PARAFAC2 AO-ADMM: Constraints in all modes

Marie Roald
Simula Metropolitan Center for Digital Engineering & Oslo Metropolitan Univ.
Oslo, Norway
mariero@simula.no

Carla Schenker
Simula Metropolitan Center for Digital Engineering & Oslo Metropolitan Univ.
Oslo, Norway
carla@simula.no

Jeremy E. Cohen
University of Rennes Inria, CNRS, IRISA
Rennes, France
jeremy.cohen@irisa.fr

Evrim Acar
Simula Metropolitan Center for Digital Engineering
Oslo, Norway
evrim@simula.no

***Abstract*—The PARAFAC2 model provides a flexible alternative to the popular CANDECOMP/PARAFAC (CP) model for tensor decompositions. Unlike CP, PARAFAC2 allows factor matrices in one mode (i.e., evolving mode) to change across tensor slices, which has proven useful for applications in different domains such as chemometrics, and neuroscience. However, the evolving mode of the PARAFAC2 model is traditionally modelled implicitly, which makes it challenging to regularise it. Currently, the only way to apply regularisation on that mode is with a flexible coupling approach, which finds the solution through regularised least squares subproblems. In this work, we instead propose an alternating direction method of multipliers (ADMM)-based algorithm for fitting PARAFAC2 and widen the possible regularisation penalties to any proximable function. Our experiments demonstrate that the proposed ADMM-based approach for PARAFAC2 can accurately recover the underlying components from simulated data while being both computationally efficient and flexible in terms of imposing constraints.**

***Index Terms*—PARAFAC2, Tensor decomposition, AO-ADMM**

I. Introduction

Tensor decompositions, in particular the CANDECOMP/PARAFAC (CP) model [1], [2], have successfully extracted meaningful patterns from complex data in many disciplines including chemometrics [3] and neuroscience [4], [5]. However, the CP model has strict assumptions of multilinearity that can be violated in practice. Another tensor model, PARAFAC2 [6], relaxes the CP model by allowing for evolving factors in one mode. This relaxation also enables decomposing stacks of matrices of varying size.

The ability to describe such evolving or irregular factors has made the PARAFAC2 model a powerful tool. For instance, in chemometrics, PARAFAC2 has been applied to chromatographic data with unaligned elution profiles [7]. PARAFAC2 has also been used to analyse unaligned temporal profiles in electronic health records [8] and for information retrieval using documents from different languages [9]. Recently, PARAFAC2 has also shown promise for tracing time-evolving patterns of brain connectivity in neuroscience (illustrated in Fig. 1) [10].

Often, the interpretability of component models, such as CP and PARAFAC2, can be improved through constraints and regularisation. However, evolving components of the PARAFAC2 model are usually computed implicitly [11]. Therefore, it is challenging to impose constraints or regularisation on these evolving factors. While previous work has showed that such constraints can improve interpretability and recovery [8], [12], there is no known algorithm for flexibly imposing regularisation with general penalty functions or hard constraints on the evolving factors. Helwig imposed smoothness on these factors by constraining them to follow a low-rank B-spline interpolation [13]. To achieve smoothness, the data tensor is projected onto the linear subspace spanned by the given B-spline interpolation matrix before decomposing with PARAFAC2. However, this approach requires that the spline knots are known a-priori, which may be difficult in practice.

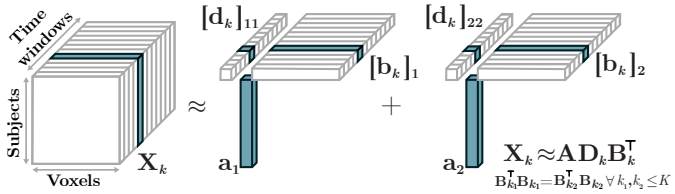

Fig. 1: Illustration of a two-component PARAFAC2 model for tracing networks in neuroimaging data.

Currently, the only way to regularise the evolving mode of a PARAFAC2 model, without knowing the subspace the components lie in, is with a flexible coupling approach [12]. This approach relaxes the PARAFAC2 constraint and finds the components by solving regularised least squares problems. Another notable approach is by Yin *et al.* [14] using a regularisation penalty inspired by PARAFAC2 to improve the uniqueness properties of regularised coupled non-negative matrix factorisation for binary data.

In this paper, we propose an alternating optimisation scheme with the alternating direction method of multipliers (AO-ADMM) to fit PARAFAC2 models with regularisation on all modes. The AO-ADMM scheme has recently been introduced for constrained CP models [15], and later also extended to regularised linearly coupled matrix-tensor factorisations [16]. Afshar *et al.* proposed using AO-ADMM to impose proximable constraints on the non-evolving factor matrices of the PARAFAC2 model [8]. Here, unlike previous work, we introduce ADMM updates for the evolving mode as well, widening the possible regularisation penalties on this mode to any proximable function. With numerical experiments on

 DOI: https://doi.org/10.23919/EUSIPCO54536.2021.9615927

simulated data, we show that our approach can accurately recover underlying components while being both flexible in terms of imposing constraints and computationally efficient.

II. Tensor decomposition with PARAFAC2

Tensors can be seen as multi-way arrays that generalise the concept of matrices to higher order data [17]. As such, a vector is a first-order tensor, a matrix is a second-order tensor, a "cube" of numbers is a third-order tensor and so forth. A tensor with more than two modes is often called a *higher-order tensor*. We denote higher-order tensors as $\mathcal{X}$, matrices as $\mathbf{X}$, vectors as $\mathbf{x}$, and the Frobenius norm of $\mathcal{X}$ as $\| \mathcal{X} \|_F$.

PARAFAC2 can be seen as a relaxed version of the CP model. CP assumes multilinearity, and for a third-order tensor, using an R-component CP model, each frontal slice is modelled as:

$$\mathbf{X}_k \approx \mathbf{A}\mathbf{D}_k\mathbf{B}^\mathsf{T}, \tag{1}$$

where $\mathbf{D}_k$ is an $R \times R$ diagonal matrix. Note that each slice, $\mathbf{X}_k \in \mathbb{R}^{I \times J}$, has the same $\mathbf{A}$ and $\mathbf{B}$ matrices. PARAFAC2, on the other hand, allows each slice to have a different $\mathbf{B}$ matrix:

$$\mathbf{X}_k \approx \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T}, \tag{2}$$

where $\mathbf{B}_k$s follow the *PARAFAC2 constraint*, i.e., $\mathbf{B}_{k_1}^\mathsf{T}\mathbf{B}_{k_1} = \mathbf{B}_{k_2}^\mathsf{T}\mathbf{B}_{k_2}$ for all $k_1, k_2 \leq K$, $\mathbf{D}_k \in \mathbb{R}^{R \times R}$ is a diagonal matrix.

III. Optimisation

A. PARAFAC2 & ALS

To solve the unconstrained PARAFAC2 problem, [11] reformulated the model to the following equivalent form:

$$\mathbf{X}_k \approx \mathbf{A}\mathbf{D}_k\mathbf{\Delta_B}^\mathsf{T}\mathbf{P}_k^\mathsf{T}, \tag{3}$$

where $\mathbf{\Delta_B}$ is a square matrix and $\mathbf{P}_k^\mathsf{T}\mathbf{P}_k = \mathbf{I}$. This problem can be solved efficiently using an alternating least squares (ALS) procedure, where the $\mathbf{P}_k$ updates are performed by solving an orthogonal procrustes problem.

B. PARAFAC2 & AO-ADMM

We wish to solve the regularised PARAFAC2 problem

$$\begin{aligned} \underset{\mathbf{A}, \{\mathbf{B}_k, \mathbf{D}_k\}_{k\leq K}}{\arg\min} & \left\{ \begin{array}{l} f\left(\mathbf{A}, \{\mathbf{B}_k\}_{k\leq K}, \{\mathbf{D}_k\}_{k\leq K}\right) \\ + g_\mathbf{A}(\mathbf{A}) + \sum_{k=1}^{K} g_\mathbf{B}(\mathbf{B}_k) + g_\mathbf{D}(\mathbf{D}_k) \end{array} \right\}, \\ \text{subject to} & \quad \mathbf{B}_{k_1}^\mathsf{T}\mathbf{B}_{k_1} = \mathbf{B}_{k_2}^\mathsf{T}\mathbf{B}_{k_2} \quad \forall k_1, k_2 \leq K \end{aligned} \tag{4}$$

where $f\left(\mathbf{A}, \{\mathbf{B}_k\}_{k\leq K}, \{\mathbf{D}_k\}_{k\leq K}\right) = \sum_{k=1}^{K} \| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \|_F^2$ is the sum of squared errors (SSE) data fidelity term, and $g_\mathbf{A}, g_\mathbf{B}, g_\mathbf{D}$ are regularisation functions. However, imposing regularisation is difficult within the traditional ALS algorithm, as it estimates the $\mathbf{B}_k$ matrices implicitly as the product of orthogonal $\mathbf{P}_k$ matrices and a $\mathbf{\Delta_B}$ matrix.

An alternative to directly solving regularised problems is to use splitting methods. In particular, we use ADMM [18] to solve split problems of the form

$$\begin{aligned} \underset{\mathbf{x}, \mathbf{z_x}}{\arg\min} & \quad f(\mathbf{x}) + g(\mathbf{z_x}) \\ \text{subject to} & \quad \mathbf{x} = \mathbf{z_x} \end{aligned} \tag{5}$$

Here, f and g represent the data-fidelity term and regularisation penalty, respectively.

To use ADMM, we have to solve subproblems on the form

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\arg\min} \quad h(\mathbf{x}) + \frac{\rho}{2} \| \mathbf{M}\mathbf{x} - \mathbf{y} \|_F^2, \tag{6}$$

with some problem dependent matrix, $\mathbf{M}$. When $\mathbf{M}$ is an identity matrix, then the above optimisation problem defines the *scaled proximal operator*, $\text{prox}_{\frac{h}{\rho}}(\mathbf{y})$, with scale parameter ρ [19] (see Section III-G for automatic selection of ρ). To use ADMM for the PARAFAC2 decomposition, we need a natural splitting scheme where all subproblems are easily evaluated. Such a scheme is known for the static modes of PARAFAC2 [8]. However, no ADMM splitting scheme has been presented for the evolving mode yet.

C. ADMM for the B mode

To specify an ADMM scheme for the problem

$$\begin{aligned} \underset{\{\mathbf{B}_k\}_{k\leq K}}{\arg\min} & \quad \sum_{k=1}^{K} f_{\mathbf{B}_k}(\mathbf{B}_k) + g_\mathbf{B}(\mathbf{B}_k), \\ \text{subject to} & \quad \mathbf{B}_{k_1}^\mathsf{T}\mathbf{B}_{k_1} = \mathbf{B}_{k_2}^\mathsf{T}\mathbf{B}_{k_2} \quad \forall k_1, k_2 \leq K \end{aligned} \tag{7}$$

where $f_{\mathbf{B}_k}(\mathbf{B}_k) = \| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \|_F^2$, we introduce two sets of auxiliary variables, $\mathbf{Z}_{\mathbf{B}_k}$ and $\mathbf{Y}_{\mathbf{B}_k}$, which respectively split the regularisation by $g_\mathbf{B}$ and the PARAFAC2 constraints, forming the problem:

$$\begin{aligned} \underset{\{\mathbf{B}_k, \mathbf{Z}_{\mathbf{B}_k}, \mathbf{Y}_{\mathbf{B}_k}\}_{k\leq K}}{\arg\min} & \sum_{k=1}^{K} [f_{\mathbf{B}_k}(\mathbf{B}_k) + g_\mathbf{B}(\mathbf{Z}_{\mathbf{B}_k})] + \iota_{\text{PF2}}\left(\{\mathbf{Y}_{\mathbf{B}_k}\}_{k\leq K}\right), \\ \text{subject to} & \quad \mathbf{B}_k = \mathbf{Z}_{\mathbf{B}_k}, \quad \mathbf{B}_k = \mathbf{Y}_{\mathbf{B}_k} \quad \forall k \end{aligned} \tag{8}$$

where $\iota_{\text{PF2}}\left(\{\mathbf{Y}_{\mathbf{B}_k}\}_{k\leq K}\right) = 0$ if $\mathbf{Y}_{\mathbf{B}_k}^\mathsf{T}\mathbf{Y}_{\mathbf{B}_k}$ is constant over k and ∞ otherwise. This problem can be tentatively solved using the ADMM algorithm specified in Algorithm 2.

The ADMM updates for the evolving modes require the efficient solution of three subproblems. The update with respect to the data fidelity function, ($f_{\mathbf{B}_k}(\mathbf{B}_k) = \| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \|_F^2$), and the proximal operators of the regularisation function ($g_\mathbf{B}$) and the characteristic function for sets of matrices with constant cross product (ι_{PF2}). The update step for the data fidelity function is the solution to the least squares problem:

$$\underset{\mathbf{B}_k}{\arg\min} \left\{ \begin{array}{l} \frac{2}{\rho_{\mathbf{B}_k}} \left\| \mathbf{X}_k - \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} \right\|_F^2 + \left\| \mathbf{B}_k - \mathbf{Z}_{\mathbf{B}_k} + \boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}} \right\|_F^2 \\ + \left\| \mathbf{B}_k - \mathbf{Y}_{\mathbf{B}_k} + \boldsymbol{\mu}_{\mathbf{\Delta_B}_k} \right\|_F^2 \end{array} \right\}. \tag{9}$$

The proximal operator for the regularisation functions can be efficiently computed for a large family of functions.

On the other hand, the proximal operator for ι_{PF2},

$$\text{prox}_{\iota_{\text{PF2}}}\left(\{\mathbf{W}_k\}_{k\leq K}\right) = \underset{\{\mathbf{Y}_{\mathbf{B}_k}\}_{k\leq K}}{\arg\min} \left\{ \begin{array}{l} \iota_{\text{PF2}}\left(\{\mathbf{Y}_{\mathbf{B}_k}\}_{k\leq K}\right) \\ + \sum_{k=1}^{K} \frac{\rho_{\mathbf{B}_k}}{2} \| \mathbf{Y}_{\mathbf{B}_k} - \mathbf{W}_k \|_F^2 \end{array} \right\}, \tag{10}$$

where $\{\mathbf{W}_k\}_{k\leq K}$ is an arbitrary collection of matrices, is not trivial to compute. Nevertheless, it can be approximated with

the method of Kiers *et al.* [11]. If we use this method, setting $\mathbf{Y}_{\mathbf{B}_k} = \mathbf{P}_k \boldsymbol{\Delta}_\mathbf{B}$ with $\mathbf{P}_k^\mathsf{T}\mathbf{P}_k = \mathbf{I}$, we obtain Algorithm 1 for the proximal operator. In our experiments, we found that one iteration of this algorithm was sufficient.

Algorithm 1: Approximate projection onto set of collections of matrices with constant cross product

Result: $\{\mathbf{P}_k\}_{k\leq K}, \boldsymbol{\Delta}_\mathbf{B}$
while *convergence criteria are not met* **do**
  for $k \leftarrow 1$ **to** K **do**
    Compute "economy style" SVD:
    $\left(\mathbf{B}_k + \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}_k}}\right)\boldsymbol{\Delta}_\mathbf{B}{}^\mathsf{T} = \mathbf{U}^{(k)}\boldsymbol{\Sigma}^{(k)}\mathbf{V}^{(k)\mathsf{T}}$
    $\mathbf{P}_k \leftarrow \mathbf{U}^{(k)}\mathbf{V}^{(k)\mathsf{T}}$
  end
  $\boldsymbol{\Delta}_\mathbf{B} \leftarrow \frac{1}{\sum_{k=1}^K \rho_{\mathbf{B}_k}} \sum_{k=1}^K \rho_{\mathbf{B}_k}\mathbf{P}_k^\mathsf{T}\left(\mathbf{B}_k + \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}_k}}\right)$
end

Algorithm 2: ADMM for the B mode

Result: $\left\{\mathbf{B}_k, \mathbf{Z}_{\mathbf{B}_k}, \mathbf{Y}_{\mathbf{B}_k} = \mathbf{P}_k\boldsymbol{\Delta}_\mathbf{B}\right\}_{k\leq K}$
while *convergence criteria are not met* **do**
  for $k \leftarrow 1$ **to** K **do**
    $\mathbf{B}_k \leftarrow$ Solve (9)
    $\mathbf{Z}_{\mathbf{B}_k} \leftarrow \text{prox}_{\frac{g_\mathbf{B}}{\rho_{\mathbf{B}_k}}}\left(\mathbf{B}_k + \boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}}\right)$
  end
  $\left\{\mathbf{Y}_{\mathbf{B}_k}\right\}_{k\leq K} \xleftarrow{\text{Alg. 1}} \text{prox}_{\iota_{\text{PF2}}}\left(\left\{\mathbf{B}_k + \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}k}}\right\}_{k\leq K}\right)$
  for $k \leftarrow 1$ **to** K **do**
    $\boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}} \leftarrow \boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}} + \mathbf{B}_k - \mathbf{Z}_{\mathbf{B}_k}$
    $\boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}k}} \leftarrow \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}k}} + \mathbf{B}_k - \mathbf{Y}_{\mathbf{B}_k}$
  end
end

D. ADMM for the A mode

To update the A-mode, we use ADMM to solve the problem

$$\underset{\mathbf{A}}{\arg\min} \quad \sum_{k=1}^K \left\| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \right\|_F^2 + g_\mathbf{A}(\mathbf{A}). \tag{11}$$

This requires us to evaluate both the proximal operator of the data-fidelity term, $f_\mathbf{A}(\mathbf{A}) = \sum_{k=1}^K \| \mathbf{A}\mathbf{D}_k\mathbf{B}_k - \mathbf{X}_k \|_F^2$:

$$\text{prox}_{\frac{f_\mathbf{A}}{\rho_\mathbf{A}}}(\mathbf{M}) = \left(\sum_{k=1}^K \mathbf{X}_k\boldsymbol{\Gamma}_k + \frac{\rho_\mathbf{A}}{2}\mathbf{M}\right)\left(\sum_{k=1}^K \boldsymbol{\Gamma}_k^\mathsf{T}\boldsymbol{\Gamma}_k + \frac{\rho_\mathbf{A}}{2}\mathbf{I}\right)^{-1}, \tag{12}$$

with $\boldsymbol{\Gamma}_k = \mathbf{B}_k\mathbf{D}_k$, and the proximal operator of the regularisation function, $g_\mathbf{A}$. With these operators, we obtain the update steps given in Algorithm 3.

Within the framework of [16], this approach can be considered as hard coupling for all matrices, $\mathbf{X}_k$, through $\mathbf{A}$, and the $\mathbf{B}_k$ updates would correspond to discovering the structure of the coupling for the $\mathbf{B}_k$ matrices.

E. ADMM for the D mode

The D-mode components are updated independently, finding diagonal matrices that solve the problem

$$\underset{\mathbf{D}_k}{\arg\min} \quad \left\| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \right\|_F^2 + g_\mathbf{D}(\mathbf{D}_k), \tag{13}$$

for each k. The proximal operator for the data-fidelity term, $f_{\mathbf{D}_k}(\mathbf{D}_k) = \left\| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\mathsf{T} - \mathbf{X}_k \right\|_F^2$ is the minimiser of a quadratic function. The minimiser is formulated using the vector containing the diagonal entries of $\mathbf{D}_k$:

$$\text{prox}_{\frac{f_{\mathbf{D}_k}}{\rho_{\mathbf{D}_k}}}(\mathbf{v}) = \left(\mathbf{A}^\mathsf{T}\mathbf{A} * \mathbf{B}_k^\mathsf{T}\mathbf{B}_k + \frac{\rho_{\mathbf{D}_k}}{2}\mathbf{I}\right)^{-1}\left(\boldsymbol{\xi} + \frac{\rho_{\mathbf{D}_k}}{2}\mathbf{v}\right), \tag{14}$$

where $*$ is the Hadamard product and $\boldsymbol{\xi} = \text{Diag}\left(\mathbf{A}^\mathsf{T}\mathbf{X}_k\mathbf{B}_k\right)$ is the vector containing the diagonal entries of $\mathbf{A}^\mathsf{T}\mathbf{X}_k\mathbf{B}_k$. This results in the update steps given in Algorithm 4.

Algorithm 3: ADMM for the A mode

Result: $\mathbf{A}, \mathbf{Z}_\mathbf{A}, \boldsymbol{\mu}_\mathbf{A}$
while *convergence criteria are not met* **do**
  $\mathbf{A} \xleftarrow{(12)} \text{prox}_{\frac{f_\mathbf{A}}{\rho_\mathbf{A}}}(\mathbf{Z}_\mathbf{A} - \boldsymbol{\mu}_\mathbf{A})$
  $\mathbf{Z}_\mathbf{A} \leftarrow \text{prox}_{\frac{g_\mathbf{A}}{\rho_\mathbf{A}}}(\mathbf{A} + \boldsymbol{\mu}_\mathbf{A})$
  $\boldsymbol{\mu}_\mathbf{A} \leftarrow \boldsymbol{\mu}_\mathbf{A} + \mathbf{A} - \mathbf{Z}_\mathbf{A}$
end

Algorithm 4: ADMM for the D mode

Result: $\mathbf{D}_k, \mathbf{Z}_{\mathbf{D}_k}, \boldsymbol{\mu}_{\mathbf{D}_k}$
while *convergence criteria are not met* **do**
  for $k \leftarrow 1$ **to** K **do**
    $\mathbf{D}_k \xleftarrow{(14)} \text{prox}_{\frac{f_{\mathbf{D}_k}}{\rho_{\mathbf{D}_k}}}(\mathbf{Z}_{\mathbf{D}_k} - \boldsymbol{\mu}_{\mathbf{D}_k})$
    $\mathbf{Z}_{\mathbf{D}_k} \leftarrow \text{prox}_{\frac{g_\mathbf{D}}{\rho_{\mathbf{D}_k}}}(\mathbf{D}_k + \boldsymbol{\mu}_{\mathbf{D}_k})$
    $\boldsymbol{\mu}_{\mathbf{D}_k} \leftarrow \boldsymbol{\mu}_{\mathbf{D}_k} + \mathbf{D}_k - \mathbf{Z}_{\mathbf{D}_k}$
  end
end

F. PARAFAC2 AO-ADMM

By combining the three update algorithms above, we obtain Algorithm 5 to fit regularised PARAFAC2 models to data. To measure convergence of the inner loops, we adapted the stopping criteria in [18] with a maximum of five iterations. For the outer loop, we followed [16] and stopped when both the loss and all relative primal feasibility gaps (e.g. $\| \mathbf{A} - \mathbf{Z}_\mathbf{A} \|_F / \| \mathbf{A} \|_F$) were either below an absolute tolerance or their relative change between two subsequent iterations was below a relative tolerance (with maximum number of iterations set to 1000).

Algorithm 5: AO-ADMM for PARAFAC2

Result: $\mathbf{A}, \{\mathbf{B}_k, \mathbf{D}_k\}_{k\leq K}$
Initialise
$\mathbf{A}, \mathbf{Z}_\mathbf{A}, \boldsymbol{\mu}_\mathbf{A}, \mathbf{B}_k, \mathbf{Z}_{\mathbf{B}_k}, \boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}}, \boldsymbol{\Delta}_\mathbf{B}, \mathbf{P}_k, \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}k}}, \mathbf{D}_k, \mathbf{Z}_{\mathbf{D}_k}$, and $\boldsymbol{\mu}_{\mathbf{D}_k}$
while *convergence criteria are not met* **do**
  Update $\left\{\mathbf{B}_k, \mathbf{Z}_{\mathbf{B}_k}, \mathbf{P}_k, \boldsymbol{\mu}_{\mathbf{Z}_{\mathbf{B}_k}}, \boldsymbol{\mu}_{\boldsymbol{\Delta}_{\mathbf{B}k}}\right\}_{k\leq K}$ and $\boldsymbol{\Delta}_\mathbf{B}$ using Algorithm 2
  Update $\mathbf{A}, \mathbf{Z}_\mathbf{A}$ and $\boldsymbol{\mu}_\mathbf{A}$ using Algorithm 3
  Update $\left\{\mathbf{D}_k, \mathbf{Z}_{\mathbf{D}_k}, \boldsymbol{\mu}_{\mathbf{D}_k}\right\}_{k\leq K}$ using Algorithm 4
end

G. Selecting ρ

For efficient ADMM updates, we need suitable ρ-parameters. In this work, we selected ρ adaptively [15]:

$$\rho_{\mathbf{B}_k} = \frac{\| \mathbf{A}\mathbf{D}_k \|_F^2}{R}, \qquad \rho_{\mathbf{A}} = \sum_{k=1}^{k} \frac{\| \mathbf{B}_k\mathbf{D}_k \|_F^2}{R}, \qquad (15)$$
$$\rho_{\mathbf{D}_k} = \frac{1}{R}\mathrm{Tr}\left(\mathbf{A}^\intercal\mathbf{A} * \mathbf{B}_k^\intercal\mathbf{B}_k\right).$$

IV. Experiments

For all methods, we used our Python implementations linked in the supplementary[1] (a MATLAB implementation is also available[2]). As a baseline, we compared the AO-ADMM algorithm with the traditional unregularised ALS algorithm [11], and for the experiments with non-negativity constraints, we also compared with the flexible coupling PARAFAC2 using hierarchical non-negative least squares algorithm (HALS), implemented following the MATLAB implementation [12]. We used the same HALS algorithm [20], default parameter values and initialisation scheme. For the proximal operator of the total variation (TV) seminorm, we used the C implementation [21] of the improved direct TV denoising algorithm from [22]. We set the stopping tolerances equal to 10^{-5} for the inner loops (ADMM updates) and 10^{-10} for the outer loop.

To measure convergence we used the relative SSE:

$$\text{Rel. SSE} = \frac{1}{\| \mathcal{X} \|_F^2} \sum_{k=1}^{K} \left\| \mathbf{A}\mathbf{D}_k\mathbf{B}_k^\intercal - \mathbf{X}_k \right\|_F^2. \qquad (16)$$

Also, we measured if different methods recovered the true components with the factor match score (FMS), given by:

$$\text{FMS} = \frac{1}{R} \sum_{r=1}^{R} \mathbf{a}_r^\intercal\hat{\mathbf{a}}_r\mathbf{b}_r^\intercal\hat{\mathbf{b}}_r\mathbf{c}_r^\intercal\hat{\mathbf{c}}_r, \qquad (17)$$

where $\hat{\mathbf{a}}_r, \hat{\mathbf{b}}_r, \hat{\mathbf{c}}_r$ represent the estimated component, taking into account the permutation ambiguity. The $\mathbf{b}_r$ and $\mathbf{c}_r$-vectors contain the concatenations of the r-th column of all $\mathbf{B}_k$ matrices and the r-th diagonal entry of all $\mathbf{D}_k$ matrices, respectively. All component vectors are normalised before computing the FMS.

To evaluate the AO-ADMM approach, we use a simulation setup inspired by [12]. The elements of $\mathbf{A}$ and $\mathbf{D}_k$ factor matrices were respectively drawn from a truncated normal distribution and a uniform distribution between 0.1 and 1.1 (to avoid near zero elements in the $\mathbf{D}_k$ matrices, which can hinder recovery of the $\mathbf{B}_k$ matrices [11]). The $\mathbf{B}_k$ factor matrices were obtained by first generating a "blueprint matrix", $\hat{\mathbf{B}}$ tailored to the constraint we wished to impose. The rows of $\hat{\mathbf{B}}$ were subsequently cyclically shifted to obtain $\mathbf{B}_k$ matrices, setting $[\mathbf{B}_k]_{jr} = \hat{\mathbf{B}}_{j_k r}$, with $j_k = ((j+k) \mod J)$.

For each experiment, we created 50 random datasets. We constructed tensor slices, $\mathbf{X}_k$, based on (2) using known factor matrices. We let $\mathcal{X}$ be the tensor with frontal slices given by our data matrices, and added random noise according to

$$\mathcal{X}_{\text{noisy}} = \mathcal{X} + \eta \| \mathcal{X} \|_F \frac{\mathcal{E}}{\| \mathcal{E} \|_F}, \qquad (18)$$

where η is the noise level and $\mathcal{E}_{ijk} \sim \mathcal{N}(0,1)$.

For each dataset, we fit models with five random initialisations, and kept the components that achieved lowest final cost value. Non-negativity constraints were always imposed on the $\mathbf{D}_k$-matrices, to resolve the sign-indeterminacy of the PARAFAC2 model [6], [23].

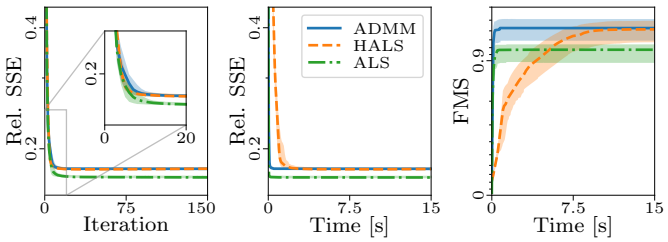

Fig. 2: Diagnostic plots for datasets with noise level $\eta = 0.5$. The lines show the median values.

A. Non-negativity constraints

To assess the performance of the AO-ADMM based algorithm for fitting a PARAFAC2 model with non-negativity constraints, we compared speed and accuracy with the baselines. We generated $\hat{\mathbf{B}}$ matrices with three components and elements drawn from a truncated normal distribution (setup 1). Noise levels were set to 0.33 and 0.5, and the tensor had dimensions $20 \times 30 \times 20$. For AO-ADMM and HALS, non-negativity was imposed on all modes, whereas for ALS, non-negativity was only imposed on the $\mathbf{A}$ and $\{\mathbf{D}_k\}_{k \leq K}$ matrices. Diagnostic plots for $\eta = 0.5$ are shown in Fig. 2 demonstrating that both non-negative PARAFAC2 algorithms outperform ALS with respect to FMS. Moreover, the AO-ADMM algorithm is as fast as the traditional ALS algorithm and faster than the flexible coupling approach. We observed the same behaviour for $\eta = 0.33$ (see supplementary material).

B. Structure imposing regularisation

The AO-ADMM algorithm also allows for structure imposing regularisation such as graph Laplacian regularisation ($\gamma_L \mathrm{Tr}(\mathbf{B}_k^\intercal\mathbf{L}\mathbf{B}_k)$) [24] and total variation regularisation ($\gamma_{TV} \|\mathbf{B}_k\|_{\text{TV}}$). To assess the effectiveness of graph Laplacian regularisation, we set the three components of $\hat{\mathbf{B}}$ equal to emission spectra from a fluorescence spectroscopy dataset [3] (setup 2). These spectra are smooth, i.e. neighbouring wavelengths have similar values, which makes graph Laplacian regularisation sensible. To impose smoothness, we set the penalty function to $g_{\mathbf{B}}(\mathbf{B}_k) = \gamma_L \sum_{jr} ([\mathbf{B}_k]_{jr} - [\mathbf{B}_k]_{j+1r})^2$. For assessing total variation regularisation, we used piece-wise constant functions with 6 jumps whose derivatives summed to zero as the three components of $\hat{\mathbf{B}}$ (setup 3). The size of the smooth and piece-wise constant data tensors were $20 \times 201 \times 40$ and $20 \times 200 \times 40$ respectively. For both types of experiments, we tested with two different noise levels:

[1] https://github.com/MarieRoald/PARAFAC2-AOADMM-EUSIPCO21

[2] https://github.com/AOADMM-DataFusionFramework/AOADMM-PARAFAC2

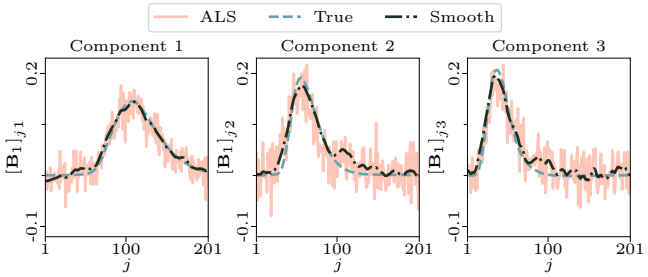

Fig. 3: True and estimated columns of $\mathbf{B}_1$ for one dataset with $\eta = 0.5$. Smoothness regularised components were fitted with $\gamma_r = 0.01$ and $\gamma_L = 1000$. $\mathbf{X}_1$ had a signal to noise ratio of 0.1 dB.

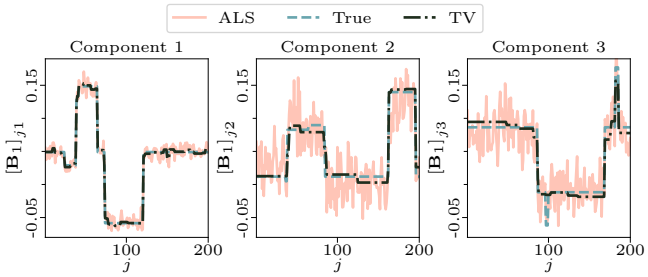

Fig. 4: True and estimated columns of $\mathbf{B}_1$ for one dataset with $\eta = 0.5$. TV regularised components were fitted with $\gamma_r = 10$ and $\gamma_{TV} = 10$. $\mathbf{X}_1$ had a signal to noise ratio of 5.9 dB.

$\eta \in \{0.33, 0.5\}$ and we imposed ridge regularisation on $\mathbf{A}$ and $\mathbf{D}_k$ ($\gamma_r(\| \mathbf{A} \|_F + \sum_k \| \mathbf{D}_k \|_F)$). The regularisation parameters were found through a grid search (details in supplementary).

The structure imposing regularisation helped recovery for both setup 2 and 3. For most parameter combinations, we observed an increase in FMS compared to unregularised models. The performance degraded only with a very high degree of regularisation. Table I shows the results for the parameters that obtained the highest mean FMS. In Fig. 3 we see that the graph Laplacian regularised models led to smooth components and Fig. 4 shows that the TV regularisation produced piecewise constant components (see supplementary for animated plots). For both setups, the ALS algorithm yielded noisy components.

V. Conclusion

In this work, we proposed an AO-ADMM-based algorithmic framework for fitting PARAFAC2 models with regularisation. Using the proposed approach, we can fit PARAFAC2 models with any proximable regularisation penalty on all factor matrices. Our experiments demonstrate that the AO-ADMM framework is faster than the flexible coupling approach for non-negative PARAFAC2, and can successfully apply structure imposing regularisation, such as TV and graph Laplacian regularisation, on the evolving mode of a PARAFAC2 model.

TABLE I: FMS results from experiments with structure imposing regularisation.

	Setup 2		Setup 3	
Method	$\eta = 0.33$	$\eta = 0.5$	$\eta = 0.33$	$\eta = 0.5$
AO-ADMM	0.99 ± 0.01	0.98 ± 0.01	0.98 ± 0.02	0.96 ± 0.05
ALS	0.96 ± 0.01	0.92 ± 0.01	0.92 ± 0.06	0.86 ± 0.08

References

[1] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA working papers in phonetics*, vol. 16, pp. 1–84, 1970.

[2] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[3] R. Bro, "PARAFAC. tutorial and applications," *Chemom. and Intel. Lab. Systems*, vol. 38, no. 2, pp. 149–172, 1997.

[4] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred, "Parallel factor analysis as an exploratory tool for wavelet transformed event-related EEG," *NeuroImage*, vol. 29, no. 3, pp. 938–947, 2006.

[5] E. Acar, C. A. Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

[6] R. A. Harshman, "PARAFAC2: Mathematical and technical notes," *UCLA working papers in phonetics*, vol. 22, pp. 30–44, 1972.

[7] R. Bro, C. A. Andersson, and H. A. L. Kiers, "PARAFAC2 - Part II. Modeling chromatographic data with retention time shifts," *J. Chemom.*, vol. 13, no. 3-4, pp. 295–309, 1999.

[8] A. Afshar, I. Perros, E. E. Papalexakis, E. Searles, J. Ho, and J. Sun, "COPA: Constrained PARAFAC2 for Sparse & Large Datasets," in *ACM Int. Conf. on Inf. and Knowl. Management*, 2018, pp. 793–802.

[9] P. A. Chew, B. W. Bader, T. G. Kolda, and A. Abdelali, "Cross-language information retrieval using PARAFAC2," in *Proc. 13th ACM SIGKDD Int. Conf Knowl Discov and Data Mining*, 2007, pp. 143–152.

[10] M. Roald, S. Bhinge, C. Jia, V. Calhoun, T. Adali, and E. Acar, "Tracing network evolution using the PARAFAC2 model," in *Proc. Int. Conf. on Acoust., Speech, and Signal Process.*, 2020.

[11] H. A. L. Kiers, J. M. F. Ten Berge, and R. Bro, "PARAFAC2 - Part I. A direct fitting algorithm for the PARAFAC2 model," *J. Chemom.*, vol. 13, no. 3-4, pp. 275–294, 1999.

[12] J. E. Cohen and R. Bro, "Nonnegative PARAFAC2: A flexible coupling approach," in *LVA/ICA'18*, 2018, pp. 89–98.

[13] N. E. Helwig, "Estimating latent trends in multivariate longitudinal data via Parafac2 with functional and structural constraints," *Biom. J.*, vol. 59, no. 4, pp. 783–803, 2017.

[14] K. Yin, A. Afshar, J. C. Ho, W. K. Cheung, C. Zhang, and J. Sun, "LogPar: Logistic PARAFAC2 factorization for temporal binary data with missing values," in *KDD*, 2020, pp. 1625–1635.

[15] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5052–5065, 2016.

[16] C. Schenker, J. E. Cohen, and E. Acar, "A flexible optimization framework for regularized matrix-tensor factorizations with linear couplings," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 506–521, 2021.

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

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[19] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Mach. Learn.*, vol. 1, no. 3, pp. 127–239, 2014.

[20] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization," *Neural Comput.*, vol. 24, no. 4, pp. 1085–1105, 2012.

[21] L. Condat. "Software." [Accessed: 2020-10-19]. (2019), [Online]. Available: https://lcondat.github.io/software.html.

[22] L. Condat, "A direct algorithm for 1-D total variation denoising," *IEEE Signal Process. Letters*, vol. 20, no. 11, pp. 1054–1057, 2013.

[23] N. E. Helwig, "The special sign indeterminacy of the direct-fitting Parafac2 model: Some implications, cautions, and recommendations for simultaneous component analysis," *Psychometrika*, vol. 78, no. 4, pp. 725–739, 2013.

[24] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines*, Springer, 2003, pp. 144–158.