KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

| | |
|---|---|
| **Citation/Reference** | Vermeylen C., Vervliet N., De Lathauwer L. (2026),<br>**Escaping swamps through complex space for real canonical polyadic decomposition**<br><br>IEEE Signal Processing Letters, March 2026. |
| **Archived version** | Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher |
| **Published version** | https://doi.org/10.1109/LSP.2026.3676367 |
| **Journal homepage** | https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=97 |
| **Author contact** | charlotte.vermeylen@kuleuven.be<br>*+ 32 (0)4 78 61 38 71* |
| **Abstract** | The canonical polyadic decomposition (CPD) can be used as a low-rank representation of tensors appearing in various applications. However, obtaining a CPD can lead to a difficult optimization problem when, e.g., the rank is higher than some of the tensor dimensions. During the optimization, a degenerate subtensor can then be formed, i.e., an ill-conditioned tensor that tries to approximate a tensor of higher rank. This can lead to regions in the optimization landscape of very slow convergence, informally known as 'swamps'. An efficient optimization framework is proposed to detect the degenerate terms and let them become complex using a parameterization of the factor vectors in their real and imaginary parts. This gives more freedom to the decomposition during convergence. The eventual solution remains real-valued because of the uniqueness of the CPD. We illustrate the approach for difficult tensors that appear in underdetermined independent component analysis, showing significantly improved accuracy and computational efficiency. |
| **IR** | url in Lirias https://lirias.kuleuven.be/4411570?limo=0 |

*(article begins on next page)*

# Escaping swamps through complex space for real canonical polyadic decomposition

C. Vermeylen, N. Vervliet, and L. De Lathauwer, *Fellow, IEEE*

***Abstract*—The canonical polyadic decomposition (CPD) can be used as a low-rank representation of tensors appearing in various applications. However, obtaining a CPD can lead to a difficult optimization problem when, e.g., the rank is higher than some of the tensor dimensions. During the optimization, a degenerate subtensor can then be formed, i.e., an ill-conditioned tensor that tries to approximate a tensor of higher rank. This can lead to regions in the optimization landscape of very slow convergence, informally known as 'swamps'. An efficient optimization framework is proposed to detect the degenerate terms and let them become complex using a parameterization of the factor vectors in their real and imaginary parts. This gives more freedom to the decomposition during convergence. The eventual solution remains real-valued because of the uniqueness of the CPD. We illustrate the approach for difficult tensors that appear in underdetermined independent component analysis, showing significantly improved accuracy and computational efficiency.**

***Index Terms*—canonical polyadic decomposition, complex numbers, numerical optimization, parameterization, higher-order tensors, independent component analysis**

## I. Introduction

Higher-order tensors are increasingly encountered in a wide range of applications, including neural networks, higher-order derivative optimization, image processing, and data analysis [1]–[11]. Many applications require the tensor to be decomposed into smaller terms that are easy to interpret. The canonical polyadic decomposition (CPD) can be used for this purpose, as it gives a unique interpretation under mild assumptions [12]–[14]. The (C)PD decomposes a tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ into a (minimal) sum of $R$ rank-1 tensors [15]:

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r =: [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!], \tag{1}$$

where $\mathbf{a}_r \in \mathbb{R}^I, \mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$, and '$\otimes$' denotes the outer product. The *factor matrices* $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ collect these factor vectors column-wise per mode. The minimum $R$ such that (1) holds is the *(CP)-rank* of the tensor. The (C)PD and the algorithm that is presented can be naturally extended to tensors of arbitrary *order* or number of dimensions.

This work was funded by: (1) the Flemish Government, under the AI Research Program. C. V. and L. D. L. are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. (2) KU Leuven Internal Funds: iBOF/23/064, and C14/22/096. N. V. holds a research grant from the Research Foundation Flanders (FWO) (12ZM223N).

All authors are affiliated to the Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10 bus 2246, Leuven B-3001, Belgium (e-mail: charlotte.vermeylen@kuleuven.be). L. D. L. is affiliated to Subfaculty Science and Technology, KULAK, E. Sabbelaan 53, Kortrijk B-8500, Belgium.

To obtain a CPD or CP approximation of a tensor, the following nonlinear least-squares (NLS) problem can be solved:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \underbrace{\frac{1}{2}\left\| [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T} \right\|^2}_{=:f(\mathbf{x})}, \tag{2}$$

where $\|\cdot\|$ denotes the Frobenius norm and $\mathbf{x} := \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$. When $f = 0$, the decomposition is exact. The problem becomes more challenging, for instance, when the rank is higher than one of the dimensions or when there are approximate linear dependencies among the factor vectors.

For tensors that are more difficult to decompose, it is well known that second-order, *all-at-once* algorithms, which optimize over all factors at once, converge much faster than alternating methods, such as the alternating least-squares (ALS) method [16]–[19]. Gauss–Newton (GN) type algorithms are an effective class of these methods [18], [19]. Efficient implementations for the CPD exist [19] and are implemented in toolboxes such as Tensorlab [20].

A complication in the optimization of (2) for more difficult tensors is the existence of *degenerate* tensors, i.e., ill-conditioned sets of rank-1 terms that approximate tensors of higher rank arbitrary well [21], [22], which is closely related to border-rank PDs [23]. A well-known example is when two terms diverge to infinity and almost become each other's opposite, largely canceling each other [24], [25], which is also observed when the rank is higher than two [26]. Even when the degeneracy is not intrinsic to the eventual solution, it can happen that an algorithm passes through such a region.

These degenerate tensors lead to regions of very slow convergence with almost no improvement in the cost function value for several hundreds or thousands of iterations—informally known as *swamps*. Swamps have been studied extensively for the ALS method [24], [25], [27], [28], which is particularly prone to them, but they are also observed for second-order, all-at-once algorithms [26]. This paper focuses on escaping *transient swamps* [24], [25], which are not caused by the solution itself, i.e., the asymptotic convergence toward the eventual optimum is fast.

Standard techniques to avoid swamps are adding regularization terms or other forms of constraints [25], [28]. However, for certain applications, such measures can be unnatural as they alter the original optimum, and solving the modified problems may require more advanced algorithms and careful parameter tuning. Recently, an optimization framework for (2) has been proposed that focuses on the degenerate terms when encountered and temporarily freezes the other terms [26]. The degenerate terms are replaced by an algebraic reinitialization,

which may cause an increase in cost function value. Another CPD algorithm employing a different reinitialization strategy was proposed in [16]. However, there is no guarantee that these reinitializations avoid new swamps.

This paper proposes to let the problematic terms become complex when detected, which gives more 'freedom' to the decomposition, such that the iterates can escape the swamp without increasing the cost function value or changing the optimum, and without relying on unpredictable reinitializations. The additional freedom can be understood in the sense that there are more free parameters to decompose the tensor. Under mild conditions, uniqueness remains to hold over $\mathbb{C}$ [12]. Consequently, the real-valued solution remains the eventual solution, and the transition from $\mathbb{R}$ to $\mathbb{C}$ is only a temporary computational choice. Note that the cost per iteration increases due to the increase in parameters but this is compensated by the higher success rate and lower number of iterations.

*Organization:* Section II gives preliminaries concerning notation and the GN method for CPD. Section III details the swamp detection, the partial transition to $\mathbb{C}$, and the computation in the complex format. Section IV presents a numerical experiment on underdetermined *independent component analysis (ICA)* which illustrates that for difficult tensors whose rank exceeds their dimensions, the proposed algorithm is significantly more successful in recovering the underlying decomposition than state-of-the-art GN methods. Because of the higher success rate and the efficiency of the algorithm in detecting degenerate terms, the *expected time to success (ETC)* is decreased significantly.

## II. Preliminaries

*Notation:* Lowercase, bold, bold uppercase, and calligraphic letters are used to distinguish between scalars, vectors, matrices, and tensors, respectively, e.g., $a$, $\mathbf{a}$, $\mathbf{A}$, and $\mathcal{A}$. A (C)PD is parameterized as an ordered collection of factor matrices: $\mathbf{x} := \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$. The $i$th element of $\mathbf{a}$ is given by $a_i$ and the $j$th column of $\mathbf{A}$ by $\mathbf{a}_j$. If $\mathcal{C} \subseteq \{1, \cdots, R\}$ is an ordered set, then $\mathbf{A}_\mathcal{C}$ denotes the submatrix of $\mathbf{A}$ consisting of the columns indexed by $\mathcal{C}$. '$\otimes$', '$\otimes$', and '$\odot$' denote the tensor or outer product, the Kronecker product, and the Khatri–Rao or column-wise Kronecker product, respectively. $\mathrm{vec}\,(\cdot)$ is the vectorization operator and is defined such that $\mathrm{vec}\,(\mathcal{T})\,(i + (j-1)J + (k-1)JK) = \mathcal{T}(i,j,k)$. Vectorization of a set of factor matrices is defined element-wise: $\mathrm{vec}(\mathbf{x}) := \{\mathrm{vec}(\mathbf{A}), \mathrm{vec}(\mathbf{B}), \mathrm{vec}(\mathbf{C})\}$. The identity matrix is denoted by $\mathbf{I}$ and the imaginary unit by $\jmath$. The Hermitian transpose is denoted by $\cdot^{\mathrm{H}}$. $\mathrm{E}[\cdot]$ denotes the expected value.

*Gauss–Newton method for the CPD:* The GN method is a second-order NLS optimization method which approximates the second-order derivatives by the GN Gramian $\mathbf{J}^{\mathrm{H}}\mathbf{J}$ of the Jacobian matrix $\mathbf{J} = \mathbf{J}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ [29], which is for (2) [19]:

$$\mathbf{J} = \begin{bmatrix} (\mathbf{C} \odot \mathbf{B}) \otimes \mathbf{I} & ((\mathbf{C} \odot \mathbf{A}) \otimes \mathbf{I})\,\mathbf{P} & \mathbf{I} \otimes (\mathbf{B} \odot \mathbf{A}) \end{bmatrix}, \quad (3)$$

where $\mathbf{P}$ is a permutation matrix such that

$$\begin{bmatrix} \mathbf{c}_1 \otimes \mathbf{I} \otimes \mathbf{a}_1 & \cdots & \mathbf{c}_R \otimes \mathbf{I} \otimes \mathbf{a}_R \end{bmatrix} = ((\mathbf{C} \odot \mathbf{A}) \otimes \mathbf{I})\,\mathbf{P}.$$

$\mathbf{J}$ has size $IJK \times (I+J+K)R$ and a null-space of dimension $2R$ because of scaling indeterminacies. More specifically, the columns of a rank-1 term can be scaled as:

$$\alpha \mathrm{e}^{\jmath\phi}\mathbf{a}_i,\ \beta \mathrm{e}^{\jmath\psi}\mathbf{b}_i,\ \frac{1}{\alpha\beta}\mathrm{e}^{-\jmath(\phi+\psi)}\mathbf{c}_i, \quad \forall\ \alpha, \beta \neq 0, \forall\ \phi, \psi. \quad (4)$$

The gradient of (2) is given by $\nabla f(\mathbf{x}) = -\mathbf{J}^{\mathrm{H}}\mathrm{vec}\,(\llbracket \mathbf{x} \rrbracket - \mathcal{T})$. To obtain the step $\mathbf{p}$, the system $\mathbf{J}^\top\mathbf{J} \cdot \mathbf{p} = \nabla f(\mathbf{x})$ is solved. The new point $\mathbf{x}_{\mathrm{new}}$ can be obtained as $\mathrm{vec}(\mathbf{x}) + \alpha\mathbf{p}$, for some $\alpha > 0$, which can be obtained with a line-search scheme. Alternatively, $\mathbf{x}_{\mathrm{new}}$ can be computed using a trust-region method based on $\mathbf{p}$ and $\mathbf{x}$ [29].

## III. Parameterized complex CPD optimization framework for real tensors

We first discuss how terms are made complex while others are kept real. This is achieved by parameterizing the complex terms with two parameters per rank-1 vector: one for the real part and one for the imaginary part. The Jacobian and gradient formulations of the parameterization are given. Afterwards, the detection of degenerate terms is discussed and how the new variables are initialized with a nonzero imaginary part. Lastly, the full framework is given.

*Complex parameterization:* Let the problematic rank-1 terms, i.e., those that need to be made complex, correspond to the indices in $\mathcal{C}$, then the factor vectors can be written as $\mathbf{a}_i = \mathbf{d}_i + \jmath\mathbf{u}_i$ with $\mathbf{u}_i = \mathbf{0}$ if $i \notin \mathcal{C}$, and similarly for $\mathbf{b}_i = \mathbf{e}_i + \jmath\mathbf{v}_i$ and $\mathbf{c}_i = \mathbf{f}_i + \jmath\mathbf{w}_i$. Let $R_{\mathrm{c}} := |\mathcal{C}| \leqslant R$ be the number of complex terms. The parameterization then becomes:

$$\mathbf{x}_{\mathrm{p}} := \{\mathbf{D},\ \mathbf{E},\ \mathbf{F},\ \mathbf{U}_\mathcal{C},\ \mathbf{V}_\mathcal{C},\ \mathbf{W}_\mathcal{C}\}. \quad (5)$$

*Jacobian and gradient formulations:* The Jacobian of the NLS problem (2) w.r.t. the parameterization (5) is:

$$\mathbf{J}_{\mathrm{p}} = \begin{bmatrix} \mathbf{J}(\mathbf{A}, \mathbf{B}, \mathbf{C}) & \jmath\mathbf{J}_\mathcal{C}(\mathbf{A}, \mathbf{B}, \mathbf{C}) \end{bmatrix},$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are complex-valued, and $\mathbf{J}_\mathcal{C}(\mathbf{A}, \mathbf{B}, \mathbf{C}) := \mathbf{J}(\mathbf{A}_\mathcal{C}, \mathbf{B}_\mathcal{C}, \mathbf{C}_\mathcal{C})$. Because $\mathbf{J}_{\mathrm{p}}$ is complex-valued, but $f$ is analytic in $\mathbf{x}_{\mathrm{p}}$, $\mathbf{J}_{\mathrm{r}} := \begin{bmatrix} \mathrm{Re}\,(\mathbf{J}_{\mathrm{p}}) \\ \mathrm{Im}\,(\mathbf{J}_{\mathrm{p}}) \end{bmatrix}$ is used [30]. Because $\mathbf{J}_{\mathrm{r}}^\top\mathbf{J}_{\mathrm{r}} = \mathrm{Re}\,(\mathbf{J}_{\mathrm{p}}^{\mathrm{H}}\mathbf{J}_{\mathrm{p}})$, the Gramian of the parameterization can be computed based on efficient implementations for the standard (complex) Gramian $\mathbf{J}^{\mathrm{H}}\mathbf{J}$:

$$\begin{aligned} \mathbf{J}_{\mathrm{p}}^{\mathrm{H}}\mathbf{J}_{\mathrm{p}} &= \begin{bmatrix} \mathbf{J}^{\mathrm{H}}\mathbf{J} & \jmath\mathbf{J}^{\mathrm{H}}\mathbf{J}_\mathcal{C} \\ -\jmath\mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{J} & \mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{J}_\mathcal{C} \end{bmatrix} \\ \Leftrightarrow\ \mathbf{J}_{\mathrm{r}}^\top\mathbf{J}_{\mathrm{r}} &= \begin{bmatrix} \mathrm{Re}\,(\mathbf{J}^{\mathrm{H}}\mathbf{J}) & -\mathrm{Im}\,(\mathbf{J}^{\mathrm{H}}\mathbf{J}_\mathcal{C}) \\ \mathrm{Im}\,(\mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{J}) & \mathrm{Re}\,(\mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{J}_\mathcal{C}) \end{bmatrix}, \end{aligned} \quad (6)$$

where each block can be obtained directly from $\mathbf{J}^{\mathrm{H}}\mathbf{J}$. The real-valued gradient of the parameterization in (5) is [30]:

$$\nabla_{\mathrm{r}} f(\mathbf{x}) = \mathrm{Re}\,(\mathbf{J}_{\mathrm{p}}^{\mathrm{H}}\mathbf{f}(\mathbf{x})) = \begin{bmatrix} \mathrm{Re}\,(\mathbf{J}^{\mathrm{H}}\mathbf{f}(\mathbf{x})) \\ \mathrm{Im}\,(\mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{f}(\mathbf{x})) \end{bmatrix}, \quad (7)$$

where $\mathbf{f}(\mathbf{x}) = \mathrm{vec}\,(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket - \mathcal{T})$. $\nabla_{\mathrm{r}} f(\mathbf{x})$ can thus again be obtained using efficient implementations for $\nabla f(\mathbf{x})$ [19].

*Detection complex terms:* To detect if terms should be made complex, the normalized inner product between two rank-1 terms can be computed:

$$d_{ij}(\mathbf{x}) := \operatorname{Re}\left(\left\langle \frac{[\![\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i]\!]}{\|[\![\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i]\!]\|}, \frac{[\![\mathbf{a}_j, \mathbf{b}_j, \mathbf{c}_j]\!]}{\|[\![\mathbf{a}_j, \mathbf{b}_j, \mathbf{c}_j]\!]\|} \right\rangle\right) \tag{8}$$
$$= \operatorname{Re}\left(\left\langle \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|}, \frac{\mathbf{a}_j}{\|\mathbf{a}_j\|} \right\rangle \left\langle \frac{\mathbf{b}_i}{\|\mathbf{b}_i\|}, \frac{\mathbf{b}_j}{\|\mathbf{b}_j\|} \right\rangle \left\langle \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}, \frac{\mathbf{c}_j}{\|\mathbf{c}_j\|} \right\rangle\right).$$

In a swamp, there are frequently one or more pairs of terms that are close to collinear with opposite sign and large norm such that they (almost) cancel each other. The set of problematic or degenerate terms is thus given by

$$\mathcal{C} = \{i \mid \exists j \neq i : d_{ij} < -\delta\}, \quad \text{for some } \delta \in [0, 1]. \tag{9}$$

Multiple canceling terms in a (hyper)plane can in principle be detected with more advanced measures, e.g., using the conditioning of $\mathbf{A} \odot \mathbf{B} \odot \mathbf{C}$, but this is out of the scope of this article.

*Initialize complex terms:* The following proposition shows that initializing the imaginary parts to a nonzero value is important for the algorithm to use the additional freedom.

**Proposition 1.** *If $\mathbf{U}_\mathcal{C}$, $\mathbf{V}_\mathcal{C}$, and $\mathbf{W}_\mathcal{C}$ are zero in* (5)*, then the GN step and gradient for these parameters are zero.*

*Proof.* If the imaginary parts are zero, $\mathbf{f}(\mathbf{x})$ and $\mathbf{J}$ are real-valued. Thus, the second part $\operatorname{Im}\left(\mathbf{J}_\mathcal{C}^{\mathrm{H}}\mathbf{f}(\mathbf{x})\right)$ of the gradient in (7) will be zero. Furthermore, $\mathbf{J}_\mathrm{r}^\top\mathbf{J}_\mathrm{r}$ in (6) will become a block diagonal matrix. The linear system to obtain the step can thus be decoupled, and the system for the imaginary parts becomes: $\left(\mathbf{J}_\mathcal{C}^\top\mathbf{J}_\mathcal{C}\right)\mathbf{p} = \mathbf{0}$. Since the GN algorithm ensures that $\mathbf{p}$ lies in the column space of $\mathbf{J}_\mathcal{C}^\top$, $\mathbf{p}$ will also be zero. □

To initialize the imaginary components, a complex rotation can be applied as in (4), which leaves the cost function value unchanged. However, the rank-1 terms then remain real-valued. That is why we additionally propose to compute a descent direction that lies (partly) in the null-space of $\mathbf{J}_\mathrm{r}^\top\mathbf{J}_\mathrm{r}$. In this way, we can find a complex valued CPD that lies closer to the tensor than the original real-valued CPD and use the additional freedom of the complex space. As the numerical computation of the null-space can be costly, the following parameterization is introduced.

**Proposition 2.** *A null-space vector of $\mathbf{J}_\mathrm{r}$ at $\mathbf{x}_\mathrm{p}$ in* (5) *is:*

$$\mathbf{n} := \begin{bmatrix} \operatorname{vec}(\mathbf{D}\mathbf{S}_1 - \mathbf{U}_\mathcal{C}\mathbf{P}_1) \\ \operatorname{vec}(\mathbf{E}\mathbf{S}_2 - \mathbf{V}_\mathcal{C}\mathbf{P}_2) \\ \operatorname{vec}(\mathbf{F}\left(-\mathbf{S}_1 - \mathbf{S}_2\right) + \mathbf{W}_\mathcal{C}\left(\mathbf{P}_1 + \mathbf{P}_2\right)) \\ \operatorname{vec}(\mathbf{D}_\mathcal{C}\mathbf{P}_1 + \mathbf{U}_\mathcal{C}\mathbf{S}_{1,\mathcal{C}}) \\ \operatorname{vec}(\mathbf{E}_\mathcal{C}\mathbf{P}_2 + \mathbf{V}_\mathcal{C}\mathbf{S}_{2,\mathcal{C}}) \\ \operatorname{vec}(-\mathbf{F}_\mathcal{C}\left(\mathbf{P}_1 + \mathbf{P}_2\right) - \mathbf{W}_\mathcal{C}\left(\mathbf{S}_{1,\mathcal{C}} + \mathbf{S}_{2,\mathcal{C}}\right)) \end{bmatrix}, \tag{10}$$

*where $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{P}_1$, and $\mathbf{P}_2$ are real, diagonal matrices. The null-space has dimension $2(R + R_\mathrm{c})$.*

*Proof.* It can straightforwardly be verified that $\mathbf{J}_\mathrm{r}\mathbf{n} = \mathbf{0}$, for all diagonal $\mathbf{S}_1, \mathbf{S}_2 \in \mathbb{R}^{R\times R}$, and $\mathbf{P}_1, \mathbf{P}_2 \in \mathbb{R}^{R_\mathrm{c}\times R_\mathrm{c}}$. □

To compute a descent direction in the null-space of $\mathbf{J}_\mathrm{r}$, we can now generate random $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{P}_1$, and $\mathbf{P}_2$ and perform a(n) (exact) plane search with the GN step $\mathbf{p}$ and this null-space vector $\mathbf{n}$ [31]. The output is a vector $\alpha_1\mathbf{p} + \alpha_2\mathbf{n}$ such that $f\left(\mathbf{x}^{\mathrm{new}}\right) < f(\mathbf{x})$ with $\operatorname{vec}\left(\mathbf{x}^{\mathrm{new}}\right) = \operatorname{vec}(\mathbf{x}) + \alpha_1\mathbf{p} + \alpha_2\mathbf{n}$. This can also be done for a few iterations $k$ as long as $\alpha_2 \neq 0$, before returning to the standard optimization method. We observed that the choice of the random $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{P}_1$, and $\mathbf{P}_2$ has little influence on the overall convergence of the algorithm. A complex rotation can first be applied such that the imaginary components $\mathbf{U}_\mathcal{C}$, $\mathbf{V}_\mathcal{C}$, and $\mathbf{W}_\mathcal{C}$ in (10) are nonzero.

*Algorithm framework:* The outline of the algorithm is shown in Alg. 1. The basis is the algorithm on line 2, e.g., a GN method with a line-search or trust-region method. Only when the degeneracy condition (9) is satisfied does the algorithm make the problematic terms complex on lines 8–13. This is done at most $k$ times, and afterwards, the framework returns to the basis algorithm on line 2. The performance of the algorithm is not very sensitive to the choice of $\delta$ and $k$. We found, e.g., $\delta = 0.9$ and $k = 30$ to be effective.

To round the output $\mathbf{x}$ to the closest real-valued CPD, angles $\phi$ and $\psi$ are sought to minimize the imaginary parts of $\mathbf{a}_i$ and $\mathbf{b}_i$ in (4). This can be done by computing the smallest right singular vectors of $[\operatorname{Re}(\mathbf{a}_i), \operatorname{Im}(\mathbf{a}_i)]$ and $[\operatorname{Re}(\mathbf{b}_i), \operatorname{Im}(\mathbf{b}_i)]$, and simply rotating $\mathbf{c}_i$ over $-\phi - \psi$; See Alg. 2.

**Algorithm 1** Complex parameterization framework

**Input:** $\mathcal{T}$, starting point $\mathbf{x}$, tolerance $\delta$ on $d_{ij}$ (8), iterations in null space $k \geq 1$, $\mathcal{C} \leftarrow \emptyset$

1: **while** not converged & $\mathcal{C}_{\mathrm{new}} = \emptyset$
2: $\quad\mathbf{x} \leftarrow$ optimization step $(\mathcal{T}, \mathbf{x})$
3: $\quad\mathcal{C}_{\mathrm{new}} \leftarrow \{i \mid \exists j \neq i : d_{ij}(\mathbf{x}) < -\delta\}$ (8), (9)
4: $\quad$**if** $\mathcal{C}_{\mathrm{new}} \neq \mathcal{C}$ **then**
5: $\quad\quad\mathcal{C} \leftarrow \mathcal{C}_{\mathrm{new}}$
6: $\quad\quad\mathbf{x} \leftarrow$ complex rotation $(\mathbf{x}, \mathcal{C})$
7: $\quad\quad$**for** $i = 1, \ldots, k$
8: $\quad\quad\quad$Solve $\mathbf{J}_\mathrm{r}(\mathbf{x})^\top\mathbf{J}_\mathrm{r}(\mathbf{x})\mathbf{p} = -\nabla_\mathrm{r}f(\mathbf{x})$ for $\mathbf{p}$
9: $\quad\quad\quad$Compute a random null-space vector $\mathbf{n}$ (10)
10: $\quad\quad\quad\boldsymbol{\alpha} \leftarrow \texttt{plane_search}(\mathcal{T}, \mathbf{x}, \mathbf{p}, \mathbf{n})$
11: $\quad\quad\quad\operatorname{vec}(\mathbf{x}_{\mathrm{p,new}}) \leftarrow \operatorname{vec}(\mathbf{x}_\mathrm{p}) + \alpha_1\mathbf{p} + \alpha_2\mathbf{n}$
12: $\quad\quad\quad$**if** $f(\mathbf{x}_{\mathrm{new}}) < f(\mathbf{x})$ & $\alpha_2 \neq 0$ **then**
13: $\quad\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x}_{\mathrm{new}}$

**Output:** $\mathbf{x}$

**Algorithm 2** Rotations to return to real-valued CPDs

**Input:** $\mathbf{x}$ in CPD format, $\mathcal{C}$ indicating complex rank-1 terms

1: **for** $i \in \mathcal{C}$
2: $\quad\begin{bmatrix}\sin(\phi)\\ \cos(\phi)\end{bmatrix}, \begin{bmatrix}\sin(\psi)\\ \cos(\psi)\end{bmatrix} \leftarrow$ 2nd right singular vectors of $\begin{bmatrix}\operatorname{Re}(\mathbf{a}_i) & \operatorname{Im}(\mathbf{a}_i)\end{bmatrix}$ and $\begin{bmatrix}\operatorname{Re}(\mathbf{b}_i) & \operatorname{Im}(\mathbf{b}_i)\end{bmatrix}$, respectively
3: $\quad\mathbf{a}_i, \mathbf{b}_i \leftarrow \operatorname{Re}\left(\mathbf{a}_i\mathrm{e}^{\jmath\phi}\right), \operatorname{Re}\left(\mathbf{b}_i\mathrm{e}^{\jmath\psi}\right)$, respectively
4: $\quad\mathbf{c}_i \leftarrow \operatorname{Re}\left(\mathbf{c}_i\mathrm{e}^{-\jmath(\phi+\psi)}\right)$

**Output:** real-valued CPD $\mathbf{x}$

*Computational complexity:* The most expensive steps in Alg. 1 for $N$th order tensors of size $I \times \cdots \times I$ are solving $\mathbf{J}_\mathrm{r}(\mathbf{x})^\top\mathbf{J}_\mathrm{r}(\mathbf{x})\mathbf{p} = -\nabla_\mathrm{r}f(\mathbf{x})$, which costs $\mathcal{O}\left((NI(R + R_\mathrm{c})^3\right)$ flop, and the construction of $\nabla f(\mathbf{x})$, which costs $\mathcal{O}\left(NRI^N\right)$

flop. For large $I$, the former can be sped up by solving the system inexactly using a modified conjugate gradient algorithm [19]. For $R_c \ll R$, the complexity matches with the GN method and for large $I$ and/or large $N$, also with ALS [19]. The metric $d_{ij}$ over all terms can be computed in $\mathcal{O}\left(NRI^2\right)$ flop and the plane search in $\mathcal{O}\left(N^6\right)$ flop [31].

## IV. Experiments on underdetermined ICA

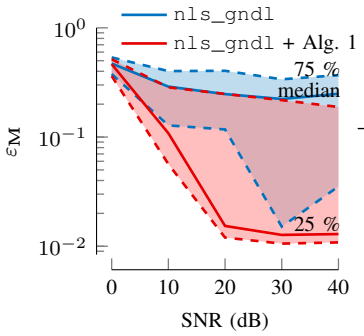
ICA is a well-known signal processing problem where one wants to identify a mixing matrix $\mathbf{M} \in \mathbb{R}^{J\times R}$ from a set of noisy observations $\mathbf{x} \in \mathbb{R}^J$ [32]–[34]: $\mathbf{x}(t) = \mathbf{M}\cdot\mathbf{s}(t) + \mathbf{n}(t)$, where $\mathbf{s}, \mathbf{n} \in \mathbb{R}^R$ are the source signals and additive independent identically distributed (i.i.d.) Gaussian noise, respectively. When the sources are statistically independent but individually correlated in time, the mixing matrix can be recovered by constructing the second-order covariance matrices of the shifted observations [35]. More specifically, the SOBIUM algorithm computes the CPD $\{\mathbf{M}, \mathbf{M}, \mathbf{D}\}$, where $d_{k,r} = \mathrm{E}\left[\mathbf{s}_r(t)\mathbf{s}_r(t+\tau_k)\right]$, of the third-order tensor $\mathcal{T} \in \mathbb{R}^{J\times J\times K}$ [36], where each slice of the tensor contains the covariance matrix $\mathbf{T}_k = \mathrm{E}\left[\mathbf{x}(t)\mathbf{x}(t+\tau_k)^\top\right]$, where $\tau_k$ are time shifts. When $J$ is smaller than $R$, the problem becomes underdetermined and the corresponding CPD problem more challenging [36], e.g., regularization terms have to be added to prevent degeneracy phenomena for GN methods [37].

*Experimental setup:* The $R = 11$ signals are generated randomly as $10^4$ elements in $\{-1, 1\}$, oversampled by a factor 8, filtered through a raised cosine pulse with roll-off $\mu = 0.3$, and each multiplied by a cosine with frequencies equidistant in $[0, 0.5]$. The number of observations $J$ and lags $K$ are both equal to seven. The mixing matrix is generated randomly with coefficients in $\mathcal{N}(0, 1)$. I.i.d. Gaussian noise is added to the observations. This experiment is repeated 100 times for each SNR level between 0 and 40 dB. We compare the performance of the GN method from Tensorlab (`nls_gndl`) [20] with and without Alg. 1, and the swamp avoiding method from [28] with a regularization parameter decaying with a factor 0.9 per iteration. The `cpd_eps` function is chosen as plane search method [20], [31]. The experiments are performed on a laptop with an AMD Ryzen 7 PRO 7840U processor and 64 GB of RAM, using Matlab R2023b and Tensorlab 4 beta, and with $\delta = 0.9$ and $k = 30$. The maximum number of iterations was set to 1000 and the tolerances `TolX` and `TolFun` to $10^{-10}$ [20]. The implementation of the algorithm and the code to reproduce the experiment results are available at www.tensorlabplus.net/vermeylen2026escaping.

*Results:* The mean value of the minimal error on the mixing matrix for the first two factor matrices is computed:

$$\varepsilon_{\mathbf{M}} := \min\left(\|\tilde{\mathbf{A}} - \mathbf{M}\|/\|\mathbf{M}\|,\ \|\tilde{\mathbf{B}} - \mathbf{M}\|/\|\mathbf{M}\|\right), \quad (11)$$

where $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are scaled and permuted versions of the factor matrices to match the columns of $\mathbf{M}$ as well as possible. A 'solution' is defined as an output $\mathbf{x}$ for which (1) $\varepsilon_{\mathbf{M}} < 2\varepsilon^*_{\mathbf{M}}$, where $\varepsilon^*_{\mathbf{M}}$ is the numerically optimal value obtained using the ground truth $\{\mathbf{M}, \mathbf{M}, \mathbf{D}\}$ as starting point, (2) $\mathcal{C} = \emptyset$ in (9) for $\delta = 0.9$, i.e., there are no degenerate terms. Success rates at 0 dB are not reported, as there are no accurate digits anymore




| ETC (s) | | nls_gndl standard | nls_gndl Alg. 1 |
|---|---|---|---|
| SNR (dB) | 10 | 17.4 | **14.9** |
| | 20 | 21.5 | **12.9** |
| | 30 | 14.3 | **13.1** |
| | 40 | 14.5 | **11.0** |

Fig. 1. The median of the error $\varepsilon_{\mathbf{M}}$ is up to an order of magnitude lower for Alg. 1 (left subfigure) and the ETC has improved for all SNR (right subfigure).

TABLE I
ALG. 1 GREATLY IMPROVES THE SUCCESS RATE UP TO A FACTOR THREE OVER 100 EXPERIMENTS FOR DIFFERENT SNR VALUES COMPARED TO `NLS_GNDL`, WITH AND WITHOUT PLANE SEARCH, AND THE ALS METHOD WITH REGULARIZATION ON THE STEP FROM [28].

| | version | SNR (dB) 10 | 20 | 30 | 40 | average |
|---|---|---|---|---|---|---|
| `nls_gndl` | standard | 21 | 18 | 27 | 25 | 22.75 |
| | plane search | 29 | 24 | 25 | 24 | 25.50 |
| Alg. 1 | rotation | 38 | 49 | 45 | 50 | 45.50 |
| | plane search | 55 | 66 | 58 | 64 | 60.75 |
| | full | **55** | **62** | **64** | **66** | **61.75** |
| ALS + regularization [28] | | 45 | 36 | 38 | 38 | 39.25 |

on this level. From both Fig. 1 and Tab. I, it can be seen that Alg. 1 greatly improves the accuracy in terms of $\varepsilon_{\mathbf{M}}$ for different SNR values. Furthermore, it can be seen that the full method proposed in Alg. 1 outperforms the version without rotation or plane search, `nls_gndl`, even with plane search, and the swamp avoiding method from [28]. For comparison, Tab. I includes the same plane search method without resorting to complex terms; it operates within the space spanned by a null-space vector of $\mathbf{J}$ and the GN step. The right subfigure in Fig. 1 shows that the ETC is lower for Alg. 1 (full version) on all SNR values: $\mathrm{ETC} := \left(p_s t_s + (1-p_s)t_f\right)/p_s$, where $p_s$ is the success rate, and $t_s$ and $t_f$ are the average time (in seconds) for a successful and failed experiment, respectively [16]. The average time per iteration was $9.7\cdot10^{-3}$ s for `nls_gndl`, $2.6\cdot10^{-2}$ s, for Alg. 1, and $4.7\cdot10^{-4}$ s for ALS. For higher $N$, a less significant difference is expected; see Section III.

## V. Conclusion

Leveraging state-of-the-art methods for canonical polyadic tensor approximation, the proposed optimization framework efficiently detects degenerate pairs of rank-1 terms, a key contributor to swamps. Afterwards, the algorithm gives more freedom by letting (only) these terms become complex. The imaginary parts are initialized by computing a descent direction in the space spanned by a null-space vector of the Jacobian matrix of the new parameterization and the GN direction. Experiments on difficult tensors (taking underdetermined independent component analysis as a use case) clearly illustrate that the algorithm finds significantly more accurate solutions over the full SNR range and at a lower cost.

## REFERENCES


[1] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 1, pp. 6–20, 2009.

[2] A. H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *Nonlinear Theory and Its Applications, IEICE*, vol. 1, no. 1, pp. 37–68, 2010.

[3] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, Oct. 2016.

[4] V. Khoromskaia and B. N. Khoromskij, *Tensor Numerical Methods in Quantum Chemistry*. Berlin, Boston: De Gruyter, 2018.

[5] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*. Heidelberg: Springer, 2012, vol. 42.

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

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

[8] L.-H. Lim, "Tensors in computations," *Acta Numerica*, vol. 30, p. 555–764, 2021.

[9] X. Bi, X. Tang, Y. Yuan, Y. Zhang, and A. Qu, "Tensors in statistics," *Annual Review of Statistics and Its Application*, vol. 8, no. 1, pp. 345–368, 2021.

[10] A. Franc, "Tensor Ranks for the Pedestrian for Dimension Reduction and Disentangling Interactions," Inrae - BioGeCo ; Inria Bordeaux Sud-Ouest, Research Report RR-9445, Dec. 2021.

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

[12] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm," *Linear Algebra and its Applications*, vol. 513, pp. 342–375, 2017.

[13] X. Guo, S. Miron, D. Brie, S. Zhu, and X. Liao, "A CANDECOMP/PARAFAC perspective on uniqueness of doa estimation using a vector sensor array," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3475–3481, 2011.

[14] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[15] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-model factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[16] P. Breiding and N. Vannieuwenhoven, "A Riemannian trust region method for the canonical tensor rank approximation problem," *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2435–2465, 2018.

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

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

[19] 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, 2013.

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

[21] W. P. Krijnen, T. K. Dijkstra, and A. Stegeman, "On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model," *Psychometrika*, vol. 73, no. 3, pp. 431–439, 2008.

[22] P. Paatero, "Construction and analysis of degenerate PARAFAC models," *Journal of Chemometrics*, vol. 14, no. 3, pp. 285–299, 2000.

[23] J. M. Landsberg, *Tensors: Geometry and Applications*. Rhode Island: American Mathematical Soc., 2011, vol. 128.

[24] X. Gong, M. J. Mohlenkamp, and T. R. Young, "The optimization landscape for fitting a rank-2 tensor with a rank-1 tensor," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 2, pp. 1432–1477, 2018.

[25] M. J. Mohlenkamp, "The dynamics of swamps in the canonical tensor approximation problem," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 3, pp. 1293–1333, 2019.

[26] C. Vermeylen, N. Vervliet, and L. De Lathauwer, "Reducing swamp behavior for the canonical polyadic decomposition problem by rank-1 freezing," *Numerical Algorithms*, vol. 100, pp. 831–859, 2025.

[27] B. C. Mitchell and D. S. Burdick, "Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies," *Journal of Chemometrics*, vol. 8, no. 2, pp. 155–168, 1994.

[28] C. Navasca, L. De Lathauwer, and S. Kindermann, "Swamp reducing technique for tensor decomposition," in *2008 16th European Signal Processing Conference*, 2008, pp. 1–5.

[29] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer, 7 2006.

[30] 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, 2012.

[31] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer, "Exact line and plane search for tensor optimization," *Computational Optimization and Applications*, vol. 63, no. 1, pp. 121–142, 2015.

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

[33] Z. Koldovsky, A. H. Phan, P. Tichavsky, and A. Cichocki, "A treatment of EEG data by underdetermined blind source separation for motor imagery classification," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 1484–1488.

[34] L. Albera, A. Kachenoura, P. Comon, A. Karfoul, F. Wendling, L. Senhadji, and I. Merlet, "ICA-based EEG denoising: A comparative analysis of fifteen methods," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 60, no. 3 Special issue on Data Mining in Bioengineering, pp. 407–418, Oct. 2012, special issue on Data Mining in Bioengineering.

[35] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, 1997.

[36] L. De Lathauwer and J. Castaing, "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1096–1105, 2008.

[37] P. Paatero, "The multilinear engine—a table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model," *Journal of Computational and Graphical Statistics*, vol. 8, no. 4, pp. 854–888, 1999.