}, \ldots, \mathbf{U}^{(N)} \right\rrbracket\right), \tag{6}$$

$$\mathbf{P}^{(n)^{\mathrm{T}}}\mathrm{vec}\left(\mathbf{U}^{(n)}\mathbf{V}^{\{n\}^{\mathrm{T}}}\right) = \mathrm{vec}\left(\left\llbracket \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right\rrbracket\right).$$

Finally, we also use the following derivative [32]:

$$\frac{\partial \mathrm{vec}\left(\left\llbracket \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right\rrbracket\right)}{\partial \mathrm{vec}\left(\mathbf{U}^{(n)}\right)} = \mathbf{P}^{(n)^{\mathrm{T}}}\left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n}\right). \tag{7}$$

II. Kronecker-structured LS-CPD a sum of CPDs with linearly constrained factor matrices

In this paper, we consider a CPD of a tensor that is only known via linear measurements. This can be formulated as a linear system with a CPD constrained solution (LS-CPD) [3]:

$$\mathbf{Ax} = \mathbf{b} \quad \text{with} \quad \mathbf{x} = \mathrm{vec}\left(\left\llbracket \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right\rrbracket\right) \tag{8}$$

in which $\mathbf{A} \in \mathbb{K}^{M \times K}$, $\mathbf{U}^{(n)} \in \mathbb{K}^{I_n \times R}$, and $K = \prod_{n=1}^{N} I_n$. Additionally, we assume that $\mathbf{A}$ admits, or can be well approximated by, a sum of L Kronecker products of smaller matrices $\mathbf{A}^{(n,l)} \in \mathbb{K}^{J_n \times I_n}$, for $1 \leq n \leq N$, and $M = \prod_{n=1}^{N} J_n$, i.e.,

$$\mathbf{A} = \sum_{l=1}^{L} \mathbf{A}^{(N,l)} \otimes \mathbf{A}^{(N-1,l)} \otimes \cdots \otimes \mathbf{A}^{(1,l)}. \tag{9}$$

By assuming Kronecker structure (9), the LS-CPD problem in (8) can be reduced to a sum of CPDs with linear constraints. First, combine (8) and (9) to obtain:

$$\sum_{l=1}^{L}\left(\mathbf{A}^{(N,l)} \otimes \cdots \otimes \mathbf{A}^{(1,L)}\right)\left(\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(1)}\right) \cdot \mathbf{1}_R = \mathbf{b}.$$

By using the mixed-product rule in (2), we can write that:

$$\sum_{l=1}^{L}\left(\mathbf{A}^{(N,l)}\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{A}^{(1,l)}\mathbf{U}^{(1)}\right) \cdot \mathbf{1}_R = \mathbf{b}. \tag{10}$$

By defining factor matrices $\mathbf{V}^{(n,l)} = \mathbf{A}^{(n,l)}\mathbf{U}^{(n)}$, $1 \leq n \leq N$, we can write (10) as a sum of L CPDs with linearly constrained factor matrices of a tensor $\mathcal{B} = \mathrm{unvec}(\mathbf{b})$ as follows:

$$\mathcal{B} = \sum_{l=1}^{L} \left\llbracket \mathbf{V}^{(1,l)}, \mathbf{V}^{(2,l)}, \ldots, \mathbf{V}^{(N,l)} \right\rrbracket. \tag{11}$$

If $L = 1$, it is clear that (11) can be related to the well-known CANDELINC. For $L > 1$, we obtain a more general model.

By stacking the factor matrices $\mathbf{V}^{(n,l)}$, $1 \leq l \leq L$, in a matrix $\mathbf{V}^{(n)} = \begin{bmatrix}\mathbf{V}^{(n,1)} & \mathbf{V}^{(n,2)} & \ldots & \mathbf{V}^{(n,L)}\end{bmatrix} \in \mathbb{K}^{J_n \times RL}$, (11) reduces to a rank-RL CPD with linear block constraints:

$$\mathcal{B} = \left\llbracket \mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \ldots, \mathbf{V}^{(N)} \right\rrbracket.$$

If all $\mathbf{A}^{(n,l)}$ have full column rank, existing uniqueness results can be used, see [29], [30]. Depending on the application, we are interested in either interpretable, and therefore unique, factor matrices or a compact representation of the underlying tensor using factor matrices which do not need to be unique.

III. Nonlinear least-squares algorithm

By properly exploiting the Kronecker structure, we derive an efficient NLS algorithm for (8)-(9). The computation can be formulated as an optimization problem as follows:

$$\min_{\mathbf{z}} f = \frac{1}{2}\,||\mathcal{F}||_{\mathrm{F}}^2 \quad \text{with } \mathcal{F} \text{ defined as in (13)}, \tag{12}$$

in which the variables $\mathbf{U}^{(n)}$, for $1 \leq n \leq N$, are concatenated in a vector $\mathbf{z} \in \mathbb{K}^{RI^+}$ with $I^+ = \sum_{n=1}^{N} I_n$, as follows: $\mathbf{z} = \left[\mathrm{vec}\left(\mathbf{U}^{(1)}\right); \cdots; \mathrm{vec}\left(\mathbf{U}^{(N)}\right)\right]$. The residual $\mathcal{F}$ is given by:

$$\mathcal{F} = \sum_{l=1}^{L} \left\llbracket \mathbf{V}^{(1,l)}, \mathbf{V}^{(2,l)}, \ldots, \mathbf{V}^{(N,l)} \right\rrbracket - \mathcal{B} \tag{13}$$

with linear constraints $\mathbf{V}^{(n,l)} = \mathbf{A}^{(n,l)}\mathbf{U}^{(n,l)} \in \mathbb{K}^{J_n \times R}$.

We can solve the optimization problem in (12)-(13) using standard qN and NLS algorithms by deriving expressions for the evaluation of the objective function, gradient, Jacobian, Gramian, and Gramian-vector product. Importantly, we exploit all available structure in order to obtain efficient implementations. In this paper we focus on the Gauss–Newton (GN) algorithm [24], but the expressions can be used for other qN and NLS algorithms as well. In order to implement our algorithm, we use the complex optimization framework from [25], [33], [34], which provides qN and NLS implementations as well as line, plane search, and trust-region methods. Additionally, we provide a computational complexity analysis.

The GN method using dogleg trust-region solves (12) by linearizing the residual $\mathrm{vec}\,(\mathcal{F})$ in each iteration k and subsequently by solving the following least-squares problem [24]:

$$\min_{\mathbf{p}_k} \frac{1}{2}\,||\mathrm{vec}\,(\mathcal{F}_k) + \mathbf{J}_k\mathbf{p}_k||_{\mathrm{F}}^2 \quad \text{s.t.} \quad ||\mathbf{p}_k|| \leq \Delta_k \tag{14}$$

with step $\mathbf{p}_k = \mathbf{z}_{k+1} - \mathbf{z}_k$, Jacobian $\mathbf{J} = d\text{vec}(\mathcal{F})/d\mathbf{z}$, and trust-region Δ_k. The exact solution to (14) is given by the linear system $\mathbf{H}_k\mathbf{p}_k = -\overline{\mathbf{g}}_k$ with $\mathbf{H}$ the Hessian, which we approximate with the Gramian of the Jacobian, and the conjugated gradient $\overline{\mathbf{g}} = (\partial f/\partial \mathbf{z})^{\text{H}}$ [24]. The variables can then be updated as $\mathbf{z}_{k+1} = \mathbf{z}_k + \mathbf{p}_k$. In this paper, we solve the linear system using several preconditioned conjugated gradient (CG) iterations in order to reduce computational complexity. The GN method is summarized in Algorithm 1.

Algorithm 1: Kronecker-strucured LS-CPD using Gauss–Newton and dogleg trust region.

Input: $\mathcal{B}$, $\{\mathbf{A}^{(n)}\}_{n=1}^N$, and $\{\mathbf{U}^{(n)}\}_{n=1}^N$
Output: $\{\mathbf{U}^{(n)}\}_{n=1}^N$
while *not converged* **do**
  Compute gradient $\mathbf{g}$ using (15) and (16).
  Use PCG to solve $\mathbf{H}\mathbf{p} = -\overline{\mathbf{g}}$ for $\mathbf{p}$ using Gramian-vector products in (17)-(18) and a block-Jacobi preconditioner as explained in Section III-E.
  Update $\{\mathbf{U}^{(n)}\}_{n=1}^N$, using dogleg trust region from $\mathbf{p}$, $\mathbf{g}$, and objective function evaluation (12).
end

A. Objective function

The objective function f can be evaluated by taking the sum of squared entries of the residual $\mathcal{F}(\mathbf{z})$ as defined in (13).

B. Jacobian

The Jacobian $\mathbf{J}$ can be partitioned in the following way:

$$\mathbf{J} = \frac{\partial \text{vec}(\mathcal{F})}{\partial \mathbf{z}} = \begin{bmatrix}\mathbf{J}^{(1)} & \mathbf{J}^{(2)} & \cdots & \mathbf{J}^{(N)}\end{bmatrix} \in \mathbb{K}^{K\times RI^+}.$$

with the nth sub-Jacobian $\mathbf{J}^{(n)}$ defined as:

$$\mathbf{J}^{(n)} = \sum_{l=1}^{L} \mathbf{P}^{(n)^{\text{T}}}\left(\mathbf{V}^{\{n,l\}} \otimes \mathbf{A}^{(n,l)}\right) \in \mathbb{K}^{K\times RI_n}$$

with $\mathbf{V}^{\{n,l\}} = \odot_{q=1,q\neq N-n+1}^{N} \mathbf{V}^{(n,l)}$.

Proof.

$$\begin{aligned}
\mathbf{J}^{(n)} &= \frac{\partial \text{vec}(\mathcal{F})}{\partial \text{vec}\left(\mathbf{U}^{(n)}\right)} = \sum_{l=1}^{L} \frac{\partial\left(\left[\!\left[\mathbf{V}^{(1,l)},\ldots,\mathbf{V}^{(N,l)}\right]\!\right]\right)}{\partial \text{vec}\left(\mathbf{U}^{(n)}\right)} \\
&= \sum_{l=1}^{L} \frac{\partial\left(\left[\!\left[\mathbf{A}^{(1,l)}\mathbf{U}^{(1)},\ldots,\mathbf{A}^{(N,l)}\mathbf{U}^{(N)}\right]\!\right]\right)}{\partial \text{vec}\left(\mathbf{U}^{(n)}\right)} \\
&= \sum_{l=1}^{L}\left(\bigotimes_{n=N}^{1}\mathbf{A}^{(n,l)}\right)\cdot\frac{\partial \text{vec}\left(\left[\!\left[\mathbf{U}^{(1)},\ldots,\mathbf{U}^{(N)}\right]\!\right]\right)}{\partial \text{vec}\left(\mathbf{U}^{(n)}\right)} \\
&= \sum_{l=1}^{L}\left(\bigotimes_{n=N}^{1}\mathbf{A}^{(n,l)}\right)\mathbf{P}^{(n)^{\text{T}}}\left(\mathbf{V}^{\{n\}}\otimes\mathbf{I}_{I_n}\right) \\
&= \sum_{l=1}^{L}\mathbf{P}^{(n)^{\text{T}}}\left(\mathbf{V}^{\{n,l\}}\otimes\mathbf{A}^{(n,l)}\right).
\end{aligned}$$

Identities (2) and (5) enable the third equation. The last two equations are obtained by using (7) and (1)-(2), respectively.

C. Gradient

The gradient $\mathbf{g}$ can be partitioned in the following way

$$\mathbf{g} = \frac{\partial f}{\partial \mathbf{z}} = \begin{bmatrix}\mathbf{g}^{(1)} & \mathbf{g}^{(2)} & \cdots & \mathbf{g}^{(N)}\end{bmatrix} \in \mathbb{K}^{RI^+}, \tag{15}$$

in which $\mathbf{g}^{(n)} \in \mathbb{K}^{RI_n}$ is defined by:

$$\mathbf{g}^{(n)} = \sum_{l=1}^{L} \text{vec}\left(\mathbf{A}^{(n,l)^{\text{T}}}\overline{\mathbf{F}}_{(n)}\mathbf{V}^{\{n,l\}}\right). \tag{16}$$

Proof. The nth subgradient is given by:

$$\begin{aligned}
\mathbf{g}^{(n)} &= \frac{\partial f}{\partial \text{vec}\left(\mathbf{U}^{(n)}\right)} = \mathbf{J}^{(n)^{\text{T}}}\overline{\text{vec}(\mathcal{F})} \\
&= \sum_{l=1}^{L}\left(\mathbf{V}^{\{n,l\}^{\text{T}}}\otimes\mathbf{A}^{(n,l)^{\text{T}}}\right)\mathbf{P}^{(n)}\overline{\text{vec}(\mathcal{F})} \\
&= \sum_{l=1}^{L}\text{vec}\left(\mathbf{A}^{(n,l)^{\text{T}}}\overline{\mathbf{F}}_{(n)}\mathbf{V}^{\{n,l\}}\right).
\end{aligned}$$

We use (4) to obtain the last equation, which can be computed efficiently using Tensorlab's `mtkrprod` implementation [35].

D. Gramian-vector product

We compute the product $\mathbf{J}^{(m)^{\text{H}}}\mathbf{J}^{(n)}\text{vec}\left(\mathbf{X}^{(n)}\right)$ efficiently, by first computing $\mathbf{t} = \mathbf{J}^{(n)}\text{vec}\left(\mathbf{X}^{(n)}\right)$ with $\mathbf{X}^{(n)} \in \mathbb{K}^{I_n\times R}$:

$$\begin{aligned}
\mathbf{J}^{(n)}\text{vec}\left(\mathbf{X}^{(n)}\right) &= \sum_{l=1}^{L}\mathbf{P}^{(n)^{\text{T}}}\left(\tilde{\mathbf{V}}^{\{n,l\}}\otimes\mathbf{A}^{(n,l)}\right)\text{vec}\left(\mathbf{X}^{(n)}\right) \\
&= \sum_{l=1}^{L}\text{vec}\Big(\Big[\!\Big[\mathbf{V}^{(1,l)},\ldots,\mathbf{V}^{(n-1,l)},\mathbf{A}^{(n,l)}\mathbf{X}^{(n)}, \\
&\qquad\qquad \mathbf{V}^{(n+1,l)},\ldots,\mathbf{V}^{(N,l)}\Big]\!\Big]\Big).
\end{aligned} \tag{17}$$

We obtain (17) by using (6). Next, we compute $\mathbf{y} = \mathbf{J}^{(m)^{\text{H}}}\mathbf{t}$:

$$\begin{aligned}
\mathbf{y} = \mathbf{J}^{(m)^{\text{H}}}\mathbf{t} &= \sum_{l=1}^{L}\left(\mathbf{V}^{\{m,l\}^{\text{H}}}\otimes\mathbf{A}^{(m,l)^{\text{H}}}\right)\mathbf{P}^{(m)}\mathbf{t} \\
&= \sum_{l=1}^{L}\text{vec}\left(\mathbf{A}^{(m,l)^{\text{H}}}\mathbf{T}_{(m)}\mathbf{V}^{\{m,l\}}\right).
\end{aligned} \tag{18}$$

We use (5) to obtain the last equation, which can be computed efficiently using Tensorlab's `mtkrprod` implementation [35].

E. Block-Jacobi preconditioner

We use a block-Jacobi preconditioner to reduce the number of conjugated gradient (CG) iterations and improve overall convergence. In that case, we have to compute the inverse of $\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)} \in \mathbb{K}^{RI_n\times RI_n}$, for $1 \leq n \leq N$, in each iteration. The (n,n)th sub-Gramian is given by:

$$\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)} = \sum_{l=1}^{L}\left(\mathbf{W}^{\{n,l\}}\otimes\mathbf{A}^{(n,l)^{\text{H}}}\mathbf{A}^{(n,l)}\right) \tag{19}$$

with $\mathbf{W}^{\{n,l\}} = \mathbf{V}^{\{n,l\}^{\text{H}}}\mathbf{V}^{\{n,l\}} \in \mathbb{K}^{R\times R}$ which can be computed as $\mathbf{W}^{\{n,l\}} = *_{q=1,q\neq n}^{N}\mathbf{V}^{(n,l)^{\text{H}}}\mathbf{V}^{(n,l)}$ using (3).

TABLE I

BY FULLY EXPLOITING THE KRONECKER STRUCTURE, WE OBTAIN A SIGNIFICANT IMPROVEMENT IN THE COMPUTATIONAL COMPLEXITY.

		Complexity	
	Calls/iteration	Our algorithm	LS-CPD
Factor matrices $\mathbf{V}^{(n,l)}$	1	$\mathcal{O}(NRIJL)$	/
Objective function	1 + it_{TR}	$\mathcal{O}(RML)$	$\mathcal{O}(RMI^N)$
Jacobian	1	$\mathcal{O}(NRMLI)$	$\mathcal{O}(NRMI^N)$
Gradient	1	$\mathcal{O}(NRML)$	$\mathcal{O}(NRMI^N)$
Gramian-vector	it_{CG}	$\mathcal{O}(NRML)$	$\mathcal{O}(NRMI)$

For $L = 1$, computing the inverse of (19) can be done efficiently by omitting the explicit construction of the Jacobians:

$$\left(\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)}\right)^{\dagger} = \left(\mathbf{W}^{\{n,l\}}\right)^{\dagger} \otimes \left(\mathbf{A}^{(n,l)^{\text{H}}}\mathbf{A}^{(n,l)}\right)^{\dagger},$$

in which $\left(\mathbf{A}^{(n,l)^{\text{H}}}\mathbf{A}^{(n,l)}\right)^{\dagger}$ can be computed beforehand and $\left(\mathbf{W}^{\{n,l\}}\right)^{\dagger}$ requires the inverse of small $(R \times R)$ matrices.

F. Computational complexity

By exploiting the Kronecker structure in (8)-(9), we obtain a significant improvement in the computational complexity, enabling an efficient algorithm for large-scale problems. In order to illustrate this, we compare the per-iteration computational complexity of our algorithm with the LS-CPD algorithm in [3], which ignores the structure of $\mathbf{A}$. For simplicity, we assume that $I_n = I$. The number of trust-region (TR) and CG iterations are denoted by it_{TR} and it_{CG}, respectively.

IV. EXPERIMENTS

A. The Block-Jacobi preconditioner is effective

By using the block-Jacobi preconditioner we can effectively reduce the number of CG iterations in different scenarios. Consider problem (8)-(9) with $N = 3$, $R = 2$, $L = 3$ $I_1 = I_2 = I_3 = I = 10$, and $K = 1000$. Taking $J_1 = J_2 = J_3 = J$, we consider three scenarios: 1) the square case with $J = 10$ and $M = 1000 = K$, 2) the underdetermined case with $J = 9$ and $M = 729 < K$, and 3) the highly underdetermined case with $J = 5$ and $M = 125 \ll K$. We simulate a typical iteration of the algorithm by computing the Gramian $\mathbf{H}$ and the gradient $\mathbf{g}$ for random factor matrices and then solving $\mathbf{Hp} = -\overline{\mathbf{g}}$ using preconditioned CG until convergence (up to a tolerance of 10^{-6}). We report the average and standard deviation of the number of CG iterations across fifty experiments in Table II using no PC and the block-Jacobi PC from Subsection III-E.

B. Graph clustering as a Kronecker-structured LS-CPD

Partitioning a graph into meaningful clusters, is crucial to analyze large networks. We show that the similarity-based clustering method in [36] can be reformulated as the computation of a Kronecker-structured LS-CPD. The similarity measure is defined as a weighted infinite sum of the number of common target nodes using neighborhood patterns of any length. It has been shown in [36] that the similarity can then be computed by finding a solution to the following equation:

$$\left[\mathbf{I} \otimes \mathbf{I} - \beta^2 \left(\mathbf{G} \otimes \mathbf{G} + \mathbf{G}^{\text{T}} \otimes \mathbf{G}^{\text{T}}\right)\right] \text{vec}\left(\mathbf{S}\right) = \text{vec}\left(\mathbf{G}\mathbf{G}^{\text{T}} + \mathbf{G}^{\text{T}}\mathbf{G}\right) \quad (20)$$

with $\mathbf{G}$ the weighted adjacency matrix of the graph, $\mathbf{S}$ the unknown similarity measure, and a parameter β. In order to reduce the computational cost, it has been proposed in [36] to find a low-rank approximation $\hat{\mathbf{S}}$ of $\mathbf{S}$ instead, reducing (20) to (8)-(9) with $L = 3$ and $N = 2$. In contrast to the method in [36], our approach can easily be extended to $N > 2$, allowing one to approximate $\mathbf{S}$ with a low-rank *tensor* model.

We illustrate our method for an Erdős–Rényi random graph with fifty nodes and a simple block structure[1] in the way explained in [36] and visualized in Figure 1. In this example, we simulate a community in which nodes primarily interact with other nodes of the same cluster, which occurs, e.g., in (online) social networks. By choosing $R \geq 3$ for this example, we can obtain a meaningful clustering of the nodes because the low-rank model provides a *good* approximation of the underlying similarity measure, as can be seen in Figure 1.

TABLE II

THE BLOCK-JACOBI PRECONDITIONER (PC) EFFECTIVELY REDUCES THE NUMBER OF CONJUGATED GRADIENT (CG) ITERATIONS IN VARIOUS SCENARIOS. WE REPORTED THE AVERAGE (AND STANDARD DEVIATION OF THE) NUMBER OF CG ITERATIONS ACROSS FIFTY EXPERIMENTS.

Scenario	No PC	block-Jacobi PC
Square	35 (6)	11 (2)
Underdetermined	38 (7)	13 (2)
Highly underdetermined	60 (0)	35 (6)

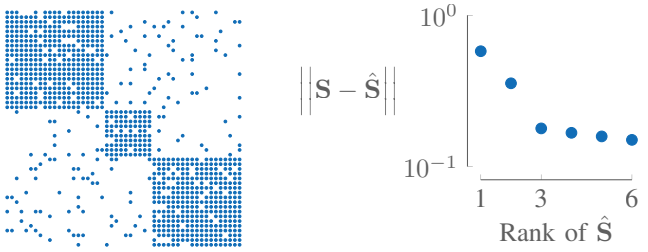

Fig. 1. A low-rank model $\hat{\mathbf{S}}$ of the similarity measure provides a *good* approximation, allowing one to extract meaningful clusters using this approach [36].

V. CONCLUSION

In this paper, we assumed that the measurement matrix in the LS-CPD paper can be approximated by a sum of a (small) number of Kronecker products. By fully exploiting the structure, we were able to reformulate the LS-CPD problem as a sum of CPDs with linear constraints. This insight allowed us to derive efficient expressions for the ingredients of well-known qN and NLS algorithms, as demonstrated by the complexity analysis, enabling us to tackle large-scale problems.

[1] We us an identity matrix as roll graph and $p_{\text{in}} = 0.9$ and $p_{\text{out}} = 0.1$ [36]. The error results in Figure 1 are the median across fifty random experiments.

Additionally, we have numerically tested the effectiveness of the block-Jacobi preconditioner and we have demonstrated our approach for graph clustering. In future work, one can derive a more efficient preconditioner for $L > 1$ in order to fully omit the explicit construction of the Jacobians in the algorithm.

ACKNOWLEDGMENT

This research was supported by: Fonds de la Recherche Scientifique – FNRS and Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160 (SeLMA), Research Council KU Leuven: C1 project C16/15/059-nD, 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. This research was also supported by: U.S. NSF grants IIS-1704074, ECCS-1807660, and ECCS-1608961.

REFERENCES

[1] A. Cichocki, D. P. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. F. Caiafa, and A.-H. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[2] N. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[3] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications," *Numerical Linear Algebra with Applications*, vol. 25, no. 6, p. e2190, Aug. 2018.

[4] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, Jan. 2013.

[5] M. Boussé and L. De Lathauwer, "Large-scale autoregressive system identification using Kronecker product equations," in *2018 6th IEEE Global Conference on Signal and Information Processing (GlobalSIP 2018, Anaheim, California, USA)*, Nov. 2018.

[6] G. Monchen, B. Sinquin, and M. Verhaegen, "Recursive Kronecker-based vector autoregressive identification for large-scale adaptive optics," *IEEE Transactions on Control Systems Technology*, 2018, (accepted for publication.

[7] M. Boussé, N. Vervliet, O. Debals, and L. De Lathauwer, "Face recognition as a Kronecker product equation," in *IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2017, Curaçao, Dutch Antilles)*, Dec. 2017, pp. 276–280.

[8] M. Boussé, G. Goovaerts, N. Vervliet, O. Debals, S. Van Huffel, and L. De Lathauwer, "Irregular heartbeat classification using Kronecker product equations," in *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2017, Jeju Island, South-Korea)*, July 2017, pp. 438–441.

[9] W. Guo, I. Kotsia, and I. Patras, "Tensor learning for regression," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 816–827, Feb. 2012.

[10] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, Feb. 2013.

[11] J. Ballani and L. Grasedyck, "A projection method to solve linear systems in tensor format," *Numerical Linear Algebra with Applications*, vol. 20, pp. 27–43, Jan. 2013.

[12] G. Beylkin and M. J. Mohlenkamp, "Algorithms for numerical analysis in high dimensions," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 2133–2159, 2005.

[13] M. Espig, W. Hackbusch, T. Rohwedder, and R. Schneider, "Variational calculus with sums of elementary tensors of fixed rank," *Numerische Mathematik*, vol. 122, no. 3, pp. 469–488, Nov. 2012.

[14] 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 Journal on Optimization*, vol. 23, no. 2, pp. 695–720, Apr. 2013.

[15] P. Paatero, "A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 38, pp. 223–242, Oct. 1997.

[16] M. Boussé and L. De Lathauwer, "Nonlinear least squares algorithm for canonical polyadic decomposition using low-rank weights," in *IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2017, Curaçao, Dutch Antilles)*, Dec. 2017, pp. 39–43.

[17] M. Boussé, O. Debals, and L. De Lathauwer, "A tensor-based method for large-scale blind source separation using segmentation," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 346–358, Jan. 2017.

[18] ——, "Tensor-based large-scale blind system identification using segmentation," *IEEE Transactions on Signal Processing*, vol. 65, no. 21, pp. 5770–5784, Nov. 2017.

[19] B. Sinquin and M. Verhaegen, "K4SID: Large-scale subspace identification with Kronecker modeling," *IEEE Transactions on Automatic Control*, May 2018.

[20] ——, "Quarks: Identification of large-scale Kronecker vector-autoregressive models," *IEEE Transactions on Automatic Control*, 2018.

[21] C. Van Loan and N. Pitsianis, *Approximation with Kronecker Products*, M. "Moonen, G. Golub, and B. De Moor, Eds. Dordrecht: Springer Netherlands, 1993.

[22] J. Nagy and M. Kilmer, "Kronecker product approximation for preconditioning in three-dimensional imaging applications," *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 604–613, 2006.

[23] K. Batselier and N. Wong, "A constructive arbitrary-degree Kronecker product decomposition of tensors," *arXiv:1507.08805v3*, 2016.

[24] J. Nocedal and S. Wright, *Numerical optimization*. Springer New York, Jul. 2006.

[25] 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.

[26] J. D. Carrol, S. Pruzansky, and J. B. Kruskal, "CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters," *Psychometrika*, vol. 45, no. 1, pp. 3–24, Mar. 1980.

[27] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[28] N. Vervliet, O. Debals, and L. De Lathauwer, "Canonical polyadic decomposition of incomplete tensors with linearly constrained factors," *Technical Report 16âĂŞ172, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2017.

[29] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: Reduction to generalized eigenvalue decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 636–660, Apr.-May 2014.

[30] ——, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm," *Linear Algebra and its Applications*, vol. 513, pp. 342–375, Jan. 2017.

[31] S. Liu and G. Trenkler, "Hadamard, Khatri–Rao, Kronecker and other matrix products," *International Journal of Information and Systems Sciences*, vol. 4, no. 1, pp. 160–177, 2008.

[32] T. G. Kolda, "Multilinear operators for higher-order decompositions," Sandia National Laboratories, Albuquerque, NM, Livermore, CA, Tech. Rep., Apr. 2006, tech. Report SAND2006-2081.

[33] L. Sorber, M. Van Barel, and L. De Lathauwer, "Unconstrained optimization of real functions in complex variables," *SIAM Journal on Optimization*, vol. 22, no. 3, pp. 879–898, July 2012.

[34] ——, "Complex optimization toolbox v1." Feb. 2013. [Online]. Available: http://esat.kuleuven.be/stadius/cot/

[35] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," Mar. 2016. [Online]. Available: http://www.tensorlab.net/

[36] A. Browet and P. Van Dooren, "Low-rank similarity measure for role model extraction," in *21st International Symposium on Mathematical Theory of Networks and Systems (MNTS 2017, Groningen, The Netherlands)*, July 2014, pp. 1412–1418.