Nonlinear Least Squares Algorithm for Canonical Polyadic Decomposition Using Low-Rank Weights

Martijn Boussé* and 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
Email: {martijn.bousse, lieven.delathauwer}@kuleuven.be

***Abstract*—The canonical polyadic decomposition (CPD) is an important tensor tool in signal processing with various applications in blind source separation and sensor array processing. Many algorithms have been developed for the computation of a CPD using a least squares cost function. Standard least-squares methods assumes that the residuals are uncorrelated and have equal variances which is often not true in practice, rendering the approach suboptimal. Weighted least squares allows one to explicitly accommodate for general (co)variances in the cost function. In this paper, we develop a new nonlinear least-squares algorithm for the computation of a CPD using low-rank weights which enables efficient weighting of the residuals. We briefly illustrate our algorithm for direction-of-arrival estimation using an array of sensors with varying quality.**

I. Introduction

Tensors are higher-order generalizations of vectors (first-order) and matrices (second-order). Recently, tensor tools have gained popularity in various applications within signal processing, data mining, and machine learning [1]–[3]. An important tensor tool is the canonical polyadic decomposition (CPD) which decomposes a tensor into a minimal sum of rank-1 tensors. The decomposition is unique under mild uniqueness conditions [4]–[6], making the CPD a powerful tool in various signal processing applications such as (large-scale) instantaneous and convolutive blind source separation [7]–[10], sensor array processing [11], [12], and telecommunications [13]. Various algorithms have been developed for the computation of a CPD such as algebraic methods [6], [14], [15], alternating-least squares (ALS) methods [1], and all-at-once optimization techniques [16]–[20]. Although ALS-based methods are popular, they are often outperformed in ill-conditioned cases by more sophisticated optimization techniques such as quasi-Newton (qN) and nonlinear least-squares (NLS) algorithms [16].

In signal processing, the data is often perturbed by noise. Weighted least squares (WLS) allows one to include prior knowledge about the noise in the least-squares cost function. A common choice is the inverse of the sample covariance matrix because it leads to the optimal estimate. In practice, the sample covariance matrix may be unknown but often some reasonable estimate can be computed by exploiting prior knowledge [21]. For example, in array processing the measurements may be observed by sensors with varying quality. In that case, the accuracy of the sensors is often known or can be estimated, allowing one to use weights that are inversely proportional to the variance of the error. Several WLS algorithms have been developed for the CPD such as an all-at-once optimization-based method [17] and a weighted ALS-based approach [22].

In this paper, we develop a WLS-based algorithm for the computation of a CPD with a weight tensor that can be modeled by a PD, enabling efficient weighting of the residuals. Note that standard least squares corresponds to a rank-1 CPD with factor vectors containing only ones. Additionally, the low-rank structure is interesting for large-scale applications because the CPD provides a compact model for the weights. The CPD structure of the weight tensor allows us to derive efficient expressions for the classical ingredients of standard qN and NLS algorithms. Special care is taken to explicitly exploit the low-rank structure in the derivation. More specifically, we focus on the implementation of a particular type of NLS algorithm, but the expressions can be used for other NLS as well as qN algorithms. We implement the algorithm using the complex optimization framework of Tensorlab, a toolbox for tensor computations in MATLAB, as numerical optimization solver [23]–[26]. Finally, the WLS-based method with CPD constrained weight tensor is illustrated for direction-of-arrival (DOA) estimation in uniform linear arrays (ULAs) with sensors of varying quality, illustrating excellent results.

We conclude this section with an overview of the notation and basic definitions. Next, we derive efficient expressions for the ingredients of standard qN and NLS methods. WLS for DOA-estimation is illustrated in Section III.

A. Notations and basic definitions

Vectors, matrices, and tensors are denoted by bold lowercase, bold uppercase, and calligraphic letters, respectively. The mode-n vector of $\mathcal{A} \in \mathbb{K}^{I_1 \times \cdots \times I_N}$ is a natural extension of the rows and columns of a matrix and is defined by fixing every index except the nth ($\mathbb{K}$ means $\mathbb{R}$ or $\mathbb{C}$). A mode-n unfolding of $\mathcal{A}$ is a matrix $\mathbf{A}_{(n)}$ with the mode-n vectors as its columns (following the ordering convention in [1]). The vectorization of $\mathcal{A}$, denoted as $\text{vec}(\mathcal{A})$, maps each element $a_{i_1 i_2 \cdots i_N}$ onto $\text{vec}(\mathcal{A})_j$ with $j = 1 + \sum_{k=1}^{N}(i_k - 1)J_k$ and $J_k = \prod_{m=1}^{k-1} I_m$.

We indicate the nth element in a sequence by a superscript between parentheses, e.g., $\{\mathbf{A}^{(n)}\}_{n=1}^{N}$. The identity matrix of size $I \times I$ is denoted by $\mathbf{I}_I$. $\mathbf{A} = \text{diag}(\mathbf{a})$ is a diagonal matrix with the entries of $\mathbf{a}$ on the main diagonal. The outer, Kronecker, Hadamard, column-, and row-wise Khatri–Rao product are denoted by $\otimes$, $\otimes$, $*$, $\odot$, $\odot^{\mathrm{T}}$, respectively.

The rank of a tensor equals the minimal number of rank-1 tensors that generate the tensor as their sum. A rank-1 tensor is defined as the outer product of nonzero vectors.

B. Canonical polyadic decomposition

The CPD is an important tensor tool in many applications within signal processing, biomedical sciences, data mining and machine learning; see [1]–[3]. The decomposition is unique under rather mild conditions [6], [15] which is a powerful advantage of tensors over matrices in many applications [3].

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 called *canonical* (CPD) when R is equal to the rank of $\mathcal{A}$.

C. Identities and derivatives

In this paper, we use the following identities [27]:

$$(\mathbf{A} \odot \mathbf{B})^{\mathrm{T}} (\mathbf{C} \odot \mathbf{D}) = \mathbf{A}^{\mathrm{T}}\mathbf{C} * \mathbf{B}^{\mathrm{T}}\mathbf{D}, \tag{1}$$

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

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

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

$$\mathbf{AB} \odot^{\mathrm{T}} \mathbf{CD} = (\mathbf{A} \odot^{\mathrm{T}} \mathbf{C}) \otimes (\mathbf{B} \otimes \mathbf{D}). \tag{5}$$

We define a permutation matrix $\mathbf{P}^{(n)}$ that permutes the nth mode of a tensor to the first mode. It holds that $\mathbf{P}^{(n)\mathrm{T}}\mathbf{P}^{(n)} = \mathbf{I}$ [28]. We use the following identity:

$$\mathbf{P}^{(n)}\mathrm{vec}\left(\left[\!\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right]\!\!\right]\right) = \mathrm{vec}\left(\left[\!\!\left[\mathbf{U}^{(n)}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n-1)}, \mathbf{U}^{(n+1)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right]\right). \tag{6}$$

Denoting $\mathbf{V}^{\{n\}} = \odot_{q=1, q\neq n}^{N} \mathbf{U}^{(N-q+1)}$, we can also define the following two identities:

$$\mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}\right) \mathrm{vec}\,(\mathbf{X}) = \mathrm{vec}\left(\left[\!\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n-1)}, \mathbf{X}, \mathbf{U}^{(n+1)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right]\right), \tag{7}$$

$$\mathbf{P}^{(n)^{\mathrm{T}}}\mathrm{vec}\left(\mathbf{U}^{(n)}\mathbf{V}^{n\mathrm{T}}\right) = \mathrm{vec}\left(\left[\!\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right]\!\!\right]\right). \tag{8}$$

Finally, we also use the following derivative [28]:

$$\frac{\partial \mathrm{vec}\left(\left[\!\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(n)} \right]\!\!\right]\right)}{\partial \mathrm{vec}\left(\mathbf{U}^{(n)}\right)} = \mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n}\right). \tag{9}$$

II. WLS for CPD using low-rank weights

The computation of a rank-R CPD $\left[\!\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right]$ of a tensor $\mathcal{T}$ using a weighted least squares approach with weight tensor $\mathcal{W}$, leads to the following optimization problem:

$$\min_{\mathbf{z}} f = \frac{1}{2} ||\mathcal{F}||_{\mathrm{F}}^2 \;\text{ with }\; \mathcal{F} = \mathcal{W} * \left(\left[\!\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right] - \mathcal{T}\right), \tag{10}$$

in which the variables $\mathbf{U}^{(n)}$, for $1 \leq n \leq N$, concatenated in a vector $\mathbf{z} = \left[\mathrm{vec}\left(\mathbf{U}^{(1)}\right);\ \mathrm{vec}\left(\mathbf{U}^{(2)}\right);\ \cdots;\ \mathrm{vec}\left(\mathbf{U}^{(N)}\right)\right] \in \mathbb{K}^{RI^+}$ with $I^+ = \sum_{n=1}^{N} I_n$. Assume $\mathcal{W}$ admits a rank-L PD:

$$\mathcal{W} = \left[\!\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\!\right] \;\text{ with }\; \mathbf{A}^{(n)} \in \mathbb{K}^{I_n \times L}. \tag{11}$$

Note that (10)-(11) reduces to regular least squares when $L = 1$ and $\mathbf{a}^{(1)} = \mathbf{1}_{I_n}$ for $1 \leq n \leq N$. As such, the rank-L PD structure enables more general weighting schemes.

In order to solve (10)-(11), one can use standard qN and NLS algorithms, which require expressions for the evaluation of the objective function, gradient, Jacobian, Jacobian-vector product, Gramian, and Gramian-vector product. In this section, we derive expressions for these ingredients which explicitly exploit the low-rank structure of the weight tensor. Although we focus on the implementation of the Gauss–Newton (GN) method, which is a particular type of NLS algorithm [29], the expressions can be used for other qN and NLS algorithms as well. We use the complex optimization framework from [23], [24] to implement the GN method. The framework provides qN and NLS implementations as well as line search, plane search, and trust-region methods. Additionally, we give an overview of the per-iteration complexity of each ingredient.

The GN method using dogleg trust-region solves (10)-(11) by linearizing the residual $\mathrm{vec}\,(\mathcal{F})$ in each iteration k and subsequently solving the following least-squares problem [29]:

$$\min_{\mathbf{p}_k} \frac{1}{2} ||\mathrm{vec}\,(\mathcal{F}_k) + \mathbf{J}_k\mathbf{p}_k||_{\mathrm{F}}^2 \quad \text{s.t.} \quad ||\mathbf{p}_k|| \leq \Delta_k \tag{12}$$

with step $\mathbf{p}_k = \mathbf{z}_{k+1} - \mathbf{z}_k$, Jacobian $\mathbf{J} = \partial \mathrm{vec}\,(\mathcal{F})\,/\partial \mathbf{z}$, and trust-region Δ_k. The exact solution to (12) is given by the linear system $\mathbf{H}_k\mathbf{p}_k = -\overline{\mathbf{g}}_k$ with $\mathbf{H}$ the Gramian of the Jacobian, which serves as an approximation for the Hessian, and the conjugated gradient $\overline{\mathbf{g}} = (\partial f/\partial \mathbf{z})^{\mathrm{H}}$ [29]. 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 (PCG) iterations in order to reduce computational complexity. The GN method is summarized in Algorithm 1.

Algorithm 1: WLS with CPD constrained weight tensor using Gauss–Newton and dogleg trust region

Input: $\mathcal{T}$, $\{\mathbf{A}^{(n)}\}_{n=1}^{N}$, 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 $\mathbf{Hp} = -\overline{\mathbf{g}}$ for $\mathbf{p}$ using Gramian-vector products in (18) and a block-Jacobi preconditioner as explained in Section II-E.
  Update $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$, using dogleg trust region from $\mathbf{p}$, $\mathbf{g}$, and objective function evaluation (13).
end

A. Objective function

The objective function f can be evaluated as follows:

$$f = \frac{1}{2} \left|\left| \left[\!\!\left[\tilde{\mathbf{U}}^{(1)}, \tilde{\mathbf{U}}^{(2)}, \ldots, \tilde{\mathbf{U}}^{(N)} \right]\!\!\right] - \tilde{\mathcal{T}} \right|\right|_{\mathrm{F}}^2, \tag{13}$$

using the weighted factor matrices $\tilde{\mathbf{U}}^{(n)} \in \mathbb{K}^{I_n \times RL}$ defined by $\tilde{\mathbf{U}}^{(n)} = \mathbf{A}^{(n)} \odot^{\mathrm{T}} \mathbf{U}^{(n)}$, for $1 \leq n \leq N$. Importantly, the weighted factor matrices are computed only once per iteration. The weighted tensor $\tilde{\mathcal{T}} = \mathcal{W} * \mathcal{T}$ can be computed beforehand.

B. Jacobian

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

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

with $I^{\times} = \prod_{n=1}^{N} I_n$. The nth sub-Jacobian $\mathbf{J}^{(n)}$ is defined by:

$$\mathbf{J}^{(n)} = \mathbf{P}^{(n)^{\mathrm{T}}} \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathbf{M}^{(n)} \in \mathbb{K}^{I^{\times} \times RI_n},$$

using $\tilde{\mathbf{V}}^{\{n\}} = \odot_{q=1, q \neq n}^{N} \tilde{\mathbf{U}}^{(N-n+1)}$ and the matrix $\mathbf{M}^{(n)}$:

$$\mathbf{M}^{(n)} = \begin{bmatrix} \mathbf{I}_R \otimes \mathrm{diag}\left(\mathbf{a}_1^{(n)}\right) \\ \mathbf{I}_R \otimes \mathrm{diag}\left(\mathbf{a}_2^{(n)}\right) \\ \vdots \\ \mathbf{I}_R \otimes \mathrm{diag}\left(\mathbf{a}_L^{(n)}\right) \end{bmatrix} \in \mathbb{K}^{RLI_N \times RI_n}. \tag{14}$$

Proof. First, define matrix $\mathbf{D} = \mathrm{diag}\left(\mathrm{vec}\left(\mathcal{W}\right)\right) \in \mathbb{K}^{I^{\times} \times I^{\times}}$. Using (9), one can show that nth sub-Jacobian equals:

$$\begin{aligned} \mathbf{J}^{(n)} &= \frac{\partial \mathrm{vec}\left(\mathcal{F}\right)}{\partial \mathrm{vec}\left(\mathbf{U}^{(n)}\right)} = \mathbf{D} \frac{\partial \mathrm{vec}\left(\left[\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}\right]\!\right]\right)}{\partial \mathrm{vec}\left(\mathbf{U}^{(n)}\right)} \\ &= \mathbf{D} \mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n} \right). \end{aligned}$$

Define $\mathbf{B}^{\{n\}} = \odot_{q=1, q \neq n}^{N} \mathbf{A}^{(N-n+1)}$. The CPD structure of the weight tensor $\mathcal{W}$ can then be exploited as follows:

$$\begin{aligned} \mathbf{J}^{(n)} &= \mathrm{diag}\left(\mathrm{vec}\left(\left[\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\right] \right) \right) \\ &\quad \cdot \mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \\ &= \mathrm{diag}\left(\mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{B}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathrm{vec}\left(\mathbf{A}^{(n)} \right) \right) \\ &\quad \cdot \mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \\ &= \left[\mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{B}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathrm{vec}\left(\mathbf{A}^{(n)} \right) \right] \\ &\quad \odot^{\mathrm{T}} \left[\mathbf{P}^{(n)^{\mathrm{T}}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \right] \\ &= \mathbf{P}^{(n)^{\mathrm{T}}} \left(\left(\mathbf{B}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \odot^{\mathrm{T}} \left(\mathbf{V}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \right) \\ &\quad \cdot \left[\mathrm{vec}\left(\mathbf{A}^{(n)} \right) \otimes \mathbf{I}_{RI_n} \right] \\ &= \mathbf{P}^{(n)^{\mathrm{T}}} \left(\left(\mathbf{B}^{\{n\}} \odot^{\mathrm{T}} \mathbf{V}^{\{n\}} \right) \otimes \mathbf{I}_{I_n} \right) \mathbf{M}^{(n)} \\ &= \mathbf{P}^{(n)^{\mathrm{T}}} \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathbf{M}^{(n)}. \end{aligned}$$

Identities (8) and (5) are used to obtain the second and fourth equation. One can obtain the second to last equation using (3) and (14). Note that we do not explicitly compute the Jacobian.

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:

$$\mathbf{g}^{(n)} = \mathbf{M}^{(n)^{\mathrm{T}}} \mathrm{vec}\left(\tilde{\mathbf{U}}^{(n)} \tilde{\mathbf{V}}^{\{n\}^{\mathrm{T}}} \tilde{\mathbf{V}}^{\{n\}} - \tilde{\mathbf{T}}_{(n)} \tilde{\mathbf{V}}^{\{n\}} \right). \tag{16}$$

Proof. The nth sub-gradient is given by:

$$\begin{aligned} \mathbf{g}^{(n)} &= \frac{\partial f}{\partial \mathrm{vec}\left(\mathbf{U}^{(n)}\right)} = \mathbf{J}^{(n)^{\mathrm{T}}} \overline{\mathrm{vec}\left(\mathcal{F}\right)} \\ &= \mathbf{M}^{(n)^{\mathrm{T}}} \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right)^{\mathrm{T}} \mathbf{P}^{(n)} \\ &\quad \cdot \mathbf{P}^{(n)^{\mathrm{T}}} \overline{\mathrm{vec}\left(\tilde{\mathbf{U}}^{(n)} \tilde{\mathbf{V}}^{\{n\}^{\mathrm{T}}} - \tilde{\mathbf{T}}_{(n)} \right)} \\ &= \mathbf{M}^{(n)^{\mathrm{T}}} \Big[\left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right)^{\mathrm{T}} \mathrm{vec}\left(\overline{\tilde{\mathbf{U}}^{(n)}} \tilde{\mathbf{V}}^{\{n\}^{\mathrm{H}}} \right) \cdots \\ &\quad - \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right)^{\mathrm{T}} \overline{\tilde{\mathbf{T}}_{(n)}} \Big] \\ &= \mathbf{M}^{(n)^{\mathrm{T}}} \mathrm{vec}\left(\overline{\tilde{\mathbf{U}}^{(n)}} \tilde{\mathbf{V}}^{\{n\}^{\mathrm{H}}} \tilde{\mathbf{V}}^{\{n\}} - \overline{\tilde{\mathbf{T}}_{(n)}} \tilde{\mathbf{V}}^{\{n\}} \right). \end{aligned}$$

using identities (4) and (7) to obtain the last equation. The matrices $\mathbf{M}^{(n)}$, $1 \leq n \leq N$, only depend on a factor matrix of the weight tensor and can therefore be computed beforehand.

D. Gramian-vector product

First, we derive an efficient way to compute the product of a sub-Jacobian $\mathbf{J}^{(n)}$ and $\mathrm{vec}\left(\mathbf{X}^{(n)}\right)$ with $\mathbf{X}^{(n)} \in \mathbb{K}^{I_n \times R}$:

$$\begin{aligned} \mathbf{J}^{(n)} \mathrm{vec}\left(\mathbf{X}^{(n)}\right) &= \mathbf{P}^{(n)^{\mathrm{T}}} \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathbf{M}^{(n)} \mathrm{vec}\left(\mathbf{X}^{(n)}\right) \\ &= \mathbf{P}^{(n)^{\mathrm{T}}} \left(\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n} \right) \mathrm{vec}\left(\tilde{\mathbf{X}}^{(n)}\right) \\ &= \mathrm{vec}\left(\left[\!\left[\tilde{\mathbf{U}}^{(1)}, \ldots, \tilde{\mathbf{U}}^{(n-1)}, \tilde{\mathbf{X}}^{(n)}, \tilde{\mathbf{U}}^{(n+1)}, \ldots, \tilde{\mathbf{U}}^{(N)} \right]\!\right] \right) \end{aligned} \tag{17}$$

with $\tilde{\mathbf{X}}^{(n)} = \mathbf{A}^{(n)} \odot^{\mathrm{T}} \mathbf{X}^{(n)}$ using identity (8) to obtain the last equation. Next, we derive an efficient expression for the Gramian-vector product $\mathbf{r} = \mathbf{J}^{(m)^{\mathrm{H}}} \mathbf{J}^{(n)} \mathrm{vec}\left(\mathbf{X}^{(n)}\right)$:

$$\begin{aligned} \mathbf{r} &= \mathbf{M}^{(m)^{\mathrm{H}}} \left(\tilde{\mathbf{V}}^{\{m\}} \otimes \mathbf{I}_{I_m} \right)^{\mathrm{H}} \mathbf{P}^{(m)} \\ &\quad \cdot \mathrm{vec}\left(\left[\!\left[\tilde{\mathbf{U}}^{(1)}, \ldots, \tilde{\mathbf{U}}^{(n-1)}, \tilde{\mathbf{X}}^{(n)}, \tilde{\mathbf{U}}^{(n+1)}, \ldots, \tilde{\mathbf{U}}^{(N)} \right]\!\right] \right) \\ &= \mathbf{M}^{(m)^{\mathrm{H}}} \left(\tilde{\mathbf{V}}^{\{m\}} \otimes \mathbf{I}_{I_m} \right)^{\mathrm{H}} \\ &\quad \cdot \mathrm{vec}\left(\left[\!\left[\tilde{\mathbf{U}}^{(m)}, \tilde{\mathbf{U}}^{(1)}, \ldots, \tilde{\mathbf{X}}^{(n)}, \ldots, \tilde{\mathbf{U}}^{(N)} \right]\!\right] \right) \\ &= \mathbf{M}^{(m)^{\mathrm{H}}} \left(\hat{\mathbf{V}}^{\{m\}} \otimes \mathbf{I}_{I_m} \right) \mathrm{vec}\left(\tilde{\mathbf{U}}^{(m)} \right) \\ &= \mathbf{M}^{(m)^{\mathrm{H}}} \left(\tilde{\mathbf{V}}^{\{m\}^{\mathrm{H}}} \hat{\mathbf{V}}^{\{m\}} \otimes \mathbf{I}_{I_m} \right) \mathrm{vec}\left(\tilde{\mathbf{U}}^{(m)} \right) \end{aligned} \tag{18}$$

with $\hat{\mathbf{V}}^{\{m\}} = \tilde{\mathbf{U}}^{(N)} \odot \cdots \odot \tilde{\mathbf{U}}^{(n+1)} \odot \tilde{\mathbf{X}}^{(n)} \odot \tilde{\mathbf{U}}^{(n-1)} \odot \cdots \odot \tilde{\mathbf{U}}^{(1)}$ in which the superscript m indicates that the mth factor matrix is omitted in the Khathri–Rao product. We used (17), (6), (7) and (2) in the subsequent steps of (18). If $m = n$, one can show that (18) reduces to:

$$\mathbf{J}^{(n)^{\mathrm{H}}} \mathbf{J}^{(n)} \mathrm{vec}\left(\mathbf{X}^{(n)}\right) = \mathbf{M}^{(n)^{\mathrm{H}}} \mathrm{vec}\left(\tilde{\mathbf{X}}^{(n)} \mathbf{Q}^{(n)^{\mathrm{T}}} \right).$$

TABLE I
PER-ITERATION COMPUTATIONAL COMPLEXITY OF THE INGREDIENTS OF THE NLS ALGORITHM.

	Calls/iteration	Complexity
Weighted factor matrices	1	$\mathcal{O}(NRLI)$
Objective function	1 + it_{TR}	$\mathcal{O}(RLI^N)$
Gradient	1	$\mathcal{O}(NRLI^N + NR^2LI^2)$
Gramian-vector	it_{CG}	$\mathcal{O}(NR^2L^2I + NR^2LI^2)$

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, one has 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. First, we derive an expression for the (n,n)th sub-Gramian:

$$\begin{aligned}\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)} &= \mathbf{M}^{(n)^{\text{H}}}\left(\tilde{\mathbf{V}}^{(n)} \otimes \mathbf{I}_{I_n}\right)^{\text{H}}\left(\tilde{\mathbf{V}}^{(n)} \otimes \mathbf{I}_{I_n}\right)\mathbf{M}^{(n)} \\ &= \mathbf{M}^{(n)^{\text{H}}}\left(\tilde{\mathbf{V}}^{\{n\}^{\text{T}}}\tilde{\mathbf{V}}^{\{n\}} \otimes \mathbf{I}_{I_n}\right)\mathbf{M}^{(n)} \\ &= \mathbf{M}^{(n)^{\text{H}}}\left(\mathbf{Q}^{\{n\}} \otimes \mathbf{I}_{I_n}\right)\mathbf{M}^{(n)}.\end{aligned}$$

with $\mathbf{Q}^{\{n\}} = \tilde{\mathbf{V}}^{\{n\}^{\text{T}}}\tilde{\mathbf{V}}^{\{n\}} \in \mathbb{K}^{RL \times RL}$. Using identity (1), we obtain $\mathbf{Q}^{\{n\}} = *_{q=1,q\neq n}^{N} \tilde{\mathbf{U}}^{(N-q+1)^{\text{H}}}\tilde{\mathbf{U}}^{(N-q+1)} \in \mathbb{K}^{RL \times RL}$. The inverse of $\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)}$ can then be computed efficiently as:

$$\left(\mathbf{J}^{(n)^{\text{H}}}\mathbf{J}^{(n)}\right)^{\dagger} = \left(\mathbf{M}^{(n)}\right)^{\dagger}\left(\left(\mathbf{Q}^{\{n\}}\right)^{\dagger} \otimes \mathbf{I}_{I_n}\right)\left(\mathbf{M}^{(n)}\right)^{\dagger,\text{H}},$$

in which $\left(\mathbf{M}^{(n)}\right)^{\dagger}$ can be computed beforehand and $\left(\mathbf{Q}^{\{n\}}\right)^{\dagger}$ requires the inverse of small matrices of size $(RL \times RL)$.

F. Complexity

In Table I, we report the per-iteration complexity for the ingredients of Algorithm 1. We assume that $I_n = I$ for $1 \leq n \leq N$. The complexity is similar to regular LS algorithms. In comparison to [16], the factor R is replaced by RL and there is an additional cost for the multiplication with $\mathbf{M}^{(n)}$.

III. DIRECTION-OF-ARRIVAL ESTIMATION USING WLS

Direction-of-arrival (DOA) estimation is an important problem in radar, sonar, and telecommunication applications [30]. We show that the performance can be improved by including prior knowledge about the accuracy of the sensors via WLS.

When considering a uniform linear array (ULA) with line-of-sight signals impinging from the far field, the DOA estimation problem can be reformulated as the computation of a CPD [11], [12], [31], [32]. Consider a ULA with M uniformly spaced and omnidirectional sensors receiving length-K signals from R narrow-band sources in the far field. In that case, the problem can be described by $\mathbf{X} = \mathbf{MS}$ in which $\mathbf{X} \in \mathbb{K}^{M \times K}$ contains the observed measurements, $\mathbf{M} \in \mathbb{K}^{M \times R}$ contains the so-called steering vectors, and $\mathbf{S} \in \mathbb{K}^{R \times K}$ contains the sources. The steering vectors are defined element-wise as $m_{mr} = \theta_r^{m-1}$ with $\theta_r = e^{-2\pi i \Delta \sin(\alpha_r)\lambda^{-1}}$. The inter-element spacing is denoted by Δ, the rth DOA with respect to the normal is denoted by α_r (i.e., -90°$\leq \alpha_r \leq$ 90°), and the wavelength is denoted by λ. One way to obtain a tensor is to reshape $\mathbf{X}$ into a third-order tensor $\mathcal{X} \in \mathbb{K}^{I \times J \times K}$ such that $M = IJ$ using segmentation; see [9], [33]. The obtained tensor $\mathcal{X}$ admits a CPD $[\![\mathbf{A}, \mathbf{B}, \mathbf{S}]\!]$ in which $\mathbf{A}$ and $\mathbf{B}$ are Vandermonde matrices from which the DOAs can be extracted.

TABLE II
INCORPORATING PRIOR KNOWLEDGE ABOUT THE ACCURACY OF THE SENSORS IN DOA ESTIMATION USING WEIGHTED LEAST SQUARES ALLOWS ONE TO DECREASE THE RELATIVE ERROR ON THE DOAS.

	Least squares	Weighted least squares	
		rank-1	rank-2
Relative error	0.0322	0.0187	**0.0122**

Consider a ULA with $M = 25$ sensors and $R = 2$ source signals of length $K = 100$. Assume there are two types of sensors with different quality: the SNR on the first seventeen sensors is 25 dB and the SNR on the last eight sensors is 5 dB. In order to accommodate for the difference in SNR between the sensors, we use the following weight tensor $\mathcal{W} = \text{unvec}(\mathbf{w})$ with $\mathbf{w}_{(k-1)M+1:(k-1)M+17} = \mathbf{1}_{17}$ and $\mathbf{w}_{(k-1)M+18:(k-1)M+25} = 0.01 \cdot \mathbf{1}_8$ for $1 \leq k \leq K$. We model the weight tensor with a rank-L CPD using $L = \{1, 2\}$; note that the weight tensor has rank two. The relative error $\epsilon_{\boldsymbol{\alpha}}$ of the DOAs is defined by $||\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}||_{\text{F}} / ||\boldsymbol{\alpha}||_{\text{F}}$ with $\boldsymbol{\alpha}$ a vector containing the R DOAs. In Table II we report the median relative error across 50 experiments. It is clear that WLS outperforms LS, even when using a rank-1 model for the weight tensor. Also, using an additional rank-1 term to model the weight tensor further decreases the relative error.

IV. CONCLUSION

A new WLS-based approach has been proposed for the computation of a CPD using low-rank weights. In particular, the weight tensor was modeled as a PD, enabling efficient weighting schemes. We have derived expressions for the ingredients of well-known qN and NLS algorithms that carefully exploit the CPD structure of the weight tensor. Also, a complexity analysis was performed for the NLS-type algorithms. We validated our algorithm for DOA estimation in ULAs, outperforming regular least squares. In future work, one can derive WLS-based algorithms for other tensor decompositions. Additionally, other tensor models for the weight tensor can be considered such as the multilinear singular value decomposition, tensor train, or hierachical Tucker model [34]–[36].

ACKNOWLEDGMENT

(1) Research Council KU Leuven: C1 project C16/15/059-nD, (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] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[2] 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.

[3] 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.

[4] I. Domanov and L. De Lathauwer, "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*, vol. 34, no. 3, pp. 876–903, July-Sept. 2013.

[5] ——, "Generic uniqueness conditions for the canonical polyadic decomposition and INDSCAL," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 4, pp. 1567–1589, Nov. 2015.

[6] ——, "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.

[7] P. Comon and C. Jutten, *Handbook of blind source separation: Independent component analysis and applications*. Academic press, 2009.

[8] L. De Lathauwer, "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*, vol. 32, no. 4, pp. 1451–1474, Dec. 2011.

[9] 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.

[10] ——, "Tensor-based large-scale blind system identification using segmentation," *Technical Report 16-109, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2016.

[11] N. Sidiropoulos, R. Bro, and G. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2377–2388, Aug. 2000.

[12] S. Miron, Y. Song, D. Brie, and K. Wong, "Multilinear direction finding for sensor-array with multiple scales of invariance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 2057–2070, July 2015.

[13] L. De Lathauwer and A. De Baynast, "Blind deconvolution of DS-CDMA signals by means of decomposition in rank-$(1, L, L)$ terms," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1562–1571, Apr. 2008.

[14] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, Sept. 2006.

[15] 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.

[16] 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.

[17] 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.

[18] G. Tomasi and R. Bro, "PARAFAC and missing values," *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, Feb. 2005.

[19] A.-H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped gauss–newton algorithms for CANDECOMP/PARAFAC," *SIAM Journal of Matrix Analysis and Applications*, vol. 34, no. 1, pp. 126–147, 2013.

[20] E. Acar, D. Dunlavy, and T. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[21] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. John Wiley & Sons, 2012.

[22] G. Hollander, P. Dreesen, M. Ishteva, and J. Schoukens, "Weighted tensor decomposition for approximate decoupling of multivariate polynomials," *Technical Report*, 2016, arXiv:1601.07800.

[23] 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.

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

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

[26] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization," in *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers (Pacific Grove, CA)*, Nov. 2016, pp. 1733–1738.

[27] 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.

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

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

[30] H. Krim and M. Viberg, "Two decades of array signal processing: The parametric approach," *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 67–94, 1996.

[31] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 7, pp. 984–995, 1989.

[32] N. Sidiropoulos, "Generalizing Carathéodory's uniqueness of harmonic parametrization to N dimensions," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1687–1690, May 2001.

[33] O. Debals and L. De Lathauwer, "Stochastic and deterministic tensorization for blind signal separation," in *Latent Variable Analysis and Signal Separation*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2015, vol. 9237, pp. 3–13.

[34] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, Apr. 2000.

[35] I. Oseledets, "Tensor-train decomposition," *SIAM Journal of Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, Sept. 2011.

[36] L. Grasedyck, "Polynomial approximation in hierarchical Tucker format by vector tensorization," Apr. 2010, preprint 43, DFG/SPP1324, RWTH Aachen.