An algebraic algorithm for rank-2 ParaTuck-2 decomposition

Konstantin Usevich

An algebraic algorithm for rank-2 ParaTuck-2 decomposition

Konstantin Usevich

Abstract

In this note we consider the problem of ParaTuck-2 decomposition of a complex three-way tensor. We provide an algebraic algorithm for finding the rank-2 decomposition for ParaTuck-2 tensors. Our approach relies only on linear algebra operations and is based on finding the kernel of a structured matrix constructed from the tensor.

1. Introduction

ParaTuck-2 decomposition of 3-rd order tensors (multi-way arrays), proposed in [HL96], can be viewed as a 2-level extension of the well-known CP (canonical polyadic) decomposition. It is relevant in several applications, such as chemometrics [Bro98] and telecommunications [FdA14].A tensor (3-way array) $\boldsymbol{\mathcal{T}} \in \mathbb{F}^{N_1 \times N_2 \times N_3}$ over a field $\mathbb{F}$ ($\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$) admits a ParaTuck-2 decomposition with rank (R, S) if it frontal slices can be written as

$$\boldsymbol{\mathcal{T}}_{:,:,k} = \boldsymbol{A}\boldsymbol{D}_k(\boldsymbol{G})\boldsymbol{F}\boldsymbol{D}_k(\boldsymbol{H})\boldsymbol{B}^{\mathsf{T}}, \tag{1}$$

$\boldsymbol{A} \in \mathbb{F}^{N_1 \times R}$, $\boldsymbol{B} \in \mathbb{F}^{N_2 \times S}$, $\boldsymbol{F} \in \mathbb{F}^{R \times S}$, $\boldsymbol{G} \in \mathbb{F}^{R \times N_3}$, $\boldsymbol{H} \in \mathbb{F}^{S \times N_3}$, and $\boldsymbol{D}_k(\boldsymbol{G})$ denotes the diagonal matrix built from a k-th column of the matrix $\boldsymbol{G}$. In a very special case, the Paratuck-2 decomposition reduces to the CP decomposition (when $R = S$ and $\boldsymbol{F} = \boldsymbol{D}_k(\boldsymbol{H}) = \boldsymbol{I}_R$ for all k in (1)). Generalizations of Paratuck-2 were also proposed for higher orders [FdA14], but we focus in this paper only on the third-order case ((1)). In this paper, we also assume that $R \leq N_1, S \leq N_1$.

Similarly to the CP decomposition, ParaTuck-2 enjoys strong uniqueness properties [HL96]. Under some conditions, the factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{F}$, $\boldsymbol{D}_k(\boldsymbol{G})$, $\boldsymbol{D}_k(\boldsymbol{H})$ can be recovered uniquely, subject to scaling and permutation ambiguities (similarly to the CP model). Despite the usefulness and the nice features of ParaTuck-2, there are very few effective algorithms to find a factorization (1). Many standard optimization tools, such as alternating least squares [Bro98, pp. 68–71] suffer from slow convergence and local minima. In many appliactions and algorithms, however, it is often considered that at least one of the factors (for example $\boldsymbol{A}$) is known [dOFFB19], [Bro98, p. 213], which simplifies considerably the problem. In [Nas20] it was suggested that double contractions may lead to another way of computing updates in the alternating lease stuares strategy. There are also reformulations of the ParaTuck-2 decomposition as a structured polyadic decomposition [FdA14], but these approaches do not currently yield a reliable way to compute the decomposition.

In this paper, we focus on the case when $R, S = 2$ and provide an algebraic algorithm to compute a rank-$(2, 2)$ ParaTuck-2 decomposition if it exists. Our approach relies only on the linear algebraic operations and make use of nonlinearly structured matrices constructed from a tensor. To our knowledge, it is the first algebraic algorithm for ParaTuck-2 decomposition.

Notation. Tensor arrays, matrices, and vectors, will be respectively denoted by bold calligraphic letters, *e.g.* $\boldsymbol{\mathcal{A}}$, with bold uppercase letters, *e.g.* $\boldsymbol{M}$, and with bold lowercase letters, *e.g.* $\boldsymbol{u}$; corresponding entries will be denoted by $\mathcal{A}_{ijk}$, M_{ij}, and u_i. Operator $\bullet_p$ denotes contraction on the pth index of a tensor; when contracted with a matrix, it is understood that summation is always performed on the second index of the matrix. For instance, $(\boldsymbol{\mathcal{A}} \bullet_1 \boldsymbol{M})_{ijk} = \sum_\ell \mathcal{A}_{\ell jk} M_{i\ell}$. We denote by $\boldsymbol{T}^{(1)} \in \mathbb{F}^{N_1 \times (N_2 N_3)}$ and $\boldsymbol{T}^{(2)} \in \mathbb{F}^{N_2 \times (N_1 N_3)}$ the first and second unfoldings respectively of a tensor $\boldsymbol{\mathcal{T}}$

2. Core tensor and its properties

In what follows, we assume $\mathbb{F} = \mathbb{C}$. We first note that the Paratuck-2 decomposition can be compactly written using contractions

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{C}} \underset{1}{\bullet} \boldsymbol{A} \underset{2}{\bullet} \boldsymbol{B}, \tag{2}$$

where $\boldsymbol{\mathcal{C}} \in \mathbb{F}^{R \times S \times N_3}$ is the core tensor with slices

$$\boldsymbol{\mathcal{C}}_{:,:,k} = \boldsymbol{D}_k(\boldsymbol{G}) \boldsymbol{F} \boldsymbol{D}_k(\boldsymbol{H}). \tag{3}$$

It is easy to see that (3) can be equivalently written as:

$$\mathcal{C}_{ijk} = F_{ij} G_{ik} H_{jk}, \tag{4}$$

where (4) is a generalization of the tensor product to product of matrices.

2.1. Basic considerations for the decomposition

Remark 2.1. *In the simple (generic) case (when* $\operatorname{rank}\{\boldsymbol{A}\} = \operatorname{rank}\{\boldsymbol{T}^{(1)}\} = R$ *and* $\operatorname{rank}\{\boldsymbol{B}\} = \operatorname{rank}\{\boldsymbol{T}^{(2)}\} = S$*), we can always restrict ourselves to the case* $R = N_1, S = N_2$*; indeed, if* $R < N_1, S < N_2$*, we can first compute the Tucker decomposition, and then compress to the case* $N_1' = R$*,* $N_2' = S$*.*

Now let us assume that $R = N_1, S = N_2$, and that matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible. If both matrices are known, then we can obtain the core tensor as $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{A}^{-1} \bullet_2 \boldsymbol{B}^{-1}$, and then the problem reduces to decomposition of the core tensor.

In the following simple (generic) situation it is very easy to retrieve all the remaining factors from the core tensor. Assume that there exists one fixed index r such that $\boldsymbol{\mathcal{C}}_{ijr} \neq 0$ for any i, j. Then we have that for any k

$$\frac{\mathcal{C}_{ijk}}{\mathcal{C}_{ijr}} = \frac{G_{ik}}{G_{ir}} \frac{H_{jk}}{H_{jr}}. \tag{5}$$

In other words, for each k, the matrix $\boldsymbol{X}^{(k)} \in \mathbb{F}^{N_1 \times N_2}$ obtained by elementwise division of $\boldsymbol{\mathcal{C}}_{:,:,k}$ by $\boldsymbol{\mathcal{C}}_{:,:,r}$ must be rank-one. Therefore, the matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ can be recovered easily from the SVD of $\boldsymbol{X}^{(k)}$.

3. Algorithms for the ParaTuck-2 decomposition

The main idea is to use the following equations for the core tensor.

3.1. Implicit equations for the core tensor

Note that the condition (5) requires the matrices $\boldsymbol{X}^{(k)}$ defined in the same section to be rank one. It is well known that the defining equations for the set of matrices of rank ≤ 1 is given by 2×2 minors. Therefore, the condition becomes:

$$X^{(k)}_{ij} X^{(k)}_{pq} - X^{(k)}_{iq} X^{(k)}_{pj} = \frac{\mathcal{C}_{ijk}}{\mathcal{C}_{ijr}} \frac{\mathcal{C}_{pqk}}{\mathcal{C}_{pqr}} - \frac{\mathcal{C}_{iqk}}{\mathcal{C}_{iqr}} \frac{\mathcal{C}_{pjk}}{\mathcal{C}_{pjr}} = 0$$

Therefore, by getting rid of the denominators, we get the equations

$$\mathcal{C}_{ijk} \mathcal{C}_{pqk} \mathcal{C}_{iqr} \mathcal{C}_{pjr} - \mathcal{C}_{iqk} \mathcal{C}_{pjk} \mathcal{C}_{ijr} \mathcal{C}_{pqr} = 0 \tag{6}$$

It is easy to see that any core tensor (4) satisfies (6). But, of course (6) may define a larger set of tensors.

3.2. Determinantal representation for the 2×2 case

We will adopt the following simplified notation for the slices of the core tensor:

$$\mathcal{C}_{:,:,k} = \begin{bmatrix} w_k & y_k \\ x_k & z_k \end{bmatrix},$$

then the equation (6) becomes simply

$$w_k z_k x_r y_r - x_k y_k w_r z_r = 0, \quad k, r \in \{1, \dots, N_3\}. \tag{7}$$

Then we can see that (7) is equivalent to

$$\operatorname{rank}\{\boldsymbol{\Psi}(C)\} \leq 1, \quad \text{where } \boldsymbol{\Psi}(C) := \begin{bmatrix} w_1 z_1 & w_2 z_2 & \cdots & w_{N_3} z_{N_3} \\ x_1 y_1 & x_2 y_2 & \cdots & x_{N_3} y_{N_3} \end{bmatrix}. \tag{8}$$

This is the idea that we will later on use for the decomposition: we will construct a structured matrix from the tensor and look at its left kernel.

3.3. An algorithm for the 2×2 case

We assume that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible, and denote the inverses as

$$\boldsymbol{A}^{-1} = \widetilde{\boldsymbol{A}} = \begin{bmatrix} \widetilde{\boldsymbol{a}}_1 & \widetilde{\boldsymbol{a}}_2 \end{bmatrix}^{\mathsf{T}}, \quad \boldsymbol{B}^{-1} = \widetilde{\boldsymbol{B}} = \begin{bmatrix} \widetilde{\boldsymbol{b}}_1 & \widetilde{\boldsymbol{b}}_2 \end{bmatrix}^{\mathsf{T}},$$

thus the core tensor will be

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{T}} \underset{1}{\bullet} \widetilde{\boldsymbol{A}} \underset{2}{\bullet} \widetilde{\boldsymbol{B}}.$$

In particular, we will have that

$$\begin{bmatrix} w_k & y_k \\ x_k & z_k \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{a}}_1^{\mathsf{T}} \boldsymbol{\mathcal{T}}_{:,:,k} \widetilde{\boldsymbol{b}}_1 & \widetilde{\boldsymbol{a}}_1^{\mathsf{T}} \boldsymbol{\mathcal{T}}_{:,:,k} \widetilde{\boldsymbol{b}}_2 \\ \widetilde{\boldsymbol{a}}_2^{\mathsf{T}} \boldsymbol{\mathcal{T}}_{:,:,k} \widetilde{\boldsymbol{b}}_1 & \widetilde{\boldsymbol{a}}_2^{\mathsf{T}} \boldsymbol{\mathcal{T}}_{:,:,k} \widetilde{\boldsymbol{b}}_2 \end{bmatrix},$$

therefore the elements of the matrix $\boldsymbol{\Psi}(C)$ can be obtained as

$$\begin{bmatrix} w_k z_k \\ x_k y_k \end{bmatrix} = \begin{bmatrix} (\widetilde{\boldsymbol{a}}_1 \boxtimes \widetilde{\boldsymbol{a}}_2 \boxtimes \widetilde{\boldsymbol{b}}_1 \boxtimes \widetilde{\boldsymbol{b}}_2)^{\mathsf{T}} \\ (\widetilde{\boldsymbol{a}}_2 \boxtimes \widetilde{\boldsymbol{a}}_1 \boxtimes \widetilde{\boldsymbol{b}}_1 \boxtimes \widetilde{\boldsymbol{b}}_2)^{\mathsf{T}} \end{bmatrix} \operatorname{vec}\{\boldsymbol{\mathcal{T}}_{:,:,k} \boxtimes \boldsymbol{\mathcal{T}}_{:,:,k}\}. \tag{9}$$

Next, we denote

$$\boldsymbol{T} = \begin{bmatrix} t_1 & t_3 \\ t_2 & t_4 \end{bmatrix}$$

and define its Veronese embedding $\mathbb{F}^{2\times 2} \to \mathbb{F}^{10}$:

$$\varphi(\boldsymbol{T}) = \begin{bmatrix} t_1^2 & t_2^2 & t_3^2 & t_4^2 & t_1 t_2 & t_1 t_3 & t_1 t_4 & t_2 t_3 & t_2 t_4 & t_3 t_4 \end{bmatrix}^{\mathsf{T}}.$$

We also denote the following map $\mathbb{F}^{2\times 2} \times \mathbb{F}^{2\times 2} \to \mathbb{F}^{10}$:

$\boldsymbol{\theta}\left(\left[\begin{smallmatrix} u_1 & u_3 \\ u_2 & u_4 \end{smallmatrix}\right], \left[\begin{smallmatrix} v_1 & v_3 \\ v_2 & v_4 \end{smallmatrix}\right]\right) =$

$$\begin{bmatrix} u_1 v_1 & u_2 v_2 & u_3 v_3 & u_4 v_4 & u_1 v_2 + u_2 v_1 & u_1 v_3 + u_3 v_1 & u_1 v_4 + u_4 v_1 & u_2 v_3 + u_3 v_2 & u_2 v_4 + u_4 v_2 & u_3 v_4 + u_4 v_3 \end{bmatrix}^{\mathsf{T}}.$$

Then we get that

$$\boldsymbol{\Psi}(\boldsymbol{\mathcal{C}}) = \begin{bmatrix} \left(\boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_1 \widetilde{\boldsymbol{b}}_1^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_2 \widetilde{\boldsymbol{b}}_2^{\mathsf{T}})\right)^{\mathsf{T}} \\ \left(\boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_2 \widetilde{\boldsymbol{b}}_1^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_1 \widetilde{\boldsymbol{b}}_2^{\mathsf{T}})\right)^{\mathsf{T}} \end{bmatrix} \boldsymbol{\Phi}(\boldsymbol{\mathcal{T}}), \quad \text{where } \boldsymbol{\Phi}(\boldsymbol{\mathcal{T}}) = \begin{bmatrix} \varphi(\boldsymbol{\mathcal{T}}_{:,:,1}) & \cdots & \varphi(\boldsymbol{\mathcal{T}}_{:,:,N_3}) \end{bmatrix}.$$

Due to the condition (8), this implies that for a ParaTuck-2 decomposable tensor, it should hold that

$$\operatorname{rank}\{\boldsymbol{\Phi}(\boldsymbol{\mathcal{T}})\} \leq 9.$$

This yields the following algorithm.

Algorithm 1: Sketch, Paratuck, 2×2 case
input : $\boldsymbol{\mathcal{T}}$
output: $\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}}$
1. Construct $\boldsymbol{\Phi}(\boldsymbol{\mathcal{T}})$
2. Determine the vector $\boldsymbol{\theta} \in \mathbb{F}^{10}$ in the left kernel, so that $\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\Phi}(\boldsymbol{\mathcal{T}}) = 0$
3. Find $\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}}$ such that

$$\boldsymbol{\theta} \in \mathrm{Span}\{\boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_1\widetilde{\boldsymbol{b}}_1^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_2\widetilde{\boldsymbol{b}}_2^{\mathsf{T}}), \boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_2\widetilde{\boldsymbol{b}}_1^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_1\widetilde{\boldsymbol{b}}_2^{\mathsf{T}})\} \tag{10}$$

3.4. Retrieving the factors from the kernels of a structured matrix

The last step of Algorithm 1 can be performed as follows. First, we define the following linear maps $\boldsymbol{M} : \mathbb{F}^{10} \to \mathbb{F}^{4\times 4}$,

$$\boldsymbol{M}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_1 & \frac{\theta_5}{2} & \frac{\theta_6}{2} & \frac{\theta_7}{2} \\ \frac{\theta_5}{2} & \theta_2 & \frac{\theta_8}{2} & \frac{\theta_9}{2} \\ \frac{\theta_6}{2} & \frac{\theta_8}{2} & \theta_3 & \frac{\theta_{10}}{2} \\ \frac{\theta_7}{2} & \frac{\theta_9}{2} & \frac{\theta_{10}}{2} & \theta_4 \end{bmatrix},$$

as well as the linear map $\mathscr{S} : \mathbb{F}^{4\times 4} \to \mathbb{F}^{3\times 3}$:

$$\mathscr{S}\left(\left[\begin{array}{cc|cc} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ \hline M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{array}\right]\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{11} & M_{31} & M_{13} & M_{33} \\ M_{21} & M_{41} & M_{23} & M_{43} \\ M_{12} & M_{32} & M_{14} & M_{34} \\ M_{22} & M_{42} & M_{24} & M_{44} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then an easy calculation shows that

$$(\mathscr{S} \circ \boldsymbol{M})(\boldsymbol{\theta}) = \begin{bmatrix} \theta_1 & \theta_6 & \theta_3 \\ \theta_5 & \theta_7 + \theta_8 & \theta_{10} \\ \theta_2 & \theta_9 & \theta_4 \end{bmatrix}. \tag{11}$$

Assuming, without loss of generality, that

$$\widetilde{\boldsymbol{a}}_1 = \begin{bmatrix} -\alpha_1 \\ 1 \end{bmatrix}, \quad \widetilde{\boldsymbol{a}}_2 = \begin{bmatrix} \alpha_2 \\ -1 \end{bmatrix}, \quad \widetilde{\boldsymbol{b}}_1 = \begin{bmatrix} -\beta_1 \\ 1 \end{bmatrix}, \quad \widetilde{\boldsymbol{b}}_2 = \begin{bmatrix} \beta_2 \\ -1 \end{bmatrix},$$

we obtain, after easy calculations that

$$(\mathscr{S} \circ \boldsymbol{M})(\boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_1\widetilde{\boldsymbol{b}}_1^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_2\widetilde{\boldsymbol{b}}_2^{\mathsf{T}})) = (\mathscr{S} \circ \boldsymbol{M})(\boldsymbol{\theta}(\widetilde{\boldsymbol{a}}_1\widetilde{\boldsymbol{b}}_2^{\mathsf{T}}, \widetilde{\boldsymbol{a}}_2\widetilde{\boldsymbol{b}}_1^{\mathsf{T}})) = \begin{bmatrix} \alpha_1\alpha_2 \\ -(\alpha_1 + \alpha_2) \\ 1 \end{bmatrix} \begin{bmatrix} \beta_1\beta_2 & -(\beta_1 + \beta_2) & 1 \end{bmatrix}. \tag{12}$$

Therefore, we have that for any $\boldsymbol{\theta}$ from the two-dimensional subspace defined in (10) the matrix $\mathscr{S} \circ \boldsymbol{M}(\boldsymbol{\theta})$ is rank-one and is a multiple of the matrix in (12). Therefore, the coefficients $\alpha_1, \alpha_2, \beta_1, \beta_2$ can be retrieved from any rank-one decomposition of $\mathscr{S} \circ \boldsymbol{M}(\boldsymbol{\theta}) = \sigma \boldsymbol{u}\boldsymbol{v}^{\mathsf{T}}$ and by taking the roots of the generating polynomials, i.e. by using the relations:

$$u_1 + u_2 t + u_3 t^2 = u_3(t - \alpha_1)(t - \alpha_2), \quad v_1 + v_2 t + v_3 t^2 = v_3(t - \beta_1)(t - \beta_2)$$

Finally, we note that from α_k and β_k we can easily find the matrices $\boldsymbol{A} = (\widetilde{\boldsymbol{A}})^{-1}$ and $\boldsymbol{B} = (\widetilde{\boldsymbol{B}})^{-1}$ themselves. Indeed, from inversion of a 2×2 matrix we have

$$\boldsymbol{A} = \begin{bmatrix} -\alpha_1 & 1 \\ \alpha_2 & -1 \end{bmatrix}^{-1} = \frac{1}{\alpha_2 - \alpha_1} \begin{bmatrix} 1 & 1 \\ \alpha_2 & \alpha_1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} -\beta_1 & 1 \\ \beta_2 & -1 \end{bmatrix}^{-1} = \frac{1}{\beta_2 - \beta_1} \begin{bmatrix} 1 & 1 \\ \beta_2 & \beta_1 \end{bmatrix}.$$

Without loss of generality (due to scaling and permutation ambiguities), we can drop the constants and also swap the columns of $\boldsymbol{A}$ and $\boldsymbol{B}$.

Algorithm 2: Algebraic decomposition algorithm for Paratuck, $R = S = 2$ case

input : $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1, I_2, I_3}$

output: $\boldsymbol{A}, \boldsymbol{B}$

1. Compute the HOSVD of rank $(2, 2, I_3)$: $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{T}}_c; \boldsymbol{A}_c, \boldsymbol{B}_c, \boldsymbol{I}_{I_1}]\!]$, $\boldsymbol{A}_c \in \mathbb{F}^{I_1 \times R}, \boldsymbol{B}_c \in \mathbb{F}^{I_2 \times S}$;
2. Construct $\boldsymbol{\Phi}(\boldsymbol{\mathcal{T}}_c)$
3. Compute the compact SVD $\boldsymbol{\Phi}(\boldsymbol{\mathcal{T}}_c) = \boldsymbol{U}_\Phi \Sigma \boldsymbol{V}_\Phi^{\mathsf{H}}$, take $\boldsymbol{\theta} = (\boldsymbol{U}_\Phi)_{:,10}$;
4. Build the matrix $\boldsymbol{S} = (\mathscr{S} \circ \boldsymbol{M})(\boldsymbol{\theta})$ according to (11)
5. Find the best rank-1 approximation $\widehat{\sigma}\widehat{\boldsymbol{u}}\widehat{\boldsymbol{v}}^{\mathsf{T}}$ of $\boldsymbol{S}$ (via SVD)
6. Find $\alpha_1, \alpha_2, \beta_1, \beta_2$ to be the roots of

$$\widehat{u}_1 + \widehat{u}_2 t + \widehat{u}_3 t^2, \quad \widehat{v}_1 + \widehat{v}_2 t + \widehat{v}_3 t^2$$

7. Set

$$\boldsymbol{A}' = \begin{bmatrix} 1 & 1 \\ \alpha_1 & \alpha_2 \end{bmatrix}, \quad \boldsymbol{B}' = \begin{bmatrix} 1 & 1 \\ \beta_1 & \beta_2 \end{bmatrix}.$$

and find $\boldsymbol{A} = \boldsymbol{A}_c \boldsymbol{A}', \boldsymbol{B} = \boldsymbol{B}_c \boldsymbol{B}'$.

8. Find the core tensor $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{T}}_c \bullet_1 \boldsymbol{A}^{-1} \bullet_2 \boldsymbol{B}^{-1}$.
9. Set $\boldsymbol{F} = \boldsymbol{\mathcal{C}}_{:,:,1}$
10. Determine the factors $\boldsymbol{G}$ and $\boldsymbol{H}$ from rank-one approximations of elementwise divisions of $\boldsymbol{\mathcal{C}}_{:,:,k}$ by $\boldsymbol{\mathcal{C}}_{:,:,1}$, see Section 2.1.

3.5. Overall algorithm

Here we summarize the complete algorithm, i.e., what is actually computed.
Some remarks on Algorithm 2:

- The algorithm works in the generic case, but it can be modified to decide whether a given tensor admit a complex ParaTuck-2 decomposition with ranks ≤ 2;
- In the noisy case, because we make a nonlinear transformation of our data (by the mapping Φ), at the step 3, instead of the SVD (total least squares), it is better to take the so-called adjusted least squares estimator (see [UM16] for more details);
- There is no restrictions in fixing the first rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ to 1. We just need to allow for projective roots (i.e. α_k can be ∞).

3.6. Some numerical experiments

Algorithm 2 is implemented in Julia.

3.6.1. Deterministic example

We consider the following test example:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & 1 \\ 1 & -3 \end{bmatrix}$$

and

$$\boldsymbol{G} = \begin{bmatrix} -5 & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 1 & -3 & 2 & -2 & -1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} -5 & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Algorithm 2 is able to compute a Paratuck-2 approximation with approximation error (squared Frobenius norm of the difference: $\|\boldsymbol{\mathcal{T}} - \widehat{\boldsymbol{\mathcal{T}}}\|_F^2$) as $3.65 \cdot 10^{-26}$; after refinement by alternating least squares, the approximation error is reduced to $8.45 \cdot 10^{-27}$. The alternating least squares was run for one iteration.

We also ran alternating least squares [Bro98, pp. 68–71] with random initialization (maximum 5000 iterations), and it gave an approximation error $\|\boldsymbol{\mathcal{T}} - \widehat{\boldsymbol{\mathcal{T}}}\|_F^2$ below 10^{-10} only in 8% of the cases (number of Monte-Carlo runs was 100).

3.6.2. Random example

We next consider random example with $I_1 = I_2 = 10$, $I_3 = 15$ and $R = S = 2$. We generate 100 examples, with $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{H}$, $\boldsymbol{F}$, $\boldsymbol{G}$ having i.i.d. entries drawn randomly from $N(0, 1)$. In 99 of 100 runs the algebraic algorithm gave the approximation error $\|\boldsymbol{\mathcal{T}} - \widehat{\boldsymbol{\mathcal{T}}}\|_F^2$ below 10^{-20} (in the remaining one example it was $\sim 10^{-18}$).

4. Acknowledgements

This research was supported by the ANR (Agence Nationale de Recherche) grant LeaFleT (ANR-19-CE23-0021). I also would like to thank Yassine Zniyed, Mariya Ishteva and Philippe Dreesen for stimulating discussions.

References

[Bro98] Rasmus Bro. *Multi-way Analysis in the Food Industry*. PhD thesis, Vrije Universiteit Brussel (VUB), 1998.

[dOFFB19] Pedro Marinho R. de Oliveira, C. Alexandre Rolim Fernandes, Gérard Favier, and Remy Boyer. PARATUCK Semi-Blind Receivers for Relaying Multi-Hop MIMO Systems. *Digital Signal Processing*, 92:127–138, 2019.

[FdA14] Gérard Favier and André LF de Almeida. Overview of constrained parafac models. *EURASIP Journal on Advances in Signal Processing*, 2014(1):142, 2014.

[HL96] Richard A Harshman and Margaret E Lundy. Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/candecomp. *Psychometrika*, 61(1):133–154, 1996.

[Nas20] Kristina Naskovska. *Advanced tensor based signal processing techniques for wireless communication systems and biomedical signal processing*. PhD thesis, Ilmenau, Jan 2020. Dissertation, Technische Universität Ilmenau, 2019.

[UM16] K. Usevich and I. Markovsky. Adjusted least squares fitting of algebraic hypersurfaces. *Linear Algebra Appl.*, 502:243–274, 2016.